

**DESIGN AND CONTROL OF A NEW  
RECONFIGURABLE ROBOTIC MOBILITY PLATFORM**

A Thesis  
Presented to  
The Academic Faculty

by

Byron Johns

In Partial Fulfillment  
of the Requirements for the Degree  
Master's of Science in the  
School of Mechanical Engineering

Georgia Institute of Technology  
May 2007

# **DESIGN AND CONTROL OF A NEW RECONFIGURABLE ROBOTIC MOBILITY PLATFORM**

Approved by:

Dr. Ayanna Howard, Co-Advisor  
School of Electrical and Computer Engineering  
*Georgia Institute of Technology*

Dr. Nader Sadegh, Co-Advisor  
School of Mechanical Engineering  
*Georgia Institute of Technology*

Dr. Harvey Lipkin  
School of Mechanical Engineering  
*Georgia Institute of Technology*

Date Approved: April 2007

## **ACKNOWLEDGEMENTS**

First I would like to thank Dr. Ayanna Howard for the opportunity to work with her in the Human Automation Systems Lab at Georgia Tech. Dr. Howard was able to give me the perfect project to bridge my background of both electrical and mechanical engineering and my interest in robotics, and provided guidance throughout my research. I wish to acknowledge my fellow HumAnS Lab graduate students who were more than helpful in aiding me in some of the road blocks that came along in this project. I would also like to express gratitude to the people at NASA and the United Negro College Fund Special Programs office, for the generous fellowship, and all of the opportunities that came with it. A special thanks to Dr. Edward Tunstel, my NASA mentor, and the scientists and engineers at the NASA Jet Propulsion Lab for assisting me in essential problem solving in my research during my summer working there. A thank you to Dr. Nader Sadeh for co-advising me and finding time in his busy schedule to aid me in my problem solving when needed. Finally, and most importantly, I would like to thank my parents, who were always there for me and supported me throughout my education.

# TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	iii
LIST OF TABLES	vii
LIST OF FIGURES	viii
LIST OF SYMBOLS AND ABBREVIATIONS	x
SUMMARY	xii
<u>CHAPTER</u>	
1 Introduction	1
1.1 Objective	1
1.2 Legged vs. Wheeled Mobility & Terrains	2
1.3 Past and Present NASA Rover Designs	3
2 Byrobot Hardware	9
2.1 The Chosen Design	9
2.2 CAD Designing – Pro Engineer	9
2.3 Hardware Properties	11
2.4 Leg Joint Torque / Servo Motor Calculations	13
2.5 Mars Shoes	21
3 Byrobot Control	23
3.1 Servo Motors	23
3.1.1 Servo Motor Control	23
3.1.2 Pulse Width Modulation	24
3.1.3 PWM Signal Generation	27
3.1.4 Controlling the Servos	28

3.2	Byrobot Open-Loop Control	30
3.3	The SSC-32 Servo Controller	31
3.4	The Eyebot Controller	34
3.5	Power	36
3.5.1	NiMH vs. NiCd Batteries	36
3.5.2	Servo Power	38
3.5.3	Current	39
4	Wheeled Kinematics	43
4.1	Robot Orientation	43
4.2	Differential Wheel Drive Kinematics	45
4.3	Robot Radius of Curvature	48
5	Legged Kinematics	49
5.1	Forward Kinematics – Geometrical Method	49
5.2	Forward Kinematics – Denavit Hartenberg Convention	50
5.3	Inverse Kinematics	56
6	Stabilizing Byrobot’s Legged Mobility System	59
6.1	Polygon of Support	59
6.2	Mobility on an Incline / Decline Plane	60
6.3	Hexapod Walking Gaits	62
6.3.1	Preferred Walking Gaits for Hexapod Robots	62
6.3.2	Analysis of Walking Gaits	64
6.3.3	The Chosen Walking Gait	65
7	Results	67
7.1	Verifying Kinematic Equations	67
7.2	Mobility on Flat Surface	68

7.3 Mobility in Mars Sand Pit	72
7.4 Traversing Up and Down a Slope	74
7.5 Conclusions	76
8 Future Work	79
APPENDIX A: BYROBOT MAIN RECONFIGURATION CODE	84
APPENDIX B: BYROBOT MAIN RECONFIGURATION FLOW CHART	88
APPENDIX C: BYROBOT PARTS LIST AND PRICES	89
APPENDIX D: WEIGHT OF BYROBOT AND COMPONENTS	90
APPENDIX E: WIRE GAUGE CHART	91
APPENDIX F: EYEBOT CONTROLLER SPECIFICATIONS	93
APPENDIX G: SSC-32 SERVO CONTROLLER SPECIFICATIONS	94
APPENDIX H: THE HS-322HD STANDARD DELUXE SERVO	95
APPENDIX I: THE HS-645MG STANDARD DELUXE HIGH TORQUE SERVO	96
APPENDIX J: THE HSR-5995TG CORELESS DIGITAL SERVO	97
REFERENCES	98

## LIST OF TABLES

	Page
Table 1.1: Pros and Cons of Legged and Wheeled Mobile Robots	3
Table 7.1: Comparison of Foot Position with Measured and Calculated Results	68
Table 7.2: Average Wheeled Linear Velocity of Byrobot at Full Speed	69
Table 7.3: Average Wheeled Linear Velocity of Byrobot at 80% Speed	69
Table 7.4: Average Wheeled Linear Velocity of Byrobot at 50% Speed	70
Table 7.5: Average Angular Velocities of Robot	70
Table 7.6: Walking (wave gait) Speed of Byrobot on Flat Surface	72
Table 7.7: Walking (wave gait) Speed of Byrobot in Mars Sand Pit	73
Table 7.8: Wheeled Speeds of Byrobot Traversing Up a 22° Incline Slope	74
Table 7.9: Wheeled Speeds of Byrobot Traversing Down a 22° Decline Slope	75
Table 7.10: Wheeled Speeds of Byrobot Traversing Up a 35° Incline Slope	75
Table 7.11: Wheeled Speeds of Byrobot Traversing Down a 35° Decline Slope	76

## LIST OF FIGURES

	Page
Figure 1.1: Bobby Rover	4
Figure 1.2: Go-For Rover	5
Figure 1.3: Rocker III Micro rover	5
Figure 1.4: MER Rover	6
Figure 1.5: The ATHLETE Rover	7
Figure 1.6: The Lemur Robot	8
Figure 2.1: Pro-Engineer CAD Models	10
Figure 2.2: Actual Photos of Byrobot	12
Figure 2.3: Comparison of CAD model leg mechanism to actual Byrobot hardware	13
Figure 2.4: Front View Diagram of ByroBot (minus the wheels)	14
Figure 2.5: Byrobot torque calculation diagram	15
Figure 2.6: Alternate Byrobot torque calculation diagram	19
Figure 2.7: Top and bottom view of Mars Sand Shoe	21
Figure 2.8: Mars Sand Shoes in place on Byrobot	22
Figure 3.1: Inside of a Servo	24
Figure 3.2: PWM and Average Voltage	26
Figure 3.3: Servo's High Pulse Width Determines the Angle Position	28
Figure 3.4: Controlling the HS-322HD servo with PWM	29
Figure 3.5: DB9 Serial Cable Null Modem Connection	35
Figure 3.7: Connection of Servo Controller to Battery Pack	40
Figure 4.1: Robot orientation in world frame	43
Figure 4.2: Reduced orientation of robot	44



Figure 4.3: Fixed Wheel Formation; Wheels on Robot Body	46
Figure 5.1: Kinematic Model of our 3R leg mechanism on Byrobot	49
Figure 5.2: Byrobot's body diagram	52
Figure 6.1: Byrobot's polygon of support	60
Figure 6.2: Byrobot's projected force and COM on an incline/decline plane	61
Figure 6.3: Hexapod gait movements of each leg with respect to time	64
Figure 7.1: Photographs of the Wave Gait sequence of Byrobot	72
Figure 8.1: Future robot Closed Loop Feedback Control	79
Figure 8.2: Possible Future Robot Encoder Add-On	81

## LIST OF SYMBOLS AND ABBREVIATIONS

D - total distance robot traveled	$l_i$ – leg number “i”, where “i” is leg 1,2,3,4,5, or 6
$d_L$ - distance traveled by left wheel	$l_o$ – perpendicular distance between robot body x-axis and the x-axis of any robot leg
$d_R$ - distance traveled by right wheel	$d_o$ – perpendicular distance between robot body y-axis and the y-axis of any robot leg
$\omega$ - angular velocity of robot	$T_G^O$ - transformation matrix from the robot body origin (G) to the origin of the robot legs (O)
$\omega_L$ - angular velocity of left robot wheel	$d_2$ – offset length of robot leg from the first revolute joint (pelvic joint) out of the origin down to the first leg segment length
$\omega_R$ - angular velocity of right robot wheel	$L_1$ – leg segment length between offset length $d_2$ and hip joint
V - linear velocity of robot	$L_2$ – leg segment length between hip joint and knee joint
$V_L$ - linear velocity of left robot wheel	$L_3$ – leg segment length between knee joint and foot
$V_R$ - linear velocity of right robot wheel	$\theta_1$ – rotational angle of leg pelvic joint
$\theta$ - radians robot turns in world frame	$\theta_2$ – rotational angle of leg hip joint
$\theta_L$ - radians of left robot wheel	$\theta_3$ – rotational angle of leg knee joint
$\theta_R$ - radians of right robot wheel	G – label of origin of robot body in its respective x, y, z frame
R - radius of curvature of robot	$y_G$ – x position of the origin of the robot body in its respective x, y, z frame
r - radius of wheel	$y_G$ – y position of the origin of the robot body in its respective x, y, z frame
t - elapsed time	
ICR - instantaneous radius of curvature	
L - horizontal length between robot wheels	
$L/2$ – half of horizontal length between robot wheels	
$R(\theta)$ - rotation of the robot car about the z-axis	
T – homogeneous transformation matrix for robot leg mechanism	

$F_{\text{down}}$ – Force of robot projected parallel to the incline/decline plane	$y_G$ – z position of the origin of the robot body in its respective x, y, z frame
$F_{\text{in}}$ – Force of robot projected perpendicular into the incline/decline plane	$X_3$ – final X position of robot foot
$F_f$ – Force of friction of robot feet on a plane	$Y_3$ – final y position of robot foot
$g$ – force of gravity	$Z_3$ – final z position of robot foot
$m$ – robot mass	$\dot{x}$ - speed of the robot with respect to the x-axis
$a$ – acceleration	$\dot{y}$ - speed of the robot with respect to the y-axis
$\mu$ - coefficient of friction of robot on a incline/decline plane	$\dot{\theta}$ - rotational speed of the robot (rotating about the z-axis)
$J$ – jacobian matrix	$S_i$ - $\sin(\theta_i)$
$W$ – wrench vector	$C_i$ - $\cos(\theta_i)$
$P(O,i)$ – position vector from origin point “O” to point “i”	$S_{i+j}$ - $\sin(\theta_i + \theta_j)$
$Q$ – robot joint loads (torque)	$C_{i+j}$ - $\cos(\theta_i + \theta_j)$
$x$ – x position of robot in x, y, z coordinate frame	$e$ – error
$y$ – y position of robot in x, y, z coordinate frame	
$z$ – z position of robot in x, y, z coordinate frame	

## SUMMARY

The development of a new family of robotic vehicles for use in the exploration of Mars and other remote planets is an ongoing process. Current rovers have to traverse rough terrain and be able to withstand various conditions on Mars. The goal of this project is to design a new Mars rover mobility system that performs to optimum capability. This project will involve the design and control of a robot that will use wheels, as well as legs, allowing the user to control which ever mobility option they want, and giving the robot the ability to traverse various terrains. Some of the legged-wheeled robots that currently exist have their wheels attached to an actuator located at the end of the robot leg. When the robot is commanded to walk, the wheel is stationary and the robot actually walks on its wheel. This causes a number of problems that hinders long-term and robust operation in remote environments. For these reasons, a new reconfigurable robot, *Byrobot*, was developed. This new hybrid legged-wheeled rover possesses a six-legged walking system as well as a four-wheeled mobility system. CAD designing for the hardware of this new robot is first done, and mechanisms and animations are run to test movement of parts. Thorough kinematic analyses are done for both the legged and wheeled mobility systems of the robot. This allows for findings such as the most stable stance and gait for walking the robot, and knowing the location and orientation of the robot in the world coordinate frame for driving and mapping. This new robotic mobility platform will facilitate future Mars exploration.

# **CHAPTER 1**

## **INTRODUCTION**

The development of a new family of robotic vehicles for use in the exploration of remote planetary surfaces, such as Mars, and remote sites on Earth, such as Antarctica, is an ongoing process [1]. Current robotic vehicles must traverse rough terrain having various characteristics such as steep slopes, icy surfaces, and cluttered rock distributions, to name a few. The goal of the Byrobot project is to design a new robotic mobility system that performs to optimum capability in remote environments, which leads to the idea of this reconfigurable legged-wheeled robot. In order to guarantee success of robotic missions for the future, technologies that can enable multi-rover collaboration and human-robot interaction must be matured. The main hurdle with this focus is the cost and system complexity associated with deploying multiple robotic vehicles having the capability to survive long periods of time, as well as possessing multi-tasking capability. To address this issue, this research focuses on modularizing both hardware and software components to create a reconfigurable robotic explorer.

### **1.1 Objective**

Science exploration in unknown and uncharted terrain involves operating in an unstructured and poorly modeled environment, and there are several robotic designs that are plausible for operating in these types of environments. The goal of this project was to design and control a new reconfigurable robotic mobility platform. This new rover, self-

named the *Byrobot*, will implement both legs and wheels giving it the ability to operate on various terrains.

The robot was first modeled on Pro-Engineer, and then the parts were constructed in the Georgia Tech MRDC Machine Shop. Byrobot is able to drive on its 4 wheels and roll over obstacles, as well as have the legs retract up, so it can stand and walk in its legged configuration for operating on rough terrain or navigating over larger obstacles. Two controllers were used for the robot in order to control the various electronic components that are used. These two controllers, the Eyebot and the SSC-32 Servo Controller, will be discussed later in Chapter 3.

When Byrobot stands, it is primarily supported by the high-torque servos that are at the "hip joint" and "knee joint" of each leg, which will be discussed in Chapter 2. Joint torques were calculated to determine the torque needed for these servos so that Byrobot could support itself standing. Kinematic analysis is done for the wheels to find velocity and position data, as well as the radius of curvature of the robot motion using different linear velocity of the wheel pairs in its differential drive. Forward and inverse kinematic analysis was done for the legged mobility system to find the joint angles in the robot legs and the corresponding foot position of each leg. This analysis will be discussed in Chapters 4 and 5.

## **1.2 Legged vs. Wheeled Mobility & Terrains**

There are pros and cons to both wheeled and legged robots. Wheels are the preferred mobility system because they are fast and easy to control. However, if the robot is on ice, sand, or even some mushy surface, then the wheels may not be able to get

traction to rotate, thus making the legs more desirable to use. Also, depending on the steepness of an incline, you have a choice of which mobility system to use. This is the primary reason this reconfigurable design was chosen. A table listing the pros and cons of legged and wheeled mobility systems is shown below.

*Table 1.1: Pros and Cons of Legged and Wheeled Mobile Robots*

	Pros	Cons
Wheeled Robots	<ul style="list-style-type: none"> <li>• Robot can operate at a fast speed.</li> <li>• Better at low energy levels.</li> <li>• Easier to control.</li> <li>• Speed can be varied with simple control mechanisms</li> <li>• Break when necessary, such as when traversing down an incline.</li> </ul>	<ul style="list-style-type: none"> <li>• Can lose traction on a slippery or mushy terrain such as ice or mud.</li> <li>• Not many choices for driving the robot. A four-wheeled robot can only be driven by turning two or four wheels at a time.</li> <li>• May lose traction on a slope that is too steep.</li> </ul>
Legged Robots	<ul style="list-style-type: none"> <li>• Can operate on terrain where the wheels may lose traction, such as a mushy or slippery surface.</li> <li>• Different walking patterns can be chosen for the gait, depending on the robot load and number of legs.</li> <li>• May be able to perform better on a slope by controlling the leg joint actuators to position the robot center of mass in a stable position.</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult to control due to number of joint actuators.</li> <li>• Difficult to stabilize</li> <li>• May tumble down a slope if it loses stability.</li> </ul>

### **1.3 Past and Present NASA Rover Designs**

The majority of NASA rovers use only wheeled mobility as a means of locomotion. Some of these past wheeled mobility robotic systems include the Robby Rover [2], the Go-For Micro rover [3], and the Rocky III Micro rover [4].

The **Robby Rover**, shown in Figure 1.1, enables researchers to develop techniques for autonomous navigation and manipulation in support of future NASA missions to Earth’s Moon and Mars. The six-wheeled articulated test bed provides all necessary onboard computing, sensing, mobility, manipulation, power, and thermal control resources for autonomous testing. Robby is approximately 4m (13ft) long, 2m (6.5ft) wide, and has a maximum height of 2.5m (8ft). Weighing about 2,000kg (4,400lbs), the rover can reach a maximum speed of 1 m/s.



*Figure 1.1: Robby Rover*

The **Go-For micro rover**, shown in Figure 1.2, is able to traverse rough terrain and climb over very large obstacles because of its novel “fork wheel” design. The vehicle has four wheels that are mounted on “forks” (pairs of struts that can rotate together on the ends of an axle through the micro rover body). A control system adjusts the positions of the forks to keep 80% or more of the weight of the micro rover over the rear wheels in its normal stance. This gives the rear wheels enough traction to thrust and lift the front wheels over obstacles as high as 70% of the length of the vehicle in the

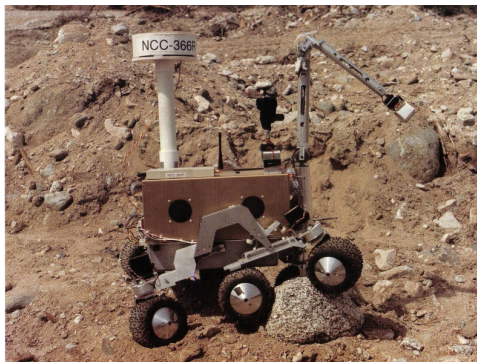


stowed or upright position. The forks are even powerful enough to right Go-For if it is overturned.



*Figure 1.2: Go-For Rover*

The **Rocky III Micro rover**, shown in Figure 1.3, was developed for an autonomous sample acquisition experiment using a computerized navigation system called “behavior control.” Rocky III is based on a six-wheel design with articulated, “rocker-bogie” suspension that enables it to traverse obstacles as high as 1.3 times its wheel diameter. Weighing 15kg (33lbs), the test bed micro rover is 60cm (23.4in) long and 45cm (17.5in) wide.



*Figure 1.3: Rocky III Micro rover*

Currently new generations of NASA rovers are being developed and used which can solve some of the mobility and manipulation problems that the past rovers had. One such rover is **The Mars Exploration Rover (MER)** [5], shown in Figure 1.4. Due to the design of its mobility system, MER can only traverse 60% of the Mars surface, where the other 40% are cliff areas that it is unable to navigate.

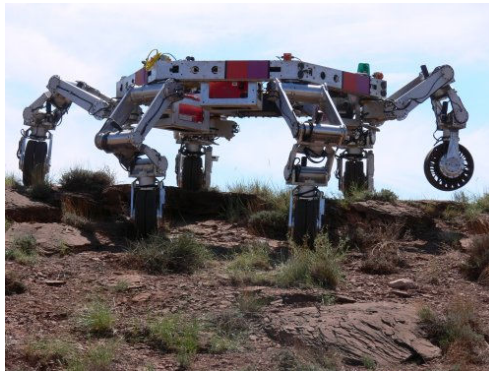


*Figure 1.4: MER Rover*

Some of the legged-wheeled NASA robot prototypes that currently exist have their wheels attached to an actuator located at the end of the robot leg. One such platform is the new JPL **ATHLETE (All-Terrain Hex-Legged Extra-Terrestrial Explorer) Rover** [6], shown in Figure 1.5. ATHLETE is capable of rolling over undulating terrain and "walking" over extremely rough or steep terrain so that robotic or human missions on the surface of the Moon can load, transport, manipulate, and deposit payloads essentially to any desired site of interest. The first version of the ATHLETE vehicle is greater than 4m in diameter and has more than 6m reach, has 6-DOF legs for generalized robotic manipulation, and has large payload capacity of 450 kg per vehicle, with much more for multiple ATHLETE vehicles docked together. This system will be able to move at 10

km/h over Apollo-like moon terrain ( $>100$  times faster than Mars Exploration Rover (MER)), climb vertical steps of at least 70% of the maximum stowed dimension of the vehicle ( $>2\times$  MER), and climb slopes of  $50^\circ$  on rock and  $25^\circ$  on soft sand.

Unfortunately, problems with this type of legged-wheeled design could occur when the robot is commanded to walk, where the wheel is stationary and the robot actually walks on its wheel. This causes a number of problems that hinders long-term and robust operation in remote environments. This can also make the robot less stable when it rolls on its wheels, since the robot will be at a potential unstable height above the ground depending on the length of the legs where the wheels are attached.



*Figure 1.5: The ATHLETE Rover*

The assembly, inspection, and maintenance requirements of permanent installations in space demand robots that provide a high level of operational flexibility relative to mass and volume. For this, the **Lemur** [7] robot was developed, shown in Figure 1.6. Lemur explores mechanical-design elements and provides an infrastructure for the development of algorithms. The physical layout of the system consists of six 4-degree-of-freedom limbs arranged about a hexagonal body platform. These limbs

incorporate a "quick-connect" end-effector feature that allows the rapid change-out of any of its tools. Lemur is being used to investigate several aspects of climbing-system design, including the mechanical system (novel end-effectors, kinematics, joint design), sensing (force, attitude, vision), low-level control (force-control for tactile sensing and stability management), and planning (joint trajectories for stability).



*Figure 1.6: The Lemur Robot*

These rover systems summarize the state-of-the-art in robotic platforms for space exploration. Although some of these systems have been deployed in previous NASA missions, they are not fully able to traverse the entire spectrum of terrain found on remote planetary surfaces. As such, in order to both capitalize on the benefits provided by wheeled locomotion, while also taking advantage of the positive attributes provided by legs, we have developed the Byrobot, our new hybrid reconfigurable legged-wheeled rover.

## **CHAPTER 2**

### **BYROBOT HARDWARE**

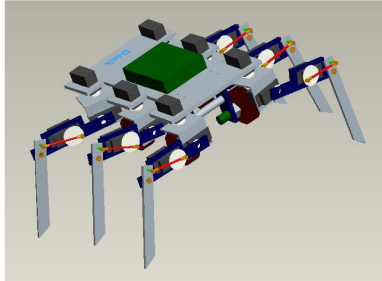
#### **2.1 The Chosen Design**

A legged robot has to have at least two legs in order to walk. The more legs that a robotic vehicle possesses, the more difficult it becomes to control due to the increase in the number of actuator variables that need to be commanded [8]. The positive side to this is that more legs on the robot an increase in stability and a more diverse of walking gaits becomes available for the robot to implement. The Byrobot was designed to have a hexapod legged configuration, where having six legs provides maximum robot stability, with a minimum number of variables to control (as further discussed in chapter 6). Each leg has 3 joints (pelvic, hip, and knee), similar to a human leg, for a three-revolute (3R) kinematic chain to give the robot three degrees of freedom (3-DOF). Each of the joints is actuated by a servo motor, which has 180° of rotation. The wheeled mobility system is a 4 wheel drive with each wheel attached to a DC motor. The legs can retract to lower the robot onto its wheels, and can also raise the robot to return to a walking configuration in the case where the wheels aren't able to function at a desired performance specification, such as when navigating on a icy or sandy surface.

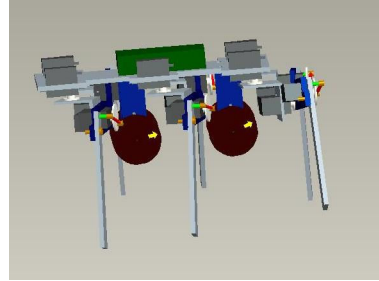
#### **2.2 CAD Designing – Pro-Engineer**

Byrobot was first designed in Pro-Engineer CAD software. Constructing a CAD model saves time and money in the manufacturing process. Pro-Engineer allowed us to virtually design and assemble all of the robot parts, including DC and servo motors, and

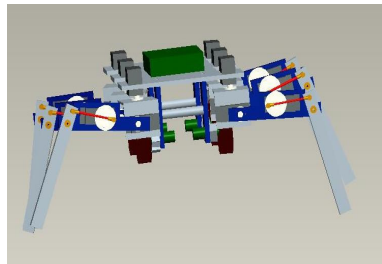
run the mechanisms to test for part interference. Once the model is constructed, CAD drawings can be easily printed out for each of the parts. These drawings were then taken to the MRDC Machine Shop at Georgia Tech to be used to cut out the actual robot hardware parts. Figure 2.1 shows some of these CAD drawings.



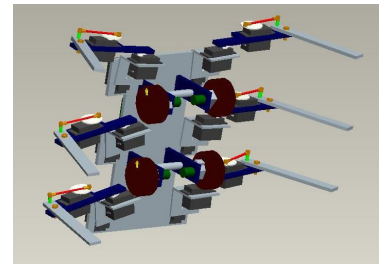
Top view, legs extracted. The green box represents where the controllers and batteries will be located.



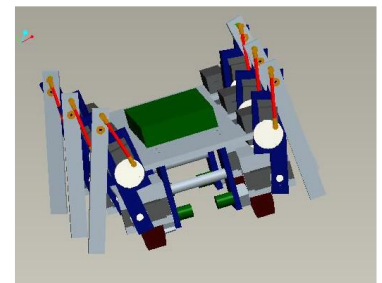
Side view, legs extracted. Only about 4" of ground clearance between feet and wheels



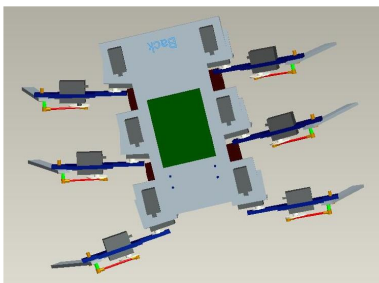
Front view of robot standing.



A view underneath the robot showing the wheels and all 18 of the servo motors.



Legs are retracted so robot can roll on wheels



Top view of robot with legs extracted.

*Figure 2.1: Pro-Engineer CAD Models*

As mentioned earlier, the CAD model can be considered almost an exact virtual replica of the robot hardware.

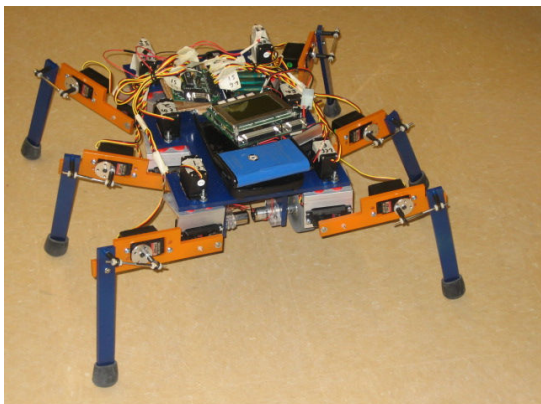
### **2.3 Hardware Properties**

The robot body material is polycarbonate plastic (which is nice and robust, and more lightweight than aluminum or some other metal). This plastic was used for the top robot body platform, the legs, and the structure to hold the legs in place. A thin grade of aluminum was used for the L-brackets that connect the pelvic and hip servos so that they can move together. A light weight durable foam tire was used on the wheels of the robot. These foam tires allow for minimal slippage so that the robot is able to turn in place.

The robot body frame was designed to accommodate the 18 servo motors required to operate the robot legs, as well as the two controllers, the Eyebot and SSC-32 (which are discussed in Chapter 3), and the necessary batteries. A standard push-rod and ball link joint is used to connect the knee servo to the knee joint, so that the knee will not rotate but swing in and out like a human knee. Figure 2.2 shows photos of Byrobot both standing and “sitting” on its wheels on the floor, and in our HumAnS-Lab Mars sand pit.

In Figure 2.3, we show a comparison of Byrobot’s CAD model and actual hardware. We can see from the figure all three revolute joints, controlled by the servo motors, of the 3R leg mechanism on the robot. The pelvic, hip, and knee servos are labeled.

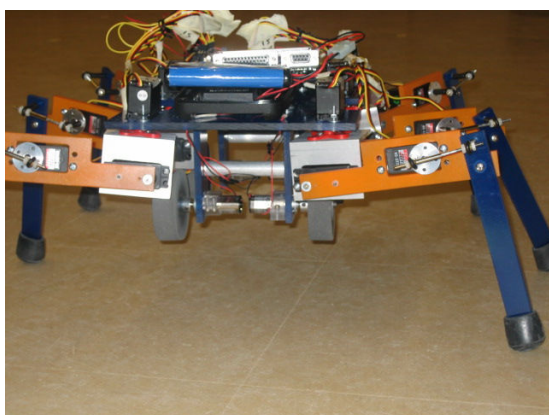




Byrobot standing on legs.



Byrobot standing on legs in lab Mars sand pit



A view underneath Byrobot as it stands on its legs.



Byrobot rolling on wheels in lab Mars sand pit.



Byrobot walking on its feet in the HumAnS Lab Mars sand pit.



More of Byrobot navigating on its legged mobility system.

*Figure 2.2: Actual Hardware Photos of Byrobot*



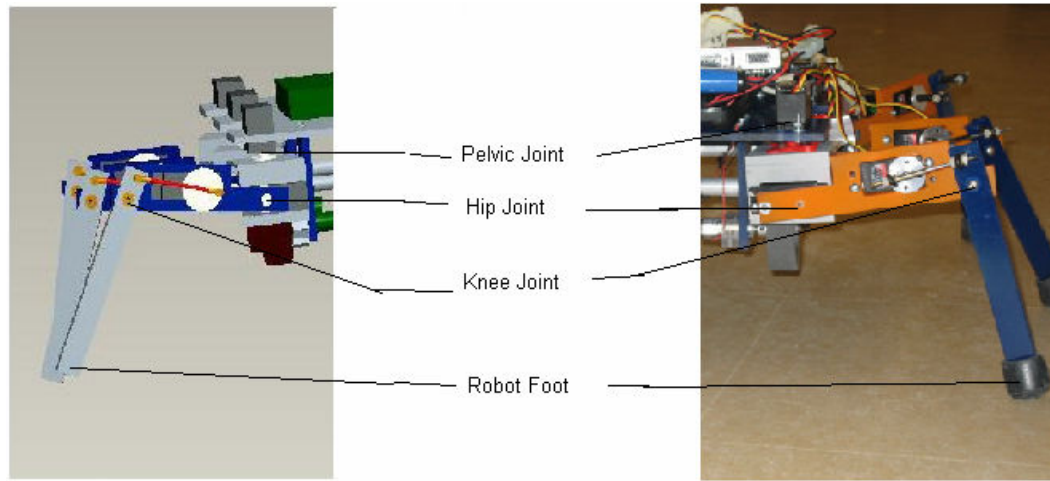
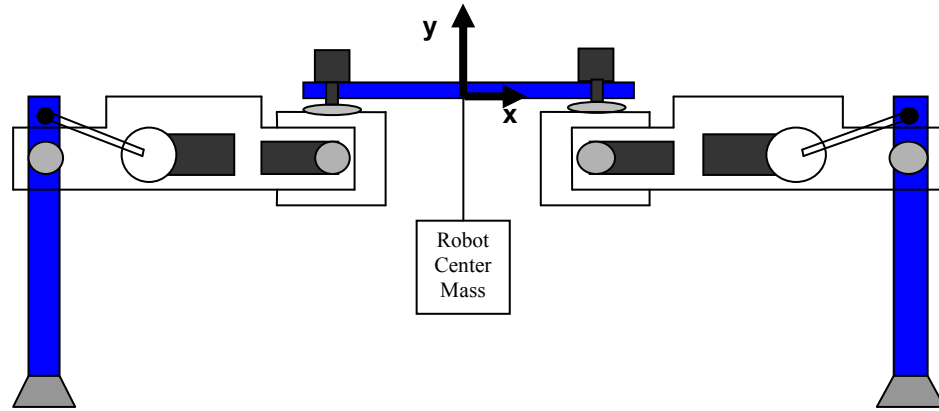


Figure 2.3: Comparison of CAD model leg mechanism to actual Byrobot hardware.

## 2.4 Leg Joint Torque / Servo Motor Calculations

Based on the Byrobot design, the joints at the “hip” and “knee” positions would experience the largest torque when the robot was standing and walking. As you can see from Figure 2.3 above, these joints will support the entire robot load when the Byrobot is utilizing its legged mobility system. So just how much torque will be applied to these joints? Answering this question also tells us how much torque is needed in the servo motors at these joints. The robot needs to be in *static equilibrium* [9] when standing still on its legs (i.e. the sum of all the forces and moments should equal zero). Depending on how many legs the robot will have simultaneously positioned on the ground at an instance of time while walking, there will be a fraction of the robot load (its body weight) that each leg will have to support. High-torque servos will be needed at these joints to guarantee stability for the robot. In this section, we calculate the joint torques related to the minimum torque required for the robot leg servos. A diagram of the front of Byrobot

standing on its legs, minus the wheels, is shown in Figure 2.4. Although there are six legs in our hexapod design, we assume that the robot's weight is approximately equally distributed, so that each leg supports  $1/6^{\text{th}}$  of the robot weight. This weight is estimated by adding up the weight of all the robot parts.

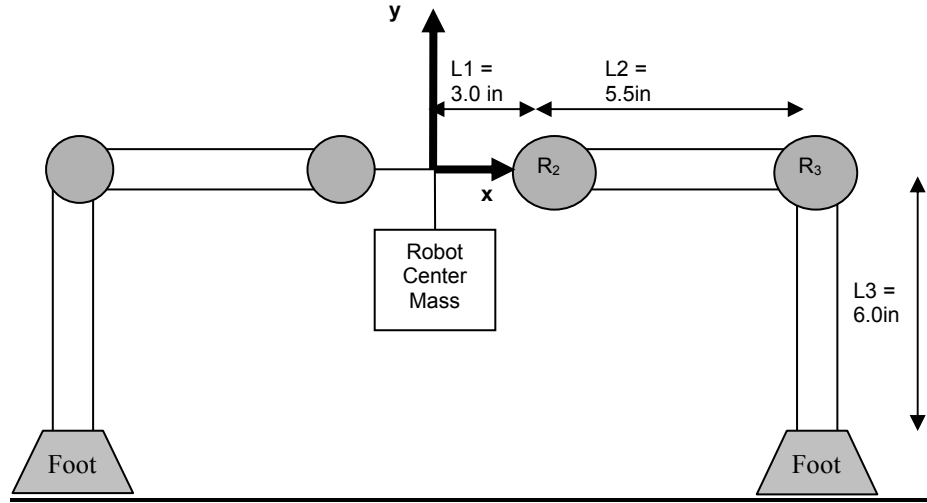


*Figure 2.4: Front View Diagram of Byrobot (minus the wheels), standing on its legs. Only the front 2 legs are shown.*

The joint torques is determined by solving a *static analysis* problem, where the robot represents a planar case of this problem. In the planar case, the wrench (the simplest representation of a system made up of forces and couples) will only force components in the x and y directions, and a moment in the z direction.

Based on the assumption that the weight of the robot will be evenly distributed about all the legs in contact with the ground, we use one leg of the robot for our analysis. We also assume that the robot will be standing on a minimum of **three** legs at one time, (the associated three-legged walking gait is discussed in Chapter 6). Since the robot will have at least three legs down on the ground at an instance of time, and any more legs down will simply minimize the amount of torque we need in our joint servos, this three-

legs-down assumption provides a good safety factor. As such, the Jacobian [10] matrix used in our calculation needs to be set up for the rotation points (leg joints) on the robot leg. In figure 2.5, a diagram of the robot from the front is depicted (only two legs are shown to minimize complexity of the diagram). The leg joints are rotating about the z-axis in the diagram (which is coming out of the page). The mass of the robot body when it is standing on its legs is represented by the box hanging from the center of the picture. The diagram represents the hip rotational joint ( $R_2$ ) and knee rotational joint ( $R_3$ ), as well as the foot of the robot, which also can rotate (slip) and is shown in the calculations. The pelvic joint isn't needed in this calculation since it rotates perpendicular to the z axis (about the x axis).



*Figure 2.5: Byrobot torque calculation diagram, where L1, L2, and L3 represent leg segment lengths.*

The Jacobian matrix is a 6x3 matrix represented by:

$$J = \begin{bmatrix} P(O,2) \times Z_1 & P(O,3) \times Z_2 & P(O,F) \times Z_3 \\ Z_1 & Z_2 & Z_3 \end{bmatrix} \quad (2.1)$$

Where  $P(O,2)$  is a 3x1 position vector matrix that represents the distance from the robot center of mass to the  $R_2$  revolute joint (hip) in the x, y, and z direction.  $P(O,3)$  is the same using the distance from the robot center of mass to  $R_3$  (knee joint), and  $P(O,F)$  is the distance from the center of mass to the Foot. We are including the robot foot as a “revolute joint” because it can still rotate due to the center of mass load on the robot body, even though there is no servo motor there to act as an actuator. The 3x1 matrices  $Z_1, Z_2$ , and  $Z_3$  are the unit vectors of the z-direction. With this constraint, Jacobian matrix becomes:

$$J = \begin{bmatrix} 3X_o \times Z_o & (3.0 + 5.5)X_o \times Z_o & (8.5X_o - 6Y_o) \times Z_o \\ Z_o & Z_o & Z_o \end{bmatrix} \quad (2.2)$$

Where  $Z_1, Z_2$ , and  $Z_3$  are equal to  $Z_o$ , which is the unit vector of the Z-axis of the robot body, extending straight out of the page. Putting the P-distance vectors and  $Z_o$  -vectors into matrices, we can compute the Jacobian below:

$$J = \begin{bmatrix} \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 8.5 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 8.5 \\ -6 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -6 \\ -3 & -8.5 & -8.5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.3)$$

For a planar case, such as this, we can remove the 3 rows of zeros in the middle of the Jacobian to simplify it into a 3x3 matrix.

$$J = \begin{bmatrix} 0 & 0 & -6 \\ -3 & -8.5 & -8.5 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.4)$$

To find the loads in each of the joints, the wrench is needed. A wrench is a 6x1 that includes a force (x, y, and z components) and a moment (x, y, and z components).

$$wrench = W = \begin{bmatrix} x_{force} \\ y_{force} \\ z_{force} \\ x_{moment} \\ y_{moment} \\ z_{moment} \end{bmatrix} \quad (2.5)$$

For a planar wrench, there are only force components in the x and y directions, and a moment in the z direction (which is the rotation of the joint about the z-axis). This simplifies the wrench into a 3x1 matrix.

$$W = \begin{bmatrix} x_{force} \\ y_{force} \\ z_{force} \\ x_{moment} \\ y_{moment} \\ z_{moment} \end{bmatrix} = \begin{bmatrix} x_{force} \\ y_{force} \\ z_{moment} \end{bmatrix} \quad (2.6)$$

For our robot, the robot will only contain a value of a force in the -y direction, which is the weight of the robot center of mass. There is no force in the x direction. Since we are trying to find the torque required in the joints to maintain static equilibrium, we want to find the torque required in the joints that will counter the rotation being caused by the robot center of mass. Therefore, we set the value of the moment in the z direction to be zero, so there will be no rotation about the z-axis for our calculation, maintaining this static equilibrium.

The robot mass is approximated at 60oz. For the robot having a minimum of three legs down, we assume this 60oz mass is evenly distributed among the three legs. Hence, each leg supports approximately 20oz. This 20oz is the force in the -y direction that a robot leg will support, so our wrench vector becomes:

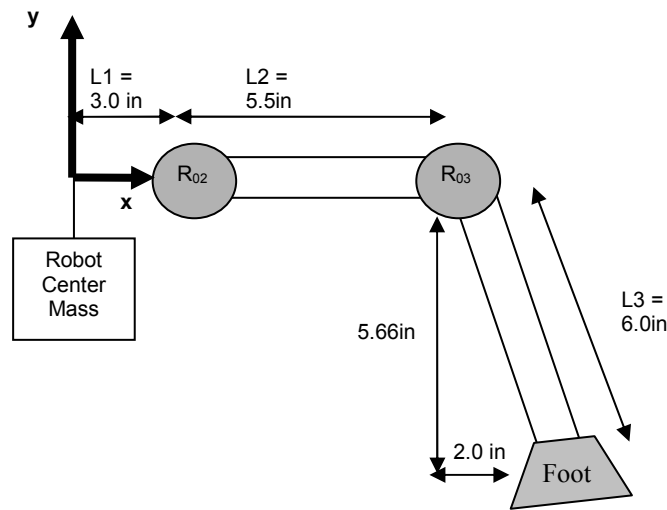
$$W = \begin{bmatrix} x_{force} \\ y_{force} \\ z_{force} \\ x_{moment} \\ y_{moment} \\ z_{moment} \end{bmatrix} = \begin{bmatrix} x_{force} \\ y_{force} \\ z_{moment} \end{bmatrix} = \begin{bmatrix} 0 \\ -20 \\ 0 \end{bmatrix} \quad (2.7)$$

We can multiply the Jacobian transposed, times the wrench, and find the joint loads at the hip and knee revolute joints.

$$Q = J^T W = \begin{bmatrix} 0 & 0 & -6 \\ -3 & -8.5 & -8.5 \\ 1 & 1 & 1 \end{bmatrix}^T \begin{bmatrix} 0 \\ -20 \\ 0 \end{bmatrix} = \begin{bmatrix} 60 \\ 170 \\ 170 \end{bmatrix} \quad (2.8)$$

This shows us that we will need a torque of at least 60oz-in at the hip revolute joint and at least 170oz-in at the knee joint.

But how do we know that Figure 2.5 represents the best orientation of the robot leg? Maybe if we put the foot at a more outward position, we could use servo motors that supply less torque? Here we considered moving the robot legs more outward and calculated the torque the same way. We don't want to have the robot leg moved inward for a "bow-legged" robot, since this would reduce the stability of robot. In figure 2.6, we see the robot leg in a different orientation, to see what effect this would have in the joint loads:



*Figure 2.6: Alternate Byrobot torque calculation diagram, assuming we have the foot expanded in a different position.*

We calculate the new Jacobian the same way; only we now have different values in the last column due to the new position distance vector of the robot foot.

$$J_{new} = \begin{bmatrix} \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} x \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 8.5 \\ 0 \\ 0 \end{bmatrix} x \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 10.5 \\ -5.66 \\ 0 \end{bmatrix} x \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -5.66 \\ -3 & -8.5 & -10.5 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.9)$$

Using the same wrench vector, we calculate the new robot joint loads:

$$Q_{new} = J_{new}^T W = \begin{bmatrix} 0 & 0 & -5.66 \\ -3 & -8.5 & -10.5 \\ 1 & 1 & 1 \end{bmatrix}^T \begin{bmatrix} 0 \\ -20 \\ 0 \end{bmatrix} = \begin{bmatrix} 60 \\ 170 \\ 210 \end{bmatrix} \quad (2.10)$$

This shows us that more torque load will be applied in the foot of the robot with this new orientation. The load in the robot foot has increased from 170oz-in to 210oz-in, concluding that the farther out we place the robot leg, the more load will be placed on the foot, giving more of a possibility that the foot could slip and the robot would lose stability. As such, it is best to keep the robot leg at the 90° angle used in our first calculations to minimize torque.

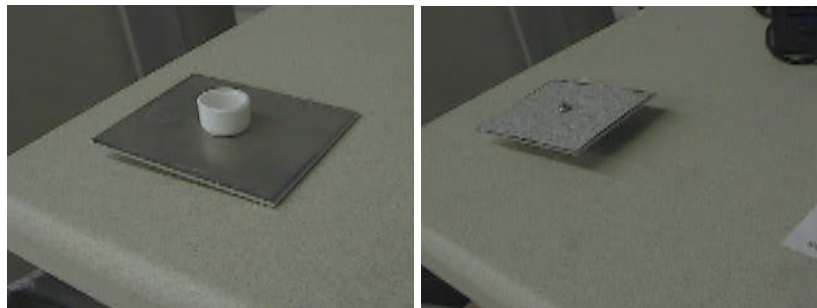
To this end we decided to use servos that had torque-ratings that were **twice as high** as these values, to ensure stability was maintained when adding additional load caused by robot sensors, cameras, and other components, as well as ensuring stability when the robot is walking. Having a bit more than the required torque in our servo motors at these joints is beneficial, but too little torque provided in the joints would cause the robot to not be able to stand up and collapse. To this, we chose our knee joint servos



to have a rating of about 130 oz-in at 6.0V, and our hip joint servos to have a rating of about 333 oz-in at 6.0V.

## 2.5 Byrobot Mars Shoes

When Byrobot is put into an unnatural terrain such as mud or sand, the feet on the robot can easily sink into the sand due to the amount of force put on it by the robot body. The small surface area of the feet can only sustain so much force before it starts to “dig” into the surface. As such, a set of “Mars shoes” was needed to prevent this situation from occurring. The Mars shoes needed to increase the surface area of the robot feet, as well as increase friction between the shoes and the surface. The Mars shoes were designed simply by constructing a 3.5” x 3” x 1/16” piece of sheet metal to be used for each of the six shoes. Sand paper was placed on the bottom for increasing the coefficient of friction in the sand pit, since the shoes could easily slip while walking in the sand. A rubber stopper is placed on the top of the sheet metal, and is attached with a screw and nut through the center, as well as super glue between the stopper and metal for a permanent fixture. The shoe can now easily be slid on and off the robot feet. A picture of the Mars Shoes is shown in Figure 2.7.



*Figure 2.7: Top and bottom view of Mars Sand Shoe*

Figure 2.8 shows the Mars shoes on the feet of Byrobot. Byrobot is now ready to walk through the Mars sand!



*Figure 2.8: Mars Sand Shoes in place on Byrobot*

## **CHAPTER 3**

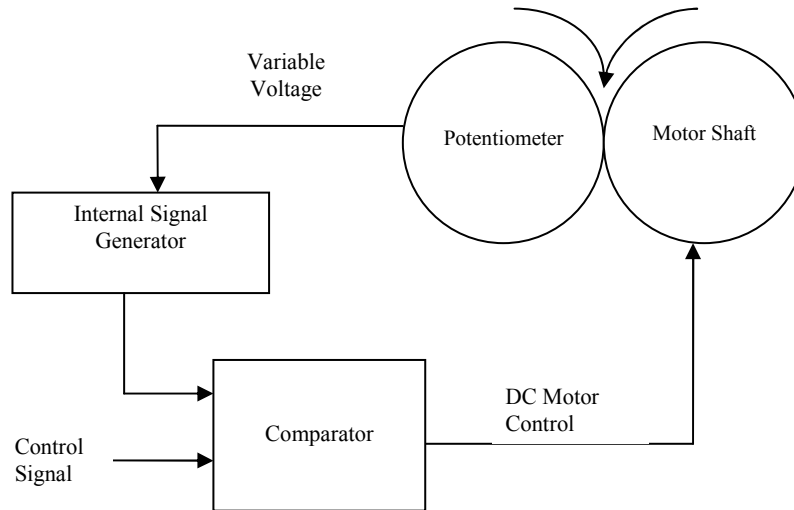
### **BYROBOT CONTROL**

#### **3.1 Servo Motors**

A common motor used in robotics is a servo motor. Instead of providing constant rotation, like most motors, servos are used for precise angular positioning but are typically limited to only 180° of maximum rotation. Servos are common in devices such as radio-controlled cars to control steering, radio-controlled air-planes to control rudders, or even in the cruise control systems of cars. Servos are ideal for applications requiring absolute positioning of a motor shaft.

##### **3.1.1 Servo Motor Control**

Microcontrollers are an excellent and inexpensive device for controlling servos [11]. In order to properly control a servo with a microcontroller, it is necessary to apply a few techniques, such as properly generating a control signal for a servo. Servos may be purchased in a prepackaged form, or specialized servos can be built using a few common components. Internally, a servo can be thought of as a direct current (dc) motor (which rotates an external motor shaft but provides no process to determine the amount of rotation) with a built-in controller. The control circuitry compares an angular position, determined by a control signal, to the current position of the motor shaft (as shown in Fig. 3.1).



*Figure 3.1: Inside of a Servo [12]*

The motor shaft's angular position is often determined by a potentiometer, which is rotated by the motor shaft. A potentiometer is a three-terminal resistor whose center connection has variable resistance, usually controlled by a slider or dial. The potentiometer acts as a variable voltage divider. The voltage from the center connection of the potentiometer represents the angular position the motor shaft is in. Other methods to determine angular position and rotation exist for larger servos, but a potentiometer is the most common for small servos. The built-in controller generates an internal signal from the voltage controlled by the potentiometer, compares it to the control signal, and then provides power to the dc motor to rotate the shaft in the appropriate direction to match the two. Servos usually require a pulse-width-modulated control signal.

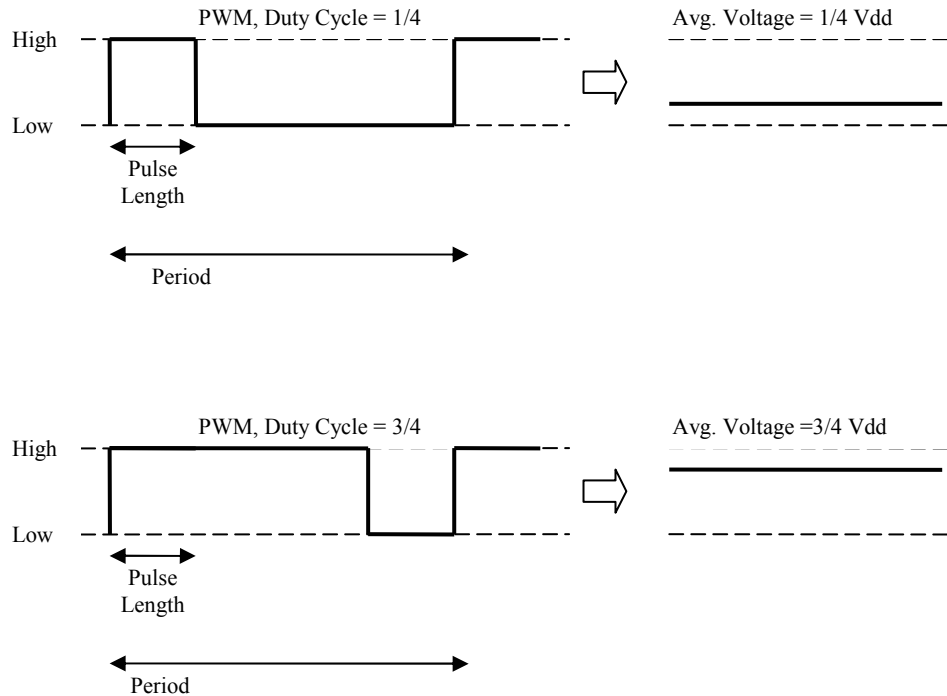
### **3.1.2 Pulse Width Modulation (PWM)**

Often, when controlling an analog device, the ability to drive a signal with variable power ( $P = I * V$ ) is needed [12]. For example, you may want to adjust the

speed of a dc motor or dim a light-emitting diode (LED). This can be a challenge when the signal is generated by a digital device. Different methods to convert a digital signal to an analog signal exist, one of which is a digital-to-analog converter. Using a converter adds complexity to a project, so generating a variable power signal with existing circuitry is desirable to reduce the number of components.

A simple method to vary the power using a digital signal, when an analog signal isn't available, is by using a method called pulse-width modulation (PWM) [12]. Instead of controlling the current or voltage of a signal, a pulse-width-modulated signal works by repeatedly pulsing the digital signal high and low at a fast rate. When sufficiently fast, the signal creates an effective average voltage. A shorter PWM period (the length between the rising edges in the signal) will create a cleaner average voltage, because the signal is effectively less “jittery” (i.e., less discharge from the capacitance in the line is needed to smooth the signal), but the minimum period will be limited by the speed of the device generating the signal. The period of the PWM signal is usually constant for a given application, and the high pulse width (the duration of the signal being driven high within one period) is usually variable, so that the average voltage of the signal can be changed. The ratio of high pulse width to period of the signal is called the *duty cycle*. By varying the duty cycle, you can vary the average voltage, as shown in Fig. 3.2.

The power through a device is proportional to the voltage supplied. Therefore, to decrease the power usage of a device (to dim an LED or to slow a motor); the duty cycle of the PWM signal should be decreased. A PWM signal can be used to limit the power to a device to save energy. This technique is used in many portable devices which have limited battery power.



*Figure 3.2: PWM and Average Voltage*

Some devices, such as servos, do not rely on the power of the signal limited by PWM but instead use the width of the high pulses to transmit information. This is also used by infrared remote controls to transmit data to control a television or radio. Pulse-width-modulated signals may be generated from many digital devices, even ones as simple as an inexpensive timer integrated circuit (such as the 8-pin 555 timer). A versatile yet inexpensive solution for many robotics hobbyists is to use a microcontroller for PWM generation. Using a micro controller has the added advantage of containing all of the control circuitry (needed for analyzing and responding to input) for a simple robot on a single chip.

### 3.1.3 PWM Signal Generation

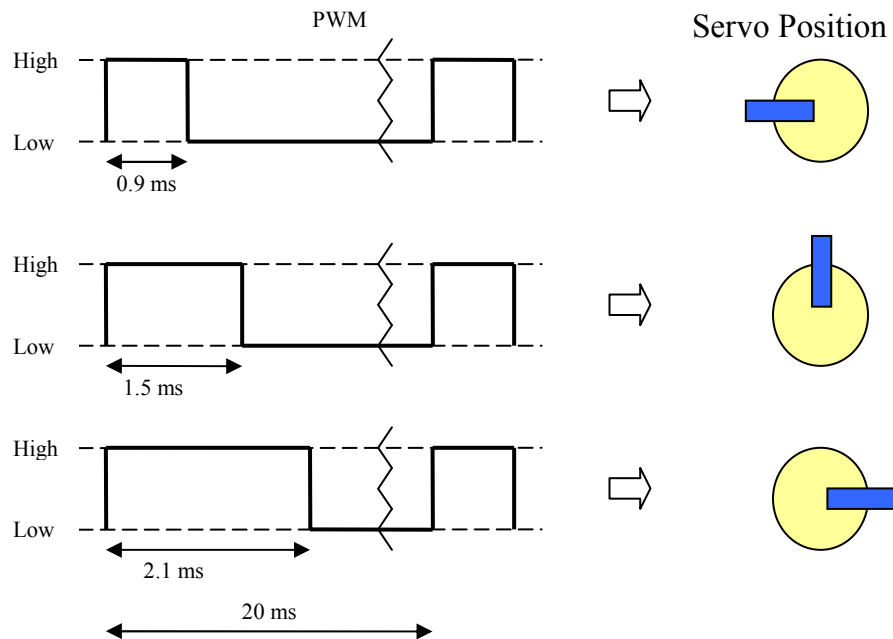
Pulse-width-modulated signal generation is easy to implement on microcontrollers. All microcontrollers will be able to generate a PWM signal, but the more expensive and elaborate ones provide hardware to make PWM generation easier, freeing up more processor time to run other tasks.

The simplest but most processor-intensive method to create a PWM signal is manually comparing a “count” to a variable that describes how long the high pulse width should be. When the count is less than the pulse width variable, the PWM signal is driven high. Otherwise, it is driven low. After the PWM period has elapsed, the count can be reset and the process started over. The PWM period will be the same as the time it takes your microprocessor to run your code. To increase the length of the PWM period, loops can be used to create delay.

The least processor-intensive method is to use a built-in PWM module if your microcontroller has one, such as the controllers used on Byrobot. When enabled, the PWM module will automatically generate a PWM signal with a period and duty cycle specified in control registers on the chip of your controller, in this case, the ATMEGA8-16PI chip on the SSC-32 Servo Controller [13]. Depending on the microcontroller being used and the speed it is running at, the built-in PWM module might not support a large enough period needed for the device you are using, such as for a servo motor (which commonly has a period of 20ms). In that case one of the previously mentioned methods must be used to generate a longer PWM signal.

### 3.1.4 Controlling the Servos

Most servos, including the Hitec RCD USA, Inc. HS-322HD servo (which is used to control the knee joint on the Byrobot), which is demonstrated here, have three pins: power, ground, and a control signal. The control signal is a pulse-width-modulated input signal whose high pulse width determines the servo's angular position, shown in Fig. 3.3.



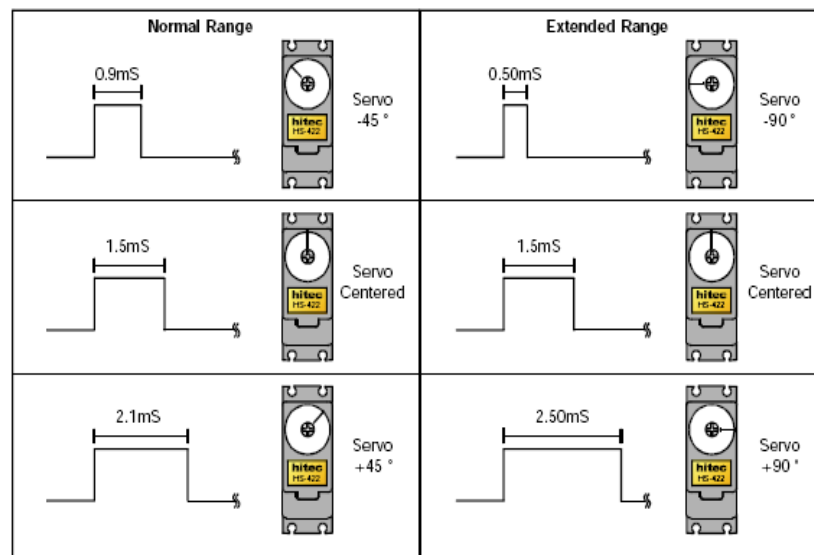
*Figure 3.3: Servo's High Pulse Width Determines the Angle Position*

Internally, the servo compares the PWM control signal to an internally generated signal, whose pulse widths are controlled by the potentiometer (which determines the shaft angle) and matches the pulse widths by rotating the motor shaft. For the HS-322HD, power can be between 4.8Vdc and 6.0V. Since the control signal (which draws a maximum of about 20mA of current) does not drive the motor directly, an additional benefit of using a servo is that the controller chip *can* control the signal directly. Most



motors draw more than 25mA of current for operation and, therefore, must be indirectly connected to the chip through a current amplifying device.

Typically servos require a PWM signal with a 20-ms period and a pulse width between 0.9-2.1 ms (0.9 ms corresponds to the minimum angle and 2.1 ms for the maximum angle); therefore the middle position is 1.5 ms (the average of the min/max pulse widths). The servo positions its output shaft in proportion to the width of the pulse, as shown below in Figure 3.4.



*Figure 3.4: Controlling the HS-322HD servo with PWM [12]*

The HS-322HD servo has a maximum angle of 180°. Servos only move a finite angular amount per cycle of the signal, so multiple cycles must be sent before the servo arrives at the correct angle. The speed/power at which the servo moves to a new position is proportional to the distance it needs to travel. So as the servo approaches the target angle, it will gradually slow. The servo will resist change away from the designated

angle as long as a signal is applied. Note that the servo's control mechanism will only engage when a signal is applied. If there is no signal, the servo's motor shaft is not driven by any circuitry and, hence, can be rotated freely, even when power is supplied to the servo.

Microcontrollers offer a simple and inexpensive solution for controlling servo motors for robotics and other electronics projects. Through the use of PWM, the angular position of the servo motor shaft can be conveniently controlled by a microcontroller for a variety of projects. PWM is an easy solution for the control of analog devices in other projects as well. Depending on the features included with the microcontroller used using, a PWM signal can be generated in a variety of ways.

### **3.2 Byrobot Open-Loop Control**

Byrobot is not exactly “autonomous” [14], since it does not have feedback sensors and a vision system to detect the type of terrain it is traversing, the slope of the terrain, where its center of mass is at all time for stability, nor its pitch and roll to know when its wheels might be stuck in sand or mud ( and not getting the desired output from the wheel motors). Byrobot executes directed motion commands using open-loop control for driving and a fixed walking gait. Hence, Byrobot does exactly what it is told in the program, all actions are scripted. Autonomous capabilities will be improved on in the future and details are mentioned in the final chapter of this thesis entitled “Future Work”.

For the programming of Byrobot, two microcontrollers used for hobby robotics purposes were incorporated, the Eyebot, which serves at the primary controller, and the SSC-32, which controls the 18 servo motors on the robot.

### 3.3 The SSC-32 Servo Controller

The SSC-32 Servo Controller from Lynxmotion [13] is a small preassembled serial servo controller with a number of relevant features. It allows control of up to 32 servos, which is plenty since we only need to control the 18 servos on our robot. The pulse width ranges from 0.50mS (milliseconds) to 2.50mS for providing a range of about 180°. A unique "Group Move" allows any combination of servos to begin and end motion at the same time, even if the servos have to move different distances. This is a very powerful feature for creating complex walking gaits for multi servo walking robots. The servo's position or movement can also be queried to provide feedback to the host computer. More specifications and technical data for the SSC-32 Servo Controller can be found in Appendix G.

In radio-control applications, a servo needs no more than a 90° range of motion, since it is usually driving a crank mechanism that can't move more than 90°. When pulses are sent within the manufacturer-specified range of 0.9 to 2.1mS, a corresponding range of motion is achieved. Most servos though have more than 90° of mechanical range. In fact, most servos can move up to 180° of rotation. The SSC-32 lets you use this extra range, which is important for walking our robot. A position value of 500 corresponds to 0.50mS pulse, and a position value of 2500 corresponds to a 2.50mS pulse. There is a linear relationship between the pulse width command sent to the SSC-32 and the actual servo angle. For instance, if the position values sent to the SSC-32 (500 to 2500) are the  $x$  values, and the actual servo angle (0° to 180°) are the  $y$  values. The *slope* of this linear line will be:

$$\frac{\Delta y}{\Delta x} = \frac{180 - 0}{2500 - 500} = \frac{180}{2000} = 0.09 \quad (3.3)$$

And the corresponding linear equation will be:

$$y - y_1 = m(x - x_1) \quad (3.4)$$

$$y - 0 = (0.09)(x - 500) \quad (3.5)$$

$$y = 0.09x - 45 \quad (3.6)$$

$y = 0.09x - 45$  is the linear equation relating the SSC-32 servo position values  $x$  (which is related to the pulse width) to the resultant servo angle  $y$ . A one unit change in position value produces a 1 $\mu$ S (microsecond) change in pulse width. The positioning resolution is 0.09°/unit (180°/2000). However, on the Byrobot, this equation is somewhat valid for only the servos at the pelvic and knee. Due to the orientation of the servos, to get them oriented correctly with the  $x$ ,  $y$ , and  $z$  axes, for a smoother forward and inverse kinematic analysis, a 90° rotation has to be subtracted from the equation. So the resultant equation for the pelvic ( $\theta_1$ , the HS-645MG Ultra Torque Hitec Servo) and knee ( $\theta_3$ , the HS-322HD Standard Deluxe Hitec Servo) servos is:

$$y = 0.09x - 45 - 90 \quad (3.7)$$

As for the hip servo( $\theta_2$ , the HSR-5995TG Ultra Torque Hitec Servo), only the pulse range from 1100 (1.1mS pulse) to 1900 (1.9mS pulse) could be used on the servo

controller. This still corresponds to a range of rotation of 180°. The *slope* of this linear line will be:

$$\frac{\Delta y}{\Delta x} = \frac{180 - 0}{1900 - 1100} = \frac{180}{800} = 0.225 \quad (3.8)$$

And the corresponding linear equation will be:

$$y - y_1 = m(x - x_1) \quad (3.9)$$

$$y - 0 = (0.225)(x - 1100) \quad (3.10)$$

$$y = 0.225x - 247.5 \quad (3.11)$$

So the resultant linear equation for the hip servos on each leg is:

$$y = 0.225x - 247.5 \quad (3.12)$$

The SSC-32 logic voltage, or electronics power input, is normally used with a 9vdc battery connector to provide power to the ICs and anything connected to the 5vdc lines on the board. There is a Low Dropout regulator onboard that will provide 5Vdc out with as little as 5.5Vdc coming in. This is important when operating the robot from a battery supply. It can accept a maximum of 9vdc in. The regulator is rated for 500mA, but was de-rated by the manufacturer to 250mA to prevent the regulator from overheating. The SSC-32 Servo Controller has 2 channels for its 32 servo ports (channel 1 has servo ports numbered 0-15, and channel 2 has servo ports numbered 16-31). The

Hitec servos used in the Byrobot operate between 4.8V and 7.4V. There are 3 options for powering the SSC-32 Servo Controller and the servos:

1. Use one battery, or other power supply, to provide the necessary power to both of the servo channels, and the logic power for the microprocessor chip.
2. Use one battery for the chip logic power supply, and another battery to control both servo channels.
3. Use one battery for the chip logic, another battery for the channel 1 servos, and another battery for the channel 2 servos.

Option 2 was chosen for this application, for it was determined to isolate the logic from the Servo Power Input, since when trying to power the microcontroller chip and the servos from the same power supply, the microcontroller may reset when many servos are moving simultaneously. Also, since we wanted to limit the load on the robot, we were concerned with the additional weight added when having two power supplies for the two channels.

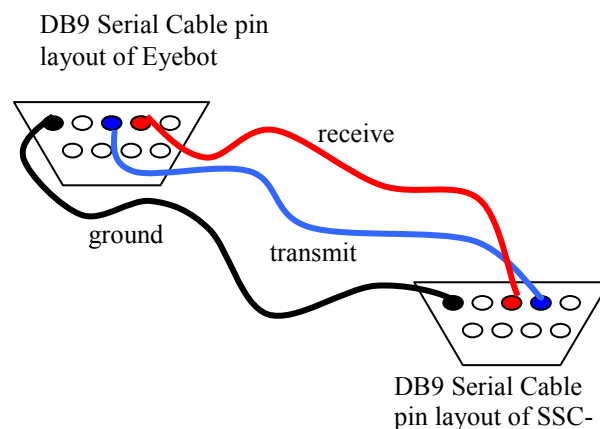
### **3.4 The Eyebot Controller**

The Eyebot [11] was used as the primary controller on the robot. It can be used for mobile robots with wheels, walking robots or flying robots. It consists of a powerful 32-Bit microcontroller board with a graphics display and a digital color camera allowing it to perform on-board image processing. The specifications of the Eyebot controller can be found in Appendix F.

Since there are only 2 motor driver ports on the Eyebot, and there are 4 motors (one on each wheel) on the Byrobot robot, the issue becomes how can we drive all 4

motors? To overcome this issue, the inputs were tied together to control two motors simultaneously from each motor port on the Eyebot. The front and back wheel motors on the left side of the robot were tied together, as well as the front and back wheel motors on the right side of the robot. Now all four motors can be driven from the two motor drivers on the controller! In order for the robot to turn, the right pair of wheels are commanded at one speed, and the left pair of wheels at another speed. The speed and direction of each wheel pair affects the radius of curvature that the robot turns, as described in the Wheeled Kinematics section of this thesis. Turning each pair of wheels at equal speeds and opposite directions will turn the robot in place, just as turning each pair of wheels at equal speeds and directions will drive the robot in a straight line.

In order to communicate between the Eyebot and the SSC-32 controller, a null modem was used to connect the serial DB9 communication ports on the controllers. This null modem simply crosses the transmit and receive lines in the DB9 communication line, so that each controller can send and receive data to and from the other controller, shown in Figure 3.6.



*Figure 3.5: DB9 Serial Cable Null Modem Connection*

This factor was important since the SSC-32 is used to operate the servos, while the Eyebot is needed to run the wheel motors, control future sensors and cameras on the robot, and for its clock for timing and delays. The Eyebot functions as the primary controller and all of the programs, done in C, are compiled and downloaded into it. The C code for the programs has commands that sends data out through the serial DB9 port into the SSC-32 Servo Controller, to control each of the Byrobot servos as desired.

### **3.5 Power**

Powering the Byrobot involves selecting the correct battery type, the correct voltage, and ensuring that the right amount of current is flowing at all times. Here we will discuss the process undertaken to power the robot.

#### **3.5.1 NiMH vs. NiCd Batteries**

When you plug a battery of X volts into any device, the actual volts the device sees is some fraction of X and depends on how high the current is, and what kind of batteries it is. So what is the best type of battery needed for optimal performance? For the Byrobot, the pros and cons were weighed against the two popular battery types NiCd and NiMH.

The advantage of NiMH cells is that for a given cell size, they have a higher capacity compared to NiCd cells. This means that the powered devices will work longer using NiMH cells. In addition, because they do not contain cadmium, NiMH batteries are more environmentally friendly. The disadvantage of NiMH cells is that they usually have much higher internal impedance. This means that if you try to draw a lot of current from



NiMH cells, they will drop excessively in voltage which can cause poor performance or cause the device they are powering to shut down. NiMH cells are also a little heavier than the same physical size NiCd cell.

However, the same life span can not be expected from a NiMH cell as compared to a NiCd cell. NiMH packs need to be replaced about twice as often as NiCd packs regardless of the manufacturer of the packs. Also, temperature extremes cause NiMH cells to lose their charge much more quickly than NiCd cells in very hot or cold climates. NiMH cells lose their charge two or three times faster than NiCd cells do. Also, NiMH cells shouldn't be charged at as high of a charge rate as a NiCd cell due to its higher internal impedance. So, if the device demands the highest possible capacity, NiMH cells and packs will work fine in most applications.

The big advantage of NiCd packs and cells is reliability. This is a mature technology that is practically "bullet proof." Thus, in critical applications, NiCds are the most reliable. In addition, NiCd cells have extremely low internal impedance which means a lot of current can be drawn without a corresponding excessive voltage drop. The NiCad batteries are less sensitive to a voltage drop caused by a large load. This makes them perfect for high current draw applications, such as the Byrobot. The down side to NiCd cells and packs is that they contain cadmium which is not environmentally friendly.

As an example, let's say we have a robot with a camera and a couple of servos attached to the controller. The camera requires 5 volts to work, and the servos can operate between 5V and 6V. The servos are drawing 1.5 amps of current from the batteries. In the NiMH case, instead of the servos seeing the 5V from the batteries, they may see about 4V, which means the camera is also seeing 4V and not able to function. It

also means that there is less power available for the servos to operate. With the NiCd batteries, which aren't as sensitive to this load, we can have 6V worth of batteries, and when the servos draw 1.5 amps of current, it still reads about 6V, which is much less of a voltage drop. Now, we are getting near-best performance from the servos, and the camera can operate in that range. The only drawback is that while you get less voltage drop of the batteries, the NiCd batteries last less than half as long as the NiMH batteries.

### **3.5.2 Servo Power**

For the SSC-32 Servo Controller, which controls the servo joints on the robot legs on the Byrobot, both NiMH and NiCd batteries were tested. First we used five 1.2V batteries to give a 6.0V input to the servos, which is the maximum servo operating voltage. However when we turned on all 18 of the servos in the control programs (such as when the robot stands on its legs, the legs retract, the robot walks, etc.), we measured the output voltage on a multimeter at only 3.8V! This was not the ideal condition. When the robot is walking, all of the servos will be used. Thus, the servos were only getting 3.4V from a 6.0V supply. The 2.6V drop was due to the higher internal impedance of the cells. Since the Byrobot draws a lot of current, these NiMH cells dropped excessively in voltage which caused poor performance of the servo device. One option we therefore decided to try was to add another battery, for a 7.2V input. We actually overcharged the batteries, so the initial input voltage was actually read at 7.9V. However, the output voltage with all the servo turned on was only 5.3V.

So the switch was made to NiCd batteries. We used six of the 1.2V batteries again, which were still overcharged to 7.9V. Then we turned on all eighteen of the robot

servos. The output voltage was 7.1V – less than one volt of a drop! The fact that the servos were all getting 7.1V, which is more than the 6.0V maximum operating voltage of some of the servos, was not a problem. It simply meant that the servos will rotate at a slightly greater speed.

### **3.5.3 Current**

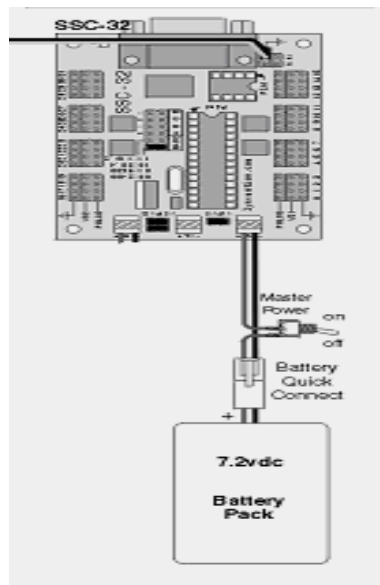
The current drawn by the 18 servos in Byrobot was a big issue that had to be addressed. The SSC-32 controller has two channels for servos, as previously mentioned. Each of these two channels can control 16 servos each, for a total of 32 servos that can be controlled by the SSC-32 controller. Each channel is capable of dealing with 15 amps of current. The Byrobot has 9 servos on each of these channels. Hence, channel 1 controls the 9 servos that control legs 1, 2, and 3, and channel 2 controls the 9 servos on legs 4, 5, and 6. There are three types of servos on the Byrobot. The pelvic, hip, and knee joint of each leg has a different type of servo. Here are a few of the specifications for the three servo types used:

1. Hitec HSR-5995TG Ultra Torque Servo (hip joint)
  - i. Operating voltage range : 4.8V – 7.4V
  - ii. Stall Torque at 7.4V (max voltage) : 416.61 oz-in
  - iii. Current drain at 7.4V: 380mA/idle, 5.2A at lock/stall
2. Hitec HS-645MG Ultra Torque Servo (pelvic joint)
  - i. Operating voltage range : 4.8V – 6.0V
  - ii. Stall Torque at 6.0V (max voltage) : 133.31 oz-in
  - iii. Current drain at 6.0V: 9.1mA/idle, 450mA at no load operating

### 3. Hitec HS-322HD Standard Deluxe Servo (knee joint)

- i. Operating voltage range : 4.8V - 6.0V
- ii. Stall Torque at 6.0V (max voltage) : 52 oz-in
- iii. Current drain at 6.0V: 7.7mA/idle, 180mA at no load operating

The Hitec HSR-5995TG Ultra Torque Servo at the hip joint draws the most current when it is operating. An amp-meter was also placed in series with the power supply output of the SSC-32 to measure the current being drawn by the batteries. The reading was 5 Amps of current when all the servos were operating. So what does this mean? One of the initial problems was determining which wires were used to connect the servo battery pack to the switch to the servo power ports on the SSC-32. To sufficiently provide enough current, the proper gauge wire must be used to connect from the “Battery Quick Connect” that you see in the Figure 3.7, to the switch, and into the servo power port.



*Figure 3.7: Connection of Servo Controller to Battery Pack[13]*

A wire has a resistance based on:

$$R = \frac{\rho * L}{A} \quad (3.13)$$

where  $\rho$  is the resistivity of the wire material,  $L$  is the wire length, and  $A$  is the cross-sectional area [15]. This resistance will limit how much current can flow through the wire, based on Ohm's Law,  $V=IR$ . Initially a thin 26-guage wire was used which couldn't handle the 5 Amps of current being drawn as Byrobot operated while walking, and thus, the robot wouldn't run for very long. The solution was to use a thicker 16-guage wire can handle up to 22 Amps of current, which is more than enough for the Byrobot.

The 5 amps of current is still a lot of current, and it will drain out the batteries!

Since the batteries and the battery pack options used on Byrobot were all rechargeable, it was a matter of determining how long they will last. Our first choice was to use a series of six 1.2V AA batteries in plastic battery holders to give the 7.2V power supply to the servos on the SSC-32 controller. The NiCd batteries, which was the first battery choice, have 1000 mAh (milli-amp hours) of battery life. So with 5A of current drawn we would have:

$$\frac{1000mAh}{5A} = \frac{1Ah}{5A} = 0.2hours = 12 \text{ min} \quad (3.14)$$

With the NiMH batteries, which have 2800 mAh of battery life, we would have:

$$\frac{2800mAh}{5A} = \frac{2.8Ah}{5A} = 0.56hours = 33.6 \text{ min} \quad (3.15)$$

So even though the NiCd batteries last a shorter period of time, they were well worth it since they have less of a voltage drop than the NiMH batteries, and there is no point of having long lasting batteries if they don't supply enough power to operate the robot. In the end, a Ni-Cd Vex battery pack with a capacitance of 2000mAh was used for Byrobot. As shown from calculation (3.16), a maximum of 24 minutes can therefore be extracted from the batteries during walking:

$$\frac{2000mAh}{5A} = \frac{2.Ah}{5A} = 0.40hours = 24 \text{ min} \quad (3.16)$$

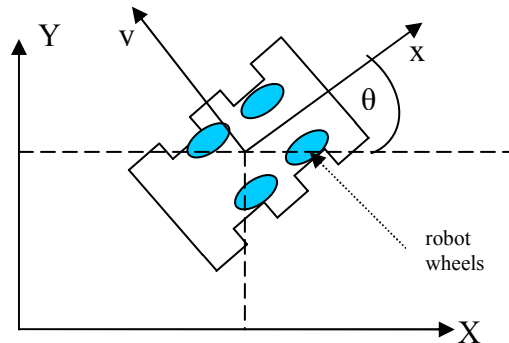
## CHAPTER 4

### WHEELED KINEMATICS

The wheeled kinematics are based on the parameters of linear and rotational speed, time, distance, and degrees the robot has rotated in the world coordinate frame. Based on this information, we will be able to know where the robot is, and direct it to go to any point in a plane.

#### 4.1 Robot Orientation

When the Byrobot is in its wheeled configuration, it can be defined as a nonholonomic system [16]. This generally means that Byrobot can not slide directly to the left or the right without slipping, due to its fixed wheel formation shown, as we see later in section 4.2. For a holonomic system, the wheels on the robot would have to be designed and/or oriented so that the robot can easily move to the side, and in all other directions without any slipping. Figure 4.1 shows what the 4-wheeled system looks like turned in the real world coordinate frame, with  $\theta$  being the number of radians turned.



*Figure 4.1: Robot orientation in world frame*

Relation between reference (world) frame is through standard orthogonal rotation transformation [17].

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

This is a standard rotation of the robot car around the z-axis (which is coming out of page in Figure 4.1).

The robot however, is actually controlling the front and rear wheel located on one side of the robot simultaneously. With this being the case, the robot reduces to Fig. 4.2, where the speed of the left and right wheels can be taken as one variable each. Here the Byrobot is differentially steered, in the two-dimensional (x, y) frame.

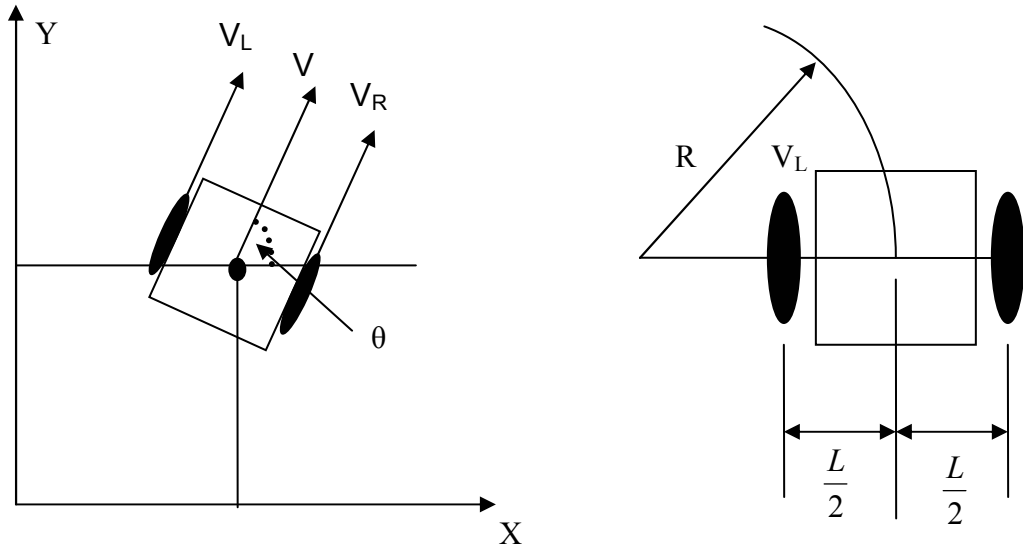


Figure 4.2: Reduced orientation of robot.



In figure 4.2, the linear velocity of the left and right wheels is represented by  $V_L$  and  $V_R$  respectively,  $V$  is the average linear velocity of the robot,  $\theta$  represents the degrees the robot has turned in the world frame,  $R$  is the radius of curvature of the robot, and  $\frac{L}{2}$  is half the horizontal distance between the right and left wheels. As the robot is rolls around on the floor or ground, it simply moves along the x and y axes in the world frame, and rotates about the z-axis as it turns.

## 4.2 Differential Wheeled Drive Kinematics

The steering is done in this 4 wheel differential drive by simultaneously rotating the left pair of wheels in one direction at one speed, and the right pair of wheels at a different speed and/or direction [18].

The states are given as:  $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ , where x is the position of the robot along the x-axis,

y is the position of the robot along the y-axis, and  $\theta$  is the radians the robot has turned in the coordinate frame (hence, rotation about the z-axis). The Byrobot uses a fixed wheel formation for the four wheels on its body. The velocity of a point p on the fixed wheel is given by:

$$V_p = (r \times \omega) \hat{x} \quad (4.2)$$

Where r is the radius of the wheel,  $V_p$  is the linear velocity of the wheel,  $\omega$  is the rotational speed of the wheel, and  $\hat{x}$  is the unit vector to the x-axis. A restriction to the

robot mobility is that the point p cannot move to the direction perpendicular to the plane of the wheel, shown in Figure 4.3.

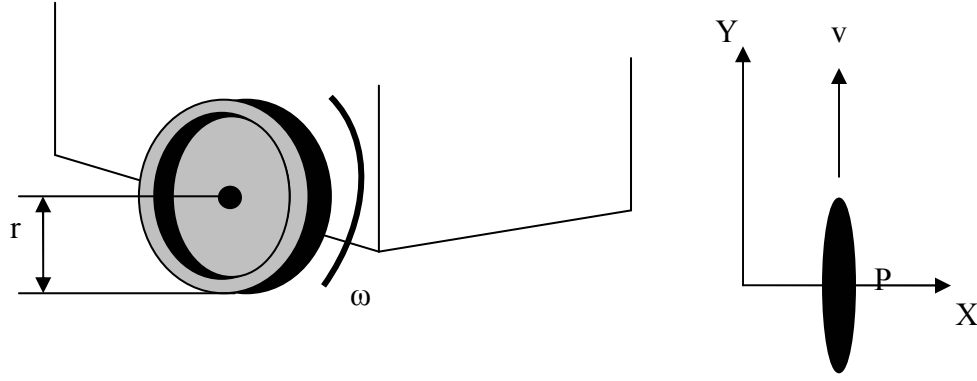


Figure 4.3: Fixed wheel formation, wheels on robot body

By taking the derivatives of our state variables,  $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ , we can find the robot velocity in the

x direction given by  $\frac{dx}{dt}$ , the velocity in the y direction given by  $\frac{dy}{dt}$ , and the rotational

speed given by  $\frac{d\theta}{dt}$ . Here we take the derivative of the state variables to find:

$$\frac{dx}{dt} = V \cos \theta = \left( \frac{\omega_R \omega_L}{2} \right) r * \cos \theta = \left( \frac{V_R V_L}{2} \right) \cos \theta \quad (4.3)$$

$$\frac{dy}{dt} = V \sin \theta = \left( \frac{\omega_R \omega_L}{2} \right) r * \sin \theta = \left( \frac{V_R V_L}{2} \right) \sin \theta \quad (4.4)$$

Where R and L labels the right and left wheels respectively. We can put this into matrix form using combining the right and left wheel equations:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} V \cos \theta & V \sin \theta \end{bmatrix} * \begin{bmatrix} \theta_R \\ \theta_L \end{bmatrix} = \begin{bmatrix} \left(\frac{1}{2}\right)r * \cos \theta & \left(\frac{1}{2}\right)r * \cos \theta \\ \left(\frac{1}{2}\right)r * \sin \theta & \left(\frac{1}{2}\right)r * \sin \theta \\ \left(\frac{r}{L}\right) & -\left(\frac{r}{L}\right) \end{bmatrix} * \begin{bmatrix} \theta_R \\ \theta_L \end{bmatrix} \quad (4.5)$$

Where  $\dot{x} = \left(\left(\frac{1}{2}\right)r * \cos \theta\right)\theta_R + \left(\left(\frac{1}{2}\right)r * \cos \theta\right)\theta_L$  , gives you the speed of the robot with respect to the x-axis,

$\dot{y} = \left(\left(\frac{1}{2}\right)r * \sin \theta\right)\theta_R + \left(\left(\frac{1}{2}\right)r * \sin \theta\right)\theta_L$  , gives you the speed of the robot with respect to the y-axis,

$\dot{\theta} = \left(\frac{r}{L}\right)\theta_R - \left(\frac{r}{L}\right)\theta_L$  , gives you the rotational speed of the robot (about the z-axis)

Since the robot is in a differential drive configuration, where we can simply vary the speed of each individual wheel, we can find general kinematic values for the robot, based on knowing other values, such as distance the robot traveled (d), its average linear velocity (v), angular velocity of the wheels ( $\omega$ ), or degrees turned in world coordinate frame ( $\theta$ ). Using the measured elapsed time (t), the radius of the robot wheels (r), and the horizontal distance between the left and right wheels (L), and depending on which of the previously mentioned variables that we know, we can find our kinematic values for the robot using a few differential equations:

$$d = Vt = \omega r t \quad (4.6)$$

$$V = \frac{V_R + V_L}{2} \quad (4.7)$$

$$\theta = \frac{d_R - d_L}{L} = \frac{(V_R - V_L)t}{L} = \frac{(\omega_R r - \omega_L r)t}{L} = \frac{(\theta_R r - \theta_L r)}{L} \quad (4.8)$$

$$V_R = \omega \left( R + \frac{L}{2} \right), \quad V_L = \omega \left( R - \frac{L}{2} \right) \quad (4.9)$$

$$\omega = \frac{d\theta}{dt} = \frac{V_R - V_L}{L} \quad (4.10)$$

.

### 4.3 Radius of Curvature of Robot

The Radius of Curvature of the robot,  $R$ , can be found by looking at the previous figures and equation:

$$R = \left( \frac{L}{2} \right) \left( \frac{V_R + V_L}{V_R - V_L} \right) \quad (4.11)$$

We see that when  $V_R = V_L$ , then  $R = \infty$  meaning there is infinite radius of curvature and the robot goes in straight motion. When  $V_R = -V_L$ ,  $R = 0$  meaning there is no radius of curvature and the robot rotates in place (spins on a dime). The radius of curvature can be used for the robot to drive around obstacles. For example, if there was a crater ahead of the robot and we knew its radius or diameter, we could set the velocities of the robot wheels accordingly so that the robot would avoid the crater by circling around it.

## CHAPTER 5

### LEGGED KINEMATICS

The forward and reverse kinematic analysis is formulated for each 3-DOF 3R leg mechanism in order to develop the overall kinematic model of a six-legged walking robot.

#### 5.1 Forward Kinematics – Geometrical Method

The Geometrical Method of the Forward Kinematics [19] is the easiest method when dealing with small degrees of freedom such as our 3R legged mechanisms. We can simply look at the legs in the  $x, y, z$  coordinate frame and be able to determine the final foot position based on leg segment lengths and angles. Figure 5.1 shows this kinematic model of one of the robot legs:

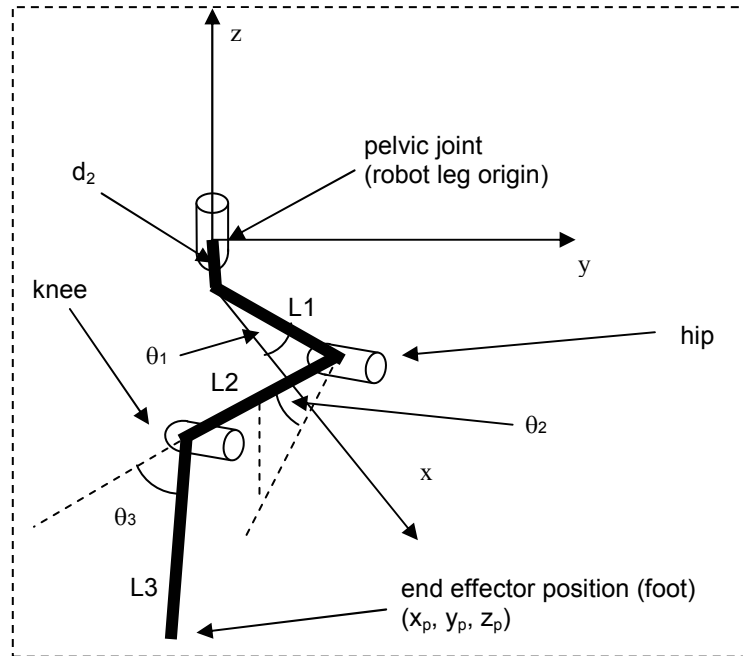


Figure 5.1: Kinematic Model of our 3R leg mechanism on Byrobot

From the model in figure 5.1, we can determine the following equations for the position of the robot hip, knee, and foot:

### **Hip Position**

$$X_1 = L_1 \cos(\theta_1) \quad (5.1)$$

$$Y_1 = L_1 \sin(\theta_1) \quad (5.2)$$

$$Z_1 = 0 \quad (5.3)$$

### **Knee Position**

$$X_2 = L_1 \cos(\theta_1) + L_2 \cos(\theta_2) \sin(\theta_1) \quad (5.4)$$

$$Y_2 = L_1 \sin(\theta_1) + L_2 \cos(\theta_2) \cos(\theta_1) \quad (5.5)$$

$$Z_2 = L_2 \sin(\theta_2) - d_2 \quad (5.6)$$

### **Foot Position (End Effector)**

$$X_3 = L_1 \cos(\theta_1) + L_2 \cos(\theta_2) \sin(\theta_1) + L_3 \cos(\theta_2 + \theta_3) \cos(\theta_1) \quad (5.7)$$

$$Y_3 = L_1 \sin(\theta_1) + L_2 \cos(\theta_2) \cos(\theta_1) + L_3 \cos(\theta_2 + \theta_3) \sin(\theta_1) \quad (5.8)$$

$$Z_3 = L_2 \sin(\theta_2) - L_3 \sin(\theta_2 + \theta_3) - d_2 \quad (5.9)$$

## **5.2 Forward Kinematics – Denavit-Hartenberg Convention**

The standard method for the forward and reverse kinematics of mechanisms is by using what is known as the Denavit-Hartenberg Parameters [20]. Denavit and Hartenberg (DH) proposed a matrix method of systematically assigning coordinate systems to each link of an articulated chain. These are a way to define the lengths of your mechanism segments, distances, and joint angles, commonly used for robot manipulators. The axis of revolute joint  $i$  is aligned with  $Z_{i-1}$ . The  $Z_{i-1}$  axis is directed along the normal from  $Z_{i-1}$  to  $Z_i$  and for intersecting axes is parallel to  $Z_{i-1}$  crossed with  $Z_i$ . The link and joint parameters may be summarized as:

- $a_i$  = the distance from  $Z_i$  to  $Z_{i+1}$  measured along  $X_i$
- $\alpha_i$  = the angle between  $Z_i$  and  $Z_{i+1}$  measured about  $X_i$

- $d_i$  = the distance from  $X_{i-1}$  to  $X_i$  measured along  $Z_i$
- $\theta_i$  = the angle between  $X_{i-1}$  and  $X_i$  measured about  $Z_i$

For a revolution axis  $\theta_i$  is the joint variable and  $d_i$  is constant, while for a prismatic joint  $d_i$  is variable, and  $\theta_i$  is constant. The Denavit-Hartenberg (DH) representation results in a 4x4 homogeneous transformation matrix,  $T$ :

$$T_{i-1}^i = \begin{bmatrix} \cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.10)$$

$$T_0^i = T_0^{i-1} T_{i-1}^i \quad (5.11)$$

In this 4x4 matrices, the last column represents the **position vector**, in the x, y, and z position respectively. The upper left 3 rows and 3 columns of this transformation matrix [21],  $T$ , is the rotational matrix, which rotates the joints from their respective coordinate frame,  $i-1$ , to the new coordinate frame,  $i$ .

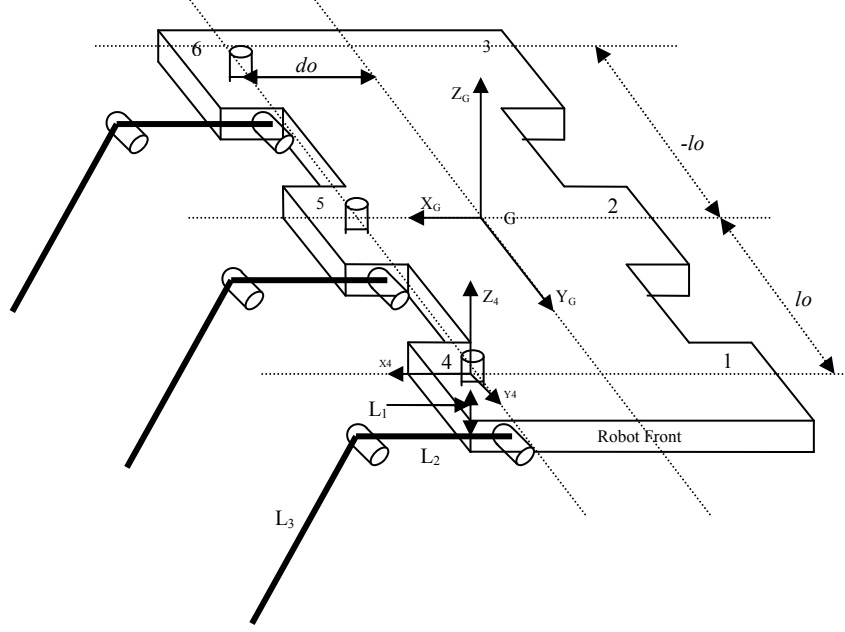


Figure 5.2: Byrobot's body diagram; only left 3 legs are shown for simplicity

So to transform from the robot body origin (G) to the origin of the robot legs (O), we use the transformation matrix  $T_o^G$ :

$$T_o^G = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & -do \\ 0 & 1 & 0 & li \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow i = \text{legs\#1,2,3} \\ \begin{bmatrix} 1 & 0 & 0 & do \\ 0 & 1 & 0 & li \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow i = \text{legs\#4,5,6} \end{cases} \quad (5.12)$$

$$T_o^G = \begin{cases} \begin{bmatrix} 1 & 0 & 0 & do \\ 0 & 1 & 0 & li \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow i = \text{legs\#4,5,6} \end{cases} \quad (5.13)$$

Where  $li$  is the distance from the robot body origin, to the origin of leg  $i$ .

From figure 5.2, we can see the following:



For leg 1 and leg 4 ( $l_1$  and  $l_4$ ),  $l_1 = l_4 = l_o$

For leg 2 and leg 5 ( $l_2$  and  $l_5$ ),  $l_2 = l_5 = 0$

For leg 3 and leg 6 ( $l_3$  and  $l_6$ ),  $l_3 = l_6 = -l_o$

Now, using the DH Convention, we can determine our transformation matrices for the legs of Byrobot. Transforming from the leg origin, to joint 1 (pelvic joint), to joint 2 (hip joint), to joint 3 (knee joint), and finally to the final foot position (we will call this joint position 4), which will give us the location of the foot based on the robot leg origin. The angles in our matrices are controlled by the servo motor angles that operate the joints in the robot legs. Each leg has 3 servo motors which corresponds to 3 revolute joints, hence, rotation about the z-axis for each respective joint.

For clarity, we use  $S_i$  and  $C_i$  to denote  $\sin(\theta_i)$  and  $\cos(\theta_i)$ , and we use  $S_{i+j}$  and  $C_{i+j}$  to denote  $\sin(\theta_i + \theta_j)$  and  $\cos(\theta_i + \theta_j)$ , respectively. Also, we denote  $L_i$  as the length of leg segment  $i$  on each robot leg.

$$T_o^1 = \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.14)$$

$$T_1^2 = \begin{bmatrix} C_2 & -S_2 & 0 & L_1 \\ 0 & 0 & -1 & 0 \\ S_2 & C_2 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.15)$$

$$T_2^3 = \begin{bmatrix} C_3 & -S_3 & 0 & L_2 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.16)$$

$$T_3^4 = \begin{bmatrix} 1 & 0 & 0 & -L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.17)$$

The transformation matrix for a joint with respect to another joint, is found by simply multiplying all the transformation matrices used sequentially [22]. So if we want to know the transformation matrix to the hip joint (joint 2), with respect to the leg origin, we would have:

$$T_0^2 = T_0^1 T_1^2 = \begin{bmatrix} C_1 C_2 & -C_1 S_2 & S_1 & L_1 C_1 \\ S_1 C_2 & -S_1 S_2 & -C_1 & L_1 S_1 \\ S_2 & C_2 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.18)$$

And likewise for the knee joint (joint 3):

$$T_0^3 = T_0^1 T_1^2 T_2^3 = \begin{bmatrix} C_1 C_{2+3} & -C_1 S_{2+3} & S_1 & L_1 C_1 + L_2 C_2 S_1 \\ S_1 C_{2+3} & -S_1 S_{2+3} & -C_1 & L_1 S_1 + L_2 C_2 C_1 \\ S_{2+3} & C_{2+3} & 0 & L_2 S_2 - d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.19)$$

And finally, the transformation matrix going from the leg origin to the final foot position (known as the end-effector for robot manipulators), is:

$$T_0^4 = T_0^1 T_1^2 T_2^3 T_3^4 = \begin{bmatrix} C_1 C_{2+3} & -C_1 S_{2+3} & S_1 & L_1 C_1 + L_2 C_2 S_1 + L_3 C_1 C_{2+3} \\ S_1 C_{2+3} & -S_1 S_{2+3} & -C_1 & L_1 S_1 + L_2 C_2 C_1 + L_3 S_1 C_{2+3} \\ S_{2+3} & C_{2+3} & 0 & L_2 S_2 - L_3 S_{2+3} - d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.20)$$

Hence, to find the transformation matrix,  $T_G^4$ , going from the robot body origin (G) down to the final foot location for a robot leg (position 4), we multiply the matrix  $T_G^O$  by the matrix  $T_0^4$  (or by all four of the smaller T matrices that transform the entire leg). In other words:

$$T_G^4 = T_G^O T_0^4 = T_G^O T_0^1 T_1^2 T_2^3 T_3^4 \quad (5.21)$$

$$T_G^4 = \begin{bmatrix} C_1 C_{2+3} & -C_1 S_{2+3} & S_1 & L_1 C_1 + L_2 C_2 C_1 + L_3 C_1 C_{2+3} \pm d_o \\ S_1 C_{2+3} & -S_1 S_{2+3} & -C_1 & L_1 S_1 + L_2 C_2 S_1 + L_3 S_1 C_{2+3} + l_i \\ S_{2+3} & C_{2+3} & 0 & L_2 S_2 - L_3 S_{2+3} - d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where the  $\pm d_o$  term depends on which set of legs that we are referring to (which side of the robot, legs 1-3 or legs 4-6), where the sign for  $d_o$  would change, as shown previously. Note that we are still using the same naming convention as previously for the  $l_i$  term. Remember that in each of these 4x4 matrices, the last column represents the **position vector**, in the x, y, and z position respectively [23]. Here we can see that the final position vectors from each transformation (to the hip, knee, and foot), are equivalent to the position found from the geometrical method above!

### 5.3 Inverse Kinematics

The inverse kinematics can be used to find what the joint angles should be in the legs given the position of the foot [23]. This is important for the walking gait of Byrobot. For one of the robot legs to take one step, it does a sequence of:

1. leg lifts up in the air a certain distance (Z direction)
2. leg moves forward (Y direction)
3. leg comes back down (Z direction)

If the legs need to swing out, this movement is done in the Y-direction for the robot. The three servo motors in each leg control the joint actuation for the knee, hip, and pelvic joint. We want to know what the angles are in these servo motors (our  $\theta$ 's). At the end of each walking sequence, the robot returns to its original stable stance, where all six legs are down on the ground. By determining the foot positions for this stable stance, we can program the servo angles accordingly as well. By knowing the position of the foot at each moment (in the X, Y, and Z direction), we can find the corresponding joint angles and use this information for the chosen walking gait.

A single set of three joint angles produces a single foot position; however, there is more than one set of joint angles that will result in a given foot position (as you can see by planting your foot on the floor and moving your hip and knee). This fact makes the inverse kinematics much more interesting, and difficult to solve, than the forward kinematics. In order to obtain the inverse kinematics equations analytically (using algebra and trigonometry to invert the forward kinematic equations), the joint angles must be constrained, because of the multiple solutions problem described above. If the joint angle possibilities are constrained, there will be only one set of angles available to reach the

given point. For the knee joint, movement is restricted to be between 0° and 90°, where 0° is with the L3 leg segment lined up on the x-axis, and 90° has L3 on the z-axis.

Given these constraints, and the forward kinematic foot position equations, the inverse kinematic equations can be derived. This process can be difficult in itself due to the fact we are dealing with polynomials of trigonometric equations. There are generally three different approaches to solve the inverse kinematics problem. The first approach is to solve the direct kinematics equations algebraically. The second approach is the geometric approach, where one can relate some important point of the structure to the end effector's position. The first two approaches are called the closed-form solution. The last approach is the numerical approach, in which the solution of the inverse kinematics problem is estimated, compared, and recalculated until the error falls below a certain threshold. This approach is generally computationally demanding, but can be applied to any manipulator structure and is guaranteed to have the same accuracy of the solution. In this research, an inverse kinematics algorithm was used that was previously introduced in [24], for a similar 3R-3DOF robot leg mechanism, and those equations are shown here:

$$\theta_1 = \cos^{-1} \left( \frac{X_3}{\sqrt{X_3^2 + Y_3^2}} \right) \quad (5.22)$$

$$\theta_2 = \cos^{-1} \left( \frac{L_2^2 + \left( \left( \sqrt{X_3^2 + Y_3^2} \right) - L_1 \right)^2 + Z_3^2 - (L_3 + d_2)^2}{2 * L_2 * \sqrt{\left( \sqrt{X_3^2 + Y_3^2} - L_1 \right)^2 + Z_3^2}} \right) + \tan^{-1} \left( \frac{Z_3}{\sqrt{X_3^2 + Y_3^2} - L_1} \right) \quad (5.23)$$

$$\theta_3 = \cos^{-1} \left( \frac{\left( \sqrt{X_3^2 + Y_3^2} - L_1 \right)^2 + Z_3^2 - L_2^2 - (d_2 + L_3)^2}{2 * L_2 * L_3} \right) + \tan^{-1} \left( \frac{Z_3}{\sqrt{X_3^2 + Y_3^2} - L_1} \right) \quad (5.24)$$

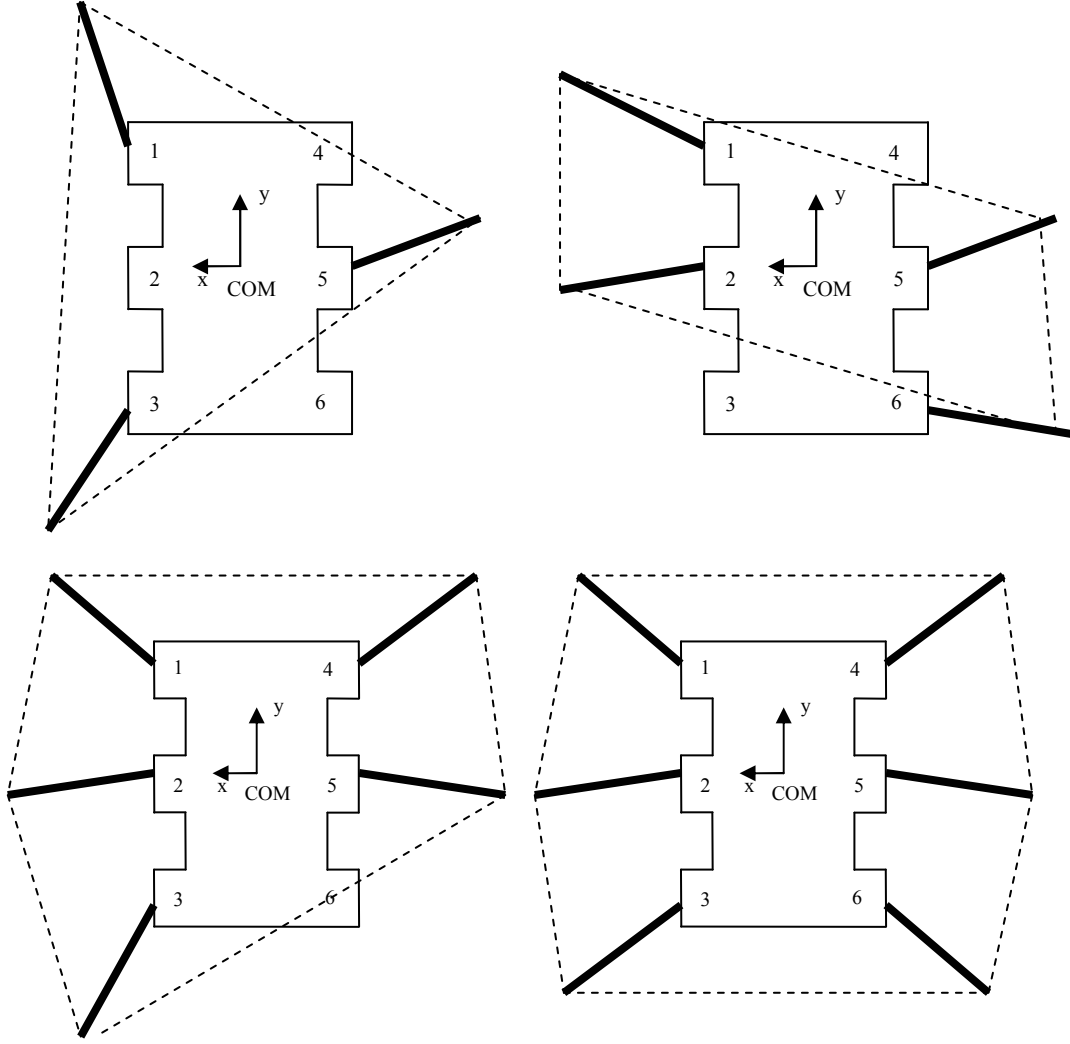
## **CHAPTER 6**

### **STABILIZING BYROBOT'S LEGGED MOBILITY SYSTEM**

#### **6.1 Polygon of Support**

For a hexapod robot, at least 3 legs have to be down at all times for the robot to be stable [25]. Therefore it is possible to have three, four, five, or six legs down at all times, with the latter meaning that the robot is stationary. In order to remain stable at any given moment, the robot's center of mass (COM) should be located within the "Polygon of Support", meaning that the center of mass is as far away from the edges of the polygon that covers the area defined by the feet in contact with the ground. This polygon is basically the projection between all of its support points onto the surface.

But what happens when a statically stable robot lifts a leg and tries to move? Does its COM stay within the polygon of support? If the COM comes too close to one of the edges of this polygon, the robot will tip over in that direction. This was essential for us because we needed our robot to remain stable as each leg lifts and moves for a walking gait. A basic assumption of a statically stable gait is that the weight of a leg is negligible compared to that of the body, so that the total center COM of the robot is not affected by the leg swing. Based on this assumption, the conventional walking gait for a robot is designed so as to maintain the COM of the robot inside of the support polygon, which is outlined by each support leg's tip position. Figure 6.1 shows the configuration of the Polygon of support for the Byrobot with 3, 4, 5, and 6 legs down, respectively.



*Figure 6.1: Byrobot's polygon of support shown for robot with 3, 4, 5, and 6 legs down respectively.*

## 6.2 Mobility on an Incline / Decline Plane

In the case when the robot is on an incline or decline, the projected center of mass (COM) is into the surface plane, just as the force of the robot weight is [26]. The robot's force is projected parallel ( $F_{down}$ ) and perpendicular ( $F_{in}$ ) to the incline or decline plane as shown in Figure 6.2.



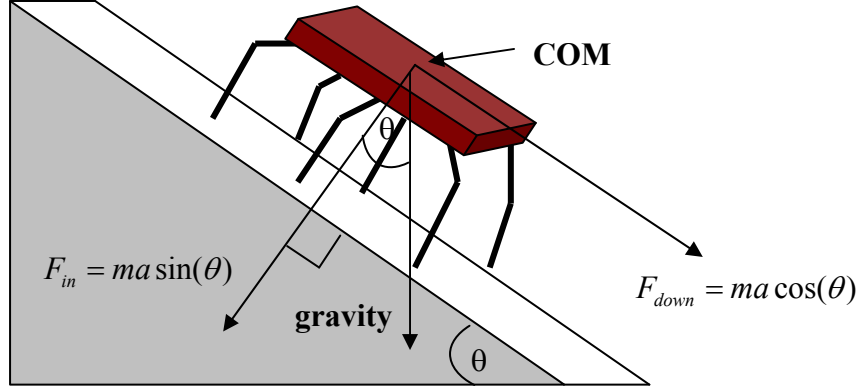


Figure 6.2: Byrobot's projected force and COM on an incline/decline plane.

Where  $m$  is the mass of the robot and  $a$  is the acceleration of the robot due to gravity. The COM is projected into the plane at  $\cos(\theta)$ , where  $\theta$  is the angle of the incline. In this situation, the walking gait needs to be modified so that the COM doesn't cause the robot to tip over. This modification can be done by adjusting the joint angles in the robot legs to correspond to the incline angle.

We can also find the coefficient of friction,  $\mu$ , for the robot standing on its legs. We want the robot to be able to walk up the incline without sliding, so the friction between the robot "feet" and the surface of the incline must be greater than the force of gravity pulling the robot down the plane, in order for the robot to remain stable.

We know that the force of friction opposes the initial motion, and is a force normal to the surface plane. Hence, the force of friction ( $F_f$ ) is:

$$F_f = \mu ma \cos(\theta), \quad (6.1)$$

and the force down the plane is still:

$$F_{down} = ma \sin(\theta), \quad (6.2)$$

so the final total force will be:

$$F_{Total} = F_{down} - F_f = ma \sin(\theta) - \mu ma \cos(\theta) \quad (6.3)$$

Now, at the instantaneous time where there is just enough friction for the robot to stand, and it will begin to slide if any less friction is present, then:

$$F_{Total} = 0$$

$$F_{down} = F_f$$

meaning that,  $ma \sin(\theta) = \mu ma \cos(\theta)$

and solving for  $\mu$  gives

$$\mu = \frac{ma \sin(\theta)}{ma \cos(\theta)} = \frac{\sin(\theta)}{\cos(\theta)} = \tan(\theta) \quad (6.4)$$

Meaning, for Byrobot to be able to stand and walk up the incline, the coefficient of friction between the feet and the plane surface,  $\mu$ , must be greater than  $\tan(\theta)$ . This is also the friction required so that rotation, when Byrobot is on its wheels, is possible to roll up the incline without slippage:  $\mu \geq \tan(\theta)$

## 6.3 Hexapod Walking Gaits

There are three distinct walking gaits used for hexapod robots; the Tripod Gait, the Ripple Gait, and the Wave Gait [27]. Here we will analyze these three common gaits and explain why we chose our selected gait.

### 6.3.1 Preferred Walking Gaits for Hexapod Robots

The **Tripod Gait** is the best-known hexapod gait. The tripod is defined by the front and back legs on one side and the middle leg on the opposite side. For each tripod,

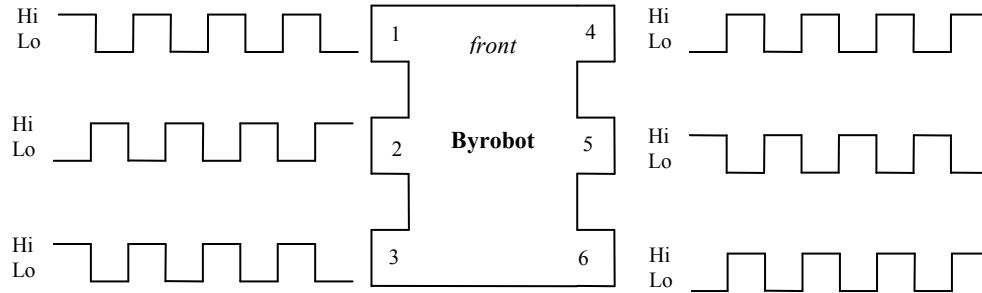
the legs are lifted, lowered, and moved forwards and backwards in unison. During walking, the weight of the hexapod is simply shifted alternately from one tripod to the other. The tripod gait is used for fast speed with a small amount of load, weight of robot components, on the robot. With rising load, it is best to use a different gait where more legs are on the ground at all times to better distribute the robot load for more stability.

In the **Wave Gait**, all legs on one side are moved forward in succession, in the order of the front leg, the rear leg, then the middle leg. This order is then repeated on the opposite side of the robot with the other three legs. This order is used to maintain robot stability. Since only 1 leg is ever lifted at a time, with the other 5 being down, the robot is always in a highly-stable posture.

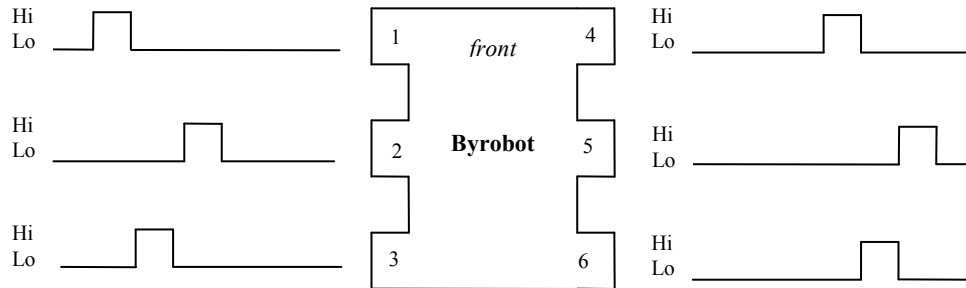
The final stride is the **Ripple Gait**. At least four legs are touching the ground simultaneously. In this gait two legs, one leg on each side of the robot, are moved forward at a time. The name “ripple” was chosen because there are different variations of this gait. The user can vary which two legs are moving based on where the center of mass is located on the robot. Keep in mind that the four legs that are down must form a stable polygon of support controlling COM position for stability. One ripple gait pattern could be the robot moving the front left and right leg on the robot, then the middle left and right leg, and finally the rear left and right legs. However the robot could also move the front right leg and the rear left leg for its first step, or some other combination of left and right legs.

Figure 6.3 shows the square wave plot of each of these gaits with respect to time. In the figure, “*Hi*” means the leg is lifted off the ground at that time and “*Lo*” means the leg is down.

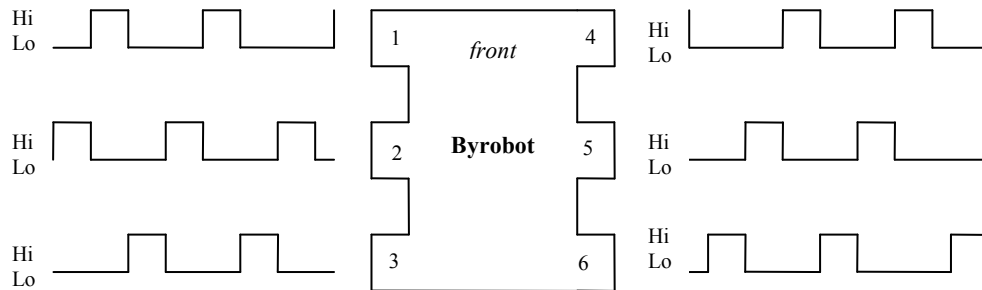
**Tripod Gait – lift 3 legs at a time (3 legs down)**



**Wave Gait – lift 1 leg at a time (5 legs down)**



**Ripple (tetrapod) Gait – lift 2 legs at a time (4 legs down)**



*Figure 6.3: Hexapod gait movements of each leg with respect to time.*

### 6.3.2 Analysis of the Walking Gaits

For each of these robot gaits, we assume the step size is held constant, and the robot legs move in a forward succession. The tripod gait will be fastest, completing a

cycle in 2 leg movements, which is the complete up-forward-down movement of one of the robot legs. The Tripod, although fastest, will also be the least stable, since it always has 3 legs in suspension. The Wave will be most stable, since it keeps the most legs on the ground at all phases of the stride. It will also be the easiest to adjust the wave gait during movement over uneven terrain. This gait, however, is the slowest of the three since it takes 6 leg movements to complete one cycle. The Ripple is second most-stable and second fastest. Only 2 legs are ever off the ground at the same time. The cycle for this gait is completed in 4 leg movements. Only 1 leg per side is ever lifted at a time, and when it is, the leg on the other side directly opposite of it is down.

### **6.3.3 The Chosen Walking Gait**

Due to the large load that is on the Byrobot (approximately 2kg), with all the batteries, servos, wheels, motors, etc.; the initial tripod gait was ineffective. Byrobot simply would collapse under its own weight as soon as the three legs would lift off of the ground, despite the leg position. The wave gait, however, is slow, but remains stable throughout its entire walk, which is most important when the robot is traversing in remote environments. The wave gait therefore became our preferred walking gait. A program was written for Byrobot, which would move the corresponding servo motors in the legs to take small steps at the top servo motor speed. The program loop was set to run eight times. This allowed Byrobot to walk with the wave gait for a total of 8 gaits (one gait equals one step of each leg, for a total of  $6 \times 8 = 48$  steps). Since the servo motors are moving as fast as possible, the only other variable factor was the position of the robot legs during each step. Because of the design of Byrobot, too large of a step will result in

interference of parts and components on the robot, hence, smaller steps are more efficient. The wave gait was successful and was used in all of our testing results, as explained in the next chapter.

## CHAPTER 7

### RESULTS

#### 7.1 Verifying Kinematic Equations

With the derivation of the leg kinematics for the robot, we wanted to verify the placement of the rover feet and the joint angles in the legs. To do this, we chose to first look at the robot during its stable stance. We can physically measure where the foot is positioned approximately based on the origin of the respective leg's coordinate frame, and use the servo motor joint angles from the program as our  $\theta$ 's. We looked at each leg in a stable stance and compare the values of our physical measurements to those from our forward kinematic equations. We then calculated the error in the measurements, and could see that most of the error came from the x-position of the legs. All of the legs have the same y-position being the exact length of leg segment  $L_3$ . This data is shown in Table 7.1.

The inverse kinematics is very important not only during the robot's standing phase, but also for walking. If we want to move each robot foot two inches forward for a particular walking gait, we can calculate the position of the robot foot and use the inverse kinematics to find the servo joint angles. However, when we compared our solutions of our inverse kinematic equations (equations 5.22, 5.23, and 5.24) to the actual angles we used in our program, our error was immensely large. This could be attributed to an error in our inverse kinematics, which is much more complicated to solve than the forward kinematics. We decided that it was best to use more of a "guess and check" method to program our walking gaits. Hence, to move each robot leg forward, we would just slightly adjust the joint angles to get our desired output distance to move the robot foot

position. By using the forward kinematics, we were able to successfully program our walking gait into the robot.

*Table 7.1: Comparison of Foot Position with Measured and Calculated Results*

Leg Number	Position (x, y, z)	Measured (inches)	Calculated (inches)	Error (e) $e = \frac{\text{measured} - \text{calculated}}{\text{calculated}} \times 100\%$
1	X	-3.69	-3.57	3.36%
	Y	4.52	4.44	1.80%
	Z	-7.50	-7.50	0%
2	X	-5.25	-5.50	4.55%
	Y	-0.50	0	Undefined
	Z	-7.50	-7.50	0%
3	X	-3.75	-3.12	20.2%
	Y	-4.55	-4.77	4.61%
	Z	-7.50	-7.50	0%
4	X	3.12	3.16	18.0%
	Y	4.50	4.60	-2.0%
	Z	-7.50	-7.50	0%
5	X	5.25	5.50	4.55%
	Y	-0.50	0	undefined
	Z	-7.50	-7.50	0%
6	X	3.10	3.12	0.64%
	Y	-4.75	-4.69	1.28%
	Z	-7.50	-7.50	0%

## 7.2 Mobility on Flat Surface

We tested the speeds of both the legged and wheeled mobility systems of Byrobot on a flat surface. For the wheeled system, we measured the robot speed at 100%, 80%, and 50% of the motor speeds. To measure the linear velocity of the robot at full speed,



we programmed the robot to travel straight forward for 5 seconds and then measured the distance traveled with a tape measure. In Table 7.2, we see the average distance traveled and linear velocity of the robot at full speed.

*Table 7.2: Average Wheeled Linear Velocity of Byrobot at Full Speed*

Rolls – Full (100%) Speed		
time (seconds)	distance (cm)	velocity (cm/s)
5.00	345.44	69.09
5.00	346.71	69.34
5.00	342.90	68.58
5.00	344.81	68.96
5.00	345.44	69.09
Average		69.01

We noticed that at less than full speed, the robot would have a slight pull to the left, which can be fixed in the future with wheel encoders to ensure that the output motor speeds stay the same for a more precise wheel alignment. Due to the slight pull to the left of the robot when traveling at less than full speed, we cut the time down to 4 seconds so that the pull wouldn't have as much of an effect on our straight-line distance. In Tables 7.3 and 7.4, we see the average linear velocity of the robot at 80% and 50% speed.

*Table 7.3: Average Wheeled Linear Velocity of Byrobot at 80% Speed*

Rolls - 80% Speed		
time (seconds)	distance (cm)	velocity (cm/s)
4.00	190.50	47.63
4.00	200.66	50.17
4.00	192.41	48.10

Table 7.3 Continued

4.00	194.95	48.74
4.00	196.22	49.05
Average		48.74

Table 7.4: Average Wheeled Linear Velocity of Byrobot at 50% Speed

Rolls - 50% Speed		
time (seconds)	distance (cm)	velocity (cm/s)
4.00	172.72	43.18
4.00	165.10	41.28
4.00	168.91	42.23
4.00	170.18	42.55
4.00	167.64	41.91
Average		42.23

Using these robot velocities, we calculated the average angular (rotational speed) of the robot wheels in revolutions per second (rev/s), using equation 4.6,  $d = Vt = \omega r t$ .

Where  $V = \omega r$  or  $\omega = \frac{V}{r}$ . The radius of the robot wheels,  $r$ , is 3.175 cm. The average angular velocities are shown in table 7.5:

Table 7.5: Average Angular Velocities of Robot

% of robot full speed	Average Linear Velocity of Robot Wheels (cm/s)	Angular (Rotational) Velocity of Robot Wheels (rev/s)
100%	69.01	21.74
80%	48.74	15.35
50%	42.23	13.30

We can see how reducing the robot's motor speed in the program is not directly proportional to the robot's reduction in its linear and angular velocity. At 80% and 50% speed, the robot's linear and angular velocities should be 80% and 50% of its velocity at 100% speed. However, our test show that this is not the case.

As mentioned previously in chapter 6, we initially tried to walk Byrobot using the tripod gait, where three legs would move at time, however having only three legs down at a time could not support the large robot load, and the robot would collapse under its own weight before it could walk one step. Therefore, the wave gait was chosen; even though it was the slowest gait, it was the most stable. For the walking wave gait, a program was written for Byrobot, which would move the corresponding servo motors in the legs to take small steps at the top servo motor speed. Each leg made the sequence of lifting up 3 inches (in z-direction), moving forward 2 inches (in y-direction), and the leg comes back down. With this information, the correct servo joint angles were found using the inverse kinematics, for each step in the gait. This sequence is done for each leg on the robot in the order: leg 1, leg 3, leg 2, then leg 4, leg 6, and leg 5. This order was chosen to move one side of the legs before the other side, so that there was always a stable polygon. At the end of the first gait when all three legs had moved a few inches forward, a "forward scoot" is done to move the robot body forward as well. At this point, all legs are then returned to their original standing position.

We tested the robot walking its wave gait for ten test trials, as shown in Table 7.6 and the distance of travel for the robot was measured.

*Table 7.6: Walking (wave gait) Speed of Byrobot on Flat Surface*

Trail	Walking on Flat Surface Full Speed (8 sequences, 48 steps)		
	time (sec)	distance (cm)	velocity (cm/sec)
1	66.00	83.82	1.27
2	66.00	81.28	1.23
3	66.00	80.98	1.23
4	66.00	78.10	1.18
5	65.00	80.10	1.23
6	66.00	75.56	1.14
7	66.00	76.88	1.16
8	65.00	73.66	1.13
9	65.00	71.12	1.09
10	65.00	60.96	0.94
Averages	65.60	76.25	1.16

The average distance walked by Byrobot with the wave gait is 76.25cm in 65.6 seconds, for an average walking speed of 1.16 cm/s. Below in Figure 7.1, we see photographs of the successful Wave Gait sequence of Byrobot.



*Figure 7.1: Photographs of the Wave Gait sequence of Byrobot*

### **7.3 Mobility in Mars Sand Pit**

Snow shoes were placed on Byrobot for better mobility in the Mars sand pit, as discussed in Chapter 2 and shown in figures 2.7 and 2.8. If the robot was traversing on its wheels in the sand pit, it would easily get stuck when it came to an uneven area such as a clump

of sand, therefore the legs were the desired system to use. Below in Table 7.7, we measured the speed of the robot walking in the sand pit, using our same wave gait program:

*Table 7.7: Walking (wave gait) Speed of Byrobot in Mars Sand Pit*

Trail	Walking in Mars Sand Pit Full Speed (8 sequences, 48 steps)		
	time (sec)	distance (cm)	velocity (cm/sec)
1	66.00	34.00	0.52
2	68.00	40.00	0.59
3	66.00	35.00	0.53
4	67.00	35.00	0.52
5	69.00	36.00	0.52
6	68.00	41.00	0.60
7	67.00	37.00	0.55
8	68.00	38.00	0.56
9	67.00	33.00	0.49
10	67.00	35.00	0.52
Averages	67.30	36.40	0.54

We see here that the average walking speed of Byrobot in the sandy terrain is 0.54 cm/s; this is approximately half the speed of the robot when walking on the flat surface. From watching the robot walk in both of these terrains, we determined that the slipping of the feet on the robot is the reason for this difference. On the sandy terrain, the robot's mars shoes will slip with each step, so as the robot is moving forward, it is also slipping and sliding backward. When on the flat floor surface, the rubber feet on the robot supply enough grip so that the slipping is significantly reduced and the robot can have full forward progress.

## 7.4 Traversing Up and Down a Slope

We wanted to see if Byrobot could traverse up and down a sloped terrain smoothly, on both its wheeled and legged mobility systems. Unfortunately, the robot was unsuccessful at this using its legs. On the adjustable ramp that we had in the lab, the legs would lose balance and tumble over when trying to traverse up or down the ramp. The reason for this was because the same flat-surface walking gait was used on the ramp. We can solve this stability problem in the future by simply changing the configuration of the legs to control the center of mass, keeping it within the new polygon of support during the gait on the incline.

We were able to test the wheeled mobility system at a couple different angles. We first measured the speed of the robot on a  $22^\circ$  ramp with a distance of 132 cm, as the robot moved up and down the ramp, and the time and speed are recorded below in tables 7.8 and 7.9:

*Table 7.8: Wheeled Speeds of Byrobot Traversing Up a  $22^\circ$  Incline Slope*

Rolls Up $22^\circ$ Incline		
time (seconds)	distance (cm)	velocity (cm/s)
3.66	132.00	36.07
3.50	132.00	37.71
3.44	132.00	38.37
3.47	132.00	38.04
3.51	132.00	37.61
Average		37.56

*Table 7.9: Wheeled Speeds of Byrobot Traversing Down a 22° Decline Slope*

Rolls Down 22° Incline		
time (seconds)	distance (cm)	velocity (cm/s)
1.31	132.00	100.76
1.44	132.00	91.67
1.40	132.00	94.29
1.36	132.00	97.06
1.41	132.00	93.62
Average		95.48

Next we put the robot on slope with an angle of 35°. Table 7.10 shows the speed of the robot rolling up the incline. What we noticed was the robot struggled to make it up this steep slope. After a certain distance, the robot wheels started to lose power and slip, and roll backwards down the ramp. The distance shown is the farthest distance the robot made it up the ramp, right before it began to slip and roll backward.

*Table 7.10: Wheeled Speeds of Byrobot Traversing Up a 35° Incline Slope*

Rolls Up 35° Incline *		
time (seconds)	distance (cm)	velocity (cm/s)
5.35	72.00	13.46
5.07	69.00	13.61
4.78	71.00	14.85
5.07	71.00	14.00
5.11	70.00	13.70
Average		13.92

Table 7.11 below shows the speed of the robot traversing down this incline. No slippage of the wheels was noticed as the robot rolled rapidly down this slope, assuming the wheels kept their traction throughout the movement.

*Table 7.11: Wheeled Speeds of Byrobot Traversing Down a 35° Decline Slope*

Rolls Down 35° Incline		
time (seconds)	distance (cm)	velocity (cm/s)
1.19	122.00	102.52
1.18	122.00	103.39
1.18	122.00	103.39
1.16	122.00	105.17
1.20	122.00	101.67
Average		103.23

We can see that the robot can move much faster on its wheels when going down an incline. We tried to increase the angle of the incline slightly to test the robot speed at a steeper slope, but the robot didn't have enough power to make it up at a measurable distance, as it immediately started slipping.

## 7.5 Conclusions

In this thesis, we have shown the kinematic model of the six-legged and four-wheeled reconfigurable Byrobot rover. The proposed kinematic models can be extended for analyzing the mobility performances of the four-wheeled six-legged robot in other operating conditions, as turning left and right on both mobility systems, avoiding obstacles with future sensors, and ascending and descending slopes and stairs. Our results show that our derived kinematic equations for the wheels and legs can be



implemented in the hardware of the actual robot and yield accurate measurements. We were able to use forward and reverse kinematics to give us exact values to find the stable polygon of support for the six-legged mobility system. We have also explained the common walking gaits used on hexapod robots and why we chose the wave gait for this robot. Testing of both the tripod and the wave gait on our actual hardware proved to us that the wave gait was the correct choice, for it remained the most stable. Our speed test showed that the walking gait, at an average speed of 1.16 cm/s, is much slower than when the robot rolls on its wheels at an average full speed of 69.01 cm/s, when operating on a flat surface. This can be expected since wheeled mobility systems generally have more speed than legged mobility systems. When the robot is in uneven terrain that causes its wheels to lose traction, such as in Mars the sand pit, we were able to see from the hardware that the legs were desirable to use to be able to traverse the robot through the sandy terrain. The average walking speed in the sandy terrain was 0.54 cm/s, less than half the walking speed on a flat surface. We contribute this speed reduction to the feet on the robot slipping backward in the sand pit slightly as it the robot body moves forward. We also saw how inefficient the legs operate currently on a slope. Without being able to change the center of mass location on the robot body, the slightest change in the slope of the terrain caused the robot to tip over on its legs. However the robot on its wheeled system was able to successfully traverse up a 35° incline at an approximate distance of 70 cm, before it began to slip backward. Any larger angle caused the robot to slip immediately.

Having a robot that can both walk and roll would be beneficial to planetary exploration since it can transition from the use of its legs or wheels based on the terrain

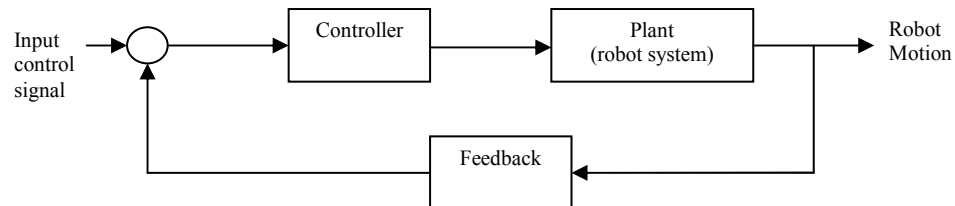
conditions [28]. When the robot is on rough terrain, the ability for it to reconfigure its mobility system from legs to wheels and vice versa will help it to successfully navigate [29]. For Byrobot to navigate on an incline, future research will allow the center of mass location to be known at all times, so the robot can adjust the joint angles in its legs accordingly to remain stable. We will also implement sensors and wheel encoders for a feedback control systems, which we will discuss in Chapter 8. This will facilitate additional future autonomous capabilities of the robot.

## CHAPTER 8

### FUTURE WORK

The Byrobot project has developed a basic platform that will be able to maneuver on multiple terrains. However, there is much further research that can be done to improve on the robotic system. The common NASA Rocker-Bogie[30] suspension system that is used on the Mars rovers could also be placed on Byrobot. This system would allow Byrobot to be able to roll over rough and uneven surfaces more smoothly. Current Rocker-Bogie systems allow traversability over obstacles up to twice its wheel diameter [31]. However, by replacing the current wheeled mobility system with this Rocker-Bogie system, there will be less of a clearance between the ground and the wheel when the robot is standing on its legs. In this situation, the Byrobot will not be able to walk over larger obstacles.

There are a number of sensors that should be used for the robot that would improve the precision and autonomous transition between its two mobility systems. Using sensor feedback, so that the robot will “know” what it is actually doing at all times, will transform the robot into a closed-loop feedback control system, shown in Figure 8.1. Closed loop control is needed for autonomous capabilities in mobile robots [32].



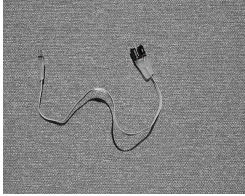
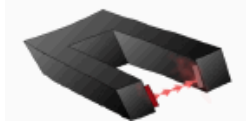

*Figure 8.1: Future robot Closed Loop Feedback Control*

For this closed loop control system, the input is simply the voltage sent into the controller. The controller converts this voltage to signal that will drive the motors on the robot, making the robot move forward. The feedback will be any sensors, encoders, or vision system that will let the robot know of its surroundings including the constant angular velocity of the wheels. Byrobot is currently being executed without any feedback making it an open loop control system [32].

Currently the Byrobot is unable to self measure its angular speed on its wheeled platform. This would be done by the constant measurement of the output rotations of the motor shafts, on each of the four DC motors that are directly attached to the wheels of the robot. The best method for this would be to use encoders for each of the wheels. One of the simple methods for a wheel encoder involves using a sensor in the shape of a “U” that sends an infrared beam across it. This U-sensor can be placed on the same axle that a robot wheel would be on, and have a disk that rotates through it. This disk can contain holes or alternating black and white colors, positioned in a circular pattern about the disk center. The robot controller can count each time the infrared beam on the U-sensor detects a hole, or a new color, to count one full rotation of the axle, which is equivalent to one full rotation of the wheel. A picture of this encoder sensor is shown in figure 8.2 [33].

This encoder, which should be used for each of the robot wheels, would keep a constant count on how many rotations the wheel has made and with the robot controller’s clock measuring elapsed time, can give you the rotational speed. This rotational speed will be useful if each of the motors isn’t outputting the same shaft speed, which would

cause each of the wheels to rotate at different speeds and throw off the wheel alignment on the robot.

		
<p>Model of the shaft encoder.</p>	<p>U-shaped so that it constantly transmits and receives an infrared beam when turned on.</p>	<p>Works best with a 6-hole LEGO wheel attached to the robot axle.</p>

*Figure 8.2: Possible Future Robot Encoder Add-On [33]*

Contact sensors should also be placed on the feet of the robot. This would let the robot know when its “feet” have touched the ground when walking on its legged mobility systems. Currently, Byrobot is depending on the exact position of its servo motors, which control the leg joints, to keep the legs stable and off the ground. However, if the leg position was slightly off, it may not touch the ground at the position which we specify for the servo motors, hence, making the robot unstable. A simple digital switch could be used for this ground detection. With contact sensors used in the walking gait, the next leg won’t move into position until the previous leg has touched the ground.

For autonomous transition between the mobility systems, Byrobot will need some sort of vision system, such as camera. It will need to be able to detect when there is a change in terrain so that it can switch from one mobility system to the other. For example, Byrobot could be rolling fast on the Mars surface on its wheels, until it comes to a sandy area where the wheels lose traction. In this situation, the vision on the robot

would see the hazardous sandy area before the wheels encounter it. When the robot sees this, it could stop rolling, have the legs extract out, stand up on its legs, and walk its way through the sand to freedom! And at the same time, when the robot sees that it is no longer in the sand and back on a normal surface, it could lower back down on its wheels and continue rolling.

Another useful device for this autonomous transaction is a feedback control loop or feedback sensors. These would be used so that the robot would be able to know the status of its electronic instruments during the operation. For example, when the wheels get stuck in the sandy terrain as mentioned above, they will stop rotating at their normal speed, even though the motors are still outputting the same speed. But the robot doesn't know that the wheels are stuck. If the roll/pitch of the wheels could be constantly measured, the robot will know when it isn't getting the output it should. With feedback sensors, which the encoders could count as these, the robot will know that the wheels have stopped rotating at the output speed of the motor shafts, and are now rotating at a slower relative speed, if not stopped completely.

Also, if the legs were to get stuck while walking, the robot needs to know that the leg position is not what it should be due to this. This would require getting feedback from the servo motors that control the leg joints, hence, getting constant feedback about the exact position of the servos. If the servo motors were supposed to reach position X in Y time, and after Y time elapses, the servos are stuck in position W (not yet attained position X), and after Z time (time after Y), they are still in this W position, that means there is a problem and that the legs are probably stuck. By having this feedback control, the robot will be able to handle problems encountered during its operation, and react

accordingly. This feedback will enable the robot to think, sense, and act, thus making it fully autonomous.

## APPENDIX A

### BYROBOT MAIN SELF-RECONFIGURATION CODE

This program was written in C and was downloaded into the Eyebot, the primary controller on Byrobot. The Eyebot is connected with the SSC-32 Servo Controller as mentioned in this thesis, thus, allowing one program to fully operate the robot. This program is the main demo program used to show Byrobot's self-reconfiguration. The program was designed to make Byrobot roll on its wheeled mobility system backward for a few seconds, then forward for a few seconds, then stand up on its legs, walk forward (taking 48 total steps), lower back down on its wheels, and roll off again before stopping.

The program sends character strings out through the null modem connecting the serial ports of the Eyebot and the SSC-32 to move the servo motors connected on the SSC-32.

```
//this program makes the Byrobot roll backward and forward, stand, walk, sit, and roll again!
```

```
#include "eyebot.h"
```

```
char newline = '\n';  
char term = 4;
```

```
char str1[] =  
"#31P1600#30P1600#29P2500#27P1650#26P1600#25P2500#23P1650#22P1600#21P2500#15P1500#14P1500#13P500#11P1500#10P1500#9P500#7P1500#6P1500#5P500T3000\r";  
//legs retract slowly (process takes 3 seconds)
```

```
char str2[] =  
"#31P1600#30P1600#29P500#27P1650#26P1600#25P500#23P1650#22P1600#21P500#15P1500#14P1500#13P2500#11P1500#10P1500#9P2500#7P1500#6P1500#5P2500T1000\r";  
//knees extract (stand up 2)
```

```
char str3[] =  
"#31P1675#30P1600#29P500#27P1675#26P1600#25P500#23P1675#22P1600#21P500#15P1400#14P1500#13P2500#11P1400#10P1500#9P2500#7P1400#6P1500#5P2500T1000\r";  
//legs lower a little bit (stand up 3)
```

```
char str4[] =  
"#31P1675#30P1100#29P500#27P1675#26P1800#25P500#23P1675#22P2250#21P500#15P1400#14P1900#13P2500#11P1400#10P1200#9P2500#7P1400#6P750#5P2500T1000\r";  
//legs move into standing position (stand up 4)
```

```
char str5[] =  
"#31P1900#30P1100#29P500#27P1900#26P1800#25P500#23P1900#22P2250#21P500#15P1150#14P1900#13P2500#11P1150#10P1200#9P2500#7P1150#6P750#5P2500T1000\r";  
//robot stands robot up, takes 1second
```

```
char str31[] = "#31P1725T500\r"; /*leg1 lifts*/  
char str15[] = "#15P1325T500\r"; /*leg4 lifts*/  
char str30[] = "#30P1350T500\r"; /*leg1 backward*/  
char str14[] = "#14P1650T500\r"; /*leg4 backward*/
```



```

char str27[] = "#27P1725T500\r"; /*leg2 lifts*/      char str11[] = "#11P1325T500\r"; /*leg5 lifts*/
char str26[] = "#26P2000T500\r"; /*leg2 backward*/  char str10[] = "#10P1000T500\r"; /*leg5 backward*/

char str23[] = "#23P1725T500\r"; /*leg3 lifts*/      char str7[] = "#7P1325T500\r"; /*leg6 lifts*/
char str22[] = "#22P2500T500\r"; /*leg3 backward*/  char str6[] = "#6P500T500\r"; /*leg6 backward*/

//these functions will lower the robot legs back to their original stand up position
char str312[] = "#31P1850T500\r"; /*leg1 lowers*/   char str152[] = "#15P1150T500\r"; /*leg4 lowers*/

char str272[] = "#27P1850T500\r"; /*leg2 lowers*/   char str112[] = "#11P1150T500\r"; /*leg5 lowers*/

char str232[] = "#23P1850T500\r"; /*leg3 lowers*/   char str72[] = "#7P1150T500\r"; /*leg6 lowers*/

MotorHandle  rightmotor;
MotorHandle  leftmotor;

void sendstr(char s[])
{
  int i;
  i = 0;
  while (s[i] != 0)
  {
    OSSendRS232(&s[i], SERIAL1);
    LCDPrintf("%c",s[i]);
    i++;
  }
}

int main ()
{
  int x;
  LCDPrintf("ByroBot is Alive!\n");
  LCDMenu(" ", " ", " ", "END");
  OSInitRS232(SER115200, NONE, SERIAL1);

  leftmotor = MOTORInit(-101);
  rightmotor = MOTORInit(-100);

  OSWait(500); //wait 5 seconds before program starts

  sendstr(str1);
  OSWait(100); //robot legs in retracted position

  OSWait(200);

  MOTORDrive (rightmotor,-100);
  MOTORDrive (leftmotor,-100);
  OSWait(200);
  MOTORDrive (rightmotor,0);
  MOTORDrive (leftmotor,0);
  OSWait(200);

  MOTORDrive (rightmotor,-100);
  MOTORDrive (leftmotor,100);
  OSWait(150);
  MOTORDrive (rightmotor,0);
  MOTORDrive (leftmotor,0);

```

```

        OSWait(150);

        MOTORDrive (rightmotor,100);
        MOTORDrive (leftmotor,-100);
        OSWait(150);
        MOTORDrive (rightmotor,0);
        MOTORDrive (leftmotor,0);
        OSWait(150);

        MOTORDrive (rightmotor,100);
        MOTORDrive (leftmotor,100);
        OSWait(200);
        MOTORDrive (rightmotor,0);
        MOTORDrive (leftmotor,0);
        OSWait(200);

    sendstr(str1);
    OSWait(100); //robot legs in retracted position

    sendstr(str2);
    OSWait(100); //knee extend

    sendstr(str3);
    OSWait(100); //legs lower slightly

    sendstr(str4);
    OSWait(100); //legs move into position before standing

    sendstr(str5);
    OSWait(300); //robot stands up, gait will start in 3 seconds


//the following FOR loop will run this backward walk wavegait, with robot taking 8 steps total (one step
per leg)

for (x=0; x<8; x++)
{

//leg 3 lift, backward, and down (0.4 second wait between movements)
sendstr(str23);
OSWait(40);
sendstr(str22);
OSWait(40);
sendstr(str232);
OSWait(40);

//leg 1 lift, backward, and down
sendstr(str31);
OSWait(40);
sendstr(str30);
OSWait(40);
sendstr(str312);
OSWait(40);

//leg 2 lift, backward, and down

```

```

sendstr(str27);
OSWait(40);
sendstr(str26);
OSWait(40);
sendstr(str272);
OSWait(40);

//leg 6 lift, backward, and down
sendstr(str7);
OSWait(40);
sendstr(str6);
OSWait(40);
sendstr(str72);
OSWait(40);

//leg 4 lift, backward, and down
sendstr(str15);
OSWait(40);
sendstr(str14);
OSWait(40);
sendstr(str152);
OSWait(40);

//leg 5 lift, backward, and down
sendstr(str11);
OSWait(40);
sendstr(str10);
OSWait(40);
sendstr(str112);
OSWait(40);

//return to original standing position (forward scoot)
sendstr(str5);
OSWait(100);

} //end for loop

OSWait(100);
sendstr(str1);
OSWait(300); //robot legs retract, wait 3 seconds

MOTORDrive (rightmotor,100);
MOTORDrive (leftmotor,100);
OSWait(100);
//robot drives backward for 1 second

MOTORRelease (rightmotor|leftmotor);
//release motors and end program

return 0;

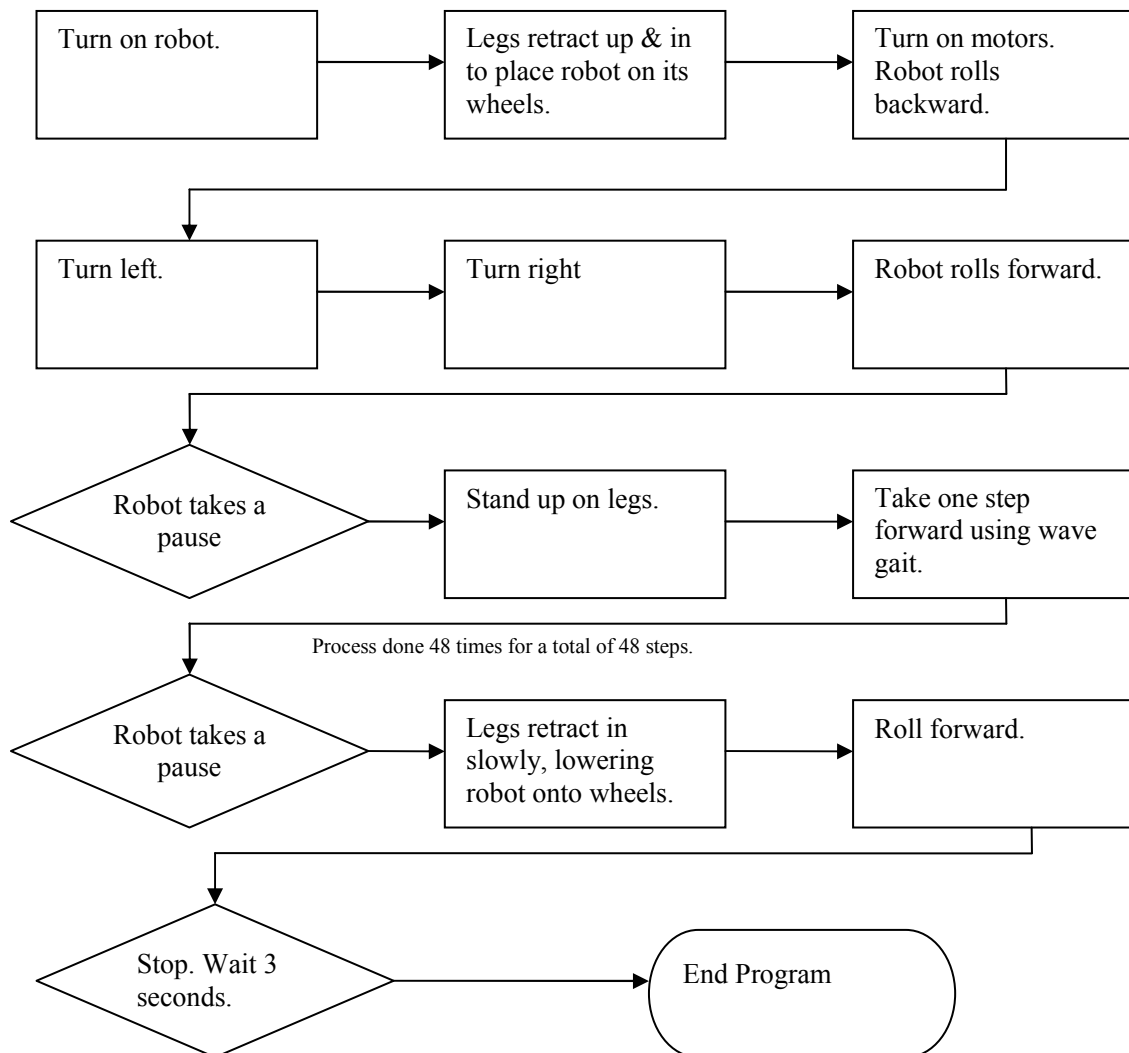
} //end main program

```

## APPENDIX B

### SELF-RECONFIGURATION PROGRAM FLOW CHART

This is the flow chart for the main demo used to show Byrobot's self-reconfiguration. The program was designed to make Byrobot roll on its wheeled mobility system backward for a few seconds, then forward for a few seconds, then stand up on its legs, walk forward (taking 48 total steps, using our wave gait), lower back down on its wheels, and roll off again before stopping. There are no sensors or other feedback devices in this open-loop controlled program.



# APPENDIX C

## PRICES OF BYROBOT PARTS

Item Description	Webpage for purchase	Price per Unit	# of units	Total Price
Servo Controller SSC-32	<a href="http://www.bromotion.com/Product.aspx?productID=355&amp;categoryID=52">http://www.bromotion.com/Product.aspx?productID=355&amp;categoryID=52</a>	39.95	1	39.95
Lite Filite' F foam Tires (pair)	<a href="http://www.robotmarketplace.com/marketplace_ant_wheels.html">http://www.robotmarketplace.com/marketplace_ant_wheels.html</a>	4.79	2	9.58
Wheel Hub/Prop Adapter, for 1/8" & 3mm shaft	<a href="http://www.robotmarketplace.com/marketplace_ant_wheels.html">http://www.robotmarketplace.com/marketplace_ant_wheels.html</a>	4.95	4	19.80
Motor Bracket Mounts	<a href="http://www.robotmarketplace.com/marketplace_ant_wheels.html">http://www.robotmarketplace.com/marketplace_ant_wheels.html</a>	8.99	4	35.96
COPAL 60:1 Gearmotor	<a href="http://www.robotmarketplace.com/marketplace_ant_motors.html">http://www.robotmarketplace.com/marketplace_ant_motors.html</a>	21.99	4	87.96
Hitec Standard Servo - Ball Bearing	<a href="http://www.robotmarketplace.com/marketplace_rc-servos.html">http://www.robotmarketplace.com/marketplace_rc-servos.html</a>	7.99	16	127.84
Hitec HS-5645MG High-Torque 2BB MG Servo	<a href="http://www.bromotion.com/Product.aspx?productID=509&amp;categoryID=93">http://www.bromotion.com/Product.aspx?productID=509&amp;categoryID=93</a>	comes as a pack	6	239.94
Dubro Full Threaded Rod 4-40 12"	<a href="http://www.2lowertohobbies.com/stock.htm#t00001p72&amp;LX0872&amp;P=M">http://www.2lowertohobbies.com/stock.htm#t00001p72&amp;LX0872&amp;P=M</a>	\$14.39	1	\$14.39
Dubro Threaded Coupler 4-40 (2)	<a href="http://www.2lowertohobbies.com/stock.htm#t00001p72&amp;LX0924&amp;P=M">http://www.2lowertohobbies.com/stock.htm#t00001p72&amp;LX0924&amp;P=M</a>	\$1.49	3	\$4.47
Dubro Heavy Duty Ball Links 4-40 (12)	<a href="http://www.2lowertohobbies.com/stock.htm#t00001p72&amp;LXFPV6&amp;P=V">http://www.2lowertohobbies.com/stock.htm#t00001p72&amp;LXFPV6&amp;P=V</a>	\$19.99	1	\$19.99
	Totals for Robot Parts to Purchase			599.88
<b>Materials</b>				
<b>Aluminum</b>				
Aluminum 6061-T6 90-degree angle, 2"x 2"x .125" (thickness), 18" length (to be cut down)	<a href="http://www.metalsupermarkets.com/">http://www.metalsupermarkets.com/</a>	10.00	1	10.00
AR6061/12 AL ROUND 6061 .500 (.5" dia. metal rod, 7" long)	donated by Georgia Tech Machine shop	1.75	0	0.00
<b>Polycarbonate Plastic</b>				
12" x 18" x .25" (thickness), Polycarbonate Plastic sheet for robot base, legs, and wheel brackets	donated by Georgia Tech Machine shop	12.00	0	0.00
<b>Miscellaneous Items</b>				
Box of 100 6-32, 1/2" long screws, washers, nuts. Other miscellaneous items such as screws, rubber feet, etc.	Ace Hardware Store, Home Depot	15.00	1	15.00
Battery holders, rechargeable batteries, wire-pin crimp connector suppliers	Radio Shack	20.00	1	20.00
	Totals for Materials			45.00
	<b>TOTAL COST FOR ROBOT (not including Eyebot)</b>			<b>644.88</b>

## APPENDIX D

### WEIGHT OF BYROBOT PARTS

Aluminum Density:	1.56 oz/in <sup>3</sup>				
Polycarbonate Plastic Density	0.6352min - 0.7744max oz/in <sup>3</sup>				
<b>Part to be constructed</b>	<b>Volume of One Part (in<sup>3</sup>) x Quantity</b>	<b>Total Volume of Parts (in<sup>3</sup>)</b>	<b>Weight of Parts using Aluminum (oz)</b>	<b>Weight of Parts using Plastic, min (oz)</b>	<b>Weight of Parts using Plastic, max (oz)</b>
Base	17.7573	17.7573	27.7	11.28	13.75
Wheel Brackets	1.01 x 4	4.04	6.3	2.566	3.13
Wheel Bracket Pole Attachment	.638 x 2	1.276	1.99	0.8105	0.99
Servo L-Brackets	.87875 x 6	5.2725	8.225	3.349	4.08
Bottom Leg Part	1.3125 x 6	7.875	12.285	5.002	6.1
Top Leg Part	1.406 x 6	8.4375	13.1625	5.3595	6.53
<b>TOTALS</b>		<b>44.6583</b>	<b>69.6625</b>	<b>28.367</b>	<b>34.58</b>
<b>Weight of Parts Using Aluminum</b>	<b>69.67 oz</b>				
<b>Weight of Parts Using Polycarbonate (Lexan) Plastic</b>	<b>28.367 oz min, 34.58oz max</b>				
<b>Parts to purchase</b>	<b>Weight of One Part (oz) x Quantity</b>	<b>Quantity</b>	<b>Weight of Part (oz)</b>		
Eyebot Controller	6.561	1	6.561		
Servos	1.55	18	27.9		
Motor Bracket Mounts	0.1	4	0.4		
Motors	0.88	4	3.52		
Wheels (2.5" dia. foam tires)	0.48	4	1.92		
Wheel Adapter Hubs	0.1764	4	0.7056		
Miscellaneous Dubro Ball Links and Rods, Nuts, Screws, etc.	5	1	5		
<b>TOTAL WEIGHT OF PARTS TO PURCHASE (oz)</b>			<b>46.0066</b>		
<b>TOTAL WEIGHT (ALL ALUMINUM)</b>	<b>115.6691</b>				
<b>TOTAL WEIGHT (ALL PLASTIC)</b>	<b>71.37oz min</b>	<b>77.59 oz max</b>			

## APPENDIX E

### AMERICAN WIRE GAUGE CHART

The following chart is a guideline of maximum current carrying capacity for copper wire, following the *Handbook of Electronic Tables and Formulas* for American Wire Gauge [34].

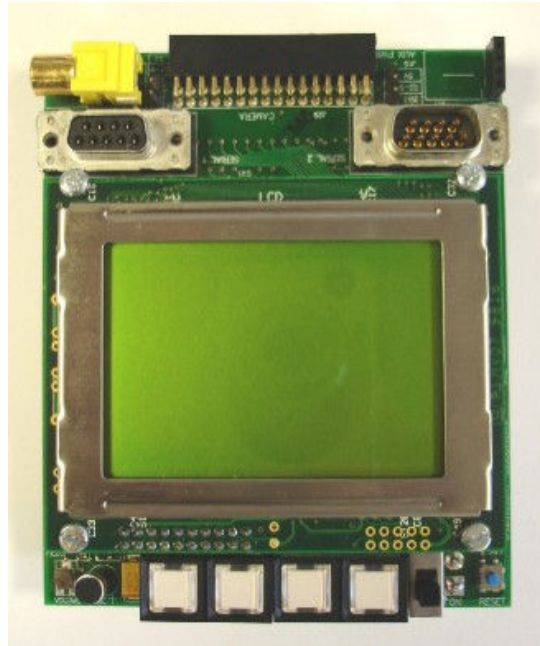
AWG gauge	Diameter (inches)	Diameter (mm)	Ohms per 1000 ft	Ohms per km	Maximum amps for chassis wiring	Maximum amps for power transmission
OOOO	0.46	11.684	0.049	0.16072	380	302
OOO	0.4096	10.40384	0.0618	0.202704	328	239
OO	0.3648	9.26592	0.0779	0.255512	283	190
0	0.3249	8.25246	0.0983	0.322424	245	150
1	0.2893	7.34822	0.1239	0.406392	211	119
2	0.2576	6.54304	0.1563	0.512664	181	94
3	0.2294	5.82676	0.197	0.64616	158	75
4	0.2043	5.18922	0.2485	0.81508	135	60
5	0.1819	4.62026	0.3133	1.027624	118	47
6	0.162	4.1148	0.3951	1.295928	101	37
7	0.1443	3.66522	0.4982	1.634096	89	30
8	0.1285	3.2639	0.6282	2.060496	73	24
9	0.1144	2.90576	0.7921	2.598088	64	19
10	0.1019	2.58826	0.9989	3.276392	55	15
11	0.0907	2.30378	1.26	4.1328	47	12
12	0.0808	2.05232	1.588	5.20864	41	9.3
13	0.072	1.8288	2.003	6.56984	35	7.4
14	0.0641	1.62814	2.525	8.282	32	5.9
15	0.0571	1.45034	3.184	10.44352	28	4.7
16	0.0508	1.29032	4.016	13.17248	22	3.7
17	0.0453	1.15062	5.064	16.60992	19	2.9
18	0.0403	1.02362	6.385	20.9428	16	2.3
19	0.0359	0.91186	8.051	26.40728	14	1.8
20	0.032	0.8128	10.15	33.292	11	1.5
21	0.0285	0.7239	12.8	41.984	9	1.2
22	0.0254	0.64516	16.14	52.9392	7	0.92
23	0.0226	0.57404	20.36	66.7808	4.7	0.729
24	0.0201	0.51054	25.67	84.1976	3.5	0.577
25	0.0179	0.45466	32.37	106.1736	2.7	0.457
26	0.0159	0.40386	40.81	133.8568	2.2	0.361
27	0.0142	0.36068	51.47	168.8216	1.7	0.288
28	0.0126	0.32004	64.9	212.872	1.4	0.226
29	0.0113	0.28702	81.83	268.4024	1.2	0.182

30	0.01	0.254	103.2	338.496	0.86	0.142
31	0.0089	0.22606	130.1	426.728	0.7	0.113
32	0.008	0.2032	164.1	538.248	0.53	0.091
Metric 2.0	0.00787	0.200	169.39	555.61	0.51	0.088
33	0.0071	0.18034	206.9	678.632	0.43	0.072
Metric 1.8	0.00709	0.180	207.5	680.55	0.43	0.072
34	0.0063	0.16002	260.9	855.752	0.33	0.056
Metric 1.6	0.0063	0.16002	260.9	855.752	0.33	0.056
35	0.0056	0.14224	329	1079.12	0.27	0.044
Metric 1.4	.00551	.140	339	1114	0.26	0.043
36	0.005	0.127	414.8	1360	0.21	0.035
Metric 1.25	.00492	0.125	428.2	1404	0.20	0.034
37	0.0045	0.1143	523.1	1715	0.17	0.0289
Metric 1.12	.00441	0.112	533.8	1750	0.163	0.0277
38	0.004	0.1016	659.6	2163	0.13	0.0228
Metric 1	.00394	0.1000	670.2	2198	0.126	0.0225
39	0.0035	0.0889	831.8	2728	0.11	0.0175
40	0.0031	0.07874	1049	3440	0.09	0.0137



## APPENDIX F

### THE EYEBOT CONTROLLER



- 25MHz 32bit Controller (Motorola 68332)
- 1MB RAM, to download programs to run immediately and/or store in ROM
- 512KB ROM (for system + user programs!) extendible to 2MB, which allows
- permanent storage of up to 3 programs. - 1 parallel port
- 8 digital inputs and 8 digital outputs
- 8 analog inputs
- 2 motor drivers
- interface for color camera
- large graphics LCD to display messages, including displaying elapsed system
- time (128x64 pixels)
- 4 input buttons
- reset button, power switch
- battery level indicator
- Programming in C or assembly language
- Free simulation system available.
- Operating RoBIOS is freely available
- Large number of sample programs

Dimensions: width x height x depth: 8.8 cm x 10.5 cm x 2.0 cm (+0.8 cm for connectors)

Weight: 186 g

## APPENDIX G

### THE SSC-32 SERVO CONTROLLER



PC board size = 3.0" x 2.3"

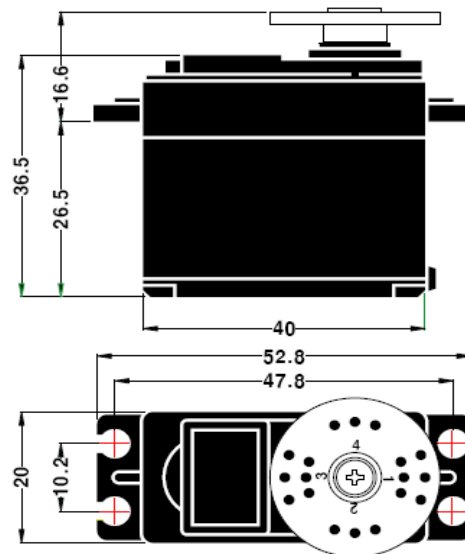
- Microcontroller = Atmel ATMEGA8-16PI
- EEPROM = 24LC32P
- Speed = 14.75 MHz
- Internal Sequencer = 12 Servo Hexapod (Alternating Tripod)
- Serial input = True RS-232 or TTL, 2400, 9600, 38.4k, 115.2k, N81
- Outputs = 32 (Servo or TTL)
- Inputs = 4 (Static or Latching)
- Current requirements = 31mA
- PC interface = DB9F
- Microcontroller interface = Header posts
- Servo control = Up to 32 servos plug in directly
- Servo travel range =  $\sim 170^\circ$
- Servo resolution = 1uS,  $.09^\circ$
- Servo speed resolution = 1uS / Second
- Servo motion control = Immediate, Timed, Speed or Synchronized.

PC board size = 3.0" x 2.3"

## APPENDIX H

### THE HS-322HD STANDARD DELUXE SERVO

<b>1. TECHNICAL VALUE</b>		
CONTROL SYSTEM	: +PULSE WIDTH CONTROL 1500u sec NEUTRAL	
OPERATING VOLTAGE RANGE	: 4.8V TO 6.0V	
TEST VOLTAGE	: AT 4.8V	AT 6.0V
OPERATING SPEED	: 0.19sec/60° AT NO LOAD	0.15sec/60° AT NO LOAD
STALL TORQUE	: 3kg.cm(41.66oz.in)	3.7kg.cm(51.38oz.in)
IDLE CURRENT	: 7.4mA AT STOPPED	7.7mA AT STOPPED
RUNNING CURRENT	: 160mA/60° AT NO LOAD	180mA/60° AT NO LOAD
STALL CURRENT	: 700mA	800mA
DEAD BAND WIDTH	: 5u sec	5u sec
OPERATING TRAVEL	: 40° /ONE SIDE PULSE TRAVELING 400u sec	
DIRECTION	: CLOCK WISE/PULSE TRAVELING 1500 TO 1900u sec	
MOTOR TYPE	: CORED METAL BRUSH	
POTENTIOMETER TYPE	: 4 SLIDER/DIRECT DRIVE	
AMPLIFIER TYPE	: ANALOG CONTROLLER & TRANSISTOR DRIVER	
DIMENSIONS	: 40x20x36.5mm(1.57x0.78x1.43in)	
WEIGHT	: 43g(1.51oz)	
BALL BEARING	: TOP/RESIN BUSHING	
GEAR MATERIAL	: 2 HEAVY DUTY RESIN	
HORN GEAR SPLINE	: 24 SEGMENTS/75.76	
SPLINED HORNS	: RESULAR/R-C,R-D,R-I,R-O,R-X,SUPER/R-XA	
CONNECTOR WIRE LENGTH	: 300mm(11.81in)	
CONNECTOR WIRE STRAND COUNTER	: 40EA	
CONNECTOR WIRE GAUGE	: 25AWG	



#### 2. FEATURES

LONG LIFE POTENTIOMETER, TOP RESIN BUSHING, 2 HEAVY DUTY RESIN GEARS

#### 3. APPLICATIONS

AIRCRAFT 20-40 SIZE, STEERING AND THROTTLE SERVO FOR CARS, TRUCK AND BOATS

#### 4. ACCESSORY & OPTION

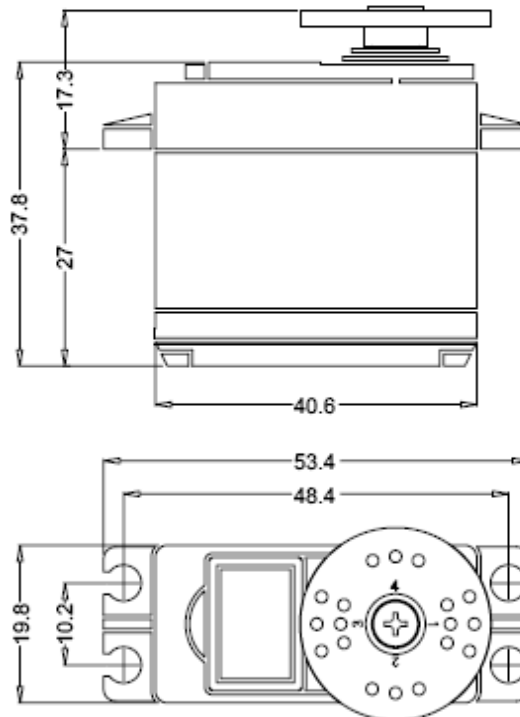
<b>CASE SET/</b>	<b>GEAR SET/</b>	<b>HORN SET/</b>
HS322T:1EA	HS322G1:1EA	R-C:1EA
HS322M:1EA	HS325G2:1EA	R-D:1EA
HS322L:1EA	HS325G3:1EA	R-I:1EA
PH/T-2 2x30 NI:4EA	HS322G4:1EA	R-O:1EA
	HS300RB:1EA	R-X:1EA
		R-XA:1EA
		WH/W 2.1x15 NI:4EA
		BST 3x5.5:4EA
		NBR 9x6.5x6:4EA

## APPENDIX I

### THE HS-645MG STANDARD DELUXE HIGH TORQUE SERVO

#### 1. TECHNICAL VALUES

CONTROL SYSTEM	:PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	:4.8V TO 6.0V	
OPERATING TEMPERATURE RANGE	:-20 TO +60°C	
TEST VOLTAGE	:AT 4.8V	:AT 6.0V
OPERATING SPEED	:0.24sec/60° AT NO LOAD	:0.2sec/60° AT NO LOAD
STALL TORQUE	:7.7kg.cm(106.93oz.in)	:9.6kg.cm(133.31oz.in)
OPERATING ANGLE	:45° ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	:CLOCK WISE/PULSE TRAVELING 1500 TO 1900usec	
IDLE CURRENT	:8.8mA	:9.1mA
RUNNING CURRENT	:350mA	:450mA
DEAD BAND WIDTH	:8usec	
CONNECTOR WIRE LENGTH	:300mm(11.81in)	
DIMENSIONS	:40.6x19.8x37.8mm(1.59x0.77x1.48in)	
WEIGHT	:55.2g(1.94oz)	



#### 2. FEATURES

- 3-POLE FERRITE MOTOR
- DUAL BALL BEARING
- LONG LIFE POTENTIOMETER
- 3-METAL GEARS & 1-RESIN METAL GEAR
- HYBRID I.C

#### 3. APPLICATIONS

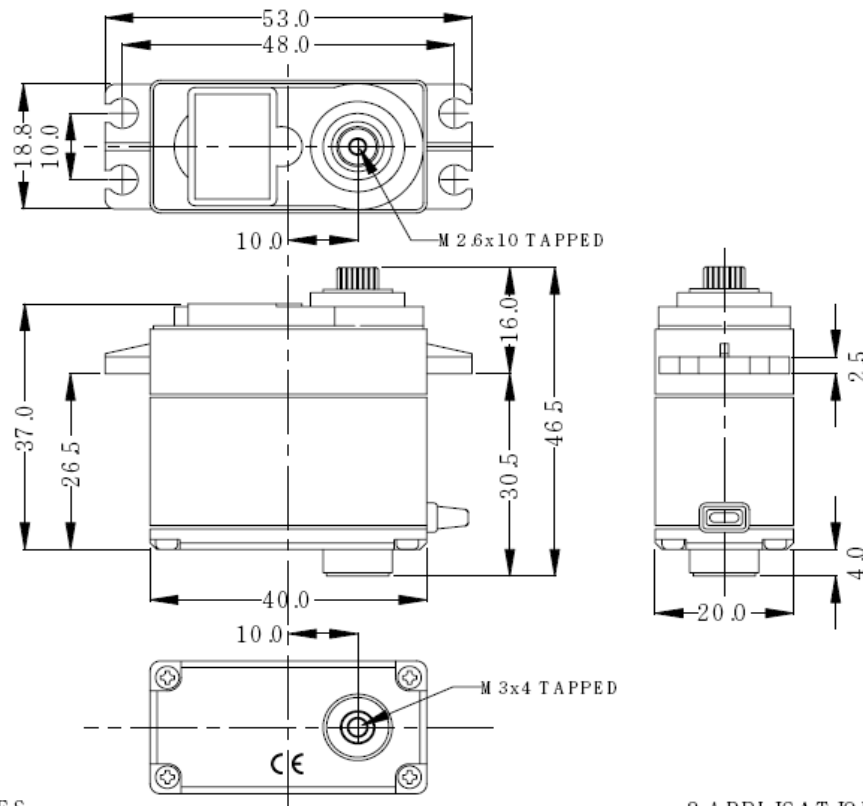
- AIRCRAFT TO 1/4 SCALE
- 30 TO 60 SIZE HELICOPTERS
- STEERING AND THROTTLE FOR 1/10TH & 1/8TH ON-ROAD AND OFF-ROAD VEHICLES

## APPENDIX J

### THE HSR-5995TG CORELESS DIGITAL SERVO

#### 1. TECHNICAL VALUE

CONTROL SYSTEM	*PULSE WIDTH CONTROL 1500usec NEUTRAL	
OPERATING VOLTAGE RANGE	4.8V TO 6.0V	
OPERATING TEMPERATURE RANGE	-20°C TO +60°C (-68°F TO +140°F)	
TEST VOLTAGE	AT 6.0V	AT 7.4V
OPERATING SPEED	0.15sec/60° AT NO LOAD	0.12sec/60° AT NO LOAD
STALL TORQUE	24.0kg.cm (333.29oz.in)	30.0kg.cm (416.61oz.in)
STANDING TORQUE	31.2kg.cm (433.27oz.in)/5° HOLD OUT	39kg.cm (541.59oz.in)/5° HOLD OUT
IDLE CURRENT	3mA AT STOPPED	3mA AT STOPPED
RUNNING CURRENT	300mA / NO LOAD RUNNING	380mA / NO LOAD RUNNING
STALL CURRENT	4200mA	5200mA
DEAD BAND WIDTH	2usec	2usec
OPERATING TRAVEL	90°/9°/ONE SIDE PULSE TRAVELING 400usec	
DIRECTION	CLOCKWISE / PULSE TRAVELING 1500 TO 1900usec	
MOTOR TYPE	CORELESS METAL BRUSH	
POTENTIOMETER TYPE	SLIDER / INDIRECT DRIVE	
AMPLIFIER TYPE	DIGITAL AMPLIFIER WITH MOSFET DRIVE	
DIMENSIONS	40x20x37mm (1.57x0.78x1.45in)	
WEIGHT	62g (2.18oz)	
BALL BEARING	DUAL/MR106	
GEAR MATERIAL	TITANIUM ALLOY	
HORN GEAR SPLINE	24 SEGMENTS/5.76	
SPLINED HORNS	REGULAR METAL/R-ML	
CONNECTOR WIRE LENGTH	300mm (11.81in)	
CONNECTOR WIRE STRAND COUNTER	60EA	
CONNECTOR WIRE GAUGE	22AWG	



#### 2. FEATURES

- PROGRAMMABLE DIGITAL AMPLIFIER WITH MOSFET DRIVE
- DURABLE TITANIUM ALLOY METAL GEARS WITH DUAL BALL BEARING
- ULTRA HARDNESS SHAFT WITH 3 AXIAL METAL BUSHING
- IP67 & DUST TIGHT
- ONE SIDE AXIAL MOUNT HOLE

#### 3. APPLICATIONS

ROBOTS

## REFERENCES

- [1]. Howard, A. & Tunstel, E. "Intelligence for Space Robotics". TSI Press, 2006
- [2]. Dulay, N. "Planetary Exploration: Navigation Methods and a Generation of Robots". Imperial College London, SURPRISE Journal Vol. 2 Article 2, 1995
- [3]. Powell, M. "Targeting and Localization for Mars Rover Operations". 2006 IEEE Conference on Information Reuse and Integration, 2006.
- [4]. Desai, R., Miller, D., "A Simple Reactive Architecture for Robust Robots". Jet Propulsion Lab, International Conference in Robotics & Automation, 1998
- [5]. Schenker, P.S., Huntsberger T. L., Pirjanian, P., Baumgartner, E. T., Tunstel, E. "Planetary Rover Developments Supporting Mars Exploration, Sample Return and Future Human-Robotic Colonization," Autonomous Robots, Vol. 14, pp. 103-126, 2003.
- [6]. Ch. Grand, F. BenAmar, F. Plumet, Ph. Bidaud. "Stability and Traction Optimization of a Reconfigurable Wheel-Legged Robot". The International Journal of Robotics Research, 2004.
- [7]. Kennedy, B., Agazarian, H., Cheng, Y., Garrett, M., Hickey, G., Huntsberger, T., Magnone, L., Mahoney, C., Meyer, A., Knight, J. "LEMUR: Legged Excursion Mechanical Utility Rover," Autonomous Robots, Vol. 11, No. 11, pp. 201-205, 2001.
- [8]. Ferrell, C. "Robust and Adaptive Locomotion of an Autonomous Hexapod". Perception to Action Conference, Lausanne, Switzerland, 66-77, 1994.
- [9]. Jun, S. "Kinetostatic Design of an Articulated Leg-Wheel Locomotion Subsystem". Master's of Science Thesis, Dept. of Mechanical and Aerospace Engineering, University of New York at Buffalo, 2004
- [10]. Craig J.J. "Introduction to Robotics, Mechanics and Control, 3rd ed". Addison Wesley, 1986, 1989, 2005.
- [11]. Braunl, T. "Embedded Robotics". Springer-Verlag Berlin Heidelberg New York, 2003.
- [12]. IEEE Potentials Magazine: Vol. 25, No. 1, January/February 2006.
- [13]. Lynxmotion, Inc. "SSC-32 Servo Controller Manual, Version 2.0", Pekin, IL, 2005.

- [14]. Fiorini, P. "Ground Mobility Systems for Planetary Exploration". IEEE International Conference on Robotics and Automation, 2000.
- [15]. Gullayanon, R. "Motion Control of a 3 Degree-of-Freedom Direct-Drive Robot". Master's Thesis, School of Electrical and Computer Engineering, Georgia Institute of Technology, 2005.
- [16]. Hiller, M., German, D. "Maneuverability of the Legged and Wheeled Vehicle ALDURO in Uneven Terrain with Consideration of Nonholonomic Constraints". International Conference on Control Automation Robotics and Vision 2002,
- [17]. Mireles Jr., J. "Kinematic Models of Mobile Robots". Automation and Robotics Research Institute, University of Texas at Austin, 2004.
- [18]. Endo, G., Hirose, S. "Study on Roller-Walker Multi-Mode Steering Control and Self-Contained Locomotion". IEEE International Conference on Robotics & Automation, 2000.
- [19]. Sciavicco, L. and Siciliano, B. "Modeling and Control of Robot Manipulators". McGraw-Hill Co., Inc. 2000.
- [20]. Barreto, J., Trigo, A., Menezes, P., Dias, J., Almeida, A.T. "Kinematic and Dynamic Modeling of a Six Legged Robot". Dept. of Electrical Engineering, University of Coimbra, 2004.
- [21]. Krovi, B. "Modeling and Control of a Hybrid Locomotion System". Master's Thesis, School of Engineering and Applied Science, University of Pennsylvania, 1995.
- [22]. Sciavicco, L., Siciliano, B., "A Solution Algorithm to the Inverse Kinematic Problem for Redundant Manipulator". IEEE Journal of Robotics and Automation, vol. 4, pp. 403–410, 1988.
- [23]. G. Figliolini, V. "Kinematic Model and Absolute Gait Simulation of a Six-Legged Walking Robot". CLAWAR (Climbing and Walking Robots) 889-896, 2004
- [24]. J. Lento, Z. Huson, J. A. Haass, M. Reilley, and J. Shrestha. "Development of Leg Control Mechanisms for A Radially Symmetric Octopedal Robot". The National Conference on Undergraduate Research (NCUR), 2005.
- [25]. Quinn, R., Nelson, G., Bachmann, R., Kingsley, D., Offi, J., Ritzmann, Roy. "Insect Designs for Improved Robot Mobility". 4<sup>th</sup> International Conference on Climbing and Walking Robots, 2001.

- [26]. Giancoli, D. C. "Physics for Scientists and Engineers", Prentice Hall, 3<sup>rd</sup> Edition, 2000.
- [27]. Ferrell, C., "Robust and Adaptive Locomotion of an Autonomous Hexapod", IEEE Perception to Action Conference, 1994.
- [28]. Howard, A., Nesnas, I., Werger, B., Helmick, D., "A Reconfigurable Robotic Exploratory Vehicle For Extreme Environments". Jet Propulsion Laboratory, TU Office NPO-20944, 2000.
- [29]. K. Iagnemma, A. Rzepniewskia, S. Dubowsky, P. Pirjanianb, T. Huntsbergerb, and P. Schenker, "Mobile robot kinematic reconfigurability for rough-terrain," in Proceedings SPIE's International Symposium on Intelligent Systems and Advanced Manufacturing, August 2000.
- [30]. Chottiner, J. "Simulation of a Six Wheeled Martian Rover Called the Rocker Bogie". Master's Thesis, Department of Mechanical Engineering, Ohio State University, 1992.
- [31]. Hacot, H. "Analysis and Traction Control of a Rocker-Bogie Planetary Rover," Master's Thesis, Department of Mechanical Engineering, MIT, 1998.
- [32]. Howell, A., Way, E., McGrann, R., Woods, R. "Autonomous Robots as a Generic Teaching Tool". 36<sup>th</sup> ASEE/IEEE Frontiers in Education Conference, 2006.
- [33]. Ferrari, M., Ferrari, G., Hempel, R. "Building and Programming LEGO Mindstorms Robots Kit". Syngress Publishing, 2002.
- [34]. Clifford, M. "Master Handbook of Electronic Tables and Formulas", TAB Books; 5<sup>th</sup> edition, 1992.