MECE E3430 – Engineering Senior Design and Concept
Professor F. Stolfi, Professor D. Vallancourt
Spring 2016

The Jaxon 5 – Final Report

A study in legged robotics through a one-legged hopping robot

Team 14 – The Jaxon 5

Daniel Garcia
Rienzi Gokea
Steve Jaycox
Diego Song Cho
Andrew Vogel

Columbia University in the City of New York
The Fu Foundation School of Engineering and Applied Science
Department of Mechanical Engineering & Department of Electrical Engineering

May 5, 2016

Executive Summary

This project approached the design of a monoped hopping robot. This jumping robot, *Jax*, operated by actuating a pneumatic cylinder timed to resonance with its natural landing period to perform consistent consecutive hops. The purpose of research into monoped hopping designs is to branch into legged locomotion. After achieving untether three-dimensional hopping, it would next be possible to approach two legged running or four legged running.

Jax's design is a double-acting pneumatic cylinder that uses a solenoid valve to actuate the leg by evacuating the lower chamber. Its design stemmed in part from the research of Marc Raibert at the MIT Leg Lab where he researched legged locomotion before founding Boston Dynamics. *Jax* was controlled via a pacemaker type control system that actuated its leg with a delay time relative to sensing landing. It could also be controlled on a two-dimensional tether with a servo to control body angle.

From this project, we gathered a good deal of information on legged locomotion and have placed ourselves in a good place to move forward and achieve two-dimensional free motion with additional time to fine tune the control system. In general, the weaknesses and major influences stemmed from poor time management in the beginning of the semester when too much time was spent trying to perfect the one-dimensional motion which we later found to be impossible.

Table of Contents

Introduction

The field of legged robotics has had a sparked increase in recent years, thanks to blockbuster science fiction movies and technological advancements in robotics and sensing. As the thought of robotics becomes a more present part of our daily lives, there is a need that the robots are capable of navigating more than simply flat-and-wide courses – to evolve beyond the limits of the wheel. The importance of this work can be seen in numerous physical applications, from assistive robotics to adaptive military transport, and even natural disaster rescue operations.

At the moment robots are primarily wheeled systems, restricted to routines along smooth, level, horizontal surfaces. Though there are improvements to wheels and systems to allow them to access rougher terrain, these systems, too, are still incapable of the feats of exploration capable in legged systems found throughout nature. It is with this mindset that scientists and companies have investigated what is an otherwise untapped field in locomotive robotics.

Our approach in studying these legged systems is very similar to the approach of the MIT Leg Lab[1]. In order to understand legged movement, it is important to understand the simplest type of legged movement, that of a one-legged hopping robot. By understanding the motion and control systems that govern this system, the development of two-legged and four-legged systems becomes far more clearer and logical.

This design project began by studying previously attempted designs and understanding their primary differences and strengths, as well as making key design decisions such as the number of legs, types of stabilization, and methods of actuation. An intense literature and patent search was conducted out of this to aid in our understanding of these topics, what successes and

[1] Official MIT Leg Lab Website. (http://www.ai.mit.edu/projects/leglab/background/background.html)

failures have come out of past designs, and what challenges to expect moving forward. Though initially an attempt was made at a statically and dynamically-stable two-legged hopper, using primarily brushless DC motors as actuators, as well as considerations of a one-legged mechanical spring and a one-legged gas spring design, our final design iteration resulted in a one-legged, dynamically-stable, pneumatically-actuated hopping robot.

Our levels of success from here were clear. The first level of success was to achieve hopping in one-dimension. This essentially comes down to creating a working leg design that is capable of achieving an appropriate vertical leap. It was also necessary to develop a control scheme for determining when the leg extends and contracts during each hop, and at what (resonant) frequency the leg would hop the highest. For this level of success, a test platform was designed using a stand and parallel bars with linear bearings, to constrain all hopping of the robot to a vertical motion normal to the floor.

The second level of success was to achieve hopping in two-dimensions. This required not only that the robot be capable of hopping at its resonant frequency, but also that a control scheme for balancing be implemented. This control scheme needs to read, in real time, both the angle of the leg and the body, and adjust both when the robot loses balance in the horizontal direction. For this level of success, a test boom was designed such that it travels along a fixed hemisphere on a surface, restricting motion to polar and azimuthal motions which were approximated as vertical and horizontal.

The third level of success was to achieve hopping in three-dimensions. Untethering the robot from any constraints, the robot would be free to move in three dimensions, assuming any wired electronics or peripherals do not constrain the ability to hop. The mechanism used to move

the leg is then required to move the leg in a two-dimensional plane, and the IMUs must take into account both pitch and roll angles. This would be a similar control scheme to the two-dimensional hopping, however two separate angles would be controlled simultaneously to balance the robot. The fourth level of success was to achieve various feats of locomotion in either 2D or 3D, from path following, to object detection, to even somersaults.

Background Information

Literature Search

Our initial literature search led us to a number of meaningful resources, most significantly including the pros and cons of previously conceptualized and tested designs. Each team member was tasked with finding and reading at least five sources, from textbooks to patents, and rank their top 3 choices for presentation to the rest of the group. This systematic approach led us to extensive and invaluable resources.

The most crucial of these resources is the textbook *Legged Robots That Balance* by Marc Raibert[1], founder of the MIT Leg Lab and later Boston Dynamics™. This book details extensively their team's findings as they progressively studied locomotive robots, from two-dimensional monopedal robots, to three-dimensional monopedal robots, to bipedal and quadrupedal robots, and finally to the work and study of movement such as running, flipping, and stair climbing. Another crucial find was a textbook called *Robotics (Modeling, Planning, and Control)*[2], which provides a clear introduction to the world of robotics, including kinematics, dynamics, controls, and environment sensing. The extensive bibliographies of both of these books have been crucial to our further research into hopping robots.

Patent Search

Our literature search also revealed a number of interesting patents amongst the different design approaches carried out by various engineers. One of these included a leg design aimed at improving inefficiencies found in the MIT Leg Lab designs, moving away from their use of pneumatic cylinders as leg actuators and instead towards a springed four-bar mechanism leg with two-way symmetry, cutting the necessary rotational torque needed in half.[3]

Designs Reviewed

- The Penn Jerboa: A Platform for Exploring Parallel Composition of Templates[4]
    - A design loosely based on the jerboa animal, this bipedal hopping robot was the inspiration for one of our earliest designs. The robot utilized a tail in order to act as counterbalance and correct orientation in the air. It was a design that we pursued in detail, but was ultimately abandoned because of its complexity.
- Minimalist Jumping Robots for Celestial Exploration[5]
    - This design incorporated a spring-loaded four-bar mechanism that was instrumental in our second design for *Jax*. The robot itself, though, was not dynamically stable and was intended to aim, release the spring, jump, and land in any orientation at which point it would have a mechanism to right itself. We were not a fan of the concept that it was intended to land falling over and had to right itself after every jump, but were intrigued about the four-bar spring possibility.
- Trajectory Planning of a One-Legged Robot Performing a Stable Hop[6]
    - This interesting design was dubbed the “Pixar Lamp” in our studies of it because of the way it achieved hopping. It would bend down, and force its weight up, jumping with all its weight similar to the lamp in the opening scenes of any pixar movie. While an interesting concept, it was still only statically stable and not what we decided to pursue.
- Experiments in Balance With a 2D One-Legged Hopping Machine[7]
    - In the end, we went back to the basics and found the work of Marc Raibert’s earliest work. His original one-legged hopping robots were referenced by almost

every paper we consulted. We traced his work back to his beginnings where he used pneumatic cylinders to actuate dynamically-stable hoppers in 2D and then 3D. It was these designs that inspired *Jax*'s final iteration.

Design Overview

Design Details

1. Leg Design

One Dimensional Design

Both the one-dimensional and two-dimensional legs are based on the same pneumatic piston-cylinder assembly. The specifics of the piston-cylinder are discussed in greater detail further on, but to summarize, the top half of the cylinder is kept at a closed pressure thus making it act as a spring while the bottom half of the cylinder is alternated between a high pressure to compress the cylinder and atmospheric pressure to extend the cylinder.

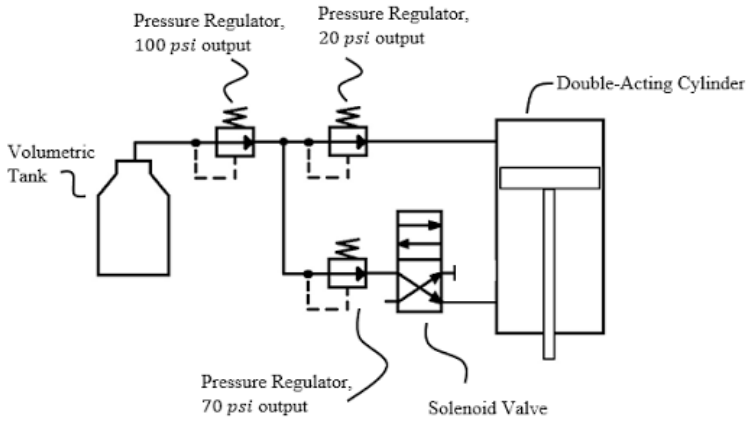

Figure 1*: Pneumatic circuit design. Brings compressed air of set volume through a system of regulators to power pneumatic actuator for repeating jumping*

In the one-dimensional design, the design itself is relatively bare. Because the movement is constrained to simple up and down linear motion, there is no need to reinforce the cylinder as done in the 2D *Jax* design. The design itself entails a 3D printed mounting clamp attached to the cylinder as well as a 3D printed foot. The foot itself is made of three 3D printed pieces along with a rubber polyurethane “toe”. The toe was made in a 3D printed mold by mixing two

chemicals together and letting them sit for a few hours as they cured into a solid, rubber structure. Two 3D printed pieces then encapsulated the top of the toe to allow it slight movement and allow for the toe to possibly activate a button in order to indicate when the foot was on the ground (this ended up being an unnecessary feature). Finally a 3D printed coupler piece holds the other two pieces together and connects to the bottom of the piston itself.

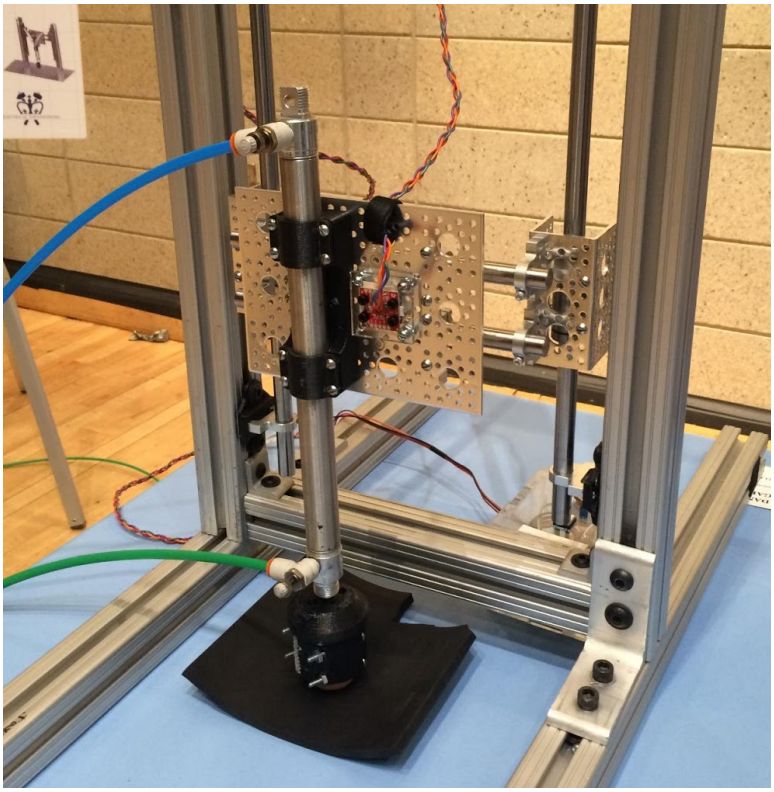

Figure 2: *The one-dimensional Jax design*

Two Dimensional Design

The two-dimensional design is slightly more involved than the one-dimensional, but the underlying principle is the same. The largest change to the pneumatic cylinder in the 2D design is an slightly extensive support structure. The cylinder is flanked by two case-hardened 8mm-diameter rods. The rods act as followers for two linear bearings which in turn are attached to ¼-inch rods which are screwed into a 3D printed "ankle." The ankle is connected to the same

two 3D-printed parts and polyurethane toe that are used for the foot in the one-dimensional design. Specially purchased sized bore clamps were purchased in order to hold the rods and bearings together.

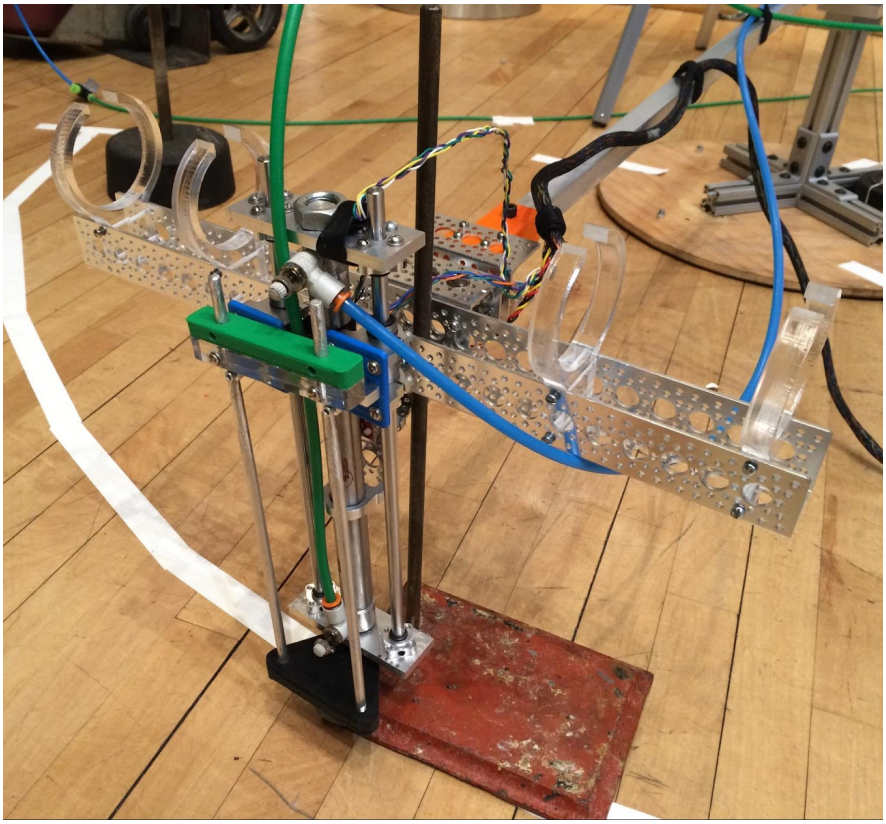

Figure 3: *Jax 2D Design*

The second aspect of the 2D design is an inertial body and servo which is used to rotate the pneumatic cylinder in order to keep *Jax* balanced. Ideally, the 2D *Jax* design was meant to rotate freely around a supporting boom that constrained it to the 2D plane. The body itself would rotate and then control a servo which would control the leg angle. One key assumption made was that the body was supposed to be much heavier than the leg itself. This was made so that the body's inertia would allow the leg to rotate in the air without changing the relative position of the body in the air. Thus a long-shaped body with weights on the ends was necessary. One original

consideration was to keep *Jax* untethered and run air off compressed air cylinders directly on the body. Unfortunately, the compressed air cylinders were too small to provide any sort of sustained jumping, and ended up acting only as the needed inertial counterweights.

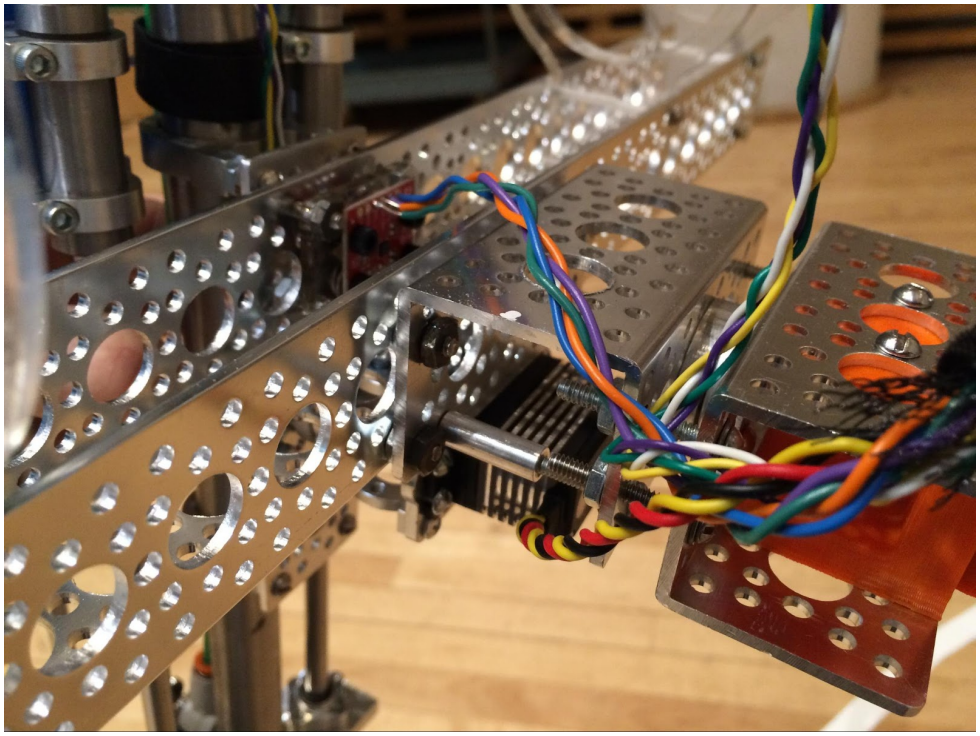

Figure 4: *Close-up of Servo and IMUs in Jax 2D Body*

The servo itself is attached to the bottom of the body and supported with an *Actobotics* servo block. This provided the necessary support and rotation of the servo that allowed easy attachment of the piston to the body. Two IMU accelerometers are also placed on *Jax* in two key positions. One IMU is placed in the center of the body in order to get data on the rise and fall of the entire robot as well as the body's roll angle relative to the ground. The other IMU is placed directly behind the pneumatic cylinder and reads the angle and acceleration of the leg relative to the body.

2. Test Platforms and Support Structures

One-Dimensional Track and Test Platform

A One-dimensional track was designed primarily using 80-20 aluminum extrusions and hardware found throughout the mechanical mechanical engineering lab. The track was originally used as a test platform for running IMU data and learning how the pneumatic cylinder works, but once the design was finalized and the 2D boom was built, the 1D test platform became used as tracks primarily for the standalone 1D version of *Jax*.

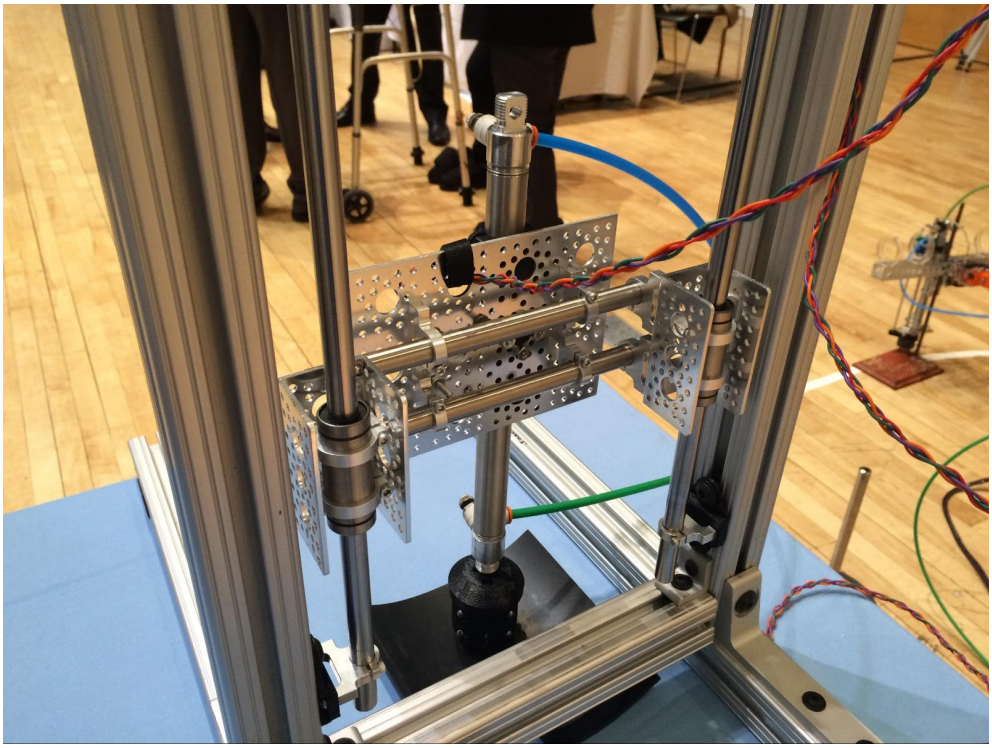

Figure 5: *The one-Dimensional Test Platform and Frame*

The platform is effectively a rectangular frame of 80-20 extrusion stood up vertically. Along the inside length, case-hardened 12-mm diameter precision rods are secured for linear bearings to follow. The linear bearings are attached together with more rods to a center plate where the 1D *Jax* leg connects.

2D Boom

For the 2D design, the support structure is relatively simpler. The structure entails a simple square-extruded-aluminum boom on a vertical 80-20 extrusion about the same height of the robot. The boom is connected to the extrusion on a rotational bearing allowing the boom to rotate freely on the axis of the extrusion. The boom itself is also connected about 2 feet from the end in order to provide a counterweight system to mitigate the effects of gravity on the boom and robot thus lengthening the time between jumps, easing the burden of computation and reducing noise in measurements. The far end of the boom has a connection piece and a swivel hub where the 2D body attaches and is allowed to rotate freely.

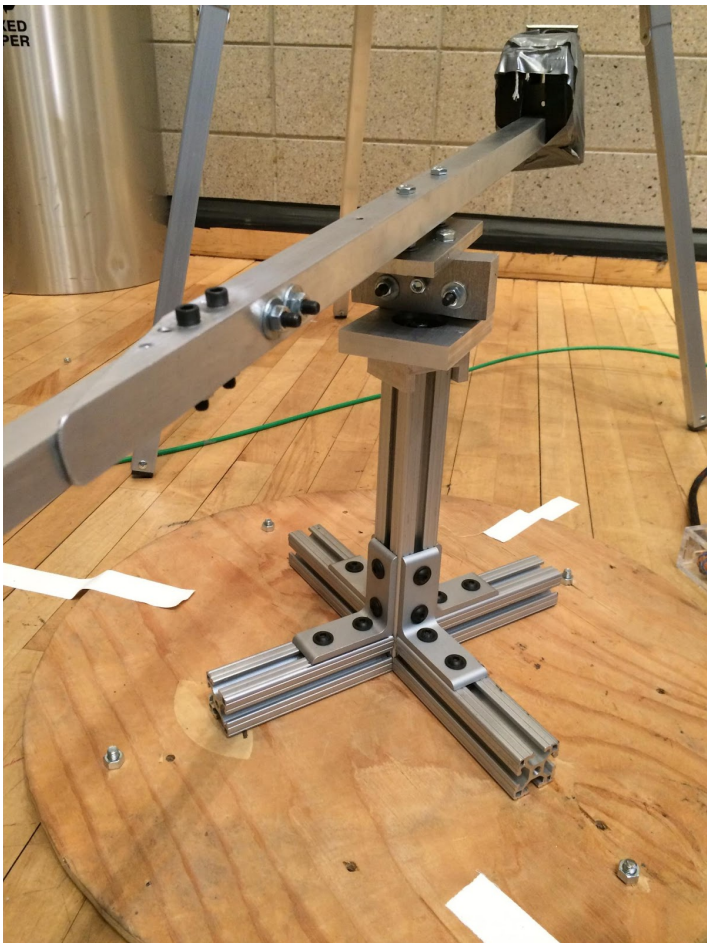

Figure 6: *2D Support Boom*

3. Control System

One-Dimensional Controls

For one-dimensional jumping, it was set out to automatically determine resonant frequency with accelerometer data. After much trial and much more error, it was determined that this would be impossible. Jumping at a frequency that is not resonant provides inconsistent data. Any two jumps could increase or decrease in jump height and period almost randomly. Additionally, approaching the resonant frequency does not guarantee any increase in jump height or regularity in jump height and period. The only method that was successful in automatically determining resonant frequency was to take 20 successive jumps, process the accelerometer data with a Fourier Transform, and average the largest peak frequencies to determine the next frequency to jump at. This next thrust frequency would be processed the same way and the recursive function would eventually converge.

Even still, many problems occurred with this method. It still requires some sort of input as to when to trigger the thrust. Using accelerometer data, there was not enough resolution on such a small timescale to anticipate the peak acceleration, so it was necessary to manually input a delay time: a value in milliseconds that would time the thrust to be a certain time after touchdown.

Because this crutch was necessary anyway, it was decided to use this to manually target the resonant frequency within a shorter amount of time. To do this, the delay time was mapped to a potentiometer. By varying the delay time, the jumping frequency could be indirectly varied. Additionally, the jumping frequency -- calculated by performing a Fourier Transform on the real time accelerometer data -- was shown in real-time and it was easy to determine when the

resonant frequency was matched because the Fourier Transform showed the first two harmonics and additionally *Jax* would shake the table from the increased force.

Two-Dimensional Controls

In two dimensions, the one-dimensional controls still apply; however, additionally balance must be accounted for. There are two phases of jumping in a controls point of view: flight and stance. During the flight phase, the foot must be positioned such that the thrust of the next jump will provide *Jax* with appropriate net velocity. Generally for these cases, the goal is to neutralize any velocity incurred. This is done by angling the foot opposite of the direction of travel. This is simply done by taking the current foot angle in the air and placing it at a scaled angle on the other side. This would hypothetically eventually converge the leg angle to zero, but due to uncertainties and imperfections in the leg angle, floor angle, etc, will not truly converge.

The other phase of controls is the stance phase. During the stance phase, it is the goal of the control system to neutralize any rotational velocity currently on the system. It would do this by rotating the body angle in a way equal and opposite to the torque already on the system causing tipping. This could be done using gyroscopic data to neutralize rotational inertia or by using differentials on gyroscopic data to determine angular acceleration and then values for torque.

4. Costs

Mechanical Costs

In our design we were able to repurpose a lot of material that was found in the Mechanical Engineering Lab. A lot of test and support structures were made through 80-20 hardware as well as free 3D printing in the Columbia Makerspace. Some of the more specific

materials could not be found at Columbia and were bought. The following table delineates the costs of materials bought:

Product Name	Vendor	Cost	Quantity	Total Cost
HS-7950T8 Servo Motor	Servo City	$149.99	1	$149.99
Standard HiTec Servo Block	Servo City	$26.99	1	$26.99
0.84 Inch Bore Clamp	Servo City	$5.99	2	$11.98
.25 Inch Bore Clamp	Servo City	$5.99	2	$11.98
15mm Bore Clamp	Servo City	$5.99	4	$23.96
8mm Set Screw Hub	Servo City	$4.99	2	$9.98
3 Inch Aluminum Bracket	Servo City	$3.99	2	$7.98
Flat Triple Bracket	Servo City	$2.49	1	$2.49
18 Inch Aluminum Bracket	Servo City	$13.99	1	$13.99
Swivel Hub	Servo City	$6.99	1	$6.99
8mm ID - 45mm L Linear Ball Bearing	Servo City	$3.49	2	$6.98
8mm D 330mm L Precision Shaft	VBX Bearing	$10.37	2	$20.74
Smooth-On Polyurethane	Lee's Art Shop	$2	4oz	$2
				$296.05

Table 1: List of Mechanical Components and their costs in the 2D Body

It should be noted that the primary vendor used was Servo City. The reason for this is twofold. Firstly, Servo City offers a very powerful framework for robotics called *Actobotics*. This framework allows for pieces fitting together very well and allows for quick assembly and disassembly. We decided early on on using this framework for our ease. Secondly, we were sponsored by Servo City, and they offered us a 15% discount on all products bought from them. This discount was not taken into account in the above table, but is described in the appendix.

Electrical Costs

Several components were either reclaimed from previous lab courses (Arduino, buttons, potentiometer, etc) or were provided by the Electrical Engineering department (power supply, banana plug cables, etc). In addition to this, a budget was provided by the Electrical Engineering department. The following table delineates the costs bought from those funds:

Product Name	Vendor	Cost	Quantity	Total Cost
6 Degrees of Freedom IMU Breakout	SparkFun	$19.95	2	$39.90
Proto Shield for Arduino Kit - Stackable Version R3	Adafruit	$9.95	1	$9.95
Premium Female/Male Jumper Wires (40) 12"	Adafruit	$7.95	1	$7.95
Servo Extension Cable 12"	Adafruit	$1.95	2	$3.90
				$61.70

Table 2: List of Electrical Components and their costs

Analysis

1. **Pneumatics**

Jax operates by alternating the pressure in the lower chamber of a double-acting pneumatic cylinder between a high pressure (at 70 psi) and atmospheric pressure. This is pushed against by a low pressure in the upper chamber (at 20 psi).

The pneumatic circuit draws its compressed air from a volumetric tank source. This is first passed through a regulator which is set at 100 psi. That initial system is referred to as the source. In testing, a constant-pressure system with a set output of 100 psi was used, but in demonstration, a pressurized tank regulated to 100 psi output was used.

The output of the source then enters the true pneumatic circuit which consists of two pressure regulators, a solenoid valve, and a double-acting pneumatic cylinder. First, the

compressed air source is split into two pathways, where each pathway passes through a pressure regulator set to a constant pressure. The upper split then is stepped down to 20 psi and passes through a check valve to allow the pressure in the upper chamber to rise as it compresses. It also prevents backflow to the compressor, possibly damaging the regulator. The bottom split is stepped down to 70 psi and output to a solenoid valve. When the solenoid valve is actuated, the inlet air goes to the lower chamber of the pneumatic cylinder, which is the second output. When the solenoid valve is in the unactuated position, the cylinder connects to an open exhaust and the inlet connects to a stopper.

The pneumatics for each jump work in a cycle: thrust, flight, and landing. Thrust starts when the body reaches the lowest point and highest acceleration of its ground phase. At this point, the solenoid valve is toggled from the actuated to the unactuated state, evacuating the chamber and thrusting the leg downward while the body moves upward. This adds an additional force with each jump, in order to make up for energy losses.

Once it leaves the ground, it leaves the thrust phase and begins the flight phase. This section of the jump cycle is characterized by an acceleration of 1 g. At the start of this phase, the solenoid valve is again actuated. This retracts the leg, allowing for a slightly higher overall jump height.

The final phase of each jump is the landing phase. This phase begins at touchdown and ends with the start of the thrust phase. There is no characteristic response by the control system during this phase. During landing, the piston compresses to create a pressure differential between the chambers that is reused in the thrust phase and combined with the added energy of the evacuation to jump.

To calculate the necessary pressures and the hopping motion, the work-energy theorem can be used. By using pressure work, initial equations can be written. Since thrust and landing are characterized by two different motions, the equations are written separately. The equation for thrust is:

$$\tfrac{\pi}{4}D^2\left(\int_{x_0}^{S} P_u dx - \lim_{x\to 0}\int_{S-x_0} P_l dx\right) = \left(\tfrac{1}{2}mV^2\right) + (mgS - mgx_0),$$

for $x = [x_0, 0)$, and the net-work done by the system during thrust is equated with the increase in potential and kinetic energy. Additionally, by equating the total kinetic and potential energy to the peak potential energy, mgH_j, an updated equation can be written that simplifies the right-hand side.

$$\tfrac{\pi}{4}D^2\left(\int_{x_0}^{S} P_u(x)dx - \lim_{x\to 0}\int_{S-x_0} P_l(x)dx\right) = mgH_j\,.$$

Now, by making the assumption that P_l remains constant during the thrust phase (a reasonable assumption due to the near-instantaneous evacuations of the lower chamber), force, balance, and ideal gas law equations can be manipulated and generalized for x as $P_u(x) = P_{u,0}\frac{S}{x}$ and the above equation can be rewritten. In this process, another necessary, but not great, assumption is made where no potential energy is initially stored. This assumption allows for the calculation of $x_0 = \frac{P_{u,0}}{P_l}S$ (the initial location of the piston), and inherently applies that at time $t = 0$, the upper and lower pressures are equalized.

$$\tfrac{\pi}{4}D^2\left(\int_{S\,P_{u,0}/P_l}^{S}\left(P_{u,0}\tfrac{S}{x}\right)dx - P_l\lim_{x\to 0}\int_{S\left(1-P_{u,0}/P_l\right)} dx\right) = \tfrac{\pi}{4}D^2\left\{P_{u,0}S\ln\left(\tfrac{P_l}{P_{u,0}}\right) + P_lS\left(1-\tfrac{P_{u,0}}{P_l}\right)\right\} = mgH_j$$

This creates a simple closed form equation that provides a decent approximation for the jump height as a function of entirely known variables: inner bore diameter, lower chamber pressure, stroke length, body mass, and the pressure ratio. All of these values are intrinsic to the system and not independent except for the lower chamber pressure and the pressure ratio. Thus, this equation can be plotted for jump height versus pressure ratio for various lower pressures.

$$H_j = \frac{\frac{\pi}{4}D^2 P_l}{mg} S \left\{ \frac{P_{u,0}}{P_l} \ln\left(\frac{P_l}{P_{u,0}}\right) + \left(1 - \frac{P_{u,0}}{P_l}\right)\right\}$$

The function above provides us with a simple, linear, closed-form function that shows for any pressure ratio, the maximum jump height occurs at $\frac{P_l}{P_{u,0}} = e^2 = 7.389$. *Jax* operates at a pressure ratio of about 3.5, which is 95% of the maximum jump height. While it could be optimized to move more efficiently, the regulators in the system do not allow much resolution at pressures approaching 100 and 0 ps, so the values were instead kept respectively at 20 and 70 psi. Additionally, due to the assumptions made, it can be assumed that not only would the jump height be lower at any pressure ratio, the peak would also likely shift to the left. This is because we are neglecting the potential energy already in the system at the start of the thrust phase. This potential energy would take some of the load off of the pressure ratio and allow a smaller pressure differential to achieve the same jump height, causing the curve to shift to the left. This lower pressure differential causes less work to be available, and achieves a lower overall height.

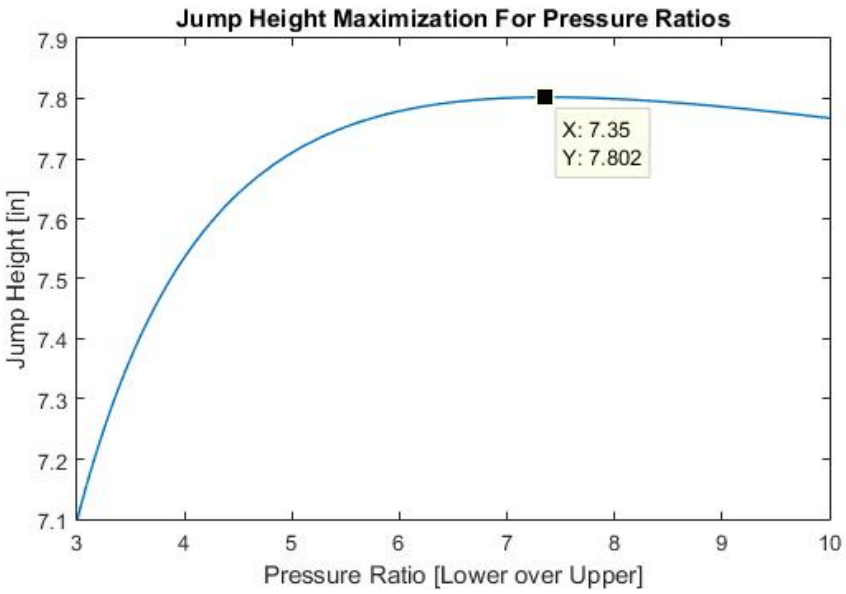

***Figure** 7: Jump height curve for mass of 1 lb, inner bore diameter of 0.75", lower chamber pressure of 70 psi, and stroke length of 8". The jump height is relative to the location of the center of mass in its fully compressed state.*

2. Electronics

Overview

For controls and data acquisition an Arduino was used, an open-source microcontroller development platform that is inexpensive, easy to use, and allows input-output system prototyping. Relevant to this work, Arduino is capable of analog input, digital input and output, and independent operation via programming in the C/C++ based Arduino programming language. Several electronic components were attached to the device as either inputs or outputs on the system.

For the one-dimensional setup, the inputs included a momentary push button, a toggle switch, a potentiometer, and an IMU (inertial measurement unit). The toggle switch controlled whether the device operated in a manual hopping mode, or an automated hopping mode. The sole output was a binary signal that controlled the solenoid valve that governed the extension and retraction of the pneumatic actuator. In manual hopping mode, the solenoid valve was controlled by the momentary push button, which allowed a user to control each jump in a discrete manner. When in automatic hopping mode, the solenoid valve opened and closed in a cyclic manner, where the time delay between jumps was controlled by a potentiometer.

The two-dimensional setup contained inputs including two potentiometers, two IMUs, a momentary push button, and a toggle switch. The outputs consisted of a signal that controlled the solenoid valve, as well as a PWM signal that controlled the angle of the device's leg through a servo motor. Similar to the one-dimensional setup, the toggle switch controlled the mode in which the device would function, either manual hopping or automated hopping. Both modes

were very similar to their one-dimensional counterparts, with the main difference being an extra potentiometer that controlled the servo angle.

IMU

The IMU used (LSM6DS3) contains an accelerometer, which measures *proper* acceleration in three dimensions (i.e. acceleration relative to free-fall, units of *g*), as well as a gyroscope, which measures the rate of rotation about those three-dimensional set of axes, for a total of six degrees of freedom. The device communicates with the Arduino through I^2C (Inter-Integrated Circuit), an asynchronous circuit communication system that supports multiple master devices to communicate with multiple slave devices. In the one-dimensional setup, a single IMU was connected to the Arduino via four wires (see **Figure 8**): power (3.3 V), ground, SDA (I^2C data line), and SCL (I^2C clock line). The Arduino was then able to communicate with the IMU through the use of open-source Arduino software libraries. We used a MATLAB-coded interface to acquire vertical acceleration data from the IMU to explore the frequency-domain components of the locomotive mechanism (see **Figure 9**).

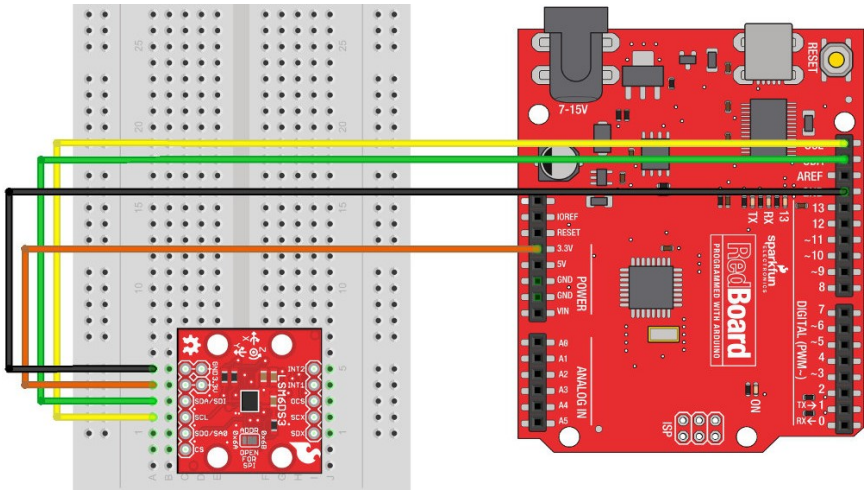

***Figure 8**: I^2C wiring for operation of the LSM6DS3 6-degree of freedom inertial measurement unit.*

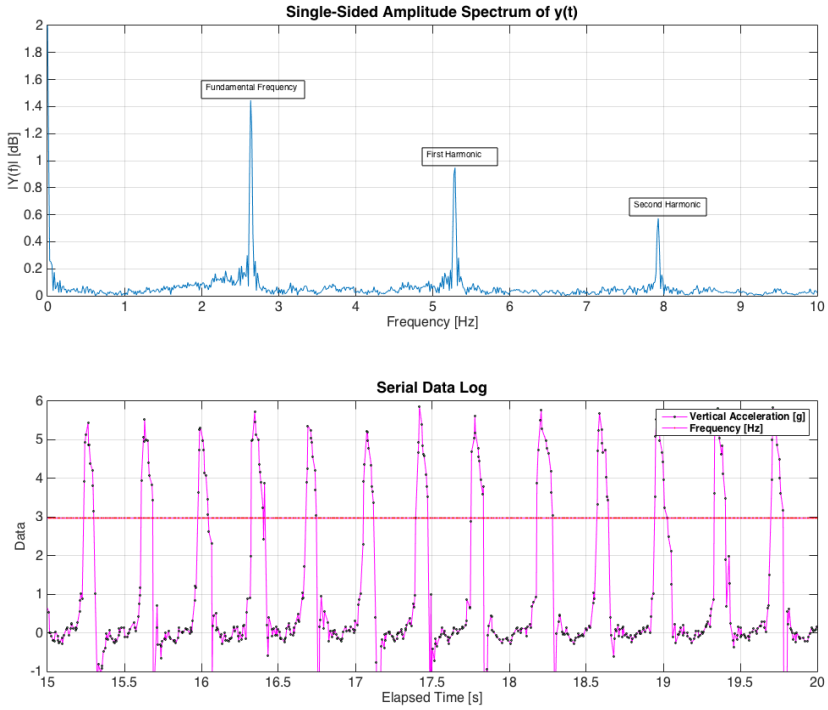

Figure 9: *Real-time acquisition using IMU vertical acceleration data from a test in automated hopping mode with hopping frequency set around 2.7 Hz.*
Upper subplot: Displays the amplitude (magnitude) spectrum of the acquired data. The most prominent peak depicts the fundamental frequency (near-resonance) of the hopping. Peaks corresponding to higher-order harmonics accumulated over time.
Lower subplot: Real-time plot of vertical acceleration and hopping frequency.

Solenoid Valve

Solenoid valves are electromechanically operated and are highly reliable in providing fast switching between ports. The actuation of most solenoid valves requires an input voltage between 6 and 12V, which is larger than the voltage that the Arduino microcontroller can provide via its regulated 5V supply. To circumvent this problem, power was supplied by a 9V, 1A DC power adapter connected to the Arduino microcontroller's built-in voltage regulator (V_{in}). Because the solenoid valve is an inductor, a diode was used to eliminate transient voltages that could surge when the magnetic coils of the valve lost power. These transient surges could potentially damage the surrounding electronics without protection from the diode. In order to

provide the required current draw of the solenoid valve, a TIP101 NPN Darlington pair (transistor) was used to amplify the available power drawn from the wall plug connected to the Arduino's voltage regulator. Like in most transistor circuits, a current-limiting 1kΩ resistor was placed at the base of the transistor to also prevent overloading the control line (base) of the transistor. **Figure 10** below depicts a circuit schematic of the switching circuit used to control the solenoid valve.

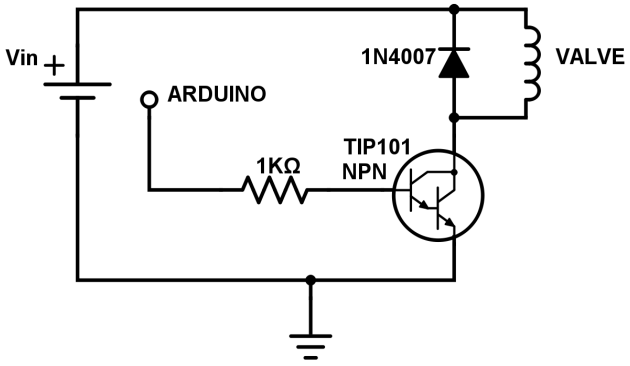

Figure 10*: Transistor-based circuit required to operate solenoid valve using the Arduino microcontroller. The port named "Arduino" represents any digital output pin that may be used as a control line.*

Servo Motor

A servo motor was used to control the leg and body of the robot in the two-dimensional setup. Both the top of the actuated leg and the center-of-mass of the body were placed about the same axis of rotation. This would allow a single servo to actuate either and both the leg and the body. Under the assumption that friction at the point of contact between the leg and the floor surface would be sufficient to prevent the leg from rotating upon actuation the servo, it would be possible to use a single motor to independently rotate the leg (flight phase) and the body (stance phase). With respect to the rest of the electrical components involved in this design, the servo motor required the highest electrical power input, so an external DC power supply (KeySight

Technologies U8032A) was used to enable the motor at its highest possible performance. Like for all servo control, the angle (of mechanical rotation) was determined by the width of an electrical pulse (PWM signal) that was applied to the control wire. To ensure proper waveform of the pulse-width modulated signal controlling the servo, the ground of the external power supply was connected to the ground of the microcontroller.

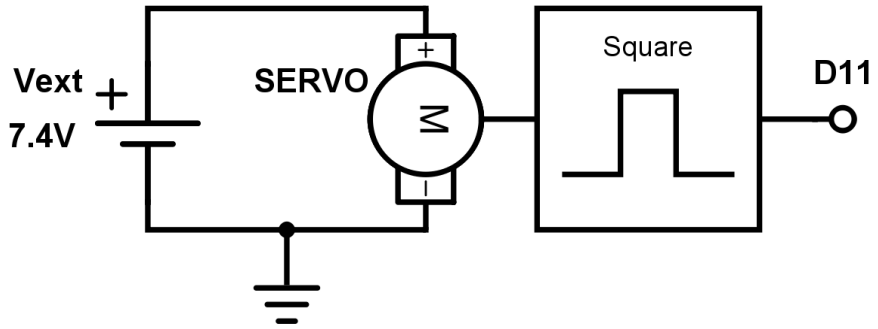

Figure 11: *Schematic of servo motor circuit. V_{ext} represents voltage from an external DC power supply set at 7.4V. An external supply was utilized to provide sufficient power input. The motor was controlled by a pulse-width modulated (PWM) signal output from the Arduino's digital pin, represented by the D11 port. The ground of the external power supply was connected to that of the microcontroller to ensure proper PWM output.*

Overall Circuit Design

Altogether, the final design iteration utilized the following electrical components (see **Figure 12**) connected to the Arduino, with the exception of the servo motor which was powered externally. The one-dimensional design utilized a solenoid valve, a momentary push-button, a toggle switch, and a potentiometer for locomotive actuation. One IMU was mounted on the leg for data acquisition and for studying of the frequency-domain components of the hopping mechanism. In the two-dimensional design, a second potentiometer was added for servo control. A second IMU device was mounted on the body to allow quantifying the roll about the plane of the body. The IMU on the leg was also configured to calculate the pitch and roll about the plane of the leg's motion. Similar to the one-dimensional setup, real-time data acquisition was possible

using a MATLAB-scripted interface that displayed the cumulative amplitude spectrum (FFT) and sampled accelerometer readings.

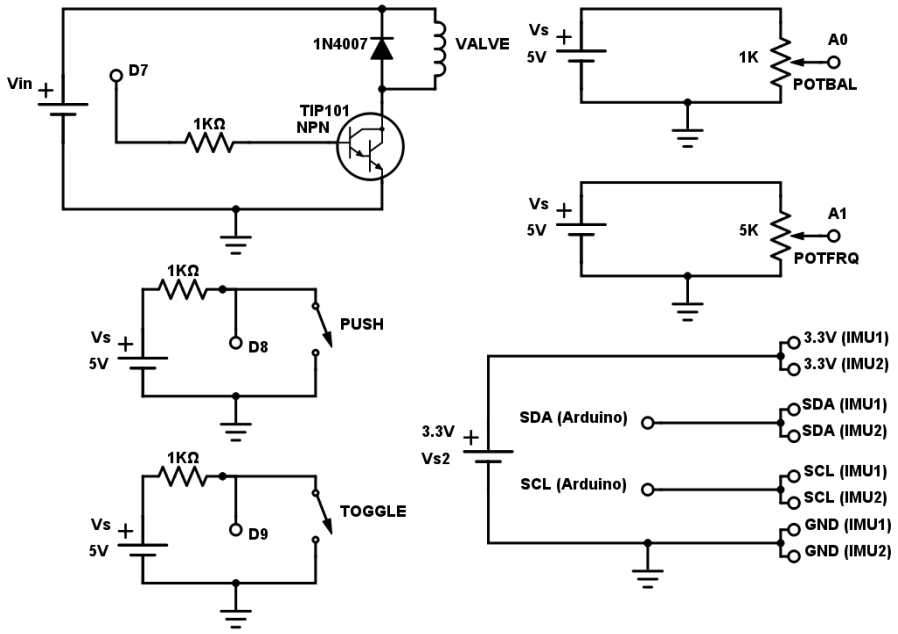

Figure 12: *Overall electronic circuit schematic using the Arduino microcontroller. All provided Ports labeled D7, D8, and D9 represent actual digital I/O pins used in the final design of the robot. Upper left: Transistor-based solenoid valve circuit. Middle and lower left: Single-pole, single-throw (SPST) push button and toggle switch circuits. Upper right and middle right: Potentiometer circuit - ports A0 and A1 represent actual analog input pins used. Lower right: Schematic representation of the I^2C mode wiring of two inertial measurement unit devices.*

3. Control System

One-Dimension

The One-Dimensional *Jax* only has the ability to extend and retract his leg. In Manual Mode, this is done via a push button. The default state is set to be with the leg retracted, the actuated state. When the push button is pressed, the leg extends, and when the push button is depressed the leg retracts. In addition, the processing software runs and gives the user a real time value for the jumping frequency and runs the accelerometer data through the Fourier Transform.

In Automatic Mode, the button pressing is done via computing. This process essentially works like a tunable pacemaker. The arduino senses the initial impact that is the landing and then, after some delay time set by the user, extends the foot. Then, when the arduino senses the body has left the ground, it retracts the foot. This methods maps the output of the potentiometer onto a small range of delay times which uses the arduino's internal clock circuit to fire a specific time after it senses the initial impact. This method adjusts a fixed rate of jumping but does so by quantifying the delay time between landing and thrust or the period of the landing phase.

Two Dimensions

The two-dimensional control system is largely the same. The Manual Mode and Automatic Mode from above translate exactly from 1D to 2D. The only major difference is that the acceleration values are read in the direction parallel to motion and are thus scaled proportional to the sine of the angle of the leg. However, the general underlying principles remain and both systems still operate effectively.

However, an additional control scheme had to be added to allow for angle control to prevent tipping. The control scheme for this was taking another potentiometer and mapping its range onto a smaller range of servo angle values. By turning the potentiometer, the servo rotates by a scaled angle similar. The servo also operates in two modes: Manual and Automatic; however, in both cases the servo knob's function remains the same and it is only the jumping that changes.

4. Materials

a. Material Selection and Processing

Jax's components were made of a short list of materials. All the parts were either aluminum, 3D-printed, or purchased through the ActoRobotics model set, aside from the foot pad made of smooth-on polyurethane. The initial design was to primarily use the ActoRobotics set, however, a desire to make the design more unique led to a more machined and printed design. Generally, the conditions that determined the material were: size and weight, necessary strength, complexity of shape, and production deadlines.

The largest parts that comprise *Jax*'s structure were purchased, such as the cross-beam that is part of the ActoRobotics framework and the Bimba pneumatic cylinder. Additionally, the pieces that bore the brunt of the weight and forces exerted were machined from aluminum, due to the fragility of 3D printed plastics. However, 3D printed parts were often used due to their light-weight and ease of production. This helped us to make *Jax* not be too heavy, as well as allow us to work on other important components in parallel while they printed.

One unique material chosen for our design is the smooth-on polyurethane. For our footpad, we needed a material that would have sufficient friction when in contact with the floor. Originally, we planned on using a piece of rubber, but this proved to be very difficult to work with. It was very hard to cut to shape, and it also proved difficult to use in conjunction with a ground-contact push button. Instead, we utilized smooth-on polyurethane, that was molded into a semi-spherical shape placed on the bottom of our device's leg. This helped to provide angular motion when testing 2D locomotion, something that we lacked with a flat rubber foot.

b. Manufacturing and Assembly

The manufacture and assembly of *Jax* and all involved parts was very important. Since the system is very susceptible to vibrations, special attention was paid to creating parts that fit well, in addition to fasteners that helped to prevent the resonance from making things fall apart. This lead to using force fits where necessary, as well as an ample supply of set screws and washers. Due to all of this, *Jax's* construction was not a simple one, especially considering our goal of weight and size reduction.

c. Packaging & Shipping

Since *Jax* was not meant to be a commercial product, not much attention was paid to packaging and shipping. Additionally, operation requires a large source of compressed air and power which would not be shipped with *Jax* nor would it be expected to be bought separately. That being said, the overall *Jax* system is relatively small, something that would lend itself to styrofoam molds. This means that *Jax* could be shipped in a fairly small box, with little to no danger and minimal assembly required.

If the two-dimensional design were to be shipped, *Jax* could be removed from the 2D boom and both could be shipped in one box together with styrofoam cutouts to hold them safely. This would require only an allen key to assemble upon delivery. The 1D *Jax* system could be shipped without any necessary requirement. However, even in both of these scenarios, the power and pressure systems would have to be acquired separately.

5. Ethical Aspects

It is important to take into account the safety considerations of this design. Particularly, this design involves rapidly-moving parts and electrical components. Both of these attributes

present a significant level of hazard for children, pets, and most inexperienced adults. Precautionary measures must be taken while actuating the device, particularly due to the highly compressed gas output of the pneumatic system. The large range of motion of the robot's moving limb is another characteristic that should one should be aware while operating. The presence of compressed gas systems and pressurized containers would require familiarity with pneumatics for safe and proper operation.

On the other hand, the electronic components of the design were ensured to be safe via the use of proper voltage and/or current regulation; no custom power regulators had to be assembled, and no significant heating was observed in the system. The only exception to the safety measures taken in the design considerations was the use of a high-torque servo motor, which has the ability to pull a large current. Though a current of around 2.5 amps may sound scary on its own, this current was appropriately provided by a regulated power supply with built-in shutdown when overloaded (i.e. excessive power supplied). If not provided by a regulated output, the servo motor could pose potential hazards should it stall during operation. In its fully assembled state, our current design poses no significant safety hazards with the exception of proper knowledge of pressurized gas usage and power supply considerations for moving parts (i.e. servo motors).

6. What We Learned

Creating *Jax* was a great learning experience for all of us. Besides the obvious experience of designing a product from start to finish, there were many necessary skills we were required to pick up along the way to aid in the overall design of *Jax*.

From a mechanical point of view, perhaps the largest subject we learned is pneumatics. Prior to this project, none of us had any experience in pneumatic design, and while *Jax*'s pneumatic system was not overly complicated, it was useful to use and understand a pneumatic system for the first time. What we foresee being the most useful skill learned over the course of this project is the practical application of machining and details from our mechanical education. From truly seeing how important the strength of a material can be, the crucial decision of different types of fits, to just learning about the choice of screw and thread size, seeing the real application of our education was truly appreciated.

From the electrical engineering perspective, studying the interface of the locomotive mechanism and the electronic and controls aspects of the design was particularly challenging. Controls algorithms did not seamlessly translate from analytical results to applied experiments by coding. Also, while programming in the Arduino environment first required a better grasp of low-level controls, the real-time operation of the robot was difficult due to sensors being susceptible to noise. Aspects of the robot's motion were obscured by other artifacts, such as sudden perturbations caused by the pneumatic system as well as unexpected readings created by the surface with which our robot was interacting. We used concepts borrowed from digital signal processing (e.g. finite-impulse response vs. infinite-impulse response filters) to reduce noise and filter irrelevant information from our sensors. Furthermore, we were able to create a convenient graphical interface via MATLAB that resembled some basic capabilities of a digital oscilloscope, such as real-time signal acquisition and simultaneous Fourier-domain analysis. Finally, in the additive process from 1D to 2D design, it was important to consider the communications protocol used amongst multiple devices, in this case, an Arduino and several sensors. In our

considerations to further expand sensing, we also explored multi-microcontroller communication, such as Arduino-to-Arduino networking (both wired and wireless).

Experiments & Test Results

1. Finite Element Analysis

An FEA Analysis was conducted on the leg design over concern of the 3D Printed pieces breaking, and the piston rod choking. Tests were conducted at angles both normal to the ground, as well as angled to the ground.

a. Assumptions

For all tests, materials were assumed to be either Aluminum 6061 or ABS Plastic for 3D printed pieces. This was made to tremendously simplify the simulation process. It was also assumed that the entire assembly is a rigid body. To further simplify the simulations, compressed air was not modeled within the pneumatic chamber. Though this would be key in deriving accurate results, my modeling the object as a complete rigid body rather than a pneumatic body, the stresses would be far higher – compressed air would make the object more springy and reduce stresses, so if we could prove through the rigid body simulations that the robot would not significantly deform, we would not need to model the complexity of compressed gases in the chamber.

Because the focus of the simulations was on the material and not the connections, all pieces were assembled with the assumption of perfect surface contacts, thus further stimulating the assembly as a rigid body. This assumes that all connections such as screws and bolts will not be points of failure, and once again significantly sped up the simulation process, without significantly impairing the nature of the results, relative to our purposes with the simulations.

The models used in the simulation were created in the 2009 release of Creo Elements/Pro (Pro/ENGINEER Wildfire 5.0), then imported and re-assembled into Autodesk Inventor 2016, remaining faithful to all constraints in the physical robot. Finally, the model was brought into Autodesk Simulation Mechanical 2016, from here on referred to as SimuMech, where the resulting simulations were carried out.

In SimuMech, simulations were conducted under Static Stress with Linear Material Models. This test assumes a static model with linear properties, undergoing a distributed load at a specified location or number of locations. It was assumed that the majority of the weight would be in the pneumatic cylinder, so this was where the weights are simplified in concentration. All forces are acting normal to the robot, with the robot's orientation normal to ground.

It is worth noting that after the first simulation, mesh size was dramatically reduced to prevent the simulation from stalling. These simulations were run on Windows 7 Enterprise, 32.0 GB RAM, Intel(R) Core(TM) i7-4790 Processor running at 3.60 GHz – it is hard to imagine a consumer-level computer more capable than this setup. The biggest limitation in this was the complexity of the assembly, with 27 parts all carefully constrained. Because of this, there is an upward limit regarding the accuracy of the simulations, no matter how many simplifications are reduced.

b. Results

Figure 13 shows a preliminary FEA test on the leg design, using the simplified assumptions above. Modeled at a mesh size of 100% according to SimuMech settings, the model was meshed into 23951 elements and 362 nodes. Placed under a 10 lb load, acting directly on the pneumatic cylinder, the maximum displacement was a miniscule 8.915×10^{-6} in. As expected, the

highest stresses were at the edges of the joints connecting each rod, as well as the 3D printed feet acting in direct contact with the floor.

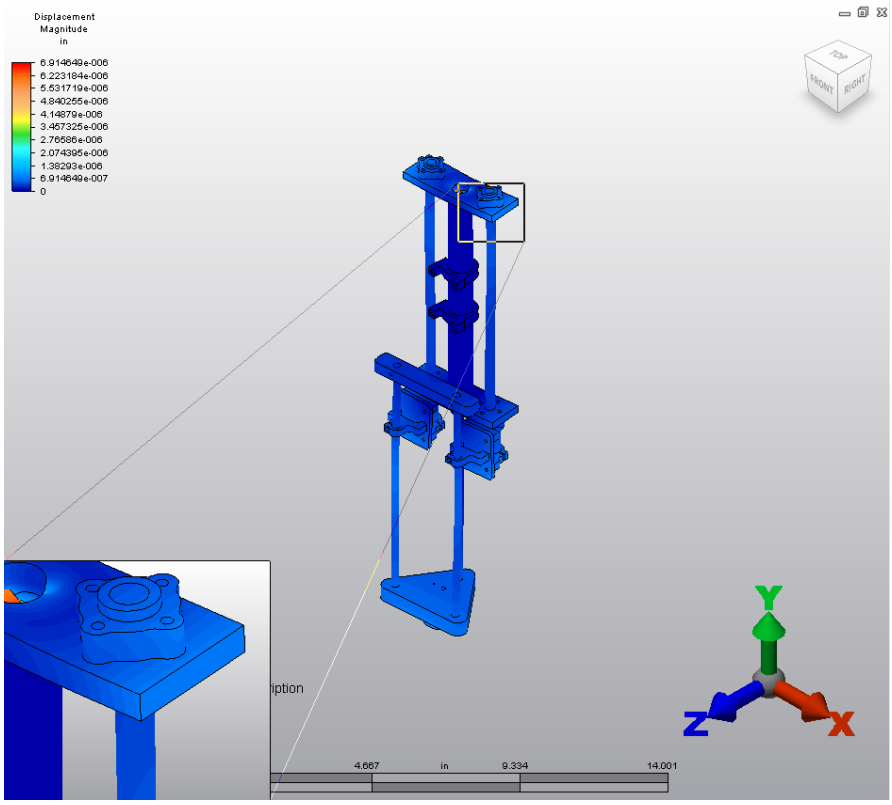

Figure 13: *Displacement of leg under 10lb load, normal to ground*

Though these numbers were heavily skewed by the necessary simplified assumptions above, a 10 lb load was our maximum expected load on the robot, so these results still suggest deformity would not be an issue.

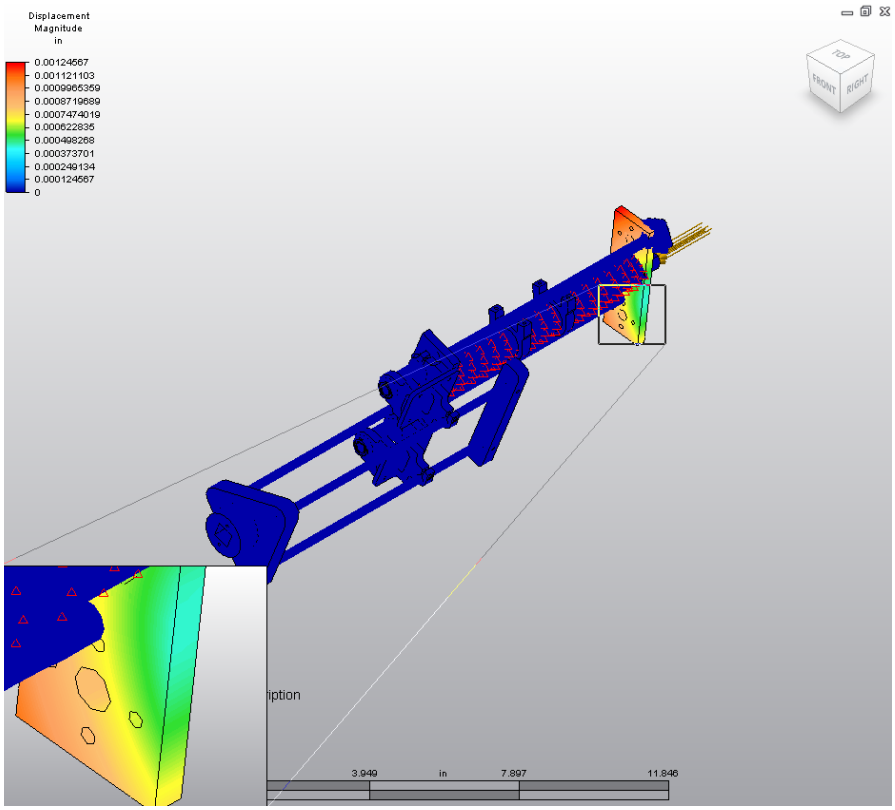

Figure 14: *Displacement of leg under 10lb load, angled to ground*

In the second simulation, the object was dropped at an angle with the same 10 lb force. And this time there was a dramatic increase in displacement. As seen in **Figure 14**, there is a maximum displacement at the top joint of 1.246×10^{-3} in, Though this is inconsequential to our design, it is indeed indicative that our greatest points of failure in the design are the joints between the rods. Assuming a repeated load like this would damage the 3D printed foot the most, as well as for noise considerations, a polyurethane mold was cast for the end of the foot, greatly reducing the stress on the joints and increasing shock absorptions.

2. Frequency Domain Processing

With *Jax* jumping in either the 1D or 2D setup, the accelerometer data was gathered by the arduino and passed through to MATLAB. In the arduino code is a low pass digital filter which smoothes out the noisy signal before passing it to MATLAB. In MATLAB, specifically the body angle and acceleration parallel to the leg were measured. Both values are measured directly by the IMU and passed through simple functions to the arduino.

In MATLAB, we performed a real-time Fast Fourier Transform which converted our data to the frequency domain (see MATLAB Source Code in Appendix). This was used to confirm the ability of the user to find the resonant frequency. If the user is able to find the resonant frequency, as *Jax* continues to jump, the harmonics start to grow with each successive successful jump. And, even if the resonant jumping curve is already established, jumps out of time with the resonant frequency will reduce the magnitude of the peaks and create some outside of the proper channels.

Figure 9 shows simultaneous time and frequency data. After a significant number of jumps, the major frequencies of the data, as found via a Fourier Transform, create high magnitude peaks in the Fourier Transform in the top subplot. The data below is the accelerometer data. The important information encoded in the accelerometer data is just how each hop's phase looks on an acceleration vs time curve. The first thing of note is that the acceleration is measured as proper acceleration in *g*'s. This means that free fall is actually 0, and $0\ m/s^2$ is $1\ g$.

The lower subplot can be divided into three phases: flight; landing; and thrust. The easiest section to define is thrust. The thrust section starts at every peak. Acceleration will reach

a maximum at two points: when compression is at its largest and *Jax* is at its lowest; and when the leg is extended, applying a large downward force on the ground. If these two points are to converge on each other -- which happens when the controls are successful, this singular point becomes the start of the thrust phase.

The thrust phase ends when the flight phase starts. This should theoretically be when acceleration reaches 0 on the graph -- representing free fall. However, in this case, because of noisy signals and jerk, the flight phase actually begins at the one lower peak caused by noise in each cycle.

This phase ends when the landing phase begins, which would be when the acceleration passes above 0. In reality, though, a threshold value is used because there is noise and jerk which makes the value of the acceleration range within a small window even during free fall. So, generally, the landing phase begins once the acceleration starts to climb rapidly or pass above 0.5.

Conclusion

This project proved to be exactly as intended: a tremendous learning experience for all involved, and an immersive education in legged robotics, as well as robotics in general. Our design process was well executed, and immense research was put into defining our design problem, as well as organizing the various limitations we face with each design choice. This process greatly aided us moving forward with our prototyping phase, as we were able to redesign our robot through three total iterations in quick succession, due to the limits of each design already laid out in our research. By knowing these limits, once the problems with a design were obvious, the alternative design required was often equally obvious. The design procedure taken, as well as a willingness to significantly reverse step and redesign, was a crucial lesson learned in this project.

Regarding future iterations of the design, this project was particularly hindered by expensive components. Many of our components, especially the pneumatic components such as pressure regulators, gauges, solenoid valves, and pneumatic cylinders, were acquired second-hand from the Department of Mechanical Engineering's machine shop. However, significant improvements could be made in buying versions of these parts that are often hundreds of dollars. Regardless of this, however, our team did an excellent job of studying a foreign topic to us and adapting in real time to what was learned, and for this we are deeply satisfied with our capstone project.

Appendices

Appendix I: References

[1] Raibert, Marc H. *Legged Robots That Balance*. Cambridge, MA: MIT, 1986. Print.

[2] Siciliano, Bruno. *Robotics: Modelling, Planning and Control*. London: Springer, 2009. Print.

[3]Delson, Nathan J. Dynamic Legged Robot. The Regents Of The University Of California, assignee. Patent US7503410 B2. 17 Mar. 2009. Print.

[4]De, Avik, and Daniel E. Koditschek. *The Penn Jerboa: A Platform for Exploring Parallel Composition of Templates* (2015): 1-26. Web. 1 Feb. 2016.

[5] Burdick, Joel, and Paolo Fiorini. "Minimalist Jumping Robots for Celestial Exploration." *Int J Robot Res The International Journal of Robotics Research* 22.7 (2003): 653-74. Web.

[6] Wu, Ting-Ying, T.-J Yeh, and Bing-Hung Hsu. "Trajectory Planning of a One-legged Robot Performing Stable Hop." *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010). Web.

[7] Raibert, M. H., H. B. Brown, and M. Chepponis. "Experiments in Balance with a 3D One-Legged Hopping Machine." *The International Journal of Robotics Research* 3.2 (1984): 75-92. Web.

Appendix II: Additional Financial Information

Table 3: Discounted Prices for Servo City Purchases

Product Name	Vendor	Cost (Discounted)	Qty	Total Cost (Discounted)
HS-7950T8 Servo Motor	Servo City	$127.49	1	$127.49
Standard HiTec Servo Block	Servo City	$22.94	1	$22.94
0.84 Inch Bore Clamp	Servo City	$5.09	2	$10.18
.25 Inch Bore Clamp	Servo City	$5.09	2	$10.18
15mm Bore Clamp	Servo City	$5.09	4	$20.37
8mm Set Screw Hub	Servo City	$4.24	2	$8.48
3 Inch Aluminum Bracket	Servo City	$3.39	2	$6.78
Flat Triple Bracket	Servo City	$2.12	1	$2.12
18 Inch Aluminum Bracket	Servo City	$11.89	1	$11.89
Swivel Hub	Servo City	$5.94	1	$5.94
8mm ID - 45mm L Linear Ball Bearing	Servo City	$2.97	2	$5.93
			Total:	$249.94

Appendix III: Source Code

MATLAB Interface for Real-Time Acquisition and Frequency-Domain Study

```
clear;
clc;
close all;

%User Defined Properties
serialPort = '/dev/tty.usbmodem1421';          % for Mac OS X
% serialPort = 'COM3';                  % for Windows
baudRate = 115200;
plotTitle = 'Serial Data Log';                  % plot title
xLabel = 'Elapsed Time [s]';                % x-axis label
yLabel = 'Data';                            % y-axis label
plotGrid = 'on';                            % 'off' to turn off grid
min = -2;                                % set y-min
max = 6;                                 % set y-max
scrollWidth = 2;                         % display period in plot, plot entire data log if <= 0
delay = 0.00001;                            % make sure sample faster than resolution

plotTitle2 = 'Single-Sided Amplitude Spectrum of y(t)';
xLabel2 = 'Frequency [Hz]';
yLabel2 = '|Y(f)| [dB]';
min2 = 0;
max2 = 2;

%Define Function Variables
time = 0;
data = 0;
count = 0;
trigger = 0;
freq = 0;
fourierFreq = 0;
magData = 0;

%Set up Plot

ax1 = subplot(2,1,1);
filterGraph = plot(ax1, fourierFreq, magData);
title(plotTitle2,'FontSize',18);
xlabel(xLabel2,'FontSize',15);
ylabel(yLabel2,'FontSize',15);
axis([0 10 min2 max2]);
grid(plotGrid);
```

```
ax2 = subplot(2,1,2);
plotGraph = plot(ax2,time,data,'-mo',...
                'LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor',[.49 1 .63],...
                'MarkerSize',2);
hold on
plotGraph2 = plot(time,freq,'-mo',...
                'LineWidth',1,...
               'MarkerEdgeColor','r',...
                'MarkerFaceColor',[.49 1 .63],...
                'MarkerSize',1);
hold off
title(plotTitle,'FontSize',18);
xlabel(xLabel,'FontSize',15);
ylabel(yLabel,'FontSize',15);
axis([0 10 min max]);
set(ax2,'YTick',[-1 0 1 2 3 4 5 6 7 8]);
grid(plotGrid);

%Open Serial COM Port
try
        s = serial(serialPort,'BaudRate',115200);
        disp('Close Plot to End Session');
        fopen(s);
catch
        fclose(s);
        error('Try failed.');
end
tic

try
        while ishandle(plotGraph) %Loop when Plot is Active
      stringData = fgets(s);
      dat = sscanf(stringData, '%f %d %f');

      if(~isempty(dat) && isfloat(dat)) %Make sure Data Type is Correct
            count = count + 1;
            time(count) = toc;  %Extract Elapsed Time
            data(count) = dat(1,1); %Extract 1st Data Element
            trigger(count) = dat(2,1); %Extract 1st Data Element
            freq(count) = dat(3,1);


            %Set Axis according to Scroll Width
```

```
            if(scrollWidth > 0)
            set(plotGraph,'XData',time(time > time(count)-scrollWidth),'YData',data(time >
time(count)-scrollWidth));
            set(plotGraph2,'XData',time(time > time(count)-scrollWidth),'YData',freq(time >
time(count)-scrollWidth));
            axis([time(count)-scrollWidth time(count) min max]);
            else
            set(plotGraph,'XData',time,'YData',data);
            set(plotGraph2,'XData',time,'YData',freq);
            axis([0 time(count) min max]);
            end

            if (length(time) > 50)
            trimTime = time(50:end);
            trimData = data(50:end);
            Ts = mean(diff(trimTime));
            Fs = 1.0/Ts;
            L = length(trimData);
            NFFT = 2^nextpow2(L);
             fourierFreq = Fs/2.0 * linspace(0, 1, NFFT/2 + 1);
            fourierData = fft(trimData, NFFT)/L;
            magData = 2.0*abs(fourierData(1 : NFFT/2 + 1));

            set(filterGraph,'XData',fourierFreq,'YData',magData);
            end

            %Allow MATLAB to Update Plot
            pause(delay);
      end
         end

         %Close Serial COM Port and Delete useless Variables
         fclose(s);
         clear count dat delay max min plotGraph plotGraph2 plotGrid plotTitle s ...
             scrollWidth serialPort xLabel yLabel ax1 ax2 Fs Ts L max2 min2 ...
             NFFT plotTitle2 xLabel2 yLabel2 trimTime trimData fourierData;

         disp('Session Terminated...');
catch
         fclose(s);
         error('Error encountered. Exiting program.');
end
% save('test.mat','time','data','freq','fourierFreq','magData','trigger')
```

Published with MATLAB® R2015a

Arduino Interface for Data Acquisition and Control Systems

Appendix IV: CAD Drawings