Dynamic Robotic Leg

ECE 445 Design Document - Spring 2019

Team 1 - Ahsan Qureshi, Kanyon Edvall, and Joseph Byrnes

TA: David Hanley

1 Introduction:

1.1 Objective:

While wheeled robots excel in flat terrain, they still face challenges with navigating a world created for humans [1]. Similarly, aerial robots suffer from difficulties such as moving through confined spaces and interacting with the environment around them. The problem of navigating multi-level environments, including stairs and unstructured environments such as a floor with debris or uneven terrain, has not been fully solved with ground robots. Although aerial robots have been deployed in these situations, they face many obstacles as listed in page 172 of [2]. Legged robots, such as quadrupeds, are able to successfully navigate a multitude of human-oriented environments without the restrictions imposed by aerial and wheeled vehicles. Specifically, dynamically stable quadrupeds hold a great deal of promise for navigating unstructured environments because of their large workspace, ability to reject disturbances quickly, and the flexibility in movements they can perform. Far more research must be conducted before legged robots will be useful in real world deployment. Currently, there is no existing open source platform which researchers are able to use to develop algorithms for legged robotics.

We propose to create a two degree of freedom robotic leg stabilized on a test frame in order to demonstrate trajectories used for walking gaits. This project will demonstrate the feasibility of inexpensive walking robots through a fully functional leg and test rig that can easily be reproduced with consumer off-the-shelf components and built upon to provide the starting point for a novel quadrupedal robot. We will provide an open-source design including all necessary structural components, a complete bill of materials, and necessary software to control the leg. It is our hope that this project can serve as the base for future research projects expanding on dynamic leg control and its applications by providing a ready-to-build leg that handles all low-level control internally.

.

1.2 Background:

While research labs and companies have been developing dynamic legged robots for years— Boston Dynamics' Spot mini [3], Unitree's Laikago [4], Ghost Robotics' Vision [5], MIT's Cheetah [6], etc— all of these robots use custom motors and/or proprietary control algorithms which are not conducive to increasing the development of legged robotics. Each of the quadrupeds listed above started as a single leg tested on a stabilized platform before integrating the leg into the entire robot. Currently, when developing a new legged robot it is necessary to develop a custom leg as a starting point much like the examples listed above. With a fully open-source, sub one-thousand dollar dynamic robotic leg and controller, we believe we can accelerate the development of legged robotics by reducing entry barrier both technically and financially to increase involvement in the field.

1.3 High Level Requirements:

- The leg must be built with inexpensive (sub $100) motors that can be purchased from major hobby parts vendors such as HobbyKing or Alien Power Systems. The motors must be usable without any modification other than adding hall sensors and changing connectors.
- The leg must be able to apply a peak torque at least four times greater than the torque required to stand from the folded position (illustrated in Figure 11, Section 2.9) through the use of a task space controller. This allows the leg to execute walking gaits and jump.
- The leg must be able to execute pre-calculated trajectories that follow distinct coordinates within the allowed work space in real time to demonstrate jumping and partial walking gaits. The end effector (foot) must follow the trajectory within 5 mm of the commanded position.

2 Design:

The leg requires five main types of electrical components for operation: brushless DC motors to drive each joint, magnetic absolute encoders to report the angle of each joint, a motor controller to command a torque to each joint, a central processor to run the task-space controller, and a suitable power supply unit to power the leg.

We will 3D print the majority of the parts for the leg. The gear and pulley reductions will be accomplished with off the shelf components. Figure 6 depicts the baseline test setup for a two degree of freedom leg. Our test setup for a three degree of freedom leg will be very similar, however, we do not have any renders of a 3 degree of freedom setup at this time. Our test stands will be based on the stands depicted on page 82 of [8] and page 300 of [9].

The leg will use a task space controller based on the dynamics of the leg derived from the Lagrangian method in order to control the force of the foot in the leg task space. We will control the leg with a hybrid position-force task space controller to demonstrate programmable compliance in the leg task space. We will use a modified version of the ODrive open source motor controller to control the torque of the joints. The joints will be driven with high torque off-the-shelf brushless DC motors. We will use high precision encoders to read the angles of each joint. The inverse dynamics calculations and system controller will run on a TI F28335 processor.

Alternative that we considered in creating our design was using prefabricated gearboxes rather than a custom belt and pulley system. However, the gearboxes more than doubled the weight of the actuator and added 50% extra cost to our actuators. Furthermore, when using gearboxes we

could not control the backlash between in the input and output, but by using belts and pulleys, we can modify the pulley and belt tension until we can achieve a satisfactory amount of backlash. Additionally, we considered using more expensive motors with higher power ratings and form factors more conducive to a legged structure, but their cost would be prohibitive to researchers seeking a low cost platform for algorithm development.

The mechanical components of the leg will be 3D printed and kept simple to focus on the other aspects of the project. The majority of the mechanical design has already been completed since the start of the semester. The test frame will be built out of V-Slot aluminum extrusion.

The central control algorithms will run on a TI F28335 processor. We will purchase a development card carrying this processor and we will design a PCB to support the development card and route signals to the motor controllers and encoders. We will also create a separate PCB to hold each of the magnetic encoders.

We will also develop a simulation using pyBullet [7] so that we can test trajectories of the end-effector (foot) before applying them to the real leg.

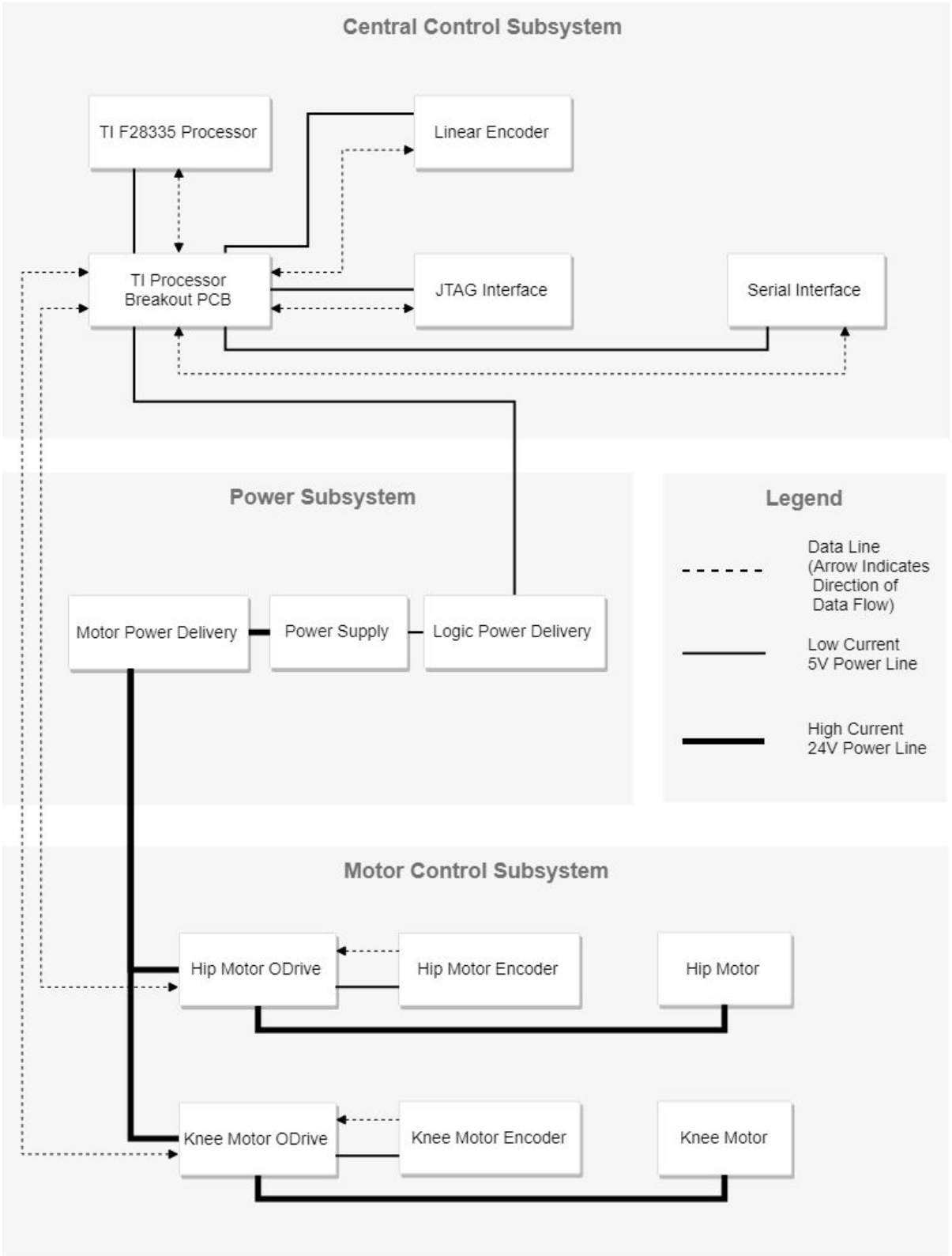

Figure 1: Block Diagram

2.1 Power Subsystem

The power system is required to supply suitable current to the motors and to power the motor controllers and central controller.

2.1.1 Power Supply

A 120V / 220V to 24V power supply will provide power to the robotic system. Due to the high power requirement a server grade power supply will be used.

Requirement	**Verification**
Provided voltage must stay between 24.5V and 25.5V	Use a multimeter to confirm that the output of the power supply stays within 24.5V and 25.5V during 30A operation of each motor.
Must provide a minimum of 30A continuous current per motor. At 24V, this corresponds to a maximum power output of 1500W	Set both motors to draw 30A continuously and confirm with a current probe that at least 30A is being supplied to each motor.

2.1.2 Logic Power Delivery

The logic power delivery system will consist of a wiring harness to connect the power supply to the Processor Breakout PCB.

Requirement	**Verification**
Use wire that does not cause a voltage drop of more than 1.5% of the power supply voltage.	Measure voltage drop across the power cables using a multimeter and confirm that drop is below 1.5% of power supply voltage.

2.1.3 Motor Power Delivery

The motor power delivery system will consist of a wiring harness to connect the power supply to each motor controller board.

Requirement	**Verification**
Use wire that does not cause a voltage drop of more than 1.5% of the power supply voltage.	Measure voltage drop across the power cables using a multimeter and confirm that drop is below 1.5% of power supply voltage.

2.2 Central Control Subsystem

2.2.1 TI F28335 Processor

We will use the *TMS320F28335 controlCARD* to run our task space algorithms and trajectories. The F28335 has support for all communication we need and will be able to handle the real time control required of our project. The robot will be connected to wall power at all times and as such there is no power restrictions for the processor. In addition, the microcontroller does not need any persistent memory aside from storing firmware.

Requirement	**Verification**
Must be able to perform all dynamics calculations for our task space controller in less than 1 millisecond.	Output a 0v signal through an output of the F28335 processor when the calculations are finished and set the signal to 5v when the periodic task starts again. Confirm using an oscilloscope that the processor is not being overloaded by measuring the duty cycle of the signal, confirm duty cycle is under 1ms.

2.2.2 Custom F28335 Breakout PCB

We will design a circuit board to break out the connections of the ControlCARD and provide power regulation and protection to the processor. The board will provide JTAG connection for programming and real time debugging. It will provide a serial port for streaming real time data to MATLAB or similar.

Requirement	**Verification**
Board must continually supply 4.9 to 5.1 volts to the ControlCARD.	Using a multimeter, read the voltage at the output of the regulators when leg is stationary and when 30A per motor is being supplied to ensure regulators are not affected by voltage sag due to the the motor power system. Verify the voltage reading is between 4.9V and 5.1V.
Regulators must stay below 50°C for at least 30 minutes of operation under room temperature	Using an infrared thermometer, measure the temperature of the regulators after 30 minutes of operation.

Must breakout the ControlCARD's JTAG interface for programming using an Atmel ICE programmer.	Connect PCB to a computer using Atmel ICE programmer and confirm proper firmware upload and debugging communication by setting a variable in real time through the Code Composer Studio expressions interface. Confirm variable has changed as expected.
Must provide USB access to the ControlCARD's serial interface.	Connect ControlCard to PuTTy Terminal and confirm debugging data such as joint angle sent through USB Serial port is functioning.

2.2.4 Linear Encoder (Leg Height Sensor)

We will use a rotational encoder to measure the movement of a belt loop attached to the gantry to determine the height of the leg. The belt will rotate around the entire vertical bar and turn the encoder. We will always start the leg at a known height upon power-up.

Requirement	**Verification**
Must measure robot height with accuracy of at least 2mm	Using calipers, move the leg up and down by 1mm at a time and record distance travelled reported by the encoder/processor. Readings are considered good if measured distances is within + or - 1mm of manually measured distance.
Must report distance travelled with a frequency of at least 1ms.	Connect encoder to oscilloscope and measure frequency of data output. Ensure period is less than or equal to 1ms.

2.3 Motor Control Subsystem

2.3.1 Motor Controller Boards

Each motor will be driven by an ODrive motor controller. This open-source board will be modified to suit our use case.

Requirement	**Verification**
Must support current sensing hardware to achieve torque-based control. Current sensing must be accurate to 0.2% of actual value.	Using an ammeter, measure current draw on motor power lines. Confirm that measured value is within 0.2% of the motor controller sensed value.

Must support current output control to achieve torque-based control. Current output must be accurate to 0.2% of set value.	Using a high-precision lab scale confirm that the torque output from the joint is within 0.2% of the set torque value.
Must deliver a minimum of 35A continuous current for 60 seconds without rising above 50°C.	Use an infrared thermometer to measure the heat being radiated from the MOSFETS in the motor drive circuit, record and plot the data over the period of 60 seconds of continuous operation. Simultaneously, use a non contact multimeter to measure the current provided to make sure it is 35A for the entire operation.

2.3.2 Motor Encoders

We will use quadrature encoders to measure the shaft position of each motor. These encoders will connect directly to each motor controller board to form a closed-loop feedback system. Their data will also be sent to the central processor for calculating the angle of the joint. We will use a quadrature decoder interface chip such as HCTL-2000 the to decode the positional data from the encoder and report it to the central processor.

Requirement	**Verification**
Must report angle within 0.25 degrees of true angle.	Connect the encoder to a stepper motor and rotate the shaft for a known amount of steps for one full rotation. Record the position data from the encoder and confirm that the encoder reports the angle within 0.25 degrees of the true angle throughout the entire rotation.
Must have data output frequency of 1000 Hz or greater to achieve joint angle updates every 1 ms.	Connect encoder with counter interface to an oscilloscope and measure frequency of data output. Ensure period is less than or equal to 1ms.

2.3.3 Motors

A motor will be mounted at each joint. KEDA 63-64 190KV motors will be used for initial prototyping and testing. These will potentially be upgraded to APS 6355 60KV motors or equivalent for more consistent torque output and a lower gear ratio between motor and joint.

Requirement	**Verification**

Must allow for up to 30A continuous current for 10 minutes without damaging the motor coils or raising the temperature of the plastic enclosure above 65°C.	Using an infrared thermometer, measure the temperature of the enclosure after 10 minutes of operation. Ensure it is below 65°C.
Must have a torque constant of at least 0.05Nm/A.	Using a spring scale or digital luggage scale, confirm that the torque output from the joint is at least 0.05Nm when 1A is applied.
Must be less than or equal to 63mm in diameter and less than or equal to 90mm long to fit in motor housing.	Measure the motor using calipers with a precision of at least 1/10 mm.

2.4 Controller Software

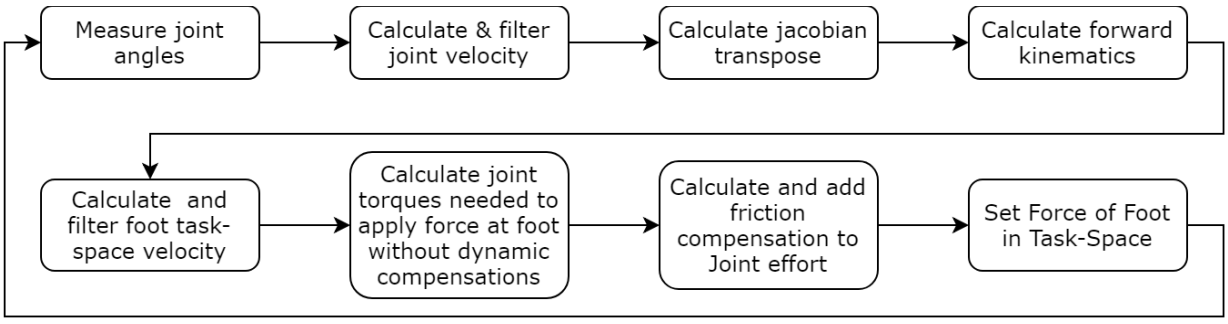

Figure 2: Task Space Controller Flow Chart

2.5 Schematics

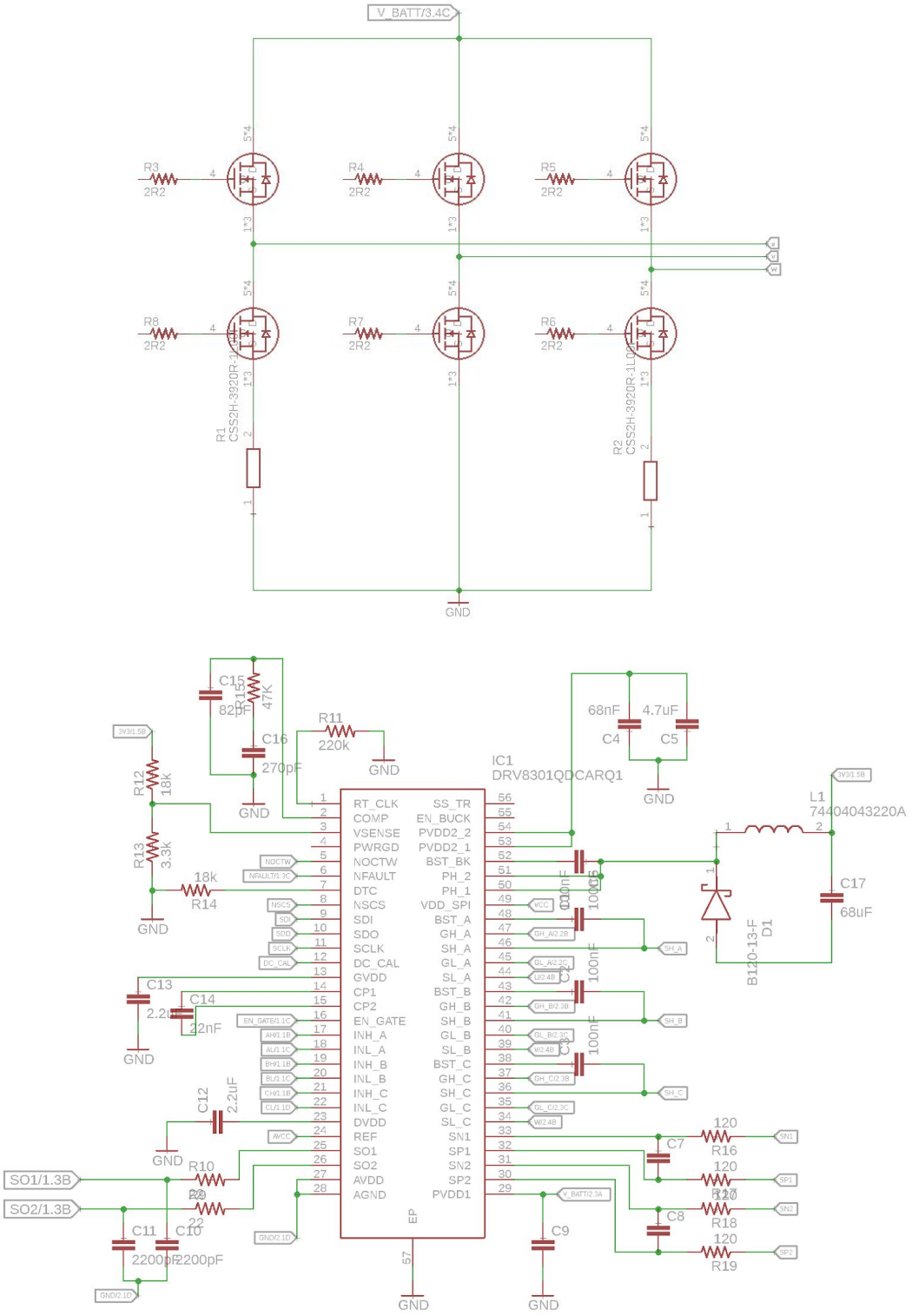

Figure 3: Motor Drive Circuit Schematic (Work In Progress)
(Based on Odrive v3.5)

2.6 Board Layout

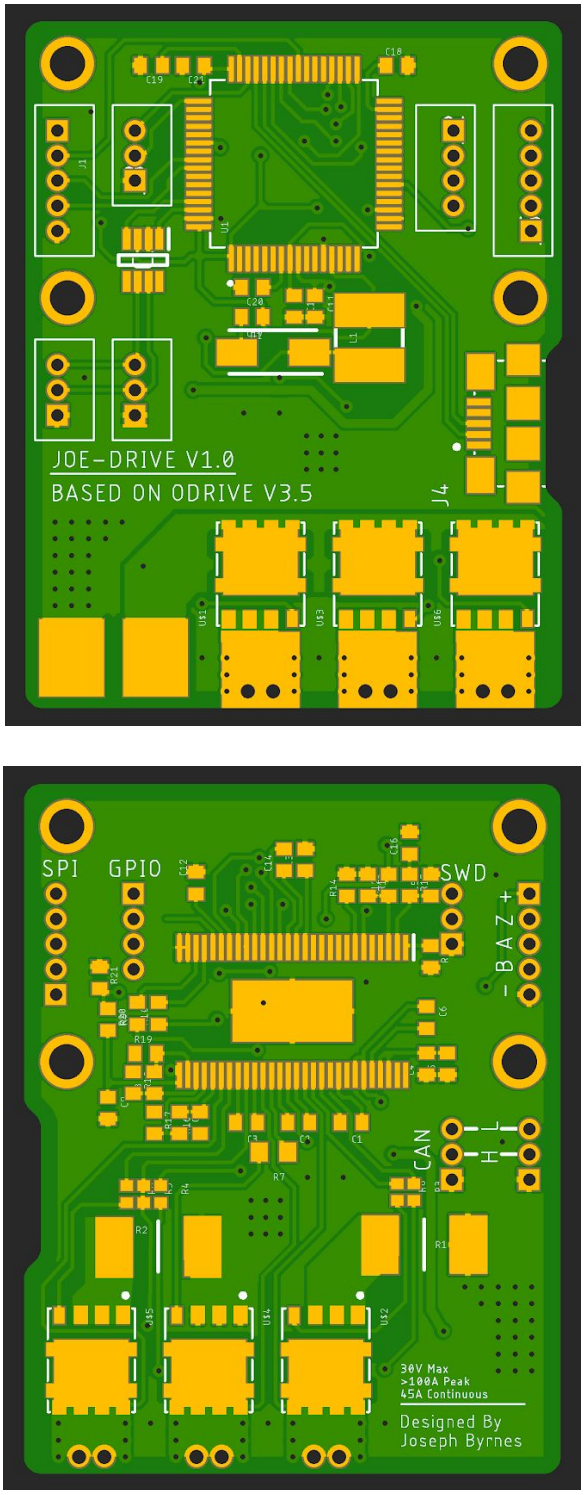

Figure 4: Renders of Custom ODrive PCB

2.7 Simulations

2.7.1 MATLAB and Simulink Simulation

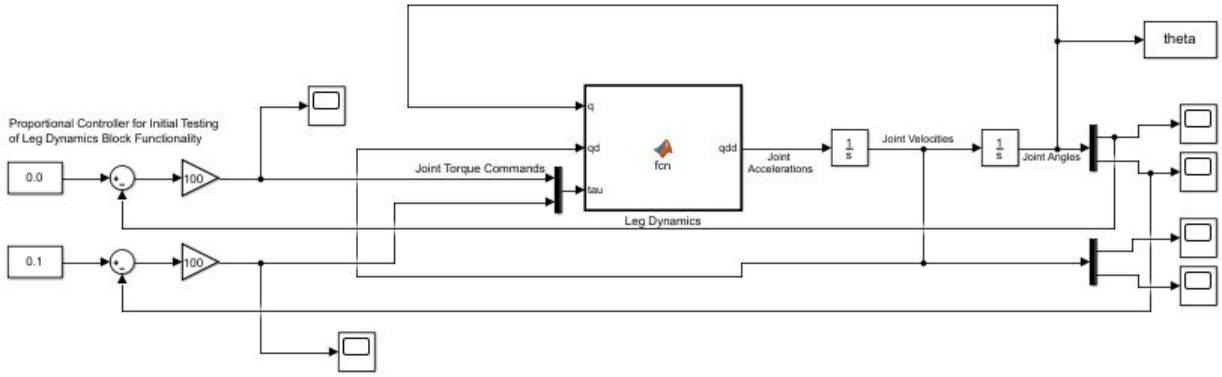

Figure 5: Simulink Model for Simulating Robot Equations of Motion

(See Appendix Section A for code)

2.7.2 PyBullet Simulation

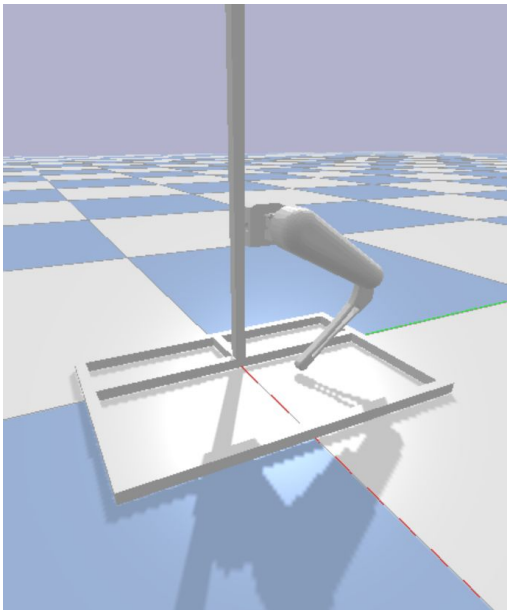

Figure 6: PyBullet Simulation for Visual Trajectory Development

(See Appendix Section B for code)

2.8 Physical Design

2.8.1 Measurements

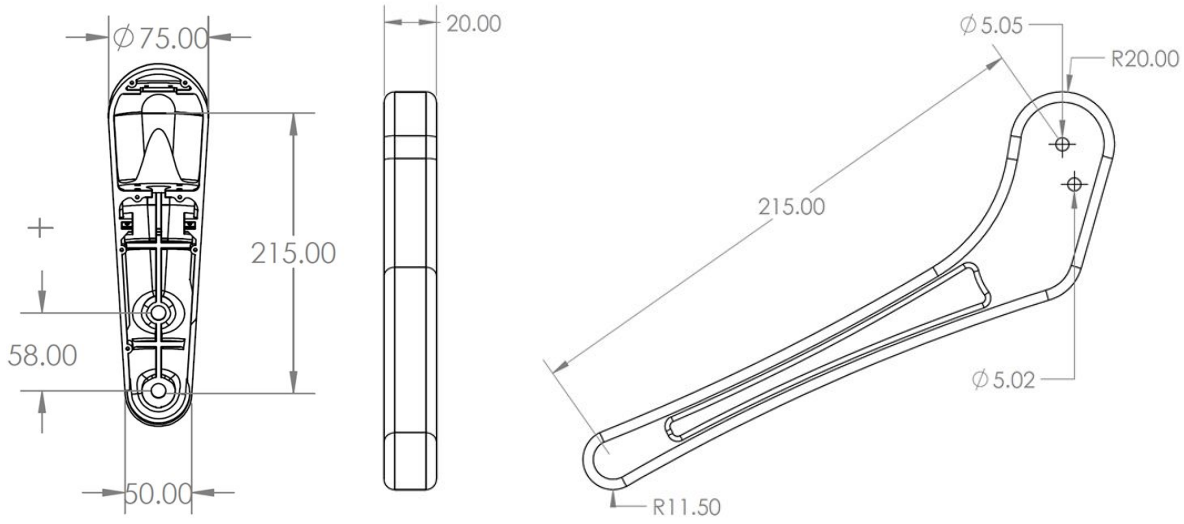

Figure 7: Upper Leg and Lower Leg Original Prototype Dimensions (mm)

2.8.2 Mechanical Renders and Prototype

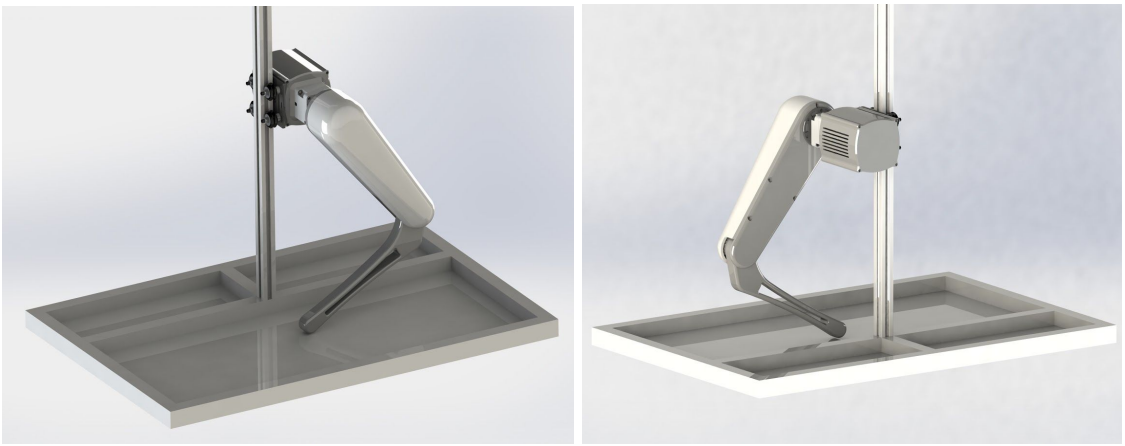

Figure 8: Test Stand with Leg 3D Model

Figure 9: Robotic Leg 3D Model

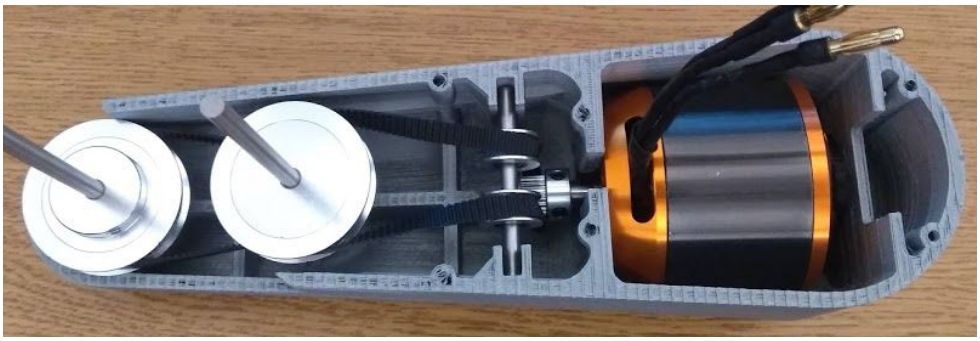

Figure 10: First Upper Leg Prototype Print

Figure 11: First Upper Leg Prototype Print

2.9 Tolerance Analysis

Correct and sufficient torque output of our actuators is the most critical requirement of our project. We must maintain a tolerance on our applied motor torque that is small enough such that the torque does not deviate to a value too low to move the leg. We will analyze the torque required of our knee motor for lifting the leg from the folded position illustrated in Figure 12.

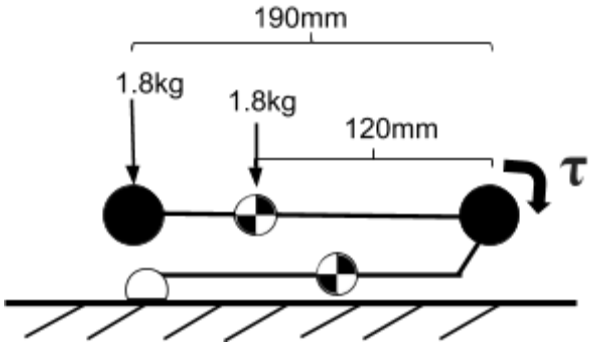

Figure 12: Leg Mass Illustration

As illustrated, the length of the upper link is 190mm. The center of mass is located 120mm away from the knee axis, and the mass of the hip can be approximated as a point mass located at the hip axis 190mm from the knee axis. Each point mass is approximately 1.8kg due to the motors and gearboxes used. Thus, we can write the torque required to hold the upper leg in place assuming the knee axis is fixed. This will give us a lower bound on the torque required to rotate the upper leg to a vertical position.

$$\tau = 1.8kg * 9.81m/s^2 * 0.19m + 1.8kg * 9.81m/s^2 * 0.12m = 5.47Nm \quad (1)$$

To effectively jump, we have chosen to use a minimum of 5x the torque required to stand from a folded position. Applying a 5x multiplier to our standing torque will provide us with a safe target for outputting a jumping force— However, once our MATLAB simulation is complete, we will verify the minimum torque chosen and plot how high it will allow us to jump. This means we require that the actuator be able to output at least 27.5Nm throughout its entire rotation. We are using motors with an expected torque constant of 0.0502Nm/A. The tolerance of the torque constant is very important to us to guarantee that our actuators will output our required joint torque. The closest available gear ratio to meet our torque requirement while providing the fastest maximum angular rate is a 20:1 reduction. Therefore, with a max current of 30A per motor, a torque constant of 0.0502Nm/A, and a 20:1 gear reduction, we see from Eq. 2 that we have a maximum torque output of 30.12Nm.

$$(\frac{20}{1})(30A)(0.0502Nm/A) = 30.12Nm \quad (2)$$

Additionally, we have specified tolerances on our motor encoder and current sensing which both affect the torque output of our motor. Specifically, if the motor encoder if incorrect by 0.25 degrees- our maximum tolerance- the resulting torque output of our motor will be 98% of the

maximum possible torque. This is due to the 6 pole pairs in our motor and the physics of the FOC algorithm driving the motor. Thus, with a worst case condition of encoder angle sensing, the torque constant of the motor is required to be at least 0.0493 Nm/A. However, the motor controller also has a tolerance on it's current sensing capability. If the current sensed by the motor controller is 0.2% higher than the true value, it will result in a torque 99.8% of the maximum value we can apply. Thus, the torque output from the motor would act as if the motor has a torque constant of 0.0492Nm.

Taking into account the worst case scenarios of the motor encoder reading and current sense reading, we find through Eqs. 3 and 4 that the tolerance on our motor torque constant must be within 6.9% of the specified value.

$$\frac{0.0458 Nm/A}{\frac{14.75}{15}(0.998)} = 0.0467 Nm/A \tag{3}$$

$$\frac{0.0467}{0.0502} = 93.02\% \tag{4}$$

Therefore, by confirming that the torque constant of our motor is within 6.9% the specified value from the manufacturer, we will ensure that our actuator is capable of delivering enough torque at all times.

3 Cost

Labor:

Team Member:	Hourly Wage	Weekly Hours	Number of weeks	Multiplier	Cost per Member
Joseph Byrnes	$35.00	30	12	2.5	$31,500.00
Ahsan Qureshi	$35.00	12	12	2.5	$12,600.00
Kanyon Edvall	$35.00	12	12	2.5	$12,600.00
				Total Labor Cost:	$56,700.00

Parts:

Part Name	Description	Manufacturer/Supplier	Quantity	Unit Cost	Total Cost
TMDSCNCD28335	F28335 ControlCard	Texas Instruments	1	$69.00	$69.00
ODrive	Motor Controller	ODrive Robotics	1	$119.00	$119.00
KEDA	Brushless Motor	KEDA	2	$20.00	$40.00
AMT102-V	Rotary Encoder	DigiKey	4	$23.63	$94.52
Central PCB	Regulator/data routing PCB	PCBWay	1	$5.00	$5.00
Planetary Gearbox	Planetary Gearbox for Motor	AndyMark	2	$84.00	$168.00
3D Printer Filament	PETG Filament	Hatchbox	1	$24.00	$24.00
Knee Axle	3/8 Inch Hex Shaft	Vex	1	$11.99	$11.99
Knee Bearings	3/8 Inch Hex bearing	Vex	2	$2.99	$5.98
Motor Mounting Screws	M4x12 Screws	Amazon	1	$3.99	$3.99
T-Slot Extrusion	10 x 1m T-slot 2020	ZYLTech	1	$49.99	$49.99
T-Slot Angle Brackets	12 x Angle brackets	ZYLTech	2	$6.95	$13.90
Power Supply	24v 1500w Server Power supply	Dell / Ebay	1	$27.99	$27.99
T-Slot wheels	10 x T-slot wheels for gantry	Amazon	1	$7.99	$7.99
			Total Parts Cost:		$641.35

Grand Total:
$56,700.00 (Labor) + $641.35 (Parts) = $57,341.35

4 Schedule:

Week	Tasks	Team Members
Feb 18th	1. Finish Matlab Simulation of Equations of Motion 2. Finish first Prototype Leg 3. Start building test frame 4. Verify linear encoder	1. AQ, JB 2. JB 3. JB,KE 4. KE
Feb 25th	1. Verify encoders 2. Verify power supply 3. Calculate leg Jacobian and jacobian inverse 4. Measure weights and calculate center of masses and inertias of all parts 5. Derive equations of motion with Lagrangian method 6. Finish full MATLAB Simulation 7. Finish schematic and begin layout of Central PCB 8. Verify motor encoders 9. Verify motors 10. Finish building test frame. Verify it meets safety standards and operation requirements	1. JB, KE 2. JB, KE 3. AQ 4. JB, KE, AQ 5. AQ, JB 6. JB 7. JB 8. KE 9. JB, KE 10. JB, KE
March 4th	1. Finish building final leg 2. Work on ODrive PCB, finalize parts and schematic	1. JB, KE 2. JB 3. JB, KE

	3. Verify central breakout PCB 4. Start task space controller 5. Implement reference tracking and friction compensation. 6. Verify TI F28335 processor 7. Finalize Central PCB and Order	4. JB, KE 5. AQ, JB 6. KE 7. JB
March 11th	**Spring Break**	
March 18th	1. Set up programming environment for F28335 2. Begin programming Task Space controller 3. Verify motor controller boards 4. Finish designing custom Odrive PCB 5. Verify Logic Power Delivery 6. Verify Motor Power Delivery	1. KE 2. AQ, JB, KE 3. JB, KE 4. JB 5. KE 6. KE
March 25th	1. Continue implementation of task space controller, verify that leg follows given trajectories 2. Prepare trajectories for demonstrating jumping / walking in MATLAB using simulation 3. Set up streaming and plotting of data through serial port 4. Integrate Motor Control and Power subsystems 5. Verify integration of subsystems	1. AQ, JB, KE 2. JB 3. KE 4. JB, KE 5. JB, KE
April 1st	1. Verify functionality of custom ODrive PCB and decide whether to use it over	1. JB, KE

	purchased ODrive used for development 2. Integrate Central Control subsystem with Motor Control and Power subsystems 3. Verify Central Control integration 4. Determine Optimal trajectories for jumping and landing in simulation	2. JB, KE 3. JB, KE 4. AQ
April 8th	1. Test complete leg with jumping trajectories 2. Verify full leg functionality 3. Finalize PyBullet Simulation to closely model real leg	1. AQ, JB, KE 2. AQ, JB, KE 3. JB
April 15th	1. Further refine leg operation and make any necessary adjustments or repairs 2. Implement adjustable simulated spring and damper reactions when leg lands from jump 3. Implement adjustable and predictable jumping heights	1. AQ, JB, KE 2. AQ, JB, KE 3. AQ, JB, KE
April 22nd	1. Prepare for presentation 2. Continue implementing and debugging Jumping and partial walking demonstrations	1. AQ, JB, KE 2. AQ, JB, KE
April 29th	1. Final Paper Due	1. AQ, JB, KE

5 Safety and Ethics

Within our project, there are several potential safety hazards.The first concern that we will deal with is disabling the robot leg if it begins to move unexpectedly. To mitigate this issue we will include an emergency stop button which will disable power to the motors. Additionally, our motor controllers will have a failsafe enabled that disables the motors if no communication is received after a predetermined number of missed packets or time. Another safety hazard that we must consider is the inclusion of high wattage power supply units within our design. Continuously drawing a high amperage may overload lab circuit breakers and power supply connections can be hazardous if not enclosed properly. We have included a power dissipation resistor to prevent the regenerative current of the motor from damaging the overall electrical system of the robot or power supply units. Additionally, we will 3D print protective enclosures to ensure all exposed connections are properly covered.

A robotic leg by itself does not prove to have ethical challenges, however, once the leg is applied to a mobile platform, it may present ethical challenges. Isaac Asimov founded the idea of robots and established the Three Laws of Robotics, being 1) A robot may not injure a human being or, through inaction, allow a human being to come to harm; 2) A robot must obey the orders given it by human beings, except where such orders would conflict with the First Law; and 3) A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws [10]. An autonomous mobile platform could be designed using our legs which could be used as a mount for a weapon, or as a vehicle to perform illegal surveillance. These acts would violate the IEEE code of ethics, specifically, code 1 and code 9 [11]. We would prevent our designs from being used for harmful purposes by withholding our leg and technology associated with it from the public sphere. We would vet potential consumers of our product so that we can prevent it being used for purposes we find to be in violation of our ethics. At the same time, our research could be used for positives as well. For example, having met with officers of the UIPD, we have discussed future applications of our robotic leg such as the integration of the leg onto a mobile reconnaissance platform. This platform would provide increased safety to the officers by distancing them from potential threats such as explosive devices or hidden suspects.

Citations

[1] Chakraborty, N. and Ghosal, A. (2019). Kinematics of wheeled mobile robots on uneven terrain. [online] Science Direct. Available at: https://www.sciencedirect.com/science/article/pii/S0094114X04001478 [Accessed 22 Feb. 2019]

[2] G. Chowdhary, F. Dellaert, M. Kaess, E. Johnson and A. Wu, "Autonomous Flight in GPS-Denied Environments Using Monocular Vision and Inertial Sensors", Arc.aiaa.org, 2013. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/1.I010023. [Accessed: 22- Feb- 2019].

[3] Bostondynamics.com. (2019). SpotMini | Good things Come In Small Packages. [online] Available at: https://www.bostondynamics.com/spot-mini [Accessed 22 Feb. 2019].

[4] Unitree.cc. (2019). Unitree-Laikago. [online] Available at: http://www.unitree.cc/e/action/ShowInfo.php?classid=6&id=1 [Accessed 22 Feb. 2019].

[5] Revolutionizing Legged Robots | Ghost. (2019). Legged UGVs | Ghost Robotics. [online] Available at: https://www.ghostrobotics.io/robots [Accessed 22 Feb. 2019].

[6] Chu, J. (2019). “Blind” Cheetah 3 robot can climb stairs littered with obstacles. [online] MIT News. Available at: http://news.mit.edu/2018/blind-cheetah-robot-climb-stairs-obstacles-disaster-zones-0705 [Accessed 22 Feb. 2019].

[7] Coumins, E. (2019). bulletphysics/bullet3. [online] GitHub. Available at: https://github.com/bulletphysics/bullet3/tree/master/examples/pybullet [Accessed 22 Feb. 2019].

[8] Kalouche, S. (2019). Design for 3D Agility and Virtual Compliance using Proprioceptive Force Control in Dynamic Legged Robots. [online] Ri.cmu.edu. Available at: https://www.ri.cmu.edu/pub_files/2016/8/kaloucheThesis.pdf [Accessed 22 Feb. 2019].

[9] Ding, Y. and Park, H. (2019). Design and experimental implementation of a quasi-direct-drive leg for optimized jumping - IEEE Conference Publication. [online] Ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8202172 [Accessed 22 Feb. 2019].

[10] B. Deng, “Machine ethics: The robot's dilemma,” *Nature News*, 01-Jul-2015. [Online]. Available: https://www.nature.com/news/machine-ethics-the-robot-s-dilemma-1.17881. [Accessed: 22-Feb-2019].

[11] ieee.org. (2019). IEEE Code of Ethics. [online] Available at: http://www.ieee.org/about/corporate/governance/p7-8.html [Accessed 22 Feb. 2019].

Appendix:

A) MATLAB Code

```
function qdd = Leg_Dynamics_1(q, qd, tau)
    % Setup for using joint angle and angular velocity inputs
    t1 = q(1);
    t2 = q(2);
    t1d = qd(1);
    t2d = qd(2);
    % Definitions of link masses, center of mass locations, and inertias
    l1 = 0.215;
    l2 = 0.215;
    lc1 = 0.05;
    lc2 = 0.10;
    m1 = 0.950;
    m2 = 0.100;
    i1 = 0.1;
    i2 = 0.1;
    % Definitions of Parameters needed to represent system , currently with stationary body
    p1 = m1*lc1*lc1 + m2*lc1*lc1 + i1;
    p2 = m2*lc2*lc2 + i2;
    p3 = m2*l1*lc2;
    p4 = m1*lc1*m2*l1;
    p5 = m2*lc2;

    % Dynamics Equations derived using the Lagrangian Method, currently for stationary body
    D = [p1 + p2 + 2*p3*cos(t2) , p2 + p3*cos(t2); p2 + p3*cos(t2) , p2];

    C = [-p3*sin(t2)*t2d , -p3*sin(t2)*t2d - p3*sin(t2)*t1d; p3*sin(t2)*t1d , 0];

    G = [-p4*(-9.81)*cos(t1) - p5*(-9.81)*cos(t1 + t2) ; -p5*(-9.81)*cos(t1 + t2)];

    qdd = inv(D)*tau-inv(D)*C*qd - inv(D)*G;
end
```

B) PyBullet Code

```
# Walking Trajectory Generated using MATLAB
walk_x = [0.0122,-0.0116,-0.011,-0.0104,-0.0098,-0.0092,-0.0086,-0.008, ... ]
walk_y = [-0.020221,-0.022388,-0.024077,-0.025346,-0.026247,-0.026834, ...  ]

while(1):
        # Inverse Kinematics Calculations
        xpos = walk_x[idx]
        ypos = walk_y[idx]
        # a1, a2 are joint lengths, q1, q2 are joint angles
        q2 = -math.acos((xpos*xpos + ypos*ypos - a1*a1 - a2*a2)/(2*a1*a2))
        q1 = math.atan2(ypos, xpos) + math.atan2(a2*math.sin(q2), a1 + a2*math.cos(q2))

        # Update simulation, pause for visual playback to appear to be realtime
        stepSimulation()
        time.sleep(1. / 240.)

        # Set joint motor positions, will be changed to torque control as project progresses

        # Zero force is applied to gantry "joint" so that it moves freely along vertical bar
        p.setJointMotorControl2(stand, 0, p.POSITION_CONTROL, cpos, force=0)
        # For now, max force in Nm is applied using the built in pybullet position controller
        p.setJointMotorControl2(stand, 1, p.POSITION_CONTROL, upper_zero_offset-q1, force=27)
        p.setJointMotorControl2(stand, 2, p.POSITION_CONTROL, lower_zero_offset-q2, force=27)
        stateL = p.getJointState(stand,2)
```