Fetching Interactive Dog Occupier (FIDO)

A Major Qualifying Project

Submitted to the Faculty of

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree in Bachelor of Science

in

Mechanical Engineering

By

Hannah Brown

Michael Goldman

Harry Lanphear

Domenic Mucci

Rebecca Renner

Greg Shannon

Date: 4/30/15

Professor John Sullivan, D. E.

Abstract

Dogs require exercise and play to stimulate them both mentally and physically, frequently through the game of “fetch”. Elderly or physically disabled dog owners, however, may lack the strength or endurance to play with their pets. Our project developed a prototype for a marketable, autonomous fetching entertainment center for dogs. To accomplish this, we studied current products on the market and innovated upon their limitations. We developed a unique launching system that can be safely operated indoors or outdoors. The Fetching Interactive Dog Occupier, or F.I.D.O., can be used with or without human intervention. It launches tennis balls at varying distances, and even offers positive reinforcement (both user-recorded audio playback and treat dispensing) to train dogs in returning the ball back to the system and placing it in the hopper.

Executive Summary

A human master and their dog can spend time practicing to fetch. The human will typically throw an object and the dog will instinctively go and get it. If the interaction is to continue, the human must teach the dog to return the object or go get it from the dog. With training some dogs become adept at retrieving the object and returning it to its master. This is part of the dog's predatory instincts of chasing. This training can take the form of positive reinforcement, such as verbal praise ("good dog!"), treats, or another toy that the dog likes.

There are products on the market today that launch a ball. Some of these products are designed specifically within this market. There are products in which the dog can utilize the device without the assistance of a human but the positive reinforcement, besides the ball launching into the same place it just went, is not offered.

The Fetching Interactive Dog Occupier (FIDO) is designed to provide a launching apparatus that stimulates the dog's retrieval instincts as well as vary the retrieval point on every throw, therefore increasing the novelty of the experience for the dog. With rotation of the launching platform, the ball can launch at up to 45 degrees of inclusive angle. The FIDO unit also offers the ability to fit the size of the environment in which the dog fetches balls. With different launching force and angle of flight the ball can travel short or long distances. In the aspect of training, the FIDO unit can either provide recorded voice commands or food treats at different interval settings controlled by the master. Another important aspect of the design involves safety for the customer and its dog. With sensors to ensure that the firing range is clear, the ball will launch. All of this is possible due to the design considerations that this project team engineered as described below.

When determining the initial product requirements, the team realized that the user customization of the design was imperative for it to be different than previous options on the market. Advanced development centered on incorporating these features but the strength of the system to launch had to be the focal point. The first design considerations for the launching mechanism were very unique and offered a different approach. A spiral hook would grab a ball and pull it back through its tines, ultimately releasing it through a hole at the end of the spiral. Due to manufacturing and stability concerns this was shelved for a more conventional pinball style puller on a spring that was pulled back by a gripper claw. This design proved to be adequate, yet the translating lead screw concept which has a pinion gear attached to the lead screw and a DC motor needs a gear ratio that increases speed from the motor to reduce lag in the system timing.

Varying the launching tube through a 15 to 45 degree angle was determined to be a user setup option that would allow flexibility during a fetching session. This was accomplished with a lead screw attached to a DC motor. Other setup options include the frequency of dispensing a treat, whether or not to provide voice prompts and if the rotating randomization through the 45 degree inclusive angle or a subset should occur. These controls are done through software written for a Raspberry Pi controller. The Raspberry Pi controls the user interfaces and the five motors power, limiting protective switching and safe firing through a distance sensor.

The outside shell and rotating platform were made by a rapid prototyping (3D printing) process. Although this process increases individual part costs for the prototype, it offers the ability to create complex geometry and have the parts manufactured in an expedited process. The shell needs to be water resistant to protect the electronics and mechanisms inside the unit from

occasional environmental conditions. FIDO is battery operated with a charging station and removable plug.

The complex design of the product lead to multiple iterations of the design before the manufacturing process began. Once the initial “proof of concept” prototype was in the process of machining and assembly, more iterations were necessary in the treat dispense, launching gripper and power delivery. The code created went through multiple iterations as well. Ultimately the concept was proved through testing.

However, there are changes that could be made to increase the functionality and overall quality of the product. The team recognizes that size, weight and speed of the system all have to be optimized on future iterations of the design. At present, the overall size of the product limits the use of the product to medium and larger dogs, and the size and weight severely limit a user’s ability to move it for use in different locations. Additionally, the current speed of the system will limit the enjoyment of the product, as the dog will have to wait a minute plus for the ball to be launched after the dog has returned it to the hopper. These three characteristics of the system are possible to engineer for optimum performance and we recommend that future work focus on these areas.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The purpose of this project was to create a prototype for a marketable, autonomous, fetch device for dogs. The team analyzed designs currently on the market to understand what is being offered and what is not. The team then used this information to create a unique product. The product is able to be used by the dog without supervision. It can be used indoors or outdoors, and can launch a ball at varying distances and angles. The product also can strengthen a dog's ability to fetch, offering positive reinforcement when a ball is returned. The team designed a unique launching apparatus that incorporated safety for the dog and others by monitoring the environment in the launching path. The device can shoot regular tennis balls and other balls similar in size. The device is intended to be transportable and have a certain level of water ingress protection.

2 Background

2.1 Dog Training

In order to best design the ball-launching product, the team sought to understand the fundamentals behind some of the behavior involved with playing fetch. Why do dogs like playing fetch? What are some basic principles of dog training? How do dogs learn to play fetch?

Certain types of instinctual, "pre-programmed" behaviors are observed in all members of the Canidae family (i.e. dogs, wolves, coyotes, etc.), and one of these behaviors is the prey instinct. If there is a moving object that draws the attention of an animal, its instinct is to chase, catch, and kill it, whether it is a prey animal or merely a tennis ball or a toy (Jacobs, 2015). Due to the wide variety of different breeds of dog that have been bred for different purposes, the intensity of the prey instinct varies. For example, retrievers have been bred to chase, grab prey, and then bring it back without damaging it, while herding dogs have been bred to focus on the chasing aspect without the grab and bite stage so they don't injure the stock they are herding (Coppinger, 2001). Some of these prey behaviors have been bred out of the dog breeds most commonly selected as household pets, but in many dogs, the basic chase and retrieve instinct remains, and these are the dogs that are most likely to play fetch. (Coppinger, 2001). If a dog has a genetic predisposition to chasing and grabbing something, it is likely that this particular dog will enjoy playing fetch, even without outside reinforcement (such as treats or verbal praise).

In the past, one major school of thought for dog training was the dominance theory, which is based on the idea that dogs are pack animals, and can be trained or controlled by a single "alpha leader" (Greenebaum, 2010). Training based on dominance theory states that the dog's guardian should serve as the alpha leader, and often involves the use of strict physical corrections to eliminate undesired behaviors, such as choke or prong collars, physical

intimidation, and "alpha rolls" – physically flipping the dog over and holding them down (Greenebaum, 2010). However, dominance theory-based training is now generally considered to be outdated and cruel, and it is not commonly used anymore.

Currently, one of the most prominent training theories is behavior modification based on rewards, rather than dominance. This theory is based on operant conditioning, which is composed of four types: positive reinforcement, positive punishment, negative reinforcement, and negative punishment (Dugatkin, 2014). Positive reinforcement is adding in a good thing, such as a treat, verbal praise, or affection, to reward desired behavior. Positive punishment involves adding a bad thing, such as physical punishment, to discourage undesired behavior. Negative reinforcement is taking away a bad thing, such as loosening a tight leash hold, to reinforce desired behavior, and negative punishment is taking away good things, such as removing a ball or toy, to discourage undesired behavior (Greenebaum, 2010).

A combination of positive reinforcement and negative punishment are the basis of reward-based training – it helps create the desired behavior in the dog, while making sure the dog remains safe and happy due to the lack of physical punishment. "The message here is that training reinforces a relationship. It is not a one-way process; both the person and the dog are being trained in a way that promotes a respectful partnership. Rather than imposing human will on the dog by force, they work with dogs so they can make decisions based on free will" (Greenebaum, 2010).

In the context of this project (e.g. training a dog to fetch a ball), training will depend on the strengths of the individual dog. The trainer will need to determine what the positive reinforcement (treats, praise, etc.) will be, what the negative punishment will be (taking away a

toy, denying affection, etc.) and how much the dog already knows. If one is starting from the very beginning of teaching a dog to fetch, the basic steps are as follows:

1. Start by tossing the ball or toy a short distance away; once the dog retrieves the ball, exchange it for a treat
2. Once the dog has attained this task, add a cue word such as “drop”.
3. Practice with the dog bringing the ball back until they drop the ball based on the cue word alone, without the treat
4. Gradually increase the distance of the ball thrown.

2.2 Dog Demographics

To be sure that our device could eventually be successful on the market, we researched the demographics of dogs and their owners. In 2012, a study done by a marketing agency reported that 72.9 million households in the United States have at least one pet. Of these households 46.3 million own dogs. Due to the fact that many people have more than one dog, there are reported to be 78.2 million dogs owned at that time. 52% of these dogs weigh less than 25 pounds, 24% are between 26-50 pounds, 24% are between 51-75 pounds, and 11% are over 75 pounds. (Mentel, 2012)

The study shows that people most likely to own larger dogs are between the ages of 25-54 years old, probably because this age group is more active and can keep up with the commitments of owning a large dog. However, the 25-54 year old age group is also the group that works the most hours in day and has the busiest lifestyles. The Bureau of Labor Statistics published a Time Use Survey that showed that Americans between the ages of 25-54 work on average 1-1.5 hours a day more than any other age group (BLS, 2014). Large dogs generally need the most exercise but the studies show that their owners are busy outside of the home more

hours of the day than any other age group, this leaves a place in the market for devices that will entertain and exercise the dog while the owner is away.

We also looked at dogs that are kept in apartments. A survey done in 2014 by Apartments.com showed that 72% of apartment renters own pets. Of these animals, 39% are small dogs and 34% are medium/large dogs. The survey also showed that more and more apartment buildings are starting to allow animals, which in turn is increasing the number of pet owning renters every year (apartments.com, 2014). With the large population of dogs that are now in apartment building which have limited exercise space, this is another area group of pet owners that our device could be marketed to.

The final demographic group that we considered was the elderly. 32% of seniors (65+) own dogs and while more than half the time these are small dogs, they still need to be exercised (Mentel, 2012). Many of these dog owners probably choose small dogs due to their lower exercise requirements because they are not physically able to keep up with the demands of a larger dog even if they wanted one. Having a device that could throw a ball a good distance for the owner could give more freedom for more elderly to own dogs as companions and to have whatever type of dog they wanted.

2.3 Market Competition

To understand the current competition, the team set out to determine what products are on the market today. These products would be ones that FIDO would compete with, as well as other ball launching apparatuses. The team determined that it was necessary to open up the search to more human interaction as well as autonomous devices to ensure that FIDO offered a unique place in the market.

iFetch is an interactive, on demand ball launcher. It throws a ball three different distances when the ball is placed into the top opening, and it is intended to work only with the 40mm balls provided. iFetch comes with three balls and an AC adapter for both 110V and 220V, and retails for $99.95. It has a simple elegant enclosed design. The manufacturer of the product does not claim any environmental information online. The product seems intended for indoor use or outdoor use during nice weather. The small balls are not well suited for larger dogs. There isn't much safety in the launching process. A dog or small child could be in front of the launch opening while the ball is in flight. With not having witnessed the product it seems the product must not launch with much force. (iFetch, 2014)

Figure 1: iFetch

The GoDogGo Fetch Machine is a dog toy that launches balls three different distances between 12 and 42 feet. Ball type and size can be changed and this will affect the distance the ball will travel. The manufacturer states that tennis, racquet, Kong Air Squeaker, and Chuckit Ultra balls can all be launched from this mechanism. GoDogGo can be set to launch a ball at 2 different time intervals when in automatic mode (7 & 15 seconds). The launcher also has a remote control and can be put into manual mode in which the time between launching is

dependent on the user pressing the button. The GoDogGo was designed with an automatic sensor/safety switch with an auto stop feature. GoDogGo is 11 lbs. and is 15X13X17.5 inches in size. The new G3 GoDogGo launching mechanism is produced by the makers of Bosch, Makita and Hitachi drill gears & motor systems. The manufacturer states that the product can be used inside and outside. The GoDogGo G3 machine retails for $139.99 (GoDogGo, 2014).

Figure 2: GoDogGo Fetch Machine

Lucky Launcher, Zinger Winger and Thunder 500 products are launching products for training dogs for hunting. They use different launch mechanisms such as ballistics and recoil springs to launch fake ducks. These all require the dog master to be involved in the process. This training is more about relationship building than dog enjoyment. (Lucky Launcher, 2014).

Figure 3: Lucky Launcher II

Other ball launching systems are tennis ball launcher and baseball launchers (Juggs machine). These products use various launching techniques such as spinning wheels or air pressure to release balls. These are intended for human use but have been used to play with a dog. These machines need require human interaction.

2.4 Launching Mechanisms

The key to a successful autonomous ball launcher is the launching mechanism itself. Without it, the device fails to complete its most basic design requirements. However, there exist a number of different methods by which a projectile may be launched, and some of these work better for the requirements of this project than others.

In their most basic form, all launching mechanisms are dependent upon energy transfer. This can be potential energy stored in a tensed spring or a pressurized sealed container, or kinetic energy in a spinning motor. The initial energy is then transferred to the projectile, which is then

launched along a certain trajectory (dependent on outside influences, such as an exit tube from the machine).

Launching mechanisms can be separated into six different groups, indicated below. The distinctions between each lie in the specific method by which they launch a projectile.

1. **Compression spring-loaded devices:** A spring is compressed, holding energy proportional to the distance compressed. When the spring is released, this energy is transferred to the surroundings. By fixing one end of the spring, and placing a ball at the other end, the energy can be completely transferred to the ball. The ball will then fly or roll (depending on the original orientation of the ball and spring) away from the spring (Nave, 1998).
2. **Tension spring-corded devices:** Similar to a bow or a slingshot, energy is stored in a cord of rope, elastic, or flexible wood. When the tension is released, the snap back into the initial position transfers the energy to the ball. This is the principle behind catapults, ballistae, and trebuchets. Despite a difference in physical shape, these systems can be modeled as tension springs, rather than compression springs from the first group (Nave, 1998).
3. **Pneumatic devices:** Air pressure is built up in a sealed container through the use of a compressor. When the projectile is in position, the air is released at that position, and the force of the pressuring gas launches the projectile. Pneumatic-based devices can be designed to use the projectile as the airtight seal on the system. However, systems with a separate chamber to pressurize and store the gas before releasing it to the projectile are more common. T-shirt cannons, for instance, function because the chamber is pressurized through one port by a separate compressor, and then depressurized through a second port just behind the t-shirt in the cannon tube (Wilson, 2005).
4. **Lever-based devices:** A lever swings; this launches the projectile upon contact. A common example of this is from America's pastime: the baseball bat hitting the baseball. The food fights begun by slamming fists down on spoons loaded with cafeteria cuisine are also an example of class 1 levers and loads. In addition, this

category includes throwing arms spun by motors, such as those used in clay pigeon launchers (Williams, 2000).

5. **Rotary-based devices:** This group is most commonly seen in tennis ball launchers or batting practice devices for baseball. One or two wheels (if two, they each have their own motor), running at high speeds, accelerate the ball as it passes tangent to it/them ("Bata 2", 2015).
6. **Linearly-actuated devices:** As the name suggests, these devices use a linear actuator motor, which creates linear, not circular, motion. By using magnets to move a plate up and down a linear rod, a great deal of force and acceleration can be provided with great control ("The Electric Linear Actuator", 2015).

2.5 Motors

An electric motor converts electrical energy into mechanical energy. (Hughes, 2013) Most electric motors operate by interactions between the magnetic field and the winding currents. This generates a force within the motor. Electric motors can be powered by either direct current (D.C.) or alternating current (A.C.) (Hughes, 2013). Electric motors produce linear or rotary force or torque, through rotational movement of the drive shaft.

A D.C. motor operates over a very wide power range from several megawatts at the top end down to only a few watts, all D.C. motors have the same basic structure, as shown in Figure 4. The motor has two separate electrical circuits. The smaller pair of terminals connects to the field windings, which surround each pole and are normally in series: these windings provide the motor magnetic field to set up the flux in the air-gap under the poles. In the steady state all the input power to the field windings is dissipated as heat – none of it is converted to mechanical output power. The main terminals deliver the 'torque-producing' or 'work' current to the brushes which make sliding contact to the armature winding on the rotor (Hughes, 2013). Figure 4 depicts the cross sectional image of a D.C. motor.

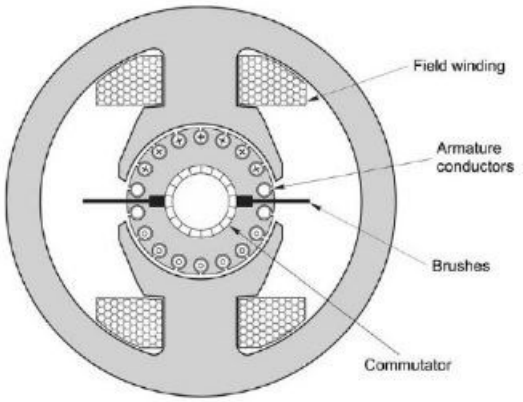

Figure 4: Conventional Brushless D.C. Motor

Stepper motors are electric motors that provide precise rotations. In a stepper motor an internal rotor containing a magnetically soft rotor with salient poles is controlled by a set of external magnets that are switched electronically. A stepper motor may not rotate continuously, instead it "steps" through quick start and stops controlled by the magnets energizing and de-energizing in sequential timing. The motor may turn forward, backwards. It may change direction, speed up, slow down or stop depending of the sequence (Hughes, 2013).

A servomotor is an electric motor often sold as a complete module. The module includes a control or speed control feedback system. The speed vs. torque curve of the servomotor is an important characteristic of the motors application and is a high ratio. Other important characteristics are the winding inductance and rotor inertia. These characteristics factor into the overall performance of the motor (Hughes, 2013). FIDO's need for the servomotor allows for the use of a conventional large powerful, but slow responding servo loop due to the dynamic response requirements not being a critical characteristic.

2.6 System Controls

The motors and switches in FIDO require some means of control. This allows for greater variation in the usage and setup of the input and output mechanisms, as well as user-defined changes to the normal operation. An embedded system is thus the obvious choice.

There are different types of embedded systems. A Programmable Logic Controller, or PLC, is commonly used in industry due to its ease of operation and repeatability. However, the ladder logic required to program it is complicated to learn, and the benefit of withstanding a high-stress shop floor environment is not a priority in this project (Parr, 1998). Microcomputers or microcontrollers can also be used in place of PLCs. A microcontroller is the single component of a computer, whereas the microcomputer is a fully functional computer (Orsini, 2014). As Orsini notes, the Raspberry Pi (a microcomputer) is better suited for software purposes than the Arduino (a microcontroller) (2014). The other notable difference is the programming language used. The Raspberry Pi is Python-based, while the Arduino is C-based.

2.7 Rapid Prototyping Rapid Prototyping

2.7.1 Fusion Deposition Modeling

Fusion Deposition Modeling (FDM) is the process of printing a 3D extrusion of plastic, layer upon layer from a computer-aided design (CAD) data. The CAD data defines a tool path for the printer to deposit molten thermoplastic and build the part from the base of the structure up. This makes a complex part easy to produce for prototyping needs.

FDM builds three-dimensional parts by melting and advancing a fine ribbon of plastic through a computer controlled extrusion head. In certain complicated structures, support structure is required. This support structure dissolves or breaks off the part easily.

The process of FDM starts with importing the CAD data into a build preparation program such as Catalyst EX. When the file is imported options such as orientation, bed location are selected. The software calculates sections and slices the part design into many layers, ranging from 0.005 inches to 0.013 inches in height. The software then generates "tool paths" or building instructions, which will drive the extrusion head (Himenez, 2015). The build begins with the two materials entering the extrusion head. Heat is applied to soften the plastics and then extruded into the thin layers.

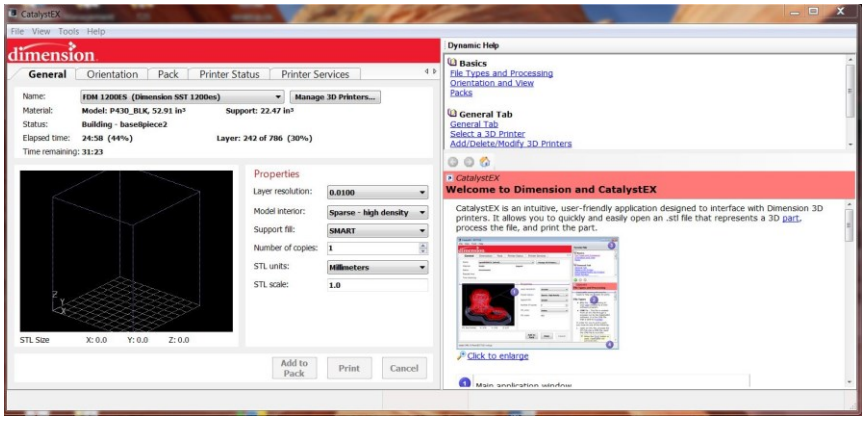

Figure 5 Catalyst EX Software GUI

FDM 3D printers have build volume capacities ranging from approximately 288 cubic inches to 31K cubic inches. The material cartridges deliver the plastic to the extrusion head. The heat chamber contains the head which moves in the x and y axes while the material is liquefied and deposited. The Z platform moves down to create the third dimension (Marshall, 2015).

2.7.2 Stereolithography

Stereolithography (SLA) is the process to create solid plastic 3D parts from CAD models. The process deposits thin layer upon thin layer of material, typically a liquid resin which is then hardened by a scanning laser beam. (Atkins & Escudier, 2013). The SLA machine has a tank filled with gallons of liquid photopolymer, a clear liquid plastic. A perforated platform in the

tank can move up and down. An ultraviolet laser driven by a computer is aimed at the tank and the photopolymer hardens due to its sensitivity to the light.

Figure 6: Stereolithography Machine Mars

2.7.3 Process

The process begins with a 3D model of the part. The data is exported into software that dissects the model into thin layers 5 to 10 millimeters thick. The laser "paints" one of the layers, exposing the liquid plastic in the tank and hardening it. As the platform drops into the tank the next layer is hit by the laser. The process repeats until the 3D part is created. The process is not very quick. It can take less than a minute to several days run time depending on the size of the part. Once the process has finished, the object is rinsed with a solvent and "baked" in ultraviolet oven that cures the plastic (Marshall, 2015).

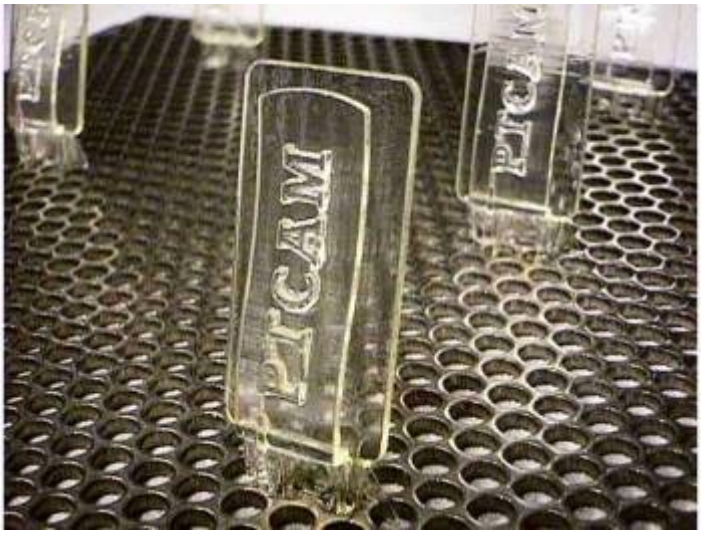

Figure 7: The platform at the end of a given print run

3 Design Process

3.1 Functional Requirements

The system shall:

- Launch tennis balls between 10 and 50 feet
- Provide autonomous fetch option
- Deliver treat on differing intervals
- Protect the launch path
- Be weather resistant
- Deliver differing launching angles and distances
- Be home transportable

3.2 Design Requirements

- Provide user input of inclusive launching angle
- Provide user input for launching distance
- Design a funnel style hopper for retrieval
- Provide user input of interval treat training
- Design a dispensing system for the treats
- Sense the launching path before firing ball
- Provide a weather proof shell that limits water ingress
- Provide a size and shape that is transportable about a home environment

3.3 Design Specifications

3.3.1 Product Definition

FIDO is a ball launching device that provides entertainment for dogs through launching tennis balls. The launcher has a built in training feature. The launcher can be used both inside and outside, and can be set to different launching distances. It sits on a rotating base so that the ball can be launched in different directions each time, helping the dog not become bored due to monotonous launching patterns. The strong plastic housing protects the internal mechanisms from some environmental conditions, such as water splashes and high/low temperatures; however, it is not intended to stay outside through all weather. The battery operated launcher can last for several hours, and includes an AC adapter. It also includes safety shut-off switches to prevent the ball from being launched while the dog (or a person) is too close to the launcher opening.

We expect FIDO to include:

- Ball launching mechanism
- Casing that encloses all mechanical and electrical elements and is both durable and safe for pets and humans alike
- Battery and charger
- Treat dispensing mechanism
- The ability to be moved easily by owner, but sturdy enough that pets cannot move it.

3.3.2 Product Requirements and Design Constraints

The FIDO system shall:

- Be a portable dog training/entertainment system
- Launch standard-size tennis balls using a pinball-type spring-operated launching mechanism
- Have an easily-accessible hopper for the dog to return the ball to

- Incorporate a treat-dispensing mechanism
- Operate with a rechargeable battery
- Have an exterior casing that will:
    - Cover all moving parts and electronics
    - Be durable and resistant to pet teeth and claws
    - Be weather-resistant
- Be operable by pet (once switched on by human)
- Be able to launch balls at varying distances and rotation angles
    - Range of launching angle: 15° - 45°
    - Range of distance: 5’ – 50’
- Be portable, but sturdy enough so that it will not be moved by a dog
- Include a power supply that is enclosed in the housing
- Include an AC adapter for charging the battery
- Operate both indoors and outdoors
- Operate autonomously

3.4 Component Designs

After finalizing our Product Design specifications and constraints we then broke down our entire project into the following design groups: Launching, pullback, loading, lifting, exterior. For each of these groups the team came up with preliminary designs, analyzed them and then selected the final designs for production.

3.4.1 Launching Mechanism

Before starting Computer Aided Design (CAD) modeling for the launching mechanism, we needed to research the different options currently being used on the market as well as other options not being used. After reviewing the competitions designs we concluded that they were using, wheel based devices (pitching machine) and lever based devices. Knowing that we wanted

to create a unique design that could be marketable, we researched alternatives to the current market designs. The other launching methods consisted of spring loaded devices, tension-corded devices, pneumatic devices, and linear actuated motor devices. The next step in determining the right propulsion method for our design was to create a design matrix, rank the methods and then select the best option.

Device Type	Pro	Con	Rank
Spring-Loaded Device	Simple. Very few moving parts. Easy to cover.	To be automated, needs a motor or actuator to compress spring before launching.	4
Tension-Corded Devices	Easy to cover.	High stress in tension rope. Many parts. Large.	2
Pneumatic Devices	Simple. High force to input power ratio. Easy to cover. Very few moving parts.	Could be quite large to hold enough air. Stability of system might require heavy base. Latency between shots may be longer than desired.	3
Lever-based devices	Simple. Compact.	To cover might make quite large.	3
Wheel-based devices	Consistent. Tried and true concept.	Already on the market being used in dog fetch products. Controls for safety are hard.	2
Linear Acuator Motor	Consistent. Small package size. Size to performance ratio is excellent.	Complicated software needed to control the actuator.	3

Table 1: Launching Mechanism Decision

Table 1 shows our decision matrix for selecting the ideal launching mechanism. After carefully reviewing all of our options, taking into consideration the ease of use, ease of implementation, if it was currently being used on the market and cost, we determined that a spring loaded device would be the best fit for our project.

3.4.1.1 Preliminary Designs

After determining that a spring-loaded device was the way that we are going to propel the tennis ball we came up with the idea similar to a pinball plunger. A rod (plunger) with a sphere attached on the end would compress a spring and once released it would strike the tennis ball.

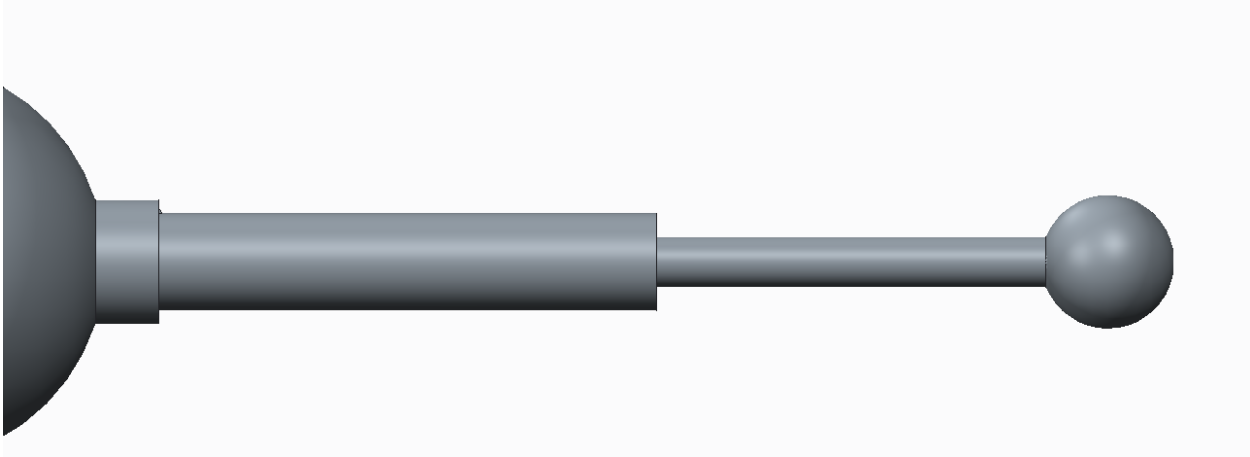

Figure 8: Spring Housing for Launching Mechanism

Our preliminary design for how this would work was that attached to the launching tube there would be a "spring housing" as seen in Figure 8 above. The plunger would have a stop on the end opposite from the sphere so that it is capable of compressing the spring.

The first flaw with the preliminary design is that it did not have any vertical supports. After a quick redesign of this design we added support but there was a much more pressing issue with this design. After purchasing the spring for the mechanism and quantifying how stiff it was, we determined that the preliminary design would not be able to withstand the impact forces that would occur during a tennis ball launch. After many uses the 70 pound impact force generated by the spring would do significant damage to launching tube making it unsafe to use. Since the spring housing would have to be 3-D printed/rapid prototyped, the material of it would be plastic thus making its overall strength low. Other than the impact force, the 70 pound force in the

opposite direction that is being applied on the back of the spring housing would also create many stresses that are not ideal.

3.4.1.2 Final Design

In order to accommodate for the many flaws of our preliminary design we came up with the design in Figure 9. It has two vertical aluminum plates that are supported by two L-brackets a piece. The spring is located between the two vertical plates, with there being felt on the other side of the washer on the plunger and another on the plunger next to the sphere. Not seen in this image are the fasteners, on the base of each L-bracket are two bolts and going through the vertical plate and both L-brackets are another set of two bolts. Therefore for the design shown below there would be 6 bolts supporting each vertical plate. The two pieces of felt add damping to our system so that there will be less of a hard impact which will allow for our product to last longer. The best aspect of this design is that it has superb support in the vertical direction, when the plunger is pulled it will be able move completely horizontally. Another aspect of this design that was significantly improved was the support in the horizontal (pulling) direction. The L-brackets added much needed support for the load created by the spring.

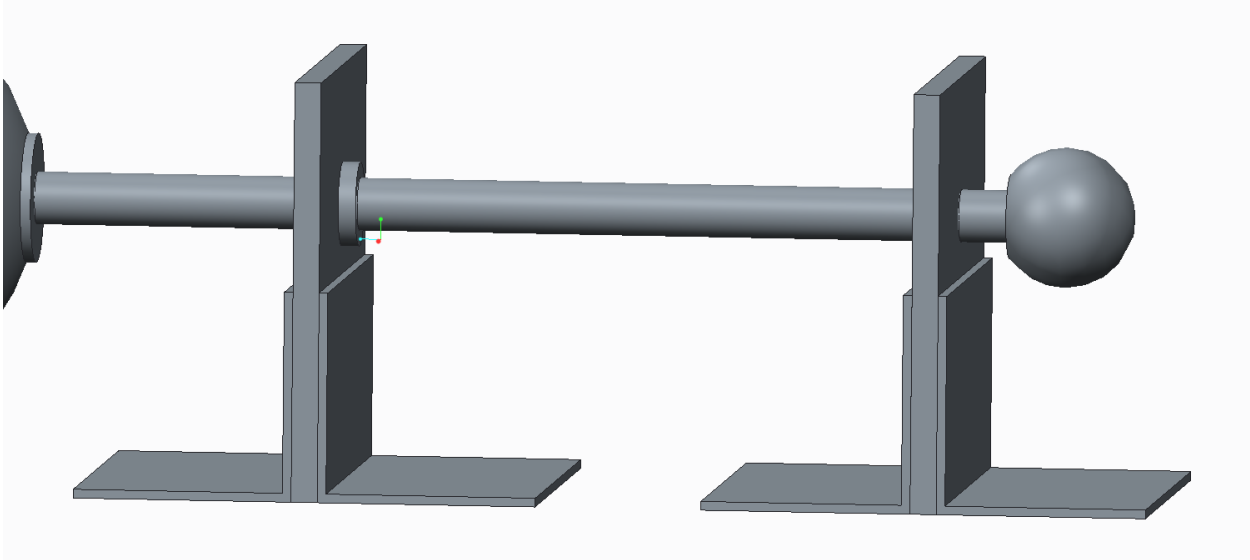

Figure 9: Finalized Launching Mechanism Design

3.4.1.3 Spring Constant Derivation

To calculate the initial velocity of the ball in order for it to travel 50 feet, projectile motion equations must be utilized. Given the angle of the tube is 45° and the desired maximum distance of travel is 15.24 meters (50 feet):

$$x = \frac{V_o^2 \sin(2\theta)}{g} \rightarrow v_o = \sqrt{\frac{xg}{\sin(2\theta)}} \rightarrow v_o = 12.0388 \frac{m}{s}$$

Now that the velocity necessary for the ball to travel 15.24 meters has been calculated the spring constant can be found using conservation of energy. Using the potential energy of the spring and the kinetic energy in the ball, the spring constant can be calculated. In order to conserve space in the interior of the dog ball thrower, the spring will be compressed .0508 meters (2 inches). The United States Tennis Association states the range of the weight of a tennis ball is from .056 kilograms to .0594 kilograms.

$$PE = \frac{1}{2} k x^2$$

$$KE = \frac{1}{2} m v^2$$

$$PE = KE$$

$$\frac{1}{2}kx^2 = \frac{1}{2}mv^2 \rightarrow k = \frac{mv^2}{x^2}$$

$$k = 3145.046\frac{N}{m}$$

This spring constant only accounts for getting the ball out of equilibrium, so it must be doubled to get the results desired.

$$k = 6290.092\frac{N}{m} \rightarrow k = 35.917\frac{lb}{in}$$

3.4.2 Pullback

The next step in our design process was to determine how the plunger would be pulled back. This design would not only have to include a way to grab to the ball at the end of the plunger and eventually release it, but also have a mechanism that could pull the plunger back at varying lengths. To solve both of those problems, we separated the system into two parts; the grabbing mechanism and the mechanism that moves the grabber. This system also has to be rigidly mounted to the angled plate since it acts in line with the plunger and the tube.

3.4.2.1 Preliminary Design

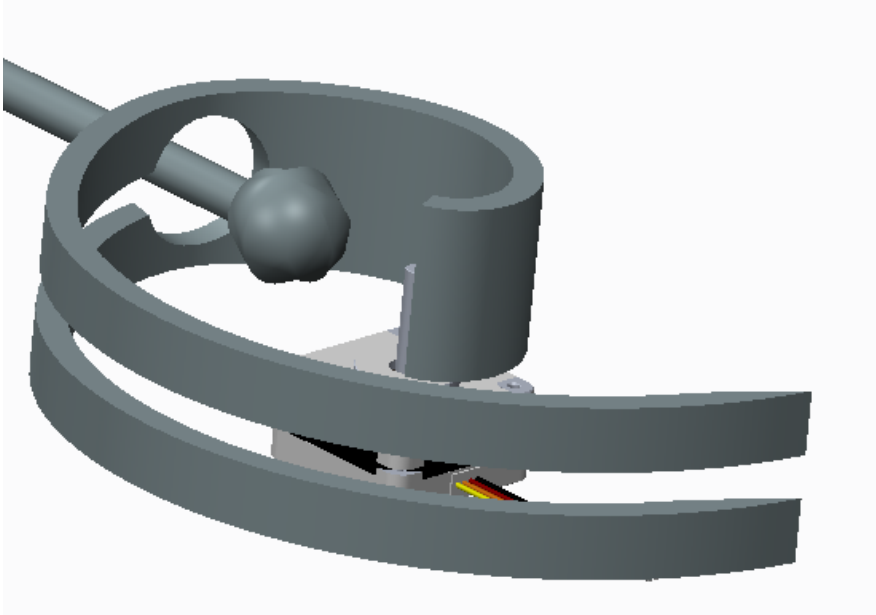

Figure 10: Grabber Original Concept

Our first design for the grabber was an original design. As shown in Figure 10, our first design was simply two forks that spiraled around a center point, which would be rotated by a stepper motor. As the two forks rotated, they would "pick up" the ball on the end of the plunger. Continuing through their rotation, the forks would be pulling the plunger ball linearly closer to the center of rotation as the forks' radii grew smaller. Eventually, there would be a larger gap between the two forks that the plunger ball could fall through, therefore releasing it and allowing the plunger to move forward and strike the tennis ball.

This design had numerous revisions and went through many design iterations. Originally, it was oriented to be rotating vertically, and would be rigidly mounted to the base of the whole system. This would be placed in the same plane as the angled plate, and the amount of pullback would be determined by the angle of that plate. At a lower angle, the point of "pick up" along the forks would be closer to where the fall through was, meaning that the plunger ball did not get pulled back very far. At a higher angle, the pick-up point would be closer to the end of forks,

allowing the forks to pull back the plunger ball even more. This tied the power and angle options together under the assumption that you would want less power at a lower angle and more power at a higher angle. A lower angle would be more likely used inside and therefore would want less power, while a higher angle would be more likely used outside and therefore would want more power.

After deciding that we wanted to separate power control and angle control, we decided to mount the rotating forks horizontally on the angled plate. This meant that the power would now be controlled by how far away the center of the rotating forks was from the plunger ball. By simply moving the forks closer to the ball the pick-up point would be closer to the fall through, and moving them away would change the pick-up point to somewhere closer to the tip of the forks. This simplified the system, but now also required us to create a new mechanism to move the forks forward and backward along the angled plate.

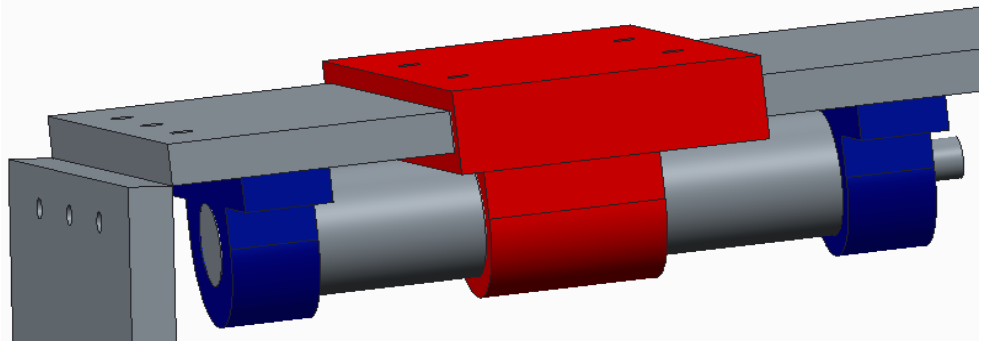

Figure 11: Power Control Mechanism Design (Angled View)

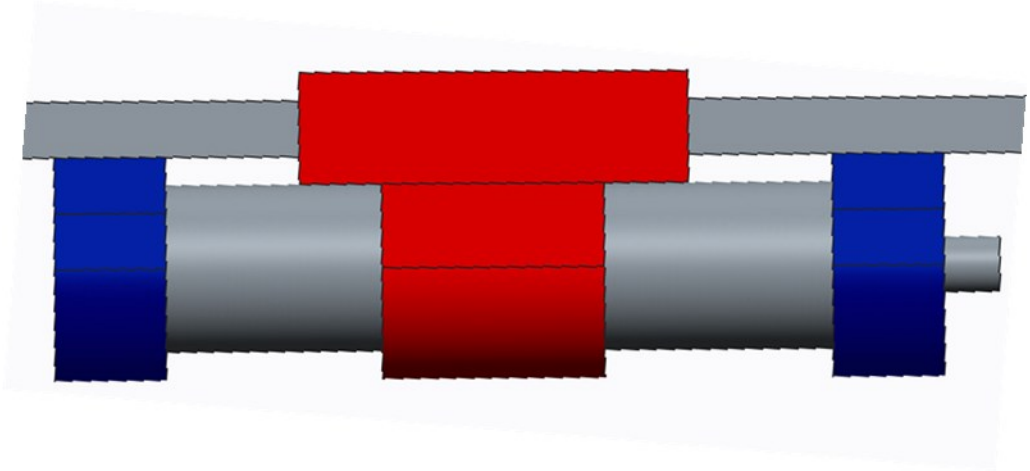

Figure 12: Power Control Mechanism Design (Side View)

For our design, we wanted a mechanism that would provide the most mechanical advantage to overcome the force of the spring. We decided that a lead screw would solve the problem. Mounted on the underside of the angled plate, it originally was designed to be attached in line with a motor and be held in place on its other side of a bearing. A bracket would then be built to attach to the lead screw, ride along the angled plate, and be able to mount a motor on top of it. This motor would then have the rotating forks mounted on it.

This design also had many challenges to it. With the amount of force on the hooks, we would have to make sure that they would not be pushed apart and allow the plunger ball to slip through before it was supposed to. We originally were planning to solve this by adding two plates on both the top and bottom of the hooks to keep them in their place. There was also an issue with if the plunger ball would easily traverse the hooks as they rotated or if there would be too much friction between the two. The most important of the challenges, though, was manufacturing. We could not come up with a way to make the hooks, whether it was somehow bending them to the right shape or making each hook separately and somehow attaching them to each other. With that in mind, the hook design was scrapped and a new grabber design was chosen.

Figure 13: Purchased Claw (c/o Weebly.com)

The next iteration of design for the grabber came in the form of a claw, as shown in Figure 13. Using a claw design, we could mostly salvage the mechanism that would move the claw forward and back. Changing to a claw meant that we also had to change the bracket that it would ride on and the motor it would use; most commercial claws required a servo motor. But, while attempting to move forward with this design, we came across even more problems. The claw's two grippers did not close symmetrically; the grippers were not shaped to grab the plunger ball properly, and provided no way to lock in place so that the plunger ball did not slip out of its grasp.

3.4.2.2 Final Design

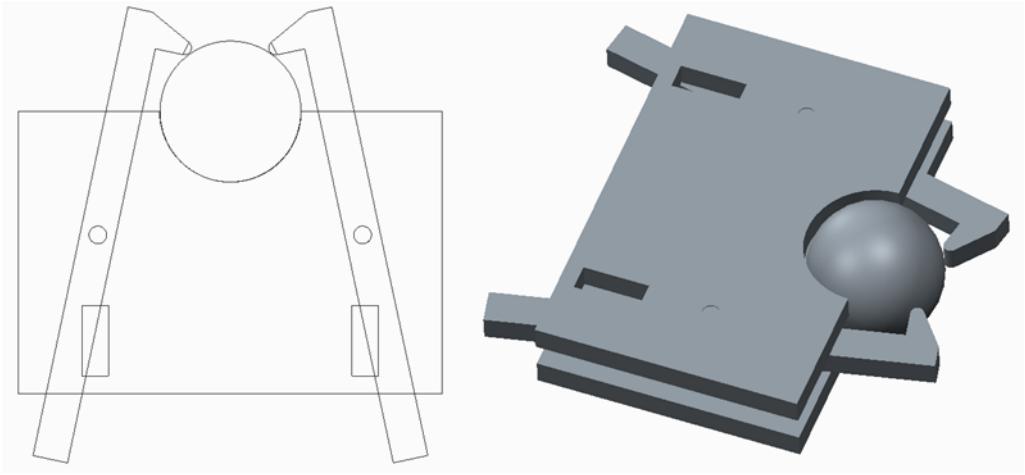

Figure 14: Claw Design in 2D (left) and 3D (right)

For our final design of the grabber, we again went with something original. As shown in Figure 14, two tongs are used to grab the plunger ball while pinned to two plates holding them in place. A spring attached to both tongs holds enough tension to keep them closed, but as they approach the plunger ball, they are forced open. When they close on the other side of the ball, the servo motor engages the locking mechanism. The servo spins, which in turn spins the locking mechanism into position. This prevents the two tongs from being forced out of place by the plunger ball as the whole system is pulled back. When it is time to launch, the servo motor retracts the locking mechanism, releases the plunger ball, and the plunger strikes the tennis ball.

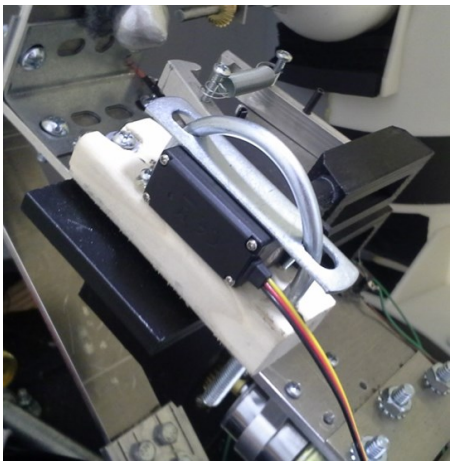

Figure 15: Final Claw Design

This final design also came with some slight changes to how the grabber is moved forward and back. The lead screw is now held in place by two pillow block bearings on either end. On one end, the lead screw extends past the bearing and was turned down to a diameter to fit a gear on it. This gear is then connected to another gear that is mounted on a DC motor, allowing for a 2 to 1 gear ratio used to speed up the rotation of the lead screw. The actual connection from the lead screw to the bracket was also adjusted slightly. Originally designed to be all one piece and threaded to fit onto the screw, there is now a nut that rides on the lead screw. This nut has two pins on either side of it that are pressed fit into the nut, but loose fitted to the bracket. This interaction may allow for some rotation, but the fact that the bracket is designed to ride along the plate constrains the system to only travel along one axis with no rotation.

3.4.3 Loading Mechanism

Knowing that the ball was going to be propelled via a plunger the only design specification that pertains to the launching tube is that it must be able to fit a standard tennis ball in it while having a hole for the plunger to strike the ball through. All of our preliminary designs were based on the fact that the tennis ball needed to be launched out of a similarly shaped

structure. Another commonality between all of the designs created was that once the ball was inside the designed mechanism there was a spherical shaped resting place for the ball in order to ensure repetitive launching results. The idea being that if the ball is resting in the same position before every launch the plunger will be more likely to hit the tennis ball in the same spot every time.

3.4.3.1 Preliminary Design

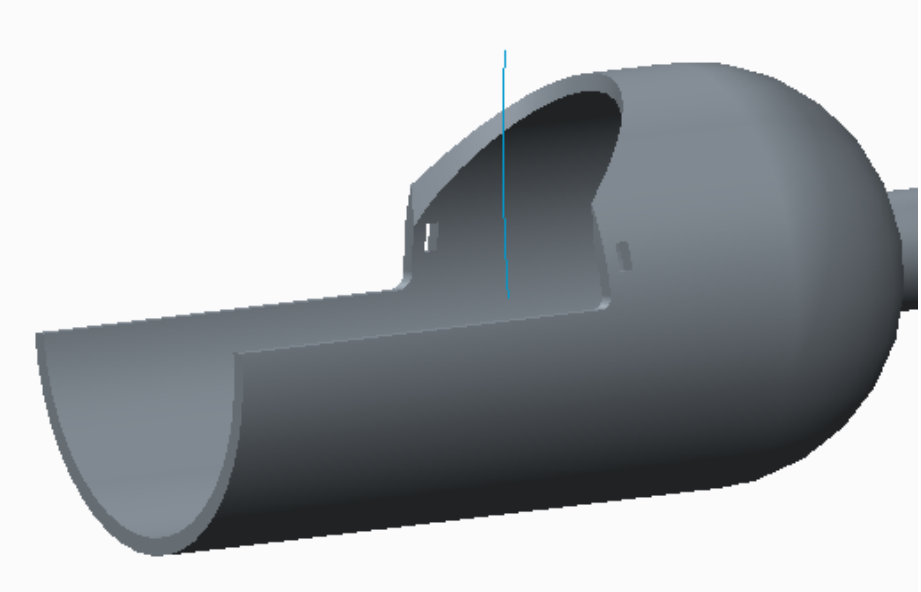

Figure 16: Launch Tube Original Concept

The design seen in Figure 16 is the first iteration of a launching tube. This design has a chute like exit which while allowing for easy tennis ball entrance, it does not have enough tube for the ball to have a smooth trajectory once launched. Being the biggest flaw the smooth trajectory was a design specification not thought of before the design process, thus the next design iteration would have to account for it.

While simultaneously coming up with launching tube designs we were designing tennis ball transportation designs. The main idea behind these designs was that they were to be gravity

driven and to be collapsible. The collapsible design constraint is critical because while the tube angle increases this transportation device would also have to move upward at a varying angle. The transportation device would have to compress so that the outer shell would not have to be so large.

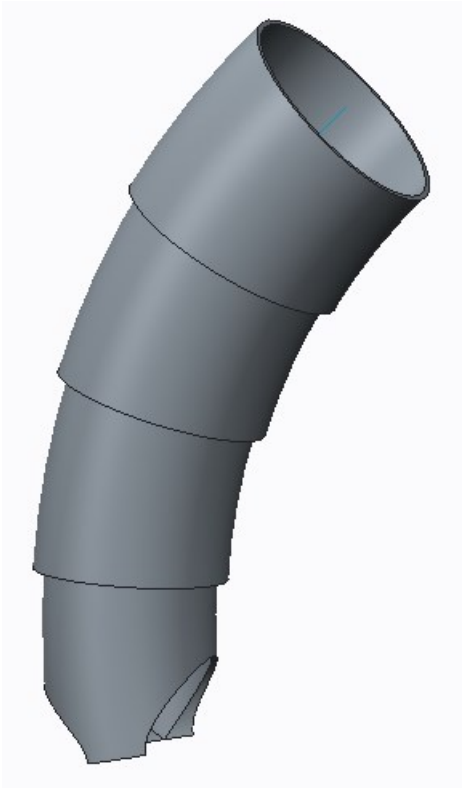

Figure 17: Collapsing Transport Tube Original Concept

The design in Figure 17 is for a collapsing tube tennis ball transportation system. Designing the tubes to collapse at the same radius as the launching tube, this design provided a direct and simple path to the launching tube. The flaws of this design were that once the connection to the outer shell was made they took up valuable real estate within the shell as well as constraining the rest of the lifting and launching mechanisms.

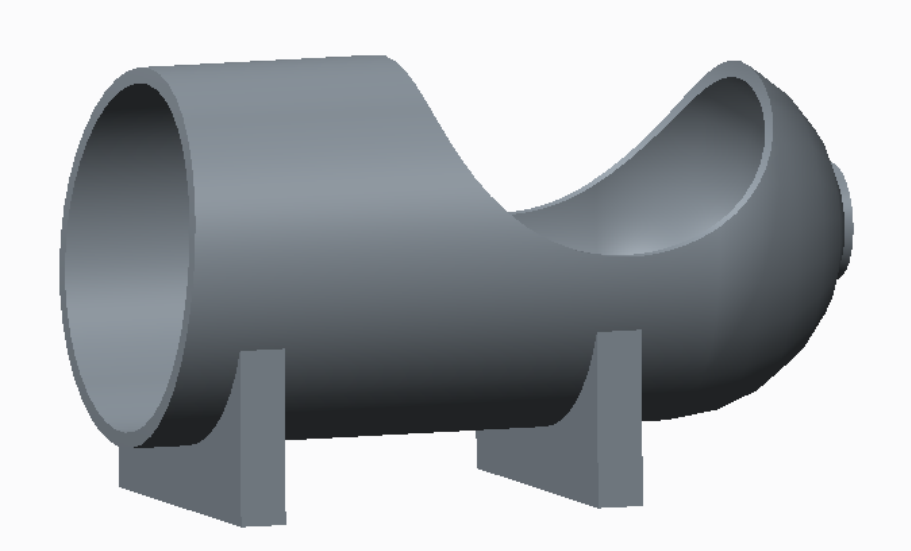

Figure 18: Launch Tube Revised Design

As shown in Figure 18, the next preliminary design was that of a full length tube which had a circular slot in it. This design for a while seemed like it was going to be our final design. The flaws with this design were that it was not long enough and the circular slot, although a good design and it meshed well with our collapsing tube, it needed to be redesigned once we decided that the collapsing tube ball transportation method was not the path we were looking to take.

3.4.3.2 Final Design

Figure 19: Dryer Hose (c/o Amazon.com)

In order to be able to easily attach to the tennis ball entry point, we decided to scrap the collapsible tube idea and go with a much more flexible and cost effective product. Instead of 3-D printing the collapsible tubes we purchased dryer hose. Dryer hose is an accordion like tube as seen in Figure 19. This product fit every design aspect we were looking for, it easily collapses, a tennis ball roll down it, the movement of it is not constrained to only one direction and it was time and cost effective.

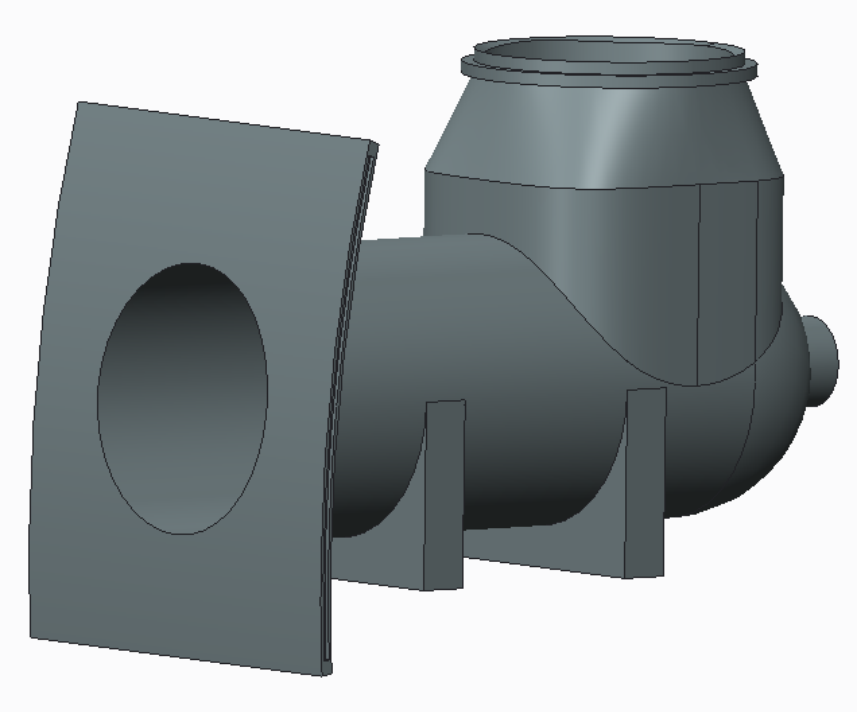

Figure 20: Finalized Launch Tube Design

After deciding that the dryer hose was the ball transportation device we were going to use, we needed to slightly adjust the design of the launching tube. Using the design in Figure 20, we redesigned it to have a longer barrel, for better accuracy, and redesigned the tennis ball entry hole to accommodate for the dryer hose.

3.4.4 Lifting Mechanism

This category of designs fulfills the design specification that the final product must be able to change the angle in which it can launch tennis balls.

3.4.4.1 Preliminary Design

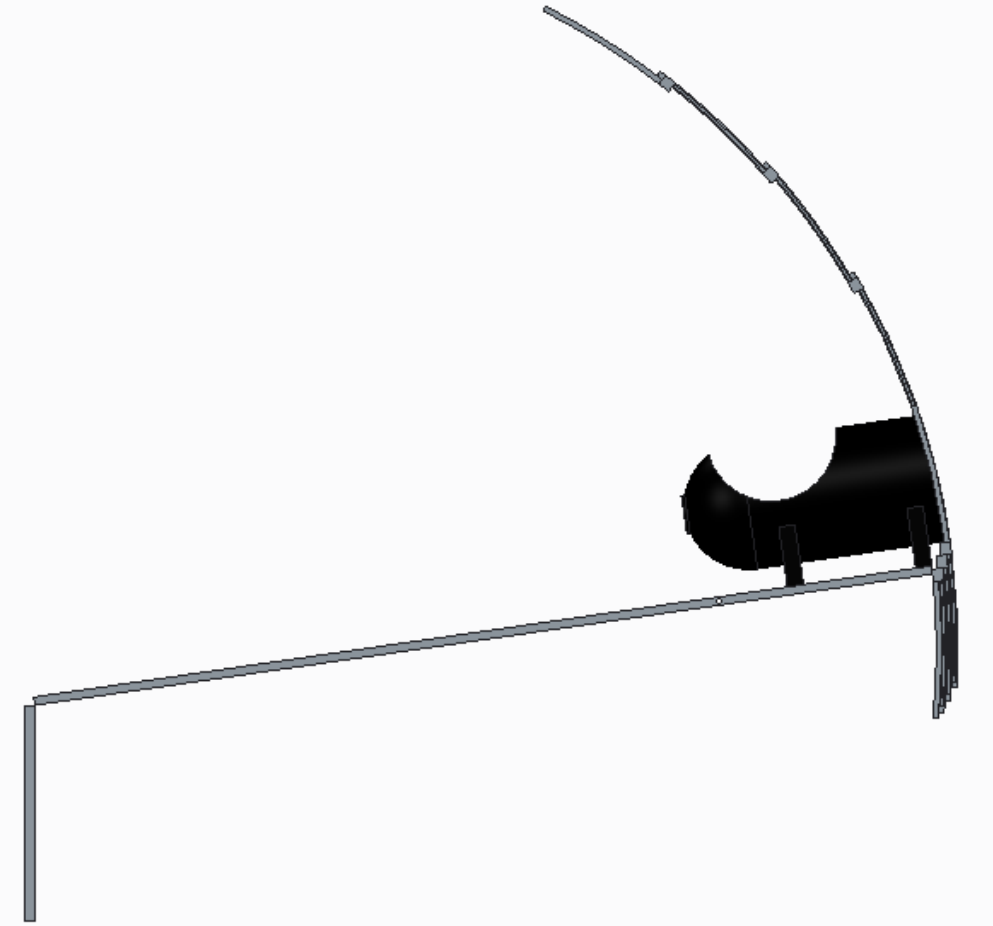

Figure 21: Lifting Mechanism Original Concept

There were a handful of possibilities to achieve the angular change that our team was looking for. One of the definite choices we made in our design was to pin the back of the angled plate in the back of the system and up a few inches from the base. This gave our team a longer angled plate, which would be needed for mounting the entire launching mechanism. This also allowed for space underneath the angled plate for whichever lifting mechanism we chose. This original design is shown in Figure 21.

Most of our team's early design iterations for the lifting mechanism included another linkage connected to the angled plate. This linkage would be pinned to the angled plate somewhere towards the front of the plate, where the tube is mounted. The other end of the

linkage would be controlled by whichever mechanism we chose, but would likely push the end of the linkage forward and backward. This forward and backward motion would translate to a vertical change on the other end of the linkage, creating the change in angle that we were looking for.

Our team then had to choose the exact system that we wanted to control our lifting mechanism. We wanted something that provided enough mechanical advantage to overcome the amount of force that would be needed to lift all of the weight on the angled plate, as well as something that could provide the most possibilities in terms of different stopping points that would produce different angles of launch, One of the early designs was a multiple pin site system. Like in most cable gym equipment, you would release the pin from one point, and pin it back in at another to achieve the change you were looking for. This design may have provided the most support for the system, but did not allow for precision when changing the angle and it would require a lot of floor space on the base to have an adjustable handle on the outside. A second early design included a pulley system. This design would either not include the lifting link or pull the angled plate up on its own, or, if combined with a spring, it would still be used to engage the lifting link linearly by pulling it in one direction while allowing the spring to return it in the other. The pulley system seemed interesting to our team, but ultimately was not chosen either. Our third and final preliminary design was based on a rack and pinion system. A pin in the bottom of the lifting link would have two pinion gears attached to it that would travel along two racks, creating the motion we needed. This design, however, was also flawed. We were unsure if the pinion gears would remain in place or if they would be moved by the weight of the system. We also were unsure how to control those gears properly, and with that, the design was scrapped.

3.4.4.2 Final Design

The final design we ended up selecting was that of the lead screw and lifting link, similar to the mechanism used to control the grabber of the pullback system. This design consists of a lead screw that is supported by two pillow block bearings, a hex nut that attaches directly to the lifting link, the lifting link then is attached to the propulsion track via a pin. The propulsion track attached to a vertical plate through a hinge, this track is where the launching tube and spring system are placed.

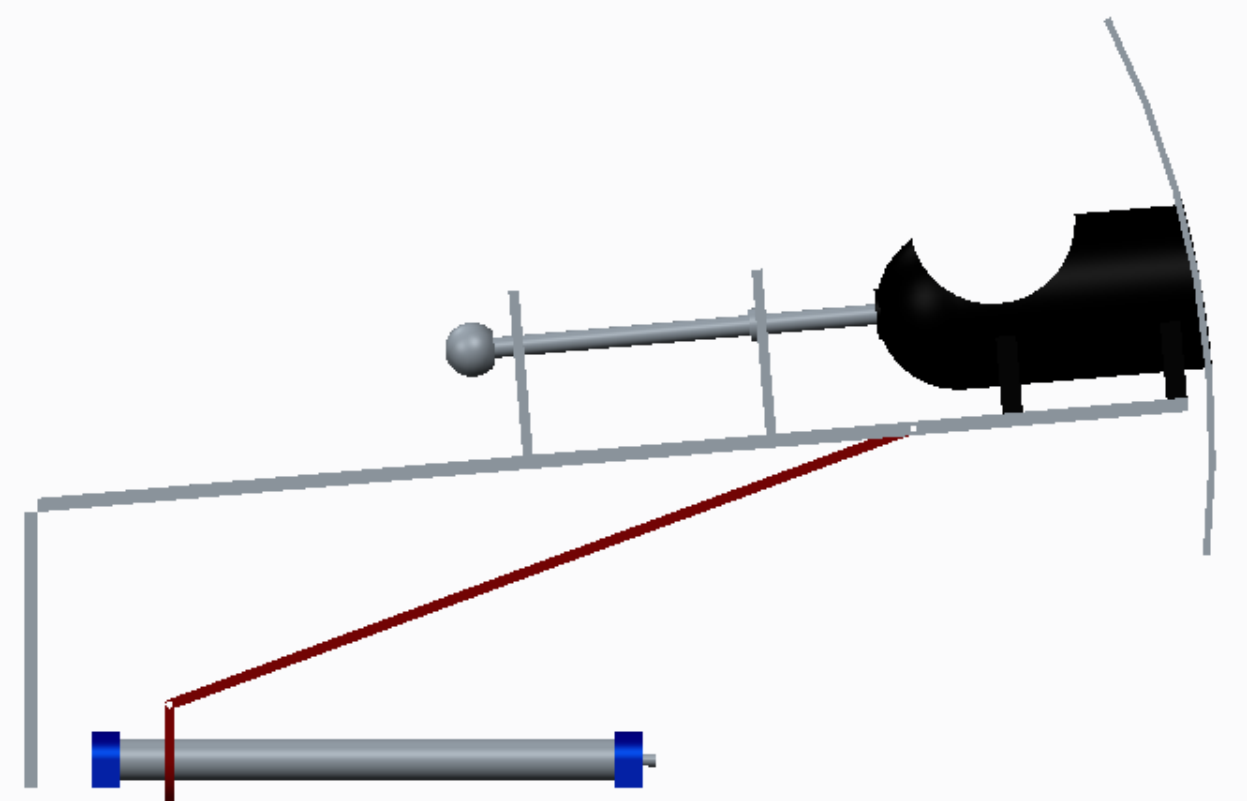

Figure 22: Lifting and Launching Mechanisms Overlay, Revised Design

The way in which the propulsion track's angle is changed is through moving the nut forward or backward on the lead screw. Once the nut is moved it pushes the lifting link which in turn changes the angle of the propulsion track. The lifting link is pinned on both ends and has the ability to rotate. The DC motor which drives the lead screw has a 24 tooth gear on it and has an

angular velocity of 20 RPMs. In order to step this speed up the screw has a 12 tooth gear on it which increases the angular velocity to 40 RPMs.

3.4.5 Exterior

This category of designs is going to show how we enclosed all of the internal systems. There are a few design specifications that deal with the designs seen in this category. The first specification that needed an enclosure design was the requirement of having an exterior shell that covers all moving parts. Going off of that same specification, a design needed to be made so that when the angle of the launching tube changes there would be no space for a dog or person to get pinched or caught.

3.4.5.1 Shell

The exterior shell that encloses all of our internal components may be the most pivotal part to provide protection to our internal system. After lots of design iterations to accommodate for all of the extra parts added the final design below was created. The shell is made from a rapid prototyping method called stereolithography. This part provides proper geometric form without the need of tooling to create the part. The shell incorporates the slot for the launching mechanism and the funnel to return the ball to the launching tube; it also holds the treat dispenser and the display, and has a shelf to help smaller dogs reach the funnel.

Figure 23: Shell

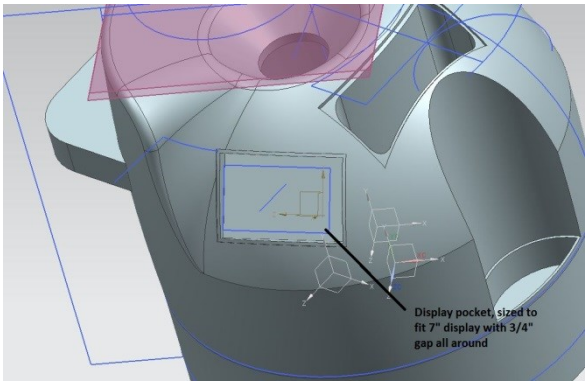

Figure 24: Display pocket in Shell

3.4.5.2 Collapsible Shield

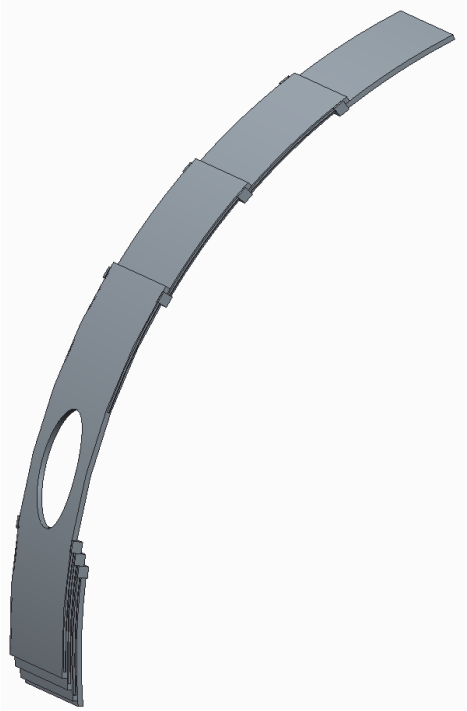

Figure 25: Collapsible Shield Original Concept

The collapsible shield design (shown in Figure 25) covers the space between the launching tube and the exterior shell so that neither dogs nor people will get injured during changes in launching angle. Similar to the collapsible tubes, the shield slides on itself and collapses as the launching angle changes. There is a collapsible shield on both above and below the launching tube so that when the launching angle increases the top shield is collapsing while the bottom expands thus having no space to get injured. The flaws of this design were that it would have to be 3-D printed thus making it not operate very smoothly.

3.4.5.3 Platform

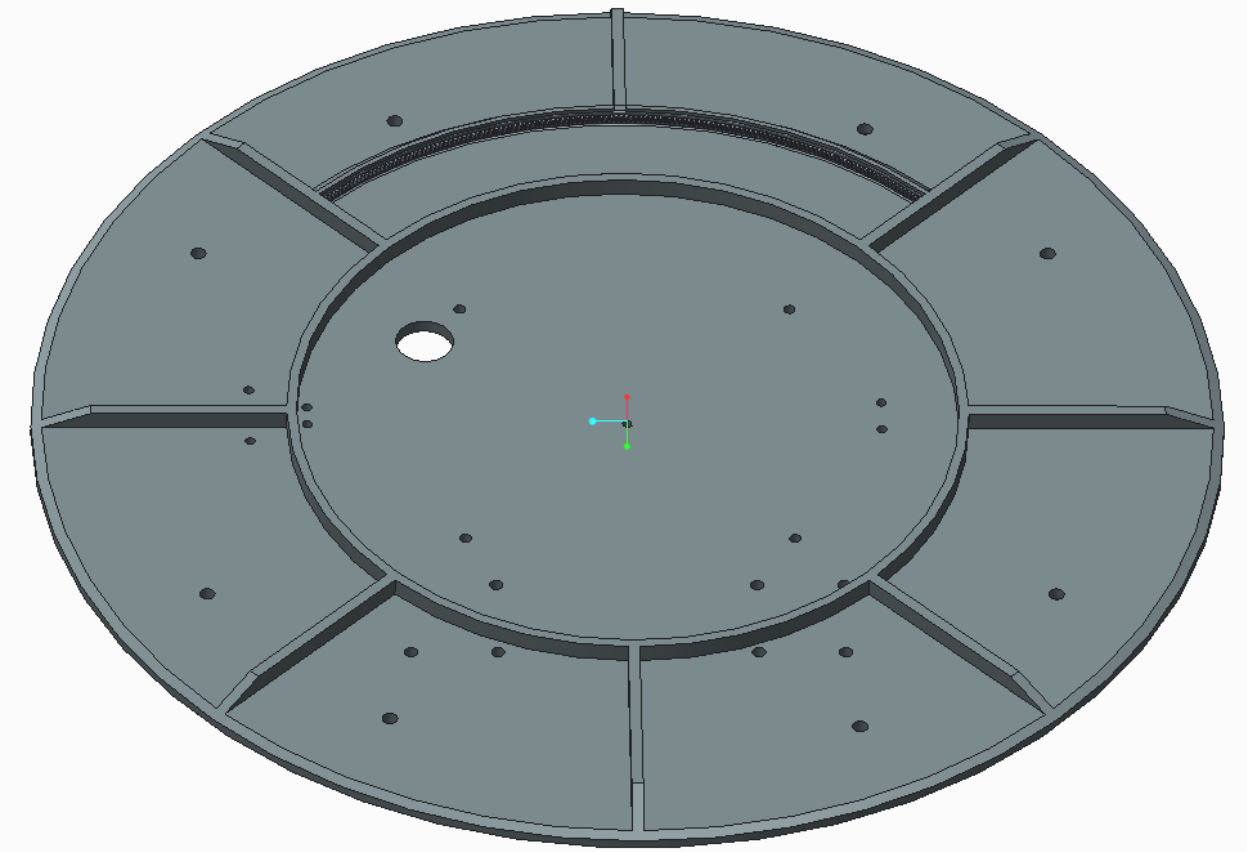

Figure 26: Rotating Platform, Bottom View

As shown in Figure 26, the platform is the mechanism of our system that enables it to change in launching direction or yaw. In order for the yaw to be changed there is a toothed track, seen in the figure below, which is driven by a geared stepper motor. The lifting mechanism, battery, Raspberry Pi, electrical components and exterior shell all are connected directly to this platform. In order for the platform to be able to smoothly spin with all of the weight of the other systems, we purchased and attached an industrial lazy susan bearing.

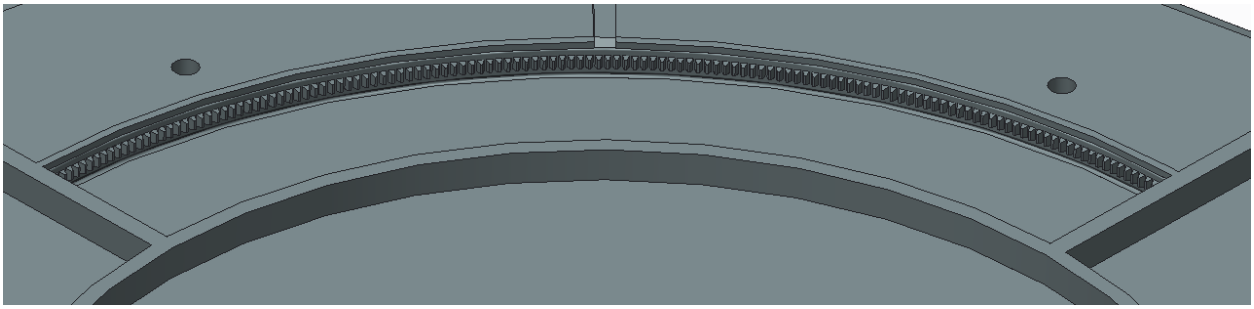

Figure 27: Gear Rack for Rotational Servo Motor

3.4.5.4 Base

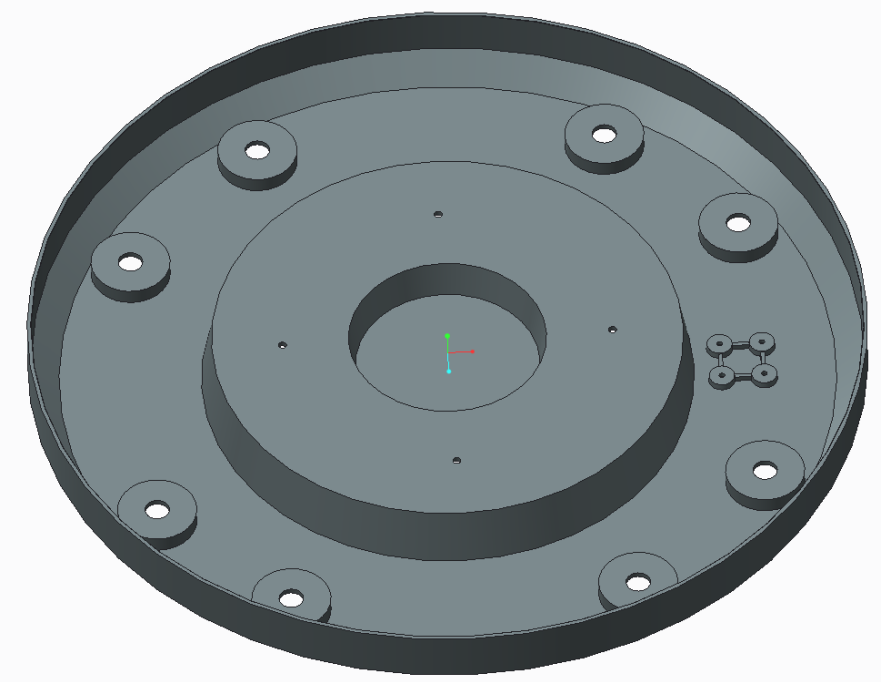

Figure 28: Base Top View

The base is the foundation for all previously mentioned systems. This component (shown in Figure 28) is where the other side of the industrial lazy susan connects. Also attached to this base is the geared stepper motor that controls the yaw of the entire system. This part was made out of stereolithography material. It was designed to encapsulate the full outer diameter of the system. It has clearance holes in the feet that allow for attaching the cover to the platform. Those

holes are covered by rubber feet. The middle top hat section was created to give the part strength for carrying the load of the rest of the system through the bearing.

3.4.6 Treat Dispenser Mechanism

When considering different designs for the treat dispenser, we knew that since dog treats come in many different shapes and sizes, we would need a system that could handle non-uniform objects. Because of this constraint, we could not use the types of systems that are generally used for gumballs. Instead we looked at using a small auger to organize and move the treats.

3.4.6.1 Preliminary Design

Our preliminary design used PVC pipe and fittings to house the auger that would move the treats down to the bowl. The two inch PVC pipe started from the treat hopper and ran straight down to a 45 degree fitting that connected to another piece of PVC pipe that would drop the treats down into the bowl. The auger would then organize the treats and help them to not jam in the angled piece.

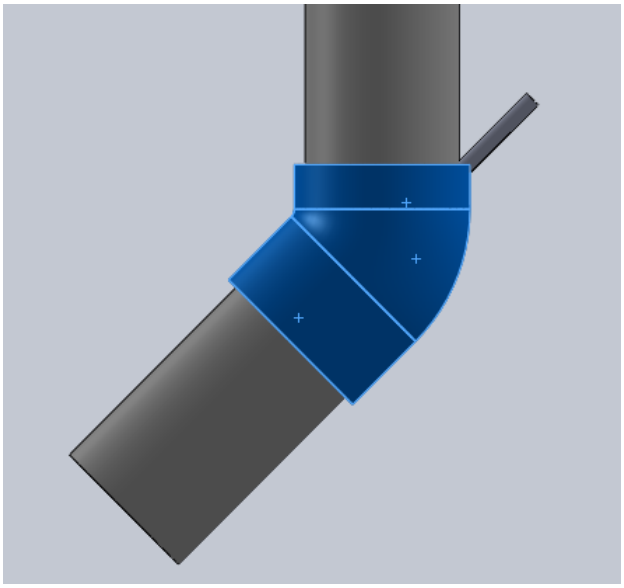

Figure 29: Treat Mechanism Original Concept (Side View, External)

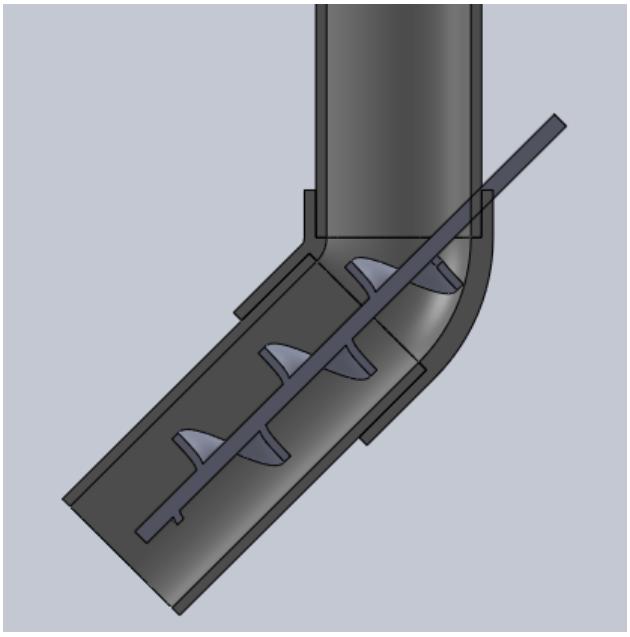

Figure 30: Treat Mechanism Original Concept (Side View, Cross Section)

However, upon testing this design, we found that while larger or abnormally shaped treats went through alright, any small or spherical treats would either roll through the auger on their own or they would drop out if the assembly was bumped or vibrated. Since that was not

acceptable for this system, we had to redesign it. We also had problems working out a feasible way to support the auger where it had to be suspended inside the pipe at the end.

3.4.6.2 Final Design

Our final design changed the auger to moving horizontally through a length of PVC pipe. The treats still fall straight down from the hopper but then the auger moves them through a 90 degree turn and across the flat pipe to another angle that drops the treats down into the dish. The end of the auger is then supported by the second angled PVC fitting which solved our support problem from the previous design.

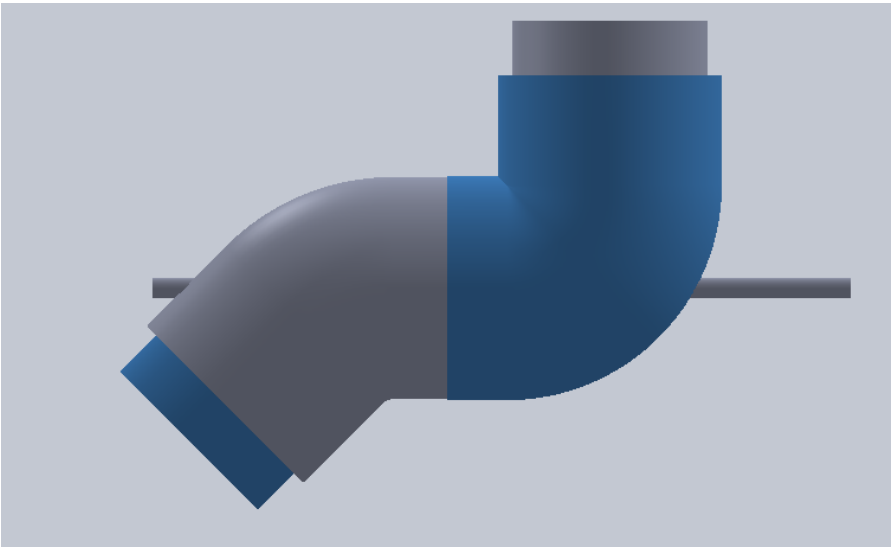

Figure 31: Treat Mechanism Final Design (Side View, External)

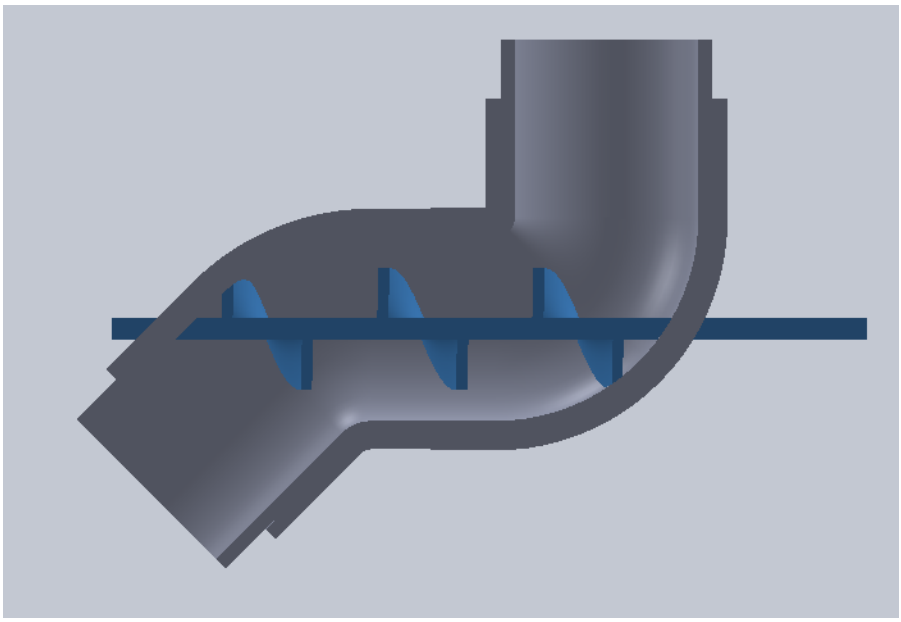

Figure 32: Treat Mechanism Final Design (Side View, Cross Section)

3.5 Power Source

Once all the motors and computing systems were chosen, we were able to decide on a way to power the machine. As per our initial design specifications, we needed the entire system to be able to run on battery. We discussed having a battery that could be charged outside of the machine and then inserted but then decided to have the battery and charger both inside the casing so it would be simpler for the user.

There are many different types of batteries that we looked at, but we settled on using lithium ion batteries. Lithium ion batteries have a large power capacity and do not have a memory like nickel-cadmium batteries, and their power output does not diminish as the power supply dies down. With the battery having to run the Raspberry Pi system as well as the screen, it is better for the power to die off completely rather than slowly taper off, allowing some things to continue running longer than others. This is a benefit that a using lithium ion battery offers.

The only challenge with lithium ion batteries is that they are sensitive and must have many fail safes built into the charger to make sure they do not overheat or have power spikes.

Considering this, we chose to go with a commercially available power tool battery and charger for ease and cost effectiveness. Knowing what the power usage of all the components were, we were able to decide on a 12 volt battery with a 3 amp hour rating to run the system for as long as possible.

To plug the charger into the wall to charge the battery we used an IEC 320 C6 connector (the type found on most laptop chargers) so the cord would be easily replaceable for the user if needed. It is also much more cost efficient that using a more obscure type of connector.

Figure 33: IEC 320 C6 Connector

3.6 Electrical Input Devices

The input devices are all devices that take information from the environment and transmit that into the computer. In FIDO, these are the switches and sensors, though microphones would be considered input devices as well.

3.6.1 "Limit Switches" (Snap Action Micro Switches)

Each micro switch has 3 terminals, of which the setup for FIDO only uses 2. The C, or Common, will go to ground. The NO, or normally open, will be the pin for that switch. It should always read in as 0V until the connection is made, at which point it reads in HIGH (3.3V). In this

way, the connection is made when the limit is reached. Alternatively, one could connect the NC, or normally closed, pin, which would read in HIGH until the connection is broken.

3.6.1.1 Pull-Up / Pull-Down, Rising, and Falling

If the input switches are connected from power, through the switch, to the pin, they are pull down resistors. By declaring it a pull-down resistor, the Pi knows that minute fluctuations in the voltage on the pin should be "pulled down" to LOW and not considered. The moment the switch is tripped and the connection is made, the pin reads in a HIGH voltage. The "rising edge" captures the moment the pin voltage changes from LOW (0V) to HIGH (3.3V) at the moment the connection is made.

If the inputs switches are connected from the pin, through the switch, to ground, they are pull up resistors. By declaring it a pull-up resistor, the Pi knows that minute fluctuations in the voltage on the pin should be "pulled up" to HIGH and not considered. The moment the switch is tripped and the connection is made, the pin reads a LOW voltage. The "falling edge" captures the moment the pin voltage changes from HIGH to L at the moment the connection is made.

3.6.2 "Force Sensor" (Force-Sensing Resistor)

The force sensor isn't a true "switch" in that one can read the voltage directly to a GPIO pin. Rather than a simple switch, it is a resistor. This means that it must be connected to a voltage divider, and the voltage measured in relation to a fixed voltage, to determine a voltage for the IO pin. The pin reads a voltage based on the voltage divider equation

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

R2 is fixed, and Vin is fixed at 5V. Given the resistance range on the force sensing-resistor, a 47 kilo Ohm or more resistor is an appropriate choice for R2.

3.6.3 "Proximity Sensor" (Passive Infrared Sensor [PIR Sensor])

The PIR sensor has three terminals, but only requires one GPIO pin. The three pins are Ground, Output, and Power. The power runs directly to +5V, and the ground runs directly to ground. The output pin goes to the GPIO part of the Pi.

In addition to the terminals, the sensor features two potentiometers changed with a Philips screwdriver. One adjusts the sensitivity of the sensor (i.e. how far out it looks) and the other the delay (how long it waits between taking a new look out there). Additionally, the jumper sets on the lower left can set if the trigger is non-repeatable or repeatable. If non-repeatable, the signal automatically returns to low after the delay time is up and it was previously triggered to high because of movement. If repeatable, the delay will extend if movement is detected. For FIDO, the jumper is set to non-repeatable, so the reset trigger can be controlled in the Python code.

3.6.4 Touchscreen

A touchscreen was added, so that the user could adjust the settings for angles and spring pullback in a simple and effective way. Unfortunately, the "touch" component never worked. The drivers to convert the capacitive touches to mouse clicks never compiled into the kernel for the Pi's Linux-based system. However, the hardware remains in place for the touchscreen, and future iterations of the FIDO system may succeed with developing the touchscreen and a graphical user interface as an input system.

3.7 Electrical Output Devices

The output devices are all devices that take a signal from the computer and transmit that back to the environment. In FIDO, these are the motors, though speakers would be considered output devices as well.

3.7.1 Stepper Motors

A stepper motor operates based on turning on and off small electromagnets in the motor. Each “step” occurs when the electromagnet incrementally spins the rotor toward it. Stepper motors are controlled by sending on/off pulses to each set of magnets in succession. By varying the order of pulses sent, the direction can be reversed. Stepper motors require their own integrated circuit chip. Though it is a 4-wire lead on the motor, the stepper motor requires 5 pins - 4 pins for each wire lead and one for the enable. Our stepper motors can use the NTE 1749 chip. The 4 wire leads from the motor, become the 4 outputs, and 4 GPIO pins become the 4 inputs, and the two enables go to one GPIO pin. This motor can also be run from a NTE 2018 chip, but the NTE 1749 is slightly easier to use.

3.7.2 DC Motors

A DC motor is simple in concept. Running power to one of the leads (and connecting the other to ground) spins the motor, and the spin can be reversed by running power in the opposite direction. To do this, the DC motors require a NTE1749 chip. Pin 8 provides the motor power, and hooks directly to the battery, since the Pi can't provide that. Pin 16 provides the chip power, and hooks directly to a 5V pin. One NTE 1749 chip can run two motors. The two leads from the

motor become outputs 1 and 2, and two GPIO pins become inputs 1 and 2. The common enable for inputs 1 and 2 becomes the third GPIO pin associated with that motor. The other half of the chip can be used to run a second motor. Providing power to one pin but not the other turns it a certain way. i.e. NTE1749pin2 HIGH, NTE1749pin7 LOW = clockwise, NTE1749pin2 LOW, NTE1749pin7 HIGH = counterclockwise.

3.7.3 Servo Motors

Servo motors explicitly turn to a specific angle, but only in the range from 0 to 180 (common) or 0 to 360 (less common, but still available). More rarely, servos make multiple turns (i.e. 720, for remote controlled boat racing winches). The servo motor has three leads, but two go directly to power and ground. The third pin is the signal, and this motor only takes a single pin input signal (direct from the GPIO pin). The pulse length of the signal determines the angle position of the servo. This means that a servo can lock in to a specific angle and hold it much better than a stepper can.

3.8 Programming

Given that the team's programming specialists were more comfortable using Python than C, the Raspberry Pi Model B+ became the obvious choice. The 40 pins (26 of which were General Purpose Input / Output (GPIO) pins) on the Pi B+ connected the six limit switches, force sensing resistor, PIR sensor, stepper motor, servo motor, and three DC motors. Of the four USB ports on the B+, only two were needed at most, to connect a mouse and keyboard to the system for programming and testing. In the final design, the USB ports held two receivers, one for a wireless touch-based number pad / mouse combination, and the other for a wireless touch-based

full keyboard/mouse combination. The number pad was used for user-input during program operation, while the keyboard was for quick changes to the code, as well as starting and stopping full operation during testing and demonstrations.

The final revision of the operating system image would run the FIDO code upon startup, eliminating the need to have a full keyboard. Providing too much user control could lead to unintended results, including the ability to force-quit the running code, which could cause the motors to crash and damage or shatter critical components (including the launch tube or rotating platform) on the product.

The programming of the Pi can be essentially condensed into a basic flowchart. This "pseudocode" flowchart is shown in Figure 34. It can be written in longhand as follows. The system waits for a ball to be placed, doing nothing until the ball is in place. Once the ball is in the launch tube, the system retrieves the user input. Ideally, this would be from the touchscreen; at current, it is from the wireless number pad. Once the user input values for the angles on rotation, elevation, and the spring pullback are submitted, the system converts those values into instructions for the motor. This can be the number of steps (and the direction) to turn the stepper motor, or the time in seconds (and the direction) to turn the DC motors.

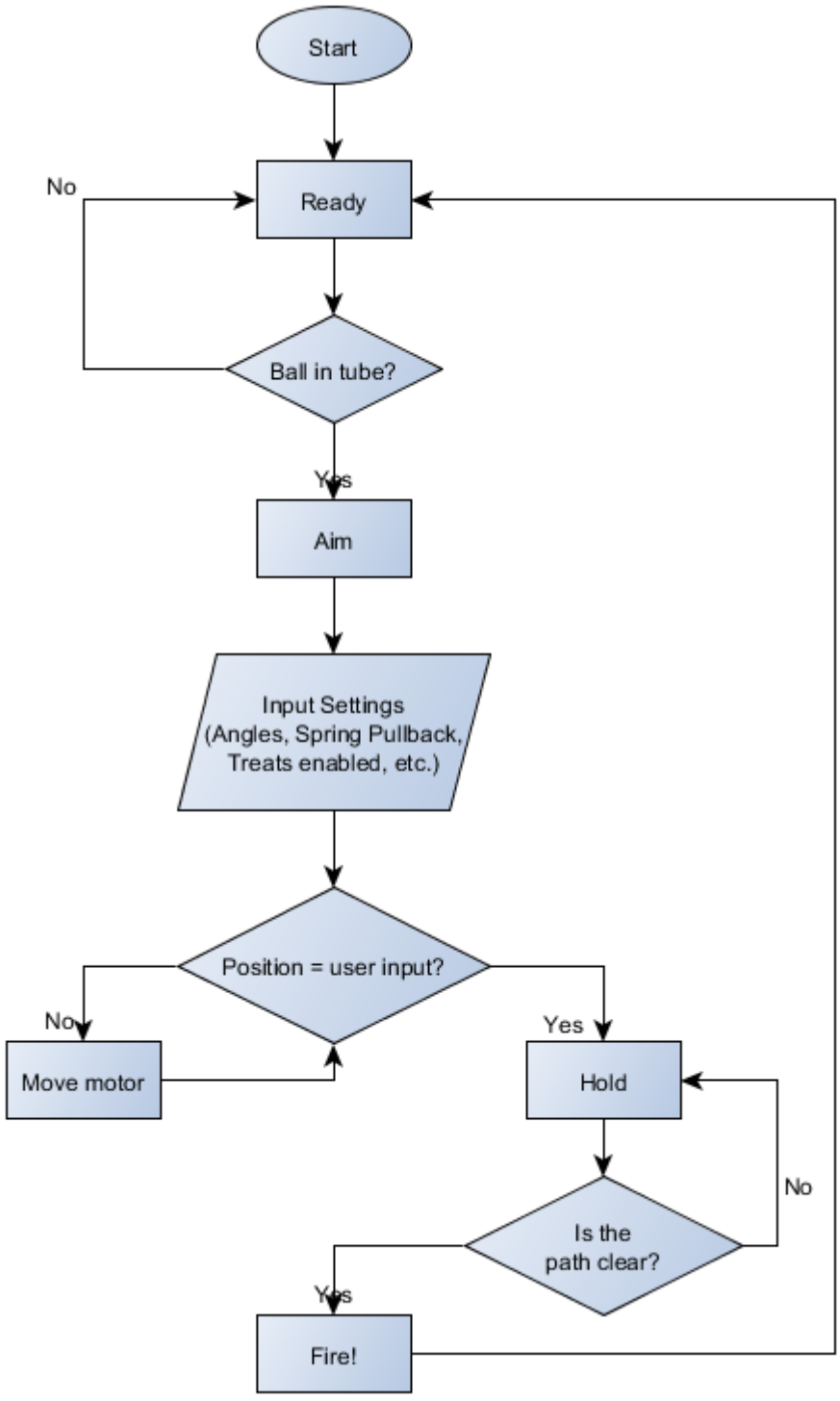

Figure 34: Pi Pseudocode Flowchart

The code used is included in Appendix A: Pi Code. The majority of the code defines subroutines and functions designed to handle a specific action. These functions can (and often

do) rely on other functions, so that often-used code is only written once, and does not need to be needlessly repeated. A tiered system is used to clarify the organizational structure. Tier 0 functions are those that directly take inputs from or send outputs to the GPIO pins, as well as those that only use functions and operations built into Python. Tier 1 functions are functions that rely on Tier 0 functions. Tier 2 functions rely on Tier 1 and Tier 0 functions. Tier 3 functions rely on Tier 2 and lower-tier functions, and so on. The tiered list, along with function definitions and further code analysis, is in Appendix B: Code Analysis.

Given how few lines of code it takes, the servo motor is set up in-line of the program. This is also because of the different GPIO code used to control the servo motor when compared to the DC or stepper motors.

The servo motor runs based on pulse width modulation. In other words, the signal sent by the code to the servo motor is analyzed not for its value or status of HIGH or LOW, but for the width of the pulses sent. For instance, a pulse 1.2 ms long sent every 10 ms will set a certain angle on the motor, but a pulse 1.9 ms long sent every 10 ms will set a different angle. Future revision of the code will use another function to define the servo motor's operation, to ensure that this code is not accidentally erased while revising the main running of the program.

Notably, particular GPIO functions are used to great effect. The GPIO library contains code that can "ping" or call a particular input value to check its value as HIGH or LOW. Depending on the function selected to make this call, the input value can be checked once, continuously, or passively. If the input is checked continuously until a condition is met, the program will not continue to the next line of code until the input value is read as a particular state. This is used to ensure that the force sensing resistor reports a ball in place before the program continues onto the aiming and firing of the device. A passive check allows for other

code to run, but changing the state of this input will cause a specific function to run, no matter what other circumstance are occurring. This is used with the limit switches. Though used for calibration purposes in the initial setup, a limiting factor is used to ensure that the normal running of the code does cause a switch to be tripped. In this way, their purpose has changed from “calibration” to “emergency failsafe.” If something goes wrong and a switch is tripped, the passive check will run an emergency stop function that holds all motors, slowly removes all tension out of the spring, and re-runs the calibration functions of all the calibrated motors.

4 Construction and Testing

4.1 Machining and Printing

4.1.1 Printing

The printing of the shell and base were done via the stereolithography process (SLA). SLA process required an output of a 3D para-solid part file. For the shell and the base, the parts had to be segmented into 4 pieces as the single part design was too large for the SLA working table volume. This required that the parts be glued together post creation. This was done with both cyanoacrolate and urethane glue. The cyanoacrolate gives an instant bond to hold the parts together and then finish with the more flexible urethane glue which is needed to ensure the bond is not too brittle.

The platform, display door, treat door, launching tube, claw mount and locking block were all made using Fused Deposition Modeling (FDM). The FDM process requires a 3D .stl part file which is read into the Catalyst software. There, the user had position the model in the working platform and build the support structure. When the parts were done being made, they had to be put into a vat of Drano to remove any leftover support material. The platform was created with 9 separate pieces that were glued together with urethane glue. Again, this was done because the design was too big to fit into the machine. The FDM parts have material properties very close to Acrylonitrile butadiene styrene (abs) plastic. This allows for machining and modifications post processing. This was a very key advantage for our team. Conceptual development of the first prototype resulted in the need to modify and change the design. This was easier due to the material choice of these key components.

4.1.2 Machining

4.1.2.1 Threaded rods

The purchase of two foot length of zinc-plated steel threaded rod provided our team with enough length to create the two lead screws for the lifting mechanism and the pullback mechanism. First cut to rough length in a band saw, they were then put in a lathe and faced off to the correct length. At first we worried about damaging the threads in the chuck of the lathe. We attempted to hold it in place with two nuts tightened to each other, but the hold from the nuts to the chuck was not concentric. It was then determined that putting the threaded rod directly into the chuck did not damage the threads to a point where they could not be cleaned up by running a dye across them. With the threaded rods in the chuck, the ends were turned down to the predetermined size; one side had a half inch diameter to fit into a bearing, while the other side had both the half inch diameter and another step down to a quarter inch diameter to fit inside a gear. Once the ends were turned down, the threaded rods were brought over to a milling machine and placed inside a collet. A flat was then milled on the small diameters of each threaded rod with an end mill, creating a location for a set screw to prevent the corresponding gear from rotating. With the milling of the flats, the two threaded rods were completed.

4.1.2.2 Plates

All of the aluminum plates were cut from a 24” x 24” x 3/16” stock piece of aluminum purchased by the team. The vertical plate, spring plates, lifting link, and angled plate were all cut to rough length using a band saw. From there, they were brought over to a milling machine and cut to a finish length using an end mill. Each plate had a different set of holes that needed to be drilled. For example, the vertical plate needed only mounting holes, while the spring plates

needed mounting holes and a hole for the rod. Despite the different sizes and locations of holes, the process for each was the same. Using drawings developed from our CAD model, exact locations for the holes were defined. After using an edge finder to zero the axes and create a point of origin, the mill was computer navigated to the correct hole locations. Once located, each hole was center-drilled then, depending on the size of the hole, stepped up to its correct size using one or two larger drill bits.

4.1.2.3 Lifting mechanism

The interactions between the angled plate, lifting link, and lead screw required more precision and engineering than other systems. Between the angled plate and the lifting link, a standard interaction similar to a door hinge was used, but required a slot on the angled plate to allow for clearance. For the slot, a computer program in the mill was utilized. After the program finished, we then used an end mill to create the teeth on the inside of the slot that would mesh with the lifting link to create a better connection between the two pieces. After creating the teeth on lifting link in the same fashion, 1/8" holes were drilled through the lengths of the lifting link and the angled plate where the corresponding teeth were located. Leaving a wall thickness of only 1/32", this was the most delicate machining we accomplished. A 1/8" rod was cut to length and the teeth on the plate and link were rounded down with a file to create a smooth rotation, completing their interaction.

A similar style of interaction was planned for the nut that rode along the lead screw and the lifting link, but required more strength. Luckily, the machine shop had an extra piece of stock that had a greater thickness and was slightly wider than the lifting link. The greater amount of material allowed for a stronger bond between the nut and the link. Holes were drilled in a square

pattern on the new plate and the lifting link to attach them to each other. A slot was then cut into the side of the new plate for the nut to sit in, and a hole was drilled through the length of the plate to accommodate the pins. A hole was then drilled through both walls of the nut to also accommodate the pins. This interaction required press fit pins on either side of the nut. The pins were about .13" in diameter but were able to be compressed. By creating a tight fit to the nut and a loose fit to the plate, the pins would stay rigid to the nut but allow the link to rotate on them. Therefore, the holes on the nut were drilled using a 1/8" drill, while the holes on the plate were drilled using a .136" drill. The pins then had to be hammered into place, but not deep enough into the nut to reach the threads on the inside. A second nut was also drilled through in preparation for a similar interaction on the pullback mechanism.

4.1.2.4 Plunger and gripper

The plunger is made out of 3/8" diameter aluminum stock, cut to rough length in a band saw. After, it was brought over to a lathe where it was then cut to its finish length. One of its ends was then turned down to the proper size for the use of a ¼ -20 dye. This threading was for the connection to the plunger ball, which was purchased and then filed down to a point on one side for an easier interaction with the gripper. We then had to make the two grooves for the e-clips that would hold a washer in place on the plunger. This washer would be what ultimately compresses the spring into one of the spring plates. Using a groove tool, two grooves were placed to the left and right of the exact location of the washer, at the depth provided by the e-clip packaging.

The gripper is composed of a few different parts; two plates, two claws, two pins, and two screws used to mount a spring. The two plates were created like all of the other plates; cut to

rough size, cut to finish length with an end mill, found a point of origin with an edge finder, then drilled the corresponding holes. Once the required holes were drilled, semicircles were cut into the sides of the plates where it would be driven toward the ball, allowing enough space for the ball to move into the gripper's claws. The two claws were made from ½" x ½" stock, cut to rough length in a band saw and then milled to their correct shape using an end mill. Holes were drilled through the length of the claws similar to the holes through the lifting link; the holes on the claws would be a loose fit to the pins while the holes on the plates would be a tight fit, therefore holding the claws in place but allowing for their rotation. Two more holes were drilled toward the head of the claws for the locations of the two screws that would be used to mount the compression spring. Finally the heads of the claws were rounded down with a file to create a smoother interaction between the claws and the plunger ball.

4.2 Construction & Assembly

Figure 35: Picture of the Constructed Overall FIDO Prototype

4.2.1 Mechanical Systems

4.2.1.1 Base & Platform Assembly

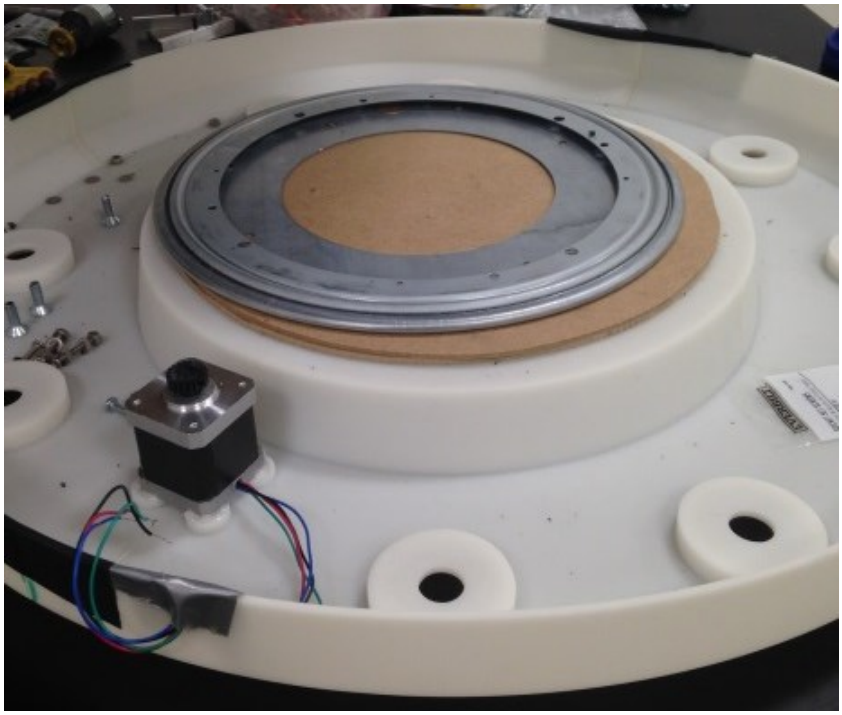

Figure 36: Picture of the Base

After the four pieces of the base were 3D printed and glued together the next step was to assemble the lazy susan bearing. Before bolting down the bearing we cut and placed a circular piece of fiberboard down to give extra clearance for the rotating platform. Also being attached to the base is the stepper motor that drives the yaw movement of FIDO. This stepper motor also has special 3D printed pinion gear that drives the rack on the underside of the rotating platform.

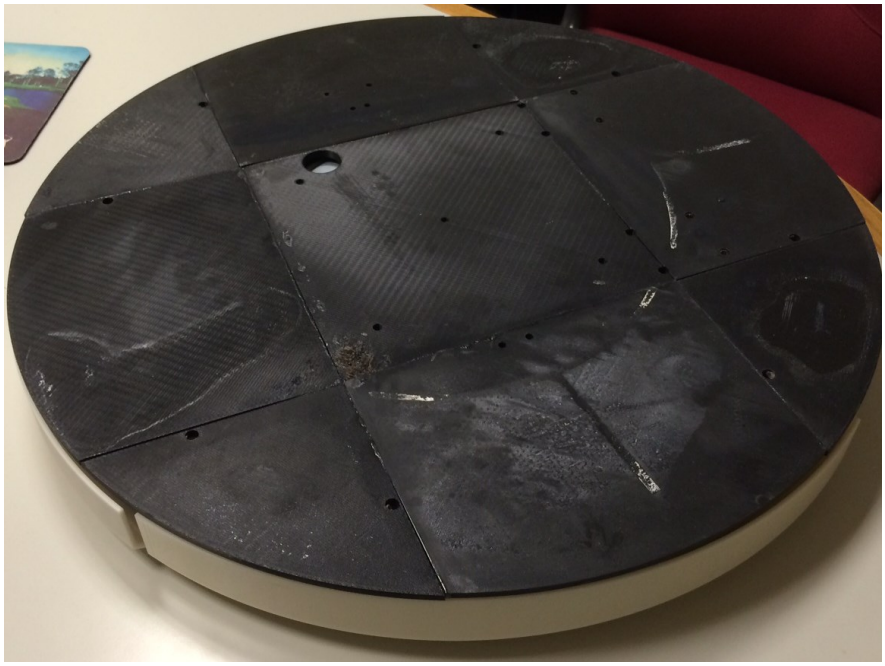

Figure 37: Picture of the Base and Rotating Platform

After the nine pieces of the rotating platform were glued together, we then attached it to the lazy susan bearing aligning the rack on the rotating platform and the pinion gear.

4.2.1.2 Lifting Mechanism

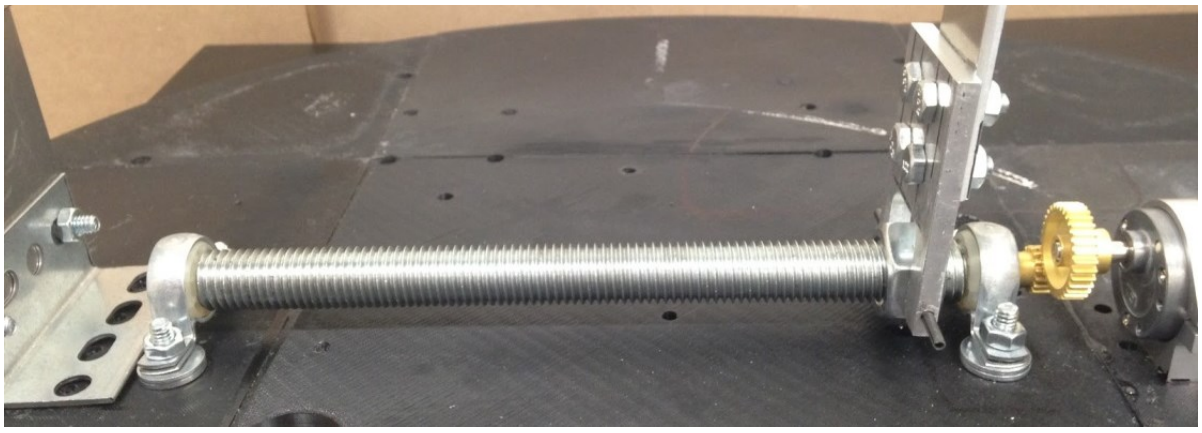

Figure 38: Overall Lifting Mechanism Picture

The lifting mechanism consists of two ½" pillow block bearings, threaded rod that was machined to fit the pillow blocks and motor, a DC motor with a bracket, gears and the lifting link

assembly. The lifting link assembly attaches to the threaded rod via a nut which attaches to the lifting link through two tight fit pins. The way that the lifting assembly works is that when the geared DC motor is activated it turns the threaded rod which moves the ¾" nut which changes the angle of the launching plate.

To start the construction of the lifting mechanism we first needed to find the locations of where the pillow block bearings need to be bolted down. To accomplish this we took the threaded rod and bearing assembly and then placed it as close to the vertical plate's bracket as possible. After finding this location we drilled holes in the platform and then bolted down the bracket closest to the vertical plate. Before assembling the threaded rod with the two bearings, we needed to create the lifting arm assembly.

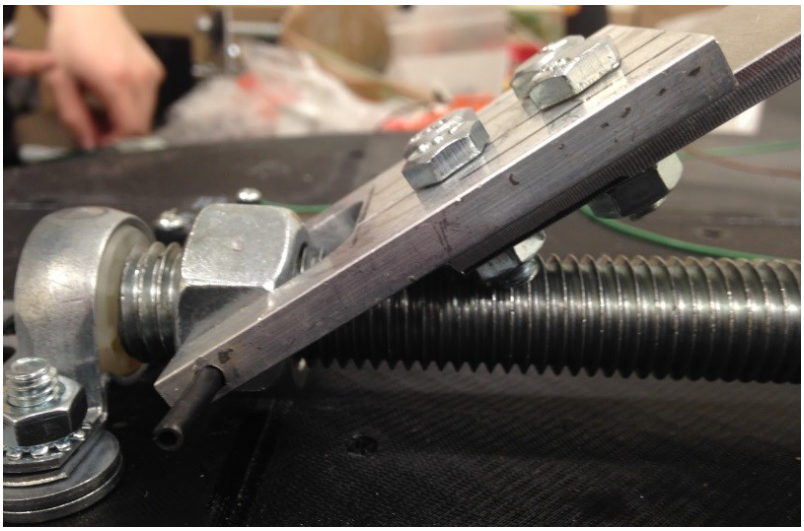

Figure 39: Picture showing the bearing closest to vertical plate

To construct the lifting link we took two pieces of machined aluminum and then bolted them together. After creating the lifting link we then attached the ¾" nut through two pins that have a tight fit in the nut yet loose fit on the arm so that it can rotate. Since the lifting link assembly was finished we attached it to the threaded rod through the nut. After screwing in the

lifting link assembly we then put the threaded rod into both bearings and secured them to the rotating platform.

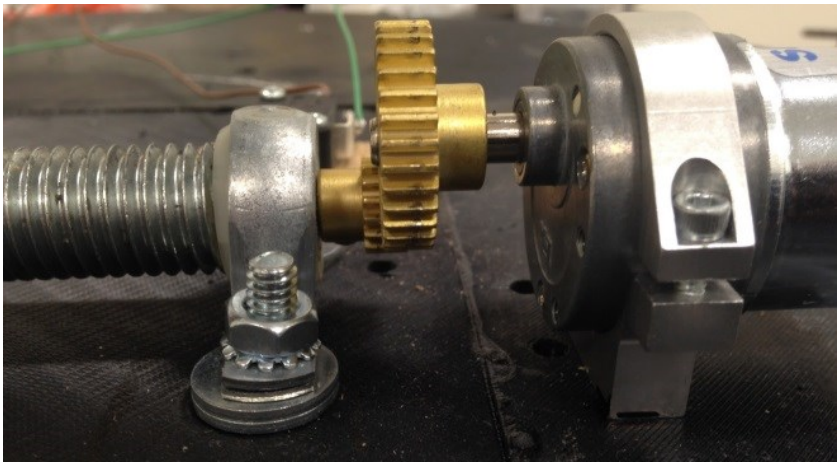

Figure 40: Picture showing the Lifting Mechanism's connection to a DC Motor

After finishing the lifting mechanism, we attached a gear to the threaded rod through a set screw. The geared DC motor that controls the threaded rod needed to be secured to the rotating base. In order to find the best location to do this we meshed the gears together and then marked where the motor was positioned. After finding this location we drilled holes in the platform and then screwed the motor into place.

4.2.1.3 Plunger and Spring Assembly

The plunger and spring assembly consists of two machined spring plates, four L-brackets, a compression spring, machined aluminum rod (plunger), 1" threaded aluminum ball, two e-clips, various washers, nuts and bolts.

The plunger assembly is simple, attach on of the e-clips to the plunger, then place a washer on top of the clip and then attach the second e-clip on the other side of the washer so that it cannot move.

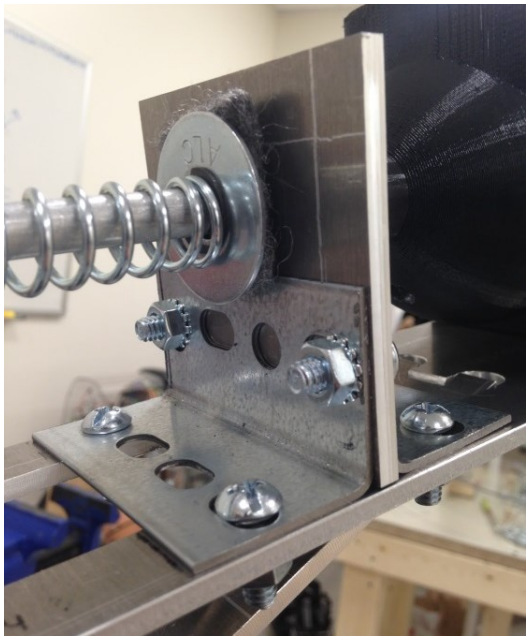

Figure 41: Picture of the Spring and Plunger Assembly

In order to start construction of this assembly, we needed to find the location of where to put the two spring plates and then mark where their corresponding L-brackets would go. After finding the L-bracket locations we drilled holes in the launching plate to secure the brackets. After the two brackets closest to the firing tube are bolted to the launching plate, one of the spring plates was inserted between them and was then bolted in place. Before the final spring plate was bolted in we needed to insert the plunger into the first vertical plate and then put the spring on the plunger.

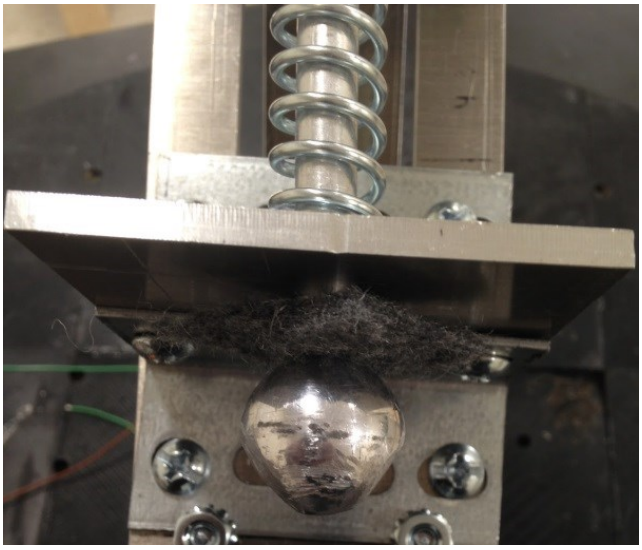

Figure 42: Picture showing the final vertical plate and threaded ball

In a similar fashion, the other set of brackets were bolted down as well as the spring plate being bolted in. While inserting the final spring plate, we had to put the plunger through it. After having put the plunger in its final position, we screwed the aluminum ball onto the plunger.

After realizing that there was a lot of force being exerted on each of the spring plates we added felt dampers in front of the aluminum ball and also in between the washer and the spring plate nearest to the launching tube.

4.2.1.4 Claw Mechanism

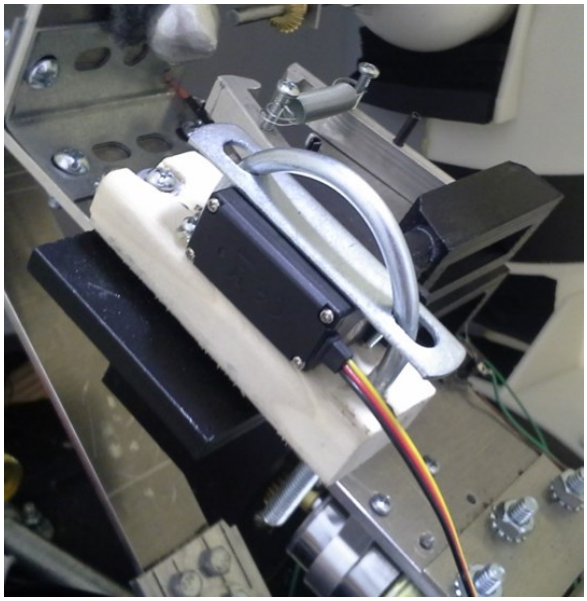

Figure 43: Picture of Overall Claw Mechanism

The Claw Mechanism consists of three sub-assemblies, the claw, the threaded rod and the moving bracket. In Figure 43, you can see all of the sub-assemblies combined to create the overall Claw Mechanism. The way in which this mechanism works is that a high torque DC motor on the undercarriage of the launching plate drives a threaded rod in the same way as the lifting mechanism. The Claw Mechanism then moves towards the aluminum ball and since the shape of the claw arms is angled the ball opens the claw and then once all the way in the tension spring closes the claw. After the claw is closed the servo motor activates and rotates down a locking block that keeps the claw closed as the entire Claw mechanism pulls the spring back.

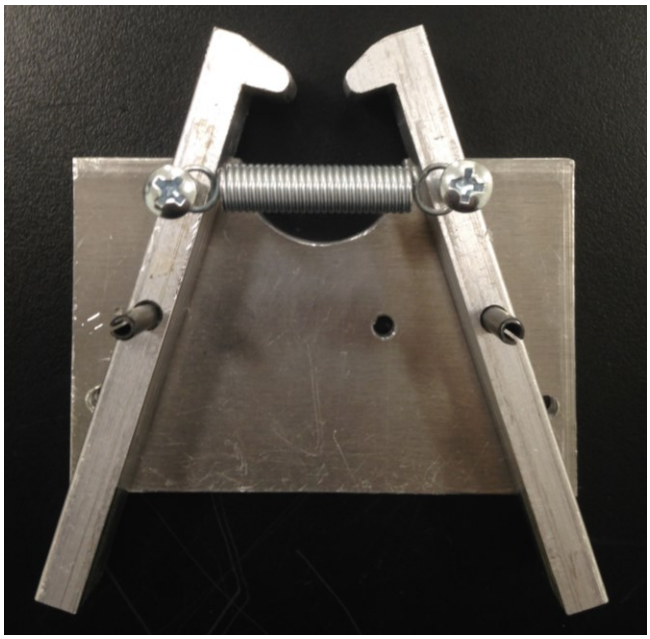

Figure 44: Picture of the Claw sub-assembly

The first sub-assembly that was created was the actual claw. After the aluminum pieces were machined to the desired shapes, there were some modifications that needed to be made. We drilled holes in the claw arms so that we could put in the tension spring. The first step in assembling the claw was to pin the claw arms into place on the claw base. After the arms were in place we found the locations were we wanted to put the limit bolts, the holes are seen in Figure 44 on the lower section of the claw base on the outside of the claw arms. These limit bolts have two functions, the first is to limit the claw movement and the second is to secure the claw to the 3D printed mount that moves on the threaded rod.

The next sub assembly that needed to be created was the moving bracket assembly. The moving bracket assembly starts with the 3D printed claw mount. In the same way as the lifting mechanism, we used two pins to secure a ¾" nut to the mount for the threaded rod to go through. Next the claw was bolted down to the claw mount. Finally the servo motor was mounted onto a side section of the claw mount using a U-bolt and 1" section of a 2x4. Once the servo was in

place we attached the locking block in place using gorilla glue to attach it to the servo and on the other side of the block we attached a pin to ensure smooth an even rotation.

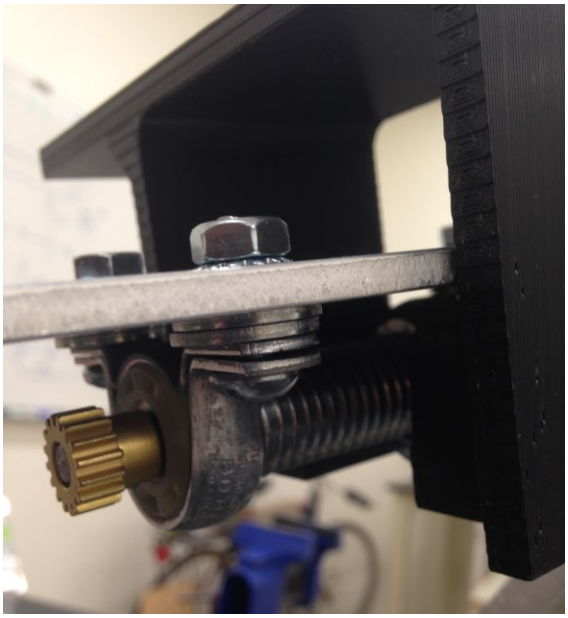

Figure 45: Picture of Threaded rod for the Claw movement

After the moving bracket assembly was create the only thing left from completing the overall Claw Mechanism was the threaded rod assembly. In the same way as the lifting mechanism, we found the locations of where the two pillow block bearings needed to be located so that the claw could grab the aluminum ball and be able to pull it back two inches. After finding the locations we then drilled holes and then bolted down the bearing closest to the spring. We then inserted the threaded rod, put on the moving bracket assembly and then attached the final bearing. Finally we found the correct location of where the DC motor needed to go and screwed it into the launching plate.

Once we started to test the Claw Mechanism we realized there were a few changes that needed to be made. After attempting to pull the spring back, the DC motor stalled out and the moving bracket assembly tilted down due to the stress from the spring. To fix this we bought a

higher torque DC motor and added a stability block underneath the moving bracket. Both of these fixes allowed for a smoother and more successful pullback of the spring.

Other changes that needed to be made were changing the spring, and re-shaping the aluminum ball. The original spring that we purchased was rated at 40 pounds per inch of pullback, we quickly realized that this far exceeded what our design and budget could handle and we purchased a new spring with half of the spring stiffness as the original. This new spring allowed for a much easier pullback while still being able to fire a tennis ball a moderate distance. Another change that allowed for ease of strain on our DC motors was reshaping the aluminum ball. When the perfectly spherical ball attempted to open the claw on its own, it ended up just pushing directly onto the entire claw mechanism and put serious stress on the DC motor. The easiest fix for this was to grind the aluminum ball into a cone shape so that its pointy tip would be able to spread the claw arms apart with ease.

4.2.1.5 Treat Dispenser Assembly

The treat dispenser assembly is made up of 2 inch PVC pipe parts and a small steel auger. To create the pipe shape that we wanted we used two 90° elbows that were connected with a 1.5” piece of PVC pipe. This allowed for one end of each elbow to abut against each other for support while maintain the shortest assembled length possible. A hole was drilled through the end of each elbow to allow the shaft of the auger to pass through once it was placed inside. The auger that we used was 1.25” in diameter and was originally 18” in length. We cut it down to the desired length and secured it inside the PVC piping. The shaft on the motor end of the auger was left long to allow adjustment for the motor attachment. It was also let to stick out .25” in the

front of the assembly to allow for a shaft collar to be mounted onto the front to hold the auger in place even if the treats were to jam.

Figure 46: Picture of Overall Treat Dispenser Assembly

To mount the motor to the treat dispenser assembly, we used L-brackets and a plastic plate to both stabilize the shaft of the auger and to mount the motor. Because the horizontal space devoted to the treat dispenser was very limited, we used bevel gears to go between the motor and the auger shaft. The auger shaft was left a little long and the motor mounted low enough that both bevel gears had room to be moved for any fine tuning adjustments that might have to be made in the future.

A short PVC pipe was used to connect the top of the treat dispenser to the bottom of the hopper. The only other support of the dispenser assembly when mounted in the unit comes from the hole in the cover of the unit that it passes through. Any extra spaces between the cover and the dispenser were filled with foam to provide a solid fit. More foam was cut and applied to the front of the unit to make it more waterproof and give it a finished look.

Figure 47: Treat Dispenser Installed (External View)

4.2.2 Electrical Systems

4.2.2.1 Power System

The battery system we chose to power the system was the Milwaukee 12v 3amp hour battery and charging base. In order to pull power off the battery, we removed the casing of the charger and soldered wires onto the positive and negative terminals on the circuit board. This wire was then run back through the case to attach into the rest of the system.

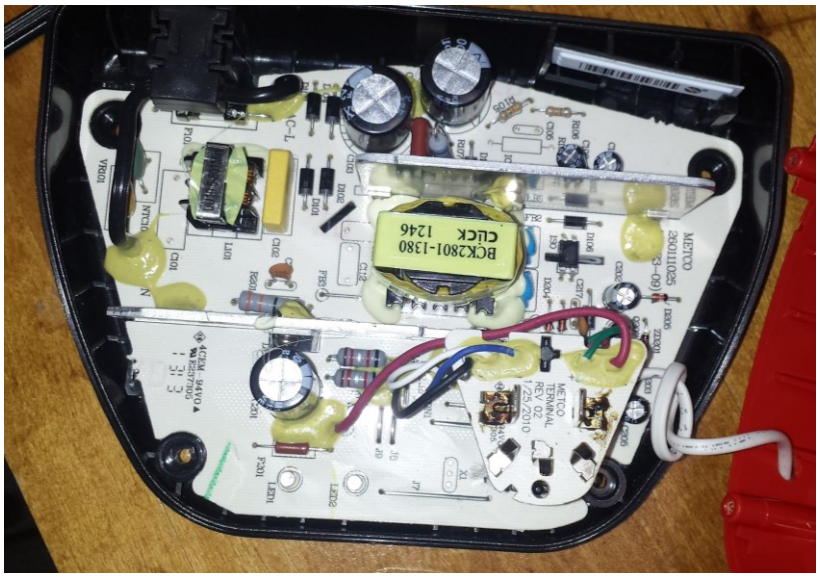

Figure 48: Power System Overview (Top View)

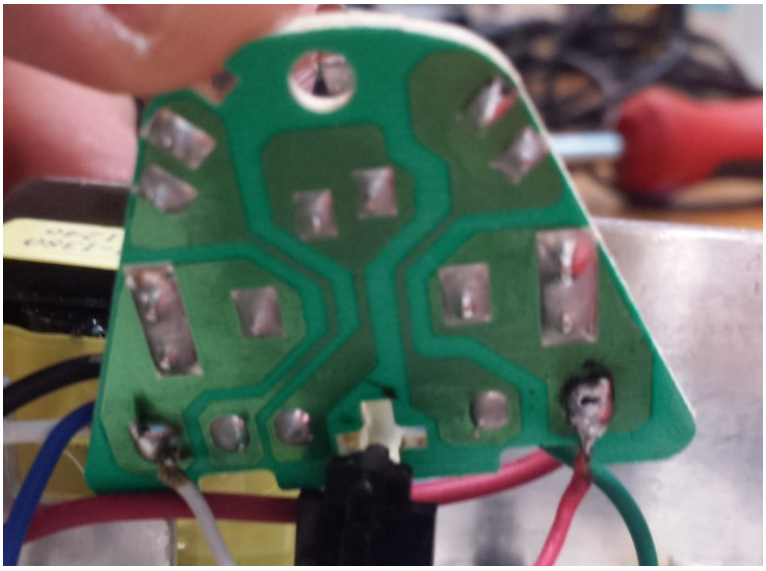

Figure 49: Power System Wiring Modification

To provide the charging base with power once it was encased inside the outer shell of the device, we screwed the IEC 320 C6 connector to the outer edge of the rotating base. Once it was secured, the cord could be attached to the unit for charging and removed for use. The unit can also operate while the cord is attached if the user wanted to charge it while it was in use.

Figure 50: Attached IEC 320 C6 Connector

Figure 51: Cord Plugged in to Charging Port

4.2.2.2 Computer System

The aforementioned wires were plugged into a solderless breadboard, which then provided power to all the motors, switches, sensors, and devices. This included the Pi and the screen as well. We used 22-gauge stranded wire as a standard means of connecting the switches and motors to the solderless board; however, the strands frayed unless they were soldered, at which point the leads were too wide to fit into the holes on the breadboard. As a solution, male-

male test leads designed for used in the solderless boards were soldered to the strands, and the end of the leads fit into the boards. A breakout cable (called a Pi Cobbler) was used to connect the GPIO pins on the Pi to the breadboard. The full wiring documentation, showing the connections between the breadboards and the Pi, is in Appendix C: Wiring Diagram.

4.3 Testing

The purpose of testing and analyzing our product is to ensure that our prototype satisfies as many product specifications as possible and functions in the appropriate manner. In order to prove that the prototype can actually preform to the specifications, we have constructed and conducted multiple tests.

The design specifications that are being tested are:

- FIDO shall launch balls
    - Range of Launching Angle: 5° - 45°
    - Range of Launching Distance: 5 - 50 feet
- The system shall incorporate a teaching mechanism
    - The system shall release treats
    - The system should give vocal encouragement
- FIDO shall operate with a rechargeable battery
- FIDO shall operate autonomously
    - FIDO shall have a means of telling itself to release the ball automatically
    - Motion Sensor, when pet is close to system it will not throw ball
    - Switch, every time the ball is brought back to the system by the pet the ball will hit a switch and the ball will instantly be thrown again

- The casing shall cover all moving parts
- The casing shall be durable and be able to withstand pet bites and scratches
- The system shall operate both indoors and outdoors
- FIDO's form factor shall be one that is easy to transport around the home.
- Fido's industrial design should accommodate for a comfortable one-hand carry that allows easy transportation around the house.
- FIDO's power supply should be built into the enclosure for easy placement and transportation around the home.

4.3.1 Test Guidelines

After selecting which design specifications fully defined FIDO, we proceeded to create the following test guidelines to successfully prove that our prototype proved the concept of what we were trying to create.

4.3.1.1 Launching Tests

Design Specifications

FIDO shall launch balls

a. **Range of Launching Angle: 5° - 45°**

b. **Range of Launching Distance: 5 - 50 feet**

In order to provide sufficient evidence that this design specification is satisfied, two tests need to be conducted.

1. What is range of motion of the vertical angle?
    a. Min
    b. Max

c. How long does it take to lift the firing tube to get from the minimum to the maximum angle?
d. How long does it take to lower the firing tube to from the maximum to the minimum angle?

2. What is the range of the launching distance?
    a. Min
    b. Max
    c. How long does it take to grab the firing ball and lock the claw in place?
    d. How long does it take to bull the firing ball back to the maximum launching power?

The first test can be simply conducted using a level that shows angles and then measuring the minimum and maximum angles achieved from using the lifting mechanism. While conducting the first test time how long it takes to raise and lower the angle. To efficiently conduct the second test, launch ball at minimum and maximum power at the minimum and maximum angles. While conducting the launching tests time how long it takes to grab and then pull back the firing ball.

In order to provide clear and accurate results, we will conduct the first test twice (4 results total), and the second test 5 times (ten results total).

4.3.1.2 Teaching Mechanism Tests

Design Specifications

The system shall incorporate a teaching mechanism

a. The system shall release treats

b. The system should give vocal encouragement

In order to provide sufficient evidence that this design specification is satisfied, two tests need to be conducted.

1. Can the Treat Dispensing Mechanism dispense treats?
    a. How long does it take to dispense one treat?
    b. How many treats are dispensed at a time?
2. Is there a Vocal Encouragement system?
    a. Can sounds be recorded?
    b. Can the recorded sounds be played back?

To conduct test one, place ten treats in the treat hopper, then start the auger motor and wait until the first treat is dispensed then instantly stop the motor. Take note of how many treats actually dispensed and the amount of time that it took for this to happen. The second test is simple, using the code written, try to record and then play back a voice recording.

In order to provide clear and accurate results, we will conduct the first test 10 times, and the second test 5 times.

4.3.1.3 Power Supply Testing

Design Specifications

a. **FIDO shall operate with a rechargeable battery**
b. **FIDO's power supply should be built into the enclosure for easy placement and transportation around the home.**

In order to provide sufficient evidence that these design specification is satisfied, one test needs to be conducted.

1. Does the battery recharge?
    a. Can FIDO operate off of the battery?
2. Is the power supply housed within the enclosure?

This test is conducted simply from charging the battery and then trying to run FIDO off of the battery. The second specification mentioned doesn't need a test rather it just needs to be checked off that the power supply is enclosed within FIDO.

In order to provide clear and accurate results, we will conduct this test twice.

4.3.1.4 Control Systems Testing

Design Specifications

FIDO shall operate autonomously

a. **FIDO shall have a means of telling itself to release the ball automatically**

b. Motion Sensor, when pet is close to system it will not throw ball

c. Switch, every time the ball is brought back to the system by the pet the ball will hit a switch and the ball will instantly be thrown again

1. In order to provide sufficient evidence that these design specification is satisfied, three tests need to be conducted. Can FIDO launch a ball automatically?
2. Does the Motion Sensor stop launches?
    a. Measure the distance at which the launching mechanism will engage
3. Does the pressure switch engage when a tennis ball is on it?

The first test can be conducted by having the Raspberry Pi conduct a firing sequence. The second test will be completed by standing in front of the senor and slowly moving away from it until FIDO activates and starts its firing sequence. After finding the spot that FIDO engages, measure the distance from the base to the point of engagement. The third test will be conducted by dropping a tennis ball into FIDO's hopper and then seeing if the pressure switch is triggered by the tennis ball landing on it.

In order to provide clear and accurate results, we will conduct the first, second and third tests all 5 times each.

4.3.1.5 Durability Testing

Design Specifications

Durability

a. **The casing shall cover all moving parts**
b. **The casing shall be durable and be able to withstand pet bites and scratches**
c. **The system shall operate both indoors and outdoors**

In order to provide sufficient evidence that these design specification is satisfied, two tests need to be conducted.

1. When the exterior shell is fully attached does it cover all of FIDO's moving components?
2. Is the exterior shell durable enough to withstand play with pets?
3. Can the system operate both indoors and outdoors?

The second question will answered by having a pet test FIDO out. The third question will be completed by conducting launching tests indoors and outdoors.

4.3.1.6 Ergonomic Testing

Design Specifications

Ergonomics

a. **Fido's industrial design should accommodate for a comfortable one-hand carry that allows easy transportation around the house.**
b. **FIDO's form factor shall be one that is easy to transport around the home.**

In order to provide sufficient evidence that these design specification is satisfied, two tests need to be conducted.

1. Can FIDO be carried with one hand, considering the weight?
2. Is FIDO challenging to move from one location to another, considering the shape and size?

The first test has two parts to it, first FIDO will be weighed and then it will be picked up several times using only one hand to see if it is reasonable to carry. The second test will be conducted by carrying the fully assembled FIDO from one designated location to another, approximately 50 feet, taking notes about how easy or challenging it was to carry.

In order to provide clear and accurate results, we will conduct the first and second test both 5 time. After one person completes these 10 tests, another person will conduct the same exact tests to confirm the results.

5 Results

5.1 Testing results

5.1.1 Launching Test Results

The first test conducted for the launching system was the minimum and maximum angle tests. The maximum angle that FIDO can achieve is 45°, while the minimum is 17°. It took 2 minutes and 42 seconds to go from the minimum to up to the maximum angle and 2 minutes and 34 seconds to go from the maximum down to the minimum angle.

The next test was the distance test. The distances measured were, maximum launching distances for 17 ° and 45° as well as the absolute minimum launching distance. The absolute minimum launching distance was measured by firing a tennis ball just hard enough so that it would exit the launching tube. The absolute minimum distance FIDO can launch a tennis ball is 8 inches. The maximum launching distances were measured from the base of FIDO to the exact spot that the ball landed, meaning that the distance measured do not include roll. The graph below shows the outcomes from the 5 launch trials for 17° and 45°. The average maximum distance at 17° is 9.43 feet and at 45° it is 17.4 feet, again, this is not including roll.

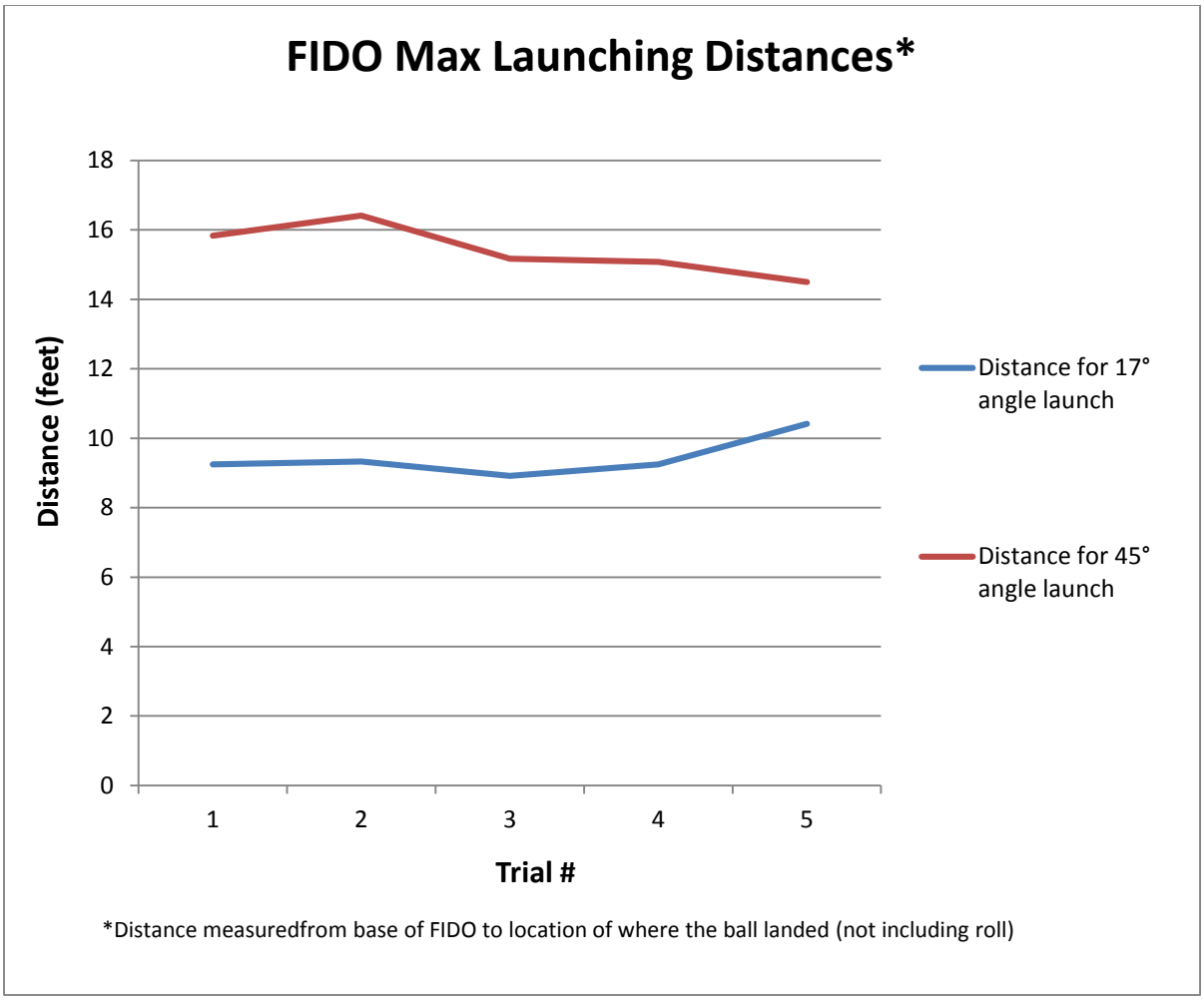

Figure 52: Max Launching Distance Graph

While we were conducting the distance tests we timed the pull back a ball grab times. It took 2 minutes and 18 seconds for the grabbing mechanism to grab the firing ball and lock the claw into place. After the claw was locked into place, it took 2 minutes and 50 seconds to pull the ball back to the maximum distance.

5.1.2 Teaching Mechanism Testing

The treat dispenser system can be used with hard treats ranging from .5-.75 inches at the largest point. The system does much better with treats that are close to spherical but other shaped treats could be used as long as they slide well against each other. Because it is necessary for the

treats to move by one another, soft treats that stick together do not work as they jam almost instantly in the auger. Once the treats are loaded in the hopper it takes 10-20 seconds depending on the size and ease of movement of the treats to fully load the auger. Once the first treat has been dispensed, all other treats can be dispensed within one to two seconds of starting the motor. All treats except for very small ones can be dispensed one at a time without any trouble.

The vocal encouragement system was a tertiary feature, designed to be a luxury add-on. Unfortunately, complications from the Raspberry Pi prevented the system from ever working. The Raspberry Pi can direct sound through either the 3.5mm audio in/out port, or through the HDMI cable. FIDO utilized the HDMI port to go to the touchscreen, but did not transmit sound over HDMI. As such, the Pi was configured to force sound out the 3.5mm jack. The USB speakers then picked up and redirected the signal. However, this prevented the 3.5mm jack from then being used to provide the audio input from the microphone, as the system would have been confused.

The audio playback would have worked, theoretically. Programs and libraries exist, both for the Pi and for Python, respectively, that allow sound files to be played. However, this was never implemented nor tested due to time constraints.

Finally, the recorded sound would be a great system; however, given the problems with the hardware input of an audio signal, as well as the digital coding problems of naming the file such that the Python code can read it, overriding commands to play the default included sound files, and more, this system was never implemented nor tested.

5.1.3 Power Supply Testing

The battery and base charger is securely mounted inside the casing of FIDO. The charging cord can be attached to recharge the battery and detached for use so keep any dogs from chewing on it. The battery takes about 30 minutes to recharge once it has been completely discharged; however, discharge time is highly varied by frequency of use by the dog and whether or not the motors to change position are activated.

5.1.4 Control Systems Testing

The PIR sensor purchased for FIDO never worked. It only read in a HIGH signal, indicating that the sensor is continuously seeing movement. This is the case even when the signal is fixed in place in an empty room and allowed to calibrate for a period of 30 minutes or more. As a result of this, the FIDO code could never fire as written; waiting for the PIR sensor to read a low signal for 3 seconds continuously would never occur. However, when the PIR sensor was bypassed, the code ran smoothly. The hardware did not interface with the code, and hooking up the Pi to the power system resulted in a microcomputer that did not turn on and motors and ran without computer input. The mechanics of the Pi system were tested and functioned. The sensors of the Pi were tested and functioned (excepting the PIR sensor). Unfortunately, the joining of the two did not work as intended.

The force-sensing resistor worked perfectly. When the ball was on it in the launch tube, the resistance value decreased from effective infinite to 25 kilo Ohms. A voltage divider utilizing the force sensing resistor and a fixed resistor sent a HIGH signal to the Pi.

5.1.5 Durability Testing

The exterior shell is in four parts and when attached to the base it successfully covers all moving parts. After testing FIDO with a dog, the exterior shell was strong and durable enough to withstand all of the playing from a dog. The exterior shell was design with no rough edges so that dogs wouldn't be able to bit the shell; the only exception is the step. Test was not done for an extended period of time so we do not know if the step would be subject to bites. FIDO can operate both indoors and outdoors due to its variable launching powers and angles.

5.1.6 Ergonomic Testing

Our current design of FIDO must be carried with two hands, although we suggest moving it with two people. In its current state FIDO is large and would be challenging to move for one person. We would like to have made FIDO smaller and easier to transport but in order to fit all of the mechanical and electrical systems.

6 Conclusions and Recommendations

Our prototype demonstrated the concept, albeit with needed areas of improvement. FIDO's large footprint and heavy weight exceeded our sizing constraints. It is unlikely small dogs would be able to return the ball, or that an average person could easily transport it on their own. The pullback system had numerous design changes and became more complicated than originally planned. The distances, angles, and the speed of the system achieved were less than those originally forecasted. Due to changes that needed to be made in the spring and the design of the interaction between the lead screw and the lifting link, our team lost distances and angles in the transition from theory to practice. The software, albeit components individually tested, was not fully functional as a stand-alone final product. The code for a full calibration, launch, and return was not fully completed.

FIDO is about two and a half feet cubed, and the team needed two people to carry it due to its weight. The main reason for its size was the launching mechanism. All of its components were in plane with each other. That meant that we needed to raise the platform to fit the lifting system directly underneath the angled plate. If a different lifting mechanism is chosen, for example a pulley system above the angled plate, the amount of space required underneath could be reduced and the angled plate could be lowered. Also, if the hopper and feeder tube were off to the side or in general a more rigid system, the space above the lifting mechanism could be reduced. We also decided to rotate the system from the very end of the angled plate. Rotating it from the end rather than about some middle point meant that the radius of the system was a lot larger and needed more vertical space to achieve greater angles.

Reducing the footprint and the amount of metal would also help reduce the weight. We designed the base plate to be circular for aesthetic reasons and at a diameter to ensure enough

floor space for all of the systems. Our team also chose aluminum or steel as a material for some of the plates or brackets. In further iterations of design, those could likely be switched to a plastic. Our recommendations to reduce the size and weight are to change the point where the system rotates, move some of the lifting systems out of plane or change it to a pulley system, create a more compact footprint, and to change the aluminum plates to plastic. These changes can produce a much smaller system and therefore require less material, reducing both the size and the weight simultaneously.

Our final design for the pullback mechanism was one that could be easily accomplished, but ultimately did not work to its fullest extent. It required three separate systems, and even though each one was simplistic, the combination remained problematic. Using the lead screw to move the gripper was slow, and despite the change in motor, was unable to fully compress the spring. The bracket that translated the lead screw movement to the gripper was not sturdy enough and could not handle the force of the spring. It was allowing the gripper to be tilted forward, losing some of the potential pullback distance. The servomotor used to lock the grip on the plunger was neither strong enough nor fast enough to be an efficient lock and release method.

Our team's recommendations for improving the pullback system are three different courses of action; make the necessary improvements to the current design, return to the original pullback design, or come up with an entirely new one. Stronger motors for the locking mechanism as well as the lead screw rotation would also be required for the existing design. The original design of the vertically mounted spiraling fork could be analyzed further. An entirely new design could allow for better spatial planning within the shell, which could also reduce the size of FIDO as a whole. Revisiting the launching mechanism design matrix could provide better

insight on new methods, or entirely new launching mechanisms could be added that our team did not research.

While attempting to accomplish all of the physical and mechanical goals of this project, the electronics fell behind. With many redesigns to the rest of the system, our team did not have the time to focus on the coding of the system. We were able to test the motors individually, but we never attempted to produce a full launch, return, and reward cycle. Our team's recommendation for improving the electronics of FIDO is to continue working with what we currently have. Significant code has been developed, the sensors could work with more testing, and the electronics for the speakers and the touch screen have already been purchased. With more time to understand these components, this can become a fully working electrical system. Once it is completely tested and functional, a more permanent solution to the wiring than our breadboard connections would be implemented.

Our team created FIDO in its current state, from the ground up. We created an entire physical system based on an idea. We took the first attempt at creating a solution, while also providing enough groundwork for another team to eventually improve on what we produced. With our recommendations, another iteration of design can be a significant improvement on what we accomplished. FIDO has the capabilities to be a successful dog entertainment center and, with more prototypes and iterations of design, become a patent-viable product and enter the market.

7 References

Apartments.com. (2014) “Survey Reveals Pet Ownership Among Renters at All-time High” Retrieved February 3, 2015, from http://corporate.apartments.com/press-room/apartments-com-survey-reveals-pet-ownership-among-renters-at-all-time-high/

“Bata 2”. (2015). Retrieved April 29, 2015, from http://www.pitchingmachinepro.com/Bata-2-Baseball-Pitching-Machine-p/bata-201.htm

Brain, Marshall. (2000). "How Stereolithography 3-D Layering Works" HowStuffWorks.com. Retrieved February 19, 2015, from http://computer.howstuffworks.com/stereolith.htm

Bureau of Labor Services (BLS). (2014). “American Time Use Survey – 2013 Results” Retrieved February 3, 2015, from http://www.bls.gov/news.release/pdf/atus.pdf

Dugatkin LA. *Principles of Animal Behavior, 3rd Edition.* New York, NY: W.W. Norton & Company; 2014.

“The Electric Linear Actuator.” (2015). Retrieved April 29, 2015, from http://www.linak.com/about/?id3=2316

GoDogGo Remote Fetch. (2014). Retrieved April 29, 2015, from http://www.godoggoinc.com/

Greenebaum JB. Training dogs and training humans: Symbolic interaction and dog training. *Anthrozoos.* 2010. http://go.galegroup.com/ps/i.do?id=GALE%7CA236163954&v=2.1&u=mlin_c_worpoly&it=r&p=AONE&sw=w&asid=37b8423386c8a17979e4f44ce8fbd8cf.

Hiby EF, Rooney NJ, Bradshaw JWS. Dog training methods: Their use, effectiveness, and interaction with behavior and welfare. *Animal Welfaire*. 2004, 13: 63-69. http://www.researchgate.net/publication/261106650_Dog_training_methods_their_use_effectiveness_and_interaction_with_behaviour_and_welfare.

Hiemenz, Joe. (2008). "3D Printing with FDM" Stratasys®, Inc. http://www.stratasys.com/~/media/Main/Secure/White%20Papers/Rebranded/SSYS_WP_3d_printing_with_fdm.pdf

Hughes A, Drury B, Knovel Electronics &, Semiconductors Library - Academic Collection, Knovel Electrical &, Power Engineering Library - Academic Collection. Electric Motors and Drives: Fundamentals, Types and Applications. San Diego: Newnes [Imprint]; 2013

iFetch Pet Toy - Automatic Ball Launcher. (2014). Retrieved April 29, 2015, from http://goifetch.com

Jacobs D. Why are dogs so obsessed with endlessly playing fetch? The Huffington Post. http://www.huffingtonpost.com/quora/why-are-dogs-so-obsessed-_b_6407164.html. Published January 5, 2015. Coppinger D, Coppinger L. *Dogs: A New Understanding of Canine Origin, Behavior, and Evolution.* Chicago, IL: The University of Chicago Press; 2001.

Jensen, Per. Behavioural Biology of Dogs. Cambridge, MA, USA: CABI Publishing, 2007. ProQuest ebrary. Web. 19 April 2015. Copyright © 2007. CABI Publishing. All rights reserved.

Lindsay S. Handbook of Applied Dog Behavior and Training, Adaptation and Learning, Volume 1: Principles of Behavior Adaptation and Learning. Wiley-Blackwell; 2008

Lucky Launcher - RRT Launcher Home. (2014). Retrieved April 29, 2015, from http://rrtlauncher.com/

Mintel. (2012) "Americas Pet Owners – US – 2012" Retrieved February 3, 2015, from http://btoellner.typepad.com/kcdogblog/2012/11/us-pet-ownership-statistical-breakdown.html

Nave, C. (1998). Elastic Potential Energy. Retrieved April 28, 2015, from http://hyperphysics.phy-astr.gsu.edu/hbase/pespr.html

Orsini, L. (2014). "Arduino Vs. Raspberry Pi: Which Is The Right DIY Platform For You?" Retrieved April 29, 2015, from http://readwrite.com/2014/05/07/arduino-vs-raspberry-pi-projects-diy-platform

Parr, E. (1998). Industrial control handbook (3rd, ed.). Oxford: NN.

"Tennis Ball Specifications." (2011). Retrieved April 28, 2015, from http://www.usta.com/2011_tennis_ball_specifications/

Williams, B. (2000). The Lever. Retrieved April 28, 2015, from http://www.ohio.edu/people/williar4/html/haped/nasa/simpmach/lever.htm

Wilson, T. (2005). Pneumatic - How Spud Guns Work. Retrieved April 28, 2015, from http://home.howstuffworks.com/spud-gun3.htm

8 Appendix A: Pi Code

```
# Fido Catcher v1.0
# Michael Goldman
# Hannah Brown
# 3/6/2015

"""
CURRRENT STATUS:
    Systems starts up, calibrates M1, M2, M4.
    Determines max, neutral positions of each.
    Adds e-stop capabilities only after calibration.
    (Calibration disables e-stop trigger to prevent infinite loop.)
    Functions defined to:
        Move DC, Stepper motors forward, backward, stop.
        Move Servo to any angle within range of motion.
        Move DC, Stepper motors to any position within range of motion
    Functions alphabetized for ease of reading.
    Functions ask for M1, M2, M4 input.
    Offers randomization for M1 (lateral angle)
    Moves system into position.
"""

"""
STILL TO DO:
    Gosh Darn IR sensor.
"""

# Important! Servo needs timing functions only available on GPIO18 (Board
12).
# Motor 1 pin 3 is moved to pin 38 (free pin).
# Motor 5 enable is moved to pin 12.
# Pin 35 becomes free pin.


"""
Flowchart
    INITIALIZATION. Set up board, in/out pins,
    libraries, referenced variables, arrays.
    CALIBRATION. Test M1, M2, M4. Determine
    steps for M1 and seconds for M2 and M4.
    READY. Bring M1 and M2 to neutral positions.
    Remove edge detect to M4(?) Careful with
    edge detect on M4, since safest position
    for spring is at edge with no pull.
    AIM. Randomize angle (within limits). Move
    into position. Run clearance check.
    Pullback pinball handle to specified level.
    FIRE. Check for clear path. Release handle.
    Return to neutral.
"""
####SUBROUTINES REFERENCED IN MAIN PROGRAM####

def CalcPointSlope(x1, y1, m, x):
    """
```

```
    Given point, slope, and x-value, calculates y-value of line as float.
    """
    return float((m * (x - x1)) + y1)

def CalcSlope(x1, y1, x2, y2):
    """
    Given two points, calculates slope between them.
    """
    return (float(y2-y1) / (x2-x1))

def CalibrateDC(motor):
    """
    Calibrates given DC motor.
    Returns time for that motor's full range of motion.
    """
    lower = 0
    upper = 0

    # Removing event detect from limit switches.
    # These will be reinstated after calibration.
    # If the e-stops are hit after the startup calibration,
    # The EStop function will re-run calibration, and hitting
    # the limit switches in the calibration feature could trigger
    # the EStop function from the event detection again, causing
    # an infinite loop. L5 had no event detect (safest place for
    # spring pullback is at the limit switch), so no event is removed.
    GPIO.remove_event_detect(L3)
    GPIO.remove_event_detect(L4)
    GPIO.remove_event_detect(L6)

    if (motor == M2):
        lower = L3
        upper = L4
    if (motor == M4):
        lower = L5
        upper = L6

    #Move to an extreme
    while (GPIO.input(lower) == 0 and GPIO.input(upper) == 0):
        DCForward(motor)
    DCStop(motor)

    #Stay at the extreme, but not on the e-stop switch
    while (GPIO.input(lower) == 1 or GPIO.input(upper) == 1):
        DCBackward(motor)
    DCStop(motor)

    #Time moving to the other extreme
    start1 = time.time() #Start timer
    while (GPIO.input(lower) == 0 and GPIO.input(upper) == 0):
        DCBackward(motor)

    #Stay at the extreme, but not on the e-stop switch
    #Reduce time accordingly
    end1 = time.time() #End timer
    start2 = time.time() #Start timer for reduction
    while (GPIO.input(lower) == 1 or GPIO.input(upper) == 1):
```

```
        DCForward(motor)
    DCStop(motor)

    end2 = time.time() #End timer for reduction

    totaltime = (end1-start1)-(end2-start2) #Time for full movement

    #Set Neutral Position for Motor
    global M2NEUTRAL
    global M4NEUTRAL

    neut = (1.0*totaltime / 2)

    if (motor == M2):
        M2NEUTRAL = neut
    if (motor == M4):
        M4NEUTRAL = neut

    #Set Present Position for Motor
    global M2PRESENT
    global M4PRESENT

    if (motor == M2):
        M2PRESENT = (1.0*totaltime)
    if (motor == M4):
        M4PRESENT = (1.0*totaltime)

    #Move Motor to Neutral Position
    MoveDC(motor, neut)

    #Reduce time by limiting factor.
    return (totaltime*LIMFACT)

def CalibrateStepper(motor):
    """
    Calibrates given stepper motor.
    Returns ticks for that motor's range of motion.
    """
    count = 0

    # Removing event detect from limit switches.
    # These will be reinstated after calibration.
    # If the e-stops are hit after the startup calibration,
    # The EStop function will re-run calibration, and hitting
    # the limit switches in the calibration feature could trigger
    # the EStop function from the event detection again, causing
    # an infinite loop.
    GPIO.remove_event_detect(L1)
    GPIO.remove_event_detect(L2)

    #Move to an extreme
    while (GPIO.input(L1) == 0 and GPIO.input(L2) == 0):
        StepForward(motor)
    StepStop(motor)

    #Stay at the extreme, but not on the e-stop switch
    while (GPIO.input(L1) == 1 or GPIO.input(L2) == 1):
```

```
        StepBackward(motor)
    StepStop(motor)

    #Count ticks moving to the other extreme
    while (GPIO.input(L1) == 0 and GPIO.input(L2) == 0):
        StepBackward(motor)
        count = count + 1

    #Stay at the extreme, but not on the e-stop switch
    #Reduce count accordingly
    while (GPIO.input(L1) == 1 or GPIO.input(L2) == 1):
        StepForward(motor)
        count = count - 1
    StepStop(motor)

    #Set Neutral Position for Motor 1
    global M1NEUTRAL
    M1NEUTRAL = int((1.0*count / 2))

    #Set Present Position for Motor 1
    global M1PRESENT
    M1PRESENT = int(count)

    #Move Motor 1 to Neutral Position
    MoveStep(motor, M1NEUTRAL)

    #Reduce count by limiting factor.
    return int(1.0*count*LIMFACT)

def CountBalls():
    """
    Compares current thrown ball count against set limit.
    """

    if ballCount >= ballLimit:
        return True
    else:
        return False

def CountTime():
    """"
    Compares elapsed time against set limit.
    """

    if (time.time()-timeStart) >= timeLimit:
        return True
    else:
        return False

def DCBackward(motor):
    """
    Turns DC motor 'backward.'
    """
    GPIO.output(motor[0], True)
    GPIO.output(motor[1], False)
    GPIO.output(motor[2], True)
```

```
def DCForward(motor):
    """
    Turns DC motor 'forward.'
    """
    GPIO.output(motor[0], True)
    GPIO.output(motor[1], True)
    GPIO.output(motor[2], False)

def DCStop(motor):
    """
    Turns DC motor off.
    """
    GPIO.output(motor[0], False)
    GPIO.output(motor[1], False)
    GPIO.output(motor[2], False)

def EStop():
    """
    With GPIO event, safely stops all motors, re-runs calibration.
    """
    global M1STEPS
    global M2SCNDS
    global M4SCNDS

    StepStop(M1)
    DCStop(M2)
    DCStop(M3)
    # Removes tension on spring before stopping spring.
    MoveDC(M4, M4SCNDS)
    DCStop(M4)

    M1STEPS = CalibrateStepper(M1)
    M2SCNDS = CalibrateDC(M2)
    M4SCNDS = CalibrateDC(M4)

def InputPinsDOWN(pin, name):
    """
    Sets pin as pull-down resistor input for given pin.
    """
    GPIO.setup(pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
    print "Switch " + name + " set up."

def isClear():
    """
    Checks to see if the path of the ball is clear using the IR sensor.
    """
    if GPIO.input(P1) == 1:
        print "Path Blocked"
        return False
    else:
        return True

def MoveDC(motor, position):
    """
    Moves DC motor to specified position.
    """
    global M2PRESENT
```

```
    global M4PRESENT

    if (motor == M2):
        diff = position - M2PRESENT
    if (motor == M4):
        diff = position - M4PRESENT

    start = time.time()
    if diff > 0:
        while time.time()-start < diff:
            DCBackward(motor)
        DCStop(motor)
    elif diff < 0:
        while time.time()-start < abs(diff):
            DCForward(motor)
        DCStop(motor)
    else:
        DCStop(motor)

    if (motor == M2):
        M2PRESENT = position
    if (motor == M4):
        M4PRESENT = position

def MoveServo(angle):
    """
    Moves servo motor to specified angle if angle within range of servo
motion.
    """
    if angle >= 0 and angle <= 180:
        duty = float(angle) / 10.0 + 2.5
        # I don't understand the above code, but I found it on a website
        # and it worked in testing, so I kept it.
        # Original Code Credit: Simon Monk (RazzPiSampler.OReilly.com)
        pwm.ChangeDutyCycle(duty)

def MoveStep(motor, position):
    """
    Moves stepper motor to specified position.
    """
    global M1PRESENT
    diff = position - M1PRESENT

    if diff > 0:
        for i in range(0,diff):
            StepBackward(motor)
        StepStop(motor)
    elif diff < 0:
        for i in range(diff,0):
            StepForward(motor)
        StepStop(motor)
    else:
        StepStop(motor)

    M1PRESENT = position

def OutputPins(motor):
```

```
    """
    Sets pins as outputs for given array of pins.
    """
    for i in range(0,len(motor)-1):
        GPIO.setup(motor[i], GPIO.OUT)
        GPIO.output(motor[i], False)
    print "Motor " + motor[len(motor)-1] + " set up."

def SetAuger():
    """
    Sets the treat dispenser to be enabled or disabled.
    """

    while True:
        try:
            t = int(raw_input("Enable treat dispenser? 1 for 'yes', 0 for
'no': "))
            if t == 1 or t == 0:
                break
            print "Invalid entry. Please enter either 1 for 'yes', or 0 for
'no'."
        except ValueError:
            print "Invalid entry. Please enter either 1 for 'yes', or 0 for
'no'."

    # Above forces user entry of either 1 or 0.
    # If 1, enables auger. If 0, disables auger.

    if t:
        return True
    else:
        return False

def SetElevation():
    """
    Sets the value for the seconds for the user-given elevation angle.
    """

    while True:
            try:
                degree = float(raw_input("What angle should M2 be set to? "))
                if degree >= M2MIN and degree <= M2MAX:
                    break
                print "Angle must be between %d and %d degrees!" % (M2MIN,
M2MAX)
            except ValueError:
                print "Invalid entry. Numerical entry required."

    # Verification
    print "Elevation angle set to %d degrees." % degree

    # Define points for the equation of the form y = m(x-x1) + y1
    # This will relate x (user input degrees) and y (seconds)
    pt1x = float(M2MIN + M2MAX) / 2
    pt1y = M2NEUTRAL
    pt2x = M2MAX
    pt2y = M2SCNDS
```

```
    m = CalcSlope(pt1x, pt1y, pt2x, pt2y)

    time = CalcPointSlope(pt1x, pt1y, m, degree)

    return time

def SetFinish():
    """
    Defines conditions for full run of program.
    """
    global ballLimit
    global timeLimit

    # Queries user for end after X balls thrown or Y minutes elapsed.
    print "Program cannot run indefinitely. End condition must be set."
    print "Throw balls until ball count reached or until time limit elapsed?"

    while True:
        try:
            c = int(raw_input("End condition choice. 1 for 'ball', 0 for
'time': "))
            if c == 1 or c == 0:
                break
            print "Invalid entry. Please enter either 1 for 'ball', or 0 for
'time'."
        except ValueError:
            print "Invalid entry. Please enter either 1 for 'ball', or 0 for
'time'."

    # Above forces either 1 or 0. If 1, user chooses ball limit.
    # If 0, user chooses time limit.

    if c:
        while True:
            try:
                ballLimit = int(raw_input("Total number of balls to throw:
"))
                if ballLimit > 0:
                    break
                print "Number of balls must be greater than zero."
            except ValueError:
                print "Invalid entry. Numerical entry required."
        print "Ball limit set to %d balls." % ballLimit

    # Sets time in minutes
    else:
        while True:
            try:
                t = int(raw_input("Total time (in minutes) to run machine:
"))
                if t > 0:
                    break
                print "Time must be greater than zero."
            except ValueError:
                print "Invalid entry. Numerical entry required."
        print "Time limit set to %d minutes." % t
```

```
        # Time functions handle seconds, not minutes. Conversion required.
        timeLimit = t*60

def SetPullback():
    """
    Sets the value for the seconds for the user-given power level.
    """

    while True:
            try:
                power = float(raw_input("What percent of spring power should
be used? "))
                if power >= M4MIN and power <= M4MAX:
                    break
                print "Percentage must be between %d% and %d%!" % (M4MIN,
M4MAX)
            except ValueError:
                print "Invalid entry. Numerical entry required."

    # Verification
    print "Spring power set to %d degrees." % power

    # Define points for the equation of the form y = m(x-x1) + y1
    # This will relate x (user input percentage) and y (seconds)
    pt1x = float(M4MIN + M4MAX) / 2
    pt1y = M4NEUTRAL
    pt2x = M4MAX
    pt2y = M4SCNDS

    m = CalcSlope(pt1x, pt1y, pt2x, pt2y)

    time = CalcPointSlope(pt1x, pt1y, m, power)

    return time

def SetRotation():
    """
    Sets the value for the steps for the user-given rotation angle.
    """

    # Offers randomization of lateral angle
    while True:
        try:
            r = int(raw_input("Randomize lateral angle? 1 for 'yes', 0 for
'no': "))
            if r == 1 or r == 0:
                break
            print "Invalid entry. Please enter either 1 for 'yes', or 0 for
'no'."
        except ValueError:
            print "Invalid entry. Please enter either 1 for 'yes', or 0 for
'no'."

    # Above forces user entry of either 1 or 0. If 1, randomly assigns angle
    # from within user-defined range. If 0, user chooses angle.

    if r:
```

```
        # Forces lower bound for randomization
        # within possible bounds of device.
        while True:
            try:
                low = int(raw_input("Randomizing within range. Minimum angle
is %d. Lower bound: " % M1MIN))
                if low >= M1MIN and low <= M1MAX:
                    break
                print "Angle must be between %d and %d degrees!" % (M1MIN,
M1MAX)
            except ValueError:
                print "Invalid entry. Numerical entry required."

        # Forces upper bound for randomization
        # within possible bounds of device and lower bound.
        while True:
            try:
                high = int(raw_input("Randomizing within range. Maximum angle
is %d. Upper bound: " % M1MAX))
                if high >= low and high <= M1MAX:
                    break
                print "Angle must be between %d and %d degrees!" % (low,
M1MAX)
            except ValueError:
                print "Invalid entry. Numerical entry required."

        degree = random.randint(low, high)
    else:
        while True:
            try:
                degree = float(raw_input("What angle should M1 be set to? "))
                if degree >= M1MIN and degree <= M1MAX:
                    break
                print "Angle must be between %d and %d degrees!" % (M1MIN,
M1MAX)
            except ValueError:
                print "Invalid entry. Numerical entry required."

    # Verification
    print "Rotation angle set to %d degrees." % degree

    # Define points for the equation of the form y = m(x-x1) + y1
    # This will relate x (user input degrees) and y (steps)
    pt1x = float(M1MIN + M1MAX) / 2
    pt1y = M1NEUTRAL
    pt2x = M1MAX
    pt2y = M1STEPS

    m = CalcSlope(pt1x, pt1y, pt2x, pt2y)

    # M1 needs integer count of steps.
    step = int(CalcPointSlope(pt1x, pt1y, m, degree))

    return step

def SetStep(motor, w1, w2, w3, w4):
```

```
    """
    Sets the step for a given stepper motor array and wire setup.
    """
    GPIO.output(motor[1], w1)
    GPIO.output(motor[2], w2)
    GPIO.output(motor[3], w3)
    GPIO.output(motor[4], w4)

def StepBackward(motor):
    """
    Moves stepper motor one cycle backward.
    """
    GPIO.output(motor[0], True)
    global SPCOUNT
    beat = SPCOUNT % 4

    if beat == 0:
        SetStep(motor,0,0,0,1)
        time.sleep(STEPDLY)
    if beat == 1:
        SetStep(motor,0,1,0,0)
        time.sleep(STEPDLY)
    if beat == 2:
        SetStep(motor,0,0,1,0)
        time.sleep(STEPDLY)
    if beat == 3:
        SetStep(motor,1,0,0,0)
        time.sleep(STEPDLY)

    SPCOUNT = SPCOUNT + 1

def StepForward(motor):
    """
    Moves stepper motor one cycle forward.
    """
    GPIO.output(motor[0], True)
    global SPCOUNT
    beat = SPCOUNT % 4

    if beat == 0:
        SetStep(motor,1,0,0,0)
        time.sleep(STEPDLY)
    if beat == 1:
        SetStep(motor,0,0,1,0)
        time.sleep(STEPDLY)
    if beat == 2:
        SetStep(motor,0,1,0,0)
        time.sleep(STEPDLY)
    if beat == 3:
        SetStep(motor,0,0,0,1)
        time.sleep(STEPDLY)

    SPCOUNT = SPCOUNT + 1

def StepStop(motor):
    """
    Stops stepper motor.
```

```
    """
    GPIO.output(motor[0], False)
    SetStep(motor,0,0,0,0)

####END SUBROUTINES. MAIN PROGRAM BELOW.####

####INITIALIZATION####

# Library Imports
import RPi.GPIO as GPIO
import random
import time

# Board Setup
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)

# GPIO Pin Allocation                       #Variable Name
# 01: Reserved - constant 3.3V source
# 02: Reserved - constant 5.0V source
# 03: USED - Limit Switch 1                 #L1
# 04: Reserved - constant 5.0V source
# 05: USED - Limit Switch 2                 #L2
# 06: Reserved - ground
# 07: USED - Limit Switch 3                 #L3
# 08: USED - Stepper Motor 1 pin 1          #M11
# 09: Reserved - ground
# 10: USED - Stepper Motor 1 pin 2          #M12
# 11: USED - Limit Switch 4                 #L4
######### 12: USED - Stepper Motor 1 pin 3              #M13 NO LONGER TRUE.
# 12: USED - Servo Motor 5 enable           #M5e
# 13: USED - Limit Switch 5                 #L5
# 14: Reserved - ground
# 15: USED - Limit Switch 6                 #L6
# 16: USED - Stepper Motor 1 pin 4          #M14
# 17: Reserved - constant 3.3V source
# 18: USED - DC Motor 2 pin 1               #M21
# 19: USED - Force Sensor                   #F1
# 20: Reserved - ground
# 21: USED - PIR Sensor                     #P1
# 22: USED - DC Motor 2 pin 2               #M22
# 23: USED - Stepper Motor 1 enable         #M1e
# 24: USED - DC Motor 3 pin 1               #M31
# 25: Reserved - ground
# 26: USED - DC Motor 3 pin 2               #M32
# 27: Reserved - ID_SD I2C ID EEPROM
# 28: Reserved - ID_SC I2C ID EEPROM
# 29: USED - DC Motor 2 enable              #M2e
# 30: Reserved - ground
# 31: USED - DC Motor 3 enable              #M3e
# 32: USED - DC Motor 4 pin 1               #M41
# 33: USED - DC Motor 4 enable              #M4e
# 34: Reserved - ground
######### 35: USED - Servo Motor 5 enable               #M5e NO LONGER TRUE.
# 35: <free>
# 36: USED - DC Motor 4 pin 2               #M42
# 37: <free>
```

```
######### 38: <free>                                          NO LONGER TRUE.
# 38: USED - Stepper Motor 1 pin 3          #M13
# 39: Reserved - ground
# 40: <free>

#SWITCHES
# Limit switch 1. L1. Rotation (lower limit). Static Base. Pin 3.
# Limit switch 2. L2. Rotation (upper limit). Static Base. Pin 5.
# Limit switch 3. L3. Elevation (lower limit). Rotating platform. Pin 7.
# Limit switch 4. L4. Elevation (upper limit). Rotating platform. Pin 11.
# Limit switch 5. L5. Pullback (min power). Angled platform. Pin 13.
# Limit switch 6. L6. Pullback (max power). Angled platform. Pin 15.
# Force sensor 1. F1. Launching (weight). Launch Tube. Pin 19.
# PIR/prox sw. 1. P1. Launching (path). Launch Tube. Pin 21.

#MOTORS
# Motor 1. M1. Rotation. Static Base. Stepper. Pins 23[e],8,10,38,16.
# Motor 2. M2. Elevation. Rotating platform. DC. Pins 29[e],18,22.
# Motor 3. M3. Auger. Rotating platform. DC. Pins 31[e],24,26.
# Motor 4. M4. Pullback. Angled platform. DC. Pins 33[e],32,36.
# Motor 5. M5. Grabber. Angled platform. Servo. Pin 12[e].

# Variable Initialization
# Pin values. Motors pins condensed to arrays for ease of reference.
L1 = 3
L2 = 5
L3 = 7
L4 = 11
L5 = 13
L6 = 15
F1 = 19
P1 = 21
M1e = 23
M2e = 29
M3e = 31
M4e = 33
M5e = 12
M11 = 8
M12 = 10
M13 = 38
M14 = 16
M21 = 18
M22 = 22
M31 = 24
M32 = 26
M41 = 32
M42 = 36
M1 = [M1e, M11, M12, M13, M14,'M1']
M2 = [M2e, M21, M22,'M2']
M3 = [M3e, M31, M32,'M3']
M4 = [M4e, M41, M42,'M4']
M5 = [M5e,'M5']
# To prevent accidental input to the system, free pins designated as outputs.
free = [37, 35, 40,'FREE PINS']

# Motor 1 Variables
M1STEPS = 0             # Steps count for M1 motion
```

```
M1NEUTRAL = 0           # Neutral Position for M1.
M1PRESENT = 0           # Present Position for M1.
M1MIN = -15             # Minimum Value for M1. (deg)
M1MAX = 15              # Maximum Value for M1. (deg)

# Motor 2 Variables
M2SCNDS = 0             # Travel time for M2 motion
M2NEUTRAL = 0           # Neutral Position for M2.
M2PRESENT = 0           # Present Position for M2.
M2MIN = 5               # Minimum Value for M2. (deg)
M2MAX = 45              # Maximum Value for M2. (deg)

# Motor 3 Variables
TREATSCNDS = 0          # Time to turn auger to dispense one treat.

# Motor 4 Variables
M4SCNDS = 0             # Travel time for M4 motion
M4NEUTRAL = 0           # Neutral Position for M4.
M4PRESENT = 0           # Present Position for M4.
M4MIN = 1               # Minimum Value for M4. (%)
M4MAX = 100             # Maximum Value for M4. (%)

# Other variables
STEPDLY = 0.01          # Delay between steps of stepper motor.
SPCOUNT = 0             # Counter for stepper motion
LIMFACT = 0.95          # Percentage of total range of motion allowed.
ballCount = 0           # Number of balls successfully thrown by FIDO.
ballLimit = 0           # Max balls thrown before FIDO stops.
timeStart = time.time() # Start time of FIDO operation.
timeLimit = 0           # Time limit before FIDO stops.
Current_State  = 0      # Current State of the IR sensor.
Previous_State = 0      # Previous State of the IR sensor.
DELTA = 0.1             # Time between checks of IR sensor (s).
CHECKTIME = 3/DELTA     # Total number of checks. 3 seconds delay = 30
checks.

# Pin Setup
# Set GPIO Pins as Outputs
OutputPins(M1)
OutputPins(M2)
OutputPins(M3)
OutputPins(M4)
OutputPins(M5)
OutputPins(free)

# Set GPIO Pins as Inputs - Pull Down Resistors
InputPinsDOWN(L1,'L1')
InputPinsDOWN(L2,'L2')
InputPinsDOWN(L3,'L3')
InputPinsDOWN(L4,'L4')
InputPinsDOWN(L5,'L5')
InputPinsDOWN(L6,'L6')
InputPinsDOWN(F1,'F1')
InputPinsDOWN(P1,'P1')

####CALIBRATION####
```

```
M1STEPS = CalibrateStepper(M1)
M2SCNDS = CalibrateDC(M2)
M4SCNDS = CalibrateDC(M4)

#After calibration, setup servo.
pwm = GPIO.PWM(M5[0], 100) # Sends a pulse at 100Hz, every 10ms.
pwm.start(5)
MoveServo(0) # Assuming 0deg holds ball, and 90deg releases ball
#### ABOVE SUBJECT TO CHANGE PENDING MOUNTED SERVO TEST.
#### SERVO SHOULD ENGAGE PINBALL AT THIS POINT.

# GPIO Events
GPIO.add_event_detect(L1, GPIO.RISING, callback=EStop, bouncetime=300)
GPIO.add_event_detect(L2, GPIO.RISING, callback=EStop, bouncetime=300)
GPIO.add_event_detect(L3, GPIO.RISING, callback=EStop, bouncetime=300)
GPIO.add_event_detect(L4, GPIO.RISING, callback=EStop, bouncetime=300)
GPIO.add_event_detect(L6, GPIO.RISING, callback=EStop, bouncetime=300)

####MAIN PROGRAM####

SetFinish()

while not (CountBalls() and CountTime()):

    # Ready - gets user input.
    Rot_steps = SetRotation()
    Elev_secs = SetElevation()
    Pull_secs = SetPullback()
    Aug_onoff = SetAuger()

    # Ready - does not proceed past this line until ball in place.
    GPIO.wait_for_edge(F1, GPIO.RISING)

    # Aim - moves into position
    MoveStep(M1, Rot_steps)
    MoveDC(M2, Elev_secs)

    # Engages spring:
    MoveDC(M4, Pull_secs)

    # Aim - waits for clearance
    for dt in range(0, int(CHECKTIME)):
        if isClear() != True:
            dt = 0
        time.sleep(DELTA)

    # Fires
    MoveServo(90)

    # Finish - Dispenses treat, increases ball count
    if Aug_onoff:
        MoveDC(M3, TREATSCNDS)
    ballCount += 1

GPIO.cleanup()
```

9 Appendix B: Code Analysis

The list below outlines the different tiers, and each function in that tier. After the function is listed (and its purpose is briefly stated), the called functions are also notated. In this way, we show the progression from base level coding to higher-level functions. This also gives a rough notion of the chronological order of the coding process. Though not true in all cases, the Tier 0 functions were written before the Tier 1 functions, which were written before the Tier 2 functions, and so on.

Tier 0

- CalcPointSlope – Given the coordinates of a point, the slope of a line intersecting that point, and the x-value of a second point on the line, calculates the y-value of the second point.
    - Calls: None
- CalcSlope – Given the coordinates of two points, calculates the slope of the line defined by them.
    - Calls: None
- CountBalls – Compares the current thrown ball count against the user-defined limit, and returns a “True” or “False” accordingly.
    - Calls: None
- CountTime – Compares the current elapsed time against the user-defined limit, and returns a “True” or a “False” accordingly.
    - Calls: None
- InputPinsDOWN – Given a pin number and its associated name, defines the pin as a pull-down input using GPIO commands, and reports that information to the user.
    - Calls: None
- OutputPins – Given a motor array, defines the pins as an output and sets the output to off using the GPIO commands, and reports that information to the user.
    - Calls: None

- SetAuger – Sets the treat dispenser to be enabled or disabled based on user input, and returns a “True’ or a “False” accordingly.
    - Calls: None
- SetFinish – Sets the termination conditions based on user input (end after balls thrown or end after minute elapsed).
    - Calls: None

Tier 1

- DCBackward – Given a motor array, turns the DC motor backward using GPIO commands.
    - Calls: None*
- DCForward – Given a motor array, turns the DC motor forward using GPIO commands.
    - Calls: None*
- DCStop – Given a motor array, turns the DC motor off using GPIO commands.
    - Calls: None*
- isClear – Checks if the trajectory of the ball on the launch tube is clear via the IR sensor using GPIO commands.
    - Calls: None*
- MoveServo – Moves the servo motor to a specified angle, but only if the angle is within the range of motion, using GPIO commands.
    - Calls: None*
- SetElevation – Sets the elevation angle travel time based on user input of desired angle.
    - Calls: CalcSlope, CalcPointSlope
- SetPullback – Sets the spring pullback travel time based on user input of desired power percentage.
    - Calls: CalcSlope, CalcPointSlope
- SetRotation – Sets the step count based on user input of desired rotation angle (may be randomized based on user input).
    - Calls: CalcSlope, CalcPointSlope

- SetStep – Given a stepper motor array and a wire set up, sets the motor wires to those values using GPIO commands.
    - Calls: None*

Tier 2

- MoveDC – Given a motor array and a timed position, moves the DC motor to the specified position using a timing function.
    - Calls: DCBackward, DCForward, DCStop
- StepBackward – Given a motor array, moves the stepper motor one step backward.
    - Calls: SetStep
- StepForward – Given a motor array, moves the stepper motor one step forward.
    - Calls: SetStep
- StepStop – Given a motor array, stops the stepper motor from moving.
    - Calls: SetStep

Tier 3

- CalibrateDC – Given a motor array, determines the full range of motion of that motor within the limits of the limit switches, moves the motor to the midpoint of the range of motion, and returns the number of steps in the full range of motion.
    - Calls: DCForward, DCBackward, DCStop, MoveDC
- MoveStep – Given a motor array and a step position, moves the stepper motor to the specified position using a step-counting function.
    - Calls: StepBackward, StepForward, StepStop

Tier 4

- CalibrateStepper – Given a motor array, determines the full range of motion of that motor within the limits of the limit switches, moves the motor to the midpoint of the range of motion, and returns the number of steps in the full range of motion.
    - Calls: StepForward, StepBackward, StepStop, MoveStep

Tier 5

- EStop – In conjunction with an "event detect" feature of the GPIO commands, safely stops all motors and re-runs calibration.
    - Calls: StepStop, DCStop, MoveDC, CalibrateStepper, CalibrateDC

*One might argue that these Tier 1 functions would be better defined as Tier 0 functions. After all, the function *DCBackward* directly sends HIGH or LOW values to GPIO pins. However, those values are based on the assumption that the pins have already been defined as output pins. Thus, while *DCBackward* does not directly call the Tier 0 function *OutputPins*, the Tier 0 function must have been written and run to call *DCBackward*. Similarly, function *isClear* checks the status of the GPIO input on the proximity sensor, which requires (though does not call) *InputPinsDOWN* to ensure that the pin is defined as a pull-down resistor input. For this reason, these functions are considered Tier 1.

10 Appendix C: Wiring Diagram

Row	NEGATIVE	POSITIVE	J	I	H	G	F	E	D	C	B	A	NEGATIVE	POSITIVE	Row
1	GND	3V3			3V3					5V			GND	5V	1
2	GND	3V3			L1					5V			GND	5V	2
3	GND	3V3			L2					GND			GND	5V	3
4	GND	3V3			L3					M11			GND	5V	4
5	GND	3V3		I5-C5	GND					M12			GND	5V	5
6	GND	3V3			L4					M5e			GND	5V	6
7	GND	3V3			L5					GND			GND	5V	7
8	GND	3V3			L6					M14			GND	5V	8
9	GND	3V3		I9-C9	3V3					M21			GND	5V	9
10	GND	3V3			F1					GND			GND	5V	10
11	GND	3V3			P1					M22			GND	5V	11
12	GND	3V3			M1e					M31			GND	5V	12
13	GND	3V3			GND					M32			GND	5V	13
14	GND	3V3			I2C					I2C			GND	5V	14
15	GND	3V3			M2e					GND			GND	5V	15
16	GND	3V3			M3e					M41			GND	5V	16
17	GND	3V3			M4e					GND			GND	5V	17
18	GND	3V3			<free>					M42			GND	5V	18
19	GND	3V3			<free>					M13			GND	5V	19
20	GND	3V3			GND					<free>			GND	5V	20
21	GND	3V3											GND	5V	21
22	GND	3V3											GND	5V	22
23	GND	3V3											GND	5V	23
24	GND	3V3											GND	5V	24
25	GND	3V3											GND	5V	25
26	GND	3V3											GND	5V	26
27	GND	3V3											GND	5V	27
28	GND	3V3											GND	5V	28
29	GND	3V3											GND	5V	29
30	GND	3V3											GND	5V	30
31	GND	3V3											GND	5V	31
32	GND	3V3											GND	5V	32
33	GND	3V3											GND	5V	33
34	GND	3V3											GND	5V	34

35	GND	3V3	GND	5V	35
36	GND	3V3	GND	5V	36
37	GND	3V3	GND	5V	37
38	GND	3V3	GND	5V	38
39	GND	3V3	GND	5V	39
40	GND	3V3	GND	5V	40
41	GND	3V3	GND	5V	41
42	GND	3V3	GND	5V	42
43	GND	3V3	GND	5V	43
44	GND	3V3	GND	5V	44
45	GND	3V3	GND	5V	45
46	GND	3V3	GND	5V	46
47	GND	3V3	GND	5V	47
48	GND	3V3	GND	5V	48
49	GND	3V3	GND	5V	49
50	GND	3V3	GND	5V	50
51	GND	3V3	GND	5V	51
52	GND	3V3	GND	5V	52
53	GND	3V3	GND	5V	53
54	GND	3V3	GND	5V	54
55	GND	3V3	GND	5V	55
56	GND	3V3	GND	5V	56
57	GND	3V3	GND	5V	57
58	GND	3V3	GND	5V	58
59	GND	3V3	GND	5V	59
60	GND	3V3	GND	5V	60
61	GND	3V3	GND	5V	61
62	GND	3V3	GND	5V	62
63	GND	3V3	GND	5V	63
64	GND	3V3	GND	5V	64

Table 2 : Breadboard 1 Wiring Diagram

Row	NEGATIVE	POSITIVE	J	I	H	G	F	E	D	C	B	A	NEGATIVE	POSITIVE	Row
1	GND	5V			M1e	G1-D8	1749_1_1	1749_1_16	5V (Pi)				GND	12V	1
2	GND	5V				M11	1749_1_2	1749_1_15	M14				GND	12V	2

			Motor			Motor				
3	GND	5V	Black	1749_1_3	1749_1_14	Red	GND	12V	3
4	GND	5V	GND	1749_1_4	1749_1_13	GND	GND	12V	4
5	GND	5V	GND	1749_1_5	1749_1_12	GND	GND	12V	5
			Motor			Motor				
6	GND	5V	Green	1749_1_6	1749_1_11	Blue	GND	12V	6
7	GND	5V	M12	1749_1_7	1749_1_10	M13	GND	12V	7
8	GND	5V	POS	1749_1_8	1749_1_9	G1-D8	GND	12V	8
9	GND	5V					GND	12V	9
10	GND	5V					GND	12V	10
11	GND	5V					GND	12V	11
12	GND	5V					GND	12V	12
13	GND	5V	M2e	1749_1_1	1749_1_16	5V (Pi)	GND	12V	13
14	GND	5V	M21	1749_1_2	1749_1_15	M32	GND	12V	14
			M2			M3				
15	GND	5V	Purple	1749_1_3	1749_1_14	White	GND	12V	15
16	GND	5V	GND	1749_1_4	1749_1_13	GND	GND	12V	16
17	GND	5V	GND	1749_1_5	1749_1_12	GND	GND	12V	17
			M2			M3				
18	GND	5V	White	1749_1_6	1749_1_11	Purple	GND	12V	18
19	GND	5V	M22	1749_1_7	1749_1_10	M31	GND	12V	19
20	GND	5V	POS	1749_1_8	1749_1_9	M3e	GND	12V	20
21	GND	5V					GND	12V	21
22	GND	5V					GND	12V	22
23	GND	5V					GND	12V	23
24	GND	5V					GND	12V	24
25	GND	5V	M4e	1749_1_1	1749_1_16	5V (Pi)	GND	12V	25
26	GND	5V	M41	1749_1_2	1749_1_15		GND	12V	26
			M4							
27	GND	5V	Purple	1749_1_3	1749_1_14		GND	12V	27
28	GND	5V	GND	1749_1_4	1749_1_13	GND	GND	12V	28
29	GND	5V	GND	1749_1_5	1749_1_12	GND	GND	12V	29
			M4							
30	GND	5V	White	1749_1_6	1749_1_11		GND	12V	30
31	GND	5V	M42	1749_1_7	1749_1_10		GND	12V	31
32	GND	5V	POS	1749_1_8	1749_1_9		GND	12V	32
33	GND	5V					GND	12V	33

34	GND	5V					GND	12V	34
35	GND	5V					GND	12V	35
36	GND	5V					GND	12V	36
37	GND	5V					GND	12V	37
38	GND	5V					GND	12V	38
39	GND	5V					GND	12V	39
40	GND	5V					GND	12V	40
41	GND	5V					GND	12V	41
42	GND	5V					GND	12V	42
43	GND	5V					GND	12V	43
44	GND	5V					GND	12V	44
45	GND	5V					GND	12V	45
46	GND	5V					GND	12V	46
47	GND	5V					GND	12V	47
48	GND	5V					GND	12V	48
49	GND	5V					GND	12V	49
50	GND	5V					GND	12V	50
51	GND	5V					GND	12V	51
52	GND	5V					GND	12V	52
53	GND	5V					GND	12V	53
54	GND	5V			V.R	Input	GND	12V	54
55	GND	5V			V.R	Ground	GND	12V	55
				to power					
56	GND	5V		rail	V.R	Output	GND	12V	56
57	GND	5V					GND	12V	57
58	GND	5V					GND	12V	58
59	GND	5V					GND	12V	59
60	GND	5V					GND	12V	60
61	GND	5V					GND	12V	61
62	GND	5V					GND	12V	62
63	GND	5V					GND	12V	63
64	GND	5V					GND	12V	64

Table 3 : Breadboard 2 Wiring Diagram