Project Squirrel 2.0:
A Tree Climbing Robot

A Major Qualifying Project

Submitted to the Faculty of WORCESTER POLYTECHNIC INSTITUTE In partial fulfillment of requirements of the Degree of Bachelor Science

Authors:

Elizabeth Brown

David Pounds

Michael Strickland

Alexander Stylos

Date:

4/30/2015

Submitted to:

Professor Kenneth Stafford, Co-Advisor

Professor Michael Gennert, Co-Advisor

Worcester Polytechnic Institute

Abstract

Project Squirrel 2.0 continues the series of tree climbing robots intended to aid in the discovery of evidence of Asian Longhorned Beetle infestations. The current method of beetle detection calls for people with binoculars to look for the holes left by the beetles. If an area is blocked from view or if the ground crew is unsure if an area is infected, a team of climbers is called in to inspect these portions. This robot is intended to be used by the ground crews to reduce the frequency that trees need to be climbed by humans. It is comprised of two sets of spring-loaded claws separated by a four degree of freedom linkage. An on-board camera allows the robot to be operated remotely from the ground even when the view of the robot is obstructed by the tree. The robot moves with an inchworm-like gait and has the ability to climb trees of varying diameters. On-board sensors determine the orientation of the robot, as well as contact with and distance from the tree to aid in detection and avoidance of branches.

Table of Contents

Table of Figures

EXECUTIVE SUMMARY

The Asian Longhorned Beetle eradication program calls for two separate teams of inspectors in order to completely survey a tree. Allowing the (first) ground crew team to get a more thorough view of hard to see areas would enable them to inspect more quickly and reduce the need for a (second) climbing team.

PROPOSAL

1. A robot can be employed by the ground team in order to reduce the frequency that the climbing team would need to be called in. The robot climbs to any problem areas noted by the ground crews and sends footage back to the operator. A user is able to control the robot remotely without direct visual contact.
2. The robot will grasp the tree using spring loaded legs designed to not expend any power to remain in place. The robot consists of two of these claws which allow for a secure grip on the tree without causing any damage.
3. The robot will have the degrees of freedom required to climb up and down the tree as well as negotiate around and onto branches. This is accomplished through the use of a gimbal joint, allowing for rotation in two directions to navigate the tree.
4. The robot will be controlled through a Graphical User Interface, running on a laptop that will also provide feedback on the status of the robot. This interface displays a view of what the robot sees as well as other necessary feedback for robot operation, such as battery level, connection status, position relative to the tree, and status of each leg.

Authorship

Abstract: Michael Strickland

Executive Summary: Elizabeth Brown & Michael Strickland

Introduction: All

Background Research: All

Initial Design: All

Robot overview (claws and legs): Michael Strickland

Robot overview (Middle segment): Elizabeth Brown

Robot Design Analysis (Statics Analysis – Other Sensors): Michael Strickland

Robot Design Analysis (Middle Segment – Power Requirements): Michael Strickland & Elizabeth Brown

Robot Design Analysis (Electrical Design): Alexander Stylos

Robot Design Analysis (Software Design): Dave Pounds

Testing Procedures: Dave Pounds

Results: Dave Pounds

Conclusions: All

Future Work and recommendations: All

Social Implications: All

Budget: Michael Strickland

Introduction

The purpose of this project is to continue the development of an Asian Longhorn Beetle (ALB) detecting robot. The ALB poses a massive threat to hardwood trees nationwide and has no natural predators in this country [1]. The ALB is unable to be combated by conventional insect control methods, the current method involves using human inspectors to climb trees and look for evidence of the beetles [2]. This can be dangerous and time consuming for the beetle inspectors, the perfect opportunity for a robot to take over. The process is also prone to human-error because one missed hole or egg site by an inspector can render the entire effort ineffectual.

Previous student projects have produced a working prototype able to maintain static position on a vertical tree and complete two gait cycles up a tree. The team plans to improve the performance and efficiency of the robot by enabling it to move in more than one direction on the tree. The team plans on increasing the distance traveled by the robot by finding and stabilizing a proper gait for climbing. In addition to robotic changes, the team will include a user friendly interface for operators to use the robot without needing prior handling experience. These changes will require a thorough understanding of the mechanical process of climbing in multiple directions up a tree, as well as a suitable grasp on control systems to provide a stable gait for the robot to operate with. Integrating all sensors to provide useful and accurate feedback will be a necessity in giving the control system adequate data. Finally, an understanding of the process flow for climbing is required to build a reliable program logic base for the robot to follow.

Project Goals

Primary Goals

Climb 10 feet up and down a vertical maple that is between 5 and 24in in diameter

Achieve an upwards velocity of 2.5ft/min

Maintain static position on a tree

Climb without damaging tree

Climb down a vertical tree

Avoid protrusions from the surface the robot is climbing on

Generate an intuitive and user friendly UI with camera feedback for navigation/identification

Secondary Goals

Climb other species of tree such as Oak and Ash

Climb onto and off of branches of diameter greater than 15cm which make a 25 degree or less angle from the current tree surface

Background Research

Research for this project includes an array of previous tree climbing robots as well as information concerning ALB. This will enable the team to understand the strengths and weaknesses of certain designs and incorporate the best aspects into the new robot version. The team can also learn how to best identify the presence of ALB, and incorporate that into the design.

The ALB lay their eggs in oviposition sites (shown below). The eggs hatch into larvae which burrow into the tree to mature into adult beetles and then emerge from the tree leaving dime-sized holes in the bark (see below). These holes are one of the best ways to determine the presence of beetles. The beetles themselves spend most of their lives inside the tree thus seeing the beetle itself is rare [1].

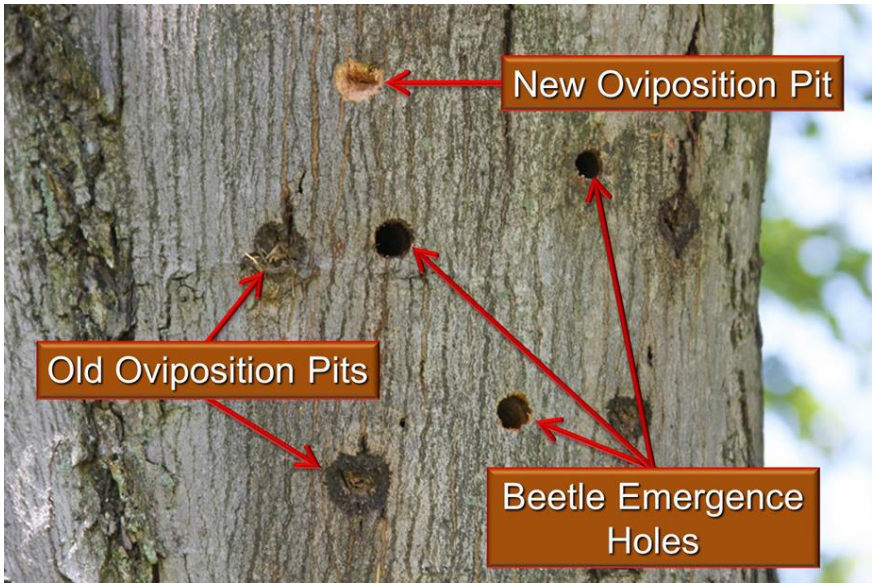


Fig 1. An image showing egg sites and exit holes **[3]**

Treebot

Treebot is a bipedal robot which uses a three degree of freedom continuum manipulator to climb trees. It is capable of climbing trees ranging in diameters from 2.5 in to 17.5 in. It is able to carry three times its weight and climb at a rate of 9 in/min. Treebot uses two compliant grippers to climb, needing no actuator to properly align the claws. Each gripper has four claws, which are made up of two parts, similar

to finger joints, with surgical needles at the ends to better grip trees. Only one gripper is necessary to have a static hold on a tree, allowing the other gripper to move to the next position. Using its continuum manipulator, Treebot can climb a tree with up to a 105° incline. It also uses this to navigate around obstacles and climb onto branches [4].

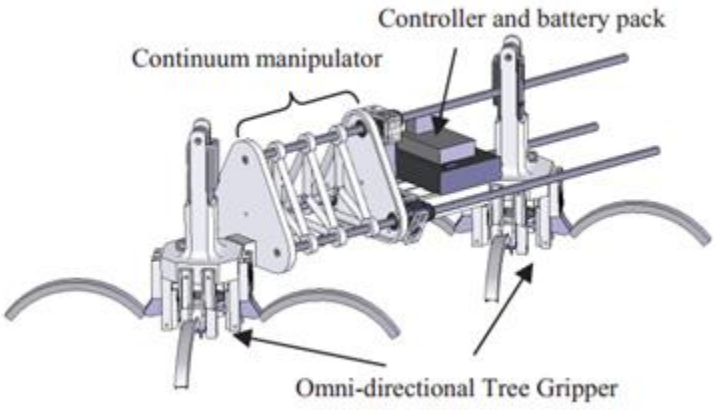


Fig 2. Treebot **[4]**

RiSE

RiSE is a project funded from the Defense Advanced Research Projects Agency (DARPA) Biodynamics Program. Its goal is to create biologically inspired robots that can both walk on land and climb vertical surfaces. The project was successful in creating three versions of a robot capable of climbing vertically up trees or telephone poles [5] [6] [7].

The first two versions, RiSE V1 and RiSE V2, both had six legs. Each leg was actuated by two motors through a differential gear drive system. The robots used a tripod gait, with three legs always attached to the tree for stabilization. Both robots also had a tail used to keep them against the tree. The gripping surface of each of the feet was composed of numerous spines attached to the surface the robot was climbing [5] [6] [8]. Using these features, the RiSE robots were able to climb several surfaces including carpet, wood, and stucco. The speed achieved by RiSE V1 was about 1 in/s while the speed achieved by RiSE V2 was about 2 in/s [8] [9].

RiSE V3 made several large changes to the design with the goal of being able to climb up straight vertical surfaces. The robot contains four legs in a quadrupedal formation. It climbs a pole using a cross between a bounded and crawl gait. The back legs thrust upwards simultaneously while the front legs recirculate one at a time. This provides the greatest speed while remaining in contact with the surface at all times. In addition to the new gait, the robot chassis has an extra degree of freedom allowing for adjustment of robot pitch. Brushless DC motors are used for each of the joints to add power density to the legs. Finally, the robot uses post-mortem surgical needles as claws to hold itself to the surface. RiSE V3 was successfully able to climb up a vertical telephone pole, achieving a speed of roughly 8 in/s [7] [9].

Carnegie Mellon University Modular Snake

Carnegie Mellon University created a modular snake-like robot, and while the main purpose of the snake is ground navigation of tight spaces, such as pipes, climbing trees is something it can also achieve. The idea behind the snake is the individual links within the robot create many degrees of freedom which provide highly functional locomotive capabilities [10]. These abilities include linear motion, translation movement, rolling, swimming, pipe climbing, and pole climbing.

The remarkable thing about the snake robot is it achieves its pole climbing capabilities without the aid of specific adhesion devices, such as claws or grippers. By coiling its body around a pole, it can achieve suitable frictional force to prevent slipping down the pole. Then, rolling its body parallel to the longitudinal axis of the pole, the snake robot can ascend and descend in a controlled fashion [11]. One of the drawbacks to this design is the pole it climbs must have a perimeter less than or equal to the length of the snake. Without this condition, the robot cannot maintain enough frictional force to keep gravity from pulling it down the pole as it rotates. This also makes the snake robot incapable of navigating around branches, since it must remain tightly wrapped around the pole to prevent falling.

The team believes the use of multiple degrees of freedom in the snake robot is a valuable trait when looking at creating designs that have the need for multiple locomotive capabilities. By incorporating

multiple degrees of freedom into the new design, the team hopes to achieve a gait that will allow the robot to dynamically turn on the tree itself.

Fig 3. Snakebot **[10]**

Previous Projects

Project Squirrel 1.0

Project Squirrel is a previous Major Qualifying Project (MQP) done by a group of students from WPI. The design is based on how squirrels interact with trees in the real world. This robot includes an innovative spring loaded, needle point claw, similar in function to those of a squirrel, which provided enough force the keep a reliable hold on the tree. In addition, the robot also used a spring loaded tail to provide enough counterbalancing force for the entire robot to remain in contact with the tree, similar to how a squirrel uses its own tail. This biologically inspired design allows Project Squirrel to adhere to the tree while using only three points of contact, giving it freedom to move a claw into position whenever it wants. The drawback to this design is the robot is unable to move in any sort of horizontal direction,

making it impossible to avoid branches that might be in the path. However, the team plans to make use of the claw design as it provides a very stable approach to gripping a tree.

Branching

Most existing tree climbing robots have no mechanism to navigate around branches because of the nature of tasks they have to complete. Two examples, a snake robot [10]and a pruning robot [12], are designed to either not move past branches or remove branches as it climbs. However the ability to circumvent branches and venture onto them is essential for the task of ALB detection. The team's approach will be very different from existing methods intends to modify the existing chassis if possible instead of starting from scratch. The current robot will need additional Degrees of Freedom (DoF) in order to move around branches. There are several ideas for how to increase the mobility of the robot.

The first idea is to have the top and bottom half of the robot separated by a one or two DoF linkage such that the top half of the robot can translate and re-align itself to the tree. This translation motion is completely decoupled from the vertical climbing. The advantage of this design is it only requires two additional DoF. However, this is an incomplete solution because while this design allows for movement around branches, it does not allow for transition onto branches themselves.

Another idea is to once again split the robot into a top and bottom half and add a ball joint between the two halves and add control to the single DoF of the tail. This will permit the robot to pitch, yaw, and roll on the tree allowing for the transitions to and from branches. This will also allow for the robot to be able to adjust its distance from the tree. This will be a huge advantage especially seeing as how the previous MQP group was concerned the robot may eventually work its way off the tree [13]. The disadvantage of this is it is a very complex design with many variables to solve. Significant time and thought will have to be put into it to make sure it works well.

Active Tail

RiSE V2 has an active tail, meaning the position of the tail can be adjusted by the robot during operation. This allows the robot to modulate the force the tail exerts on the tree and be able to climb down

the tree backward. This is not possible in Project Squirrel's design because the tail is always engaged and reversing would cause the tail to catch on the tree, forcing the robot to fall or get stuck. The tail is an important system that exerts a moment about the lower legs to counteract gravity pulling the top of the robot off the tree. The current tail is spring-loaded to push against the tree at all times and has a variable preload system, but this can only be altered by a human operator [13]. One idea for how to make an active tail is to have the tail both motor driven and spring-loaded. Without the preload the motor would have to exert constant force on the tree while climbing, causing the motor to run at or close to stall. This will potentially damage the motor and draw a large amount of current, reducing the robot's operating time. A system similar to the ballistic legs, created by last year's group, could be used. The ballistic leg system has a preloaded spring and a motor which drives the leg until a latch releases, decoupling the motor from the leg and sending the leg into the tree [13]. The latch-trigger system can be seen in Fig 4, below.

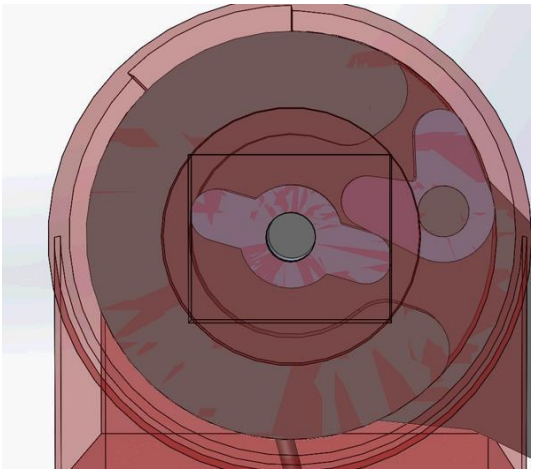


Fig 4. The Existing Ballistic Trigger Mechanism **[13]**

The difference in the tail mechanism is it will have a one way lock positioned immediately before the tail reaches the limit of its travel away from the tree. This will allow the tail to be held away from the tree without power. At this point, driving the tail further away from the tree will cause it to snap back to the tree and continue normal operation.

Initial Design

The initial idea for the robot is inspired by Treebot, mentioned above. An initial model of this design is shown below in Fig 5.

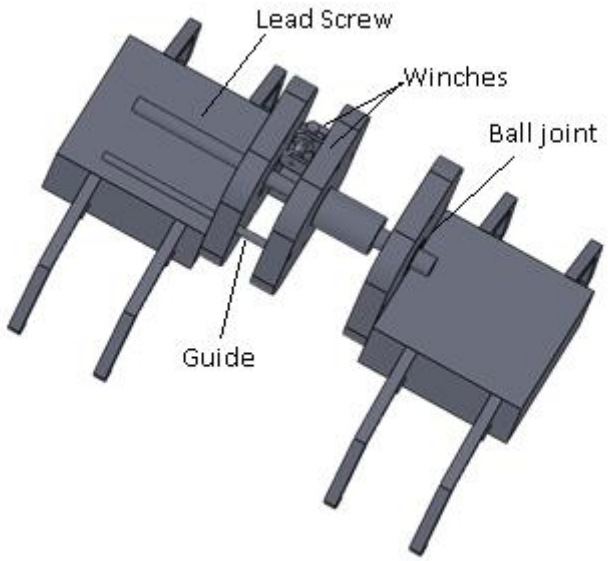


Fig 5: Rough CAD Model of Robot

The robot has two claw assemblies linked by a ball joint and lead screw (Fig 5). The ball joint will have at least two controlled degrees of freedom, pitch and yaw, with a maximum angle of 25 degrees in all directions. It is unknown at this time if the roll of the joint will be controlled by a motor or compliant as in Treebot. In either case, the pitch and yaw will be controlled by worm driven winches and cables. The motion of the cables will be opposed by linear springs which allow the robot to move in both directions without the need for extra motors or complex cable routing and tensioning. The winches will be worm driven and therefore non-back-drivable meaning the robot does not have to use power to hold itself in position, which would stall the motors. The ball joint is attached to the bottom half of the robot and the

top half moves with rotation of the lead screw. This configuration will allow the robot to navigate the tree by bending and reaching in different directions.

The robot has three primary gaits it will use to navigate the tree. All gaits, telescoping, branching, and turning, are necessary for climbing different areas of the tree. The telescoping gait (Fig. 6) will move the robot vertically by simply using the lead screw to move in the z-direction relative to the robot. This gait will work both when climbing up or down.

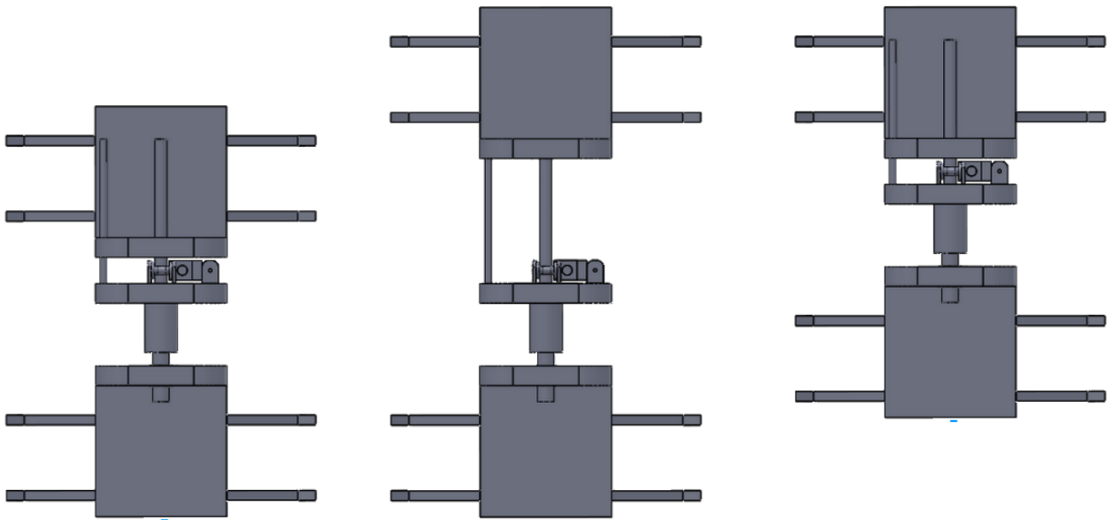


Fig 6: Telescoping Gait

Navigating onto branches [Fig. 7] will be accomplished by the robot pitching back and extending the lead screw up to the destination branch and grabbing on. The robot will then release the bottom claw, collapse the telescope, pitch forward and close the second claw. The robot will then use the telescoping gait to continue along the new branch until meeting another bend. A modification of this gait can also be used to adjust the robot's distance from the tree and pull the robot closer to the tree if it begins to work its way off.

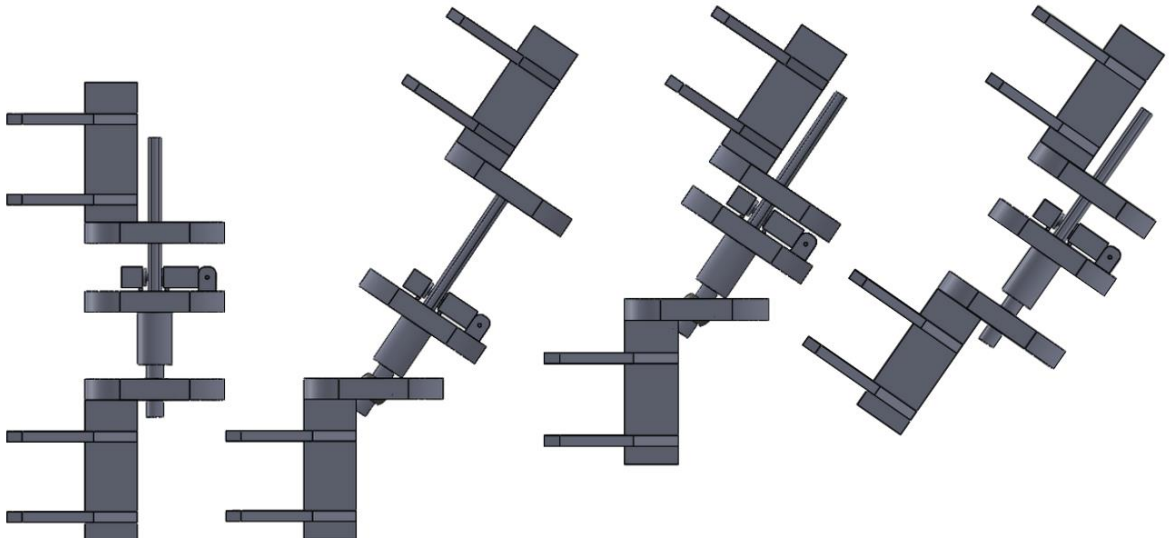


Fig 7: Branching Gait

Climbing around the tree [Fig. 8] is the most complex gait and will need the most attention and testing. The robot will first yaw and then extend the lead screw. Once extended, the robot will pitch into the tree. If the roll joint is controllable the robot will use this to align the claw tangent to the tree. If the joint is passive the robot will simply continue pushing against the tree until the claw is set properly. The front claw will close and the robot will collapse the lead screw and grab the tree with the bottom claw.

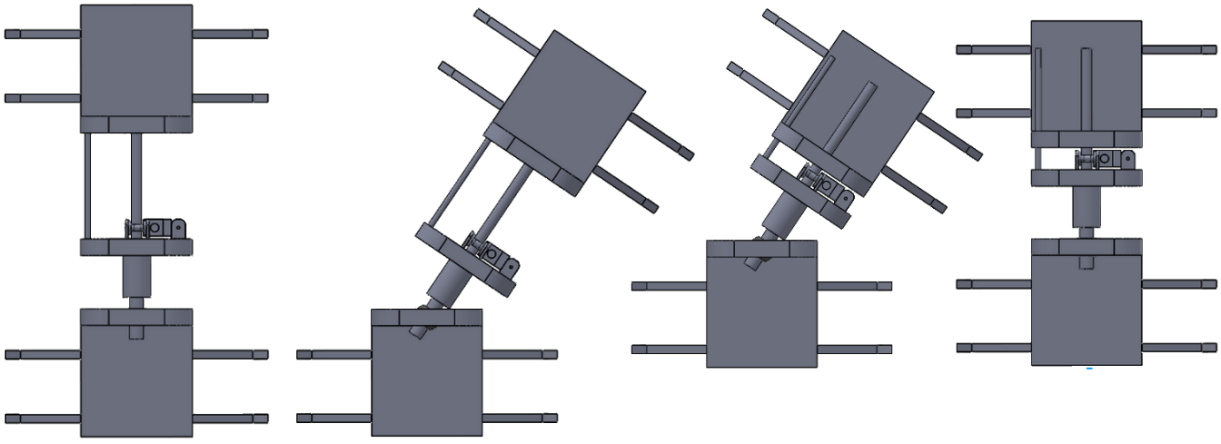


Fig 8: Turning Gait

After turning the robot can either repeat the gait and continue up the tree in a helical motion or repeat the same motion but downward. The advantage of the second option is it requires less space and therefore the robot will be able to move sideways to avoid a large branch or several small adjacent branches.

Initial static analysis shows there will be a large amount of force on the lower claws when the robot is branching. Assuming the back half of the robot weights 4.5 lb and the front part weighs 2.25 lb, and the force is evenly distributed among the four claws. Using moment sum around the lower CoG, the force would then be:

$$\frac{4.5[lb] * 5.5[in] + 2.25[lb] * 3.15[in]}{4 * 1.18[in]} = 6.9[lb]$$

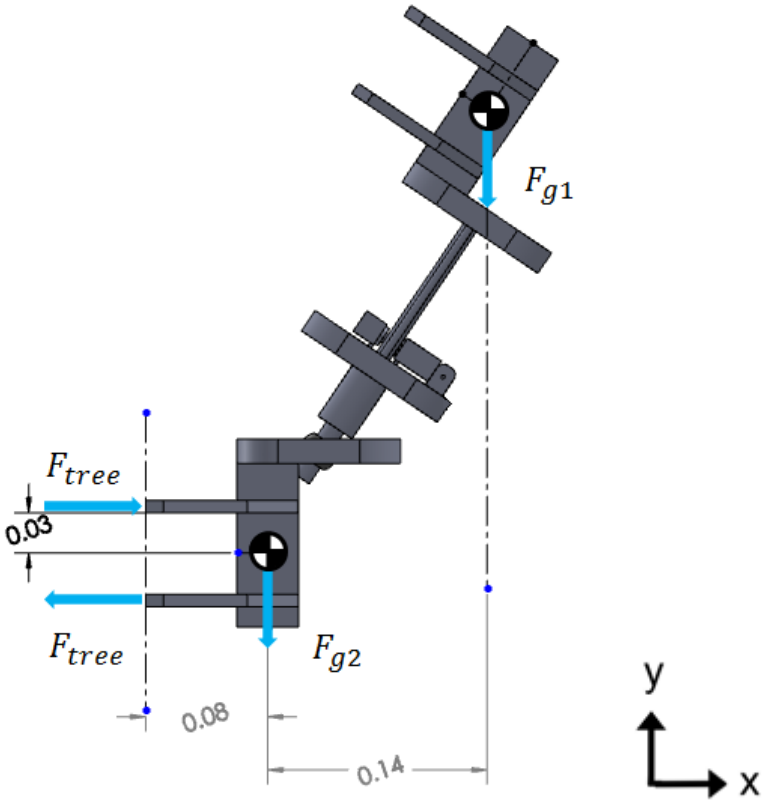


Fig 9: FBD of the robot reaching upward

This force will be pulling the upper set of claws off the tree. This force can be reduced by adding an active tail to the back segment that would counteract the majority of the force and will need to be finely tuned to optimize the counteracting effect.

Project Strategy

Robot Overview

The robot's shape resembles that of an inchworm, two claws grip the tree and an articulated segment separates them. The claws are composed of four spring loaded legs that are pulled back and released simultaneously. The adjoining segment has four degrees of freedom, three of which are powered. The extension of the robot, pitch angle, and yaw angle are controlled, while the roll is passive.

Claws

The claws are the most critical components of the robot; if they fail, the robot could get stuck, or worse, fall out of the tree. The legs rely on ballistic force to imbed the needles into the tree. In order for this method to be most effective, the claws need to move as fast as possible into the tree; back-driving the motor is not an option. The solution is an escapement mechanism between the legs and the motor. The mechanism uses spring loaded plungers to link the motor and the legs. When the torque reaches a certain threshold, the plungers will yield and the legs are free to move independent of the motor. The release force of the plungers is tunable with a set screw.

Legs

The legs of the robot are roughly 3 in long and tipped with an 18 Gauge needle for gripping the tree. The legs have a passive joint that allows for a small amount of deflection. The two halves of the leg are bridged by a strain gauge which is used to detect contact with the tree.

Middle Segment

The middle section of the robot is made up of a lead screw and gimbal joint. The gimbal joint is actuated using cables and pulleys, powered through a non-backdrivable worm gear. This joint allows for 25° of rotation from vertical in both the pitch and yaw directions. Potentiometers are used on each of the four pulleys to measure their exact angular position. The lead screw allows the robot to extend and is stabilized to prevent the upper claw from rotating.

Robot Design Analysis

Static Analysis

One extreme position of the when robot climbing a vertical tree is shown in Fig 10, below.

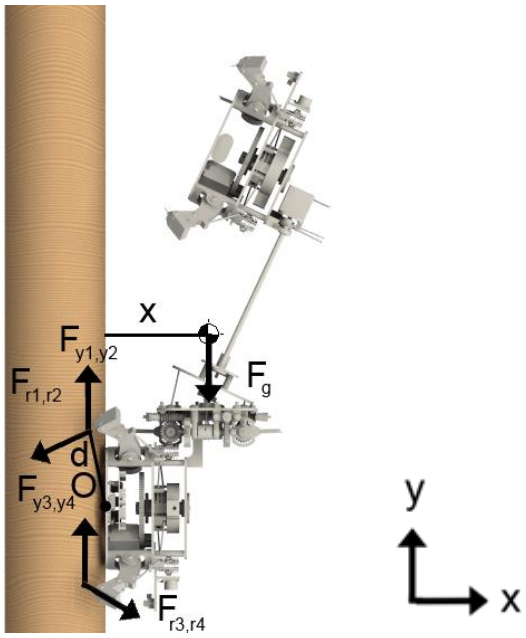


Fig 10. FBD of robot leaning away from a vertical tree

Having the claw held away from the tree as shown exerts a moment of

$$F_g \cdot x = 5.5[lb] \cdot 5.1[in] = 28.1[in \cdot lb]$$

About the point O between the 4 legs of the lower claw. The claws are all 3.3[in] from this point on the smallest possible tree and 5.6 in away on the largest tree. The force balance becomes:

The angle the claw reaction forces make with the horizontal is $25°$

$$\Sigma F_x = 2F_r \cos(25°) - 2F_r \cos(25°) = 0$$

$$\Sigma y = F_{y1} + F_{y2} + F_{y3} + F_{y4} - F_g + 2F_y \sin(25°) - 2F_y \sin(25°) = 0$$

$$\Sigma M_O = 4F_r \cdot d - 4F_y \cdot d_x - F_g \cdot x = 0$$

$$\Sigma y = F_{y1} + F_{y2} + F_{y3} + F_{y4} - F_g + \cancel{2F_y \sin(25°)} - \cancel{2F_y \sin(25°)} = 0$$

$$\Sigma y = F_{y1} + F_{y2} + F_{y3} + F_{y4} = F_g = 5.5[lb]$$

Assume $F_{y1} = F_{y2} = F_{y3} = F_{y4}$ by symmetry

$$F_{y1} = F_{y2} = F_{y3} = F_{y4} = 1.375[lb] \rightarrow F_y = 1.4[lb]$$

$$\Sigma M_O = 4F_r \cdot 3.3[in] - 4(1.375[lb]) \cdot 1.6[in] = 5.5[lb] \cdot 5.1[in]$$

$$F_r = 1.46[lb] \rightarrow F_r = 1.5[lb]$$

$$\left|\overrightarrow{F_{1,2}}\right| = \sqrt{1.5[lb]^2 + 1.375[lb]^2} = 2.0[lb]$$

$$\angle\overrightarrow{F_{1,2}} = \tan^{-1}\left(\frac{1.375[lb] + 1.5[lb] \cdot \sin(180 + 25°)}{1.5 \cdot \cos(180 + 25°)}\right) = 124°$$

$$\left|\overrightarrow{F_{3,4}}\right| = \sqrt{1.5[lb]^2 + 1.375[lb]^2} = 2.0[lb]$$

$$\angle\overrightarrow{F_{3,4}} = \tan^{-1}\left(\frac{1.375[lb] + 1.5[lb] \cdot \sin(-25°)}{1.5 \cdot \cos(-25°)}\right) = 29°$$

The other extreme position while climbing vertically is when the robot is leaning to the side (Fig 11).

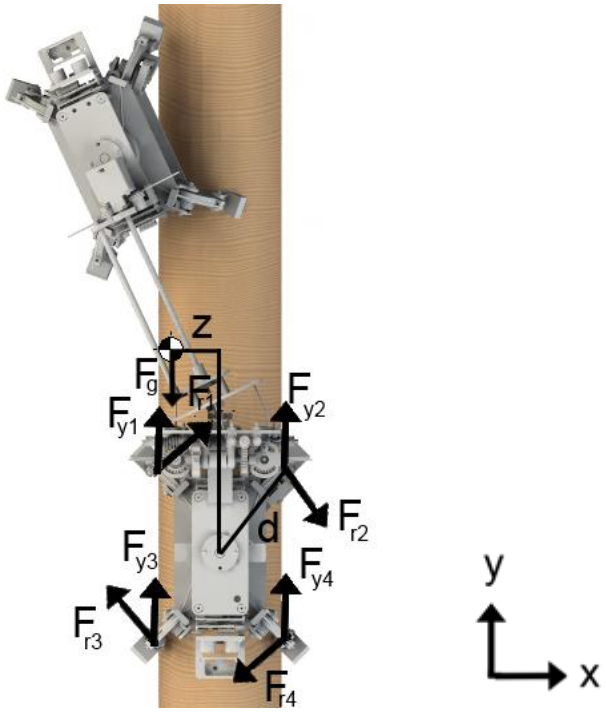


Fig 11. FBD of robot leaning to the side from a vertical tree

The angle of F_r from horizontal is $\pm 39°$

Assume:

$$F_{r1} = F_{r2} = F_{r3} = F_{r4} = F_r$$

Therefore

$$F_r = \frac{Ft \cdot z}{4 \cdot d} = \frac{5.5[lb] \cdot 2[in]}{4 * 3.3[in]} = 0.83[lb]$$

$$\Sigma y = F_{y1} + F_{y2} + F_{y3} + F_{y4} + 2 \cdot F_r \cdot \sin(39°) - 2 \cdot F_r \cdot \sin(39°) - F_g = 0$$

$$\Sigma x = 2 \cdot F_r \cdot \cos(39°) - 2 \cdot F_r \cdot \cos(39°) = 0$$

$$F_{y1} = F_{y2} = F_{y3} = F_{y4} = 1.375$$

$$\angle \overrightarrow{F_1} = \tan^{-1}\left(\frac{F_{y1} + F_r \cdot \sin(39°)}{F_r \cdot \cos(39°)}\right) = 71°$$

$$\left|\overrightarrow{F_1}\right| = \sqrt{\left(F_{y1} + F_r \cdot \sin(39°)\right)^2 + (F_r \cdot \cos(39°))^2} = 2.0[lb]$$

$$\angle F_2 = \tan^{-1}\left(\frac{F_{y2} + F_r \cdot \sin(-39°)}{F_r \cdot \cos(-39°)}\right) = 53°$$

$$\left|\overrightarrow{F_2}\right| = \sqrt{\left(F_{y2} + F_r \cdot \sin(-39°)\right)^2 + (F_r \cdot \cos(-39°))^2} = 1.1[lb]$$

$$\angle F_2 = \tan^{-1}\left(\frac{F_{y2} + F_r \cdot \sin(180° + 39°)}{F_r \cdot \cos(180° + 39°)}\right) = 127°$$

$$\left|\overrightarrow{F_3}\right| = \sqrt{\left(F_{y3} + F_r \cdot \sin(180° + 39°)\right)^2 + (F_r \cdot \cos(180° + 39°))^2} = 1.1[lb]$$

$$\angle F_4 = \tan^{-1}\left(\frac{F_{y4} + F_r \cdot \sin(180° - 39°)}{F_r \cdot \cos(180° - 39°)}\right) = 108°$$

$$\left|\overrightarrow{F_4}\right| = \sqrt{\left(F_{y4} + F_r \cdot \sin(180° - 39°)\right)^2 + (F_r \cdot \cos(180° - 39°))^2} = 2.0[lb]$$

This means the magnitude of the force on each foot is

$$\left|\overrightarrow{F_1}\right| = \sqrt{2.0[lb]^2 + 2.0[lb]^2} = 2.8[lb]$$

$$\left|\overrightarrow{F_2}\right| = \sqrt{2.0[lb]^2 + 1.1[lb]^2} = 2.3[lb]$$

$$\left|\overrightarrow{F_3}\right| = \sqrt{2.0[lb]^2 + 1.1[lb]^2} = 2.3[lb]$$

$$\left|\overrightarrow{F_4}\right| = \sqrt{2.0[lb]^2 + 2.0[lb]^2} = 2.8[lb]$$

The absolute worst case is when the robot is climbing horizontally on the underside of the tree. This would occur when the robot is crawling out onto a branch [Fig 12]. The robot should not end up in this position very often as the underside of branches can be easily seen from the ground. However there is a case where the robot could climb on the bottom of one branch to look at the top of a branch below it.

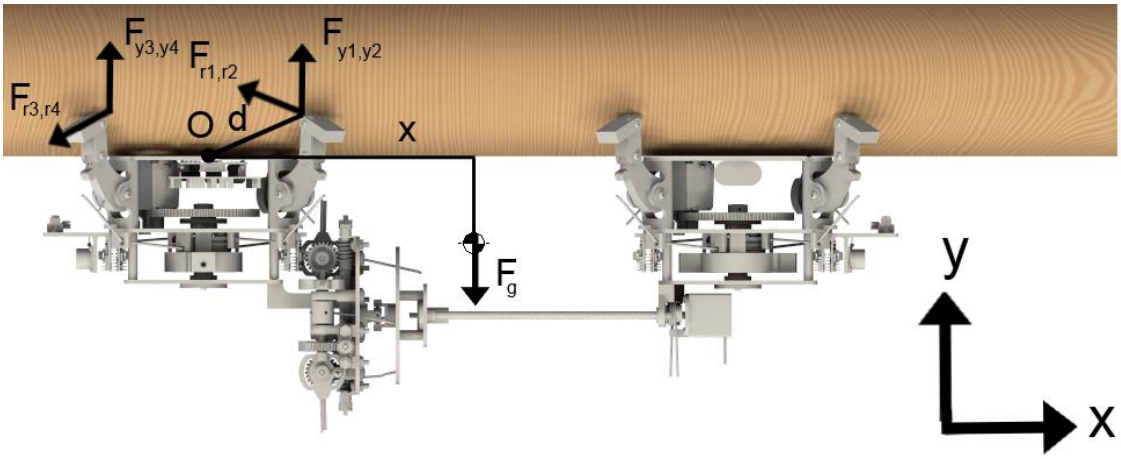


Fig 12. FBD of the robot on the underside of a horizontal branch

$$\Sigma F_y = F_{y1} + F_{y2} + F_{y3} + F_{y4} + 2 \cdot F_r \cdot \sin(25°) - 2 \cdot F_r \cdot \sin(25°) - F_g = 0$$

$$\Sigma F_x = 2F_{r1} \cdot \cos(25°) - 2F_r \cdot \cos(25°) = 5.5[lb]$$

$$\Sigma M_o = 4F_r \cdot d + 2F_y \cdot d - 2F_y \cdot d - F_g \cdot x$$

$$\Sigma M_o = 4F_r \cdot \cancel{d + 2F_y \cdot d - 2F_y \cdot d} = 5.5[lb] \cdot 8.3[in]$$

$$F_r = 3.5[lb]$$

$$F_y = \frac{5.5[lb]}{4} = 1.375[lb]$$

$$\left|\overrightarrow{F_{1,2,3,4}}\right| = \sqrt{1.375[lb]^2 + 3.5[lb]^2} = 3.72[lb]$$

Leg Spring Force and Motor Selection

The max torque of the spring $T_{max} = 7.5$ in · lb

The max deflection angle of the spring $\theta_{max} = 270°$

Spring Pre-load $\theta_i = 45°$

Spring Angle at max leg height $\theta_f = 135°$

Therefore the maximum torque exerted by the spring on the leg is:

$$7.5\left[\frac{\text{in}}{\text{lbs}}\right] * \frac{135°}{270°} = 3.38\left[\frac{\text{in}}{\text{lbs}}\right]$$

Due to the changing angle of the cable relative to the leg, the force required to lift it is not linear. The max force for a single leg is 4.6 lbs.

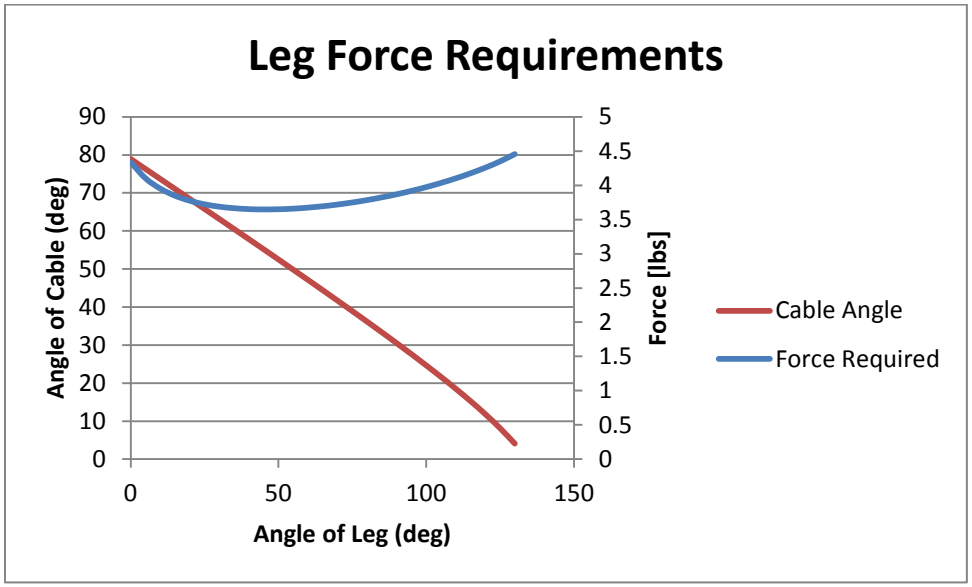


Fig 13. Graph of force needed to move legs as a function of leg angle

The speed at which the legs raise is given by:

$$\frac{S_{motor}}{N * D_{Pulley}}$$

The Torque required for the motor to lift the legs is given by:

$$F_{leg} * N * D_{Pulley} * \eta^{n}$$

The legs need to open in 2 seconds and the linear length of cable that needs to be taken up to move the legs is 2.1 in. This means the tangential velocity of the pulley is then 1.1 in/s

The output power required by the motor is:

$$4.6[lb] * 4 * 1.067\left[\frac{in}{s}\right] = 19.64\left[\frac{inlb}{s}\right] * \frac{0.113[W]}{1\left[\frac{inlb}{s}\right]} = 2.22[W]$$

$$\left(\frac{4F \cdot D}{2N \cdot \eta} \cdot -6.7751\left[\frac{RPM}{inlb}\right] + 100[RPM]\right)\left(\frac{4F \cdot D}{2N \cdot \eta} \cdot \frac{4.45\left[\frac{N}{lb}\right]}{39.37\left[\frac{in}{m}\right]}\right) =$$

Plugging in 2.2 W for P and reduction N=8 gives a diameter of approximately 1.75”

The motor has an 8:1 reduction. Thus, the output torque of the motor is

$$4.6[lb] * 4 * \frac{1.75[in]}{2} * \frac{1}{8} *.95^1 = 1.91[in \cdot lb]$$

The output speed of the motor at 1.91 in · lb is 86 RPM

$$86[RPM] * 1.75[in] \cdot \pi \cdot \frac{1}{8} \cdot \frac{1[min]}{60[sec]} = .99\left[\frac{in}{s}\right] \approx 1.067\left[\frac{in}{s}\right] \; the\; target\; speed$$

Several motors were considered but the only one found met this power requirement and fit the form factor of the robot was the VEX393 motor. A note that the VEX motor is overpowered for this application, because the motor was originally intended to lift the legs with almost double the spring force.

Cable Routing

All the cables are controlled by the same motor, and this creates some problems. Firstly, the close proximity of the cables requires the pulley to have two grooves to ensure the cables nearest each other would not cross as they are wound in. If one or more of the legs contacts the tree before the others there will be slack in the cables. To keep the slackened cables from catching on other parts of the robot they are tightly controlled by captive fairleads and a pulley cover. [Fig 14] The cover prevents the cables from jumping to the other groove on the pulley and the fairleads ensure the slack piles up in a predicable area away from potential ensnarement hazards. [Fig 15]

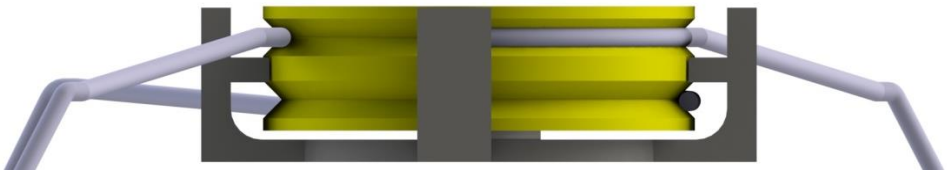


Fig 14. Close up view of pulley and grooves

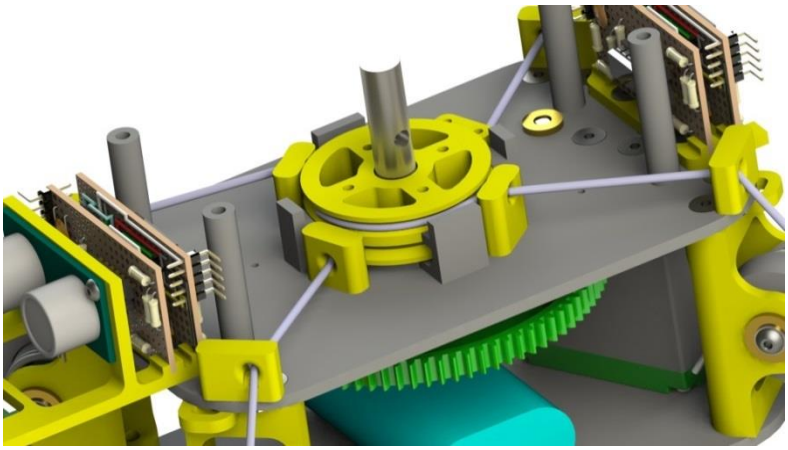


Fig 15. Cable Routing

Torque Release Mechanism

The torque release mechanism operates by balancing the force of pulling back the four torsional springs on the legs with two compression springs [Fig 16].

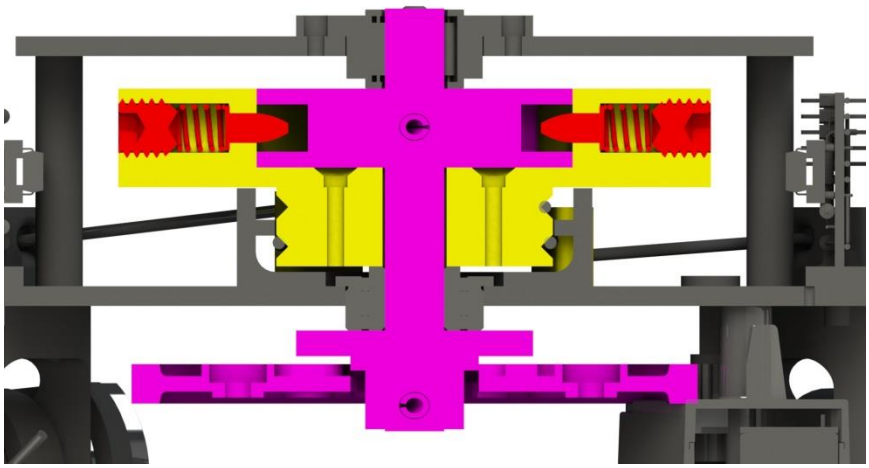


Fig 16: Cut-away view of the torque release mechanism

The inner disk (yellow) has two tabs that interact with the plungers. The plungers are backed by the compression springs and set screws. The shaft is pinned to both the inner disk and the drive gear (not shown) and transmits all the force to the legs. In order to guarantee the combined spring force of all four legs does not back-drive the motor one of the bearings is a one-way needle bearing, allowing rotation only in the direction that causes the legs to lift up.

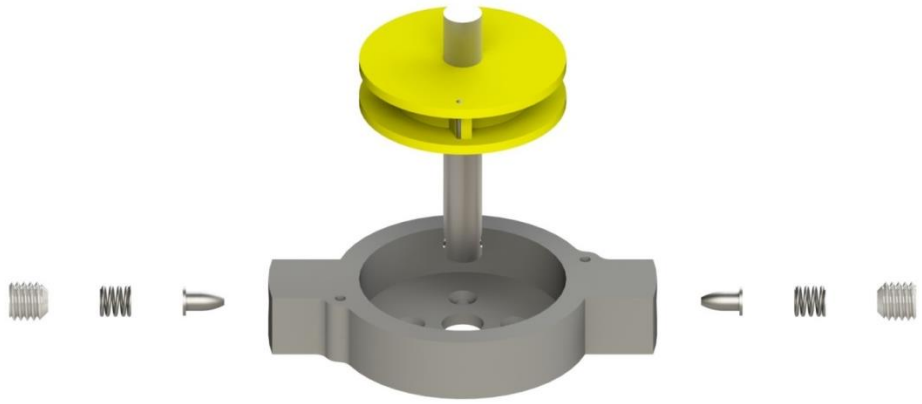


Fig 17. Exploded view of Torque Release Mechanism

Originally the parts were 3D printed from ABS but wear from contact with the aluminum plungers necessitated the leading edge be replaced with hardened steel rod [Fig 18]

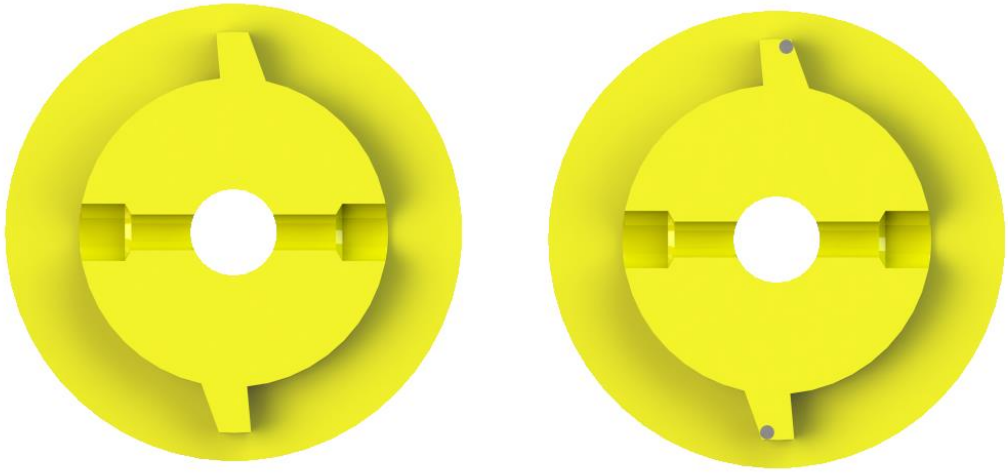


Fig 18. Inner disk before and after steel rods added

The compression springs need to release right immediately after the end of travel for the legs to ensure the motor stalls for little to no time. Setting this relationship is made easy by the set screws behind each spring that allow for the preload force to be adjusted. The process for tuning the mechanism is described in Appendix A. The springs had to be small in diameter and length to keep the thickness and diameter of the mechanism to a minimum, but also strong enough to withstand the forces from the plunger.

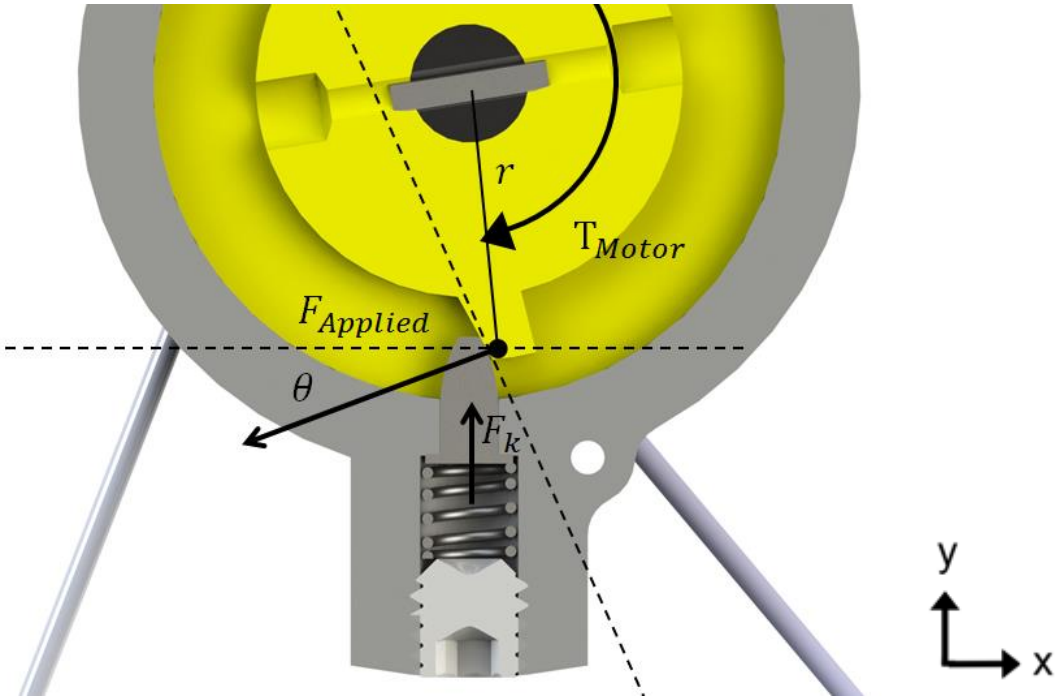


Fig 19. FBD of disk-plunger interaction

From the calculations above, $T_{motor} = 4.6\left[\frac{lb}{leg}\right] \cdot 4[legs] \cdot \frac{1.75[in]}{2} = 16.1[inlbs]$

$$F_{applied} = \frac{T_{motor}}{r} \cdot \cos(\theta) = \frac{16.1[inlbs]}{.84[in]} \cdot \cos(20°) = 18.0[lb]$$

$$F_k = F_{applied} \cdot \cos(90° - \theta) = 18.0[lb] \cdot \sin(70°) = 6.14[lb]$$

$$\frac{6.14[lb]}{2[plungers]} = 3.2\left[\frac{lb}{plunger}\right]$$

The springs chosen were 32.8 lbs/in, 10 lb max load springs. They had a short overall length, and could fit inside minor diameter of a 3/8”-16 set screw. This was the largest hole that could be put in ½” thick material without fear of material failure or needing to make the overall height of the robot larger.

Legs

Each Claw of the robot has four legs. The legs contain a torsion spring that forces it downward into the tree. Each leg is comprised of two main parts: the upper leg and the foot. The upper leg is attached to the robot and the torsion spring. The foot has an 18G needle at the end [Fig 20] which is what is used to grip the tree. These two sections are connected with a joint that allows a small amount of deflection. In between the two pieces these is a piece of steel with a strain gauge on it [Fig 21].

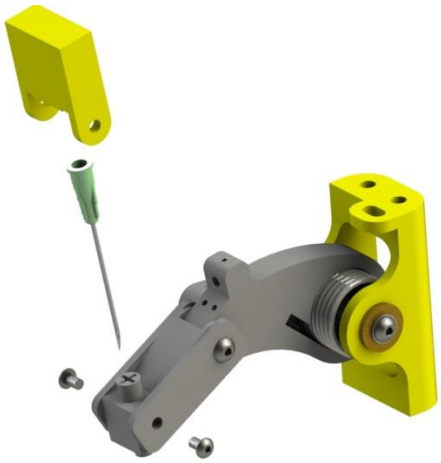


Fig 20. Needle is inserted into the foot and held in by the cap (yellow)

Fig 21. Cut-away view showing strain gauge

The strain gauge is part of an important feedback system to alert the operator if the robot does not properly grip the tree. The strain gauge will deflect when the leg is contacting the tree and will register a voltage difference on the ADC. The status of each leg is reported independently to the operator in real time on the GUI.

Other Sensors

One leg on each claw has a potentiometer on it. The potentiometer is on the shaft of the leg, but in order for the leg shaft to move with the leg a modification had to be made. A lever arm was added to attach the leg to the shaft to ensure they rotate together.

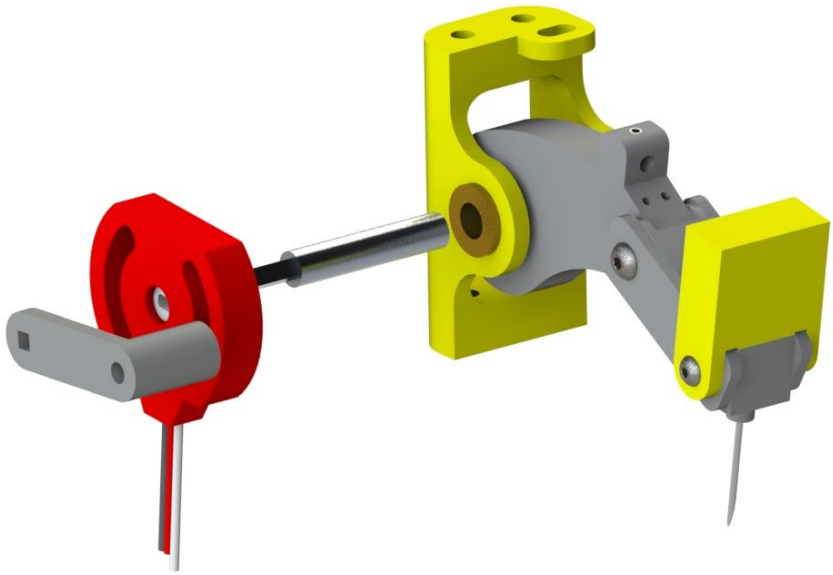


Fig 22. Exploded view showing installation of potentiometer

The front and back of the claws contain mounts for the rangefinders and the strain gauge breakouts [Fig 23]**.**

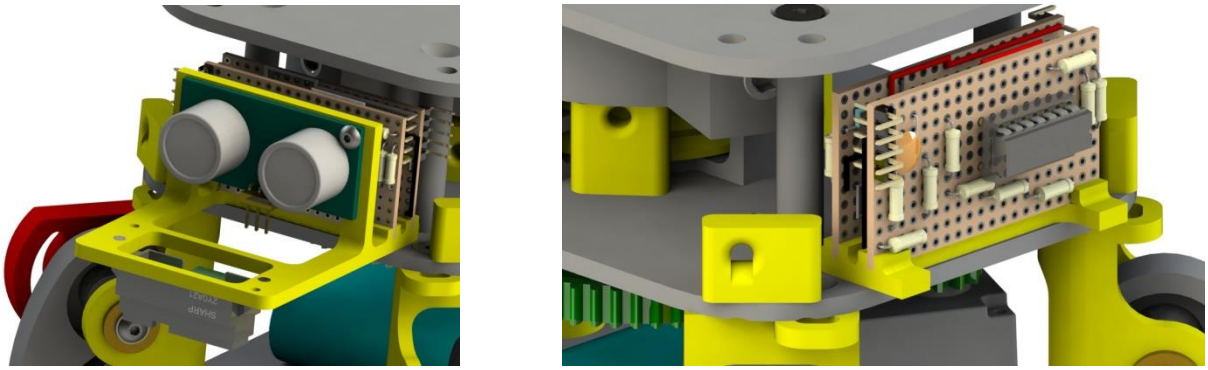


Fig 23. Mount for breakouts and rangefinders (left) and breakouts alone (right)

Middle Segment

The middle section of the robot is made up of a lead screw and gimbal joint. This joint is controlled through four cables connected at a distance of 1.8 in from the center. These cables are wound onto four pulleys. Due to the geometry of the motion of the upper plate, it was necessary to power opposing motors separately and run them at different velocities.

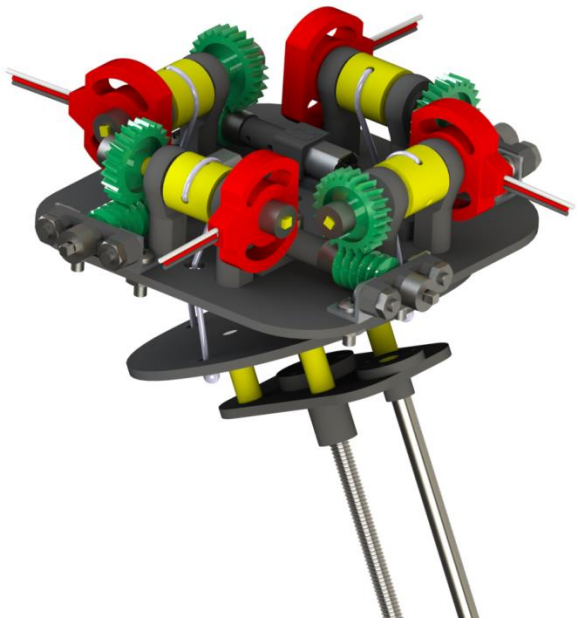


Fig 24. Middle segment

Due to the geometry of the middle segment, the attachment point for the cable moves in an arc. This causes the function of distance to this point to be non-linear.

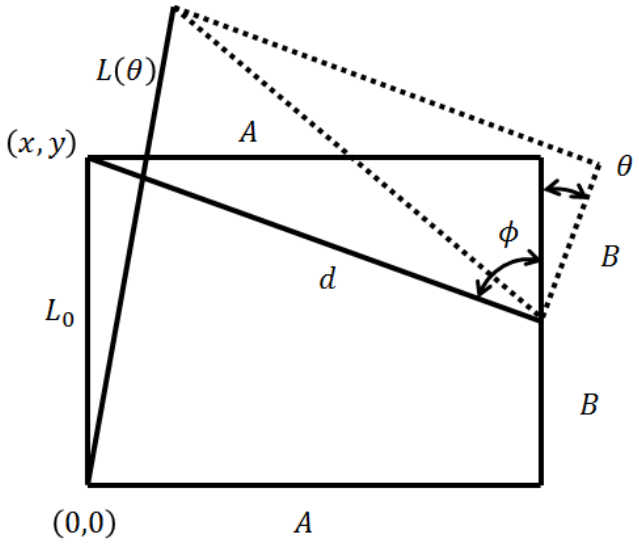


Fig 25. Middle Segment cable geometry

$$\phi = \tan^{-1}\left(\frac{B}{A}\right) = \tan^{-1}\left(\frac{.515[in]}{1.8[in]}\right) = 16°$$

$$d = \sqrt{B^2 + A^2} = \sqrt{.515[in]^2 + 1.8[in]^2} = 1.87[in]$$

$$x_{1,2} = d \cdot \cos(\phi \pm \theta) - A$$

$$y_{1,2} = d \cdot \sin(\phi \pm \theta) + B$$

$$L_{1,2}(\theta) = \sqrt{x^2 + y^2} = \sqrt{(d \cdot \cos(\phi \pm \theta) - A)^2 + (d \cdot \sin(\phi \pm \theta) + B)^2}$$

$$L_{1,2}(\theta) = \sqrt{(1.87[in] \cdot \cos(16° \pm \theta) - 1.8[in])^2 + (1.87[in] \cdot \sin(16° \pm \theta) + .515[in])^2}$$

Where L_1 is the length of the cable shown and L_2 is the length of the opposing cable

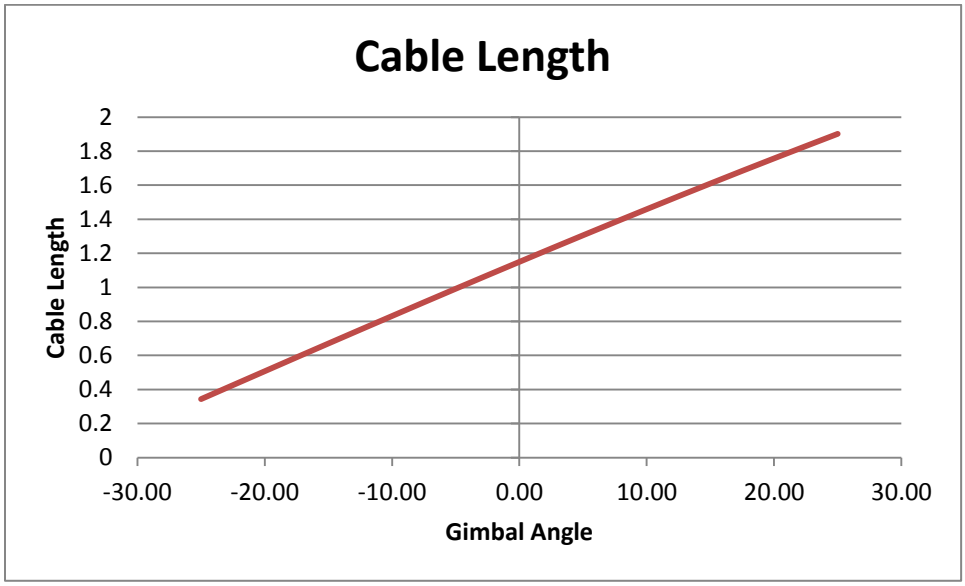


Fig **26**. Cable length of left-hand pulley as a function of gimbal angle

This graph is not a straight line it is a straight segment that can be approximated by the second order function: $L_{1,2}(\theta) = -4 \times 10^{-5}\theta^2 \pm .0312\theta + 1.1498$

The path the cable end makes as the gimbal moves is shown below

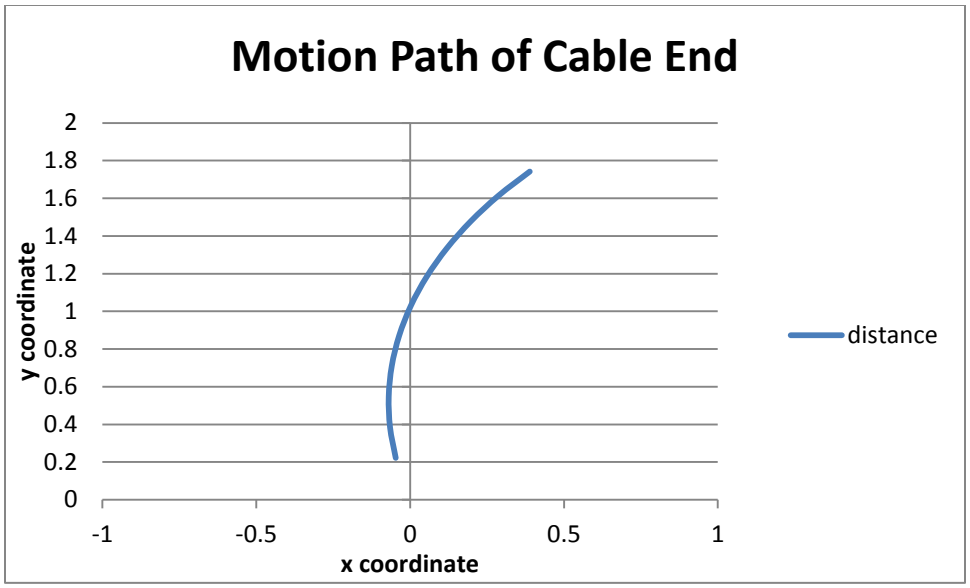


Fig **27**. Path of the endpoint of the cable

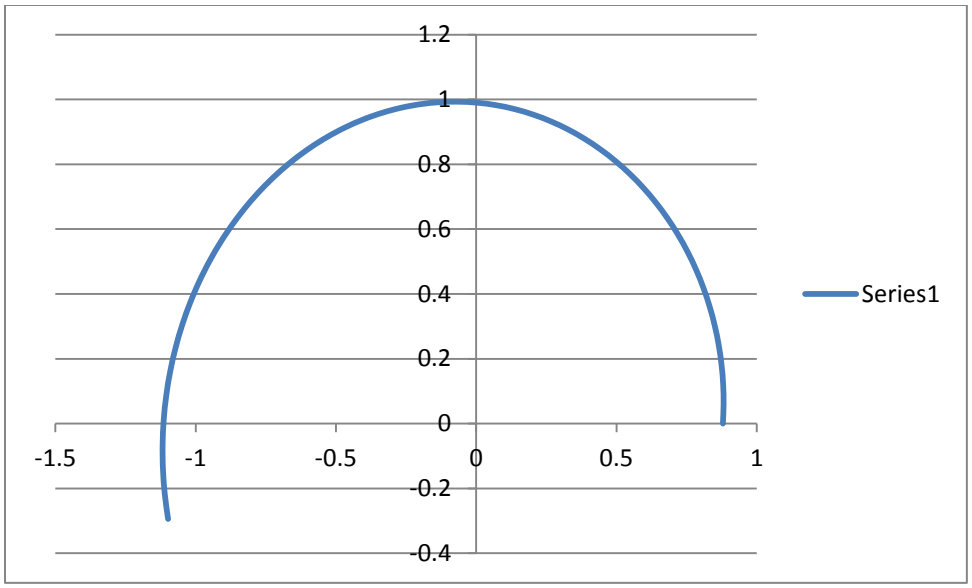
The rate at which the cable length changes is the derivative of the length curve above

$$L'_{1,2}(\theta) = 8 \times 10^{-5}(\theta \pm 390)$$

The ratio of these rates is

$$R = \frac{8 \times 10^{-5}(\theta + 390)}{8 \times 10^{-5}(\theta - 390)}$$

Plotting this ratio in polar space over the full 195° travel of the pulley yields a cam profile that could be used to achieve the speed difference required. One pulley would be a constant radius and the other would have the below profile.



Force Requirement

The torque applied T_{app} $in \cdot lb$ by the moment arm of the lead screw is

$$T_{app} = W \cdot r \cdot cos(\theta)$$

Where W is the weight of the robot in pounds, r is the distance to the center of mass in inches and θ is the rotation of the robot measured from the horizontal.

In the worst case scenario, $\theta \; thus \cos(\theta) = 1$

$$T_{app} = 2.5[lb] \cdot 11.25[in] * 1 = 28.1[in \cdot lb]$$

The output torque T_{out} of the motor transmission is given by:

$$T_{out} = \frac{T_{app}}{A} \cdot r_p$$

Where A is perpendicular the distance from the center of the gimbal to the cable and pulley and $r_p$is the radius of the pulley. A graph of the torque versus angle for three different A values is shown in Fig 28, below.

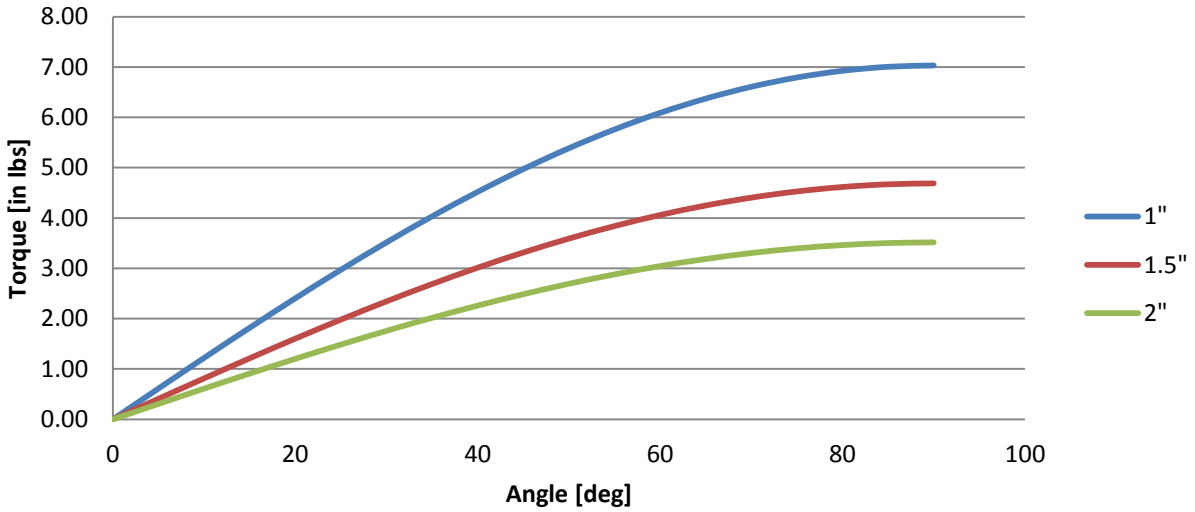


Fig 28. Graph pulley torque needed at different angles of the robot.

$$\mathrm{T}_{out} = \frac{28.1[in \cdot lb]}{1.8[in]} \cdot 0.375[in] = 5.9[in \cdot lb]$$

$$\mathrm{F} = \frac{28.1[in \cdot lb] \cdot \sin(\theta)}{A}$$

The torque required at the pulleys is

The cables are attached 1.8 in from the center of the gimbal joint. This provides a balance between the space required and the amount of force necessary for the motor to provide.

Power Requirement

The gimbal joint must be able to move between extremes in 2 seconds. As shown below, this requires 2.4W of power. Due to their ability to supply the power required and small size, four Pololu micromotors were chosen to power the pulleys.

From the cable length equations above the change in cable length ΔL over the entire rotation can be found 0.78 in

The rotation speed ω_p rpm of the pulley is given by

$$\omega_p = \frac{\Delta L}{2 \cdot \pi \cdot r_p \cdot t} \cdot \frac{60[sec]}{1[min]}$$

Where t is the time required to move from one end point to the other, chosen to be 2 seconds

$$\omega_p = \frac{0.78[\text{in}]}{2 \cdot \pi \cdot 0.375[in] \cdot 2[s]} \cdot \frac{60[sec]}{1[min]} = 9.9[rpm]$$

The output power in W of the transmission is given by:

$$P = \frac{\omega_p \cdot \text{T} \cdot 2 \cdot \pi}{60} \cdot \frac{1[W]}{8.89\left[\frac{in \cdot lb}{s}\right]}$$

$$P_{out} = \frac{9.9[rpm] \cdot 5.9[in \cdot lb] \cdot 2 \cdot \pi}{60} \cdot \frac{1[W]}{8.89\left[\frac{in \cdot lb}{s}\right]} = 0.69[W]$$

To supply the required torque while being anti-backdrive a 24:1 worm gears

The input Torque to the transmission T_{in} in $in \cdot lb$ is given by:

$$\text{T}_{in} = \frac{\text{T}_{out}}{N \cdot \eta}$$

Where N is the torque ratio and eta is the efficiency. Assuming an efficiency of 60%

$$\text{T}_{in} = \frac{5.9[in \cdot lb]}{24 \cdot .6} = 0.41[in \cdot lb]$$

Looking at the performance curve of the Pololu micro motor [Fig 29] the equation for the output speed as a function of output torque is:

$$\omega_{\text{in}} = -170.67\frac{[rpm]}{[in \cdot lb]} \cdot \text{T}_{\text{in}} + 320[rpm]$$

Plugging in T

$$\omega_{\text{in}} = -170.6\frac{[rpm]}{[in \cdot lb]} \cdot 0.41[in \cdot lb] + 320[rpm] = 240[rpm]$$

The output power of the motor P_{in} in watts is given by:

$$P_{in} = \omega_{\text{in}} \cdot \text{T}_{in} \cdot \frac{1[W]}{8.89\left[\frac{in \cdot lb}{s}\right]} = 240[rpm] \cdot 0.41[in \cdot lb] \cdot \frac{1[W]}{8.89\left[\frac{in \cdot lb}{s}\right]} = 1.15[W]$$

Referring once more to the performance curve, the efficiency η_m of the motor at this power is 48% The input power to the motor P_{supply} in watts is given by

$$P_{supply} = \frac{P_{in}}{\eta_m} = \frac{1.15[W]}{.48} = 2.4[W]$$

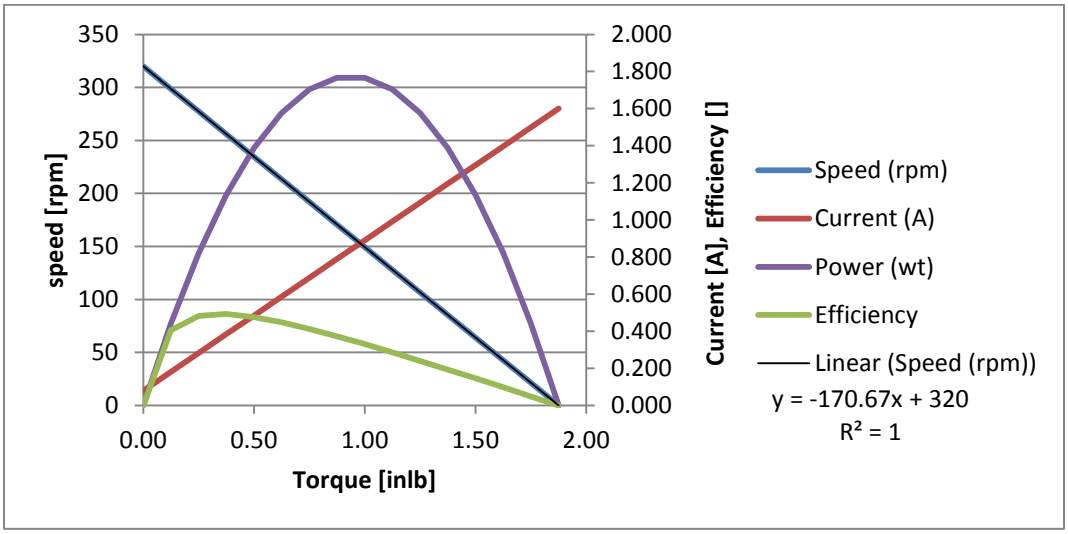


Fig 29**.** Performance curve of the Pololu 100:1 Micro Motor

Electrical Design

In order to keep track of the various positions of the robot, the controller must interface with a multitude of sensors. This ultimately influenced the microcontroller choice. A balance of necessity and complexity is needed to make sure the robot is able to gather enough information to make informed

decisions about its movement, while being limited in complexity by the number of I/O pins. A printed circuit board (PCB) was designed to provide an easy way to interface all of the sensors with the controller, including any extra resistors, capacitors, or integrated chips (ICs) required by the robot

Sensor Choices

Distance Detection

The robot needs to be able to detect its distance from the tree, especially when it is extending and contracting. Infrared range-finders, specifically the Sharp Model 2Y0A02, are used to accomplish this. There is one mounted on each claw, to sense how far off the tree each individual segment of the robot is. This information is used to keep the robot as close to the tree as possible, in order to avoid unnecessary torque on the needles. The design constraint for this involves being able to place the sensors far enough away that they are operating outside of the indeterminate region at the low extreme of their range. The sensor is connected through an ADC to convert the voltage produced to a usable value for the controller.

The robot also needs to know distances to its front and rear, to prevent extending into a situation where movement becomes too constrained and impossible for the robot to get out of without wasting time. For this, ultrasonic range-finders are used instead of infrared range-finders, because there is no risk of sunlight causing false positives with the signal.

In contrast from the easy to connect IR Sensors, the Ultrasonic sensors are more complex. Ultrasonic sensors use a system of echo and trigger signals to function. Sending a quick pulse on the trigger pin causes the sensor to send out a burst of ultrasonic sound. The sensor will then place the echo pin high for the duration that this burst of sound takes to travel to an object and return. The first model considered was the 5-Volt Parallax Ping sensor. This is a three pin sensor, that combined the trigger and echo pins into one. This necessitates rapidly switching the I/O pin between input and output. This is too complex to do reliably, and there was difficulty sensing correct distances using this sensor. The decision was made to use the 5-Volt HC-SR04, by Micropik, which uses a standard four pin design, with dedicated trigger and echo pins. Using this, the robot was able to make far more accurate distance measurements.

The last hurdle for connecting the Ultrasonic sensors to the microcontroller is reliably converting their 5-Volt output signals to 3.3-Volt. The initial plan was to use a voltage divider to divide down to two thirds of the signal; however it was pointed out that this would dissipate an unacceptable amount of power through the resistors. An alternative suggested was using logic-level shifters, which implemented field-effect transistors (FETs) to shift the voltage level. These were a similar cost alternative that would allow reliable translation of voltage levels at a minimal power loss. These level shifters are also used for the same purpose with other sensors on the robot.

Measuring Positioning on the Tree

It is vital for the robot to know its angular position in the world and on the tree. To ensure the robot could measure this, a SparkFun 6-Degree-Of-Freedom (DOF) Inertial Measurement Unit (IMU) is used. This breakout board incorporates both a gyroscope (ITG-3200) and an accelerometer (ADXL345). The accelerometer can be used to determine which direction the ground is in while the robot climbs the tree. This can be used in combination with the gyroscope to sense the orientation of the robot with respect to the ground by evaluating the movement data over time.

The SparkFun IMU was chosen because it not only provides sufficient sensitivity in both the accelerometer (from 2 to 16g's), and the gyroscope (2000°/s) but also interfaces very easily with the microcontroller. The IMU uses an Inter-Integrated Circuit (I2C) bus to send data, and also has two interrupts to inform the controller when to data is ready to be read. This means two pins are needed on the microcontroller to monitor the interrupts and two pins to serve as the I2C bus. The benefit of this is the two pins that serve as the I2C bus can be used for other I2C devices as well.

Measuring Claws and Lead Screw Positions

Potentiometers are used to detect the position of the claws. Originally, current sensing was to be used to check how much current the claw motors were drawing. The more current they used, the closer the claws were to the final position. However, we did not have the space in our initial PCB design to be able to fit enough current sensing ICs. In addition to this, external changes, such as friction, would cause the readings to vary, which could make detecting the correct peak current unreliable. Potentiometers, on

the other hand, could be placed on the axle of the claw, and measure the rotational position of the claw directly. Potentiometers are also used to monitor the angular position of each pulley on the middle segment. From this, the Gyroscope, the accelerometer, and trigonometric equations, the orientation of the robot can be determined. Due to availability and familiarity, the team chose to use common VEX potentiometers. Connecting the potentiometers to the microcontroller is simple; each of them to connects directly to an ADC.

Detecting Claw Impact

To determine if the robot has a good enough grip on the tree, the team decided it to incorporate a system that would inform the operator about which claws had impacted the tree, and possibly how much force they were applying. The initial design used limit switches, since they were easy to configure, and provide a binary value for contact with the tree. Several concerns prompted moving away from this design: fear over the robustness of the switch having to survive many impacts, the fact that the needles would be somewhat loose inside the foot, and the fact they convey no information about the force with which the claw impacted the tree.

The alternative is to use strain gauges, which allow the robot to see which claws had impacted, due to a spike in their voltage signal. The force the claws should be experiencing when they are attached to the tree is known. Comparing the actual and expected values the software can determine if the robot's grip on the tree is powerful enough, or if it will need to re-grip the tree. However, this setup necessitates more complicated circuitry than the limit switch required. The strain gauge must be placed in a Wheatstone bridge, to pull out how much its voltage has changed in response to the strain. The output of this Wheatstone bridge is then placed through differential amplification. The end result is a near 0 volt signal means there is virtually no strain on the gauge, and the closer the signal gets to 3.3-volts, the larger the strain on the gauge is. This data is sent to the controller through an ADC.

The strain gauge circuits do not fit on the PCB, due to their complexity and number. In order to resolve this, the team decided to use additional PCBs. The strain PCBs possess a self-contained signal amplifier for a single strain gauge, and have pins to output the signal to the main PCB. It was discovered

that further space could be saved by placing the amplifier circuits for two strain gauges on the same PCB and this became the final design.

Microcontroller Selection: Raspberry Pi B+

The team compared many different controllers before choosing the final controller, including the Raspberry Pi B+, the Beaglebone Black, and the Arduino Mega 2560. There were many factors which ultimately played into the decision to choose the Raspberry Pi B+. The first being many different sensors had to interface with the controller, this requires multiple I/O ports. In addition, ICs which could only communicate over SPI and I2C protocols are used thus the controller must support both of these, especially in a usable software library. The robot has a total of 22 different sensors, 16 of which are analog and six of which are digital. In addition to the sensors, there are seven different motors to control. H-Bridges are used to control the motors, this means 3 to 5 pins on the controller are needed for each motor, totaling 23 pins on their own. Table 1 shows the breakdown between how many of each type of pin required, and how many each controller provides.

Table 1: Comparison between Pins Required, and Pins provided by specific Microcontrollers

	Required	Raspberry Pi B+	Beaglebone Black	Arduino Mega 2560
Analog Pins	16	0	7	16
SPI Pins	3	5	3	3
I2C Pins	2	2	2	2
Additional Pins	37	21	61	49

All three microcontrollers have the ability to use SPI and I2C, and these features are supported by software libraries. In this sense, the Arduino Mega 2560 would be the best choice because it has more than enough additional pins and supports 16 analog sensors. However, using Analog to Digital Converter (ADC) IC's (chip: MCP3008), the Raspberry Pi B+ and the Beaglebone Black are both able to support

more analog sensors. In addition, a GPIO expander IC (chip: MCP23017) allows both the Raspberry Pi B+ and the Arduino Mega 2560 to have more additional pin I/O. Overall, the only extra cost to using one of these microcontrollers over the other is the necessity of additional ICs. Due to this, specific pin usage weighed very lightly on the final decision.

The second factor in determining the microcontroller was the ability to incorporate a lightweight camera. Two of the larger goals of the project were to allow the robot to be controlled while out of sight of the operator and to aid in the detection of the ALB. To accomplish this, the team decided a camera would be the best option for both goals. Due to weight concerns, the camera had to be lightweight, but also able to capture at a decent resolution. The Raspberry Pi B+ has the ability to use a Raspberry Pi specific camera. The camera has the capability for 5 megapixel images, and supports 1080p at 30 frames per second, and 720p at 60 frames per second. In addition, the camera has a built in, intuitive library for using the camera. The Beaglebone Black has a camera attachment that takes images at 3.1 megapixels, and supports 720p at 60 frames per second. However, the cost of this camera is almost twice that of the Raspberry Pi Camera. This camera is not supported by a dedicated software library, except for one written for android, which would not be run on the Beaglebone. The Arduino Mega 2560 does not have the ability to use a camera, unless the camera connects to the Arduino through the USB 2 Type A protocol. This means finding a commercial camera for the Arduino, and finding libraries to support it written in C. In accordance with these metrics, the Raspberry Pi B+, was selected for its better camera capabilities and the convenience of a pre-written software library.

The third concern for the microprocessor was the cost. The team already had an Arduino Mega 2560 from the previous MQP, bringing the comparison for price down to the Raspberry Pi B+ and the Beaglebone Black. The average costs for these boards are 35 and 55 dollars respectively. Table 2 shows the final breakdown between our criterions:

Table 2. Microcontroller comparison

	Raspberry Pi B+	Beaglebone Black	Arduino Mega 2560
Has required Pins?	No. Requires additional ADC and I/O expander IC's	No. Requires additional ADC IC's	Yes
Camera Option	5 Megapixel 1080p @ 30fps 720p @ 60fps $27	3.1 Megapixel 720p @ 60fps $50	No
WiFi Network Support	Yes	Yes	No
Cost	$35	$55	Free

The chart above shows the Arduino is ruled out due to not supporting a camera or WiFi communications. Comparing the Beaglebone and the Raspberry Pi, the Raspberry Pi is cheaper and offers a better camera. The Raspberry Pi has the added bonus that the camera is supported by software libraries, making it easier to integrate.

The final and most important factor, is the ability to connect to an operator's computer. This includes allowing the robot to interface with the operator's computer as well as allowing for quick transfer of the camera images. At minimum, the team wanted to stream 640x480 pixel images at 30 frames per second, through a wireless protocol. This requires a throughput of 9216000 bit/s, or 9.2 Mbit/s. Common Bluetooth technologies, such as XBee, have maximum transfer rates up to 250 kbit/s, which is far lower than was needed. By comparison, some WiFi transfer rates can exceed 500 Mbit/s. This led the team to use WiFi to connect to a wireless network between the microcontroller and the operator's computer. The Arduino Mega 2560 is unable to communicate over a WiFi network, because a wireless dongle requires the use of USB ports which the Arduino lacks. The Beaglebone Black has the ability to use USB WiFi dongles to communicate with wireless networks with its singular USB port. The Raspberry Pi B+ also has the capability to use USB WiFi dongle, and it has four total USB ports. Another factor to consider is the ability to create a network, rather than join one. By creating a network, the robot becomes independent of the operator's computer, and any computer is compatible with the robot. This eliminates

the need to attach a keyboard and monitor to the Rasberry Pi to connect it to the network. Both the Beaglebone and the Raspberry Pi have operating system support for creating and broadcasting a network, given the proper wireless chip.

Due to these factors, the team decided to use the Raspberry Pi. It will be broadcasting its own wireless network. The lack of necessary pins will be augmented by two MCP3008 (ADC ICs) and one MCP23017 (I/O Expander IC) to allow it to interface with all of the sensors required.

Printed Circuit Board Design

The complexity of the circuitry needed to interface the Raspberry Pi with the rest of the robot indicates a Printed Circuit Board (PCB) would be optimal to conserve weight and space. The first iteration of the board did well in connecting signal traces to the Raspberry Pi, but accomplished little more than that. Fig 30 shows the early concept of the PCB.

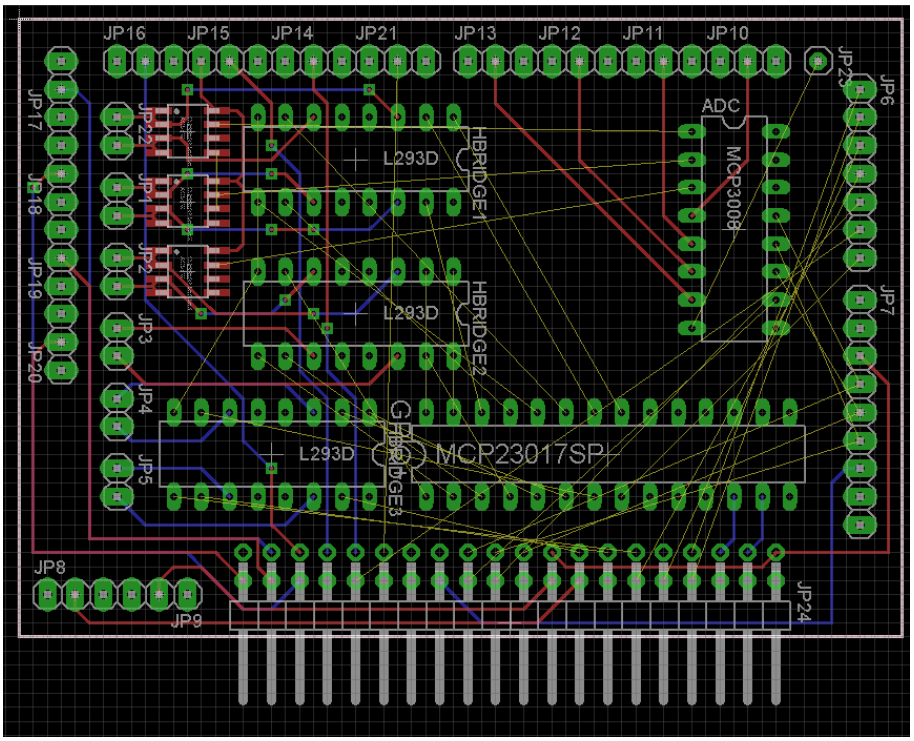


Fig **30**. An early design for the PCB layout

This design has poor routing and planning, and a sub-optimal layout for pin headers. The way space is used on the board is haphazard, and it was learned that larger through hole chips were less ideal in a PCB setting compared to surface mounted chips. Working on this first design brought up the issues of proper selection of parts in PCB layout software, lack of mounting holes, and no use of copper pours.

The second iteration of design incorporated mounting holes and a fixed size which would be the basis for future board iterations. In addition, the H-Bridge chips were replaced with a SOIC-16 surface mount chip, the MCP3008 replaced with a SOIC-16 surface mount version, and the MCP23017 replaced with a SOIC-28 version of the chip. This created more space on the board to route traces and place pin headers. This version of the design was reviewed by a fellow student who had more experience in designing PCBs. The review highlighted new problems in the design. First, the trace width in the power traces was too narrow to transfer the correct amount of current to all of the chips on the board. The width was 0.01 in, and could supply only 1 amp of current. This was not enough, as the motors together would be drawing upwards of 2-3 amps of current. It was recommended to use a trace width calculator to find the optimal width, which for the largest traces is five times larger than the traces originally were. Testing that occurred between the second and third iterations of the board showed the H-Bridge chip selected could not supply enough current to the motors it was driving. It was found that using a DRV8838 and DRV8835 chips would drive motors at 1.8 amps and 3 amps respectively at peak draw. This suited the requirements, and these chips were incorporated in the third iteration.

Work on the third iteration of the main PCB, the team led the team to use breakout PCBs for our strain gauge sensors to allow a large amount of space to be saved on the main PCB. Fig 31 shows the final design for the strain gauge boards, the left-hand side shows a breakout board accommodating a single strain gauge circuit, and the right-hand side shows a board accommodating two circuits, with a shared power and ground to reduce the number of wires and pin headers used. The team chose to use the dual-circuit design due to its smaller size and easier wire routing in the cable harness. This change allowed for a more efficient layout in the third iteration of the PCB design.

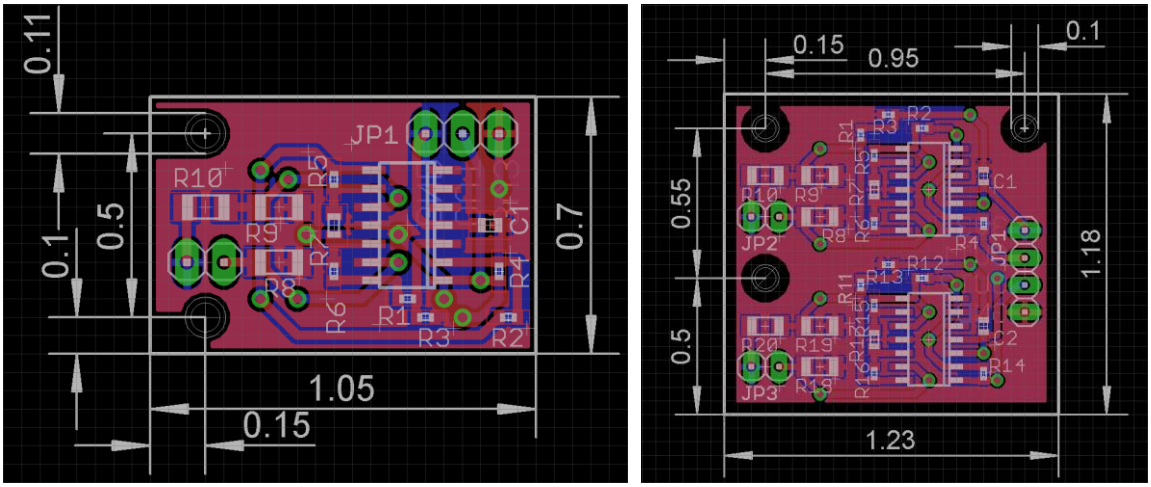


Fig **31**. Strain Gauge Breakout Board Design

The third iteration saw improvements in space but found difficulty in routing power traces efficiently enough. The suggestion given was to use plated holes with wires soldered in to route the voltage off the board instead of on the board itself. It essentially creates a third layer on the PCB without worrying about overlapping traces. This change brought about the fourth and final iteration of our PCB design, shown in Fig 28. This image shows the board without the copper pours that give it a power and a ground plane, however it does show the trace routing and the tight design necessitated by the size of the board.

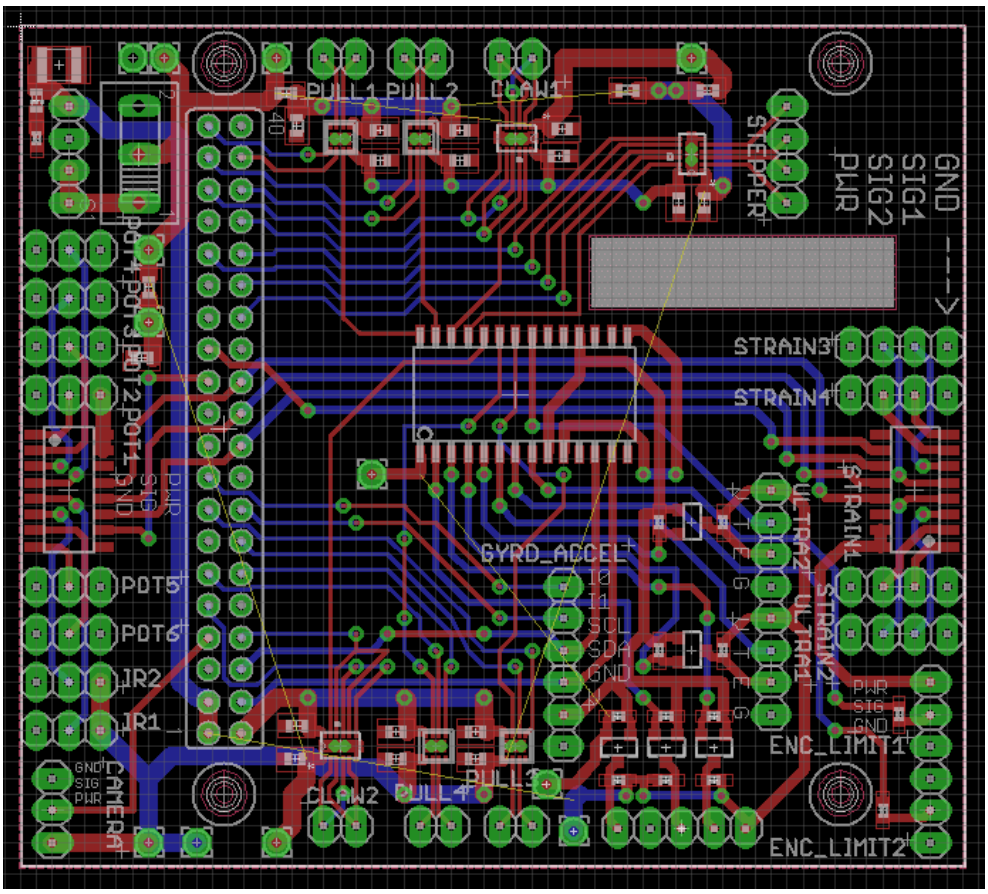


Fig 28. Final PCB Design

This final design has a few flaws that were found during testing its perforated board equivalent. The first being the lack of a unity gain buffer on the voltage dividers. Without this buffer, the motors connected to these links act as a load in the voltage divider circuit, which causes the voltage to drop to an undesired value which cannot power the motors. Also, two of the pins on the Raspberry Pi which are connected to H-bridges are unusable by the software. They are reserved by the Raspberry Pi for other uses, this was corrected by re-soldering the wires to other pins. This board also requires a 2 oz/in^2 copper thickness to get the required current to the motors, instead of the usual 1 oz/in^2. This is a problem which comes from the way the DRV8838 and DRV8835 H-bridge chips are laid out. Optimally, the output of these chips would be connected to a larger trace plane, instead of a single trace line. These problems would have to be fixed before connecting this board to the Raspberry Pi to get desired performance.

The team wanted to order the PCB after it was designed but the purchase was canceled. This is due to being unable to get a sponsorship for the project and receiving very high priced quotes from the company. The board's 2 oz/in^2 copper thickness requirement excludes it from matching deals offered by the company. Due to the inability to purchase a PCB, the team prototyped a perforated board instead. While this board is significantly larger than the PCB would have been it is functionally the same.

Software Design

The software design is based on two main concepts. The first is to make the robot as easy and intuitive as possible for the operator to control. The second is to make the code modular for easy modification and testing. Every major decision regarding the code took these two concepts into consideration.

The software is divided into two main parts at a high level, the operator code and the robot code. The operator code, running on their laptop, is responsible for controlling all of the interactions between the driver and the robot through a Graphical User Interface (GUI). The robot code, located on the Raspberry Pi, is responsible for controlling the robot's movement and gathering sensor information about the robot. The two parts of the code communicate through a wireless network hosted on the Raspberry Pi.

Operator Code

The operator code is designed to make the robot controls easy for the user to comprehend. It contains a simplistic GUI, shown in Fig 32, for interactions with the robot. The window of the GUI is divided into three main parts: camera view, robot graphic, and control panel.

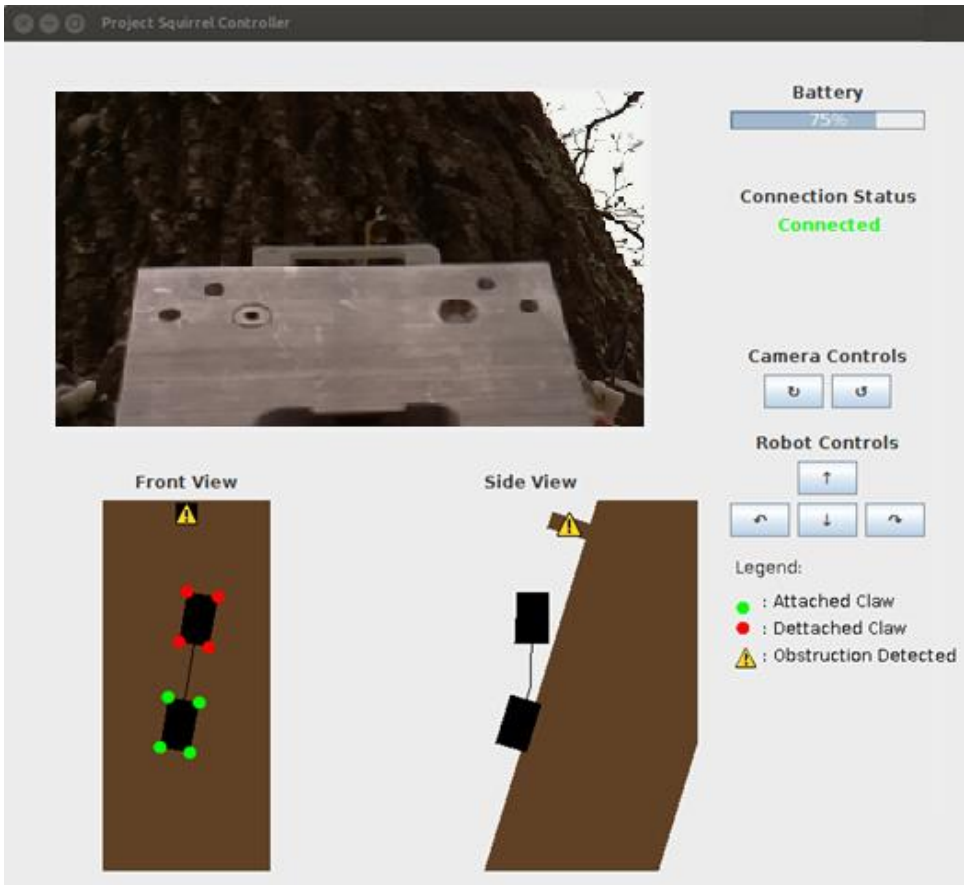


Fig 32. Graphical User Interface

The camera view displays a live feed from the Raspberry Pi camera on the robot. This occupies the top left portion of the window. It allows the operator to easily view the area around the robot to find evidence of ALB and obstacles the sensors are unable to detect.

The robot graphic contains a two dimensional model of the robot located in the bottom left portion of the window. The model gives the operator a straightforward view of the robot's current state the camera is not able to provide. It displays the current orientation of the robot relative to the tree, the claws attachment status to the tree, and the distance to any obstructions detected.

The control panel occupies the right side of the window. The number of buttons is kept to a minimum to make the controls easy to use. There are two buttons for rotating the camera and four buttons for controlling the robot's gait. In addition to these buttons, the control panel also contains information concerning the battery level and connection status.

The operator code was written in Java primarily using Swing, a GUI widget toolkit for Java. This was chosen for its extensive Application Programming Interface. It provides libraries with pre-built components that make it easy to build a GUI.

The GUI's code was written using the Model View Presenter (MVP) design pattern, a common solution for building GUI applications. There are three parts: the Model, the View, and the Presenter. Each part provides a different abstraction for the code base. A general overview of this pattern can be seen in Fig 33.

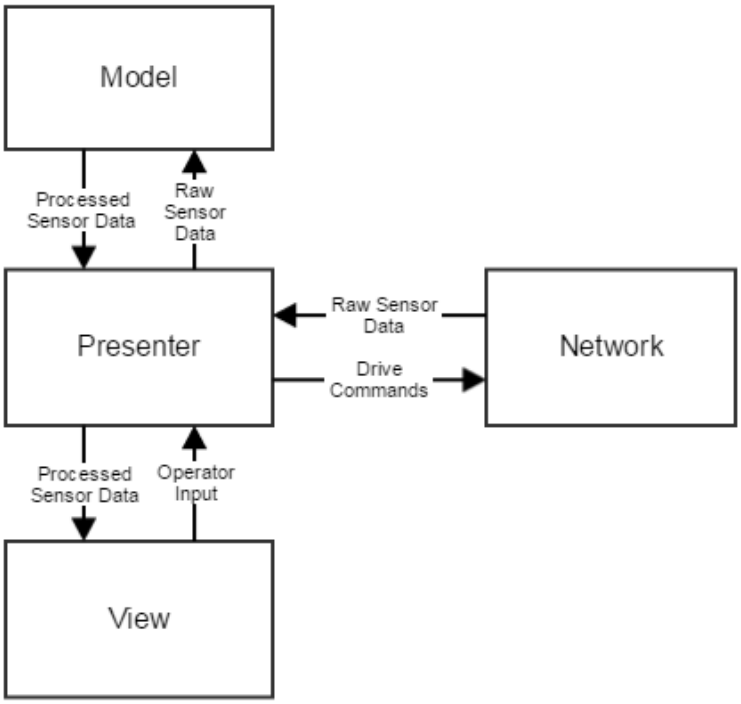


Fig 33. MVP Diagram

The Model is responsible for structuring and storing data. In the operator code, the data stored is the sensor information sent by the robot. The Model is forwarded this data from the Presenter, processes it if necessary, and notifies the Presenter of the processed data.

The View is responsible for displaying data and receiving input from the operator. It receives the processed data from the Presenter and sends operator input to it. In the operator code there are three main views corresponding to the parts of the window mentioned above.

The Presenter is responsible for interactions between the Model and the View. It sends new information to the Model and receives processed data. The Presenter then updates the View with this processed data and handles the operator input. In the case of the operator code, the new information received is sensor data from the robot and the operator input comes from the control commands.

In addition to the MVP, there is additional code to handle wireless network interaction with the robot. This code sends information received by the wireless network to the Presenter. Camera and robot commands are received from the Presenter and sent to the wireless network.

Robot Code

The robot code is written in Python. It runs on the Raspberry Pi using Python libraries. The libraries provide a high level abstraction for controlling the low level General Purpose Input/Output (GPIO) on the microcontroller. This makes the code more readable and easier to write.

The code runs in two processes: the main process and the camera process. A summary of these processes is shown in Fig 34. The main process controls the majority of the robot. It gathers sensor information from the GPIO, receives commands from the wireless network, and executes the gait based on the commands sent. The camera process is devoted to capturing frames from the Raspberry Pi Camera and sending them to the wireless network. It is run in its own process to maximize the speed at which camera images are streamed.

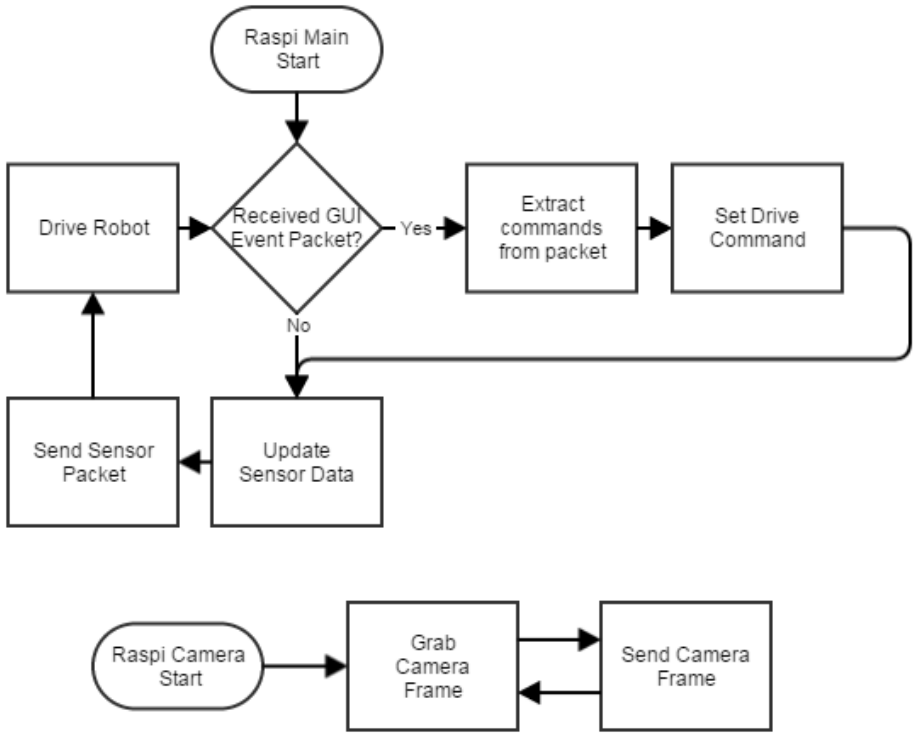


Fig 34. Raspberry Pi processes

The work is divided among three main classes within the main loop: the network manager, the sensor manager, and the drive manager. The network manager is responsible for sending sensor information to and receiving drive commands from the wireless network. The sensor manager is responsible for monitoring the GPIO and is called whenever information the sensors are used to take data. Finally, the drive manager is responsible for controlling the camera rotation and determining the correct gait step based on the most recent command and the current sensor information.

The robot has four different gaits: forwards, backwards, left, and right. These four gaits contain eight possible steps. These steps include attaching or detaching the claws, turning the middle segment, and extending or retracting the lead screw. While moving the lead screw, the double gimbal is also rotated towards or away from the tree in order to maintain a set distance from the tree. The gait is chosen based on the most recent drive command and the current step of the gait is dependent on the status of the claws

and lead screw. There are soft limits programmed at each step to ensure a motor does not drive past its limit.

The logic for choosing the next step of the forward and backward gaits is shown in Fig 35. The gaits are identical other than the direction the lead screw moves. Each gait can be divided into three parts dependent on which set of claws is attached.

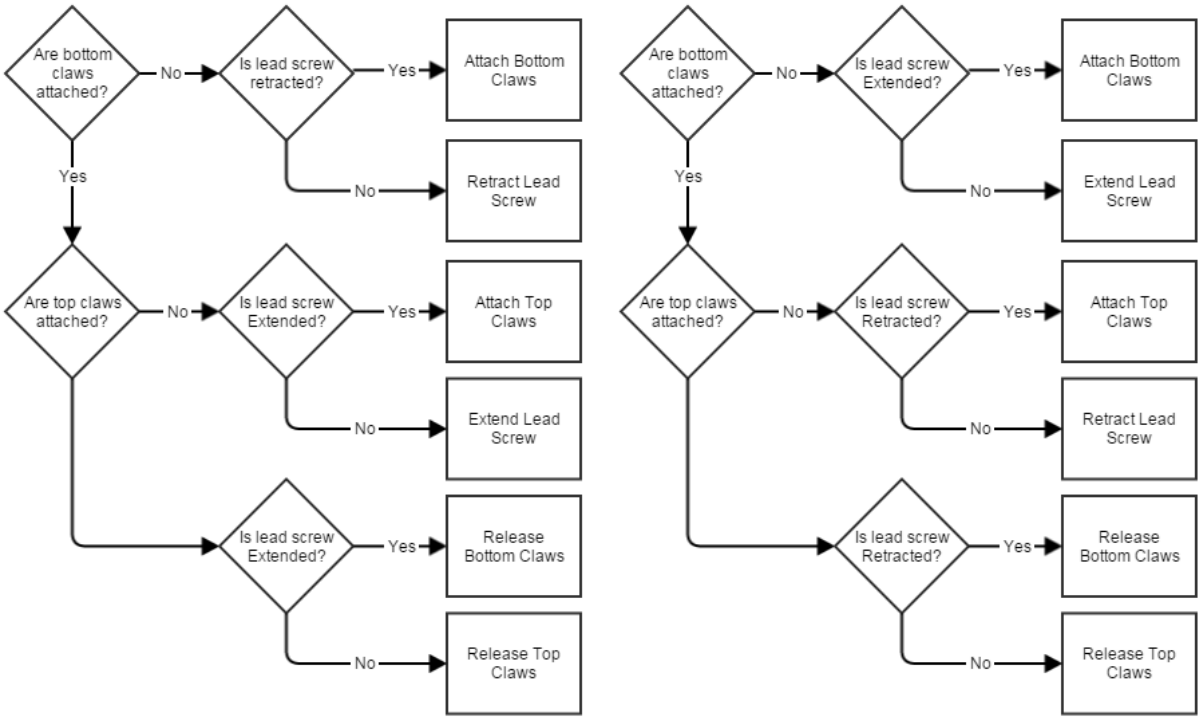


Fig 35. Forward Gait (left) and Backward Gait (right)

First, the bottom claws are checked to see if they are attached. If they are not attached, then the lead screw position is checked. If the lead screw is in the correct position it attaches the bottom claws, else it moves the lead screw towards the correct position. The correct position of the lead screw depends on which gait is being executed.

If the bottom claws are attached, then the top claws are checked. If they are not attached, then similar to the first part of the gait, the lead screw position is checked. If the lead screw is in the correct position it attaches the top claws, else it moves the lead screw towards the correct position.

If both sets of claws are attached then the lead screw position is checked. Based on the lead screw position and current gait, either the top or the bottom claws are released.

The logic for choosing the next step in the left and right gaits is less complex. It is shown in Fig 36. If the bottom claws are not attached, then they will be attached. If the bottom claws are attached and the top claws are attached then the top claws are released. If the bottom claws are attached and the top claws are released, the middle segment joint is turned left or right based on the current gait.

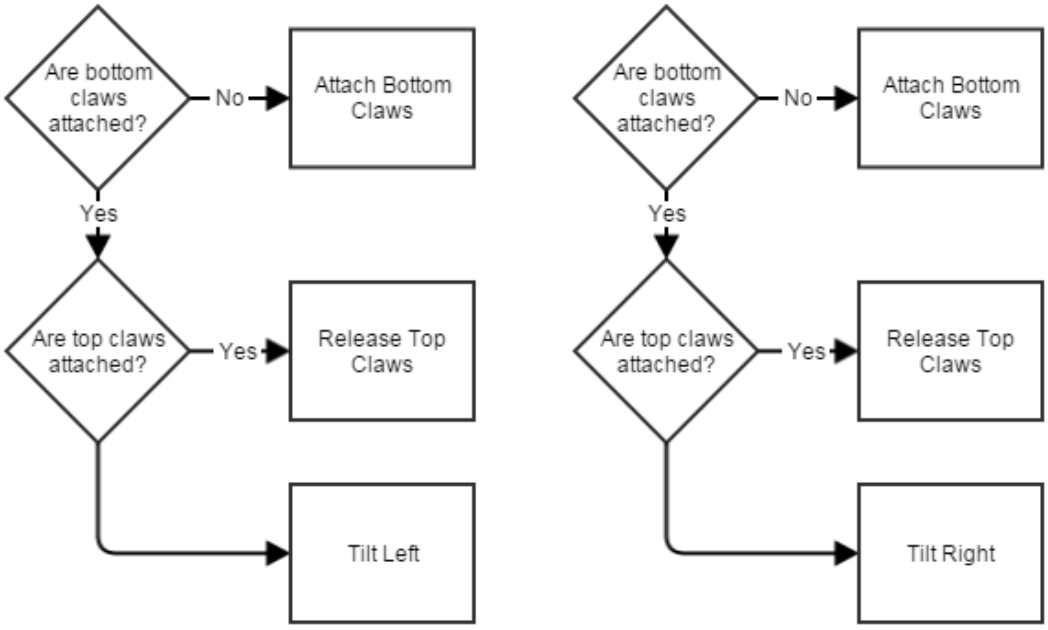


Fig 36. Left Gait (left) and Right Gait (right)

Communication between Operator and Robot Code

The operator code and the robot code communicate through a wireless network hosted on the Raspberry Pi. This was chosen over other wireless methods due to its larger signal range and bandwidth. The network is set up using a client-server architecture, with the client in the operator code and the server in the robot code. The client connects to the server over two separate ports on the same IP address. A socket is set up on each port for passing data. One socket is used for the camera stream and the other is the main socket, used for sensor data and drive commands.

Communication over the camera socket occurs in one direction. The camera stream sends as a series of images from the server to the client after the client connects to the server. Each image is sent as a JPEG file. The byte size of the file is sent as a ten digit number before the file is sent. The client reads the size and then the number of bytes specified by the size.

Communication over the main socket occurs in both directions. The client sends drive commands to the server and the server sends sensor data to the client. Both the sensor data and the commands are encoded in JavaScript Object Notation (JSON). Each packet of JSON data is sent as a string followed by a newline character.

Testing Procedure

There are two main areas of testing for the project: the robot and the GUI. Testing for the robot involves ensuring each part was working correctly individually and ensuring they could interact together to form the gait. Testing for the GUI involves validating communication to and from the robot. A full robot control test is then performed using the GUI to navigate the robot up and down a tree after the GUI and robot are tested individually.

Robot Testing

Validation of the robot is divided into five sets of tests. Each group of tests is run using a collection of scripts that control various aspects of the robot with minimal human interaction. Each of the scripts imports commands from the main robot code base. The five sets of tests were the following:

1. Preliminary sensor and motor
2. Claw
3. Lead screw
4. Gimbal
5. Gait

Preliminary sensor and motor testing involves testing each sensor and motor individually to ensure they can be controlled directly from software. A script is run for each sensor that continuously prints the value of the sensor while a human changes its state manually. The test is considered successful if the expected values are printed correctly for each of the sensor states. The claw motors are tested by a script that runs them at maximum speed in one direction. The test is considered successful if the claw motors ran in the correct direction without incident. The lead screw motor is tested by a script that runs it in both directions at a constant speed. The test is considered successful if the motor can be driven in both directions. The gimbal motors are tested by a script that runs them in both directions at two different

speeds. The test is considered successful if the motors are able to run in different directions at different speeds.

Claw testing involves ensuring the motors operate in collaboration with the sensors on the claw in order to attach and detach from the tree. Each claw is run by a script that opens the legs to a preset angle, pauses for five seconds, and attaches the legs into the tree. The script is then run again to ensure the legs can detach from the tree. The test has two success conditions. The first is the claws legs can be attached using the script and can hold the robot on a vertical tree in the worst case scenario of the upper part of the robot leaning away from the tree. The second is the legs can be detached from the tree by rerunning script.

Lead screw testing involves retracting and extending the lead screw while the robot is rotated at 0, 45, and 90 degrees relative to the ground. The script for this test retracts the lead screw for a set distance, pauses for five seconds, and extends the lead screw the same distance. The test is considered successful if the lead screw can be successfully be extended and retracted at each angle.

Gimbal joint testing involves ensuring it is able to be driven to various angles for pitch and yaw while supporting the full weight of the robot. The angles driven to are 5, 15, and 25 degrees from vertical in both directions. A script is run to drive the gimbal to each angle. This test is considered successful if the gimbal can drive to these angles while the robot is in a horizontal position with respect to the ground. In addition, the gimbal will have to hold the robot in position within .5 degrees to maintain a less than 2% error for positioning the robot to attach the claws.

Gait testing involves making sure that the claws, lead screw, and gimbal can work together to run the forward gait. A script is run that attaches the robot to the tree by opening both claws, pausing for 5 seconds to allow positioning by a person, and finally closing both claws. Using a second script, the following sequence is executed in triplicate:

1. Release back claw
2. Rotate gimbal 5 degrees away from tree
3. Retract lead screw set distance
4. Rotate gimbal pitch back to starting position

5. Attach back claw
6. Release front claw
7. Rotate gimbal 5 degrees away from tree
8. Extend lead screw set distance
9. Rotate gimbal pitch back to starting position
10. Attach front claw

The test is considered successful if the robot can make it through the sequence all three times without falling off.

GUI testing

Testing for the GUI is divided into three parts:

1. Testing the display of robot state and camera data to the operator.
2. Testing the execution of robot commands sent from the operator.
3. Testing the ease and intuition with which the operator could make use of the GUI.

The first two parts involve passing data back and forth over the wireless network and observing the reaction of the GUI and robot. The third involves having people interact with GUI and report their experience.

The first test involves sending a live stream from the Raspberry Pi camera over one socket while sending preset robot data over the other with a 5 second delay between each transmission. The preset data represented the robot in various positions. The different sets of preset values that were sent are shown in Table 3. There are two success conditions to the test. The GUI displays the camera stream with less than 200ms of lag time and the GUI displays the expected graphics and values for each set of data sent.

Table 3: Robot state data relayed during the camera test

Front Angle to Ground (deg.)	Yaw (deg.)	Side Angle to Ground (deg.)	Pitch (deg.)	Extension (in.)	Claw IDs attached?	Battery % Left	Length to Top Obstruction (in.)	Length to Bottom Obstruction (in.)
0	0	0	0	0	1,2,3,4	0	0	0

90	90	90	90	5	5,6,7,8	5	5	5
180	180	180	180	25	None	75	25	25
270	270	270	270	90	1,2	95	90	90
360	360	360	360	100	5,7	100	100	100
45	45	45	45	500	1,2,3,4,5,6,7,8	200	500	500
-45	-45	-45	-45	-10	-2,1,8,10	-5	-10	-10

The second test involves sending commands from the GUI to the robot. The command received by the GUI is printed by the robot code. The test is considered successful if the correct command for each button was printed when pressed.

The third test involves having fellow students interact with the GUI and report the quality of their experience. They are asked subjective questions about the usability of the interface. The first set of questions is on a scale of one to five about how intuitive the interface is. The questions are divided into two groups: positive and negative, scores of five on positive questions and scores of one on negative questions indicated an easier and more intuitive GUI. The next group of questions is what they did and did not like about the GUI and what could be done to improve it. The GUI was shown to a USDA representative for additional feedback on the design.

Full robot control

Full robot control involves two tests. Both tests use the GUI to drive the robot remotely 10 feet up and down a tree. The first test is on a vertical tree with no branches. The second is carried out on a vertical tree with at least one branch in the path of the robot. Both tests are considered successful if the robot is able to climb up and down the tree without any human interaction aside from the GUI.

Results

Robot

The preliminary tests showed every motor was able to be driven successfully. The potentiometers and limit switches displayed correct values for all of their states. The strain gauges, ultrasonic sensors, IR sensors, and IMU have not been fully tested due to time constraints

The tests for both claws were successful. The legs were able to open to a specified angle, pause for 5 seconds, and attach themselves to the tree. One set of legs was able to hold the robot in position in the worst case scenario of a pitch angle of 25 degrees on a vertical tree. The claws were then able to detach after holding the robot on the tree.

The test for the lead screw was unsuccessful. Torsion in the system along the axis of the lead screw led to binding which caused the motor to stall. Therefore, the lead screw mechanism could not drive in either direction.

The test for the gimbal was partially successful. The gimbal was able to drive to the angles specified. There was slack left over in the pulley mechanism and the gimbal was not able to hold the robot within .5 degrees of the specified angles. This slack can be attributed to incorrect translation of the gimbal geometry to the driving equations and slight deviation of the gimbal geometry from the CAD.

Due to the lead screw not working the gait test was only partially successful. The robot moved appropriately at each step with the exception of the extension and contraction.

GUI

The GUI was successfully able to receive the camera stream and placeholder sensor data sent from the robot. The camera stream displayed with a lag time estimated by viewers to be less than 200ms. Additionally, the GUI displayed the expected graphics and values for each set of placeholder data. The GUI was also able to successfully send drive commands to the robot. For each button pressed the correct command was printed by the robot code on the Raspberry Pi.

The WPI students surveyed found the GUI easy and intuitive to use. The survey resulted in an average score of 4.17 for positive questions and 1.53 for negative questions. For the open response questions, they suggested adding more labels to the robot graphic. In addition to the WPI students' survey feedback, the USDA representative commented he would be able to train employees in under a day to drive the robot using the GUI. The representative did not suggest any improvements to the GUI.

Full Robot Control

Issues with binding with the lead screw prevented the robot from being able to climb a tree. Therefore, the tests for full robot control were considered unsuccessful.

Conclusions

The binding in the lead screw mechanism prevented many of the goals from being met. The team was unable to meet any secondary goals as well as the primary goals of climbing trees of varying diameters, climbing at a velocity of 2.5 ft/min, climbing up and down a vertical tree, and avoiding branches. However, given a lead screw mechanism with more support, the team believes these goals could have been met.

Despite the issues, several of the goals were met. The robot is able to maintain position on the tree while testing the claws. It is also able to do this without doing any damage to the tree. The results also support the GUI being easy and intuitive to use. The GUI was also successful in displaying the camera stream. However, a larger variety of users could have been used to get a more accurate description on how easy and intuitive the GUI was to use. In addition, they were significant limitations to their interactions with GUI since they could not directly drive the robot using the GUI due to the lead screw binding issue.

Testing determined the opposing pulleys for the gimbal joint do not need to be run at different speeds. This is likely do to the ability of the cable to stretch and absorb the small differences in the cable speed. Analyzing the relative rates of the pulleys shows the speeds differ by only ±1%. The decision to use four motors on this joint was made based on incorrect math which suggested the difference in speeds needed was much greater.

The team learned the importance of off-board peripherals when designing the PCB for the robot. There was not enough room on the main board for the strain gauge measurement circuits. These smaller circuits were off-loaded to auxiliary boards located near each leg. This not only reduced clutter on the board but reduced the length of cable needed to service the strain gauges. The PCB was never ordered due to its high cost. The likely cause of the price was the large number of plated holes for the header pins.

Future Work and Recommendations

Torque Release Mechanism

The plungers are only supported over a short length, this combined with the forces on them means there is a high likelihood they will bind. Extending the end of the plunger and using the set screw as a bushing would reduce the binding risk to near zero [Fig 37].

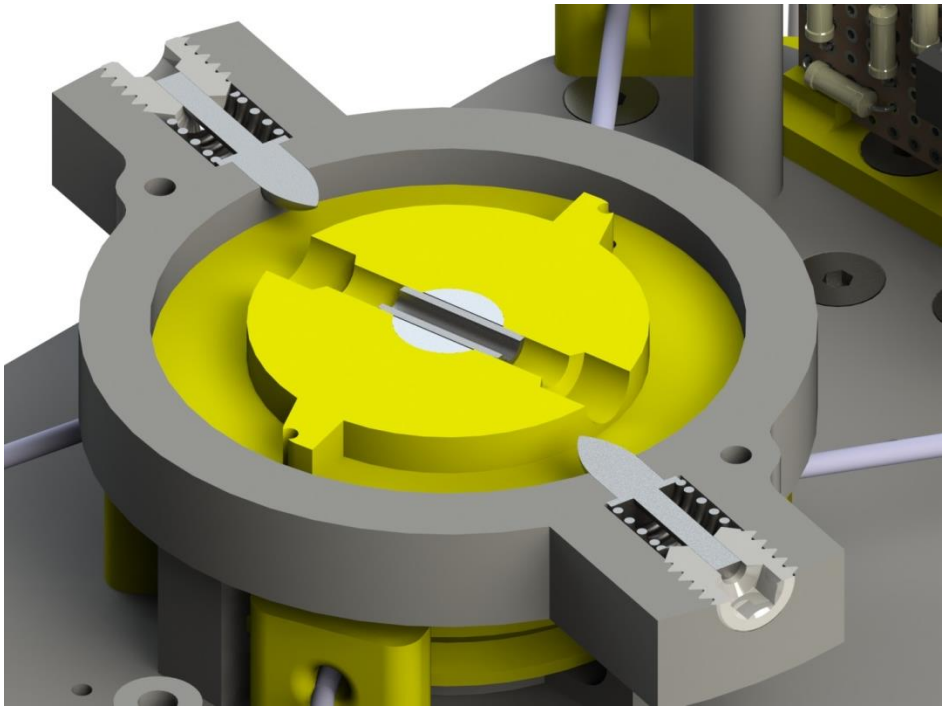


Fig 37. Torque release mechanism revised to prevent binding

Using a mechanism that releases on torque and not position introduces a potential problem. If the force required to extract the needles from the tree is greater than expected, the torque release mechanism would not be able to remove them. Without changing the entire mechanism, the efficacy of the cable system could be increased by changing the path of the cable such that it is tangent to the rotation of the claw along the entire motion path .

Strain Gauges

The strain gauges are a very useful and innovative mechanism but they have some drawbacks. They are not very robust and may break under standard robot operation. Also, once the legs are assembled with the strain gauges it becomes nearly impossible to take the feet off the legs without damaging the strain gauges. This is due to the fact that the steel piece bonded to the strain gauge stays in the foot when pulled apart rather than the leg while the wire leads stay on the leg. Looking at this design from a functional perspective, the foot actuation and stiffness of the load cell could be tuned such that the claw would be more sensitive to force on the tree. The sensitivity could be increased electrically by altering the gain of the instrumentation amplifier, but this will add extra noise that could result in a false trigger.

Middle Segment

Movement of the middle segment could be improved by reducing the number of motors needed for the gimbal joint. The opposing motors being run at different speeds causes problematic controls. One way this could be achieved is to directly power the gimbal joint itself. This would have a significantly greater torque requirement but would allow for the removal of the cables and pulleys.

In addition to reducing the amount of motors in the gimbal joint, the middle segment could be improved by adding additional support to the lead screw by adding more guides. Currently, there is only one guide to prevent the robot from twisting about the lead screw. Based on the results of testing, this one guide is not enough to fully prevent torsion. This causes binding in the motor which consequently stalls when trying to drive the robot. Adding additional guides for support could potentially reduce the amount of twisting and therefore the binding as well. Correcting this problem is a major step in completing the climbing related goals the robot did not achieve.

Electrical

The design of the PCB could be improved by changing several aspects. First, the addition of unity buffers on the traces leading to the motors would improve motor reliability. A voltage regulator would further increase the performance of the motors, by consistently providing each motor with the correct

voltage and current. Generous use of breakout boards would improve the complexity of the PCB design, allowing the designer to fit more components onto the board without sacrificing space. One component not implemented on the PCB was a battery life measurement circuit. This would allow the robot to broadcast its battery level to the operator, allowing him or her to have an estimate of when to have the robot climb down the tree.

Software

Several additions can be made to improve the software. These include a watchdog timer, a PID loop to control the distance from the tree, and adding more error handling to the network. The watchdog timer would be reset by the main loop whenever it received a new command from the operator. The watchdog timer will stop the robot from moving, while there is no connection to the operator's laptop. A PID controller in the gait code would help keep the top claws in optimal position for being attached to the tree as the robot extends and contracts. Error handling in the network would add fail-safes that would notify the operator and attempt to reconnect automatically in the event of a network crash.

Social Implications

The use of robots in society can bring about social implications. There is a widespread concern these robots will replace human jobs or they may cause injury to operators and observers. The Project Squirrel 2.0 robot presents potential privacy issues due to its ability to capture images and video. The most direct impact this robot has is it's interaction with its operator.

The operator has the most interaction with the robot. This means the operator should be aware of the safety hazards involved with using it. The current design does not factor in the safety of the robot or operator should the robot fall off of the tree. This makes the tree the robot is climbing dangerous to be around. This safety concern is mitigated by the fact that the robot can be controlled from a large range. In addition, unless the spikes have their safety caps attached, the spikes on the legs remain a significant hazard. The operator will be exposed to a slight amount of risk while initially attaching the robot to the tree, as this involves holding onto the claws. The operator runs the risk of getting stuck or scraped by the needles during this process. Mitigation of this risk comes from holding the robot properly as it goes through the initial attaching phase. The torque release mechanism can also injure fingers that are placed incorrectly on the claws. In the short term, this is also mitigated by holding the robot properly as it attaches to the tree. This can be addressed in a future revision of the robot by adding a guard around the moving parts of the mechanism. Finally, network security is a concern. The network broadcast by the robot is not as secure as it could be and attempts to hijack control of the robot can be made. Minimizing these network security risks improves the usability of the robot and prevents unintended operation.

While the current robot was designed for the USDA to inspect trees for the ALB, it has other possible applications. The robot can be used to identify any kind of visible tree damage. This does not completely replace the need for tree inspectors, but rather augments the job they do by providing a tool to perform inspections from the safety of the ground. This affects inspectors by requiring them

to understand the system and how to make the best use of it. In future iterations, it might also require the inspectors to understand how to repair or augment the robot in the field.

Another possible use for the robot is in providing localized networks and surveillance for military purposes. The robot broadcasts its own network over which it is controlled. Currently, the robot is not taking full advantage of the capabilities of this network. Should this network be secured, it can be used to create a localized private network on which communications could be sent without the risk of information being intercepted. At the same time, images of other areas, not specifically of the tree the robot climbs, could be sent securely from the robot. This means the robot could be used for surveillance as it removes the possibility for a human to come into direct contact with the target. However, it also creates a privacy issue if the robot is used to gain visuals of a person's home. This ethical dilemma poses a severe problem if the robot is to be sold commercially. Sales of the robot could be limited to military organizations to prevent the common criminal from accessing the product. Nevertheless the robot could still be misused by the organizations allowed to buy them. There is no clear answer to the problem, which would have to be taken into consideration when implementing the final design.

The robot has several direct impacts not only on its operator but on society as a whole. By allowing vision and networks in what are typically hard to reach locations, the robot has a potential for misuse. There is already a growing concern for privacy in America, which is not limited to digital privacy. There are not enough software precautions to prevent the current iteration of the robot from being repurposed for malicious intent. The robot is has the potential to be a good tool for helping inspectors see into the tops of trees, without needing to climb the tree themselves. The robot is not ready to be introduced into society as a standalone platform before implementing further security and design changes.

Budget

Table 4: Proposed robot budget and actual expenses

Item	Budgeted Cost	Actual Cost
Tree Model (2 Meter Sugar Maple)	$60	$0 (not purchased)
3D Printed Parts - High Quality Material	$200	$160
3D Printed Parts - Average Quality Material	$290	$0 (Donated)
COTS Components	$200	$636
Raw Materials	$100	$78
PCB Replacement, Wiring and electronics	$200	$580
Camera Module	$150	$35
Tools	$0	$55
Total	**$1200**	**$1544**

Works Cited

[1] W. T. Hlubik, "The Asian Longhorned Beetle," 2009. [Online]. Available: http://www.boylston-ma.gov/pages/boylstonma_webdocs/albslide.

[2] R. G. Nisley, "New Pheromone Traps Lure Asian Longhorned Beetles Out of Hiding," *USFSNRS Research Review,* vol. 15, pp. 1 - 5, Winter 2012.

[3] J. Boggs, A. Stone and D. Herms, "How To Detect & Manage Two Types of Beetles," 16 Dec. 2013. [Online]. Available: http://www.totallandscapecare.com/detect-beetles/.

[4] T. L. Lam and Y. Xu, "A Flexible Tree Climbing Robot: Treebot - Design and Implementation," in *IEEE Intl. Conf. on Robotics and Automation*, Shanghai, China, 2011.

[5] Kodlab, "RiSE V1 Climbing Robot," 2014. [Online]. Available: http://kodlab/seas.upenn.edu/RiSE/RiSEV1.

[6] Kodlab, "RiSE Robotic Hexapod Version 2.0," 2014. [Online]. Available: http://kodlab.seas.upenn.edu/RiSE/RiSEV2.

[7] Kodlab, "RiSE Version 3 Prototype," 2014. [Online]. Available: http://kodlab/seas.upenn.edu/RiSE/RiSEV3.

[8] A. Saunders, D. I. Goldman, R. J. Full and M. Buehler, "The RiSE Climbing Robot: Body and Leg Design," in *Proc. SPIE Unmanned Systems Technology VIII*, Orlando, FL, 2006.

[9] G. C. Haynes, A. Khripin, G. Lynch, J. Amory, A. Saunders, A. A. Rizzi and D. E. Koditschek, "Rapid Pole Climbing with a Quadrupedal Robot," in *IEEE Intl. Conf. on Robotics and Automation*, Kobe, Japan, 2009.

[10] Biorobotics Lab Carnegie Mellon University, "Modular Snake Robots," 2013. [Online]. Available: http://biorobotics.ri.cmu.edu/projects/modsnake/.

[11] Biorobotics Lab Carnegie Mellon University, "Snake Gaits," 2010. [Online]. Available: http://biorobotics.ri.cmu.edu/projects/modsnake/gaits.html.

[12] E. Ackerman, "No Tree Is Safe From This Chainsaw-Wielding Robot," 25 Feb 2014. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/industrial-robots/no-tree-is-safe-from-this-chainsaw-wielding-robot.

[13] E. Demaio, D. C. Ilacqua, M. J. Simpson and L.-J. V. Mistretta, "Project Squirrel," Worcester Polytechnic Institute, Worcester, MA, 2013.

Appendix A: Tuning the Torque Release Mechanism

Step 1: loosen the set screws until they are barely threaded into the hole.

Step 2: Run the motor in reverse (the correct direction) and see how far the legs come up

Step 3: Alternate tightening the 2 set screws a little bit

Step 4: Repeat steps 2 and 3 until the legs peak at the desired position