

Spring 2015

Modular Biped Robotic Base

Lawrence T. Chiavaroli
The University Of Akron, ltc8@zips.uakron.edu

Andrew J. Forchione
The University Of Akron, ajf53@zips.uakron.edu

Wesley A. Miller
The University Of Akron, wam13@zips.uakron.edu

Joseph M. Drockton
The University Of Akron, jmd115@zips.uakron.edu

Please take a moment to share how this work helps you [through this survey](#). Your feedback will be important as we plan further development of our repository.

Follow this and additional works at: http://ideaexchange.uakron.edu/honors_research_projects



Part of the [Biomedical Commons](#), and the [Electrical and Electronics Commons](#)

Recommended Citation

Chiavaroli, Lawrence T.; Forchione, Andrew J.; Miller, Wesley A.; and Drockton, Joseph M., "Modular Biped Robotic Base" (2015). *Honors Research Projects*. 115.

http://ideaexchange.uakron.edu/honors_research_projects/115

This Honors Research Project is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.

Modular Biped Robotic Base

Senior Project Final Report

Design Team 13

Larry Chiavaroli, Electrical Engineering

Joseph Drockton, Electrical and Biomedical Engineering

Andrew Forchione, Electrical Engineering

Wesley Miller, Electrical Engineering

Advisor: Dr. Nathan Ida

April 22nd, 2015



**Department of Electrical and Computer Engineering
The University of Akron
Akron, OH 44325**

Table of Contents

LIST OF FIGURES	4
LIST OF TABLES	5
ABSTRACT	6
INTRODUCTION	6
Need	6
Objective	6
Background	7
Marketing Requirements	9
Engineering Requirements	9
Objective Tree	11
ACCEPTED TECHNICAL DESIGN.....	12
Engineering/Marketing Matrix.....	12
Mechanical Design	13
Prototyping.....	13
Static Model & Torque Simulation.....	16
Project Comparison	21
Hardware	23
Level 0 Block Diagram	23
Level 0 Functional Decomposition	23
Level 1 Block Diagram	24
Level 1 Functional Decomposition	24
Master Controller	27
Actuators	27
Motor Drivers.....	28
Limit Switches	29
Sensors	30
Software Design	33
Overview	33
C# Application	33
Master Controller	35

Slave Controller	36
Controller Pseudo-Code.....	36
Driver Pseudo-Code.....	37
Controls	38
Controller Concepts	38
System Modelling	39
BUDGET	43
Proposed Bill of Materials.....	43
Mechanical Components.....	43
Electronic Components	44
PROJECT SCHEDULE.....	45
Design Gant Chart.....	45
Implementation Gantt Chart.....	46
REFERENCES	48
APPENDICES	50
Appendix A: MATLAB Code and Results for Climbing Stairs	50
Appendix B: Data Sheets	52

LIST OF FIGURES

Figure 1. Hierarchical Display of Project Objectives	11
Figure 2. Prototype Leg in Testing	13
Figure 3. CAD Model of Prototype Leg Segment	13
Figure 4. Final Leg Assembly.....	14
Figure 5. Degrees of freedom	15
Figure 6. Static Model of Robot from Hip to Knee	17
Figure 7. Static Model Showing Forces Acting On Robot Hip to Ankle	18
Figure 8. Angles used for Static Model of Robotic Leg from Hip to Ankle	18
Figure 9. Lengths Used for Static Model of Robotic Leg from Hip to Ankle.....	18
Figure 10. Kadaba Walking Hip Angle	19
Figure 11. Kadaba Walking Knee Angle.....	19
Figure 12. Kadaba Stair Climbing Hip Angle	19
Figure 13. Kadaba Stair Climbing Knee Angle.....	19
Figure 14. MATLAB Walking Hip Angle.....	20
Figure 15. MATLAB Walking Knee Angle	20
Figure 16. MATLAB Stair Climbing Hip Angle.....	20
Figure 17. MATLAB Stair Climbing Knee Angle	20
Figure 18. MATLAB Walking Hip to Knee Torque	21
Figure 19. MATLAB Walking Total Hip Torque	21
Figure 20. Level 0 Block Diagram; System Inputs and Outputs	23
Figure 21. Level 1 Block Diagram; System Architecture.....	24
Figure 22. Master/Slave Relationship.....	29
Figure 23. MPU 9150 Motion Processing Unit and Interfacing Circuit.....	30
Figure 24: Slave Schematic	31
Figure 25: PCB Layout of Slave Circuitry	32
Figure 26: Example of Computer Interface GUI	35
Figure 27. Control System Block Diagram.....	38
Figure 28. Time Series of Inverted Pendulum Model.....	39
Figure 29. Inverted Pendulum with Variables Defined	40
Figure 30: Desired output (θ) waveform for compensated system.	42

LIST OF TABLES

Table 1: Marketing Requirements Revised After Preliminary Research and Calculations.....	9
Table 2: Engineering Requirements Revised After Preliminary Research and Calculations.....	9
Table 3. Engineering Requirements vs. Marketing Requirements Matrix Comparison.....	12
Table 4. Competitive Benchmark Matrix	22
Table 5. Functional Decomposition of Level 0 Block Diagram.....	23
Table 6. Functional Decomposition of Level 1 Block Diagram.....	24
Table 7. Controller Trade-Off Matrix.....	27
Table 8. Actuator Trade-Off Matrix	28
Table 9: Verb Byte Layout	34
Table 10: Number Byte Layout	34
Table 11: Definition of Initial Conditions	40
Table 12: Mechanical Component BOM.....	43
Table 13: Electrical Component BOM	44

ABSTRACT

This report contains the final developments and research involved with the modular biped robotic base. A need was first identified in 2011 when President Obama announced the National Robotics Initiative, an initiative focused on the funding of robotic development to work alongside or cooperatively with humans. This scope of this project concerns building a robotic base modeled after human legs and hips, capable of interfacing with future modular subsystems depending on what tasks are trying to be accomplished. Firstly, a mathematical torque simulation of the hip, knee, and ankle joints was developed in MATLAB. Using this information, complimentary actuators and driver circuitry were selected. A 3-D model of the leg and hip structure was drawn and simulated in SOLIDWORKS. Communication between the motors and the master controller was developed to provide precise control over each individual motor. After individual motor testing, a leg model was assembled and troubleshooting took place to determine proper alignment and placement of position sensors. The legs and hips were then fully integrated. A successful model was achieved capable of walking with full integration with subsystems of various types. (LC)

INTRODUCTION

Need

“In June 2011, U.S. President Obama announced the National Robotics Initiative, a \$70 million effort to fund the development of robots ‘that work beside, or cooperatively with people.’” (Robots Like Us). NASA, NSF, and other agencies are working in joint cooperation with universities and corporations to develop mobile robotic systems (nsf.gov). A modular, mobile robotic base is needed to expedite the development of full robotic systems for a wide range of applications. (AF)

Objective

The goal of this project is to design and prototype a modular, biped robotic base capable of integrating with other modular subsystems. The base system must be able to receive predefined commands using a standard communication protocol in order to easily integrate with other robotics (i.e. arms, eyes, etc.). The mobile robotic base must be a stable platform and provide a means of locomotion for many externally developed subsystems. Interacting with humans is paramount, and in order to do so the base must be capable of traversing urban terrain including steps and ramps while remaining balanced. Power, control, and sensory feedback signals of an overall system that incorporates the modular base should be obtained from a subsystem other than the base itself. (AF & JD)

Background

The design of a biped robotic base consists of a mechanical structure capable of human-like locomotion integrated with sensors, actuators, and control systems. The implementation of a biped robotic base relies on control structures, algorithms, and programs used to perform overall stability, disciplined movements, and common functional motions.

The mechanical structure of a biped robot consists of rigid materials arranged in such a way as to imitate the human body in terms of function and appearance. It is common to see joints similar to those of humans, like the ball and socket, and hinged joints. Servo motors often are used for actuators, although pneumatic actuators are being used to better emulate fluid limb motion of humans. The KHR-1, 2, and 3 developed at the Korea Institute of Science and Technology all have 12 degrees of freedom in their legs (Ill-Woo Park). These three robots were built to study biped walking and use brushed DC motors with reduction gears to produce the torque necessary to move the mechanical members. Ill-Woo Park states the reason for using brushed DC motors as opposed to brushless DC motor is the ability of brush-type motors to tolerate thermal stresses resulting from harsh driving conditions like high speed and high torque. Lightweight and strong materials are used in the mechanical design to reduce energy needed for locomotion while keeping the base stable and robust. Aluminum alloy was chosen for KHR-3, however Ill-Woo Park recommend the use of high strength steel. The use of steel along with use of planetary gear sets and other gear reductions are recommended to minimize deflection of the limbs and increase determinacy/minimize error of limb position and actuator output.

The mechanical design accounts for only one aspect of biped robots; a sensory feedback network, control processor, and control algorithms are also needed. Onboard control systems are used for command processing from the user along with continuously calculating and adjusting actuator output to maintain stability. Hernandez-Santos, Soto, and Rodriguez detail a novel method for dynamic modeling of a humanoid robot. Researchers and engineers are able to design components in SolidWorks and directly import the system into SimMechanics toolbox for MATLAB and obtain the equations of motion quickly, accurately, and efficiently (Hernandez-Santos, C., R. Soto, and E. Rodriguez). Simulations results and equations of motion are useful for developing control algorithms for moving the robotic base. Kim, J.Y. Lee, and J.J. Lee describe “A Balance Control Strategy of a Walking Biped Robot in an Externally Applied Force”, where four actions need to take place in order to ensure balance during walking (Yeoun-Jae Kim, Joon-Yong Lee, and Ju-Jang Lee). These methods will need to be taken into consideration when designing the control system for the robot.

After the mechanical structure is designed and actuators respond to commands to make the legs move, another primary function of the robot is the ability to respond properly to external disturbances. Ferreira, Cristostomo, and Coimbra discuss common motion control techniques for

biped robots involving calculations of the zero-moment point, or ZMP, through signals received from force sensors on the ground engaging planes of the machine. Research and development is being done on many approaches to compute the ZMP, including; genetic algorithms, wavelet networks, neural-fuzzy logic (NF), and support vector regression (SVR). These methods of computing are used to process the information generated by the sensors quickly in order to keep the robot stable about the forwards/backwards, or sagittal, directions. “The SVR and NF controllers exhibit similar stability, but the SVR controller runs about 50 times faster” (Ferreira, Cristostomo, and Coimbra). ZMP computing and SVR stabilization techniques will need to be considered during the design of the stability control system for the mobile robotic base.

The final objective is the achievement of a modular design of both mechanical systems and software. Since this project’s primary focus is the design and control of legs, less focus will be placed on the final software portability. Although future senior design groups or even cliental could implement whatever plug and play modules desired, ideally the following development strategy would be used. A hierarchical software system should be developed. A ‘brain’ module will resolve conflicts between modules, control total system movement, and maintain seamless system flow. A standard plug and play mechanical mounting and electrical power/communication must be developed to suit a wide range of auxiliary robots. Current technology can be improved through assigning unique I.D.s to attachments. This improvement would allow the brain unit to automatically account for change in center of gravity, moments of inertia and other behaviors. The communication protocol and connection must allow for real time data to be used in the control system along with ease of integrating distal sensors (Taira, Kamata, and Yamasaki). Basic function based software architecture for bipedal robots can be found in invention disclosure from Sony Corporation patent US 6961640 B2 (2005). (AF & JD)

Marketing Requirements

Table 1: Marketing Requirements Revised After Preliminary Research and Calculations

Marketing Requirements	
1.	Mechanical and electrical plug and play compatibility.
2.	Capable of stably carrying a load attachment.
3.	Simple assembly and manufacturing.
4.	Open source hardware and software development.
5.	System must have low cost.

The marketing objectives shown in Table 1 explain the need for the robot. The robot must be mechanically and electrically plug and play compatible. This will enable the robot to accept additional attachments designed by future developers, such as hobbyists or universities. Since attachments will be added to the robot, the robot needs to be able to carry the weight of these new attachments. The legs will be able to carry a weight up to and including a weight equivalent to their own weight. Working harmoniously with the idea of future development, the robot should have intuitive software that enables changes to the basic functionality of the robot without an in depth knowledge of the code. Ideally, open source hardware and software development would be utilized in order to further achieve the goals of more minds contributing to the advancement of this technology and improve robustness and functionality. Finally, the robot must have a low cost in order to make it accessible to a wide range of users such as engineers, hobbyists, and future entrepreneurs. (LC, JD, DF, & WM)

Engineering Requirements

Table 2 explains how each marketing requirement relates to the engineering requirements and gives the necessary justification. (LC, JD, DF, & WM)

Table 2: Engineering Requirements Revised After Preliminary Research and Calculations

Marketing Requirements	Engineering Requirements	Justification
2,3,5	Mechanical frame fabricated from Polylactic Acid.	Frame material must be commonly accessible, easily machined, rigid, lightweight, and cost effective.
2	Hip actuator needs to provide torque equal to 43.34 kg-cm or greater.	The hip motor will be lifting weight extended over longer lengths than any other joints. Hip actuator torque requirement considers dynamic motion, 20% practical speed and motion allowance, and a 100% safety net allowance.

2,3	High torque DC geared motors must provide twice the max torque required by the largest moment at a joint. (43.34 kg-cm)	The actuators need to respond quickly in order to keep the system stable under all conditions while responding at reasonable speeds. Additionally, the actuators need to be small and lightweight
1,4	The modular legs should accept commands using a serial protocol such as RS-232.	Communication between all subsystems is vital. A high data rate serial protocol is necessary.
2	ZMP or a similar algorithm should be developed for controlling the stability of the modular legs.	Obviously, the legs need to be able to be balanced under all operating conditions.
1,2,3,4,5	Production cost should not exceed \$1200.00	This is based on estimated cost of construction using currently specified materials.

Objective Tree

The objective tree shown in Figure 1 provides a high level overview of the robot's functionality. Four main categories modular, locomotion, joint mobility, and balance shown in blue describe the primary goals of this project. Below each main category are a number of requirements that need to be met in order to achieve the above primary goal. (LC, JD, DF, & WM)



Figure 1. Hierarchical Display of Project Objectives

ACCEPTED TECHNICAL DESIGN

Engineering/Marketing Matrix

Table 3. Engineering Requirements vs. Marketing Requirements Matrix Comparison

	Lightweight & Rigid Frame	Actuator Torque Level	Electric Motor Usage	Serial Command Usage	Stability Algorithm	
Sign of Correlation	±	+	+	+	+	+
1) Plug-Play Compatible	+		↑↑	↑↑	↑↑	↑
2) Capable of Carrying Leg Equivalent Load	+	↑↑	↑↑	↑		↑
3) Capable of Flat Ground Traversal	+		↑			↑↑
4) Capable of Inclined Plane Traversal	+		↑			↑↑
5) Capable of Climbing Stairs	+	↑	↑↑			↑↑
6) Stable During Disturbances	+		↑			↑↑
7) Capable of Motion at Various Speeds	+		↑	↑↑		
8) Capable of Measuring CoG Movement	+			↑		↑↑
9) Cost Under \$1500	+	↓↓	↓↓	↓↓		↓↓

Table 3 shown above shows correlation between marketing and engineering requirements. A strong positive correlation is indicated by two up arrows, weak positive correlation is represented by one up arrow. A strong negative correlation is indicated by two down arrows, weak negative correlation is represented by one down arrow. As seen in by the abundance of up arrows, most of the marketing objectives have a positive correlation, meaning the marketing requirements and engineering requirements are working in tandem. (LC, JD, WM, AF)

Mechanical Design

Prototyping

Some initial questions that needed answered were types of joints, joint locations, degrees of freedom, necessary torque, and motor type. From the prototype shown in Figure 2, the knee joint was chosen to be the test subject of the above variables because of its accessibility and ease of design. The prototype provided much valuable insight. It proved that a medium sized stepper motor could not provide the holding torque necessary to support the weight of rest of the leg, or enough torque to move the limb.

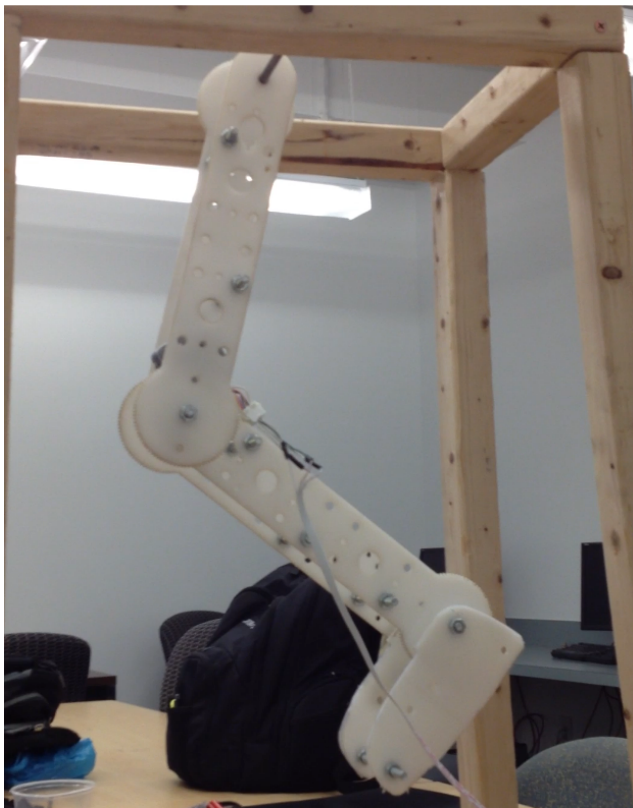


Figure 2. Prototype Leg in Testing

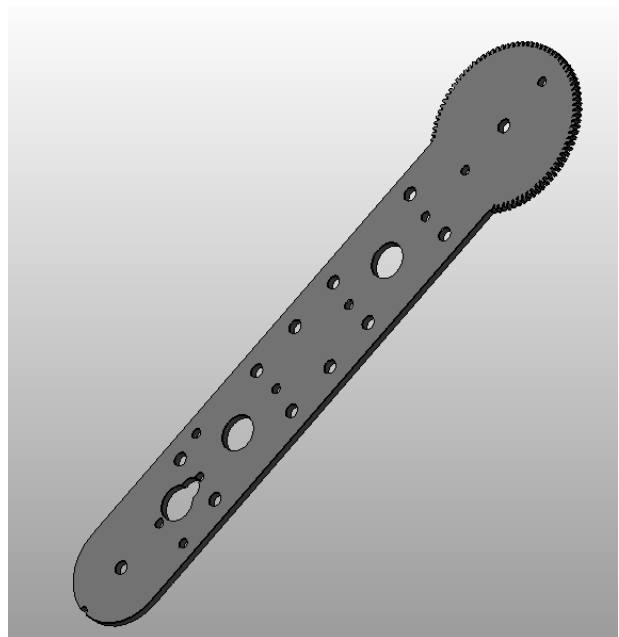


Figure 3. CAD Model of Prototype Leg Segment

Figure 2 is comprised of four pieces of the drawing of Figure 3. The goal was to achieve moving the knee joint that moved to a given position and make it hold the given position until it was instructed to move again. The reasoning for choosing the knee joint was because it was the easiest to design and build. Figure 3 was drawn up in Autodesk inventor fusion then four pieces were cut out of a sheet of plastic with a 2D water-jet CNC machine. After water jetting the parts, nuts and bolts were used to build the prototype leg. At the knee joint a large gear was cut directly

into the leg that was three times larger than the gear on the stepper motor to supply more torque. The stepper motor was controlled through an Arduino Mega 2560 and position was given by a 3d printing program that told the motors to move in ranges from 1mm, 10mm, or 100 mm. Figure 2 shows the leg holding a position using a holding torque supplied to the stepper motor. This appeared to work great until an additional motor was put in the foot placement because it was significantly heavier and the stepper motor at the knee joint was not strong enough to move past an angle smaller than 25 degrees.

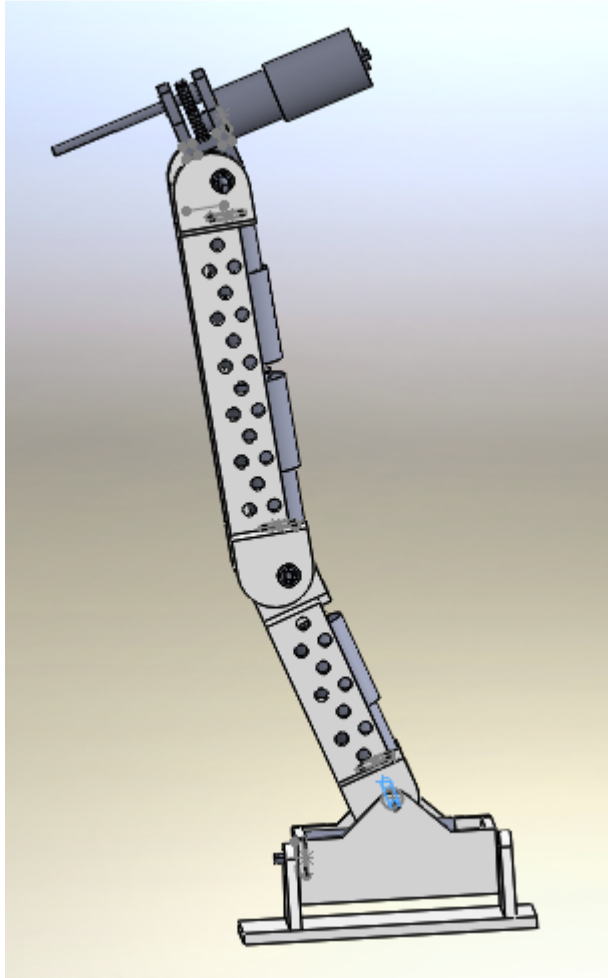


Figure 4. Final Leg Assembly

Figure 4 shows the final assembly of the legs without the base. The motors are geared DC motors that fit inside U channels that comprise the leg. There are 3 degrees of freedom in the hip, one degree of freedom in the knee, and two degrees of freedom in the ankle. Various simulations were used via MATLAB and SolidWorks to insure that these motors could move the leg the way it is intended to be moved. After changing the material properties in SolidWorks from aluminum to plastic, the overall weight of the assembly was reduced to nearly half. The reduction of weight allowed us to satisfy the engineering requirement of having twice the torque of the largest moment provided by the leg.

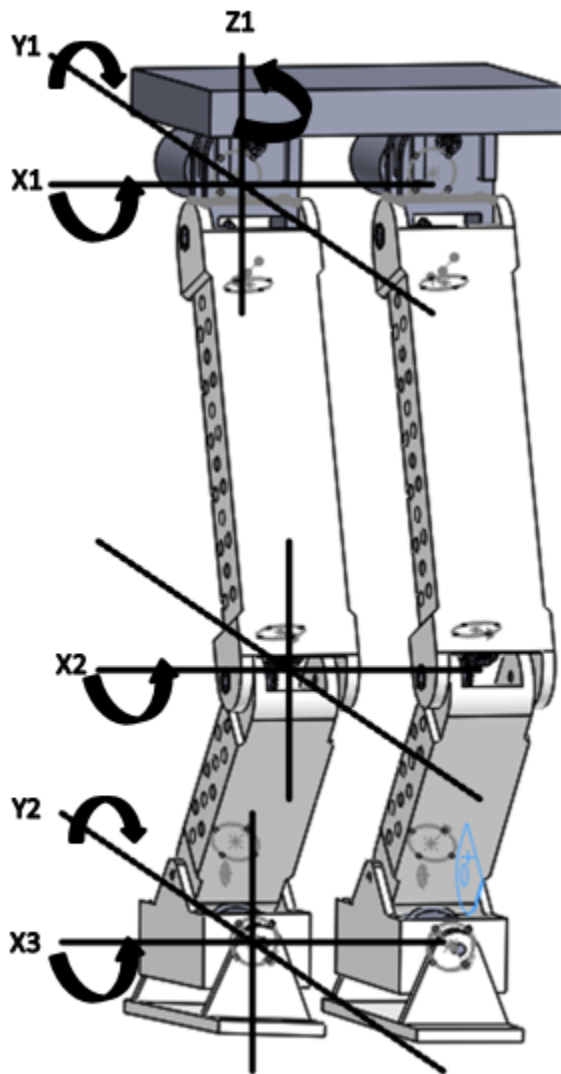


Figure 5. Degrees of freedom

Figure 5 shows the degrees of freedom of each robotic leg in the biped assembly. There are two degrees of freedom at the ankle, one degree of freedom at the knee, and three degrees of freedom at the hip for a sum total of six degrees of freedom for each leg. The number of labeled axes indicate the degrees of freedom used at that joint. X1, X2, and X3 motors are needed for forward motion. Y1 and Y2 are needed to shift the center of mass to one leg during the walking process as well as being able to side-step in the event of a disturbance. The Z1 motor is needed as an aid in keeping the structure stable when a lateral disturbance is introduced. This way the leg is able to be pointed in the falling direction and the X1, X2, and X3 motors will be used to move the foot out to catch the structure when falling. The Y1 motors are used to step laterally, and thus enable faster changes in direction and eliminate extra rotational momentum introduced by alternate methods of movement. If the Y1 motors were not present and a disturbance came in

from the side, the leg would be raised up by the X1, X2, and X3 motors then the leg would be spun by the Z1 motors. This would introduce an extra rotational force due to the weight of the leg being at a further distance from the body.

The system would be capable of walking with fewer motors but the movement and compensation flexibility would be much more restricted. The extra motors at the hip joint will allow for disturbances from the sides. This too could be accomplished by two motors at the hip joint but adding a third motor will allow for extra ways of moving to a point. This complicates the software, because there are multiple ways to get to a point, but will allow for the system to reach the desired location more quickly, which should reduce the risk of the system falling. It was decided to have two degrees of freedom at the ankles in the X and Y direction. This will allow the foot to be placed flat on a surface during walking or recovery from a disturbance. The Z axis was not incorporated in the ankle because the foot will have no reason to rotate about the Z axis while both feet are standing on a surface. Since hips have a Z rotation, this will allow the foot to twist if it is needed while the foot is off the surface.

Static Model & Torque Simulation

Actuator selection for the modular biped robotic base was determined based on the reasons previously discussed in Actuators section. The requirements for the actuators were calculated using an overview of the purpose of the robot and its scale. Important specifications during motor selection were torque, speed, weight, size, input voltage, input power, and shaft type. In order to maintain modularity of the robot all motors in the legs were the same type and size.

To specify a value for the torque needed by the motors, a static model of one half of the robot was developed. The torque required from each motor used in the robot was largely dependent on the scale of the complete system. The static model for one leg was developed with attention to the hip joint. The moment at the hip was important because that is where the maximum torque moment would be during walking. Beginning at the hip joint and including the leg down to the knee, a simple static model was developed with point masses at strategically placed points on the thigh portion, as shown in Figure 6.

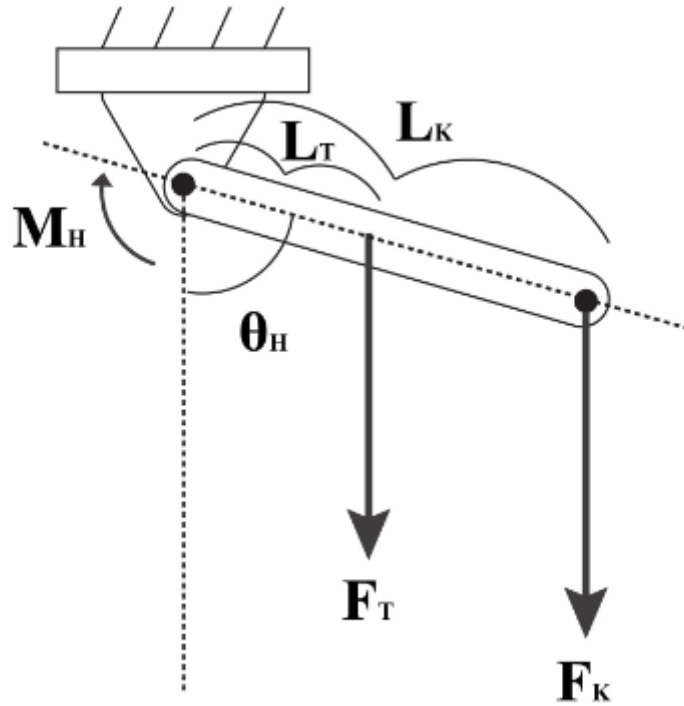


Figure 6. Static Model of Robot from Hip to Knee

The overall moment at the hip is dependent on the angle of deflection from the knee directly below the hip, which would result in a moment of zero. The equation governing this section of the leg is given as,

$$\dots \quad (1)$$

Clockwise rotation is defined to be positive.

Next, a static model was developed that included the section from hip to knee as well as a section from knee to ankle of one robotic leg, as shown in Figure 7, Figure 8, and Figure 9.

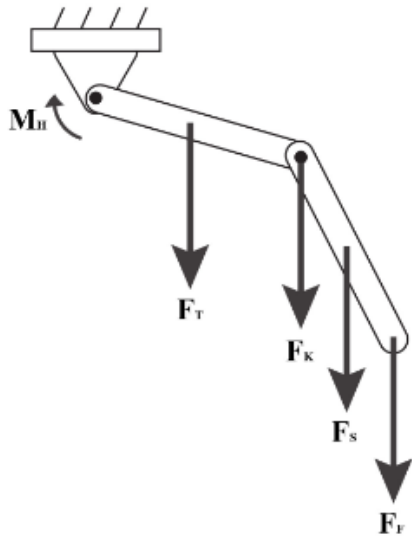


Figure 7. Static Model Showing Forces Acting On Robot Hip to Ankle

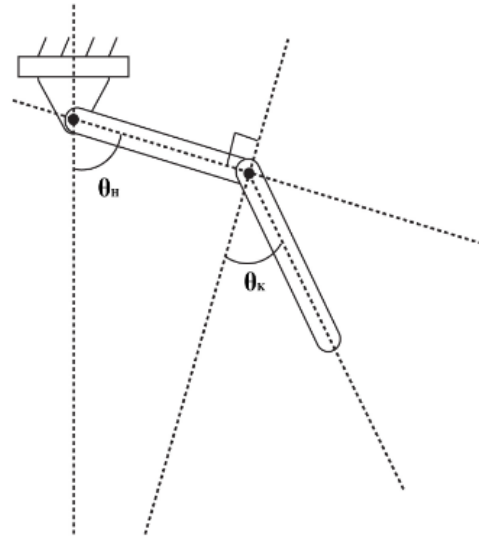


Figure 8. Angles used for Static Model of Robotic Leg from Hip to Ankle

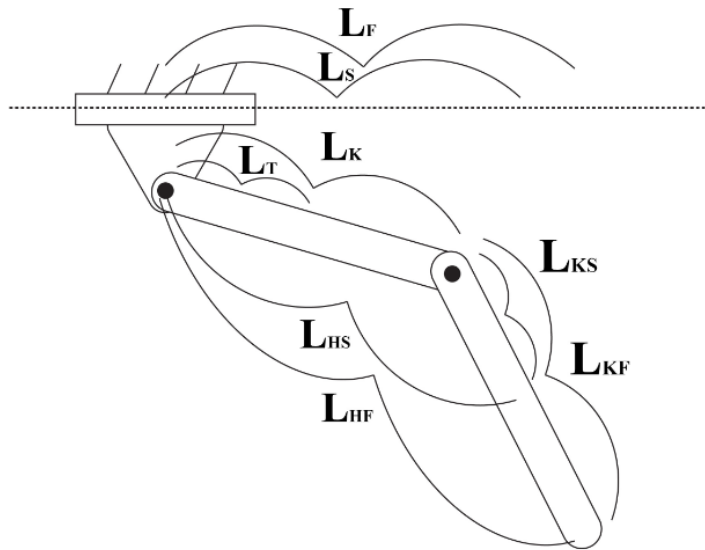


Figure 9. Lengths Used for Static Model of Robotic Leg from Hip to Ankle

This model takes into account the angle of the hip and angle of the knee while computing the moment at the hip. The governing equation for this system that includes the hip to the knee to the ankle joint for one leg and point masses in strategic places is given as

(2)

Considering the length of the foot compared to the length of other components was small, the foot was modeled as a point mass at the ankle joint.

After developing a static model for the moment at the hip, actual torque data was obtained from Kadaba's study which analyzed the hip and knee angles of human legs while walking. Since the robot's range of motion will mimic human gait, the actuators will need to be able to produce these required torques. This data is shown Figure 10 and Figure 11. All the following figures are all plotted versus the percentage of gait cycle from heel strike to heel strike, whether that human is simply walking or climbing stairs (Kadaba & Ramakrishnan, 1990).

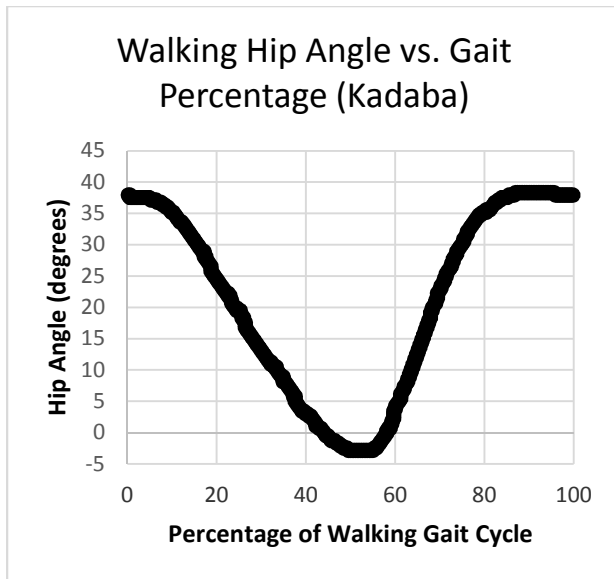


Figure 10. Kadaba Walking Hip Angle

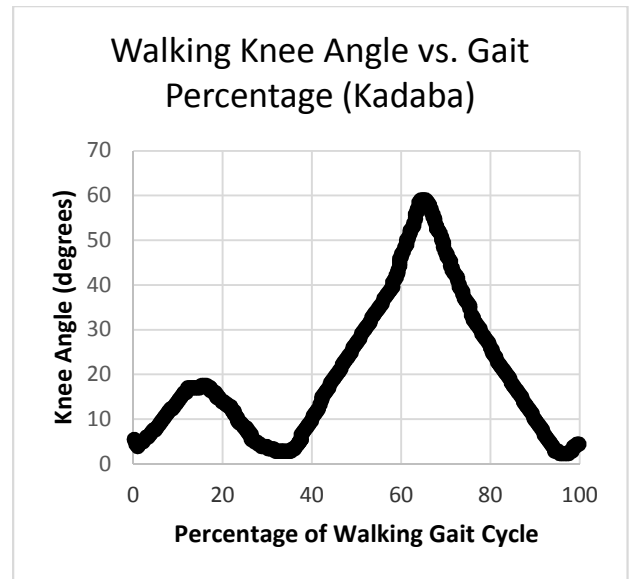


Figure 11. Kadaba Walking Knee Angle

From Kadaba, the hip and knee angles for a human leg while climbing stairs were also observed. These graphs are shown in Figure 12 and Figure 13 (Kadaba & Ramakrishnan, 1990).

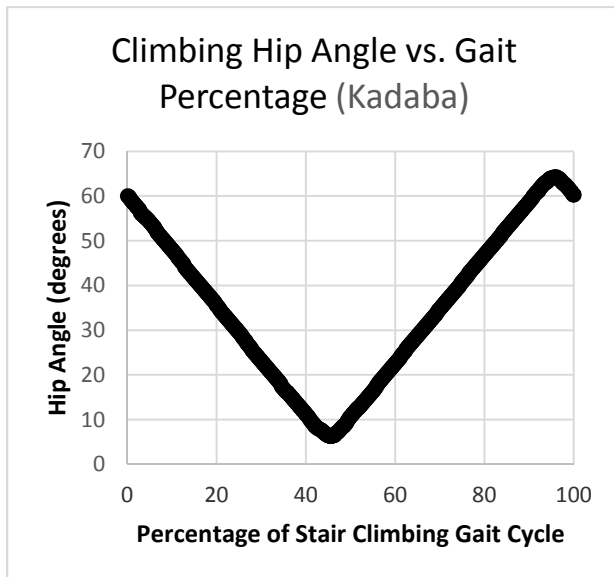


Figure 12. Kadaba Stair Climbing Hip Angle

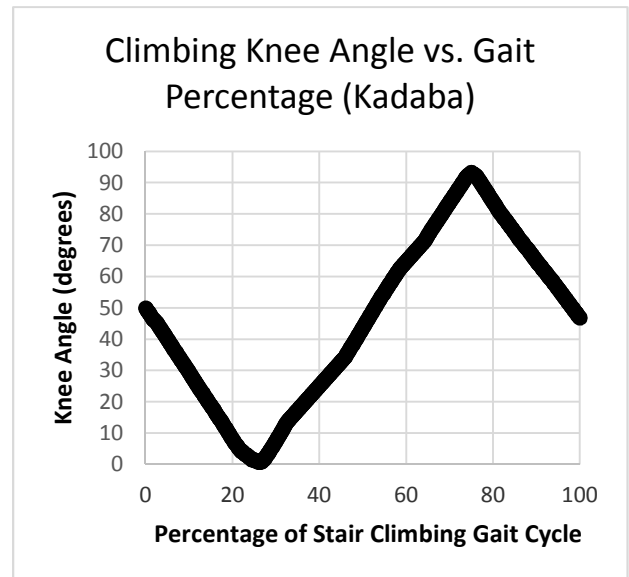


Figure 13. Kadaba Stair Climbing Knee Angle

The angles from Figure 10, Figure 11, Figure 12, and Figure 13 were inserted into the MATLAB static model simulation to ensure that the same angles used as an input to the code were identical at the output. These calculated angles are shown in Figure 14, Figure 15, Figure 16, and Figure 17.

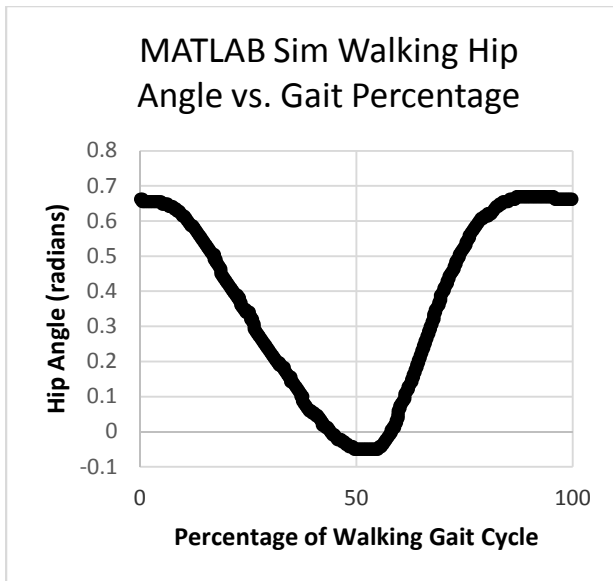


Figure 14. MATLAB Walking Hip Angle

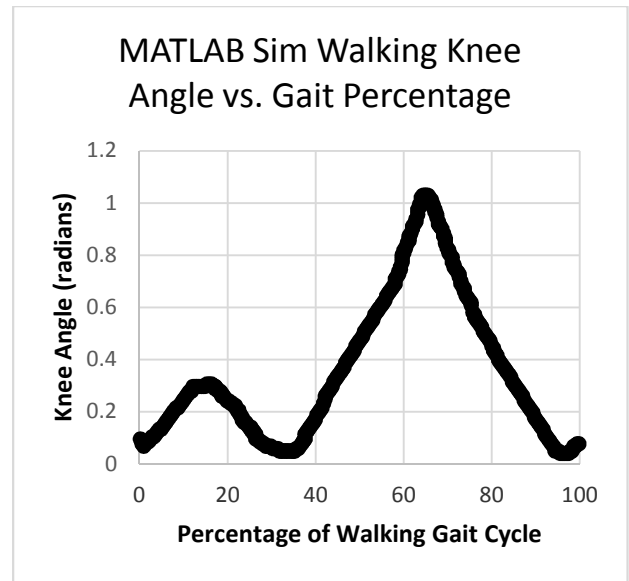


Figure 15. MATLAB Walking Knee Angle

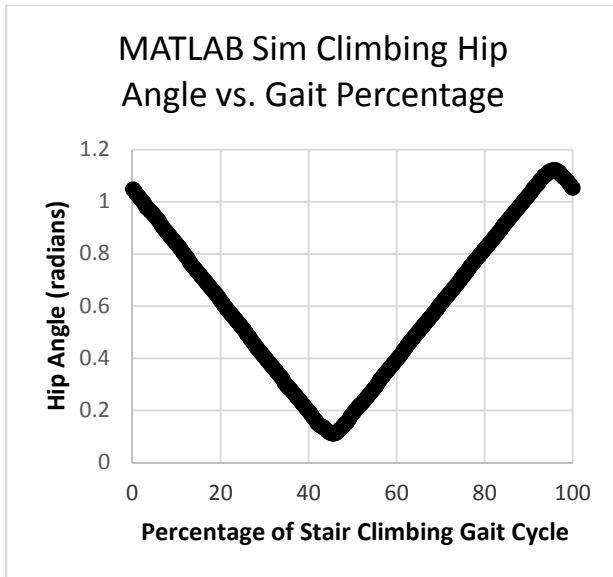


Figure 16. MATLAB Stair Climbing Hip Angle

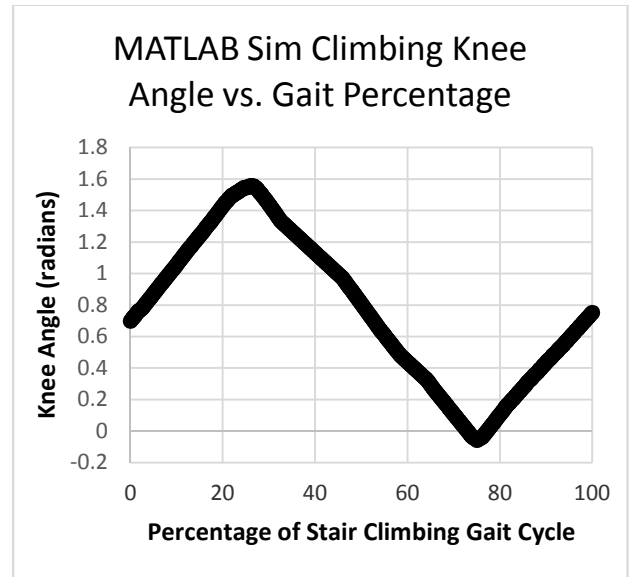


Figure 17. MATLAB Stair Climbing Knee Angle

When comparing the MATLAB simulation climbing knee angle graph to the Kadaba climbing knee angle graph it is easy to see that the MATLAB simulation graph is the inverse of the Kadaba climbing graph. This is because for the stairs simulation, Kadaba used clockwise rotation as a positive rotation, and the MATLAB simulation uses clockwise rotation as a negative rotation. However, it can still be determined that the two graphs are comparable as the MATLAB simulation is simply the negative of the other (Kadaba & Ramakrishnan, 1990).

For the torque calculations, MATLAB exported the data to excel and plotted that data against the percentage in the walking/climbing cycle vs. torque needed at that point. This data is shown in Figure 18 and Figure 19. (LC, JD, WM, AF)

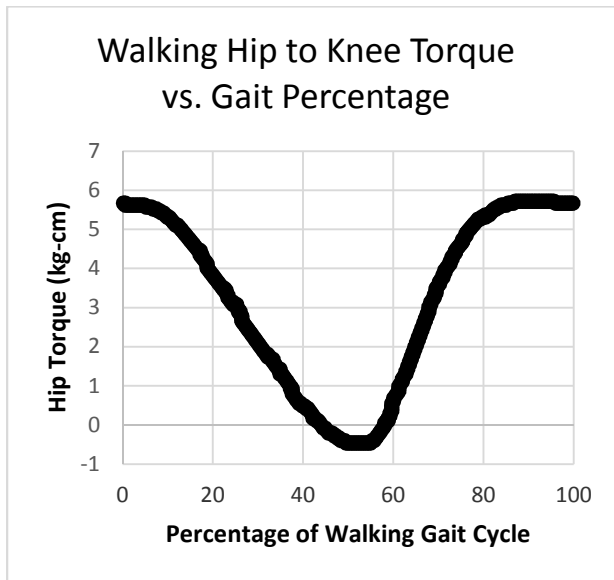


Figure 18. MATLAB Walking Hip to Knee Torque

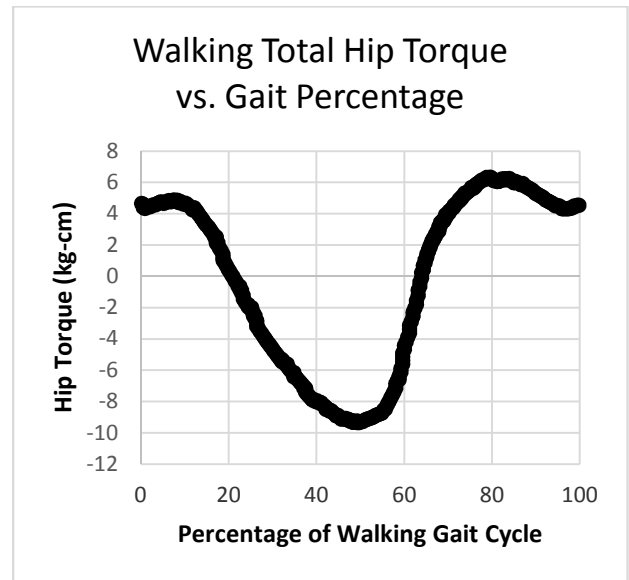


Figure 19. MATLAB Walking Total Hip Torque

Project Comparison

A very broad comparison of existing technology can be made to validate the design goals of this project. Comparing characteristics of existing bipeds will give insight into the reasons why the modular biped robotic base stands out among competing technology. Key factors including cost, modularity, stability, and actuator type are compared in Table 4 below. Akron Dynamic's SL1 is the name for the modular biped base detailed in this report.

Table 4. Competitive Benchmark Matrix

ROBOT	Akron Dynamics SL1	Boston-Dynamics Atlas	University of Tokyo's JSK HR3P-JSK Lab	Trossen Robotics Darwin	Honda Robotics ASIMO
DOF Total	10 +	28	10 +	20	57
DOF Legs	10	?	10	12	12
Cost	\$1,500	\$XX,000,000	\$40,000	\$16,000	\$1,000,000
Available to Public	✓	X	X	✓	X
Modularity	✓	✓	✓	X	X
Stability	✓	✓✓✓	✓✓	X	✓✓
Power	Tether	Tether	Tether	Battery	Battery
Speed	1 km/h	7 km/h	5 km/h	0.86 km/h	2.7 km/h
Actuators	Electric	Pneu/Hydraulic/Elec	Novel Electric	Electric	Electric

The most obvious difference between biped robots is the cost. Akron Dynamic’s SL1 low cost of \$1,000 allows for a wide base of interested research groups to develop their modular robotic subsystems that interact with the biped system. Having the input power provided by a tether reduces limitations that a battery pack would provide for additional subsystems that are attached. Also, having all electric actuators is more convenient since no pneumatic or hydraulic tanks or valves need to be carried. Having Akron Dynamics’ SL1 available to the public will allow for the development of subsystems to increase the overall value of the biped robotic base. (LC, JD, WM, AF)

Hardware

Level 0 Block Diagram

Figure 20 below shows the level 0 block diagram, highlighting the overall system inputs and outputs. Table 5 further defines system inputs and outputs shown in Figure 20. (LC, JD, WM, AF)

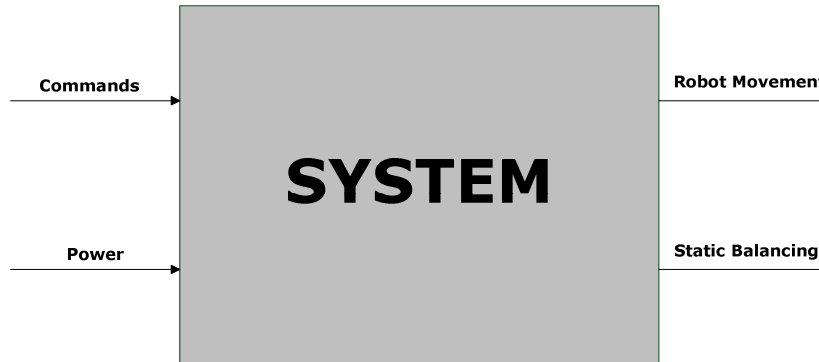


Figure 20. Level 0 Block Diagram; System Inputs and Outputs

Level 0 Functional Decomposition

Table 5. Functional Decomposition of Level 0 Block Diagram

<i>Module</i>	System
<i>Inputs</i>	<p>Commands: A general set of functional commands; such as, walk, stop, squat, turn, and climb stairs.</p> <p>Power: Energy must be supplied to run the actuators, drivers, and controllers.</p>
<i>Outputs</i>	<p>Robot Movement: System will execute the desired functional command.</p> <p>Static Balancing: System will remain balanced using inputs from sensors when no functional commands have been given.</p>
<i>Functionality</i>	<p>This module will accept input data from the user and then process and execute corrective means to remain balanced or complete functional commands.</p>

Level 1 Block Diagram

Figure 21 below shows the level 1 block diagram, highlighting the overall system inputs and outputs. Table 6 further defines system inputs and outputs shown in Figure 21.

(LC, JD, WM, AF)

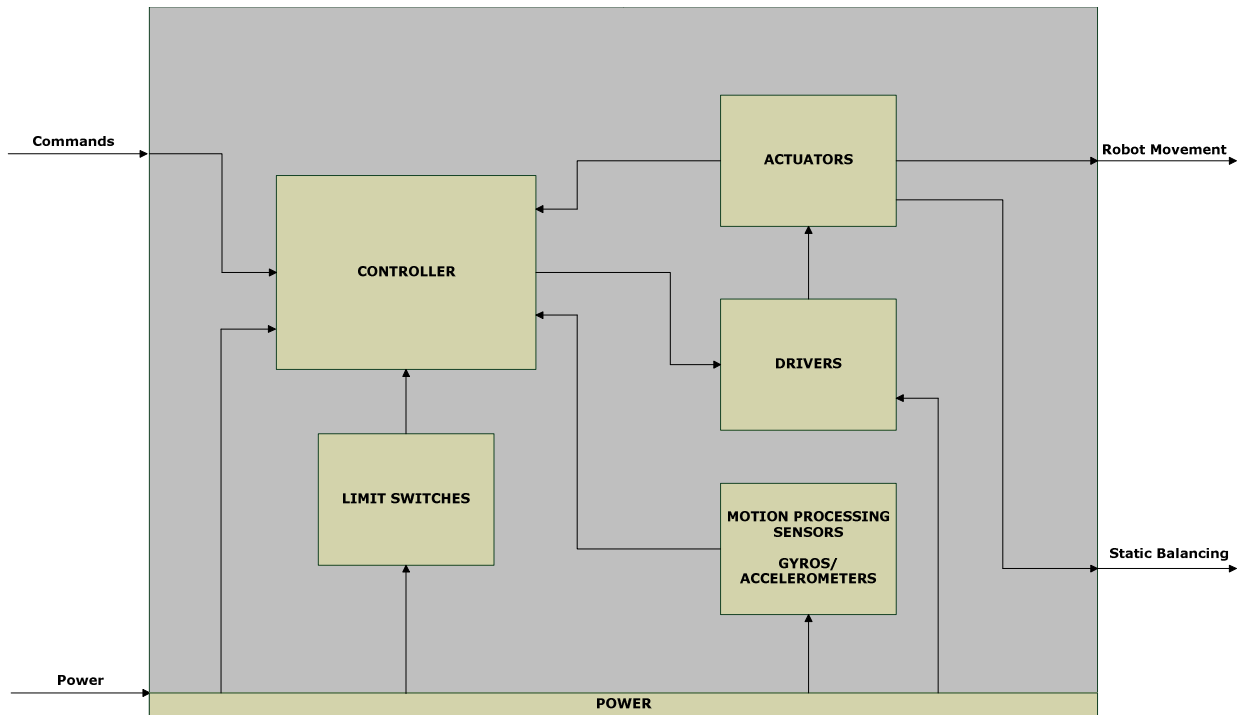


Figure 21. Level 1 Block Diagram; System Architecture

Level 1 Functional Decomposition

Table 6. Functional Decomposition of Level 1 Block Diagram

Module	Controller
Inputs	<p>Commands: A general set of functional commands; such as, walk, stop, squat, turn, and climb stairs.</p> <p>Power: Fed by 12 VDC source</p> <p>Actuator Feedback: Provide information about the position/velocity of the actuators</p> <p>Motion Processing Information: Includes data such as falling angle and</p>

	overall velocity of system as a whole.
<i>Outputs</i>	Driver Signal: System will execute the desired functional command. Static Balancing: System will remain balanced using inputs from sensors when no functional commands have been given.
<i>Functionality</i>	This module will accept input data from the user, actuator feedback, motion processing unit, and limit switches. It will perform computations on the inputs to provide an output to the actuator drivers.

<i>Module</i>	Limit Switches
<i>Inputs</i>	Power: 12 VDC
<i>Outputs</i>	Limit Signal: A signal to indicate if limb position maxima or minima are met.
<i>Functionality</i>	This module will detect when a limb meets its maximum position and send a signal to the controller.

<i>Module</i>	Motion Processing Unit
<i>Inputs</i>	Power: 5 VDC
<i>Outputs</i>	Motion Information: Data including velocity, and angle at a specific point of the robot.
<i>Functionality</i>	This module will measure and transmit motion data to the controller.

<i>Module</i>	Actuators
---------------	-----------

<i>Inputs</i>	Power: 12 VDC at 20A max Drive Signals: Signals used to make actuators move.
<i>Outputs</i>	Robot Movement and Static Balancing: Actuators will respond to the desired input commands from the slave drivers.
<i>Functionality</i>	The actuators will move based on drive signals from the drivers.

<i>Module</i>	Drivers
<i>Inputs</i>	Power: 12 VDC at 20A max Controller Commands
<i>Outputs</i>	Drive Signals: control power to actuator block.
<i>Functionality</i>	The driver will take in commands from the controller and convert those commands into drive signals for the motors.

Master Controller

Table 7. Controller Trade-Off Matrix

	Arduino Due	Intel Galileo	Tiva-C Launchpad	Connected Launchpad	Tiva-C	Mbed
Architecture	32-bit, ARM	32-bit, x86	32-bit, ARM	32-bit, ARM	32-bit, ARM	32-bit, ARM
Cost	\$49.95	\$79.00	\$12.99	\$19.99	\$19.99	\$49.95
Ease of Use	✓+	✓	✓-	✓-	✓-	✓-
Cross-Platform	✓	✓	✗	✗	✗	✓
Free IDE	✓	✓	✓	✓	✓	✓
Open Source	✓	✓	✓	✓	✓	✓
Max Clock Speed	84 MHz	400 MHz	80 MHz	120 MHz	120 MHz	100 MHz
Input Voltage	7-12V	7-15V	7-15V	7-15V	7-15V	7-15V
RISC or CISC	RISC	CISC	RISC	RISC	RISC	RISC
Debugging Capability	✓	✓	✓	✓	✓	✓
SRAM	96 kB	512 kB	32 kB	256 kB	256 kB	64 kB
EEPROM/Flash	512 kB	512 kB	256 kB	1 MB	1 MB	512 kB

Table 7 shows an overview of a small selection of microcontrollers available on the market. The matrix was broken down in to several categories and different trade-offs had to be taken into account to make a proper selection. After some discussion the Arduino Due was chosen due to its amount of ram, its reasonable price and its ease of use.

Actuators

The actuator selection focused on several factors such as cost, control, force/torque, and power while at the same time focusing on consistency and ease of implementation. Six actuators were selected for debate. The following chart, Table 8 was created to select the best actuator. The numbers one through five indicate the degree of each factor.

Table 8. Actuator Trade-Off Matrix

	COST	CONTROL	FORCE/TORQUE	POWER
± Correlation	-	+	+	+
Hydraulic	5	1	5	Hydraulic Pressure
Pneumatic	3	2	4	High Pressure Air
Linear Actuator	4	3	3	Electric Motor
Servo motor	4	5	2	Electric Motor
Stepper Motor	2	3	3	Electric Motor
DC Geared Motor	3	4	4	Electric Motor

Comparing the different actuator types, the DC geared motor has the benefits of required torque and ease of control at the expense of higher cost than the stepper motor. For these reasons, it was determined that the DC geared motor was the best choice of actuator for the robot. The DC geared motor is long and cylindrical, fitting into the mechanical structure of the legs. Using the torque simulation discussed above in the Static Model & Torque Simulation section, the DC geared motor's max torque, 138 kg-cm, more than triples the max torque, 38 kg-cm, calculated by the MATLAB walking and stair climbing torque simulations. (LC, JD, WM, AF)

Motor Drivers

The chosen motor for the project has a no-load current of 0.53 amps and a max stall current of 20 amps. The original intention was to drive the motors with the use of a commercially available H-bridge IC. However, 20 amp H-bridge integrated circuits are not available and driver boards are not available within our available budget. The solution to this problem is to use MOSFETs as the switches to the H-bridge. The MOSFETs are driven by dedicated MOSFET drivers which in turn are controlled by the PIC PWM output. The MOSFET driver enables very fast MOSFET switching in order to reduce internal heating and the possibility of shoot through current.

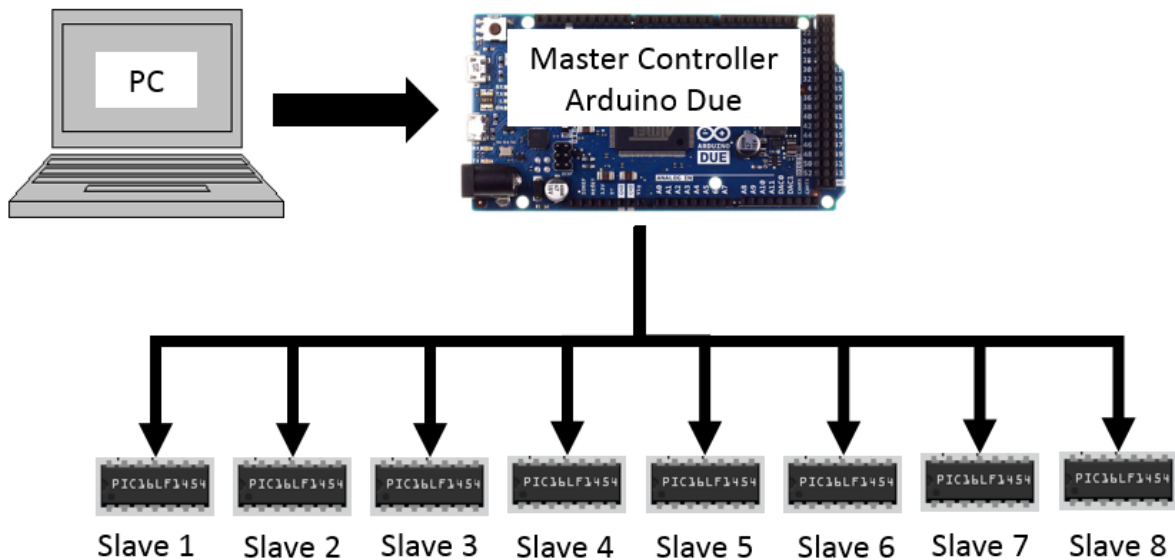


Figure 22. Master/Slave Relationship

Figure 22 shows the motion of the system will be achieved by a Master-slave relationship. Each motor will be assigned a slave PIC that will take commands from the master. Once user sends the master a command, the master controller then sends each slave what position it needs to move to. The slave then uses an analog voltage from the potentiometer and converts it to a digital signal for the slave. From this, the slave can determine the current joint angle and then calculate how far the motor needs to rotate to get to its destination angle. The joint position cannot accurately controlled due to the inertial nature of the system. Therefore the slave must be responsible for compensating through a PID controller. (LC, JD, WM, AF)

Limit Switches

Similar to human joints, the robot has a limited range of motion for each joint due to the physical construction. In order to prevent the robot from damaging itself, switches are needed shut down the motors in the case of joint over travel. Every motor should have two limit switches, one at its maximum limit and one at the minimum. These switches should never be tripped during normal operation. They may however be activated if a large external disturbance is applied, a software or hardware malfunction occurs, or during calibration (homing the motors). Momentary DC microswitches are being considered due to the small size, low cost, and low power.

Sensors

In order to achieve stability, the robot needs to be cognizant of many variables including the rotation of the base and translation of the base along with many others. These variables can be obtained through the use of potentiometers, gyroscopes, and accelerometers. InvenSense provides a clean solution to sensing rotation and linear acceleration in the MPU 9150. This sensor has three orthogonal gyroscopes and three orthogonal accelerometers along with a magnetometer and temperature sensor all in one IC package. The peripherals for implementing this chip were selected based on recommendations in the datasheet/application note. Figure 23 shows the circuit necessary to utilize the MPU in the robotic base.

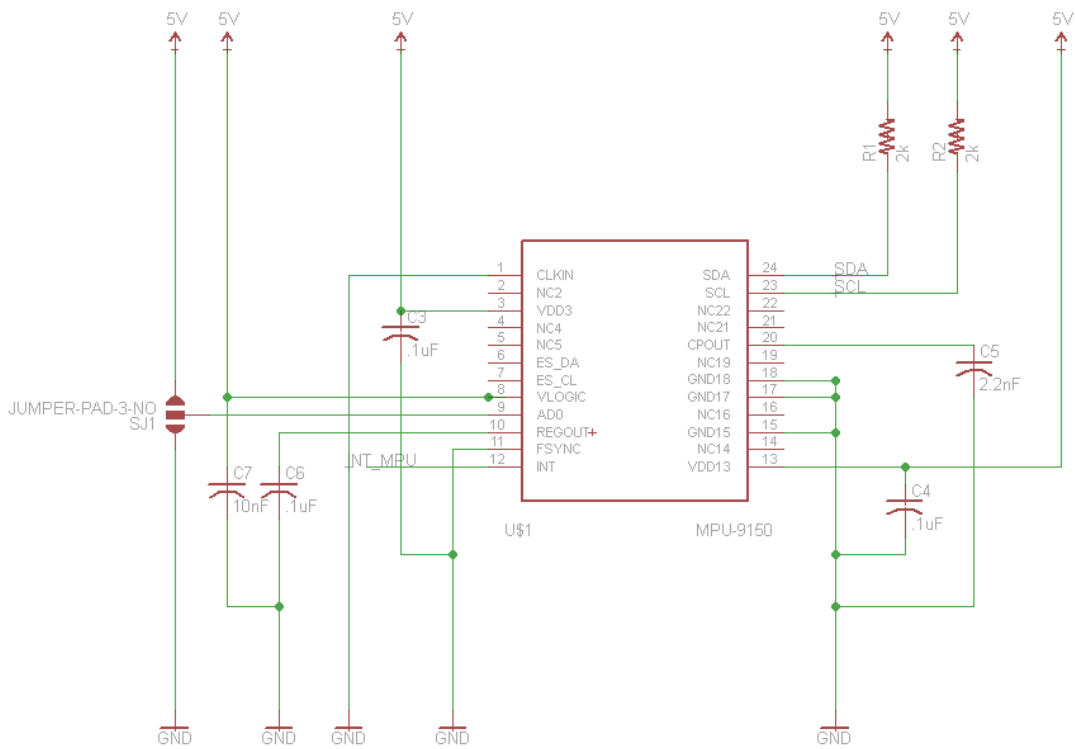


Figure 23. MPU 9150 Motion Processing Unit and Interfacing Circuit

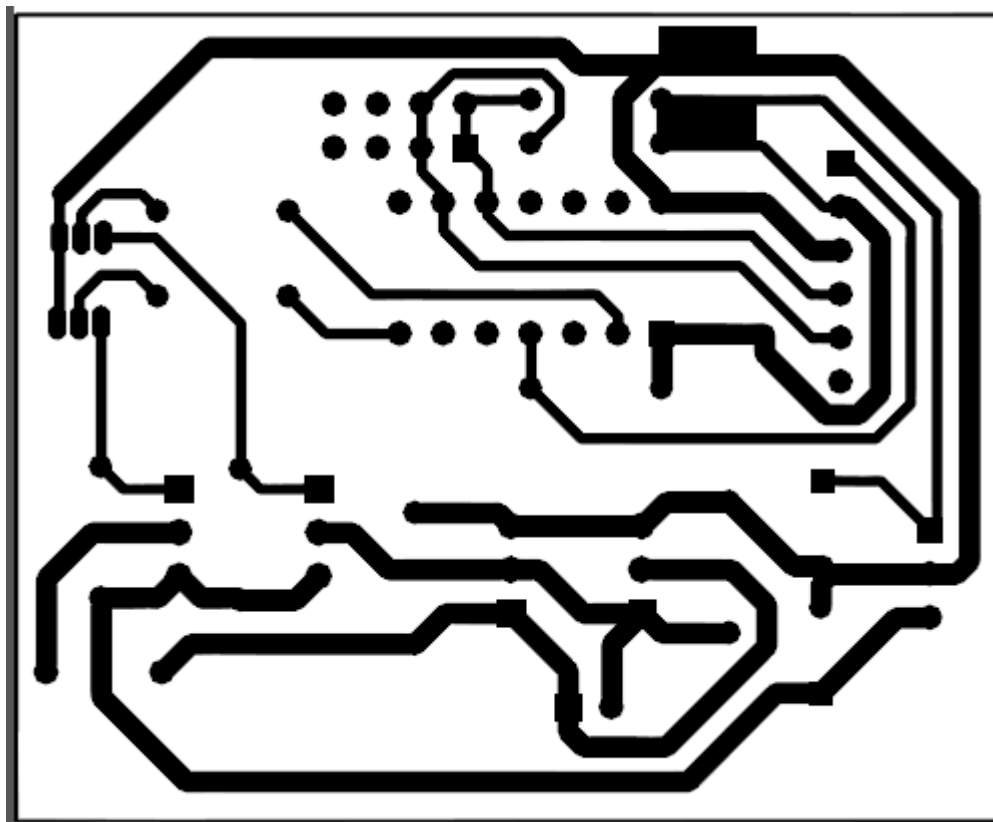


Figure 25: PCB Layout of Slave Circuitry

Figure 25 is the PCB layout for the slave schematic in Figure 24. This board is one sided with one removable jumper that is opened when programming the PIC and closed when the circuit is in operation.

Software Design

Overview

Software running on the modular biped base needed to be designed to work in harmony with the physical system to perform the movements necessary to meet marketing and engineering requirements.

Three pieces of software need to be designed; a C# application running on a nearby PC, master controller firmware running on an Arduino Due, and slave controller firmware running on a PIC16LF1454.

C# Application

The C# application will be created using Microsoft Visual Studio and will act as a user interface between the person controlling the robot and the robot's hardware. The application will have inputs like buttons and drop-down selection tools for users to interact with, and corresponding text to inform the user about their selections. When the user is ready to send commands to the robot they will press a send button that will package the necessary information to be sent through USB to the robot. As the information is sent to the robot it will also be logged on the PC in case the user needs to look back on what they sent.

The packet formation process will take in all user inputs and generate a two byte command that is sent to the robot serially via the selected COM port. These two bytes will be read serially on the Arduino Due's UART and processed accordingly. The UART supports full duplex communication and therefore, an acknowledge message can be sent back to the PC from the Arduino Due at any time that will inform the user of current activity.

The two byte command that is sent from PC to Arduino Due consists of two main pieces of information. The first byte is the verb byte shown in Table 9. The verb byte tells the robot which function to execute, whether it be stand, walk, or shuffle. The first 6 bits of the verb byte are allotted for which function to implement. This gives 64 possible verbs that can be sent to the robot. The last two bits of the verb byte are for requesting an acknowledgement from the robot. The second byte sent to the master is a number byte, shown in Table 10. The number byte will indicate N if the verb byte requires N. The PC user can request a full status report, a partial report, or no report. A full report will return all available information on the Arduino Due, including all position angles, IMU data, and communications data. A partial report will only return communications data from the Arduino Due to the PC. No report means no data will be communicated to the Arduino from the PC. An example of user interface is shown in Figure 26.

Verb Byte								
VB7	VB6	VB5	VB4	VB3	VB2	VB1	VB0	
Function Bits						ACK Bits		
Bits 7-2		Command				Bits 1-0		Acknowledge?
000000		No Function				00	No Acknowledge	
000001		Stand Upright				01	Partial Acknowledge	
000010		Squat				10	Full Acknowledge	
000011		Walk Forward N steps				11	Unused	
000100		Walk Backwards N steps						
000101		Turn Right N degrees						
000110		Turn Left N degrees						
000111		Walk Forward N inches						
001000		Walk Backwards N inches						
001001		Unused						
001010		Unused						
001011		Unused						
001100		Unused						
...		Unused						
111111		Unused						

Table 9: Verb Byte Layout

Number Byte								
VB7	VB6	VB5	VB4	VB3	VB2	VB1	VB0	
Number Bits								
Bits 7-0								
00000000		If Verb Byte requires N, this byte=N						
00000001								
00000010								
00000011								
00000100								
00000101								
00000110								
00000111								
00001000								
00001001								
00001010								
...								
11111111								

Table 10: Number Byte Layout

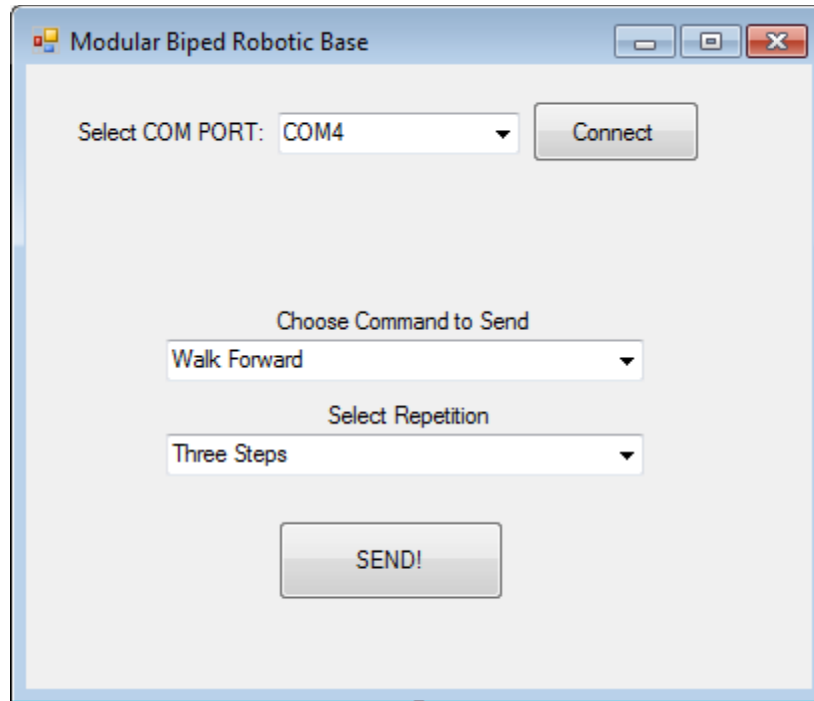


Figure 26: Example of Computer Interface GUI

Master Controller

The master controller is responsible for maintaining system stability and processing input commands from the PC. The master accepts two byte input commands on its UART and sends acknowledgement messages back to the PC. Also, the master generates commands that are sent to all 12 PIC16F1454 slave controllers. Overall system stability is accomplished through reading IMU data, processing that data, and determining what motion is necessary to keep the robot balanced.

Considering the software complexity of the master controller, this controller will use an object oriented programming paradigm and the C++ language. Some example objects include the IMU, UART, and slave. Having objects means that the information within each object will be compartmentalized from the rest of the program. This reduces the risk of unintentionally changing variables, or having some functions interacting with variables they should not be interacting with directly. One other advantage to the object oriented programming approach is that since the robot uses 12 slaves, instead of writing one function to control each slave we only needed to write code to define one object and then instantiate 12 of those objects to control all 12 actuators.

The IMU object will have parameters like the current gyro values for all three axes and current accelerometer values for all three axes. Public methods for the IMU object will include initializing, reading sensor values, and calibrating the sensors.

The UART object will have parameters like data in buffer and data out buffer. These will be the buffers for reading from and communicating with the PC. Methods for this object will be reading and writing to buffers, initializing, and replying to pinging from the PC.

The slave object will have parameters like current angle, limit switch status, and speed

states. These parameters are sent to the slaves to appropriately drive the motors. Methods for the slave object include initialize, write angle to the slave, read angle from the slave, and ping the slave for status.

When commands are received via UART from the PC, the master will store those commands in a variable integer array. The master will then parse to see whether motion is required and whether an acknowledge message is required to be sent back to the PC. Depending on the acknowledge bits of the verb byte sent from PC to master, the master will either send all of its known positions and IMU data, only partial IMU data, or no data.

Slave Controller

The slave controller is responsible for accepting commands from the master and generating motor drive signals. There will be 12 slave controllers, one per actuator. Each leg will require six actuators. The slave controllers will be programmed in C using MPLAB X and the free xc8 compiler. The slave firmware will implement z-domain PD compensators to control the output angle for each joint. The output angles will be measured with potentiometers through A/D converters on the PICs. The slaves will use their I2C peripheral to communicate with the master and each slave will have its own address. When a master sends a command to a slave, the slave will compare the desired angle to the actual output angle. If there is a difference between the two, the slave will send PWM signals to the motor driver circuits to generate motion of the motor. When the master requests the current angle the slave will communicate back to the master via I2C to relay the needed information.

Controller Pseudo-Code

```
Define ISRs
Initialize Peripherals
    Driver Communications
    High Level Communications
    MPU
Prepare Interrupts
    Enable Interrupts
Loop
    Check for commands,
        if there is a command
            Breakdown command to the needed motions
            Compare current position to desired position
            Generate speed and angle commands
            Send speed and angle commands to drivers
            Store data as current position
        Else
```

Read MPU data
Check to see if recovery needed
 If recovery is needed
 Send “balance” command

Driver Pseudo-Code

Initialize communication with controller
Loop
 Receive commands from controller
 Measure current position
 Compare current position to desired position
 Generate movement

Controls

Controller Concepts

There are two choices concerning the modeling of the system. The system could either be modeled through classical control or state space control. Modeling the system using classical controls is easier and less time consuming, but introduces an unstable pole into the system; therefore, this method was not used. State space has an increased level of difficulty, but does not introduce an unstable pole into the system. This method was chosen.

To simplify the controls system of the robot, an inverted pendulum approximation is made to reflect overall stability. By assuming a point mass connected by a single straight rod to a base of support, the mathematical analysis of the complete robot can be drastically simplified. The modular biped robot will have properties like falling angle and falling angular velocity that will be controlled using the control loop shown in Figure 27 below. A closed loop controller is needed for monitoring and maintaining overall system stability. The reason for choosing to control the falling angle and falling angular velocity is because the inertial measurement unit (IMU) can directly measure changes in falling angular velocity. By integrating these changes within the state estimator, the actual falling angle and falling angular velocity can be mathematically estimated and compared to the inputs.

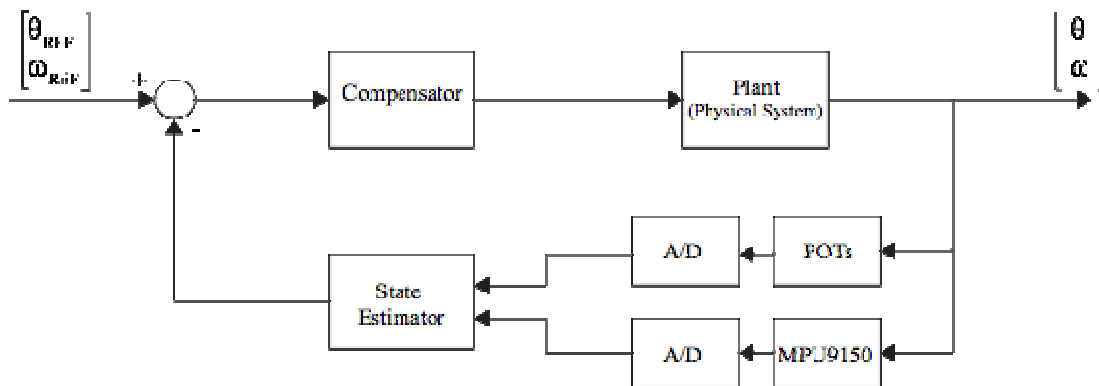


Figure 27. Control System Block Diagram

A discrete time compensator takes the error between what the target robot angle and angular velocity is compared to the actual angle and angular velocity at a sampling instance, k . The compensator then generates a target position for the foot for the next time, $k+1$, based on what is needed to reach the target angle and angular velocity set by the user. An example of the output from the compensator for a one dimensional controller is shown in Figure 28 below.

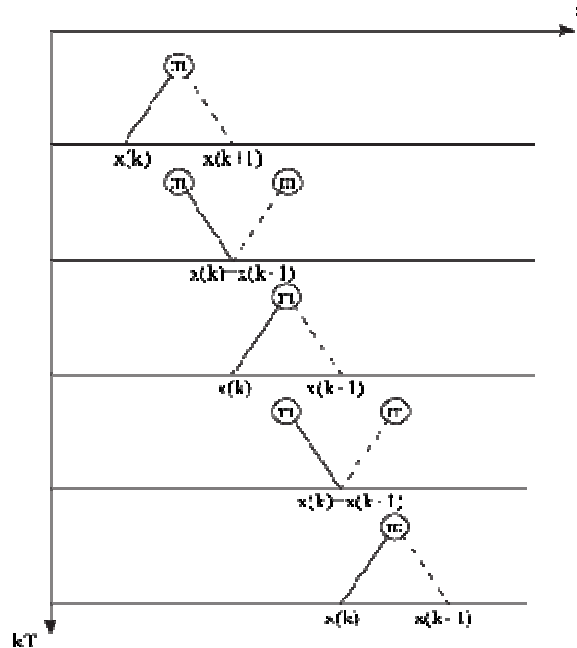


Figure 28. Time Series of Inverted Pendulum Model

By controlling the falling angle and falling angular velocity of the overall system, the linear position and linear velocity can be controlled indirectly. Implementing this technique in multiple dimensions will allow for a control system that balances during walking and during the steady state standing position.

During a typical gait cycle, the robot will encounter three modes of support. Double support mode is when the robot has two feet down as a base of support. Single support left and single support right modes are the modes that the robot will be in when only one of the feet are down as a base of support. During walking the robot will cycle through all three of these modes, but for ideal analysis it can be considered that the robot will only enter the single support modes during walking. For every other discrete time step, the support mode will alternate between single support left and single support right. When standing still the robot will be in double support mode and PID loops on the slave controllers will keep the robot balanced until the falling angle velocity changes drastically enough to induce a response from the compensator.

System Modelling

A simplified model of the robot in steady state walking, shown in Figure 29, was developed for the sagittal plane with the help of Dr. Robert Veillette at the University of Akron.

Linear Case:

Rotational Case:

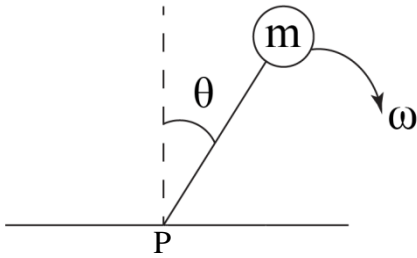


Figure 29. Inverted Pendulum with Variables Defined

Summing the moments about P results in Equation 1

Equation 1. Sum of Moments about point P

Linearizing the equations with the small angle approximation and writing the equations of motion for the inverted pendulum results in Equation 2.

Equation 2: Simplified Sum of Moments About Point

g is gravity 9.81 m/s^2

l is length of the massless support

m is a point mass representation of the sum of the leg and base mass. The robot's mass calculated from the solid model, is given as $m=2.925 \text{ kg}$

J is the moment of inertia for a point mass at a distance is $J = ml^2$. The moment of inertia calculated from the SolidWorks model of the robot is 0.17689474

Substituting the point mass moment of inertia into Equation 1 results in $J\ddot{\theta} + mgl\sin\theta = 0$

A discrete time model is created using the sequence shown below in Table 11 with the time step T defined as one step by the robot.

Defining initial and desired steady state conditions/relations, where ω and θ are defined as the angular velocity and position respectively in Table 11 below.

Table 11: Definition of Initial Conditions

The state variables are θ and ω , and state equations are defined as seen below in Equation 3 and Equation 4.

State Equations

$$\dot{\theta} = \omega$$

Equation 3: State Variable #1

$$\dot{\omega} = \frac{g}{l} \theta$$

Equation 4: State Variable #2

The continuous time state variable A matrix is formed from the state equations

$$A_p = \begin{bmatrix} 0 & 1 \\ \frac{g}{l} & 0 \end{bmatrix}$$

Converting to model discrete time state space starting by taking the inverse Laplace transform (ignoring initial conditions) of $(sI - A_p)^{-1}$

$$(sI - A_p)^{-1} = \frac{\begin{bmatrix} s & 1 \\ \frac{g}{l} & s \end{bmatrix}}{(s^2 - \frac{g}{l})}$$

Partial Fractions was performed on the $(sI - A_p)^{-1}$ matrix. Taking the inverse Laplace transform of and substituting trigonometric identities results in the discrete time state variable A matrix seen below.

$$A = e^{A_p T} = \begin{bmatrix} \cosh\left(\sqrt{\frac{g}{l}} * T\right) & \left(\sqrt{\frac{l}{g}}\right) \sinh\left(\sqrt{\frac{g}{l}} * T\right) \\ \left(\sqrt{\frac{g}{l}}\right) \sinh\left(\sqrt{\frac{g}{l}} * T\right) & \cosh\left(\sqrt{\frac{g}{l}} * T\right) \end{bmatrix}$$

Writing the state space equations from the A matrix results in equations seen below.

$$\begin{aligned} \theta(k+1) &= \cosh\left(\sqrt{\frac{g}{l}} * T\right) \times \theta(k) + \left(\sqrt{\frac{l}{g}}\right) \sinh\left(\sqrt{\frac{g}{l}} * T\right) \times \omega(k) - \frac{u(k)}{l} \\ \omega(k+1) &= \left(\sqrt{\frac{g}{l}}\right) \sinh\left(\sqrt{\frac{g}{l}} * T\right) \times \theta(k) + \cosh\left(\sqrt{\frac{g}{l}} * T\right) \times \omega(k) \end{aligned}$$

The discrete time variables θ and ω above define angular velocity and angular position in the discrete time state equations. The variable Δu is the state space controller input generated from the state estimator discussed previously and the reference angular position and velocity. The state estimator and the controller are currently under development. Preliminary design for the compensator is a pole placement state space controller, feeding the plant an adjusted step length proportional to the error in the desired and estimated angular position and velocity. This compensator will use gain matrix, K , to move the location of the unstable pole into the unit circle and on top of the stable pole, thereby making the system response stable. The step response of the uncompensated system reaches an angle of $\pi/2$, which means the robot has fallen onto the ground. It is important to note that the model used for the simulation and compensator design breaks down after angles greater than 17 degrees due to the small angle approximation used to linearize the model. After the compensator is applied, the step response should look like a saw tooth waveform shown in Figure 30 below. The vertical axis shows the controlled angle, which, when varied, will change the step length of the robot. The controlled angle will remain within the boundary of the critical angle at all times, this critical angle is the maximum or minimum controllable angle for system stability.

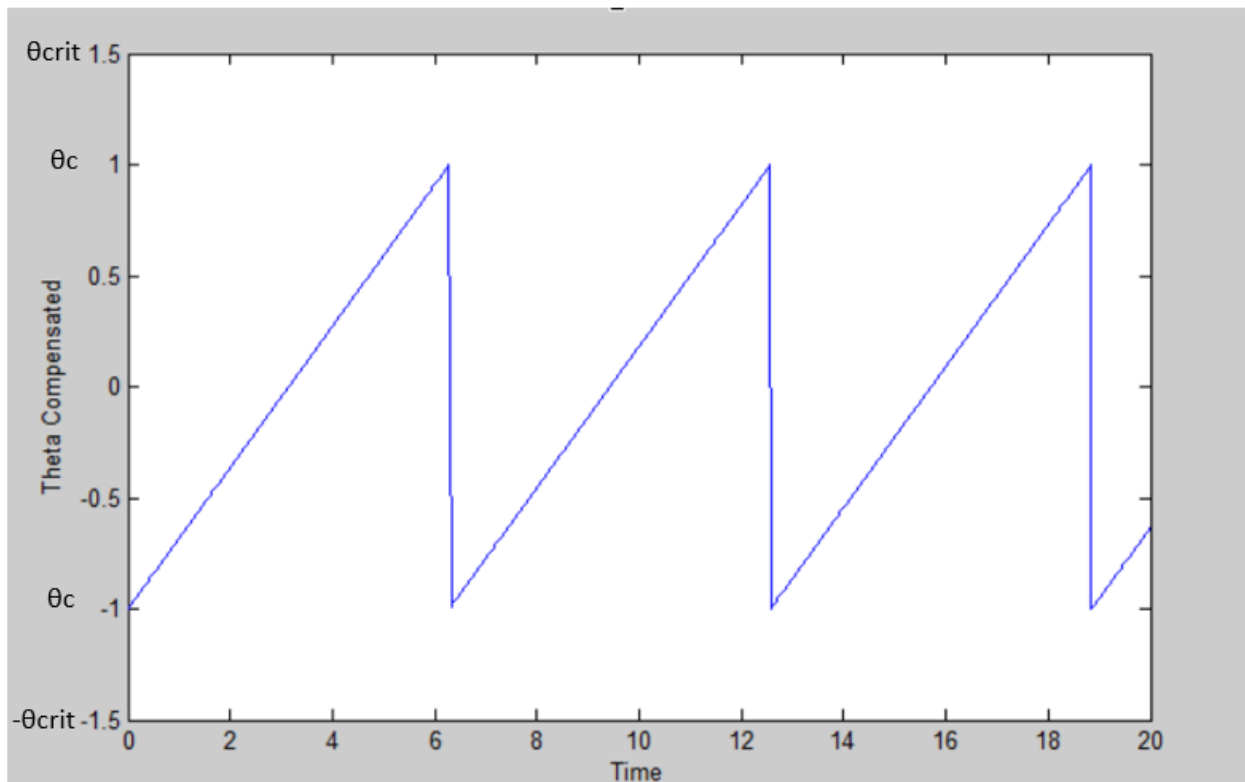


Figure 30: Desired output (θ) waveform for compensated system.

BUDGET

Proposed Bill of Materials

Below are the Electrical and Mechanical Bill of Materials (BOM) for the Modular Biped Robotic Base. The mechanical Bill of materials was generated using the SolidWorks bill of material creator. This names and quantifies all parts in the assembly. The electrical BOM was created based off of Level 1 block diagram and functional decomposition. Costs were estimated using manufacturer websites. The mechanical and electrical BOMs can be seen in Table 12 and Table 13.

Mechanical Components

ITEM NO.	Manufacturer	PART NUMBER	DESCRIPTION	QTY.	Cost Per Unit	Total Cost
1	Akron Dynamics	SL1-TH1.001	Thigh Structure & Motor Housing	2	\$35.83	\$71.66
2	Akron Dynamics	SL1-SH1.001	Thigh Structure & Motor Housing	2	\$29.83	\$59.66
3	Actobotics	535220	6 mm flange bearing	18	\$1.50	\$27.00
4	Actobotics	615406	6 mm bore bevel gear	12	\$5.99	\$71.88
5	Actobotics	634292	100 mm length 6 mm shaft	10	\$1.49	\$14.90
6	Actobotics	638272	60 RPM HD Precision Planetary Gear Motor	12	\$39.99	\$479.88
7	Actobotics					\$0.00
8	Akron Dynamics	SL1-F1.001	intermediate ankle joint	2	\$25.65	\$51.30
9	Akron Dynamics	SL1-F2.001	Foot	2	\$25.54	\$51.08
10	Akron Dynamics	SL1-H1.001	Hip joint	2	\$11.12	\$22.24
11	Actobotics		Shaft Collar	12	\$1.50	\$18.00
12	Actobotics	615266	32 Pitch (6mm Bore) Gearmotor Pinion Gear	2	\$12.99	\$25.98
13	McMaster Carr		Set Screws	35	\$0.75	\$26.25

Mechanical Subtotal: \$919.83

Table 12: Mechanical Component BOM

Electronic Components

ITEM NO.	Manufacturer	PART NUMBER	DESCRIPTION	QTY.	Cost Per Unit	Total Cost
1	Texas Instruments	CSD18504KCS	40V N-Channel NexFET™ Power MOSFET	48	\$1.31	\$62.65
2	Adafruit	1076	Arduino Due	1	\$49.95	\$49.95
3	Microchip	PIC24HJ32GP302	Slave	12	\$2.76	\$33.12
4	Digikey	Misc	Slave Peripherals	16	\$2.13	\$34.08
5	Invensense	MPU9150	Motion Processing Unit	2	\$9.11	\$18.22
6	Digikey		Wiring/Connect	24	\$4.50	\$108.00
7	Samsung		Micro USB Cable	1	\$10.00	\$10.00
8	Microchip	Pickit3 Express Debug	Programmer w/ dev board	1	\$75.00	\$75.00
9	Digikey		DC Microswitch	12	\$1.03	\$12.36
10	Digikey		Potentiometer	12	\$1.50	\$18.00

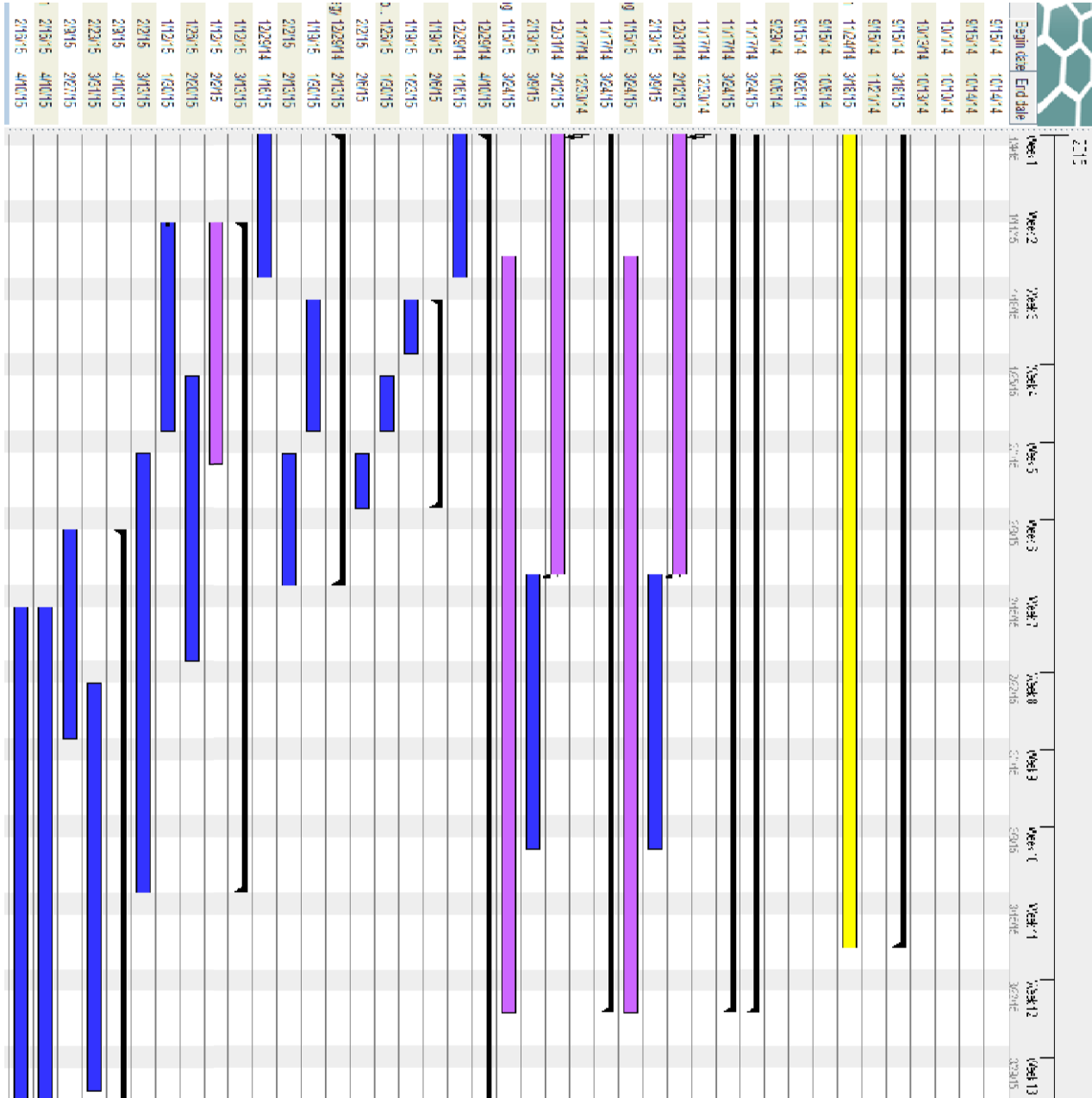
Electrical Subtotal: \$421.38

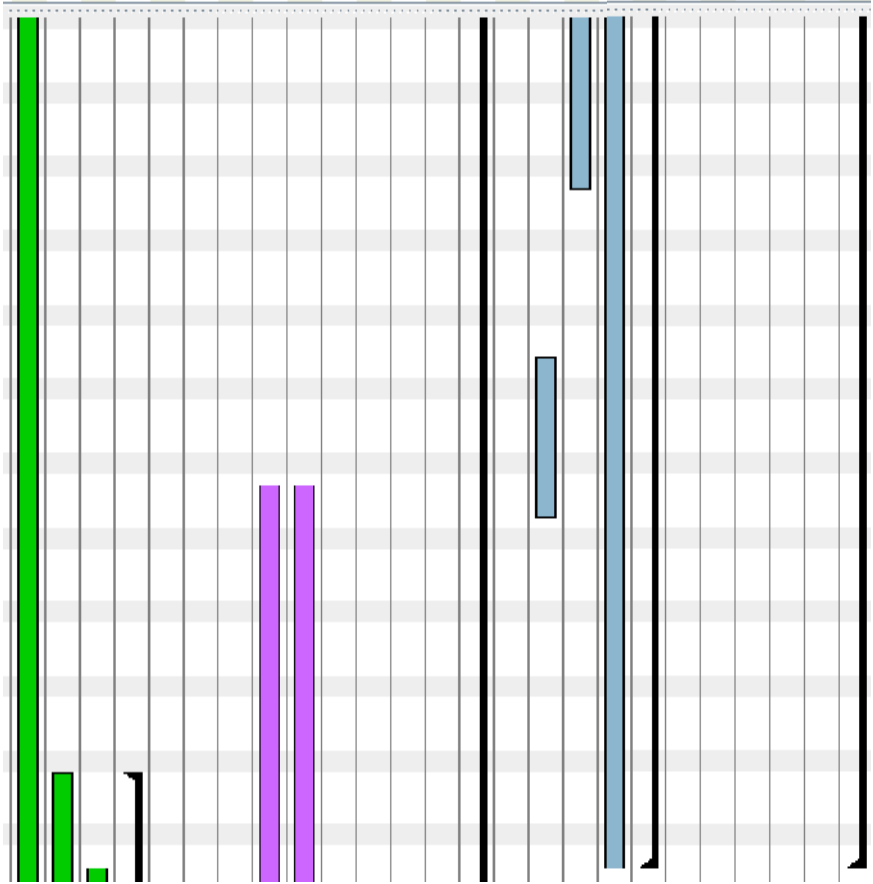
TOTAL \$1,341.21

Table 13: Electrical Component BOM

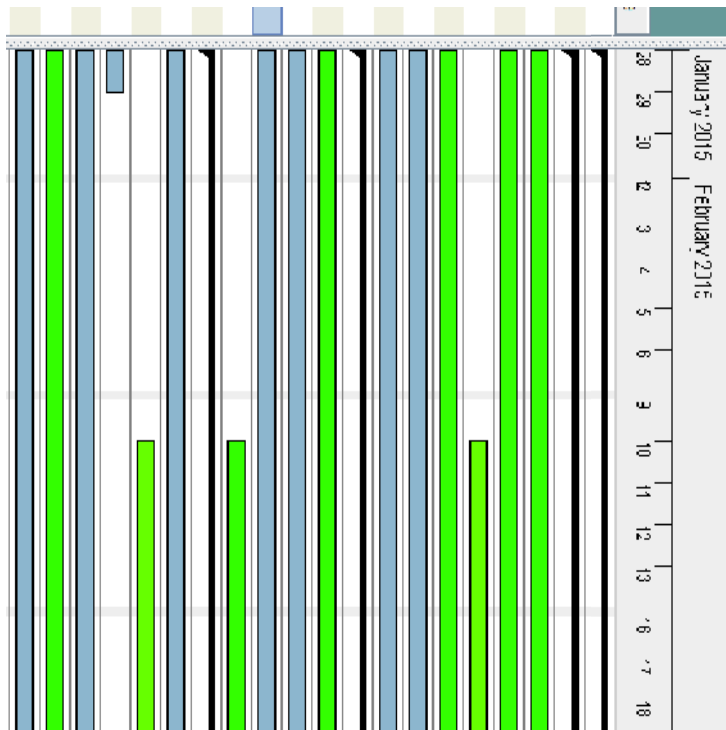
PROJECT SCHEDULE

Design Gant Chart





Implementation Gantt Chart



REFERENCES

- Ferreira, Joao P., M. M. Crisostomo, and A. P. Coimbra. "SVR Versus Neural-Fuzzy Network Controllers for the Sagittal Balance of a Biped Robot." *Neural Networks, IEEE Transactions on* 20.12 (2009): 1885-97. Web.
- Hernandez-Santos, C., R. Soto, and E. Rodriguez. "Design and Dynamic Modeling of Humanoid Biped Robot e-Robot". *Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2011 IEEE*. Web.
- National Robotics Initiative (NRI) <http://www.nsf.gov/pubs/2014/nsf14500/nsf14500.htm>.
` Advertisement. National Science Foundation September 2012: n.pag
- Ill-Woo Park, et al. "Mechanical Design of Humanoid Robot Platform KHR-3 (KAIST Humanoid Robot 3: HUBO)". *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*. Web.
- "Robots Like Us." *Communications of the ACM* 55.5 (2012): 12-3. Web.
- Kadaba, M. P., & Ramakrishnan, H. K. (1990). Measurement Lower Extremity Kinematics During Level Walking. *Journal of Orthopaedic Research*, 383-392.
- Protopapadaki, A., & Drechsler, W. (2007). Hip, knee, ankle kinematics and kinetics during stair ascent and descent in healthy young individuals. *Clinical Biomechanics*, 203-210.
- Yeoun-Jae Kim, Joon-Yong Lee, and Ju-Jang Lee. "A Balance Control Strategy of a Walking Biped Robot in an Externally Applied Force". *Information and Automation (ICIA), 2012 International Conference on*. Web.
- Taira, T., N. Kamata, and N. Yamasaki. "Design and Implementation of Reconfigurable Modular Humanoid Robot Architecture". *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*. Web.

Patents

- Ozawa, Nobuaki. Locomotion Control System of Legged Mobile Robot. Honda Giken Kogyo Kabushiki Kaisha, assignee. Patent US5838130 A. Nov 17, 1998.
- Ugo Di Proo, Tokyo (JP), Saitama (JP) Masahiro Fujita, Saitama (JP) Tsuyoshi Takagi, et al. Robot Apparatus, and Behavior Controlling Method for Robot Apparatus Tokyo (JP) Sony Corporation, assignee.
- Woong KWon, Seongnam-si (KR). Robot Walking Control Apparatus and Method Thereof Samsung Electronics Co., Ltd., SuWon-Si (KR), assignee. Patent US 8,612,054 B2. 2013.

Yoshihiro Kuroki, Kanagawa(JP), Tokyo (JP) Tatsuzo Ishida, and Jinichi Yamaguchi,
Tokyo (JP). Motion Control for a Legged Robot Tokyo (JP) Sony Corporation,
and Tokyo (JP) Jinichi Yamaguchi, assignee. Patent US 6,961,640 B2. Nov. 1, 2005

APPENDICES

Appendix A: MATLAB Code and Results for Climbing Stairs

```
%% Senior Design Team 13
% Biped Modular Robot
% Larry Chiavaroli
% Joseph Drockton
% Andrew Forchione
% Wesley Miller
% September 17, 2014

% Leg Torque Calculator
clc
close all
clear all
% Be sure to close the excel file that is being writtent to
% before running this MATLAB program.

% All units: Mass in kg, Length in m, Acceleration in m/s^2
%% Moment about the hip joint
% Initialize Variables
    % Lengths
Lhk = .254; % Length from hip to knee
Lht = Lhk/2; % Length from hip to mid-thigh
Lkf = .254; % Length from knee to ankle
Lks = Lkf/2; % Length from knee to mid-shin
    % Read Excel File Data
filename = 'Walking Joint Sim.xls'; % Excel file to read from
originalData = 1; % Sheet of data
kneerange = 'C2:C321'; % Range of data
hiprange = 'B2:B321'; % Range of data
ThetaKdegree = xlsread(filename, originalData, kneerange); % data to vector
ThetaHdegree = xlsread(filename, originalData, hiprange); % data to vector

    % Angles
ThetaH = ThetaHdegree*(pi)/180; % Angle of hip joint (down = 0
degrees)
ThetaK = ThetaKdegree*(pi)/180; % Angle of knee joint
(perpendicular to hip)

    % Masses
Maluminum = 0.0105; % Mass per unit length of aluminum
Mk = .700; % Mass summed at knee
Mf = .700; % Mass summed at foot
Mt = Lhk*Maluminum; % Mass of thigh
Ms = Lks*Maluminum; % Mass of shin
    % Forces and Accelerations
a = 9.8; % Gravitational Acceleration
Ft = Mt*a; % Force at mid-thigh
Fk = Mk*a; % Force at knee
Fs = Ms*a; % Force at mid-shin
Ff = Mf*a; % Force at foot
```

```

% Governing Equations
%fprintf('Moment at Hip - Up to knee, in Nm');
MH1 = (Lht*sin(ThetaH)*Ft)+(Lhk*sin(ThetaH)*Fk);

if (ThetaK < 0)
    MH2 = MH1 +((Lhk*sin(ThetaH)+Lks*sin(ThetaK-((pi/2)-
ThetaH)))*Fs)+((Lhk*sin(ThetaH)+Lkf*sin(ThetaK-((pi/2)-ThetaH)))*Ff);
else
    MH2 = MH1 +((Lhk*sin(ThetaH)+Lks*sin(ThetaK-((pi/2)-
ThetaH)))*Fs)+((Lhk*sin(ThetaH)+Lkf*sin(ThetaK-((pi/2)-ThetaH)))*Ff);
end
%fprintf('Moment at Hip - Total, in Nm') ;
MH2;

% Conversions
Nm2kgcm = 10.1971621298;          % Conversion factor between N-m to kg-cm
fprintf('Walking: Max Moment at Hip - Up to knee, in kg-cm');
MHKkgcm = MH1*Nm2kgcm;
[MaxMHKkgcm, MaxMHKkgcmIndex]=max(abs(MHKkgcm));
MHKkgcm(MaxMHKkgcmIndex)
fprintf('Walking: Max Moment at Hip - Total, in kg-cm');
MHTkgcm = MH2*Nm2kgcm;
[MaxMHTkgcm, MaxMHTkgcmIndex]=max(abs(MHTkgcm));
MHTkgcm(MaxMHTkgcmIndex)

% percentGait = rot90([0:(100/319):100]);
% figure(1)
% subplot(2, 1, 1)
% plot(percentGait, MHKkgcm)
% title('Hip to Knee Torque - No Leg (kg-cm)')
% subplot(2, 1, 2)
% plot(percentGait, ThetaH)
% title('Angle of Hip Joint - 0 is "down"')
% figure(2)
% subplot(3, 1, 1)
% plot(percentGait, MHTkgcm)
% title('Hip to Ankle Torque (kg-cm)')
% subplot(3, 1, 2)
% plot(percentGait, ThetaH)
% title('Angle of Hip Joint - 0 is "down"')
% subplot(3, 1, 3)
% plot(percentGait, ThetaK)
% title('Angle of Knee Joint - 0 is "bent"')

xlswrite('Walking Joint Sim.xls',ThetaH,'Exported Data','B2');
xlswrite('Walking Joint Sim.xls',ThetaK,'Exported Data','C2');
xlswrite('Walking Joint Sim.xls',MHTkgcm,'Exported Data','D2');
xlswrite('Walking Joint Sim.xls',MHKkgcm,'Exported Data','E2');

fprintf('Data has been exported to Excel');
winopen('Walking Joint Sim.xls')

```

Appendix B: Data Sheets

Microcontroller

<http://ww1.microchip.com/downloads/en/DeviceDoc/41639A.pdf>

P-Channel MOSFET

<https://www.fairchildsemi.com/datasheets/FQ/FQP47P06.pdf>

N-Channel MOSFET

<https://www.fairchildsemi.com/datasheets/FD/FDPF320N06L.pdf>

Small Signal NPN Transistor

<http://www.st.com/web/en/resource/technical/document/datasheet/CD00002987.pdf>

Arduino Due 1076

<http://www.adafruit.com/product/1076>