UNIVERSITY OF CALIFORNIA, SAN DIEGO

Mechanism Design And Control Implementation of a Hopping Robot

A Thesis submitted in partial satisfaction of the requirements for the degree

Master of Science

in

Mechanical Engineering

by

Robert Paul Hughes

Committee in charge:

Professor Tomas Bewley, Chair
Professor Frank Talke
Professor Daniel Tartakovsky

2008

The Thesis of Robert Paul Hughes is approved, and it is acceptable in quality and form for publication on microfilm.

_____

_____

_____
Chair

University of California, San Diego

2008

DEDICATION

In recognition of inspiration, moral and financial support I dedicate this thesis to my parents and grandparents. Without their assistance I would have never been given the opportunity and spark to learn and achieve at this high level.

Thank you

David & Susan Hughes

Dorothy & Clyde Czernek

Nancy & Wilmont Hughes

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Tomas Bewley for his support as the chair of my committee. Through constant prodding and keen insight his guidance has proved to be invaluable. Additionally his book, Numerical Renaissance (which has yet to be published) proved to be a very valuable tool.

I would also like to acknowledge Chris Schmidt-Wetekam, his assistance and prior work made this thesis possible. Appendix A is a reprint of his equations of motion published in the CDC 2007.

ABSTRACT OF THE THESIS

Mechanism Design And Control Implementation of a Hopping Robot

by

Robert Paul Hughes

Master of Science in Mechanical Engineering

University of California, San Diego, 2008

Professor Tomas Bewley, Chair

This thesis is the conjunction of several mechanical designs as well as how they can be joined with control algorithms to give a sense of intelligence to their movements. By using their intelligence these mechanisms can perform more complex tasks then if they were mechanisms alone.

The first design is a double pendulum swing-up test stand that is available to perform control experiments. This test stand was utilized to solve the problem of inverting two pendulums of different lengths and balancing them. I have discussed in this paper how to invert a single pendulum in detail.

The second design is similar in theory to the inverting of a pendulum but

In practice different variables need to be considered. This I-Hop Robot it was already designed but I have made modifications to it so it may hop approximately 300% higher. Another modification that has the option to be added to I-hop is I-fling which as an apparatus that allows the I-Hop robot to have the capability to throw a light weight ball.

These designs along with the theory of their operation both mechanical and control operation compose the entirety of this thesis.

INTRODUCTION

The use of software to control hardware is a very powerful tool. This strategy is a way for a robot or mechanism to become more than a lifeless mess of carefully strewn together parts, it brings knowledge about the surroundings, and the ability to react to them intelligently. The science of Control, as it is described in this thesis, is relatively new in the whole of science itself but brings a new way to inspire the lifeless that has changed the world making robots and mechanisms more autonomous allowing faster and more precise manipulation then in years prior to its conception.

In this paper I will walk the reader through some of the design processes that created mechanisms for the University of San Diego California. These mechanisms have many applications in the non academic world and can be used in the realms of the military to the toy market. The main focus of this thesis is to walk the reader on how the concept of modification to the I-Hop robot was made in order for it to hop higher thus expanding its rage of freedom allowing for more applications. Other designs that are related in theory or application to the motion if the I-hop robot are also discussed and used as a stepping stone to how this robot can be controlled.

In closing this thesis talks about Computer Aided design and how it can be used very succinctly in the design process. I also touch upon the benefits of the software package I have used and how it may be seen as superior in this application.

1 Single Pendulum Design

1.1 Introduction

During my undergraduate career at UCSD I developed a design for a dual inverted pendulum swing up test stand. I designed, built, and tested this unit, however unfortunately at that time the software required to implement complex control algorithms that meshed with the hardware was not available with the meager budget of one hundred dollars. Since our lab became licensed in the software required, I was able to come back and rebuild the test stand to be more viable for control experiments[1]. The basic design of the test stand is to be able to invert two long rods connected to a free pivot and balance them.

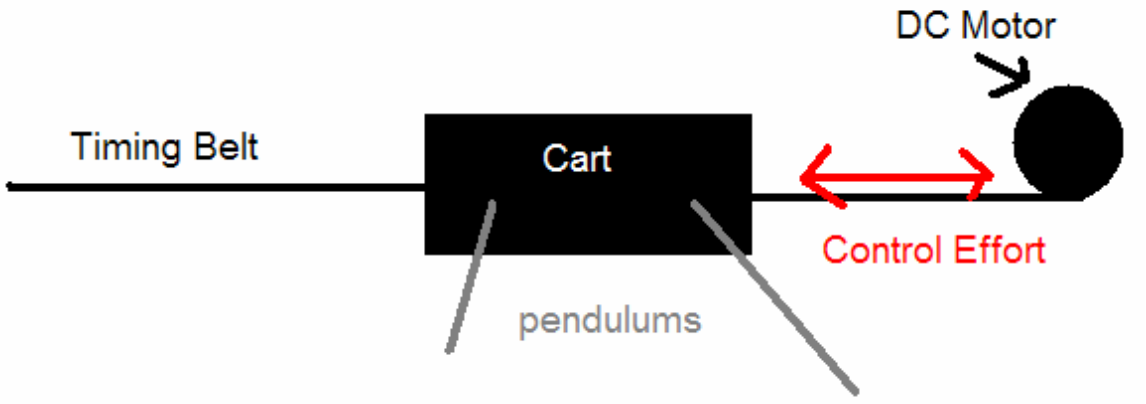

Figure 1: Diagram from original dual inverted pendulum swing up test stand.

[1] David Seto later utilized this test stand in this thesis

1.2 Undergraduate Design

The undergraduate design seen in Figure 2, Figure 3, and Figure 4 utilized a wireless mouse's encoders to relay back angular position of the two pendulums. Additionally, a potentiometer was used to derive a position measurement of the cart.

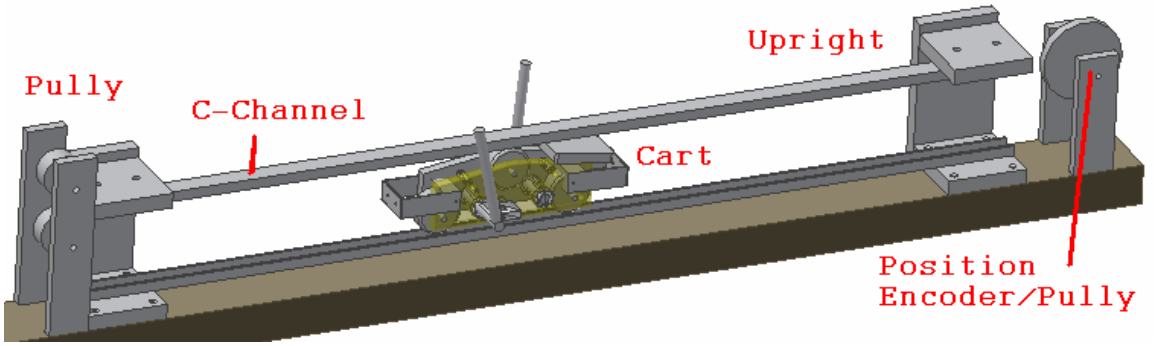

Figure 2: Overview of CAD model for undergraduate dual inverted pendulum test stand

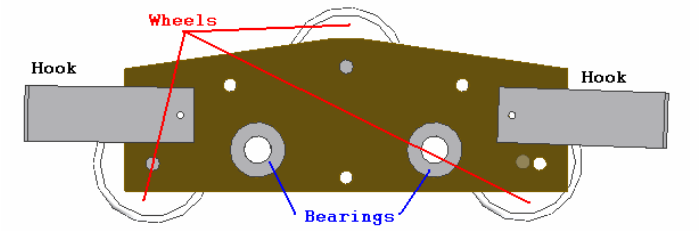

Figure 3: Close up of Cart design for under graduate dual inverted pendulum test stand.

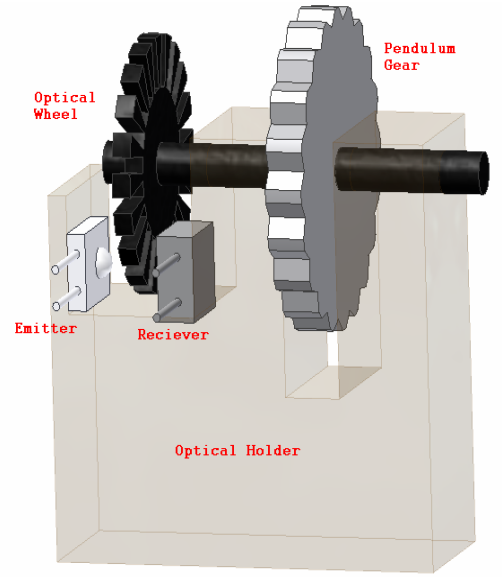

Figure 4: Close up of wireless mouse encoder used for transmitting pendulum angle to the computer for undergraduate dual inverted pendulum test stand.

The use of the wireless mouse proved to be able to transmit a signal to the computer which was converted into angular measurement with interrupt commands in Windows®. Much of the information, however, was lost in transmission and the angle proved too inaccurate a measurement as too many counts of the optical wheel were dropped via the Bluetooth™ mouse transmission. Additionally the potentiometer used was not precise enough in the linear measurements of the cart.

A myriad of other problems plagued this design, all of which yielded a great deal of valuable information when it came to design the second prototype.

1.3 Dual Inverted Pendulum Swing up Test Stand Design Iteration 2

Subsequently, our lab now had a larger budget and could afford more commercial off the shelf parts (COTS). Two ideas were prompted: a rack & pinion design and a timing belt design. After much deliberation it was decided to continue

with a timing belt system of cart driving. I located a company (Rollon Linear Evolution) that virtually refined my cart design into an all in one package seen in **Error! Reference source not found.**. The "Slider" seen in **Error! Reference source not found.** was the "Cart" in the previous design and moved about the six foot long extruded load bearing.

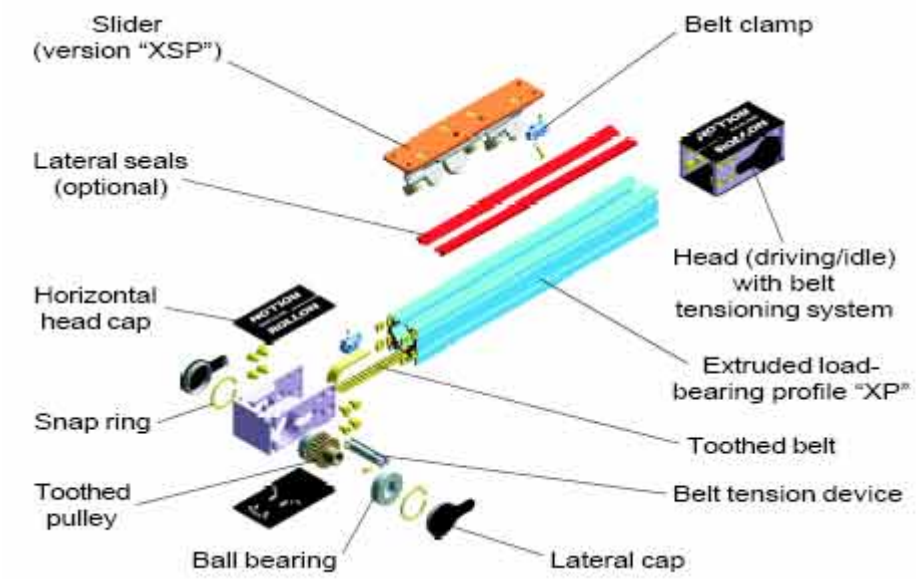

Figure 5: Rollon Linear Evolution Ecoline track

Through the integration of the "Toothed pulley" to a motor (Pittman) with internal optical encoding, the test stand would have a very accurate way of measuring the linear position of the Slider. This design had to mount the hole of the pulley inline with the motor shaft, failure to so would introduce a nonlinearity in the transfer function from voltage applied to movement of the slider caused by a "sticking" sector of rotation. Mounting was fine-tuned with many screws that precisely positioned the motor.

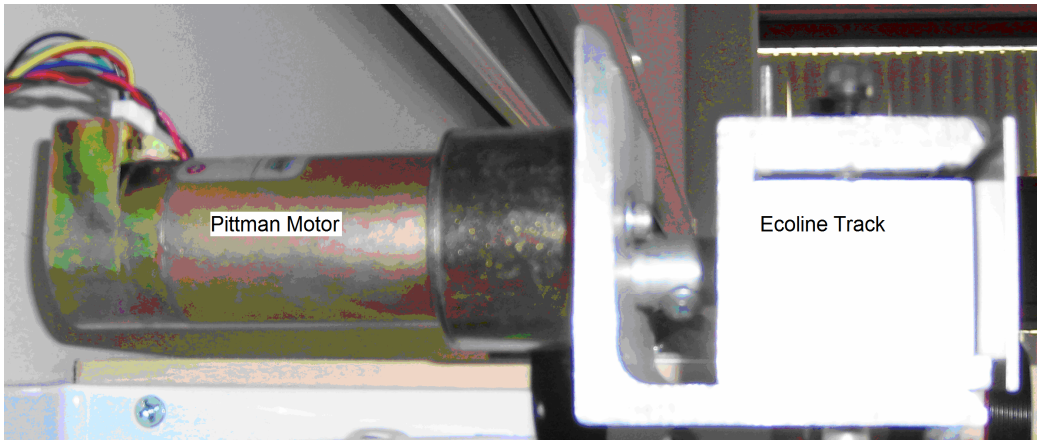

Figure 6: Linking Pittman motor with optical encoding with Ecoline track's "Toothed pulley"

This new design also incorporated a high flex ribbon cable (similar to cables found in printer heads) to relay angular information from the "slider" to the DAQ board, instead of Bluetooth™. The new cart design (which is mounted to the Ecoline slider) made use of optical encoders to extract angular measurements from the pendulums (See Figure 7.) These encoders were selected to have a sufficient number of counts to provide enough angular position accuracy for the dual inverted pendulums operations.

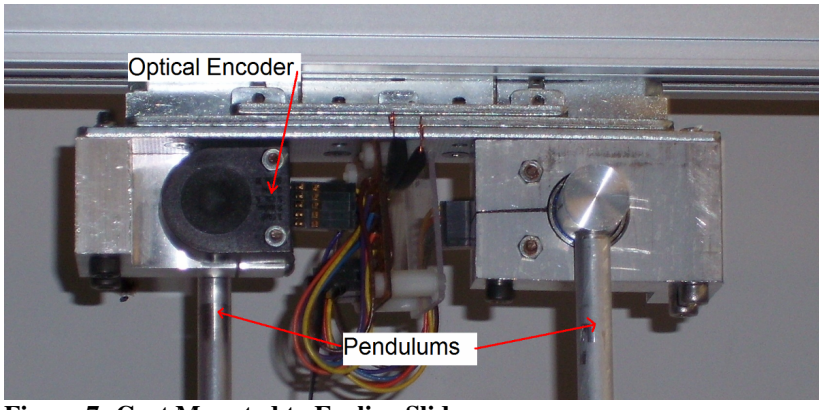

Figure 7: Cart Mounted to Ecoline Slider

2 I- HOP DESIGN

2.1 Quick release design process introduction

Problem: In order to put a sufficient amount of spring energy into the system (enough for a 1 meter hop) there are several things to consider. First it is necessary to "suck" the pogo assembly back up into the robot quickly and in mid-air as to extend the hop by another 4-5 inches. To do that we must release all the spring tension in mid-air and quickly. There are many ways to release something quickly, including camming, clamping, pinning, or a myriad of other methods.

Additionally, amount of spring tension on the robot must be controlled so we can fix the rope/spring assembly at any point. This is required so that any height of hop can be achieved within the system boundaries. Furthermore, as it was our intention to dissipate the energy on a down hop, it was important to actuate the pogo assembly as to completely transfer the system energy to the ground, which also requires somewhat precise tensioning of the rope/spring assembly.

Finally, we could not put a high amount of force on the motor shaft for extended periods of time as it would warp.

We needed to perform all of these functions and not add a great deal of weight or take up much space on the robot. It was also important to retain a robot that has a continuous hopping mode.

2.1.1 Design Iteration 1:

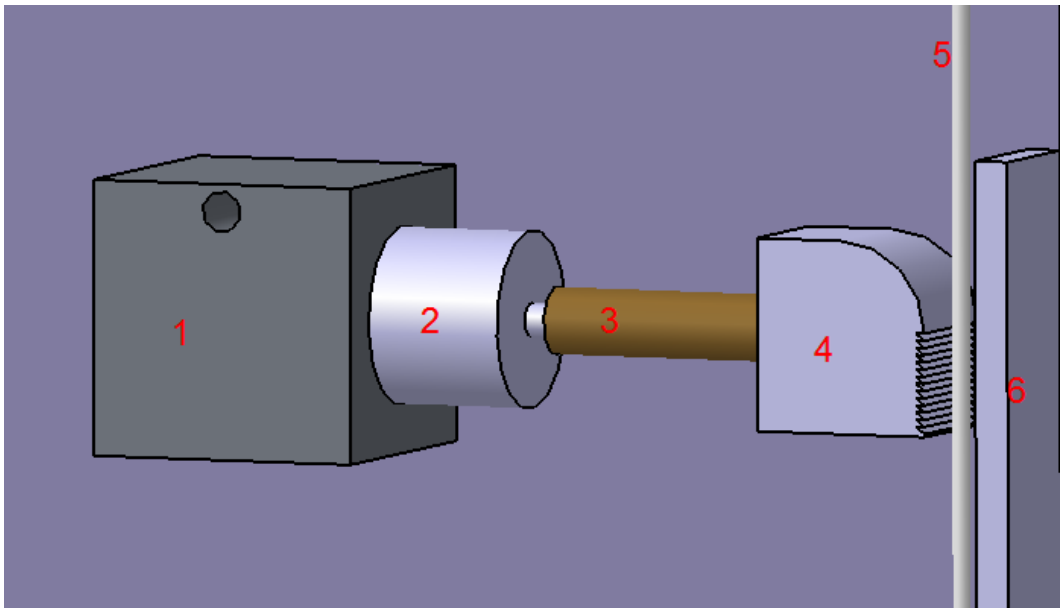

Figure 8: Design Iteration 1 of Cam device

2.1.1.1 Concept

This iteration was intended solely to solve the problem of the quick release. Initially rope 5 would be pulled down, which would disengage the camming action as the motor mount 1 would pivot about its visible hole in Figure 8. When camming was required, motor 2 would turn worm gear 3 and push cam 4 to cam and clamp the rope 5 between the cam and wall 6. Once engaged this would be a very solid fixing of rope 5. Additionally when it was required to quickly release rope 5, motor 2 needs only to reverse direction and the rope would be quickly able to slide out of its fixed position.

2.1.1.2 Test

A test fixture was built to see if this device was a viable way to fix and quick release a rope under tension. Moreover, we wanted to experiment with different types of rope to find which was the easiest to fix. Finally we wanted to see what spring would give us enough force in the constrained deflection on the robot body.

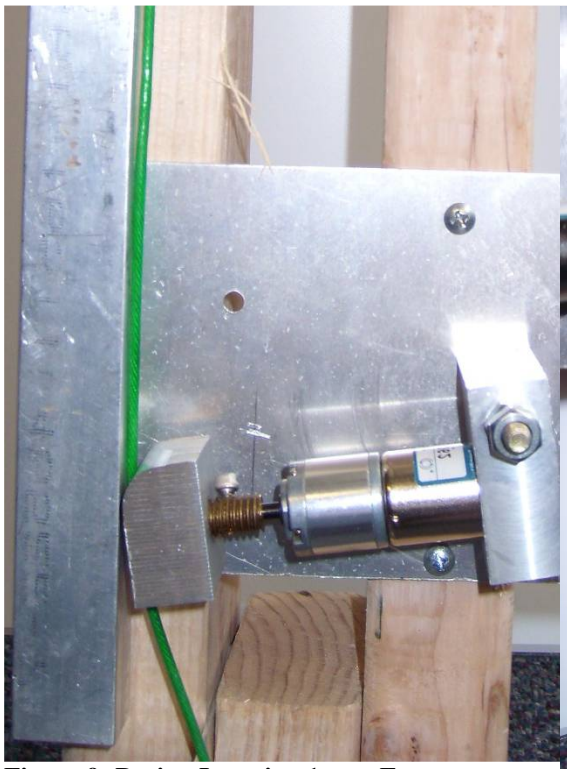

Figure 9: Design Iteration 1 test. Front view.

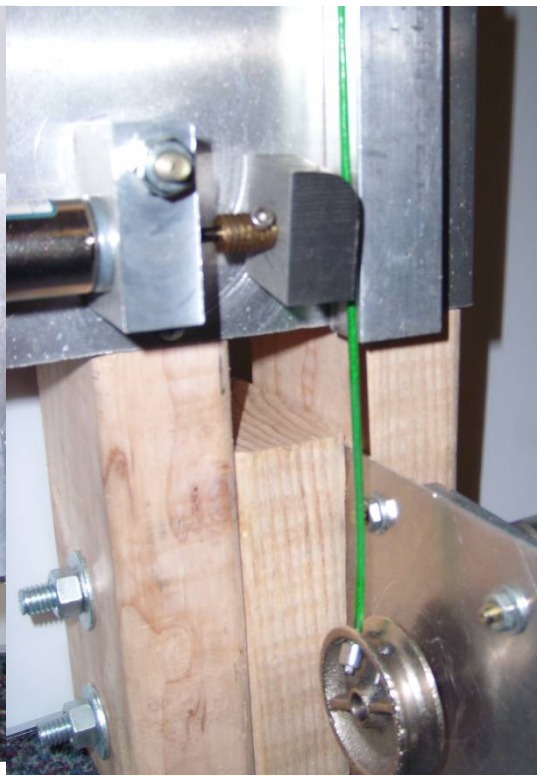

Figure 10: Deign Iteration 1 test. Isometric View

2.1.1.3 Results

Rope:

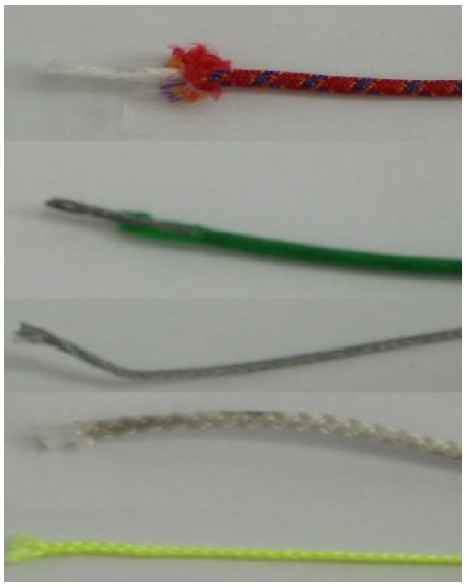

Figure 11: Rope Tests: (from top to bottom) Spectra Core rope, Poly Vinyl coated steel cable, Steel cable, Braided nylon rope with core, Nylon rope

Steel cable: No matter how much force we could apply with the motor used we were not able to fix it at all. This was due to the fact that steel braided cable does not deform unless under extreme force and thus does not have enough surface area to provide sufficient friction to keep it fixed under moderate loads (20lbs+).

Poly Vinyl coated steel cable: Initially we thought that poly vinyl coated steel cable was the best type of rope, but the coating degraded very quickly and would stretch. It turned out that the poly vinyl coating was not well fixed to the steel braided cable.

Braided nylon rope with core: This rope is intended for tie-down purposes. Unfortunately, the core was made out of paper (heavy weight) which was not sufficient for the target loads.

Nylon rope: The rope was also used for tie-down purposes, but was a small diameter. This worked the best at holding weight of 20-30 lbs but slipping was still a problem as there was not enough surface area for sufficient friction.

Spectra Core rope: This rope was bought from REI and was rated at 400lbs. Its use was for recreational rock climbing. This rope has a texture on the outside that was a significant increase to its coefficient of friction. Furthermore, its diameter was large enough to provide enough surface area when compressed for sufficient friction under moderate-high loads (0-200lbs). This was the best rope by far and exceeded our requirements and had a factor of safety of approximately 5.5. The only difficulty with the rope was the knots were fairly large, 3-4 times the diameter of the rope. This problem would be alleviated later by use of wrapping a rope loop with copper wire and burning the free end to create a hard large end that was 1.5 times the diameter of the rope.

Figure 12: Spectra Core Rope Bound with copper wire

Cam: We found that the Cam did its job fixing the rope sufficiently well, provided a proper rope, but there was room for improvement. The motor shaft would be taking a lot of the loads, and for such a small shaft we could see problems arising in the future from potential bending or breaking shafts. Also the worm gear mounted directly on the shaft was an insufficient design as it was fixed to the shaft with a set screw, which would not provide good performance over time with strong axial loads. The aluminum cam itself did not seem to engage on the rope very easily and often had to be pushed manually to begin the camming action.

Spring: Initially Surgical tubing was bought from McMaster-Carr, and found to have the highest energy density[2]. I was able to find higher performance surgical tubing-like rubber bands used for spear guns. These rubber bands came in many sizes and were available locally at San Diego's many dive shops.

[2] See Appendix E

2.1.2 Design Iteration 2

2.1.2.1 Areas of improvement:

For this next iteration I worked to improve several problems:

1) Eliminate the motor taking any axial loads.
2) Engage the cam without manual effort.
3) Build up sufficient slack to have the “quick release action” release all the tension on the rope/spring assembly.
4) To be able to do all of the above with only 1 motor to avoid excess weight.
5) Perform both continuous hopping and single high hops with only one motor for the driving effort.

2.1.2.2 Concept:

Through utilization of a pawl/ratchet assembly I was able to devise a method to accomplish a quick release mechanism for design iteration 2. When the motor would turn the red pulley clockwise the green cam would be pulled down by the pawl which would initially pull down a rope/spring assembly. Next, the motor would turn the red pulley the other direction, engaging the cam onto the rope, locking it in place and creating slack below the cam. Lastly, the pulley would then be turned back in the clockwise direction which would disengage the cam from the rope allowing the slack release tension on the rope/pulley assembly. (See Figure 13, Figure 14, Figure 15)

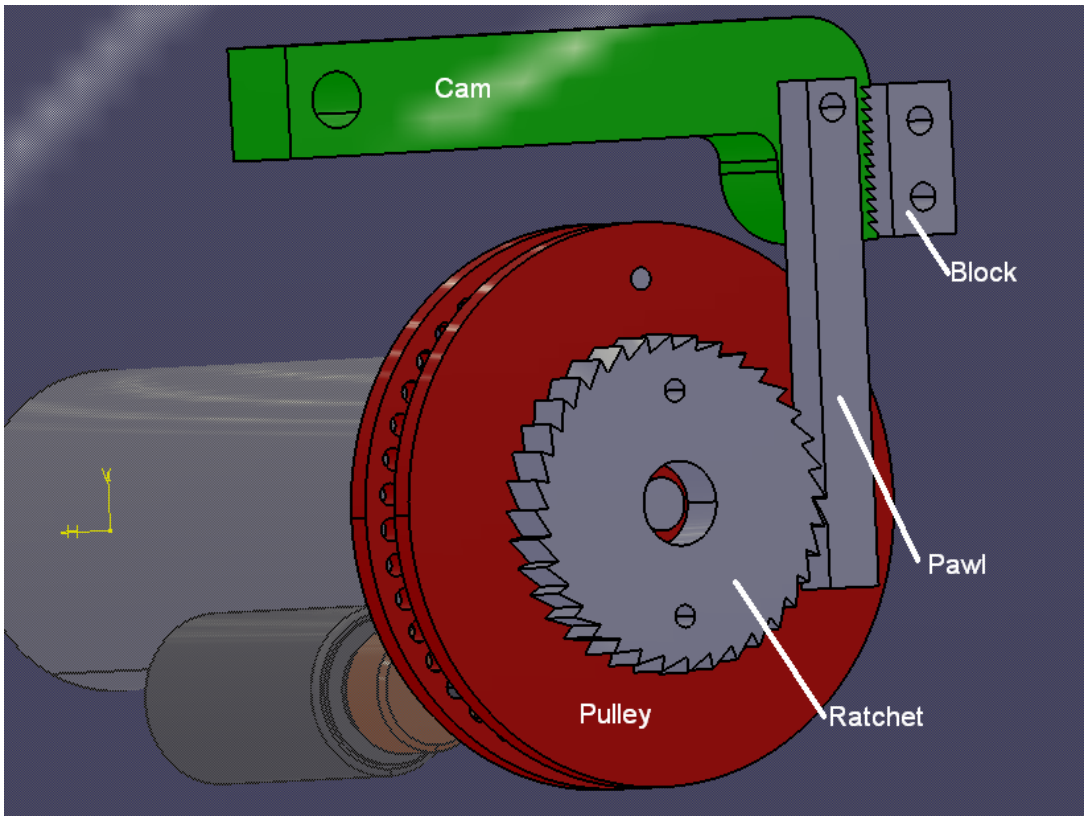

Figure 13: Iteration 2 for quick release mechanism: Masterclam 3001 (front isometric view)

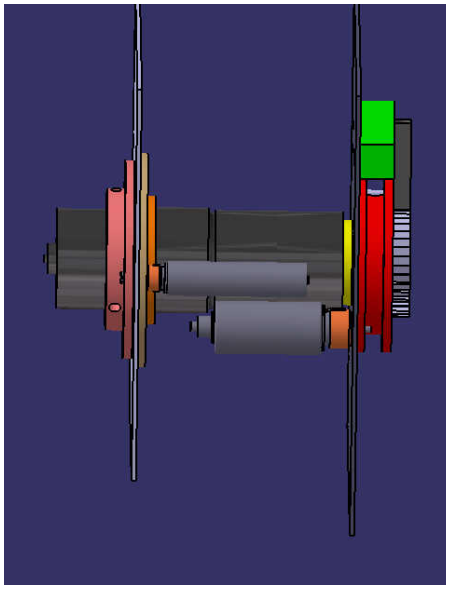

Figure 14: Iteration 2 (Side view)

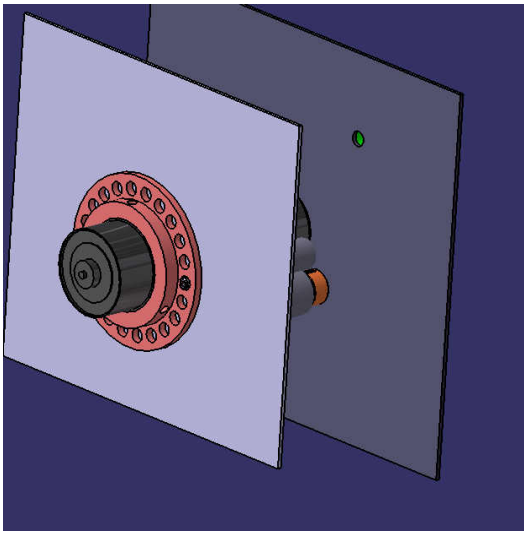

Figure 15: Iteration 2 (back isometric view)

Additionally, this design employed the use to two solenoids to act as a sort of clutch between two different modes of the motor. In one mode the motor shaft would rotate about the motor housing, but in the other mode the motor housing would rotate about the shaft. In Figure 14 energizing the "push type" solenoid would lock the red pulley to the body of the robot causing the motor's shaft to be fixed in relation to the robot. Figure 14 and Figure 15 show that not passing current through the "pull type" solenoid would lock the salmon pulley to the body of the robot. This assembly would then become fixed to the housing of the motor, causing the motor's shaft to rotate (in the conventional sense) in relation to the motor's housing.

Note : All of the solenoids have been altered by installation of a small spring that returns the solenoid pin to a state of rest that is different then its energized state. This is required because the solenoids are typically in a horizontal plane and would simply stay where they were if the current was turned off.

In this manner we now had a way to rotate the shaft or housing of the motor with respect to the robot body. When rotating the housing the salmon pulley could drive the hopping motion of the robot, while the rotation of the shaft with respect to the robot body could drive the camming technique for putting a great amount of tension on the rope/spring assembly.

2.1.2.3 Problems:

1) Upon receiving of the motor (www.lynxmotion.com) which was rated at 90lb-in of torque at stall current (10amps) we found that the motor was very much over-speced. By setting up a brief test in which a long rod was mounted perpendicular to the motor shaft and a known load placed on that rod till the motor stalled at (20amps) we determined that the max torque that the motor could output was about 20lb-in. This motor was not sufficient to do the job of tensioning the spring/rope assembly to the prescribed 50lbs to make the robot bounce approximately 30inches.
2) When producing slack under a tensioned spring/rope assembly the slack would tend to get out of the plane of the pulley. Upon relieving the tension of the spring/rope assembly (the quick release) the rope would tend to disengage from the cam device, requiring a manual engagement to get back on track.
3) The length of the cam from pivot to teeth was too long to disengage properly from the rope.
4) The holes on the salmon pulley of Figure 15 were too large; it would take extra time for the solenoid pin to engage through the holes. (This was more of a worry than a problem, as we could not test the assembly under operating conditions due to the over-speced motor.)
5) The 3 set screw method of mounting the salmon pulley to the motor in Figure 15 would slip under high loads.

2.1.3 Design Iteration 3

2.1.3.1 Areas of improvement

1) Find a motor from a company that can provide accurate manufacturer specs that meets our required specifications.
2) Create a funnel that would keep the slack on track.
3) Alter design of the cam to provide ample rotation with small linear movement of the pawl.
4) Use a different solenoid configuration with all smaller solenoid shafts.
5) Utilize different camming method on salmon pulley. (Figure 15)

2.1.3.2 Concept

The major change to this design was the use of a single double acting solenoid as opposed to two independent solenoids. This solenoid would fix the shaft of the motor to the body of the robot (See Figure 16) by engaging the solenoid pin to the tan pulley when energized. The solenoid would engage its newly machined solenoid pin extension (See Figure 17) to the orange pulley thus locking the body of the motor to the body of the robot.

A groove was instituted is the purple camming surface (Figure 16) to help keep the rope in the camming device. Additionally that camming surface was extended so that the lower face of the new camming surface and the groove of the tan pulley would act as a funnel to help keep the rope in the camming device.

Other minor modifications were made by shorting the cam pivot length and changing the clamping style of the orange pulley to a split ring clamp instead of a set

screw. Additional work was needed to swap out the motor as the shaft and its body were different dimensions and most of the parts surrounding this assembly were designed around the old motor.

We were now finally able to test the device under high loads and observed some interesting behavior of the dual 4 bark linkage under these higher loads.

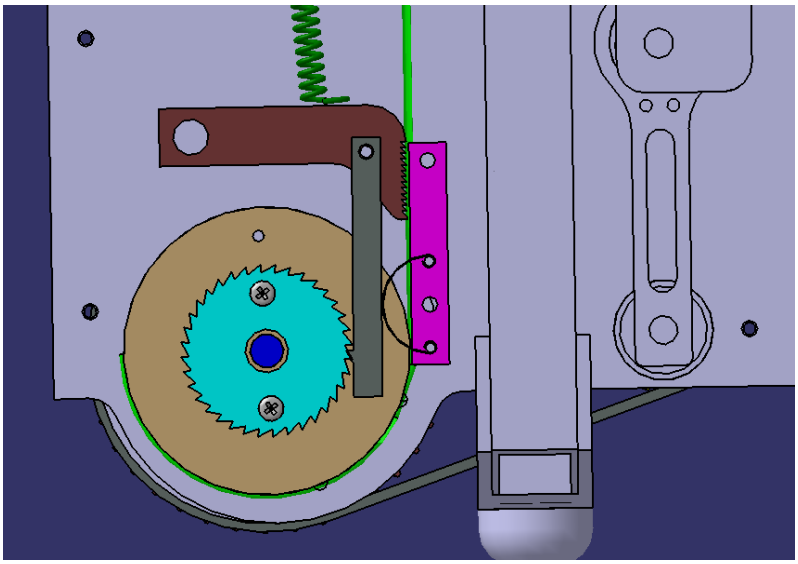

Figure 16: Design Iteration 3 (Front View)

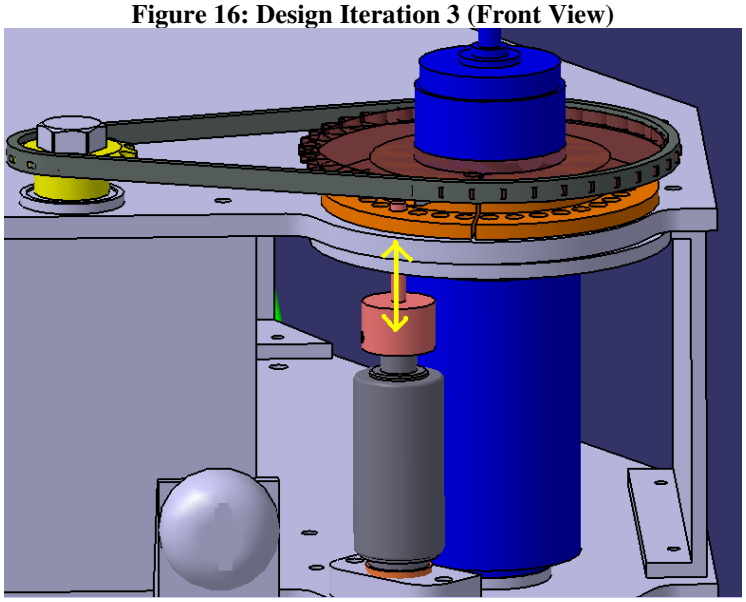

Figure 17: Design Iteration 3 (Bottom Isometric view)

2.1.3.3 Problems

1) We encountered a different mode of the dual 4 bar linkage assembly than was intended. Both links that were mounted touching the robot body (links 1 and 3) were supposed to be at the same angle; however they often encountered a mode in which they were reflections of each other about the vertical axis (See Figure 18).

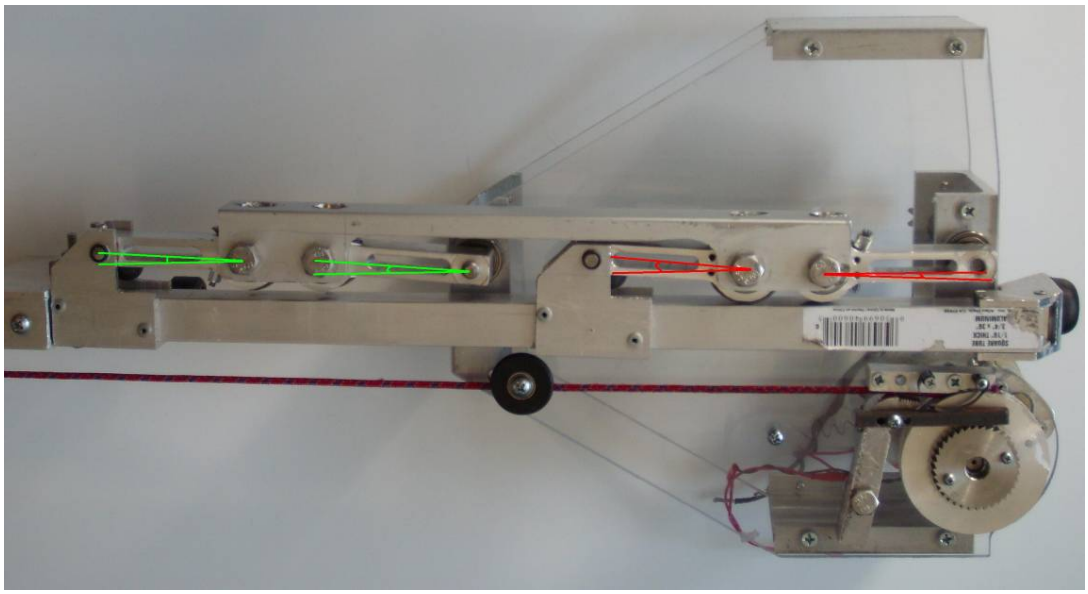

Figure 18: Picture of Robot in "Lock Mode"

In this mode the pogo assembly was locked and could not perform its function of utilization of the spring/rope assembly's tension to actuate a hopping maneuver. We dubbed this mode "lock mode." The only was to disengage this mode was to relieve all tension of the rope/spring and manually push the linkages to where they were intended to be.

2) The solenoid pin was made out of more malleable steel and bent under the high loads when engaged on the tan pulley (Figure 16).

2.1.4 Final Iteration

2.1.4.1 Areas of improvement

1) Lock mode needed to be fixed.
2) Solenoid pins needed to be strengthened or completely redesigned.

2.1.4.2 Intermediate Concepts

In the time between finalization of the design and iteration 3 there were several methods tried to fix the problem of lock mode. The primary issue was that linkages 1 and 3 needed to always have the same rotation angle.

The first method attempted was a separate 4 bar linkage on the opposite side of the robot as the original dual 4 bar linkage. This new linkage was intended to have a small footprint as it and only needed to be stiff in 1 plane of motion. This new linkage was out of phase of the original linkage by 90 degrees and intended to provide torque to linkages 1 and 3 in a low torque point of the original linkage (where the transition would occur from normal rotation to lock mode). These new links were loosely riveted together providing a very small bearing with moderate friction.

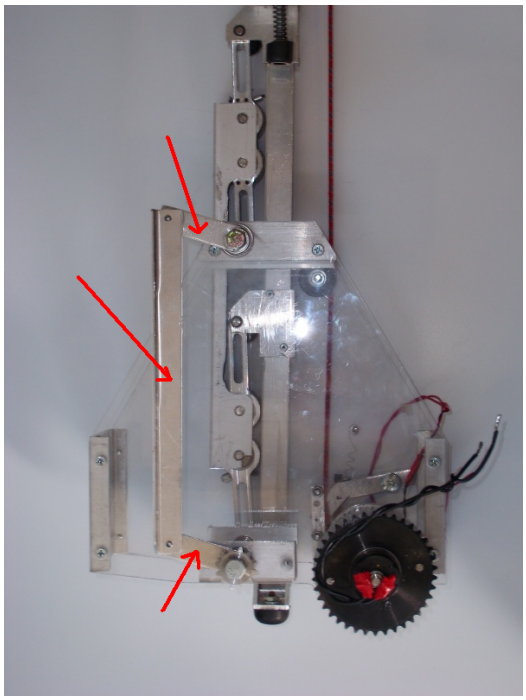

Figure 19: An additional 4-bar-linkage was added in an attempt to counteract lock mode. Red arrows indicate the new members added

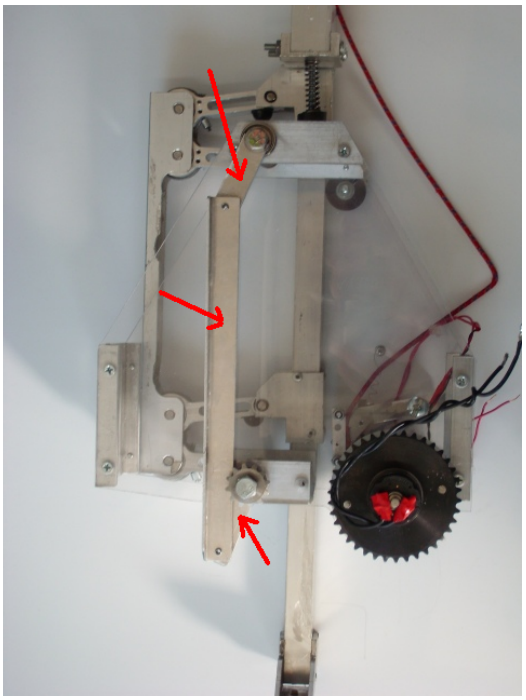

Figure 20: An additional view showing the attempted fix for lock mode once the pogo rope assembly would have been actuated.

This method worked for a while but would slip over time, and eventually we would encounter a lock mode.

The next method attempted was mounting a bearing on the opposite side of the pogo assembly as the linkages which would keep the pogo assembly vertical. By keeping the pogo assembly vertical lock mode would not be encountered as it placed the pogo assembly at a tilt of approximately 10-15 degrees of its intended angle.

This single bearing kept the upper portion of the pogo assembly along its intended travel line but added massive amounts of friction that considerably slowed down the pogo assembly. The main issue was that the parts machines were not perfect and the pogo assembly actually traveled at a slant of 1-3 degrees with no forces on it.

Additionally the bottom of the pogo assembly needed another bearing to keep the entire assembly straight (actually at the 1-3 deg slant). These bearings would essentially over constrained the system inducing massive friction if not very precisely placed. Optimal precision placement of the bearings could not be achieved in a practical manner and thus this design was scrapped.

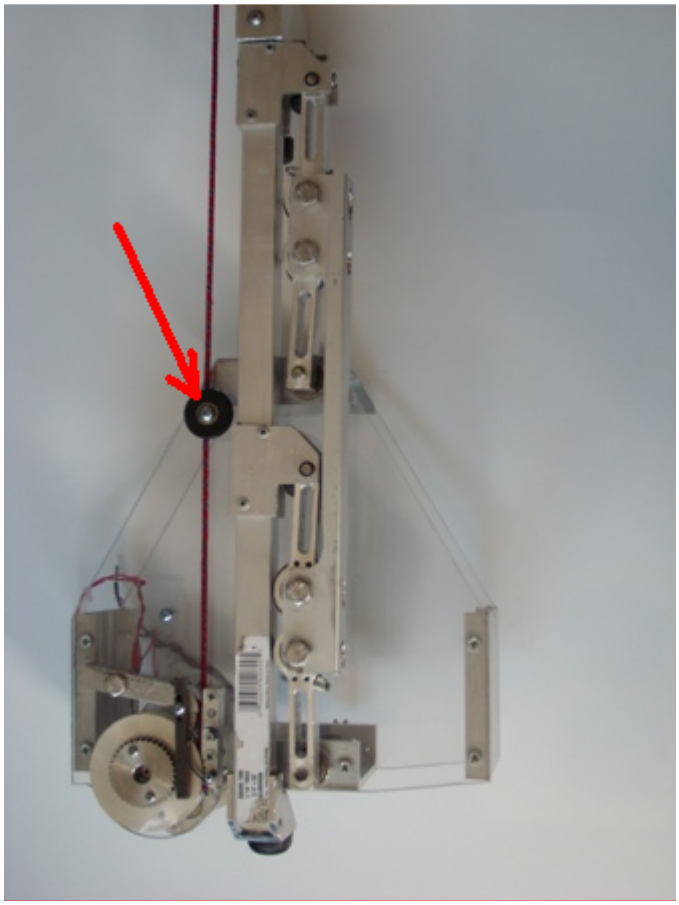

Figure 21: Implementation of one bearing to try to keep the pogo assembly straight

The next method involved gearing the heads of the linkages. This method was an attempt to reduce slop (one reason for locking mode) in the old linkages by creating precision gears. I attempted to make these parts out of sandwiched polycarbonate

pieces cut out on the lasercam. These pieces eventually meshed well (took a few attempts scaling be a factor of 1% to account for laser thickness) but the gear teeth bent too much under the high loads on the parts causing an excess of slop. Aluminum parts could have been made but it was more cost effective to proceed to the method below.

Figure 22: Toothed Linkages

Finally a chain drive (#25) was installed between linkages 1 and 3 which insured their proper rotation with very little slop. Essentially both link 1 and 3 would now be driven and stay in sync.

2.2 Up Hop procedure

2.2.1 Introduction

This is the functional description of the procedure used to hop up on top of, or over an object. This procedure requires knowledge about the object (to be navigated) such as its location and orientation as well as its dimensions. For the purpose of this section I will assume that the operator of the robot is provided this knowledge. Please

note that all springs / ropes items in red are under tension while items in green are not under any significant tension.

2.2.2 Procedure

1) Robot will be in horizontal rover mode when it encounters a stair. Recognition of a stair will be made by the human robot controller. (Future work with image recognition software in conjunction with a small onboard camera for stair recognition can be added later).

2) Upon recognition of the stair the robot will then change its configuration to be in upright rover mode. This sequence is can be preformed in one of two ways (model predictive control trajectory or "bang bang" trajectory). The robot is now ready to begin tensioning the spring in order to build up energy for a hop.

3) Now that the robot is ready to begin to store energy it will enter phase 1 (preparation phase) of the up-hop procedure. Phase one first consists of engaging the clutch pin (shown in red; by turning the pink clutch pulley in the correct direction) on Pulley B (the pulley which is connected to the motor housing). This is locks the motor housing to the body of the vehicle so that the shaft is free to turn. A locking-movement is required to perform this clutch pin engagement which turns the motor in alternating directions for a very short amount of time (<1sec).

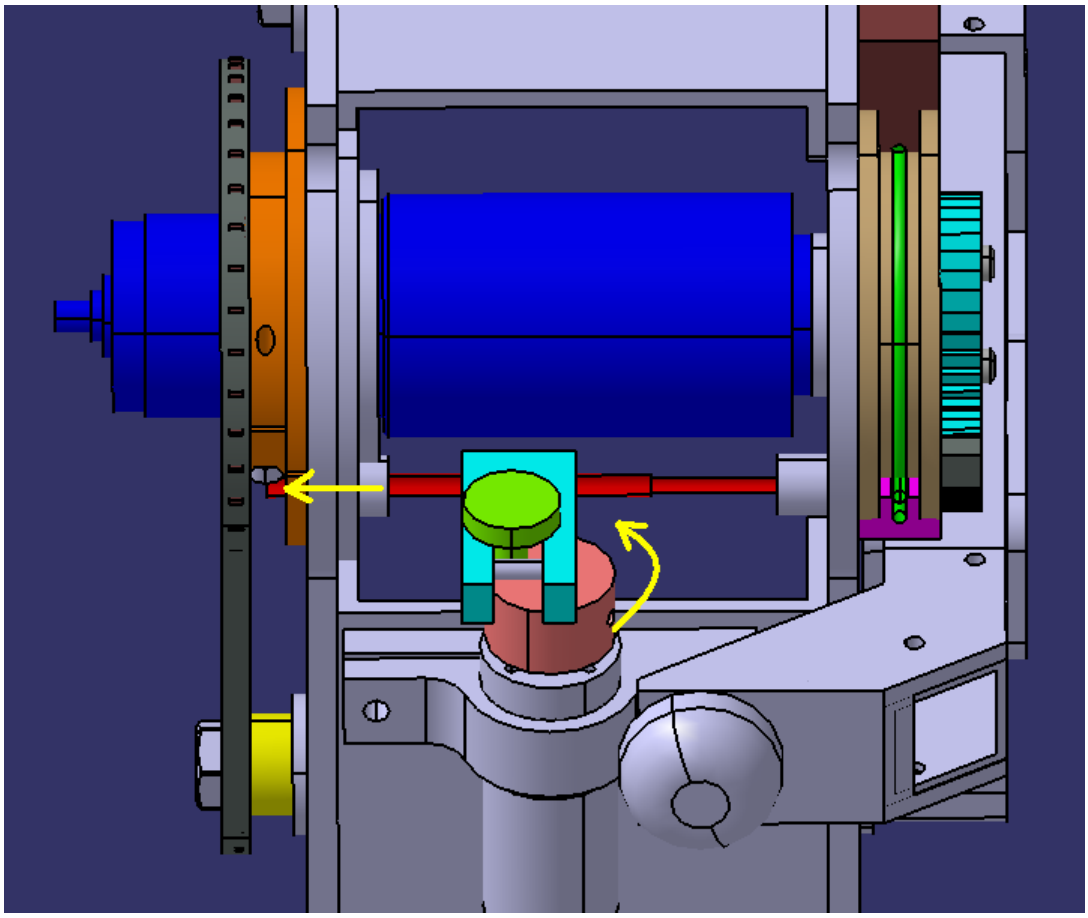

Figure 23: Salmon clutch pulley forces the green pin to push the aqua fork that is attached to the red clutch pin to lock the orange pulley in place. This action is made possible by the small motor driving the pink clutch pulley until the green pin hits the grey pin causing the motor to stall. This locks the housing of the motor to the housing of the robot.

4) Once the motor housing is firmly locked to the body, Pulley A (which is mounted on the motor shaft via a set screw configuration) is then turned clockwise (if looking at the visible face of the pulley). This action begins 3 events.

4a) First the rubber bands begin to tension, this will be our primary energy for the up-hop. A rope attached to a press fitted steel pin mounted 1.2 inches radially in the groove of the pulley pulls the rope into the pulley groove and

begins to wind up that rope to a specified tension (which will be a function of time which has been calibrated).

4b) This clockwise movement engages a pawl (forced to the ratchet by a bow spring) which on to a ratchet firmly mounted to the pulley. This pawl is then moved down (earthward in the up-right rover configuration) until which time as the ratchet teeth no longer are engaging on the pawl tooth.

4c) When the pawl is pulled earthward a cam is then turned clockwise which disengages it from clamping the tensioning rope.

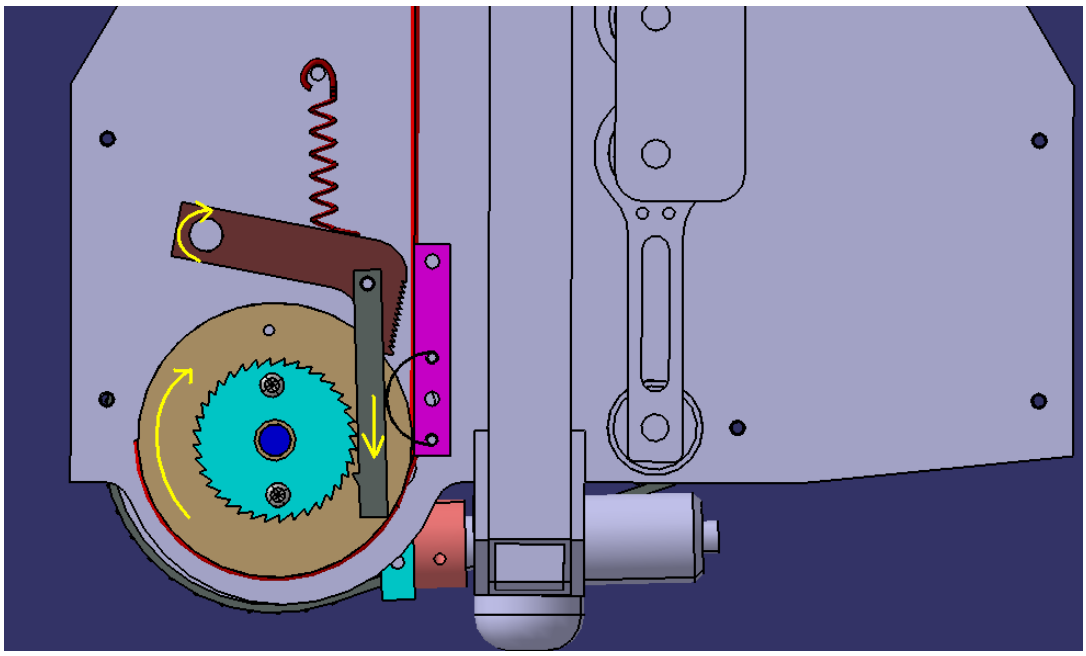

Figure 24: Tan Pulley is rotated clockwise causing the dark grey pawl to move down, which in turn rotates the brown cam clockwise causing the rope to disengage from the cam.

5) Since there is much tension now on the rope-band assembly we need to be able to quickly release that tension without damaging any component in order for the

pogo leg to be quickly returned to a rover mode configuration. This ability to quick release that tension is pivotal so that once on-top of a step the robot may be able to go into upright rover mode quickly. In order to have the capacity to quickly release any spring tension the shaft (connected to Pulley A) then turns counter-clockwise which triggers 3 events.

5a) The first event triggered by a clockwise turning pulley is the cam, that is being held to touch the rope via a tension spring attached to it (pulling the cam counterclockwise at all times), is engaged onto the rope as the rope tries to move up due to the spring energy. The harder the rope tries to move up the harder the cam clamps to keep the rope in place.

5b) When the cam begins moving counterclockwise the pawl is then pulled skywards, which would in turn move the ratchet which is connected to Pulley A counterclockwise. However Pulley A is already turning that direction so there is no initial useful movement of the pawl. As Pulley A continues to turn the ratchet pushes the pawl away from it, depressing the leave spring and leaves the pawl simply rotating about its attachment point on the cam.

5c) By turning Pulley A counterclockwise after the initial tensioning and clamping the rope at the cam location slack is built up in the rope. The Pulley

A is turned until the location of the press fitted steel shaft (which fixes the rope to Pulley A) is at the desired location to potentially release all spring tension.

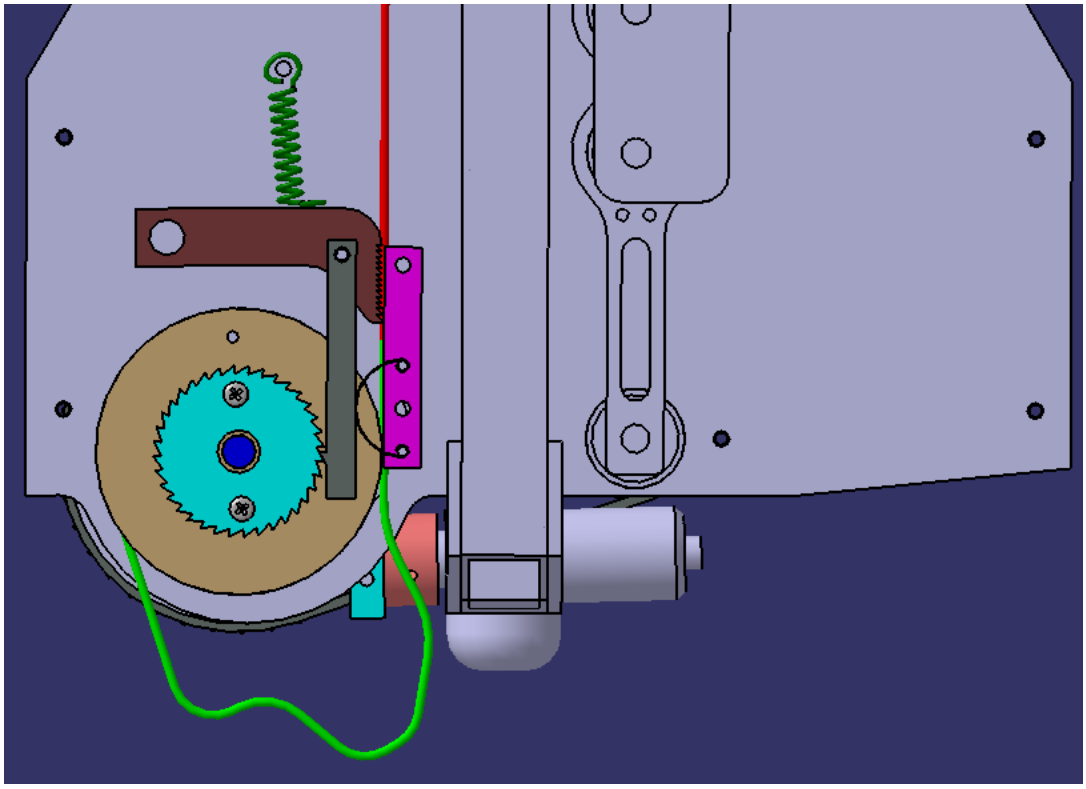

Figure 25: The tan pulley is rotated counterclockwise, which allows the dark grey pawl to move up. This action causes the brown cam to be rotated counterclockwise which engages it onto the rope.

6) During 5c when the press fitted steel shaft is within about 10 degrees of its desired location the clutch pulley is turned in the opposite direction (of the previous motion) which engages the clutch pin on Pulley A (and therefore the motor shaft) to the body of the robot. This does not require any sort of alternating locking-motion (as discussed above).

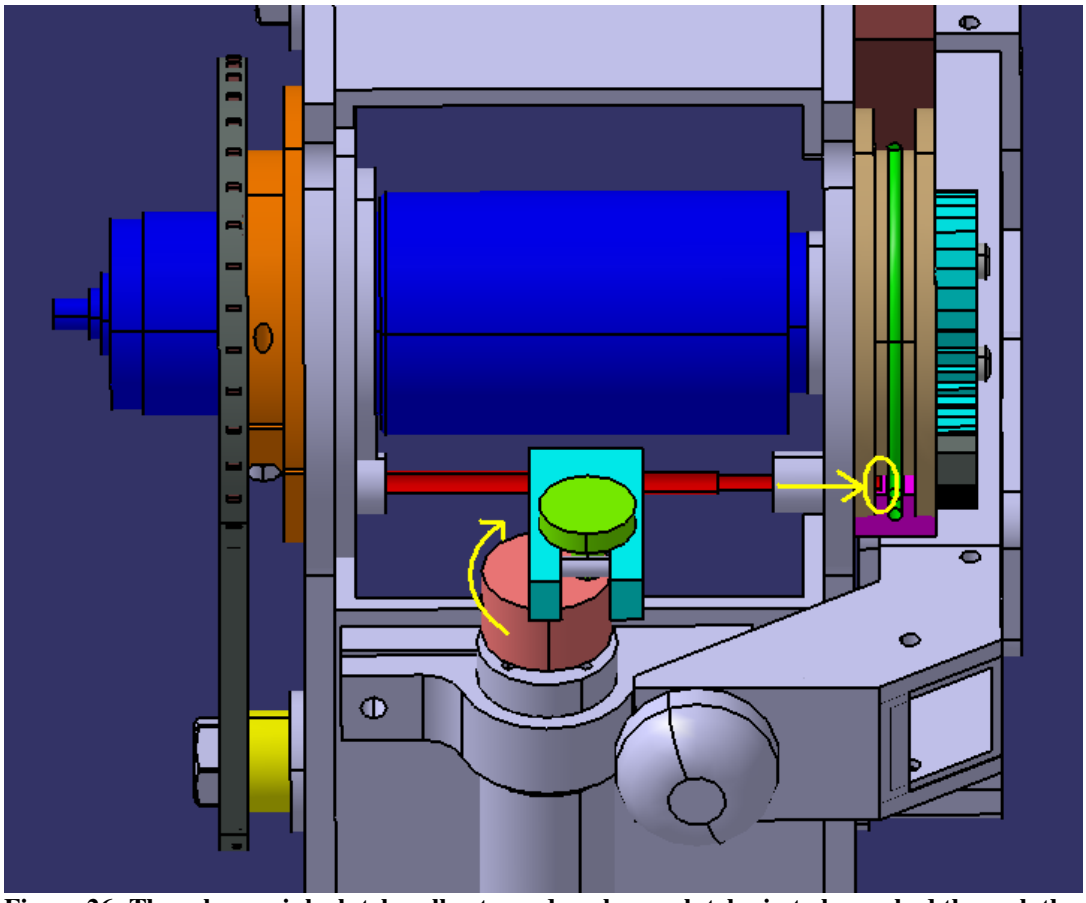

Figure 26: The salmon pink clutch pulley turned as shown clutch pin to be pushed through the tan pulley. This locks the shaft of the motor to the housing of the robot.

7) At this point the robot is ready to hop and will enter into phase 2 (hopping phase) of the up-hop procedure. All the tension we stored during phase 1 can now be used to hop by simply turning the Motor counterclockwise (if looking at the visible face of Pulley B). The housing of the motor (which is fixed to Pulley B, which is also fixed to a large sprocket) is turning around the shaft. This large sprocket turns, via a # 25 chain, a small sprocket which is connected to the lowest (earthward) linkage and disengages it from its locked position which then causes the spring tension to be utilized to force the pogo down and create a hop. Prior to the activation of the hop the

robot must be canted slightly towards the stair in order to provide the forward motion in conjunction with the upward motion required to get on top of a stair.

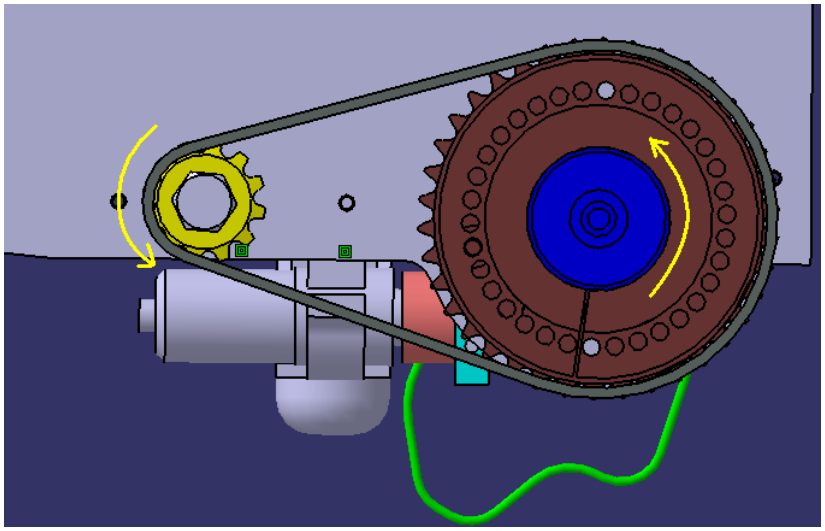

Figure 27: The Blue motor housing (which is attached to the orange pulley, which is fixed to the brown sprocket) is turned counterclockwise which turns the yellow sprocket counterclockwise via a chain.

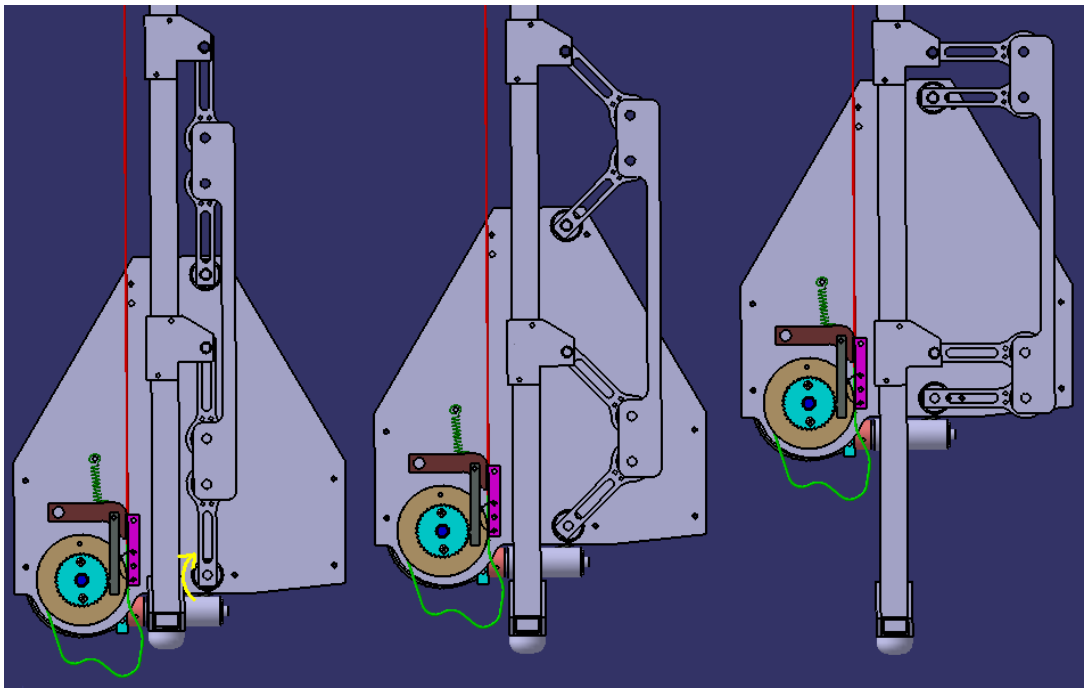

Figure 28: The yellow sprocket is attached to the bottom link of the coupled four bar linkage causing it to unlock. Once the coupled four bar linkage is unlocked the energy stored in the red rope-spring is then used to drive the pogo assembly down as shown.

8) As the robot becomes airborne the voltage of the clutch motor is then reversed causing the clutch pin engaged on Pulley B (in this configuration the body of the robot is fixed to the housing of the motor). This engagement may be require some locking movement, but if the motor simply continues to travel the same direction as it was when it was actuating the hop the pin extension of the clutch pin should slip in easily. (See Figure 23)

9) While still airborne the motor (housing fixed to the robot body) is turned the opposite direction (motor shaft is turning clockwise if facing visible side of Pulley A) triggering 3 events.

9a) Turing the motor is the above mentioned direction engages the pawl on the ratchet (which is fixed to Pulley A) pulling the pawl earthward.

9b) When the pawl is pulled earthward the cam is then turned clockwise about its pivot point which releases its clamping of the rope

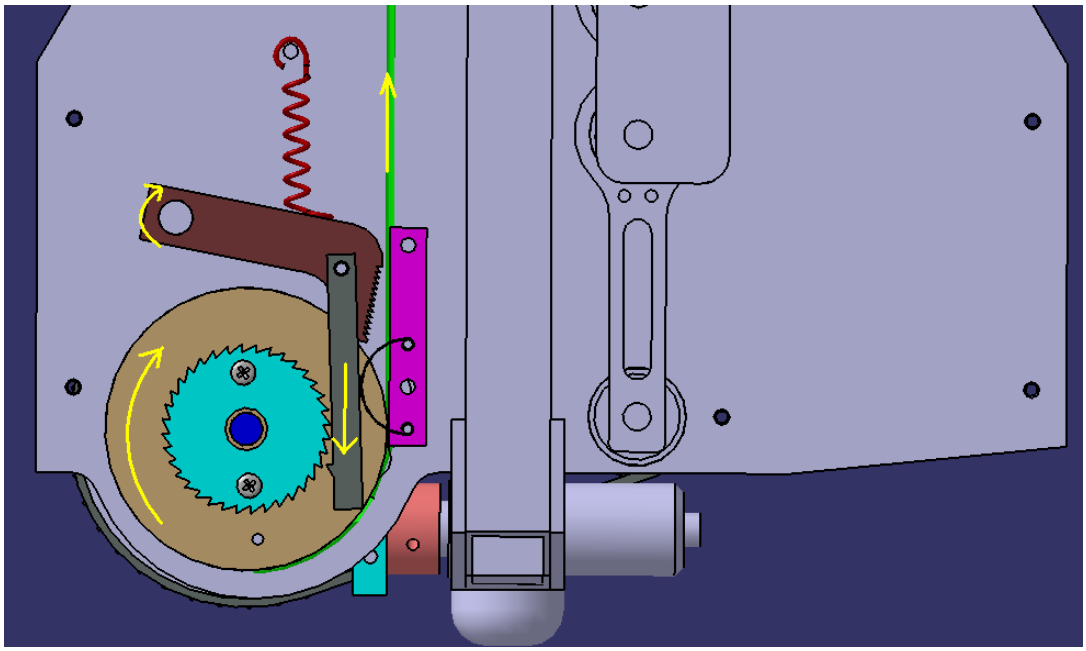

Figure 29: The Tan pulley is turned clockwise which pulls the dark grey pawl down. This motion turns the brown cam clockwise which disengages the cam action on the rope-spring. When the cam is disengaged the slack in the rope is freed quickly.

Note: At this point the robot's pogo assembly is extended, one of two operations may be implemented. Either the pogo assembly is pushed back to the locked position by the force created due to the landing of robot onto the top of the stair. Or the pogo assembly is sucked back in with the power of the motor. The latter of these options will be considered the "Optional" for the purposes of this paper.

9c) When the camming of the rope is disengaged the rope becomes free of tension making it easy for the device to be put into rover mode at the landing location. The body of the robot (acting under gravity) should be enough weight to being the robot back into rover mode.

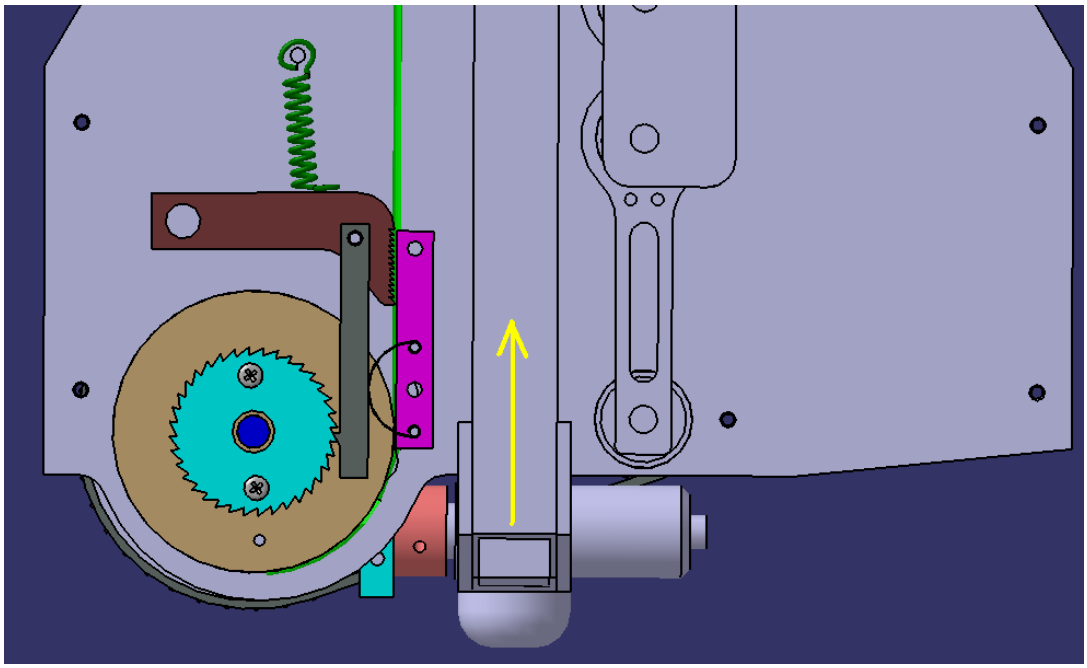

Figure 30: With no rope spring to resist motion of the pogo assembly it is easily pushed back into place my the force of the landing robot.

9c Optional) When the camming of the rope/spring is disengaged the rope/spring becomes free of tension making it easy for the device to be put into rover mode during flight. The clutch pin engaged on to the tan pulley locking the motor shaft to the housing of the robot (See Figure 26). The motor housing (which is attached to pulley B which is fixed to the large sprocket on the back) is turned clockwise (if looking at the visible face of pulley B) which turns the smaller sprocket clockwise. This action turns the bottom linkage counterclockwise (if looking at the visible face of pulley A) which sucks the pogo assembly back up into the body of the robot into the locked position.

10) During its time airborne a tracking problem will be implemented in order to bring the robot into a proper landing angle. This landing angle will most likely be the reflection of the angle at which the robot when it began its hop. This tracking will be made possible by reaction wheels (also used to movement in rover mode).

11) Though this landing should be such that there robot is perfectly in the upright rover mode, there will be un-modeled disturbances. These disturbances will be dealt with by implementing the stabilization algorithm for upright rover mode.

Note: At this point the cycle can begin again if the robot wishes to go up another step or series of steps.

2.3 Down Hop procedure

2.3.1 Introduction

This procedure details the functional description of a method to traverse a drop off should the robot need to come off of a table or other high object. Data regarding the orientation and dimensions of the object to be hopped off of will need to be provided to the robot prior to this operation.

2.3.2 Procedure

1) The robot will begin this procedure in upright rover mode. It will then proceed to locate the object's edge (the location of where the down hop is to be preformed) and orientate itself to achieve the proper landing point. Phase 1 (preparation) of the down hop procedure begins when the robot is properly configured and in upright rover mode.

2) The clutch pin will be engaged (by supplying voltage of the correct sign +/- to turn the clutch pulley the appropriate direction) to Pulley B thus fixing the housing of the motor to the body of the robot (some locking-movement may be required). See Figure 23

3) Tensioning of the spring will now commence. The motor is turned clockwise (looking at Pulley A from its visible face) which starts three events (the same events as the up hop 4a-c). See Figure 24

4) Again, as in the up hop procedure 5-6, the motor is turned the opposite way (counterclockwise). This builds up the slack (see Figure 25) on the rope below (earthward) the clamping point which can be released quickly when the motor is turned back clockwise. Additionally the steps discussed above engage the clutch pin back onto pulley A making the motor housing spin about the shaft. See Figure 26

5) Now the robot will enter phase 2 (hopping phase) of the down hop procedure. The robot will then use the solution of a trajectory map to accelerate to the proper release velocity as it drives off of the edge in upright rover mode.

6) While airborne the robot will angle itself using a solution to a tracking problem to the appropriate landing angle through use of the reaction wheels, similar to the up hop procedure.

7) Also while airborne an ultrasonic range finder will ping the ground several times to come up with an estimation as to when the robot will be within a 1-5cm range from the ground.

8) When the robot is in the 1-5 cm range from the ground (actually before as there is time required to unlock the pogo assembly.) the robot will turn the motor housing linked to pulley B and a sprocket which unlocks the pogo and transfers all of the potential energy into the ground. This action dissipates the energy from a down hop as well as re-tensions the rope/rubber band assembly.

9) As soon as the pogo has absorbed the energy from the down hop a tracking problem is then initialized which will keep the robot in upright rover mode. This tracking problem will utilize the reaction wheels to stabilize the robot during its violent landing.

10) The motor is now thrown into reverse (clockwise looking at the visible face of pulley B) locking the pogo back in place while the clutch pin is re-engage it onto pulley B.

11) Once pulley B is engaged by the clutch pin the motor will now turn pulley A clockwise (looking at the visible face of pulley A) which releases the tension on the rope/rubber band assembly. At this point we are ready to start the cycle again, or transition to an up hop cycle.

2.4 Other I-hop Related Devices

2.4.1 Ultrasonic Range Finder Test

Rationale: Test an ultrasonic range finder in order to make sure it is functional and accurate enough. It must also to be able to calibrate easily.

Test stand[3]: The test stand mounts the range finder in order to position it so that it roughly emits a sound pulse parallel to the ground. This pulse then hits a movable piece of aluminum and bounces back. The space between the object (piece of aluminum) and the range finder is marked in order to provide a scale for calibration purposes.

[3] See Appendix D for program

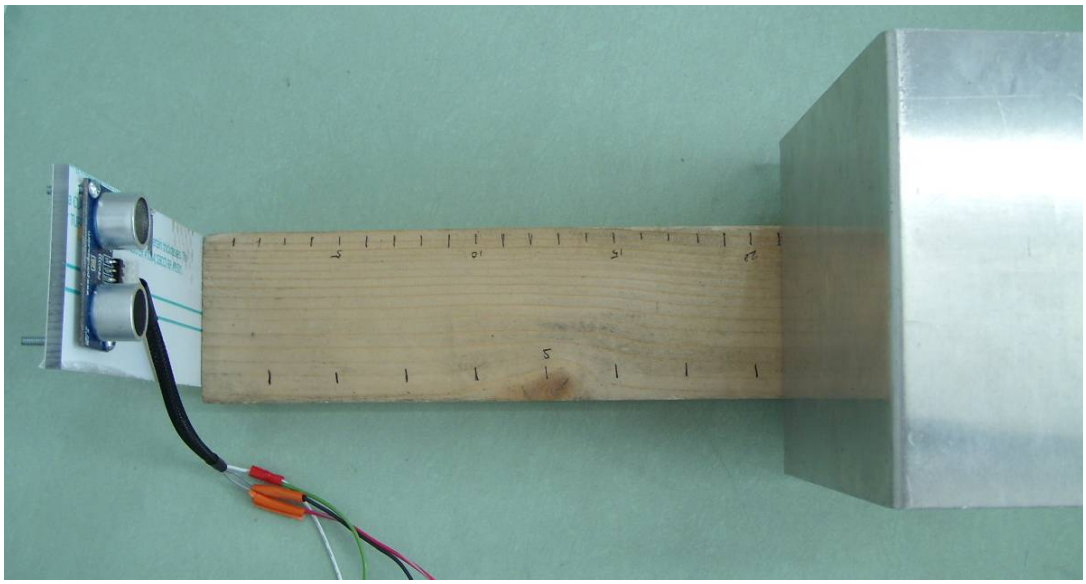

Figure 31: Ultra sonic range finder test stand

Theory of operation: First a logic level (5V) trigger pulse is sent from the DAQ card (NI PCMCIA 6036E) to the ultrasonic range finder (USRF) via the Signal I/O pin. The length of the pulse (time) must solely be greater than 10µS and in this test is several times longer. The USRF waits 200µS before sending the sound pulse out and switching the signal channel from low to high (5V). When the ultrasonic pulse echoes back and the sensor's microphone registers the ultrasonic pulse the signal channel switches back to low. The length of the pulse (after the signal pulse) is proportional to twice the distance from the range finder to the object.

Test results: With a sampling time of 8000Hz I was able to get about half an inch of resolution on the rangefinder. In implementation on the robot it would be beneficial to use a separate devoted A/D converter to achieve a better sampling time.

2.4.2 I Fling

2.4.2.1 Introduction:

In an attempt to draw in funding we started forming our robots to mesh with their intended consumer. One target market was toy companies locally based in San Diego. I was given the task of making I-hop fun. I devised a design of a Jai-alai (or Track ball) type throwing apparatus. This device could simply be mounted at the top end of I-hop, and control trajectories designed to model the flinging motion someone goes through to throw a jai-alai ball (pelota) with the woven basket (cesta).

Since this was to be marketed as a toy, some major design modifications needed to take place. Since a robot that contained enough energy to hop up approximately a meter is quite dangerous in the hands of a child the entire hopping mechanism would have to be removed converting I-hop to just a up-righting rover, which could roll around on two or three wheels. Since one of my associates in the design studio was working on wirelessly driving up-righting rover I could simply mount my design on his robot and see how it worked.

2.4.2.2 Design:

I-Fling is an apparatus that can throw a ball like a cesta but it can also pick up a ball using a flexture joint. The main components are made of 1/8 inch poly carbonate which is durable and flexible. As seen in Figure 33 the inner “track” on which the ball rolls on is also a flexture joint which allows a ball to travel through if forced as seen in Figure 35 and Figure 36. The physical device was constructed such that the distance between the flexture joints could be varied to find the optimal

distance. If the distance was too great the ball would not be able to be thrown well. If it was too small the ball would not travel through the flexture joint or be propelled by the flexture joint out of the basket. As long as this ball is light enough (we are using ping-pong or small whiffle balls) then the throwing motion should not exert enough force to push the ball back through the flexture upon throwing. This design also prevents misuse of the throwing device as if an object that would cause any significant amount of harm to an individual was placed in the flexture it would simply fall through the flexture upon actuation of the throwing motion.

The purpose of the cesta type basket is to induce a spin on the ball thrown inducting lift causing the ball to be aloft longer and thus go farther. During fabrication of the flexture joints they are cut with a Laser-Cam which leaves a rough edge on the track surface, which is advantageous as it provides a type of gripping often seen on Trak® balls which adds to the spinning. Additionally the release angle is important to obtain the maximum trajectory. This angle was found my designing the optimal release angle of the device to be at approximately 45 degrees from horizontal an testing what release angle threw the ball further. Through simple throwing motions that were stopped by a rigid body it was determined at 30 degrees was the optimal throwing angle and the design was modified such that at if the pendulum was stopped at 15 degrees past vertical by the reaction wheels the ball would be thrown at the optimal angle (shown isometrically in Figure 32.

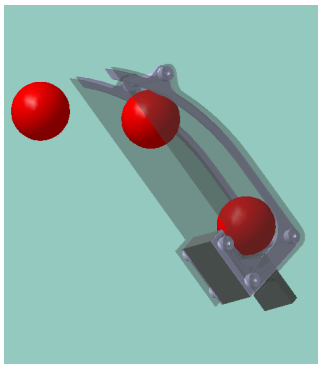

Figure 32: CAD model of I-Fling throwing a red ball

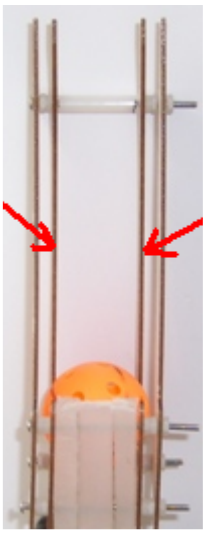

Figure 33: Physical I-fling Front view. Arrows point to flexture.

Figure 34: Physical I-Fling Side view.

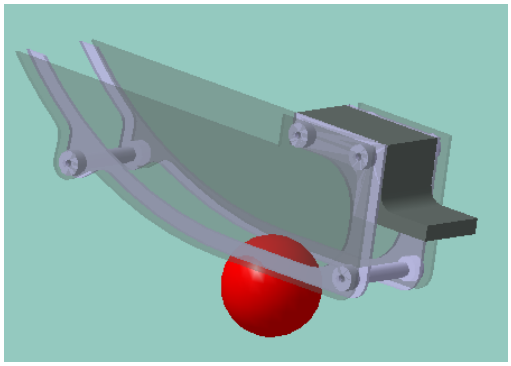

Figure 35: I-Fling Picking up a red ball step 1

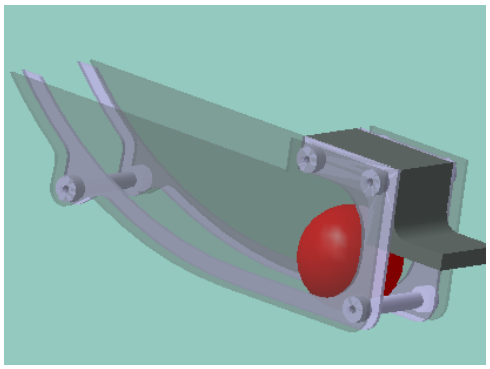

Figure 36: I fling picking up a red ball step 2

Special mounting considerations would have to allow a wheel to still be able to roll as the robot drove in 3-wheel mode yet still allow the ball to be picked up with the flexture. Round nylon skids were mounted to the surface of I-fling which would be dragged in 3-wheel mode replacing the wheel until the wheel (which would have to be light and roll in two directions) could be optimized.

3 Theory

3.1 Pendulum problem:

This is the classic inverted pendulum problem. This problem boils down to a two dimensional problem with a pendulum mounted on a rotational joint to a movable base. We will be able to measure the position of the cart and the angle of the pendulum but not the other two system states (speed of the cart, angular speed of the pendulum). The control input for this experiment is voltage which is proportional to force on the cart.

Note: In order to find a good transfer function between voltage and force on the cart experiments need to be implemented to model that relationship. For this experiment force was graphed as a function of voltage in Microsoft® Excel which was used to come up with a good transfer function via a best fit curve.

3.1.1 Equations of motion:

The equations of motion for this classic problem have been derived in Numerical Renaissance and are below in the form M*X=N (Where X is the state vector):

Equation 1:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(X3)(mp)d & 0 & I+(mp)d^2 \\ 0 & 0 & 1 & 0 \\ 0 & mt+mp & 0 & \cos(X3)(mp)d \end{bmatrix} \begin{bmatrix} \dot{X}1 \\ \dot{X}2 \\ \dot{X}3 \\ \dot{X}4 \end{bmatrix} = \begin{bmatrix} X2 \\ -gd(mp)\sin(X3) \\ x4 \\ U-bX2+d(mp)\sin(X3)(X4)^2 \end{bmatrix}$$

X1: cart position

X2: cart velocity

X3: pendulum angle

X4: pendulum velocity

Mp, I, b, d: system parameters

U: control input (force on the cart)

First I must obtain the linearized equations of motion form the nonlinearized equations. This is done by taking isolating the state variable X by moving M to the right hand side of the equation

Equation 2:

$$X = M^{-1}N$$

and then by taking the jacobian of the system with respect to both the state vector and the control input variable. The final form of the system is now in the classic control form:

Equation 3:

$$\frac{\partial X}{\partial t} = AX + BU$$

3.1.2 Full state feedback stabilization:

At this point we can proceed down two different routes, working with a discrete system or with a continuous system. The first route is conversion of the continuous time system to a discrete time system and finding the gain schedule by marching a difference equation. The second route is finding the gain schedule by marching a continuous time equation to find the gain schedule.

Note: In this case the gain schedule will not change over time as our trajectory that we are stabilizing about does not change.

3.1.2.1 Gain scheduling with a infinite horizon continuous time system:

To find the gain K for the inverted stabilizing pendulum we must solve a CARE (continuous time algebraic lyapunov equation) for an intermediate variable Ω(normally it is called X but I do not want to confuse Ω with the state). We can utilize this CARE equation because we know that this system is an infinite horizon problem and at steady state there is no change in the state (d Ω /dt=0).

Equation 4: This equation can be solved on with Matlab® with the "care( )" function where X(T)=Q_T

$$\frac{-d\Omega}{dt} = A^H\Omega + \Omega A - \Omega B Q_u^{-1} B^H \Omega + Q_x$$

Q_u, Q_X: Have weighted norms that dictate performance of the system

Equation 5: This is now the K that will minimize the cost function J which is a measure of the system objective

$$K = -Q_u^{-1} B^H \Omega$$

3.1.2.2 Gain scheduling with a infinite horizon discrete time system

First the system must be discritized using any number of methods, for the purposes of this paper I have chosen to use the c2d command in Matlab® as it is easy to use and prepackaged (the used time step depends on the application). This conversion requires you to put the system into the "sys" form.

After conversion on the system from continuous time (A B C) to discrete time (F G H), a finite difference equation is solved:

Equation 6: Can be solved using Matlab® dare( ), where X_{K+1}= 0

$$X_k = F^H X_{k+1} F - F^H X_{k+1} G (Q_u + G^H X_{k+1} G)^{-1} G^H X_{k+1} F + Q$$

The solution for this equation is used to compute the gain K (that minimizes the cost function) for the system

Equation 7:

$$K_k = -Q_u^{-1} G^H F^{-H} (X_k - Q_x)$$

3.1.3 Partial State Feedback

3.1.3.1 Utilizing Kalman Filter (Continuous time) to gain full state estimate

A Kalman Filter uses knowledge of the equations of motion as well as the measurable states to come up with an estimate of immeasurable states. Our continuous time system estimate equation boils down to the following:

Equation 8:

$$\frac{d\hat{X}}{dt} = (A + LC)\hat{X} + BU - LY$$

^: symbolizes an estimate

Y: subset of X containing measurable states

L: Output injection parameter

The term L is found to minimize a new const function (J) which is a measure of the estimation error.

Equation 9:

$$J = \int_0^T trace(P)dt \geq 0$$

P (a means to L) is now computed via a Continuous-time Riccati equation (CARE):

Equation 10:

$$\frac{dP}{dt} = AP + PA^H - PC^H Q_2^{-1} CP + Q_1 \qquad P(0) = P_0$$

This equation can be solved with an RK4 approach or a packaged Matlab® CARE solver. In my case I have used the packaged Matlab® solver. L is then computed from P which will fill in all the blanks so that we may gain a full state estimate:

Equation 11:

$$L = -PC^H Q_2^{-1}$$

This is now the L in (eq) which minimizes the estimation error of the system and can be used to come up with the next state estimate using (eq). So now we can have some sense about the full state which is needed to determine the control to be implemented.

3.1.3.2 Utilizing Kalman Filter (Discrete Time) to gain full state estimate

As in the continuous time Kalman filter our object is to come up with an estimation of the *full* state. The classic system boils down to a few finite difference equations:

Equation 12:

$$X_{k+1} = FX_k + Gu_k \qquad Y_k = H\hat{X}_k$$

Equation 13:

$$\hat{X}_{k+1} = F\hat{X}_k + Gu_k - L(Y_k - \hat{Y}_k) \qquad \hat{Y}_k = H\hat{X}_k$$

Equation 14:

$$\tilde{X}_{k+1} = F\tilde{X}_k + L\tilde{Y}_k \qquad \tilde{Y}_k = H\tilde{X}_k$$

The L matrix is designed to drive the estimation error to zero as it is based off a quadratic cost function which is a measure of both the estimation error and the corresponding control error. P which is a stepping stone to L is found by marching Equation 10

Subsequently L is found by computing Equation 11

3.1.3.3 Control for Discrete & Continuous time systems using state estimation

We have now come up with an estimation of our full state by utilizing a cost function minimization to come up with the variable L. Additionally we have used knowledge of full state to come up with a control gain K that is derived from a cost function minimization. Now to put it together to give us something useful we combine the state estimation and the gain K to come up with a control U.

Equation 15: Continuous time control

$$U = K\hat{X} + R$$

Equation 16: Discrete time control

$$U_k = K\hat{X}_k + R_k$$

As you see the estimation, (some of which is measured and some of which is computed) the full state feedback gains (K, which are computed), and some target reference state are used to compute the implemented control.

3.1.4 Trajectory Implementation

In the previous sections we were dealing with the infinite horizon solution. What happens if we do not want to just track a constant but an entire trajectory such as the inverting of a pendulum. For the purpose of this section we will assume that the

trajectory has been determined, we will determine how to come up with the trajectory in the next section.

So the only difference between trajectory tracking and constant tracking is that there is a different K and L for each step in the trajectory. I will assume that the trajectory is a discrete sequence of intended states X_I and the intended control U_I. The problem now becomes a disturbance rejection about the intended state, and will perform the same sort of operations as above but for each individual state.

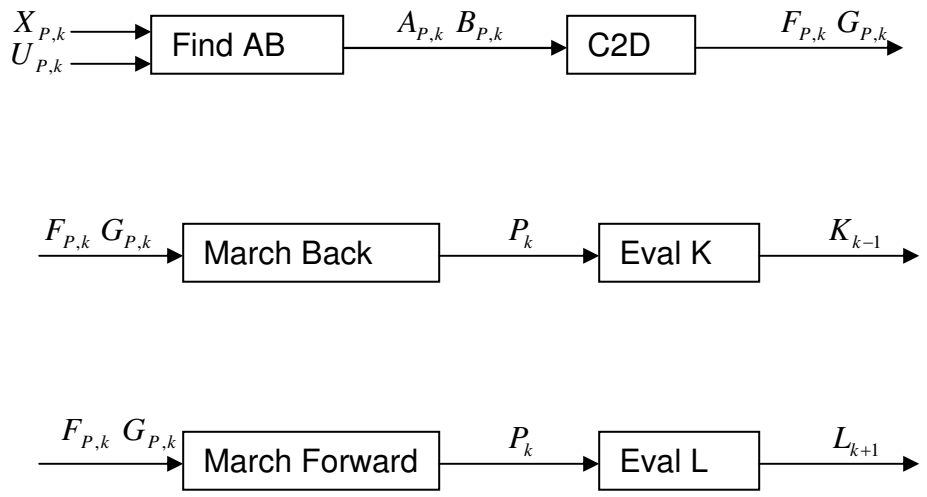

Figure 37: Discrete time flow chart showing derivation of inputs to the estimator (see Figure 38)

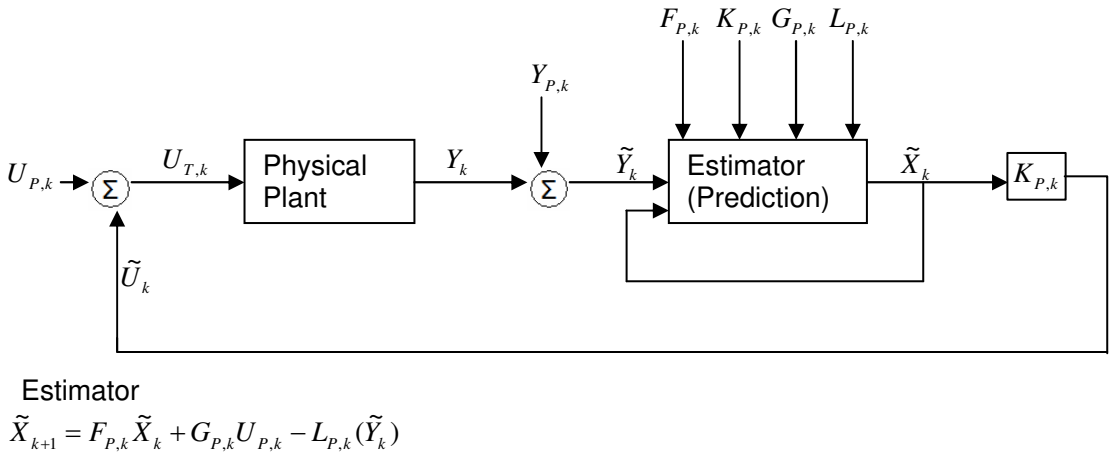

Figure 38: Implementation of control in a discrete time partial state feedback system. The subscript "P" denotes a pre-computed value. The subscript "T" represents a Total value.

3.1.4.1 Model Predictive Control

This section we will cover how to determine the pre-computed gain schedule and control schedule using model predictive control. Our lab[4] has developed a tool (MPD-OPT) to apply the theory covered in this section. Utilization of the MPD-OPT will be discussed in the following section.

When first tackling this issue of model predictive control to obtain a desired system trajectory one must first design a cost function. For this project I have chosen to select a quadratic cost function (standard) that utilizes a measure of the weighted square of the state error and control effort as well as terminal conditions.

[4] Sean Summers is the primary person who came up with the MPDopt tool that is used in this thesis to find control trajectories

Equation 17:

$$J = \frac{1}{2}\sum_{k=0}^{K}\left(X_k^H Q_X X_k + U_k^H Q_u U_k\right) + \frac{1}{2}(EX(T))^H Q_T EX(T)$$

more than others. This weighting will dictate in what manner the system will achieve its desired terminal condition.

So now we need our cost function to be a function of only the control U such that we can obtain a gradient of that cost function with respect to U. The first step to achieve this is to guess a control trajectory and march the state equation forward in time based on that guess. The method that was used for this marching is RK4.

The fourth-order Runge-Kutta method (RK4) is a way to march an ODE in time. I have selected to use it in this case for its accuracy and ease of implementation. This method uses a weighted average of 4 function (ODE) evaluations to come up with the next value as time is marched.

Equation 18:

$$k_1 = \Psi(X_n, t_n)$$
$$k_2 = \Psi(X_n + (h/2) * k_1, t_{n+1}/2)$$
$$k_3 = \Psi(X_n + (h/2) * k_2, t_{n+1}/2)$$
$$k_4 = \Psi(X_n + h * k_3, t_{n+1})$$
$$X_{n+1} = X_n + h * (\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4)$$
Ψ : *Ordinary differential equation evaluation*
h : *step size*
X : *state of system*

Now that we have an evolving state based on the control we can now march an adjoint vector (r) backwards in time which is a function of the state evolution that we just obtained.

Equation 19:

$$E^H Rr_k = F^H Rr_{k+1} + Q_x X_k$$
F : *Linearization of Nonlinear state equation*

The knowledge of the evolved state vector and the adjoint vector can now be use do calculate the gradient of our cost function with respect to our control U.

Equation 20:

$$\left(\frac{DJ}{DU}\right)_k = \left(G^H R r_{k+1} + Q_u U_k\right)$$

r : *adjoint vector*

G : *from linearization of nonlinear equation*

U : *Control input*

R, Q_u : *design selections*

Since we now know the gradient of our cost function with respect to our control U. We can minimize this gradient, or figure out what the best "direction" to push the system is, as well as how "far" to push the system. Via any number or methods (conjugate gradient will be utilized in this paper) we can arrive at our control trajectory, which will in turn give us a state trajectory (utilized above).

3.1.5 The Conjugate Gradient method

The so called "Conjugate Gradient" method for minimization utilizes slightly more knowledge than its simpler cousin the steepest decent method to come up with a better minimization of a function. Instead of simply proceeding in a direction, stopping, finding downhill (negative gradient) and then proceeding again, the Conjugate gradient method uses information that is analogous to momentum in its calculations. By using more than solely a 90º turn at each iteration the conjugate gradient method converges much faster than the steepest decent method (in any sort of function that is solved trivially in two steps with steepest decent).

The Conjugate gradient algorithm that follows uses knowledge of previous decent directions to come up with a new decent direction.

Equation 21:

$$X_k = X_{k-1} + \alpha_k P_{k-1}$$
$$P_k = R_k + \beta_k P_{k-1}$$
$$\alpha_k = \frac{R_k^T R_k}{P_{k-1}^T A P_{k-1}}$$
$$\beta_k = \frac{R_k^T R_k}{R_{k-1}^T R_{k-1}}$$
$$X: \quad State\ of\ the\ Cost\ Function$$
$$R: \quad Steepest\ Decent\ Direction$$
$$P: \quad Direction\ of\ Decent$$

The beauty of this method is that for a just a few more intermediate calculations one can converge with a greatly reduced number of steps than the steepest decent method.

4 CAD (Computer Aided Design) Notes

4.1 Introduction

The purpose of CAD in the design process I implemented is to create virtual prototypes. These prototypes are essential to progressing quickly through many iterations with out actually having to expend materials and money build something. CAD also serves as just a brainstorming platform to simply see if things are feasible; is there enough space, will this interfere with motion somewhere else? All of these questions can be answered with CAD.

A drawback to CAD is that it only models what you give it to model. You may forget to include bounds in machining tolerances, discrepancies in drawings vs. what you actually get when you order a manufacturer part. Even when doing kinematic simulations you may know the major forces on a body but not minor forces like thermal expansion or air resistance. So to overcome these problems it is always preferable to create as detailed a model as possible but don't forget it might not work in the real world.

4.2 Notes on CATIA V5

I have used many different types of software[5] through my years at UCSD and my previous employers but have found CATIA™ V5 to be the best in most situations. Some of the benefits of CATIA™ V5 include: it is a post or pre dimension sketcher, you can undo tens of steps, the file management system is not a memory hog, it has a kinematic model toolbox, editing parts is fairly easily done, and it is PC based. But like anything in life it also has a set of drawbacks: Hard to obtain, does not integrate its kinematic models with its static models easily, and the file saving system will be a memory hog if you want to create full backups of an assembly.

When you have a pre dimensioning sketcher (you add the correct dimensions as you draw a sketch) it is easy when you have an item that you know the measurements and simply want to create a solid model. But if you are designing and you do not have the part but are creating it for the first time you don't really know all of the dimensions and you would probably like the option of changing several

[5] Inventor V5-2008, Solid works, Catia V4, ProE, Autocad

dimensions after you have sketched it so that it either fits or works better. CATIA™ V5 is wonderful as it gives you the option or being a pre or post sketcher so you get the best of both worlds.

Another important aspect to CATIA™ V5 is the ability to "undo" many steps. Some programs like Pro E only allow you to undo a very limited number of steps without totally negatively altering a model but CATIA™ V5 allows you many more. This is especially critical since experimentation is most of the design process, so you have to be able to take risks in your model and feel confident that you can undo really bad errors without having to go and load a back up copy.

One of the biggest benefits to CATIA™ v5 is it allows you to save you model or assembly constantly without occupying a huge amount of computer memory. What it comes down to be you can save over your files constantly; this allows you to feel confident that if you have a major error then you can back out the old model or assembly. Other systems like Pro E create an entire back up when you save (a completely new file set).

CATIA™ V5 also has the standard Windows® based configuration as most new software does. This prevents the user from having to use a devoted machine as CATIA™ V4 requires. Additionally CATIA™ V5 has a kinematic section as most new solid modeling programs have, but it was not intuitive as to how to integrate the solid model and the dynamic model.

Finally one of the best parts about this software is that editing is easy most of the time. For example, if you are modifying a sketch of an extrusion of some part that

you have already modeled, CATIA™ V5 allows you to just locate that sketch in the drawing tree and right click and edit it. When you are done editing you can go back to 3D space and just click update. If it was a simple change that was just a minor modification of a dimension that's all you have to do. However if you are doing some major editing like changing a sketch in a constrained subassembly you will see error messages. The user can then go back in the drawing tree and address each of those error messages individually. The user doesn't have to delete many constraints or sketches but they can systematically fix all the errors by modifying the constraints or sketches affected with a user friendly editing window.

On a more clerical note CATIA™ V5 student version is somewhat difficult to obtain. Usually your school has to be in the "HEAT" program and you have to have a school email address and be an active student. But I definitely recommend purchasing the student version if you are able, many of the tools are similar to most other CAD programs which makes it easy to learn and powerful to use.

4.3 General CAD Design notes

I think that is important to cover some basics that I learned during this design process as to how to construct a good CAD file. The key points I want to address are: modularity, sub assemblies, conformity in constraints and using the drawing tree. Creating a good CAD file off the bat can save lots of time when a model has to be altered.

First on the subject of modularity you should always aim to have interchangeable parts. Use a lot of clearance holes and leave yourself room for some fabrication error. Additionally try to have each sub assembly be easily removable from the master assembly, try not to project geometries from sub assemblies so that when you alter one part you don't have to go change another before it works. When projecting a majority of geometries try to trace over the projection and use that in your sketch, this will restrict part linking which is a major problem when you try to edit a file. By not part linking you may have to change several parts that all tie into the same sketch but it avoids a myriad of errors when you change something.

Subassemblies should be used when ever you have an apparatus that can be considered its own entity. Typically I like to make things sub assemblies when they are rigidly fixed together. The most important thing about sub assemblies is to use them wisely, try not to get too deep into the drawing tree's branches (i.e. a quadruple embedded sub assembly when 2 sub assemblies will do). Also make note of the fact that as far is the top assembly is concerned each subassembly is treated as a part, meaning it moves as a rigid body. If you want to show motion of a part of the subassembly you will have to switch from editing the top assembly to editing the subassembly.

Using conformity in constraining parts (or sub assemblies) will help down the line if and when you have errors in your constraints upon altering a body. I suggest making a list as to how you constrain a rectangle to a face etc. You may even want to write your self a note on that part (under properties) saying why you used 1 hole rather

than another to make a concentric constraint. When constraining remember your 6 Degrees of Freedom (DOF), try to use as little constraints as possible to achieve your desired DOF so when you have to edit (or fix) your constraints you only alter a few constraints rather than many. Finally try to think of Datum planes, lines, points that you can constrain everything from, it will ease your process of editing if you know between which two objects the constraint is.

A final note is to make sure your drawing tree is wider then it is tall by a factor of approximately 7-15 (7-15 parts/subassemblies on a top assembly and so on). You definitely want some sub assemblies if your top assembly is getting cluttered. Making a drawing tree look uncluttered can help you find and change parts more easily later on.

Appendix A

Equations of Motion for I-Hop (derived by Chris Schmidt-Wetekam)

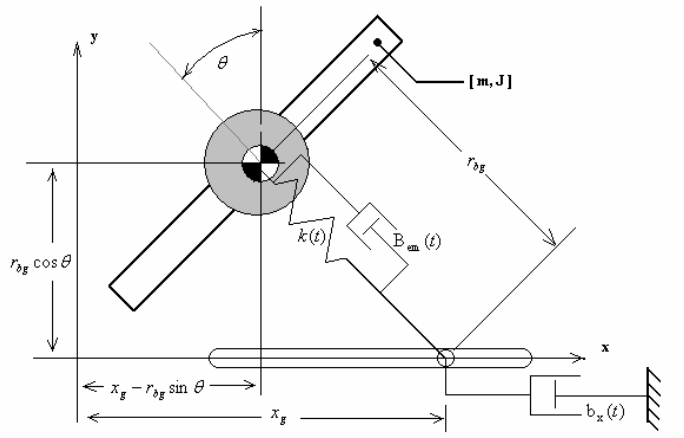

Figure 39: I-hop Equations of motion Diagram

Linearized Equations of motion

Equation 22:

$$\dot{\vec{x}} = A(t)\vec{x} + B(t)u$$

$$A = \begin{bmatrix} -Nb_{em} - b_{ap} & mgr_{bg}(t) & 0 & 0 & Nb_{em} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & m\ddot{r}_{bg}(t) & -b_x(t) & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ b_{em} & 0 & 0 & 0 & -b_{em} - b_{aw} \end{bmatrix} \text{ and } B_1 = \begin{bmatrix} -Ns \\ 0 \\ 0 \\ 0 \\ s \end{bmatrix}$$

Table 1: Appendix A variable guide

Parameter	Category	Description
g		Gravity
m	Robot Body	Total robot mass
J		Air drag damping
b_{ap}		Robot body windage
J_w	Reaction Wheels	Moment of inertia
b_{aw}		Air drag damping
N		Wheels per axis
k_t	Reaction Wheel Motors	Torque constant
γ		Gear reduction
v		Supply voltage
R		Terminal resistance
$s \equiv \frac{\gamma k_t v}{R}$		Stall Torque
$b_{em} \equiv \left[\frac{(\gamma k_t)^2}{R} + b_{friction}\right]$		Electrical and mechanical damping coefficient
k(t)	Hopping Mechanism	Spring constant
r_0		Takeoff/landing height
P		Spring pre-tension
ρ		Pinion gear radius
K_t	Hopping Motor	Torque constant
G		Gear Reduction
V		Supply Voltage
R_h		Terminal resistance
$S \equiv \frac{GK_tV}{\rho R_h}$		Stall force at supply voltage
$b_i(t)$	Radial Dynamics	Time-varying takeoff impact dissipation coefficient
$B_{em}(t) \equiv \frac{(GK_t)^2}{R_h} + b_i(t)$		Electrical and mechanical damping coefficient
$b_x(t)$	No Slip Condition	Time-varying horiz. damping coefficient

Note that the radial trajectory, $\left[\ddot{r}_{bg}(t),\ \dot{r}_{bg}(t), r_{bg}(t)\right]^T$, is assumed to be decoupled from the angular and translational dynamics during nominal operation.

Appendix B

Programming code single pendulum swing up problem

Swing Up

```
% SWING UP Discrete
%Swings up the pendulum along a given control trajectory
clc
clear all
close all
%defining initial quantities
mp=1;
mt=1;
d=1;
g=9.81;
b=0;%0.0052
I=0.083333;
Qt=eye(4);      %end state weighting
time_step=.01;
Total_time_steps=301;
C=[1 0 0 0;0 0 1 0];
Q=eye(4);
R=eye(2);
Qx=eye(4);
Qu=1;
%implementing a four loop to calculate K and L because our system is time varient

%***************KALMAN FILTERING PART FIND
L**************************************
P=0*eye(4);     %initializes the P value...a means to L
```

```
L_store(1:4,1:2,1)=zeros(4,2);

for k=1:Total_time_steps-1
   %finding A and B from linearizing equations of motion about the nominal state
of the system
    state_nom=return_state_measurement(k);
    u_nom=return_state_control(k);                %nominal control of the system
    [A, B]=findAB(mp, mt, d, g, b, u_nom, I, state_nom(1), state_nom(2),
state_nom(3), state_nom(4));

   %Convert plant into discrete space
   sys=ss(A,B,C,0);
   sysd=c2d(sys,time_step);
   F=sysd.a; G=sysd.b; H=sysd.c;

   F_store(1:4,1:4,k)=F;
   G_store(1:4,1,k)=G;

   %calculates the NEXT P which is a means to the NEXT L
   P=F*P*F'-F*P*H'*inv(R+H*P*H')*H*P*F'+Q;

   %finding L for the kalman filter and storing in 3D matrix
   L_store(1:4,1:2,k+1)=-P*H'*inv(H*P*H'+R);
end
%need to find last values of F and G
state_nom=return_state_measurement(Total_time_steps);
u_nom=return_state_control(Total_time_steps);
[A, B]=findAB(mp, mt, d, g, b, u_nom, I, state_nom(1), state_nom(2),
state_nom(3), state_nom(4));
sys=ss(A,B,C,0);
sysd=c2d(sys,time_step);
F=sys.a; G=sys.b; H=sys.c;

F_store(1:4,1:4,Total_time_steps)=F;
G_store(1:4,1,Total_time_steps)=G;

%*********PURTURBATION CONTROL PART FIND K
**************************************************************************
***
X=zeros(4);          %initializing X... a means to K
K_store(1,1:4,Total_time_steps)=zeros(1,4);
```

```
for k=Total_time_steps:-1:2
   %need to find F and G corresponding to the backwards march
   F=F_store(1:4,1:4,k);
   G=G_store(1:4,1,k);

   %finding the pervious value for X
   X=F'*X*F-F'*X*G*inv(Qu+G'*X*G)*G'*X*F +Qx

   %finding the K for the purturbation control problem and storeing in 3D matrix
   K_store(1,1:4,k-1)=-inv(Qu)*G'*inv(F')*(X-Qx);

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%IMPLEMENTATION%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%setting our initial states
x_truth=(return_state_measurement(1)).'+[0;0;.25;0];
y_measured=C*x_truth;
x_k_km1=(return_state_measurement(1)).'+[0;0;0;0];
u_purturb=K_store(1,1:4,1)*x_k_km1;

for k=1:Total_time_steps-1

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%STATE
MARCH WITH L AND K
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %A,B,K,L (pre_computed) are changing but i stored them so here im accessing
them
      F=F_store(1:4,1:4,k);
      G=G_store(1:4,1,k);
      L=L_store(1:4,1:2,k);
      K=K_store(1,1:4,k);

  %Compare y_measured with the planned trajectory to find the error
      y_purturb=y_measured-C*(return_state_measurement(k)).';

  %implements estimator using previous info and RK4
```

```
        x_kp1_k=(F+F*L*H)*x_k_km1+G*u_purturb-F*L*y_purturb

    % finds the total u to be mut into the ode
        u_purturb=K_store(1,1:4,k+1)*x_kp1_k;
        u_total=return_state_control(k+1)+u_purturb;

    %sets values for next itteration
         x_k_km1=x_kp1_k;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%MARCH
NON LINEAR SYSTEM%%%%%%%%%
    %marching the non_linear equations using RK4

    k1=eval_ode(mp, mt, d, g, b, I, time_step, u_total, x_truth);
    k2=eval_ode(mp, mt, d, g, b, I, time_step, u_total, (x_truth+.5*k1*time_step));
    k3=eval_ode(mp, mt, d, g, b, I, time_step, u_total, (x_truth+.5*k2*time_step));
    k4=eval_ode(mp, mt, d, g, b, I, time_step, u_total, (x_truth+k3*time_step));
    x_truth=x_truth+(time_step/6)*k1 + (time_step/3)*(k2+k3)+(time_step/6)*k4;
    y_measured=C*x_truth+0*rand(2,1);%finds the new state measurement and
corrupts with noise

    %stores the truth for compairson
    x_truth_store(1:4,k)=x_truth;

 end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 for k=1:300
     x=return_state_measurement(k);
     model_predictive_x_position(k)=x(1);
     model_predictive_x_speed(k)=x(2);
     model_predictive_theta_position(k)=x(3);
     model_predictive_theta_speed(k)=x(4);
 end

 actual_x_position=x_truth_store(1,1:300);
 actual_x_speed=x_truth_store(2,1:300);
 actual_theta_position=x_truth_store(3,1:300);
 actual_theta_speed=x_truth_store(4,1:300);
 time_steps_all=[1:300];
```

```
figure
plot(time_steps_all,actual_x_position,'r',time_steps_all,
model_predictive_x_position,'b');
legend('calculated x', 'predicted');
figure
plot(time_steps_all,actual_theta_position,'r',time_steps_all,
model_predictive_theta_position,'b');
legend('calculated theta', 'predicted');
figure
plot(time_steps_all,actual_x_speed,'r',time_steps_all,
model_predictive_x_speed,'b');
legend('calculated x dot', 'predicted');
figure
plot(time_steps_all,actual_theta_speed,'r',time_steps_all,
model_predictive_theta_speed,'b');
legend('calculated theta dot', 'predicted');

figure
 subplot(4,1,1)
 plot(time_steps_all,actual_x_position,'r+',time_steps_all,
model_predictive_x_position,'b');
 subplot(4,1,2)
 plot(time_steps_all,actual_theta_position,'r+',time_steps_all,
model_predictive_theta_position,'b');
 subplot(4,1,3)
 plot(time_steps_all,actual_x_speed,'r+',time_steps_all,
model_predictive_x_speed,'b');
 subplot(4,1,4)
 plot(time_steps_all,actual_theta_speed,'r+',time_steps_all,
model_predictive_theta_speed,'b');
```

Find_AB

```
function [A, B]=findAB(mp, mt, d, g, b, u, I, x1, x2, x3, x4)

A=[0,1,0,0;
   0,-(I+mp*d^2)/(-
mp^2*d^2*cos(x3)^2+mt*I+mt*mp*d^2+mp*I+mp^2*d^2)*b,mp^2*d^2*sin(x3)^
2/(mp^2*d^2*cos(x3)^2-mt*I-mt*mp*d^2-mp*I-mp^2*d^2)*g-
2*mp^4*d^4*cos(x3)^2/(mp^2*d^2*cos(x3)^2-mt*I-mt*mp*d^2-mp*I-
mp^2*d^2)^2*g*sin(x3)^2-mp^2*d^2*cos(x3)^2/(mp^2*d^2*cos(x3)^2-mt*I-
mt*mp*d^2-mp*I-mp^2*d^2)*g-2*(I+mp*d^2)/(-
```

```
mp^2*d^2*cos(x3)^2+mt*I+mt*mp*d^2+mp*I+mp^2*d^2)^2*(u-
b*x2+mp*d*x4^2*sin(x3))*mp^2*d^2*cos(x3)*sin(x3)+(I+mp*d^2)/(-
mp^2*d^2*cos(x3)^2+mt*I+mt*mp*d^2+mp*I+mp^2*d^2)*mp*d*x4^2*cos(x3),
2*(I+mp*d^2)/(-
mp^2*d^2*cos(x3)^2+mt*I+mt*mp*d^2+mp*I+mp^2*d^2)*mp*d*x4*sin(x3);
  0,0,0,1;
  0,-mp*d*cos(x3)/(mp^2*d^2*cos(x3)^2-mt*I-mt*mp*d^2-mp*I-mp^2*d^2)*b,
2*(mt+mp)/(-
mp^2*d^2*cos(x3)^2+mt*I+mt*mp*d^2+mp*I+mp^2*d^2)^2*mp^3*g*d^3*sin(x
3)^2*cos(x3)-(mt+mp)/(-
mp^2*d^2*cos(x3)^2+mt*I+mt*mp*d^2+mp*I+mp^2*d^2)*mp*g*d*cos(x3)-
mp*d*sin(x3)/(mp^2*d^2*cos(x3)^2-mt*I-mt*mp*d^2-mp*I-mp^2*d^2)*(u-
b*x2+mp*d*x4^2*sin(x3))+2*mp^3*d^3*cos(x3)^2/(mp^2*d^2*cos(x3)^2-mt*I-
mt*mp*d^2-mp*I-mp^2*d^2)^2*(u-
b*x2+mp*d*x4^2*sin(x3))*sin(x3)+mp^2*d^2*cos(x3)^2/(mp^2*d^2*cos(x3)^2-
mt*I-mt*mp*d^2-mp*I-mp^2*d^2)*x4^2,
2*mp^2*d^2*cos(x3)/(mp^2*d^2*cos(x3)^2-mt*I-mt*mp*d^2-mp*I-
mp^2*d^2)*x4*sin(x3)];

B=[0;(I+mp*d^2)/(-
mp^2*d^2*cos(x3)^2+mt*I+mt*mp*d^2+mp*I+mp^2*d^2);0;mp*d*cos(x3)/(mp^
2*d^2*cos(x3)^2-mt*I-mt*mp*d^2-mp*I-mp^2*d^2)];
```

Eval_Ode

```
function [ode_function]=eval_ode(mp, mt, d, g, b, I, time_step, u, x)
%Outputs the value of the non-linear function given its inputs
x1=x(1); x2=x(2);x3=x(3);x4=x(4);


M =[       1,           0,           0,           0;
           0, mp*d*cos(x3),           0,     I+mp*d^2;
           0,           0,           1,           0;
           0,       mt+mp,           0, mp*d*cos(x3)];

N =[x2;
   -mp*g*d*sin(x3);
    x4;
    u-b*x2+mp*d*x4^2*sin(x3)];

ode_function=(inv(M)*N);
```

Appendix C

Table 2: Cost Analysis

Project	Part	Cost	Date purchased	Supplier
High-Hop	Raw material	$14.27	2/13/07	CRMS
High-Hop	hopper leg	$4.70	1/30/07	Home Depot®
High-Hop	wire ties	$7.99	1/24/07	Linen's 'N Things®
High-Hop	rope	$2.70	1/26/07	Home Depot®
High-Hop	hook	$2.36	1/26/07	Home Depot®
High-Hop	Fasteners	$11.92	1/26/07	Home Depot®
High-Hop	rope	$1.44	1/23/07	Home Depot®
High-Hop	U-bolt	$4.78	1/23/07	Home Depot®
High-Hop	Fasteners	$14.61	1/23/07	Home Depot®
High-Hop	Batteries	$2.99	1/23/07	UCSD Bookstore
High-Hop	wire ties	$3.15	1/29/07	Home Depot®
High-Hop	Pulley	$4.42	1/29/07	Home Depot®
High-Hop	twine	$5.19	1/29/07	Home Depot®
High-Hop	Pullies	$6.41	1/16/07	Home Depot®
High-Hop	wire rope	$3.47	1/16/07	Home Depot®
High-Hop	end swedges	$1.54	1/16/07	Home Depot®
High-Hop	swedges	$2.46	1/16/07	Home Depot®
High-Hop	flash drive	$19.99	2/8/07	UCSD Bookstore
High-Hop	DC-gear motor	$44.57	1/19/07	Mcmaster
High-Hop	Dc-Gear motors	$69.48	1/11/07	Robotics Market Place
High-Hop	Ratchests & Sprokets	$76.95	3/6/07	WM Berg
High-Hop	Nylon Hollow Rod	$14.25	3/5/07	Mcmaster
High-Hop	PolyCarbonate sheet	$21.34	3/5/07	Mcmaster
High-Hop	fasteners	$29.62	3/5/07	Mcmaster
High-Hop	Bearings	$32.31	3/10/07	Sports Chalet
High-Hop	Rope	$6.95	6/26/07	REI
High-Hop	Webbing	$9.70	6/26/07	REI
High-Hop	Ratchet	$46.72	3/29/07	Mcmaster
High-Hop	Sprokets	$46.64	4/18/07	Mcmaster
High-Hop	Solenoid	$17.94	4/19/07	Mcmaster
High-Hop	Aluminum Tube	$8.80	4/23/07	Home Depot®
High-Hop	spring	$4.48	3/28/07	Mcmaster
High-Hop	solenoid	$24.11	3/28/07	Mcmaster
High-Hop	Roller Chain	$24.46	4/24/07	Mcmaster
High-Hop	Sprokets	$6.23	4/24/07	Mcmaster
High-Hop	fasteners	$16.95	4/24/07	Mcmaster
High-Hop	Aluminum	$10.09	4/24/07	Mcmaster
High-Hop	Spacers	$3.96	4/24/07	Mcmaster
High-Hop	DC-Motor (used)	$464.00	5/18/07	MicroMo

High-Hop	Latex Tubing	$9.54	7/3/07	Ocean Enterprises Inc
High-Hop	O-rings	$2.64	7/23/07	Ace Hardware®
High-Hop	Material	$20.15	7/10/07	CRMS
High-Hop	Sprokets	$27.89	7/5/07	Stock Drive Products
High-Hop	roller Chain Connectors	$6.60	7/17/07	Mcmaster
High-Hop	Fasteners	$10.55	7/17/07	Mcmaster
High-Hop	WaterJet Part	$150.00	9/14/07	CAL Waterjet
High-Hop	Nylon	$9.33	4/16/07	CRMS
High-Hop	Machine Shop	$200.00	11/2/07	Scrippts
High-Hop	Machine Shop	$173.64	1/1/07	CRMS
High-Hop	Motor	$37.19	10/3/07	Robotics Market Place
Lab	vise	$59.97	1/23/07	Home Depot®
Lab	Clock	$14.23	1/23/07	UCSD Bookstore
Lab	mandrel	$9.00	3/5/07	Home Depot®
Lab	cutoff wheels	$22.23	3/5/07	Home Depot®
Lab	Cds	$12.31	2/8/07	UCSD Bookstore
Lab	Mmandrel	$3.22	3/10/07	Sears®
Lab	Tap Set	$24.96	4/3/07	Home Depot®
Lab	counter sink	$7.94	4/3/07	Home Depot®
Lab	drills	$7.14	4/3/07	Home Depot®
Lab	Optical Mouse	$21.54	4/9/07	UCSD Bookstore
pendulum	Track	$846.00	10/25/06	ROLLON
pendulum	Cap screws	$9.40	11/27/06	Mcmaster
pendulum	Fasteners	$8.17	11/28/06	Home Depot®
pendulum	Wallmount	$10.14	11/28/06	Home Depot®
pendulum	Fasteners	$1.06	11/28/06	Home Depot®
pendulum	Drywall mounting screws & brackets	$30.04	11/28/06	Dixieline lumber
pendulum	threaded rod	$4.27	11/29/06	Marshalls Hardware

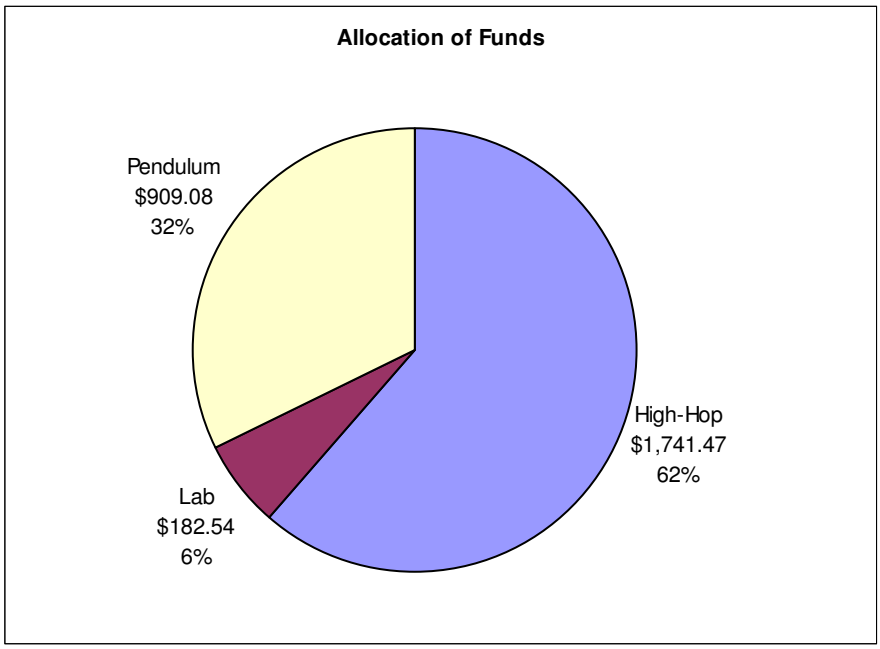

Figure 40: Pie Chart for allocation of Funds

Appendix D

Ultrasonic range finder code (Simulink)

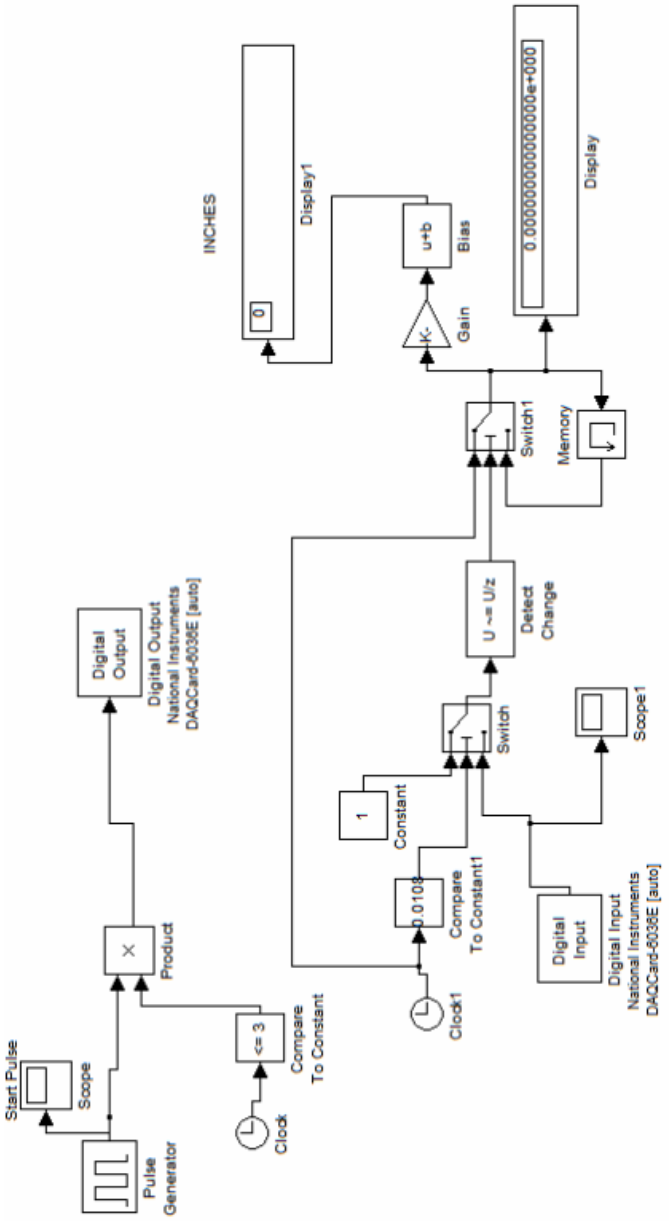

Figure 41: Simulink Picture

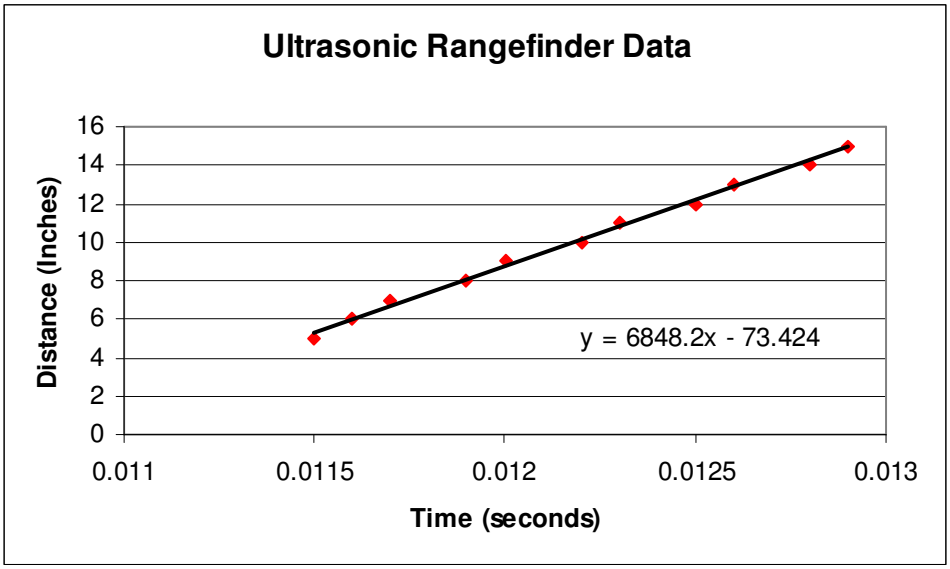

Figure 42: Finding Bias for Ultrasonic Rangefinder

Table 3: Ultrasonic Range Finder Data

Seconds	Inches
0.0115	5
0.0116	6
0.0117	7
0.0119	8
0.012	9
0.0122	10
0.0123	11
0.0125	12
0.0126	13
0.0128	14
0.0129	15

Appendix E

Energy Calculations on Microsoft Excel

Table 4: Energy Calculations

weight (kg)		tot def (m)	start	
44.1		0.285	26.5	
68.992		0.308	26.5	
113.092		0.31	26.5	
0		0.265		
Graph Slope	2084.6			
Graph Offset	-552.14			
Force (kg)	Tot deflection (m)	deflection (cm)	Spring Energy(J)	Velocity of release (m/s)
264.6	0.391796987	12.67969874	86.80261826	5.670019213
		Max Height (m)	Max Height (feet)	
Weight(lbs)				
60		1.640261116	5.38005646	
Mass of Robot (kg)				
5.4		F.O.S.(2) Height		
		0.820130558	2.69002823	

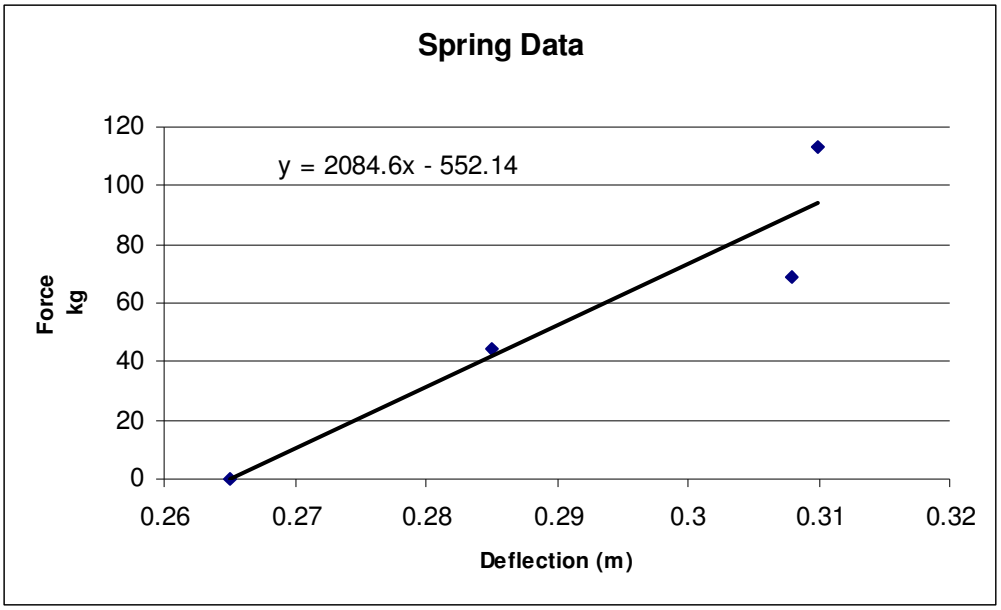

Figure 43: Energy Data

References

Bewley, Tom. Numerical Renaissance. (Not Yet Published).

Bryson, Arthor, Ho, Yu-Chi. Applied Optimal Control: Optimization, Estimation, and Control. Taylor & Francis, 1975.

Anderson, Brian, Moore, John. Optimal Filtering. Dover Publications, 2005.

Schmidt-Wetekam, Chris, Bewler, Tom, Hughes, Robert. Design, Optimization, and Control of a New Class of Reconfigurable Hopping Rovers. CDC, 2007