Implementation of a Variable Duty Factor Controller on a Six-Legged Axi-Symmetric Walking Robot

by

Steven J. Cutler

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2006

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Steven J. Cutler

Abstract

Hexplorer is a six-legged walking robot developed at the University of Waterloo. The robot is controlled by a network of seven digital signal processors, six of which control three motors each, for a total of 18 motors. Brand new custom electronics were designed to house the digital signal processors and associated circuitry. A variable duty factor wave gait, developed by Yoneda et al. was simulated and implemented on the robot. Simulation required an in-depth kinematic analysis that was complicated by the mechanical design of parallel mechanism comprising the legs. These complications were handled in both simulation and implementation. However, due to mechanical issues Hexplorer walked for only one or two steps at a time.

Acknowledgements

Special thanks are owed to my Supervisors, John McPhee and Eric Kubica. Without their support and encouragement very little would have seen fruition.

As for lab-mates, none could be better than Derek Wight. I thank him for his engaging conversation, insight, and Solidworks help.

Of course, the same is true of my family. I would like to thank them for their patience and support, which go hand-in-hand with answering the phone late at night.

Unfortunately, my clumsy and clinical words cannnot express my gratitude to Rita. Even when circuits were blown causing undoubtedly toxic vapour to fill the air, Rita gave me the reassurance and motivation I needed. I love you Rita, and I will get a job soon, I promise.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In 1997, a team of undergraduate students sought to design and build a six-legged walking robot in order to fulfill their final design project requirements. Following a trend in robotics [1, 28, 41], the robot was biologically inspired and took an insect form [20]. Divided in two by a sagittal plane, the first robot, Hexotica, featured serially manipulated legs powered by electric motors and controlled by a set of microcontrollers. While the team presented an impressive and ambitious plan to make this robot fully autonomous, their efforts were constrained by underpowered motors, and consequently the robot was not able to walk.

Another group of students took on the challenge of redesigning Hexotica. The new configuration consisted of a novel axi-symmetric design with a distributed computing architecture. The control system of the robot used Texas Instruments (TI) Digital Signal Processors (DSPs) and was entered in a TI design competition [6] and was also named NASA's 'cool robot of the week'.

Throughout the years, the problems solved by each successive group of students working on Hexplorer became markedly more tractable, though less ambitious. Although the robot was evolving, it still could not walk, and this eventually became the sole goal of the design team working on the robot [9]. In 2003, Hexplorer was entered in the Ontario Engineering Competition as a functioning, walking robot. A dynamic ADAMS simulation was created, where the feet were driven kinematically by periodic equations to create a tripod gait. These periodic equations were then transformed into joint positions using

inverse kinematics and a set of position controllers were used to ensure that all the joints arrived at the correct location at the appropriate time. This implementation resulted in a walking robot, but as can be seen in video footage of the robot, there were further improvements that could be made as the gait was fixed and the motion was jerky.

The next step in the design of the robot was to smooth out the gait. Unfortunately, in 2004, this was impossible with the existing electrical hardware, as all of the electronics had been damaged [8]. Improper voltage levels caused the DSPs flash memory to 'flash out' prematurely and repeated replacement of the DSPs damaged the circuit boards. A new circuit board design and hardware were needed.

The goal of the research behind this thesis was to enable the robot to walk with a smooth gait, improving upon the gait developed by Kwok and Cristello [9].

1.1 Goals

The directive to *make Hexplorer walk* was issued. While the task of making the robot walk seems simple and direct, the problem is actually quite general. Fast walking is a dynamic problem, where the inertia and velocity of the robot are critical to ensure that the robot remains upright. On the other hand, the robot may walk slowly enough that dynamic effects do not affect the balance of the robot. Other questions involve the topology and characteristics of the surface upon which the robot will operate. A perfectly level spongy floor in a laboratory will provide very different challenges than if the robot were exploring the Canadian shield. Once in its environment and ready to walk, a decision must be made regarding the path or heading to be taken. Does the robot decide, or is the decision made by a supervisor? The degree to which these decisions are delegated to the robot represent the degree of autonomy of the robot. Ideally, the robot would be totally autonomous, scouring hazardous environments for deposits of lucrative natural resources, all the while fending off attacks from wild animals. Realistically, however, simply following a supervisory input is quite tractable.

This research represents the addition of omnidirectional motion to Hexplorer, bringing Hexplorer one step closer to full autonomy. As such, the ability of the robot to walk should fit neatly into the standard autonomous model depicted in Figure 1.1. In this

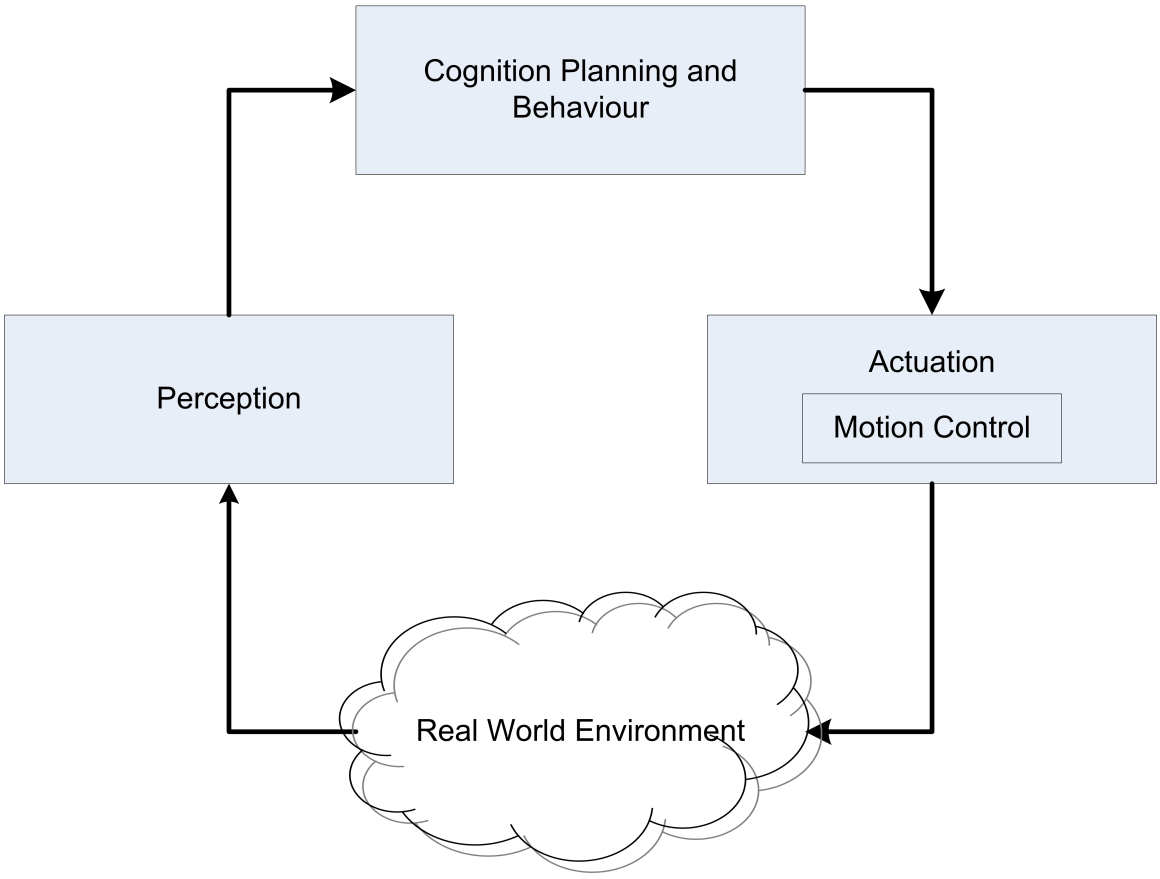


Figure 1.1: Standard autonomous model.

model, adapted from Siegwart and Nourbakhsh [39], data from the environment is sensed and processed until a plan or behaviour is selected and passed to the actuation module. If the plan or behaviour is to move the robot, then the actuation module applies a supervisory input to make Hexplorer walk. This is the interface between the research in this thesis and eventual autonomy of the robot, as the supervisory input can be specified by a user or calculated by the robot depending on its sensor input. In the research for this thesis, the supervisory input that specified the speed, heading, and yaw rate of the body of the robot was not generated autonomously, but instead by a user.

The motion of Hexplorer is generated by highly-geared electric motors. The high level of gearing indicates that speed is sacrificed in favour of torque and Hexplorer is limited to moving fairly slowly. Slow motion, coupled with high friction, as well as backlash in the couplings and gearboxes, ensure that any kinetic energy is readily consumed by the internal components of the robot. It is for these reasons, in addition to constraining Hexplorer to walk on level terrain, that a kinematic analysis and simulation of the robot is sufficient to fully describe its gaited motion.

To summarize the goals of the research for this, Hexplorer should move smoothly considering only the kinematics of the robot, on firm level terrain, under supervisory control.

1.2 Literature Review

The breadth and depth of research into robotics is astonishing. Robots range in size and shape, from the Ohio State Adaptive Suspension Vehicle (ASV) [41] that carries its operator, to a tiny inchworm robot [4]. Some robots have wheels [21], others rollerblade [5], trot, walk [1, 10, 15, 17, 19, 23, 16, 31, 48, 51, 52, 53], or run [36]. Construction of these robots varies as well, with some having compliant legs [36] and others having rigid ones [6, 53]. The methods of controlling and generating gaits for these robots are just as varied, ranging from fixed rigid gaits [31], to adaptive gaits that improve the static balance of the robot [52, 53], to gaits that focus on the dynamics of the robot [36, 23, 16, 51].

The literature review focuses on four- and six-legged walking robots and associated gait techniques. The gaits themselves can be classified into three broad categories including analytical gaits, evolutionary gaits, and heuristic gaits.

Almost all analytical gaits are based on the mathematical foundation developed by McGhee [30]. McGhee used principles from finite state machines to develop equations and notation to describe a particular gait. Song and Choi [40] extended McGhee's work, and investigated the static balance of robots that implemented wave gaits. They found that the wave gait optimally balances four-, six-, and eight-legged robots under certain conditions. This property makes the wave gait particularly appealing, and explains its frequent use [10, 17, 31, 48, 52, 53].

Many other gaits were derived from McGhee's mathematics. Song and Waldron [41] provide a comprehensive and in-depth survey of these gaits and implementation on the ASV. More recently, in 2003 Yee's doctoral dissertation [50] painstakingly details transitions between distinct gaits. Some of the work was not applicable to Hexplorer. Yee documents the transitions between a variety of follow-the-leader gaits. As discussed in Chapter 3, implementing follow-the-leader gaits is contrary to the design of Hexplorer because the robot does not have overlapping leg workspaces. Other work by Yee, such as the number of gait cycles and the leg resequencing required to change between straight-line and circular gaits, is relevant, but better handled by Yoneda et al. [52, 53] with their omnidirectional wave gait.

The most relevant and applicable work was completed by Lee and Orin [28], and Yoneda et al. [52, 53]. Before Lee and Orin's work, gaits were primarily unidirectional, and forward and backward motion, left and right motion, and rotation were handled by three distinct gaits. Lee and Orin were able to generalize these three distinct wave gaits and established an omnidirectional wave gait. This was an important contribution because only a single gait implementation was now required to achieve any motion. The relevance of this contribution is due to the axi-symmetric design of Hexplorer. Mechanically, Hexplorer is able to walk equally well in any direction, and Lee and Orin's omnidirectional wave gait achieves this mathematically.

The equations defining the omnidirectional wave gait allow Lee and Orin to introduce the concept of Constrained Working Volumes (CWVs). This concept prevents legs with overlapping workspaces from colliding with one another. In the omnidirectional wave gait, the size of the workspace affects many parameters. By providing the gait algorithm with a subset of the full workspace, the size of the steps taken by the robot can be modified.

Yoneda et al. [52, 53] capitalized on these contributions. Using the concept of CWVs and the omnidirectional gait, Yoneda et al. defined a variable duty factor wave gait. In this gait, the duty factor, or duration of a cycle in which a leg supports the robot, was selected to optimize the balance of the robot at a given speed. The result was that when moving slowly, a hexapod may only have one or two legs in the air at a time, but while moving quickly the gait became the familiar tripod gait. On a hexapod, the tripod gait always has exactly three legs supporting the robot at any given instant. Yoneda's analytic algorithm is particularly applicable to Hexplorer because he and his colleagues implemented the algorithm on an axi-symmetric hexapod. In addition, Yoneda's algorithm is based on a statically balanced gait, and only static gaits can realistically be implemented on Hexplorer, as it moves too slowly for the dynamics of the robot to affect its balance. The terrain is also an important consideration, and since Hexplorer operates in a laboratory environment and is at an early stage of gait development, level terrain is assumed. While an analytic model requires explicit knowledge of the terrain to perform optimally, an evolutionary gait may eventually be implemented and learn to deal with irregularities in the terrain resulting in an overall improvement in balance or speed over its analytic counterpart.

The next class of gaits surveyed were evolutionary gaits. Evolutionary gaits involve some component of learning or incremental improvement. Artificial neural networks [37, 29, 47], genetic algorithms [26, 32], and reinforcement learning [34] are some examples of the evolutionary techniques used. The research into gaits and artificial neural networks usually involves a central pattern generator specified by differential equations. These patterns specify leg timings and sequences, defining the gait. Because of the evolutionary process, the neural networks generating the gait can tune it to better achieve the desired results. In the case of Hexplorer, slightly modifying or altering the timings produced by an analytic gait would be pointless, because of the negligible dynamic effects and level terrain. Genetic algorithms follow a similar process. The genetic algorithms used to tune gaits are typically a special variety, named cyclic genetic algorithms [26, 32]. While this may be useful for fast moving robots that are dynamically balanced where the optimal gait may not be obvious, or, uneven terrain, the results of optimizing a statically balanced gait for speed on level terrain yielded the tripod gait, according to Parker [32]. This is the identical result of Yoneda's algorithm [52, 53]. For example, Hornby [19] tuned several parameters on the

Sony quadruped. He considered optimizing the distance travelled by the robot given a variety of parameters, such as foothold selections, and the location of the centre of mass of the robot with respect to its support legs. He devised a very clever learning environment where the robot executed gaits with these varying parameters. Upon completion of the trial by either reaching its destination or falling over, the robot would get up, return to the start position using a predefined gait, and incorporate the results of the trial into a new gait and repeat the same cycle. The most interesting concept presented by Hornby was that the robot could train itself without supervisory input. However, due to the quasi-static nature of Hexplorer's gait on smooth terrain, it is likely that the results of an evolutionary algorithm would provide obvious results, similar to Parker's [32].

Finally, a variety of robots use heuristics to define or improve gaits. All of the examples that follow provide interesting insights into the robots for which they are designed. Many are intricately linked and optimized to the architecture of the robot, or solve a specific problem with an existing gait on that robot. To this end, these algorithms are too specific or too advanced to be implemented on Hexplorer at the present time. Yoneda et al. [23, 51] fuse kinematics and dynamics using a zero moment point to dynamically control the balance of a quadruped. Shih and Klein [38] use heuristics to deal with peaks and valleys in the terrain; the gait is modified on the traditional analytical gaits developed from McGhee's work. Yang and Kim [49] investigate fault-tolerant gaits based on analytical gaits. The work is strictly theoretical, but does provide insight into how Hexplorer may be made fault-tolerant. Implementation would require re-derivation of many ideas for an axi-symmetric robot. Kinematic reconfiguration to optimize some aspect of a gait is also present in the literature. One of NASA's wheeled robots [21] changes the position of the centre of mass with respect to the tipping point of the robot to allow it to scale steep dunes. Chen and Yeo [4] present a similar technique for a legged robot. The performance of heuristic algorithms can vary greatly and depend heavily on the platform or configuration of the robot. Yoneda et al. [52, 53] conducted research on a different robot than discussed above. The depth of work, unequivocal results, and applicability to Hexplorer make the analytical gait presented by Yoneda et al. [52, 53] best-suited to Hexplorer.

Hexapode [17], Kafka [31], Hamlet [10], and the robot developed by Go et al. [15] are all digitally controlled hexapods. These robots all implement analytical gaits, most

frequently the tripod gait, despite aiming for different objectives. Hexapode fuses its gait with sensor readings to achieve quasi-static equilibrium in order to be balanced. Hamlet and the robot developed by Go et al., investigate closed-loop control of the body of the robot. The fact that these robots successfully implement analytic gaits is an indication of their effectiveness.

The effectiveness, static balance, knowledge of terrain, and optimality of Yoneda's adaptive gait algorithm make it ideal for implementation on Hexplorer.

1.3 Contributions

In the chapters that follow, the configuration of Hexplorer, Yoneda's gait algorithm and minor modifications, kinematics and the resulting simulation, and the overall implementation of the gait on the robot will be discussed.

The contributions of this research primarily include the implementation of Yoneda's algorithm on the unique architecture of the robot. The mechanical design of the robot poses special challenges: the forward kinematic equations for the legs and feet of the robot do not admit an analytical solution. Because of this, special care has to be taken to ensure that the forward kinematics can be determined iteratively to be implemented in real-time. In addition, modifications to some of the parameters in Yoneda's algorithm improve the distance travelled by the robot in a fixed number of steps, such as including the curvature of the path of the foot taken by the robot.

Chapter 2

Robot Configuration

This chapter describes the configuration of the robot from mechanical, electrical, and computing perspectives. The mechanical structure of the robot was inherited from previous work, whereas the electrical and computing platforms were redesigned for the explicit purpose of this research. The configuration of Hexplorer, in particular the axi-symmetric configuration of the six-legs and the parallel mechanism comprising each leg, makes it quite unique. Hexplorer is passively balanced and stays upright without power, uses parallel manipulators for legs, and is controlled using a network of digital signal processors.

2.1 Mechanical Configuration

The mechanical configuration of Hexplorer characterizes the uniqueness of the robot. Configured with an axi-symmetric design, the robot is afforded equal mobility in any direction. The body of the robot is comprised of two aluminum rings (Figure 2.1). These rings provide support points that define axes of rotation for the legs of the robot. The legs themselves incorporate a parallel mechanism that ably handles the vertical motion of the robot while also providing a second degree of freedom (DOF) that can contribute to the lateral motion of the robot. By rotating the legs about axes through the two rings, a third DOF is added to each leg.

The joint providing this third DOF is referred to as the hip joint. Rotation of the hip is created by connecting the leg to the body with an axle. This connection can be seen in

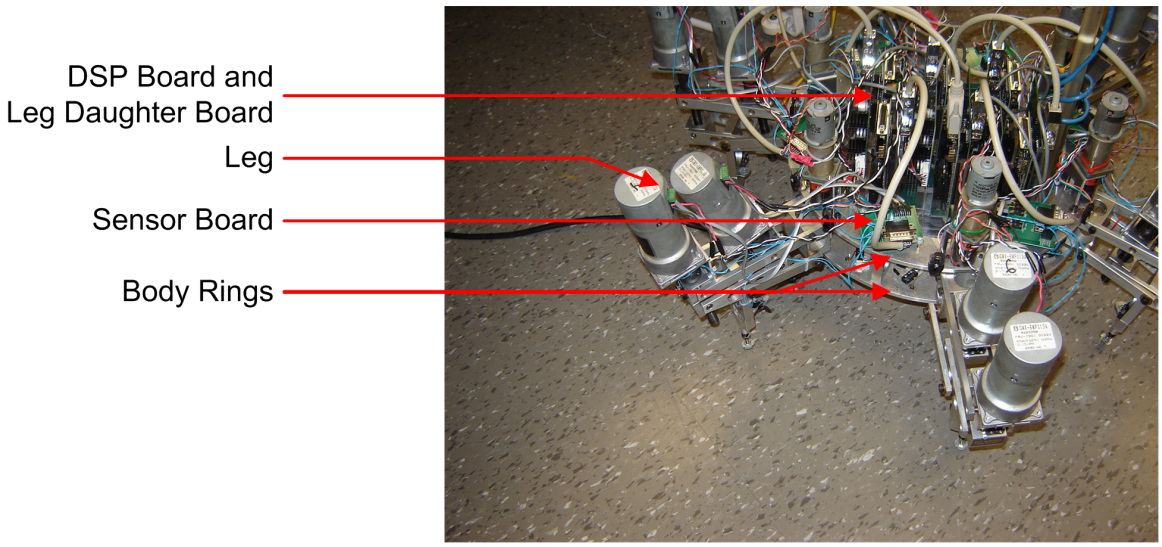


Figure 2.1: Hexplorer.

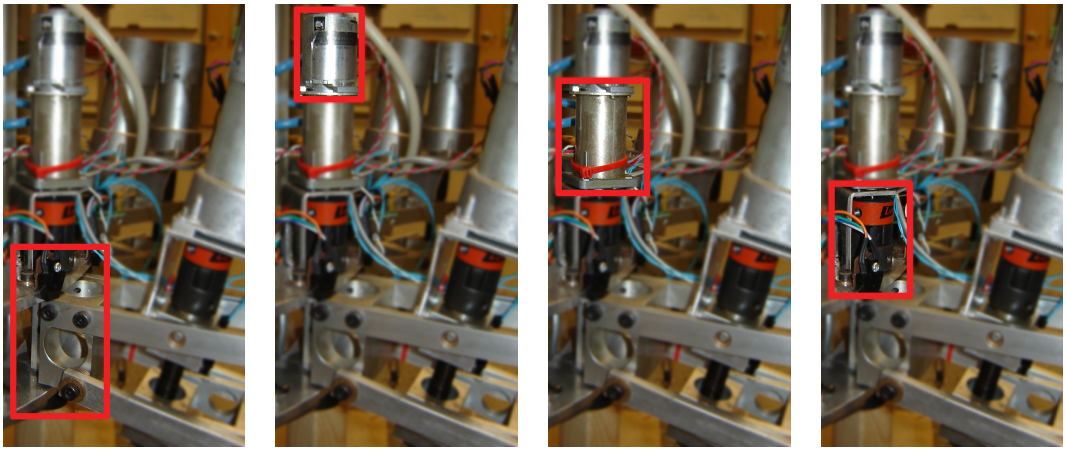
Figure 2.2. The rotation of the axle is constrained about an axis defined by two bearings that press-fit into the top and bottom rings. The reaction forces are provided by washers that hold the axle in place between these two rings. Unfortunately, this results in a large amount of friction in the joint where each leg is connected to the body. The friction for each leg also differs depending on whether the leg is supporting the weight of the robot or whether the leg is in the air. If the leg is in the air, gravity creates a moment that ultimately creates a reaction force in one direction on the washer. Alternatively, if the leg is supporting the body, the ground creates a moment and reaction forces in the opposite direction. Friction is not only different during supporting and airborne states, but is also different for each of the six legs on the robot. This is very much a factor when controlling velocity and also the position of the legs, although differences in controller parameters between legs are probably influenced more by the wear and deformation of the washers. A possible solution to this problem is to install thrust bearings held in place with stiff supports that can generate the necessary reaction forces to hold the leg in place, without yielding.

The rotation of the hip joint is produced by a DC motor connected to the rings that make up the body of the robot (Figure 2.2(b)). In the original design of the robot, the

support mechanism anchoring the motor to the body rings twisted because of large forces applied to it [9]. It was redesigned with better torsional stiffness properties and has since performed adequately. The DC motor is connected to a gearbox (Figure 2.2(c)) which is, in turn attached to each leg axle via a coupling (Figure 2.2(d)). The gearbox is not easily back-drivable because of the large gearing ratio. This is beneficial, but also potentially damaging to the robot. Because the gearbox is not back-drivable, the leg cannot be rotated when the motor is un-powered. However, if a large enough force is applied to the leg, gear teeth may be broken, damaging the gearbox.

An encoder is used to measure the rotation of the leg and is attached to the axle (Figure 2.2(e)). This DOF, or joint is referred to as the hip, and is measured as an angle. The range of the hip joint is defined by two limit switches that indicate the under-travel and over-travel configurations of the hip. The resolution of workspace range depends on the location of the encoder with respect to the gearbox. If the encoder was attached directly to the motor, located before the gearbox, it would have much better resolution. However, because it is attached to the axle, located after the gearbox, the resolution of the encoder is dramatically diminished. In fact, the encoder only rotates a fraction of a revolution between one end of the workspace of the leg, and another. Details of the encoder resolutions are discussed in Section 2.2.4. Another disadvantage of connecting the encoder to the end of the drive-train is that its measurements include backlash. Backlash is the hysteresis developed when switching between forward and reverse directions. It adds non-linearities to a system and complicates its control.

The remaining two DOFs on each leg are from the planar parallel revolute-prismatic-revolute mechanism (RPR) (Figure 2.3(a)). The mechanism has three revolute-prismatic-revolute (RPR) structures that achieve this planar motion. Two of the prismatic joints are driven by DC motors (Figure 2.3(b)) and lead screws (Figure 2.3(c)). These motors are each attached to a gearbox (Figure 2.3(d)) and a coupling (Figure 2.3(e)), similar to the configuration of the hip motor. Unlike the hip gearbox, these gearboxes are back-drivable, and the motor will rotate, if the output shaft of the gearbox is mechanically rotated. Lead screws are attached to the gearboxes and convert rotational motion to linear motion. The lead screw has two note-worthy characteristics. Lead screws are typically slow, but can be very accurate. They can also add significant friction to the joint, but this

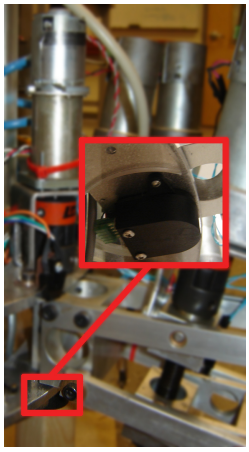


(a) Hip connection.

(b) DC motors driving hip.

(c) Gearbox driving hip.

(d) Coupling connecting gearbox to hip axle.



(e) Encoder measuring hip rotation.

Figure 2.2: Hip joint of a leg.

friction prevents the screw from being back-drivable. The end result is that the lead screw cannot be rotated by applying a linear force to it. Since the lead screw is not back-drivable (although the gearbox of the hip is back-drivable), the leg is able to retain its physical configuration without power applied to the motors. The third RPR member does not have a true prismatic joint, but is instead a fixed length (Figure 2.3(f)).

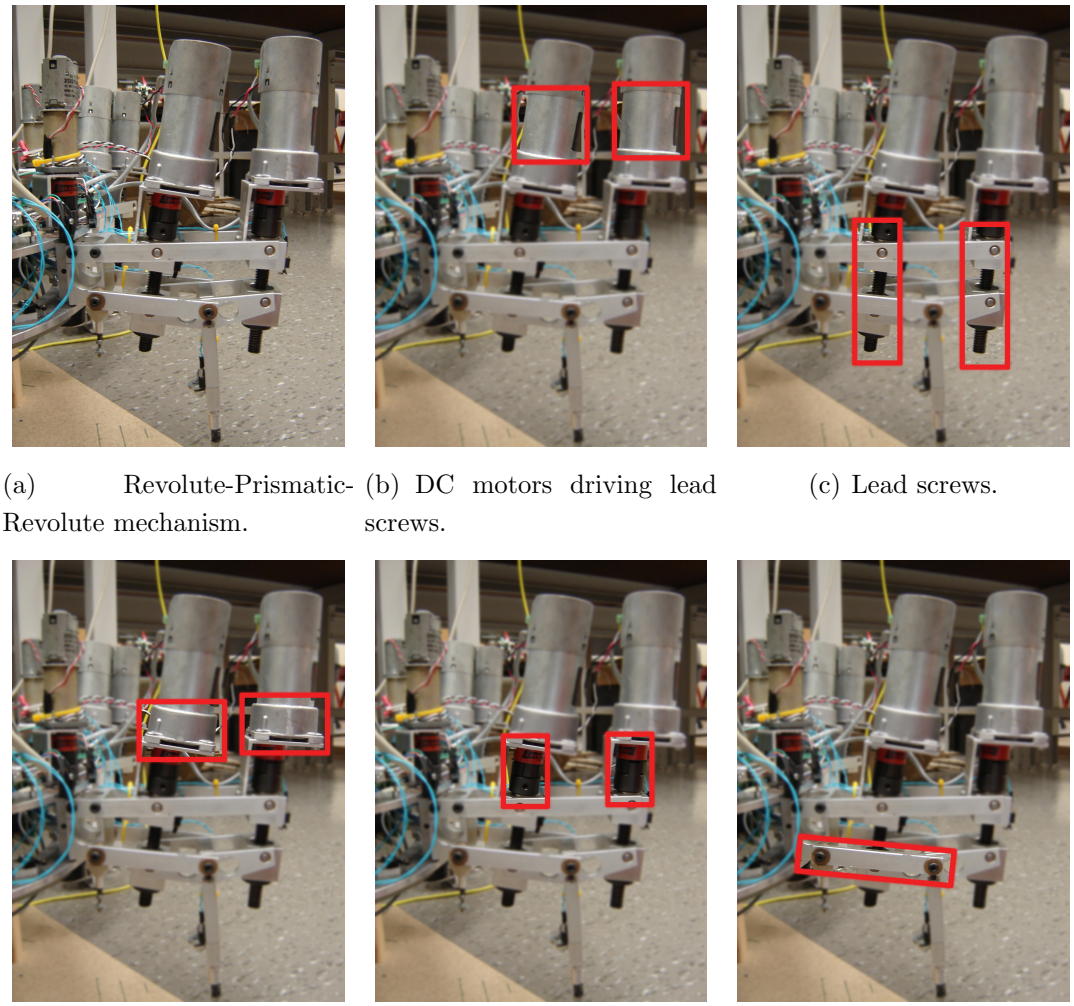
The length of each prismatic joint is determined from the rotation of the motor which is measured with an optical encoder (Figure 2.4). The encoder is attached directly to the motor, resulting in good resolution ($0.0011mm$/count) because several rotations are registered for a small change in linear displacement. Rotation is then converted into a linear displacement using the pitch of the lead screw.

With all three leg joints being passively static, the robot is afforded passive balance. Simply put, the robot will not fall down after being disconnected from a power source. This quality is favourable for debugging purposes because the robot can be halted safely by de-energizing its power source, but as mentioned above, a lot of friction is introduced into the system.

Attached to the bottom of each leg is a spring-loaded foot, shown in Figure 2.5. This provides the benefit of shock absorption properties when landing; however, it also effectively decreases the size of the vertical workspace.

2.2 Electrical Configuration

In the year 2000, Hexplorer underwent a hardware revision, in which control and computing was migrated to digital signal processor (DSP) chips manufactured by Texas Instruments (TI). Custom circuit boards incorporating the TI DSPs were manufactured by a team of undergraduate students. This enabled Hexplorer to handle the computational challenges of digital signal management and control. Unfortunately, due to a design flaw on the custom-designed circuit boards, the DSPs repeatedly 'flashed-out' after being re-programmed relatively few times, and subsequently lost the ability to be re-programmed. Fixing this problem required that the DSP chips themselves be replaced on the custom circuit boards. Repeated replacement deteriorated the circuit boards, until many of them ceased to function.



(a) Revolute-Prismatic-Revolute mechanism.

(b) DC motors driving lead screws.

(c) Lead screws.

(d) Gearbox driving lead screws.

(e) Couplings connecting gearboxes to lead screws.

(f) Fixed length instead of prismatic joint.

Figure 2.3: Revolute-Prismatic-Revolute mechanism forming a leg.

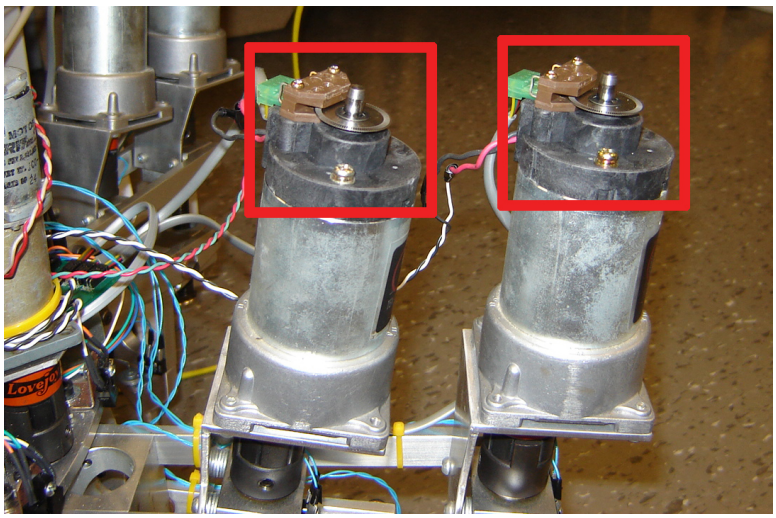


Figure 2.4: Encoders measuring lead screw displacement.

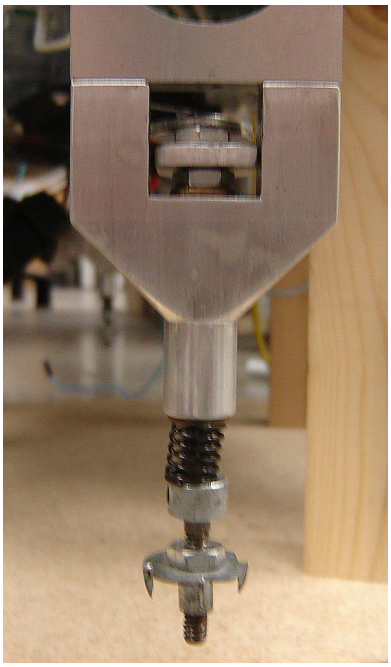


Figure 2.5: Spring loaded foot.

For this research and the overall longevity of the robot, the entire electrical system was overhauled. The design of the system was based on both the architecture of the 2000 circuit boards [6] and a prototype developed for a single leg in 2005 [8]. In 2005 a prototype board was designed with the goal of completely isolating motor circuitry and logic circuitry, as it was thought that electrically noisy motors had caused the DSPs to fail. Isolation was accomplished using opto-isolators in both a digital and analog manner. Digital opto-isolators use an electric signal from one region of circuitry to power an LED. A photo transistor then measures whether the original LED is active and returns the state of that signal to the other region of circuitry. An analog opto-isolator operates similarly, only instead of measuring the state (on or off) it measures the intensity of the LED. While many of the design flaws of the 2000 model were corrected [8, 9] and appeared to shield digital circuitry from the harsh environment generated by the motors, the redesigned 2005 prototype board was costly. Specifically, the designs in 2000 and 2005 required separate logic and motor power supplies. As development on Hexplorer proceeds, it may eventually become fully autonomous and use a single onboard power supply. If a single power supply is used, both regions of the circuit served by the opto-isolators will likely share a common ground. In this case, the noise generated by the motors would be transmitted to the logic circuitry, via the common ground plane, circumventing the entire purpose of the opto-isolators. Therefore, the expense of the opto-isolators is not justified when using a single power supply for the logic and motors.

The new, relatively cost-effective circuitry was redesigned with the goal of isolating as much electrical noise between the motors and logic as possible while using a single power source. Electric noise is mitigated by designing separate physical pathways for motor and logic current to travel from their source and back. With the exception of a few low-current signals, this design creates two separate circuit regions for logic and motor signals that are only connected at the power supply. The circuitry is divided into 5 segments: a power distribution board, leg daughter boards, sensor boards, a brain daughter board, and an off-the-shelf DSP board to parent each daughter board. The overall topology showing the interconnection of these components is shown in Figure 2.6. The voltage requirements are as follows: $24V$ for motors, $5V$ for the DSP boards, and $3.3V$ for the Brain board RAM, Ethernet, Bluetooth, and Compass. An overview of the resulting electrical system and

corresponding flow of current for the power distribution board, one leg board, and a sensor board, is shown in Figure 2.7. In this figure, the current flow for the logic and motor circuitry is clearly delineated and only share a single significant point of interconnection at the input of the power distribution board.

2.2.1 Power Distribution Board

The power distribution board requires an input of $24V$[1] from an external supply. It outputs $24V$ and $5V$ to each of the 6 DSP/Leg board combinations and $5V$ to the DSP/Brain board combination. The $24V$ input of the power distribution board is fed directly to the $24V$ output without any conditioning because this $24V$ line is only used to power the motors. Assuming the mechanical and electrical time constants of the motors are large when compared to high frequency noise that may be present in the $24V$ line, any noise in the $24V$ line will have little effect on the performance of the motors.

The $5V$ line is derived from the $24V$ input. Based on data-sheets [3, 25, 33, 42] and some preliminary testing, the DSP/Leg daughter boards require no more than $500mA$ of current per board. In addition, Table 2.1 summarizes the current requirements of the remaining peripherals attached to the robot. Incorporating a safety factor of 1.5, the power

Table 2.1: Logic-Circuitry Current Requirements

Component	Current Requirement
All 7 DSP boards with 6 leg daughter cards	$3500mA$
Bluetooth radio	$120mA$
Ethernet device	$233mA$
3-Axis Compass	$0.5mA$
Current sub-total (approx.)	$< 4A$
Total Current (safety factor of 1.5)	$< 6A$

distribution board is designed with the capability of supplying up to $6A$ at $5V$.

Successfully generating a $5V$ power source by reducing the $24V$ input depends on the current demand and operating ranges of voltage regulators. Two types of regulators were

[1] All voltage measurements are VDC.

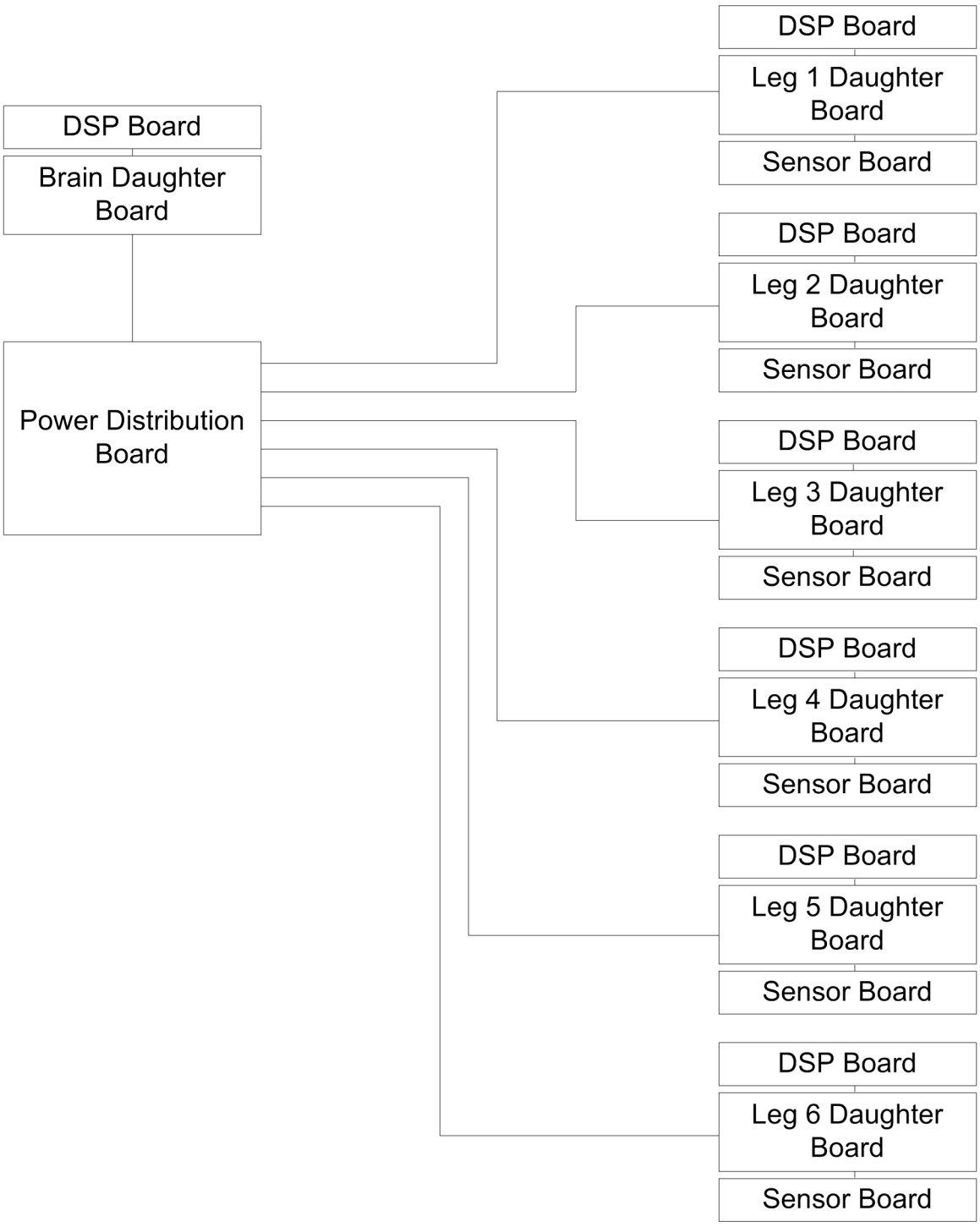


Figure 2.6: Hexplorer's power distribution topology.

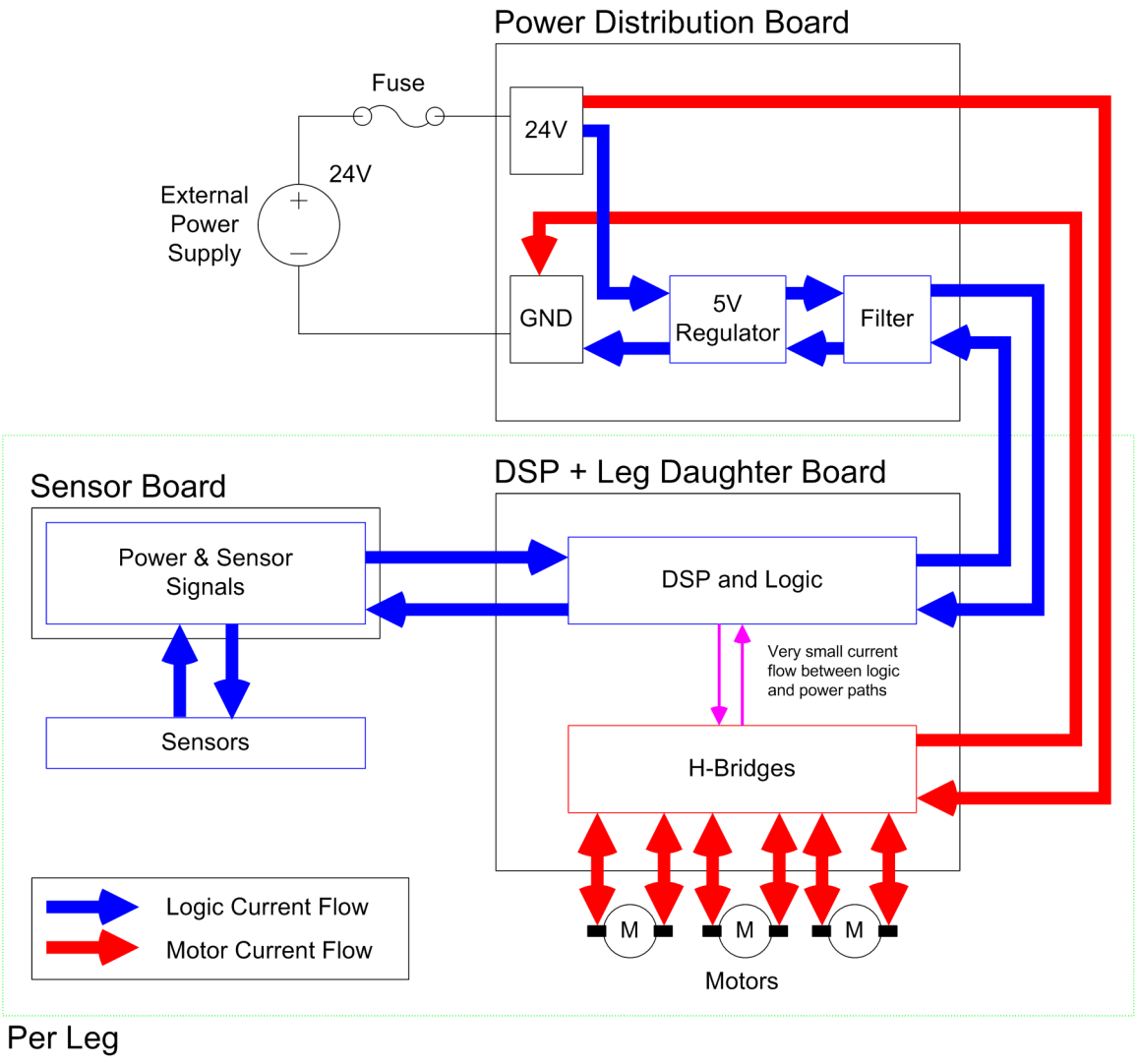


Figure 2.7: Current flow in Hexplorer's electrical system.

considered: linear and switching regulators. A linear regulator uses a linear feedback circuit to adjust the output voltage. It provides a clean output voltage and can source up to $10A$ of current [27]. However, a linear regulator can dissipate a lot of power, particularly when the input voltage is much higher than the output voltage. The excess power dissipated (P_d) by a linear regulator with an input voltage V_{in} and sourcing I_{out} amperes at V_{out} volts [44] is governed by

$$\begin{aligned} P_d &= (V_{in} - V_{out}) \times I_{out} \\ &= (24V - 5V) \times 6A \\ &= 114W. \end{aligned} \tag{2.1}$$

Such an enormous power loss (enough to power a bright light bulb) makes a linear regulator ill-suited for this task. Alternatively, a switching regulator could be used. A switching regulator uses semiconductors as on/off switches to produce the necessary output voltage. Access to the input voltage is turned on and off, such that on average, a specified voltage is obtained [27]. The output is then filtered in order to provide a relatively clean power source. Due to the switching nature of the regulator, the switching regulator is far more efficient and dissipates far less heat than its linear counterpart [27], given the voltage requirements of the robot. However, if not properly designed, the switching regulator can produce electromagnetic interference, which could seriously hamper the operation of wireless devices. A TI PTN78020W pre-built switching regulator circuit is used to ensure a proper design. Two of these regulators, each with the ability to source up to $6A$, provide ample power capacity for the logic of the robot. Two regulators are used because greater efficiency is achieved when operating each regulator in the $2A$ to $3A$ range as opposed to a single regulator in the $4A$ to $6A$ range. The voltage provided by these regulator circuits is further smoothed using a π-filter, as described in the product documentation [45]. One regulator powers three leg boards and associated DSPs, and the other powers three leg boards, the brain board and associated DSPs. Care was taken to provide parallel paths for $5V$, $24V$, and ground for each leg and the brain in an attempt to minimize electrical noise from one leg interfering with the circuitry of another.

The layout and schematics of the power distribution board can be found in Appendix A.

2.2.2 DSP Board

In order to eliminate the premature DSP failure experienced with the previous circuit board design, it was decided that an evaluation kit with a socket for the DSP would be the best solution. All of the required power regulation, filtering, and protection circuitry for the DSP was designed and tested on an evaluation board, in order to confirm that it worked correctly. In addition, many evaluation boards are also fitted with sockets to facilitate DSP replacement, if for some reason the DSP does fail or 'flash-out'. It was also decided that due to prior experience, the evaluation board should be based on the TI DSP platform.

The eZdsp^{TM} F2812 socketed evaluation board, manufactured by Spectrum Digital and available through Texas Instruments, meets all of these needs and features a Texas Instruments TMS320F2812 DSP [42]. Therefore, it was purchased and built into Hexplorer.

2.2.3 Brain Daughter Board

The brain of the robot is responsible for coordinating the leg board and handling supervisory input. The general purpose of the brain daughter card is to connect communication peripherals and memory to the DSP. The daughter card features 1MB of RAM, an ethernet communication device, a Bluetooth communication device, a CAN bus driver, and a compass. Each of these devices run at 3.3V, requiring a voltage regulator.

In this case, a linear regulator suffices because of the minimal voltage drop required from $5V$ to $3.3V$ as well as the limited current draw of less than $1A$. A fixed linear regulator, the TI TPS79633 capable of delivering $1A$ at $3.3V$, was selected to meet these needs [44].

The RAM is used for data storage. The brain is an ideal location for the RAM because almost all information regarding the states of the legs pass via the brain to the supervisory software.

The Lantronix XPort ethernet device is a network-to-serial device. It converts network data, using a TCP/IP protocol or UDP protocol, to a standard serial signal [25]. This device adds network capabilities to the robot, allowing the supervisory software to operate the robot from any computer attached to the same network as the robot. It also features a large bandwidth, up to 921.6kbps [25], that is ideal for reading data from the robot.

However, this device does require Hexplorer be tethered to an ethernet cable.

In the future, Hexplorer may become autonomous and having a cable tethered to the robot may be inappropriate. To meet this need, a BlueRadios BR-C11 Bluetooth device providing a wireless-to-serial connection was added to the robot. Unfortunately, it requires substantially more overhead than the Lantronix XPort ethernet device when handling connections, and as such, source code to support this device was removed during development as the complexity of the supervisory software grew. Since the robot requires a power tether anyway, wireless control is left for future work.

In another forward-looking move, a digital three-axis compass was purchased [33]. It is expected that in future, this compass will help provide feedback to the overall gait algorithm, closing the control loop. It is also anticipated that the compass will be used with the robot in an outdoor environment. Previous experience with digital compasses in the laboratory has shown that there is too much electromagnetic interference in the lab for the compass to provide consistent and accurate data. The software implementation of this compass, and its outdoor performance, should be evaluated in future work.

Finally, and most importantly, the brain daughter card features a Controller Area Network (CAN) driver chip to complement the built-in CAN device on the TMS320F2812 DSP. The driver chip provides a physical point of connection, connecting the CAN device on the DSP to a bus shared with similar driver chip/CAN interfaces on the Leg/DSP boards. The CAN enables inter-communication between all of the legs and the brain of the robot. CAN was originally developed to fulfill intercommunication needs between a variety of devices within an automobile [35]. CAN boasts impressive speeds of up to 1Mbps with very good noise rejection characteristics. Data is transmitted using a broadcast protocol and all devices attached to the bus receive all data transmitted on the bus. Individual devices simply ignore irrelevant data based on a number identifying the type of data encapsulated in the packet. For example, when the brain requires updates on the state of the legs, it places a single *send update* message on the bus, which, when received by the legs, execute the appropriate procedures and send updated information to the brain.

CAN communication was selected for a number of reasons. It supports the broadcast protocol and is available on TI DSPs. Unlike a raw serial specification like RS-485, it has a complex messaging scheme that includes error checking. In addition, the CAN bus

only has two signals. This means that only two wires are needed to interconnect the DSP boards.

The layout and schematics of the brain daughter board can be found in Appendix B.

2.2.4 Leg Daughter Board

The general purpose of the leg daughter boards is to connect the inputs and outputs of the DSP to electronic sensors and actuators, respectively. The requirements for the daughter board stem from the mechanical configuration of each leg. As described in Section 2.1, each leg has three DC motors each of which requires a driver, position sensor, and limit switches.

A National Semiconductor LMD18201 H-Bridge was selected to drive each motor. It contains circuitry to protect the chip and the rest of the circuit board from large current and voltage spikes caused by the motors in reaction to sudden stops, starts, or large step changes in input. This H-Bridge uses a pulse-width-modulated (PWM) signal to control the voltage applied to the motor as well as supply the current drawn by the motor. The PWM signal is a periodic square wave that operates at a specific frequency with a variable duty cycle. In this case, the duty cycle refers to the fraction of the period of the wave, in which the wave has a value of logic-high. The PWM input is a low-power signal supplied by DSP, and the H-Bridge uses this signal to activate/deactivate power electronics echoing the PWM signal to drive the motor. Because the dynamic time constant of the motor is large compared to the frequency of the PWM output wave, the resulting motion of the motor is smooth despite a pulsed energy source. The PWM signals sent to the three H-Bridges are generated by three independent PWM generators available on the TMS320F2812 DSP. The PWM generators are attached to internal timers in the DSP and the duty cycles are controlled by register values in the DSP.

The H-Bridges are mated to large heat sinks. The heat sinks adhere to the back of the H-Bridges and rest on regions of exposed ground plane. During operation, heat is drawn from the H-Bridge and dissipated to the air via convection. Heat is also dissipated via conduction, as the ground plane also absorbs heat from the heat sink. Kwok and Cristello [9] indicated in their report on Hexplorer that the H-Bridges on the previous circuit boards became hot to the touch. The larger heat sinks and use of a ground plane

as a heat sink prove to be a far more effective solution as the H-Bridge chips no longer get warm, even after extended use. It is a qualitative assessment, but it is very noticeable.

With the PWM signal from the DSP controlling the H-Bridges that drive the motors, feedback is required before the gait algorithm can be implemented. Position sensors for each motor take the form of encoders. These encoders produce a quadrature-encoded wave-train to convey a change in position or direction. Being digital, encoders are less susceptible to electrical noise than their potentiometer and analog-to-digital converter counterparts.

The US Digital E2-512-375-G [46] optical encoder used on each of the joints measures 4 changes in state of two waves and generates 2048 counts per revolution or 0.17° per count. Converting these values into joint measurements, the two lead screws have a total displacement of approximately 39000 counts or $4.3cm$, with the encoders measuring displacements of $0.0011mm$ per encoder count. The hip joint, which is measured on the other side of the gearbox, includes backlash and ranges by approximately 350 counts which is equivalent to 60°. The difference in resolution between the hip and lead screws is significant. The lead screw motor encoders are able to indicate resolutions about 111 times greater than the hip encoder. With such poor resolution, it is no surprise that velocity control of the hip motor is extremely complicated.

One disadvantage of using encoders as opposed to potentiometers is that these encoders measure relative position. In order to provide an absolute measurement, an index pulse or homing sequence is necessary. On Hexplorer the homing sequence is as follows:

1. The lead screws simultaneously retract until the under-travel limit switches of the inner and outer lead screw are activated.

2. The lead screws are now at known minimum lengths and these values are stored in the appropriate registers on the DSP.

3. The hip retracts until it reaches its under-travel limit switch and its minimum position is stored in the appropriate register.

4. Finally, because of backlash in the hip coupling, the hip is protracted until it reaches its over-travel limit switch. The hip is now at its maximum position, which is set to be 60° from its minimum position (as opposed to being determined by encoder counts).

Originally, Hexplorer had the following limit switches installed: one under-travel switch for each lead screw, one under-travel limit switch for the hip, and one limit switch to indicate ground contact. While the ground contact switch and under-travel limit switches for the lead screws were well located and functioned well, the single hip under-travel limit switch performed poorly. In its original position, the lever of the hip under-travel limit switch was bent each time the hip over-travelled. In fact, the damage caused when the hip over-travelled was compounded by the mechanical interference between the leg and the body of the robot. As shown in Figure 2.8, immediately before an over-travel collision, if the foot is extended far enough, the leg will collide with the lower ring of the body. This

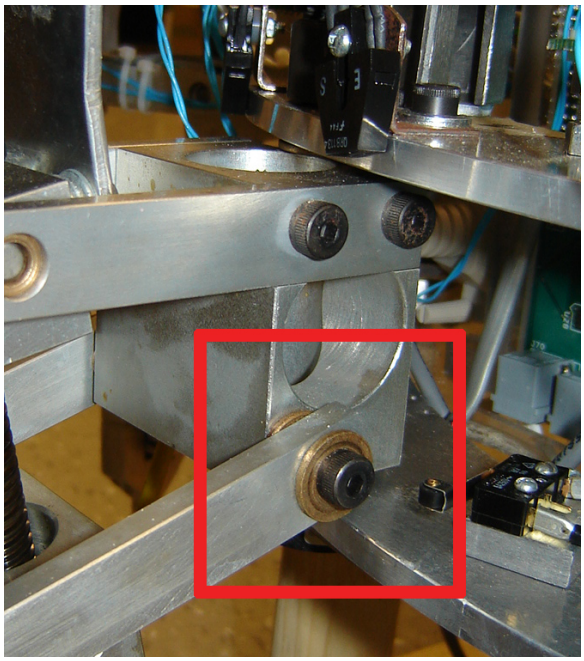


Figure 2.8: Leg collision with body rings.

type of collision is particularly destructive because the leg becomes "wedged" in between the two rings. Dislocating the leg from the rings is difficult and potentially damaging to the robot because if set screws are not loosened, the hip gearbox could be damaged by the large dislocating force. In this configuration, if the under-travel switch is not polled properly or the DSP experiences a spurious interrupt and halts, this collision could happen

on an under-travel as well.

To improve upon this design, the new leg daughter board has additional inputs for limit switches as well as Emergency Stop (E-Stop) limit switches. The original hip limit switch has been replaced with two infrared "soft limit switches". These switches, like the original, can only change the value of an input to the DSP. They are unable to physically or electrically stop the hip motor. These two infrared switches are used in the homing sequence. The E-Stop switches are mechanical limit switches located immediately outside the range of the infrared limit switches. Both infrared limit switches and one E-Stop limit switch are shown in Figure 2.9. When triggered, these E-Stop switches instantly activate

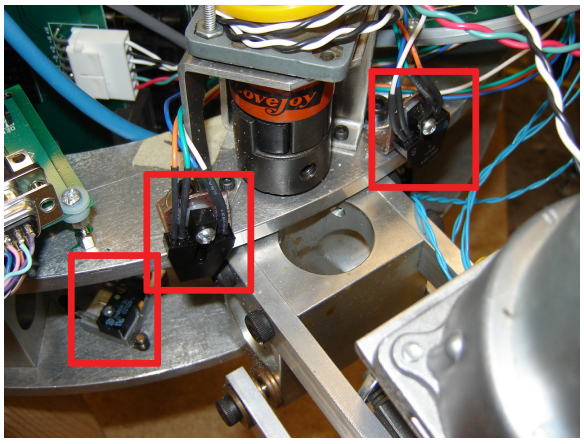


Figure 2.9: Two infrared limit switches and one mechanical E-Stop limit switch.

the brake input on all three H-Bridges halting all three motors. Latching circuitry was added to maintain electrical braking if the limit switch was only activated momentarily. This scenario would likely occur when a leg hits the limit switch and ricochets off it. Braking is maintained until the limit switch is no longer active and the DSP has reset the latch. The latching circuitry was constructed with simple sequential logic using AND and OR gate, as shown in Figure 2.10. This functionality proved very useful while debugging the robot. If a joint lost control, simply tapping one of the E-Stop switches safely deactivated the motors. With the addition of these switches, a leg has yet to collide with the body rings.

Electrically, the E-Stop switches were designed to be fail-safe. This is accomplished by

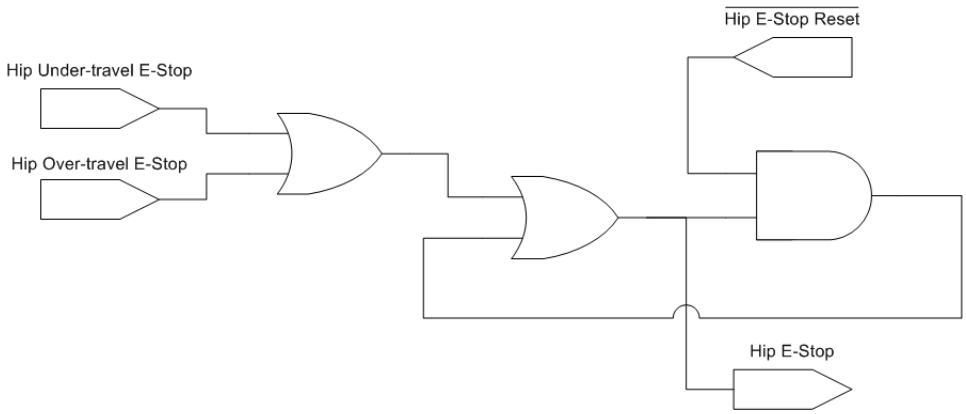


Figure 2.10: E-Stop latch circuitry.

wiring the mechanical switches to be normally closed, meaning that a signal of $0V$ indicates that an E-Stop has not been depressed. Depressing the E-Stop opens the mechanical switch, and the signal changes to $5V$. Similarly, if the wiring connecting the limit switches to the sensor board fails or breaks, the signal will also become $5V$ due to the pull-up resistor and trigger an E-Stop.

Mechanical limit switches generate noisy signals when closing, opening or even when exposed to vibration [2]. The digital circuitry accompanying the mechanical switch possesses a bandwidth orders of magnitude larger (microsecond vs. millisecond) than that of the mechanical switch. The digital circuitry, therefore reacts to mechanical noise as though it was a signal. Desensitizing the digital circuitry to this noise is known as de-bouncing the switch [2]. One method of de-bouncing an input involves adding a passive low-pass filter to the input of the digital circuitry. A simple RC filter, like the one shown in Figure 2.11, can suffice. The time constant of the RC filter, τ_{RC}, is selected to be similar to the mechanical time constant. Given that mechanical time constants of switches are in the millisecond range [2], and the electrical time constant is governed by

$$\begin{aligned} \tau_{RC} &= R \times C \\ &= 22k\Omega \times 10nF \\ &= 0.2ms, \end{aligned} \tag{2.2}$$

the resistor and capacitor values are selected as shown.

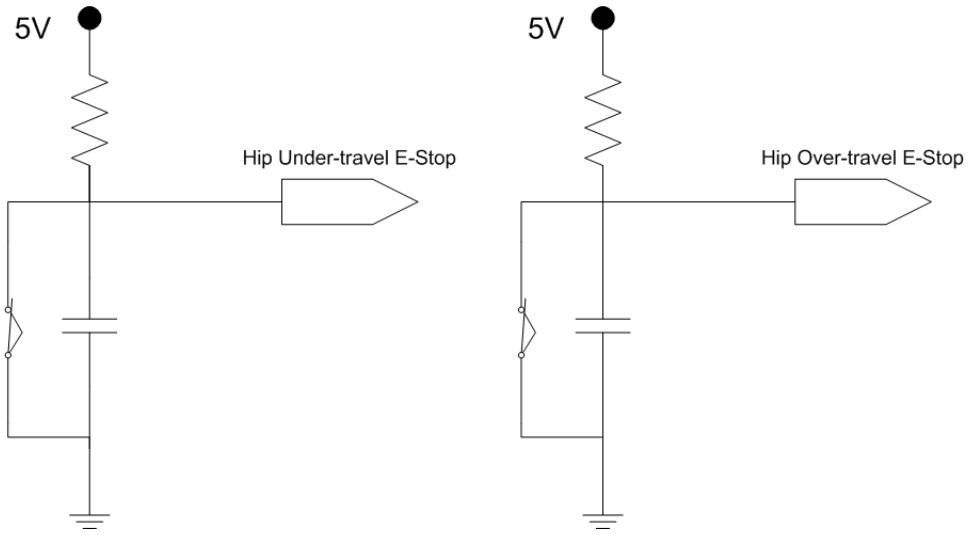


Figure 2.11: RC de-bouncing circuitry.

Like the brain daughter board, the leg daughter boards include a CAN transceiver to provide a link between the CAN interface on the DSP and the physical connections between boards.

2.2.5 Sensor Boards

The purpose of the sensor boards is to consolidate all of the wiring attached to leg sensors near the leg itself and have a single non-moving cable connect the sensor board to the leg board. In the previous circuit board design, sensors were connected directly to the leg boards. Some sensors required power and ground conductors in addition to the conductor required for the signal returned to the DSP. To keep the connectors on the leg board a reasonable size, power and ground wires for all sensors were first soldered together before being attached to a power pin and a ground pin on the connector. Attaching several wires to a small pin is physically awkward and prone to failure. In this configuration, the blobs of solder holding several power conductors together and several ground conductors together are essentially acting as power and ground planes. The sensor board resolves this problem by providing proper, mechanically sound, power and ground planes to which sensors can be connected.

The layout and schematics of the leg and sensor boards can be found in Appendix C.

2.3 Computing Configuration

In total, Hexplorer is configured with seven identical DSPs. One DSP is responsible for controlling and coordinating the actions of the remaining DSPs, which are each responsible for servicing a single leg. The DSPs must be powerful enough to handle the complex kinematics of the legs of the robot. In this circuitry revision, Hexplorer is equipped with the Texas Instruments TMS320F2812 DSP.

The Texas Instruments TMS320LF2812 150MHz DSP is a very capable processor. It features a number of memory-mapped on-chip devices specifically tailored for digital motor control, including, a set of multi-purpose timers, a set of PWM channels, two QEP decoding circuits, two serial interfaces, a SPI interface, and a CAN interface. It supports JTAG for online source code debugging and profiling. Profiling a block of source code or function involves counting the number of clock cycles required to execute that region of code. This is particularly relevant when dealing with floating-point operations, such as determining forward and inverse kinematics. The DSP is fixed-point and cannot execute floating-point operations in hardware. Floating-point operations must be emulated and can require several hundred clock cycles for a single multiplication or division. In fact, if implemented poorly, solving the forward or inverse kinematics of the robot using emulated floating-point operations could significantly reduce the maximum sample rate of the gait and joint controllers. None of the previous reports on Hexplorer [6, 8, 9, 24] investigated the effects of emulating floating-point operations on the sample rates of controllers, but it is investigated with great attention in the research for this thesis.

TI recognized the necessity of floating-point operations in some aspects of digital control and released a high performance floating-point library dubbed IQmath. The limitation of IQmath is that numbers are stored as fixed-point values where the precision of the number competes with its range. Table 2.2 indicates the two extremes of precision and range offered by this library [43]. Although individual operations using IQmath occur as floating-point operations, the resultant is stored as a fixed-point value. Fixed-point representations of numbers are even more prone to round-off error and numeric instability than floating-point

Table 2.2: Trade-off Between Precision and Range using IQMath Library [43]

Type	Min	Max	Precision
Most Precise	-2	1.999 999 999	0.000 000 001
Largest Range	-1073741824	1 073741823.500 000 000	0.5

representations. Round-off error refers to the inherent error in representing a rational number with finite precision [13]. Round-off error is compounded when it is propagated through a mathematical operation. If these errors predominately under- or over-estimate the true result, then the digital representation can drift significantly from the true value over time. Numeric instability, on the other hand, has a more immediate and devastating effect. Numeric instability in this context refers to operations that result in numbers that cannot be represented within the same range as the operands. Since the result cannot be represented, a minimum, maximum, or sentinel value, that can be represented, will be stored in place of the true result. Table 2.3 provides examples of the concepts of round-off error and numeric instability. The IQmath library may represent these numbers slightly differently.

Table 2.3: Fixed-Point Round-off Error and Numeric Instability

Number	Most Precise Representation	Largest Range Representation
$\frac{1}{3}$	0.333 333 334	0.5
$\frac{1.5}{0.5}$	1.999 999 999	3.0
1.5×1.5	1.999 999 999	2.0

A trade-off exists when using floating-point operations with a fixed-point TI DSP. True floating-point operations and representation can be emulated at the expense of computation time. Floating-point operations with fixed-point representation permits manipulation of rational numbers relatively quickly, but the range of the numbers is limited and results can be error-prone. Certain techniques can be used to mitigate the error and numeric instabilities introduced by fixed-point representation [13], but the effectiveness of these techniques is still limited by the finite precision of the DSP. An example of one such technique where variables a and b are fixed-point numbers, and $a + b$ exceeds the range of

fixed-point representation, the term $\frac{a+b}{2}$ should be represented by $0.5a+0.5b$ instead. Each complex mathematical equation that required floating-point operations was examined to achieve the best execution performance in terms of time and numeric stability.

2.4 Summary

This chapter summarized the configuration of Hexplorer from mechanical, electrical, and computing perspectives. Each of the six legs on the robot has 3 DOF, with each joint actuated by a DC motor. The motors are powered by H-Bridges located on custom designed circuit boards. The circuit boards have been designed to isolate as much noise as possible between the motor and logic, using a single power supply. Finally, the logic is powered by a Texas Instruments DSP with a high-speed floating-point library.

Chapter 3

Gait Algorithm

The wheel has become synonymous with convenience and speed and has revolutionized transportation for humanity [41]. Both efficient and simple, the wheel is able to create continuous locomotion. The simplicity of the wheel, however, is not without a profoundly negative aspect [41]. Although very efficient on a smooth surface, a wheeled vehicle demonstrates very limited mobility on rough terrain. This limitation has shaped society and is evidenced by incredible infrastructure around the entire world.

In contrast, the natural world favours a high degree of mobility and legged locomotion prevails. The large variation in the types of creatures that use legged locomotion is indicative of its robustness. Song and Waldron point out that legged locomotion carries creatures at speeds varying from standstill to upwards of $100km/h$ [41]. The range of weights of these creatures is equally impressive with ants being of negligible weight to large African elephants weighing $700kg$ [41]. Most crucial however, is that all of this movement occurs without any infrastructure on naturally occurring terrain. This bodes well when exploring environments such as caves, sea beds, or planets where it is either too dangerous or prohibitively expensive to provide infrastructure for the wheel.

While the motion of an animal's body may be nearly continuous, the motion of its legs is not. Legged locomotion requires that a foot lose contact with the ground in order to reset the position of the leg to continue to propel the animal. The coordination of the legs, and how and when the feet are lifted from the ground, is known as a gait [41]. The types of gaits are quite diverse and include horses galloping, humans walking, and spiders

climbing. Gaits used in robotics research are equally diverse.

Originally studied by Muybridge [30], understanding of the gait has improved dramatically over the years. Based on Muybridge's compilation of photographs published in the volume, 'Animals in Motion', McGhee [30] developed a mathematical foundation to quantitatively characterize gaits. Using principles associated with finite state machines, McGhee succinctly describes the gait as the ordering, subsequent timing and duration of each foot lifting off the ground. With a solid mathematical foundation, McGhee was able to create the first computer-controlled walking robot [41]. Primarily focussed on unidirectional motion, McGhee's research was later generalized to omnidirectional motion by Orin [28] and others. The specific gait algorithm implemented on Hexplorer was developed by Yoneda et al. [52, 53]. Its fundamental principles were derived from a vein of research explored by McGhee and Orin.

Three types of gaits and their properties were examined to determine which would suit Hexplorer best.

A free gait [41] is, as it sounds, free. It is not periodic and does not require any consistency between separate foot steps. A free gait functions by selecting optimal, or at least hospitable, foot-holds. This affords a robot with the ability to traverse rough terrain, albeit slowly. While an impressive gait to witness, it is inappropriate for Hexplorer and this research because of inadequate vertical range and lack of sensory data. In order to operate in rough terrain, the workspace of the leg of the robot should permit a large heave in order to clear peaks in the terrain. The parallel mechanism comprising each leg of Hexplorer provides relatively little ground clearance (*i.e.*, 4 to 5 centimetres). This problem is compounded by two factors. First, spring-loaded feet are attached to each leg and further reduce ground clearance. Second, the body of Hexplorer is not articulated and cannot increase the vertical displacement of a leg by having one segment pitch. When operating in a reasonable workspace region, each foot has only 3 to 4 centimetres of ground clearance, or about 12% of the height of the robot. Sensory information regarding pitch, roll, and yaw of the body is necessary to maintain the balance of the robot on uneven terrain. Pitch, roll, and yaw are rotations defined between a body-fixed axis and an inertial axis as shown in Figure 3.1. Although Hexplorer is equipped with a three-axis compass, electro-magnetic interference in the laboratory renders it essentially useless. The

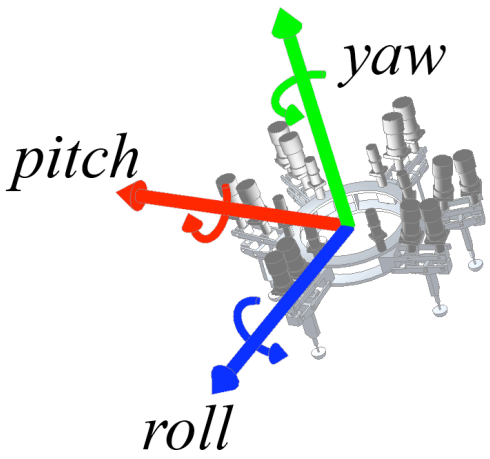


Figure 3.1: Definitions of pitch, roll, and yaw.

use of foot force sensors in a similar manner to Goulet [17] could be used to provide this information in a laboratory environment, but Hexplorer is not equipped with these sensors. Should a new revision of Hexplorer be developed, it may then become logical to examine and implement a free gait.

Another type of gait is a follow-the-leader gait [41]. In this periodic gait, only the foot-holds of the lead foot or feet are determined by the supervisor (human or artificial). All other feet simply relocate to the foot-hold vacated by the preceding foot. This is advantageous because given rough terrain, once the controllers of the lead feet have determined suitable foot-holds, the remaining foot controllers need not apply any processing power to finding foot-holds. Being able to occupy the foot-hold left vacant by an adjacent foot requires that the workspaces of these legs overlap. However, since the leg workspaces of Hexplorer do not overlap with adjacent ones, it circumvents the benefit of this gait.

The third type of gait investigated is the wave gait [41]. It features periodic motion and is well-suited for level terrain. This gait functions by dividing the legs of the robot into two groups. Two separate waves then propagate touch-down events to the feet of the robot. These events are timed such that as a leg in one group touches down the leg in front

of it lifts off. More importantly, this gait provides the ability to tune parameters offering a trade-off between speed and balance. Yoneda et al. proposed a method to dynamically modify these parameters achieving optimal balance of the robot [52, 53]. The smooth periodic motion and superior balance generated by Yoneda's modified wave gait make it well-suited to control Hexplorer. The full details of the wave gait and the modifications suggested by Yoneda are presented in Section 3.1.

3.1 Yoneda's Algorithm

The principle behind Yoneda's algorithm is rather elegant. It is based on the premise that a robot is best balanced with the most number of feet on the ground. Thus, when moving, a robot should maximize the number, and duration of, feet on the ground by minimizing the time in which each foot is airborne. Minimizing the airborne duration of the foot is accomplished by maximizing the speed of the foot. This objective is achieved within the framework of a traditional wave gait.

3.1.1 Definitions

In order to understand the underpinnings of a wave gait, important definitions are presented below.

A *statically stable* gait is one in which the inertial forces of the walker are negligible. Due to its significant weight, slow speeds, and very large amount of friction, Hexplorer is only capable of implementing a statically stable gait.

The *support phase* of a leg describes the duration of the gait in which the foot of the leg is in contact with ground. It is represented as the phase variable ϕ_{Si}, for the i^{th} leg, which varies from 0 at foot touch-down, to 1 at foot lift-off [28].

The *transfer phase* of a leg describes the duration of the gait in which the foot of the leg has been lifted off of the ground. It is represented as the phase variable ϕ_{Ti}, for the i^{th} leg, which varies from 0 at foot lift-off, to 1 at foot touch-down [28].

The *duty factor* of a gait is the fraction of a cycle in which the leg is in its support phase [41]. It is represented as the variable β. In a statically stable gait, it ranges between $\frac{1}{2}$ and 1 . With a duty factor near 1, the robot moves very slowly and usually has all of

its legs on the ground, making it well balanced. With a duty factor of $\frac{1}{2}$, a hexapod moves with a tripod gait. That is, at any instant, three legs are in a support phase and three legs are in a transfer phase. A tripod gait can achieve a maximum speed in a statically stable walker, but with only three legs in the support phase, it is also the least stable wave gait.

The *kinematic phase* of the robot, ϕ, is a fraction representing the completion of the current gait cycle. It varies between 0 and 1.

The *relative phase* of the i^{th} leg is the value of the kinematic phase when the foot touches down. It is represented as the variable ψ_i.

The *kinematic period*, τ, is the time required to complete one cycle of a periodic or semi-periodic gait [41].

The *temporal kinematic margin* of leg i, t_{Si}, is the amount of time remaining before the foot reaches its workspace boundary.

The *crab angle* of a wave gait, α, refers to the angle between the heading of the robot and the leg designated as number one. As the heading of the robot changes, so too does the crab angle and consequently ordering of the lift-off events. Reordering the lift-off events helps to maximize the balance of the robot [53].

3.1.2 Overview

The overall goal of Yoneda's algorithm is to realize motion given a desired motion command. As indicated in Section 1.1, this motion control algorithm fits well into the standard autonomous model. On a typical wheeled robot, motion control can be achieved using straightforward kinematics. The reciprocating nature of the legs on a walking robot increases the complexity of realizing motion control. Yoneda inserts gait and foot motion planners before the kinematics to manage the increased complexity [52, 53]. Input speeds and directions, a gait planner, a foot motion planner, a kinematics-based controller, and output are sufficient to realize motion on a walking robot. Yoneda provides a block diagram similar to Figure 3.2 to describe his algorithm; this section of the thesis focuses on the 'Gait Planner' block.

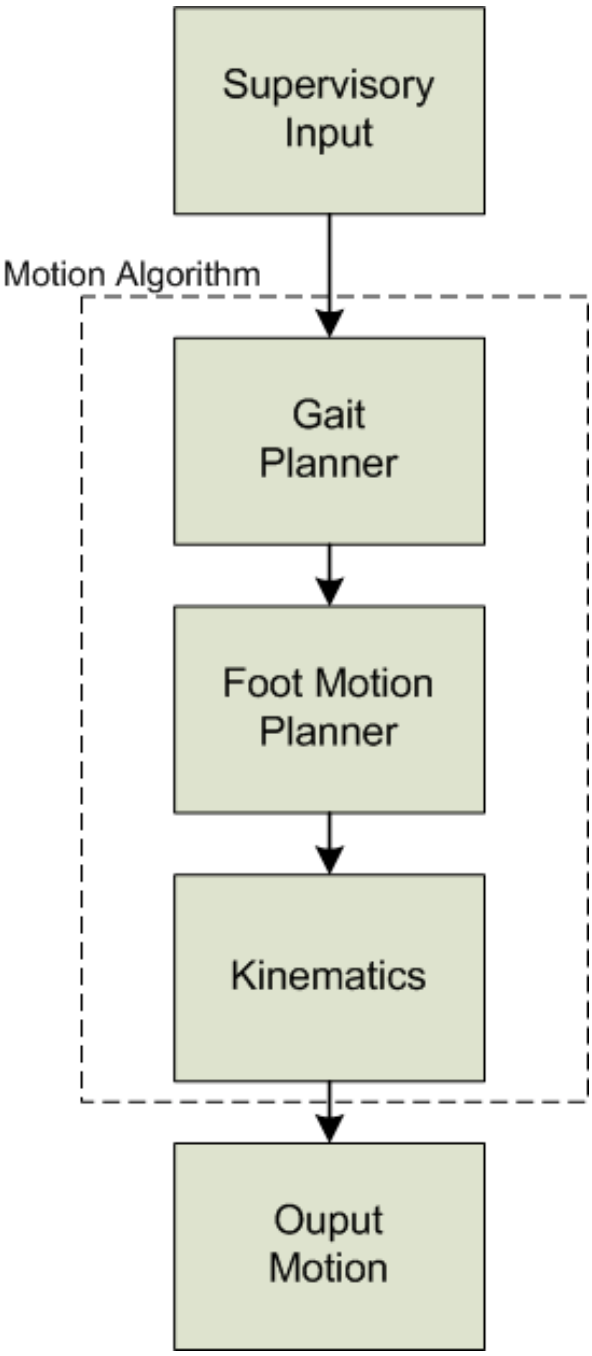


Figure 3.2: Gait algorithm overview.

3.1.3 Motion Input

The input to this system is defined as two horizontal velocity vectors, and the yaw rate of the body of the robot [52, 53]. Roll, pitch, as well as heave are held constant. The input is depicted in Figure 3.3 and generally defined as

$$\mathcal{I}(t) = \left\{ \begin{array}{c} v_x(t) \\ v_z(t) \\ \omega_y(t) \end{array} \right\}. \tag{3.1}$$

$v_x(t)$ and $v_z(t)$ are velocity components of the body of the robot, as measured in a frame fixed to the body. $\omega_y(t)$ is the angular velocity, or yaw rate of the body of the robot.

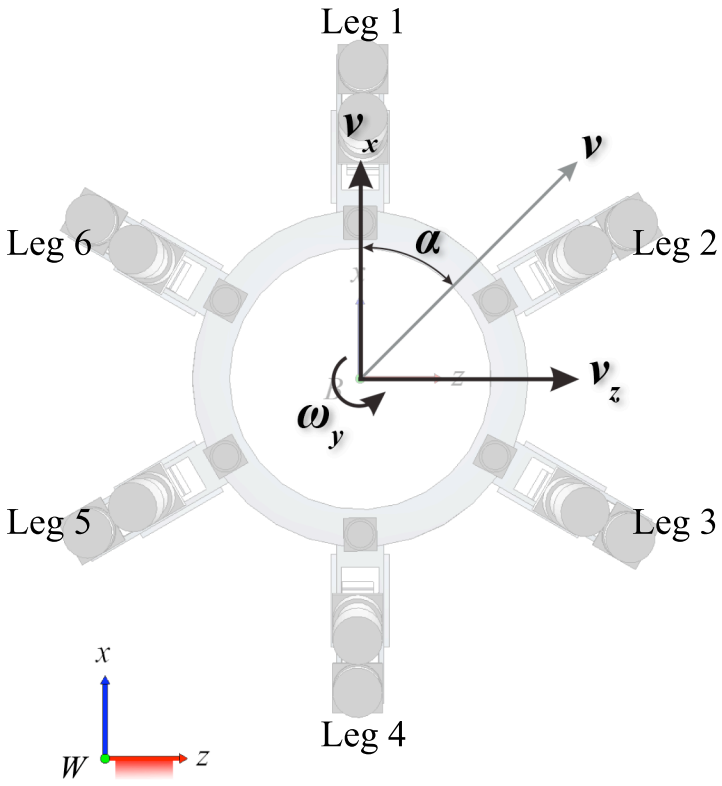


Figure 3.3: Motion input commands to Hexplorer.

The gait controller is responsible for translating the input and current state of the robot

into a set of leg states that can be used by a kinematic controller to determine relative Cartesian velocities for each foot. Yoneda's algorithm differs from a traditional wave gait in that it permits a variable crab angle and a variable duty factor. The crab angle of the robot, $\alpha(t)$, is depicted in Figure 3.3 and defined by

$$\alpha(t) = \tan^{-1}\left(\frac{v_z(t)}{v_x(t)}\right). \tag{3.2}$$

3.1.4 Conventional Forward Wave Gait

A forward wave gait divides the legs of a robot into two groups. In a conventional forward wave gait, this division is specified by the sagittal plane of the robot. On the other hand, for an axi-symmetric robot, Yoneda defines the crab angle, α to specify this division.

Having divided the legs of the robot into two groups, two waves, one per group, then propagate foot touch-down events through each group. Touch-down events are required to propagate forward, from the hindmost leg to the foremost leg in the direction of the desired heading. In addition, the touch-down events should be timed such that as one leg touches-down, an adjacent leg in the same group lifts-off. If the gait has a duty factor of β, then this corresponds to a timing or phase difference of $1-\beta$ between adjacent legs in a group. Finally, a conventional wave gait requires that the touch-down events of the two groups are out of phase, or have a phase difference of $\frac{1}{2}$. Yoneda, however, notes that this definition is only true for axi-symmetric robots with crab angles of $\alpha \in \frac{(2i-1)\pi}{6}, i = 1 \ldots 6$ [53].

In the following sections, the modifications Yoneda makes to the conventional forward wave gait is presented. Although the modifications are presented in [52, 53], details will be discussed here to benefit the reader. Yoneda proves that the final variable duty factor gait developed is in fact a legitimate forward wave gait or conventional forward wave gait, using rules derived from the definition above. A gait is considered to be a forward wave gait if:

1. touch-down events in a group propagate from the hindmost leg of the robot, to the foremost leg;

2. the phase difference between adjacent legs in a group is $1-\beta$;

In addition, a gait is considered to be a conventional forward wave gait if:

3. the conditions above, describing a forward wave gait are met;

4. the crab angle is, $\alpha \in \frac{(2i-1)\pi}{6},\ i = 1 \ldots 6$;

5. the phase difference between the two groups is $\pm\frac{1}{2}$;

To illustrate the process of determining the validity of a conventional forward wave gait, consider the following mathematical example. The axi-symmetric hexapod depicted in Figure 3.4 has duty cycle of $\frac{1}{2} \leq \beta \leq 1$, and a crab angle of $\alpha = \frac{\pi}{6}$, with touch-down timings of some legs given by

$$\phi_m = \left((7-m)\,\beta\right)_{mod1} \text{ for leg } m, \text{ with } m = 5, 6, \tag{3.3}$$

that has been adapted from Song and Waldron [41]. In this equation, ϕ_m refers to the

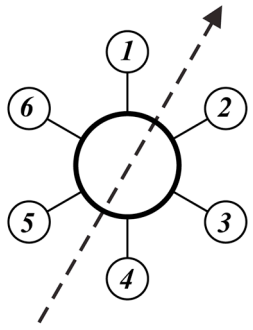


Figure 3.4: Axi-symmetric hexapod with $\frac{1}{2} \leq \beta \leq 1$, and $\alpha = \frac{\pi}{6}$

value of the kinematic phase, ϕ, when leg m touches down, using the definition

$$x \mod y = \begin{cases} \left(x - \left\lfloor \frac{|x|}{y} \right\rfloor \times y\right), \text{ if } x \geq 0 \\ \\ \left(\left\lceil \frac{|x|}{y} \right\rceil \times y - x\right), \text{else} \end{cases} . \tag{3.4}$$

Since the crab angle is $\alpha = \frac{\pi}{6}$, condition (4) is satisfied. The crab angle has divided the legs into two groups, with legs ordered from hindmost to foremost, $< 5, 6, 1 >$ and $< 4, 3, 2 >$. Touch-down phase values are given for legs 5 and 6 by Equation (3.3) evaluating

to $\phi_5 = (2\beta)_{mod1}$ and $\phi_6 = \beta_{mod1}$. To determine whether these values satisfy condition (2), subtract ϕ_5 from ϕ_6 as shown

$$\begin{aligned} \phi_6 - \phi_5 &= \beta_{mod1} - (2\beta)_{mod1} \\ &= \beta - (2\beta - 1) \\ &= 1 - \beta. \end{aligned} \tag{3.5}$$

Based on these results, condition (2) is satisfied. The phase value for leg 1 can now be obtained by applying the results from Equation (3.5) to legs 1 and 6, yielding

$$\begin{aligned} \phi_1 - \phi_6 &= 1 - \beta \\ \phi_1 &= (1 - \beta + \beta_{mod1})_{mod1} \\ \phi_1 &= 1_{mod1}. \end{aligned} \tag{3.6}$$

Since $\phi_5 < \phi_6 < \phi_1$ the order of touch-down events proceeds from hindmost to foremost leg and condition (1) is satisfied. Touch-down phase values have now been determined for the entire group of legs $< 5, 6, 1 >$. Next, apply a phase difference of $+\frac{1}{2}$ to calculate the values for the second group. This satisfies condition (5), the only remaining condition. The touch-down phase values are summarized in Table 3.1 after evaluating the $mod1$ operator.

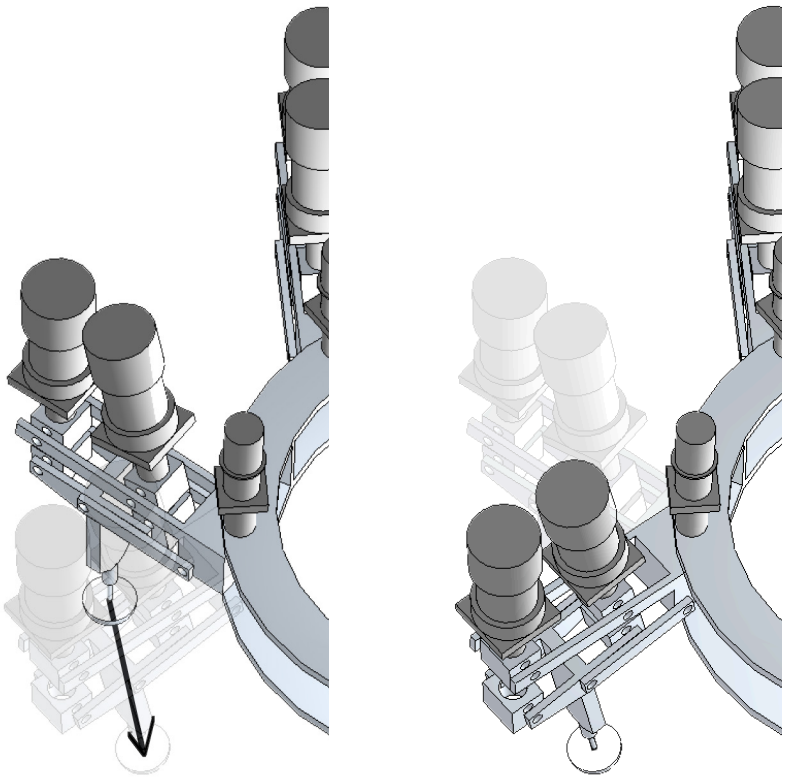
Table 3.1: Touch-down Phase Values for a Conventional Wave Gait $\left(\alpha = +\frac{\pi}{6}\right)$

Group $< 5, 6, 1 >$	Group $< 4, 3, 2 >$
$\phi_1 = 0$	$\phi_2 = \left(0 + \frac{1}{2}\right)_{mod1}$
$\phi_6 = \beta$	$\phi_3 = \left(\beta + \frac{1}{2}\right)_{mod1}$
$\phi_5 = 2\beta - 1$	$\phi_4 = \left(2\beta - \frac{1}{2}\right)_{mod1}$

Having satisfied conditions (1-5), it can be concluded that Equation (3.3) does indeed generate a conventional forward wave gait. These conditions will be used later to demonstrate that the gait generated with a variable crab angle can indeed be classified as a forward wave gait, and as such, results in optimal balance.

Variable Duty Factor

The variable duty factor is derived by considering a robot undergoing unidirectional motion. On average, the relative distance travelled between a leg and the body of the robot during its support phase should be equal and opposite to the relative distance traveled between the leg and body while the leg is in its transfer phase, as seen in Figure 3.5.



(a) Position of leg at beginning and end of support phase.

(b) Position of leg at beginning and end of transfer phase.

Figure 3.5: Equivalent relative distance traveled in support and transfer phases.

Considering an average relative velocity of the foot in its support phase to be v_{av} and an average relative velocity of the foot in its transfer phase to be u_{av}, the relative distances

traveled can be equated using the duty factor and kinematic period with

$$|v_{av}|\,(\tau\beta) = |u_{av}|\,(\tau\,[1-\beta])\,. \tag{3.7}$$

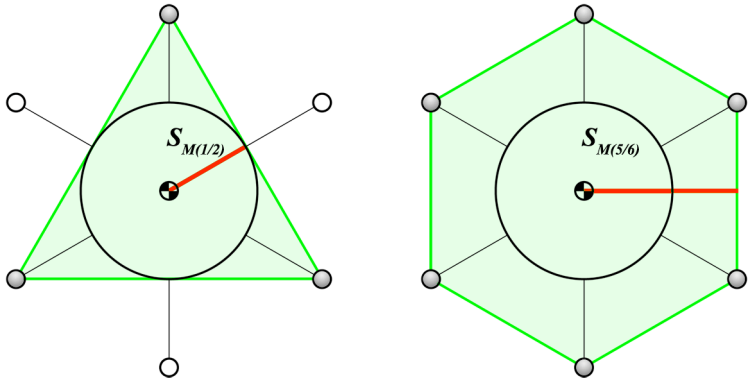
Rearranging Equation (3.7) for the duty factor yields

$$\beta = \frac{|u_{av}|}{|v_{av}| + |u_{av}|}. \tag{3.8}$$

Equation (3.8) represents the trade-off between the balance and top speed achieved by the robot. Song and Waldron indicate that increasing the duty factor increases balance [41]. In the literature, the degree to which the robot is balanced is referred to as the stability margin [41]. The stability margin is measured as the normalized minimum distance between the centre of mass (COM) of the robot and the polygon created by all of the support legs [41]. This is a static measure. Figures 3.6(a) and 3.6(b) visually demonstrate the relationship between an increased stability margin or balance, and an increased duty factor. In Figure 3.6(a), the duty factor of the robot is $\beta = \frac{1}{2}$. Accordingly, three legs are in their support phase (shaded feet), and three legs are in their transfer phase. A support triangle is created between the feet of legs in their support phase, and the resulting stability margin S_M is indicated. In contrast, Figure 3.6(b) depicts the robot with a duty factor $\beta > \frac{5}{6}$. With $\beta > \frac{5}{6}$ there are periods in the gait cycle in which all six legs are in their support phase. The support hexagon created by the robot, in this case, clearly permits a larger stability margin, leading to the conclusion that a larger duty factor can increase the balance of the robot. A thorough proof of this relationship is available in [41].

Since balance and consequently duty factor are to be maximized, consider the variables composing Equation (3.8). $|u_{av}|$ is free to vary in any manner because there are no velocity constraints imposed by the inputs, $\mathcal{I}(t)$, on an airborne foot. It is constrained however by the physical characteristics of the propulsion system of the robot. In the case of Hexplorer, a transfer leg can only travel as fast as the DC motors will allow. If this maximum speed is defined as U_{max}, then β is maximized by setting $|u_{av}| = U_{max}$. $|v_{av}|$, on the other hand, is constrained by the inputs $\mathcal{I}(t)$ and a kinematic equation involving a rotating reference frame. Ginsberg [14] provides Equation (3.9) that measures the velocity of a point P from a translating and rotating reference frame with origin O,

$$\mathbf{v}_P = \mathbf{v}_O + \boldsymbol{\Omega} \times \mathbf{r}_{P/O} + (\mathbf{v}_P)_{xyz}\,, \tag{3.9}$$



(a) An example of a stability margin for a wave gait with $\beta = \frac{1}{2}$.

(b) An example of a stability margin for a wave gait with $\beta > \frac{5}{6}$.

Figure 3.6: Improved balance of the robot due to larger duty factor β. Support feet are shaded.

where $\mathbf{v}_P$ is the velocity of point P, $\mathbf{v}_O$ is the velocity of point O, $\mathbf{\Omega}$ is the rotation of the frame with its origin at point O, $\mathbf{r}_{P/O}$ is the displacement between points P and O, and $(\mathbf{v}_P)_{xyz}$ is the relative velocity of point P with respect to the moving reference frame. Applying Equation (3.9) to the values at hand results in

$$\mathbf{v}_{foot} = \mathbf{v}_{body} + \mathbf{\Omega} \times \mathbf{r}_{foot/body} + \mathbf{v}_{av}. \tag{3.10}$$

In this equation, $\mathbf{v}_{foot}$ is the velocity of the foot, but, when the leg is in its support phase, the foot is in contact with the ground and as such has zero velocity. $\mathbf{v}_{body}$ is the velocity of the body measured from a translating and rotating frame. $v_x(t)$ and $v_z(t)$, specified in $\mathcal{I}(t)$, define the velocity of the body of the robot. $\mathbf{\Omega}$ is the rotation of the body frame, and based on the constraints of no pitch nor roll, $\mathbf{\Omega}$ results in the yaw rate of the robot $\omega_y(t)$. $\mathbf{r}_{foot/body}$ is simply the displacement between the body and foot, and $\mathbf{v}_{av}$ represents the relative velocity between the body and foot. Substituting these values into Equation (3.10), and considering maximum magnitudes, the following equation results:

$$|v_{av}| = \sqrt{v_x(t)^2 + v_z(t)^2} + |\omega_y(t)| r_{max}. \tag{3.11}$$

In this equation, r_{max} refers to the maximum horizontal distance between the foot and the centre axis of the robot [52, 53].

Combining the equations, inputs, and physical constraints of the robot above, an instantaneous duty factor, $\beta(t)$, can be computed as

$$\beta(t) = \frac{U_{max}}{\sqrt{v_x(t)^2 + v_z(t)^2} + |\omega_y(t)| r_{max} + U_{max}}. \tag{3.12}$$

Variable Crab Angle

Once a variable crab angle, $\alpha(t)$, and duty factor, $\beta(t)$, have been defined, the sequencing of individual legs can be examined. In a conventional forward wave gait, these parameters are fixed, as is the leg sequencing. Yoneda uses linear interpolation between several of these fixed parameter gaits to achieve smooth variable leg sequencing. Figure 3.7 shows a gait diagram depicting leg sequences for conventional wave gaits with a variety of duty factors.

Recall that the relative phase of leg i, ψ_i, is the value of the kinematic phase when the foot touches down and begins its support phase. Since the crab angle and duty factor of the gait will affect the timing of the legs, the relative phase is a function of these variables, *i.e.*, $\psi_i(\alpha(t), \beta(t)) = \text{function}(\alpha(t), \beta(t))$.

A continuous function for a dynamic relative phase of leg i, $\psi_i(\alpha(t), \beta(t))$, must be determined [52, 53]. It is important to remember that the resulting function, $\psi_i(\alpha(t), \beta(t))$, should be continuous. Continuity is important to ensure a smooth transition between support and transfer phases. If the function is discontinuous, a slight change in crab angle could require that a fast moving transfer leg, high above the ground, instantly becomes a very slow moving support leg that is in contact with the ground. Kinematically, this sudden change is impossible because time is required to dissipate kinetic energy from and transfer kinetic energy to the joints of a leg.

Figure 3.8 is a plot of relative phases, for a conventional wave gait, for leg 1 with $\alpha = \frac{\pi}{6}$ and β as indicated [52, 53]. The values used in this plot are taken directly from Figure 3.7. Based on the figure, a continuous linear relationship between β and ψ_1 emerges as

$$\psi_1\left(\frac{\pi}{6}, \beta(t)\right) = m\beta(t) + b. \tag{3.13}$$

Substituting in the appropriate values, and generalizing it to all legs, Yoneda arrives at

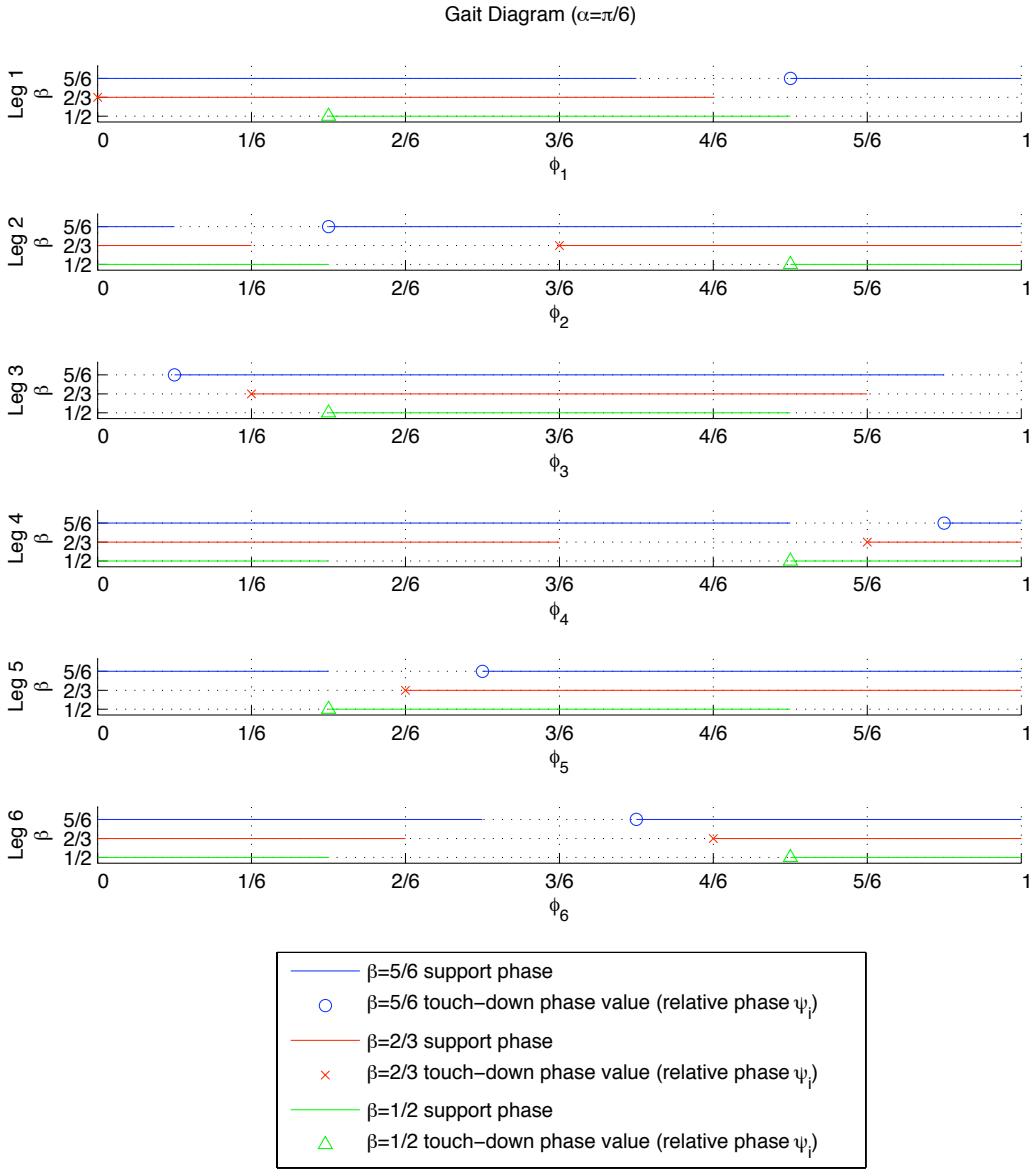


Figure 3.7: Leg sequences for a forward wave gait with $\alpha = \frac{\pi}{6}$.

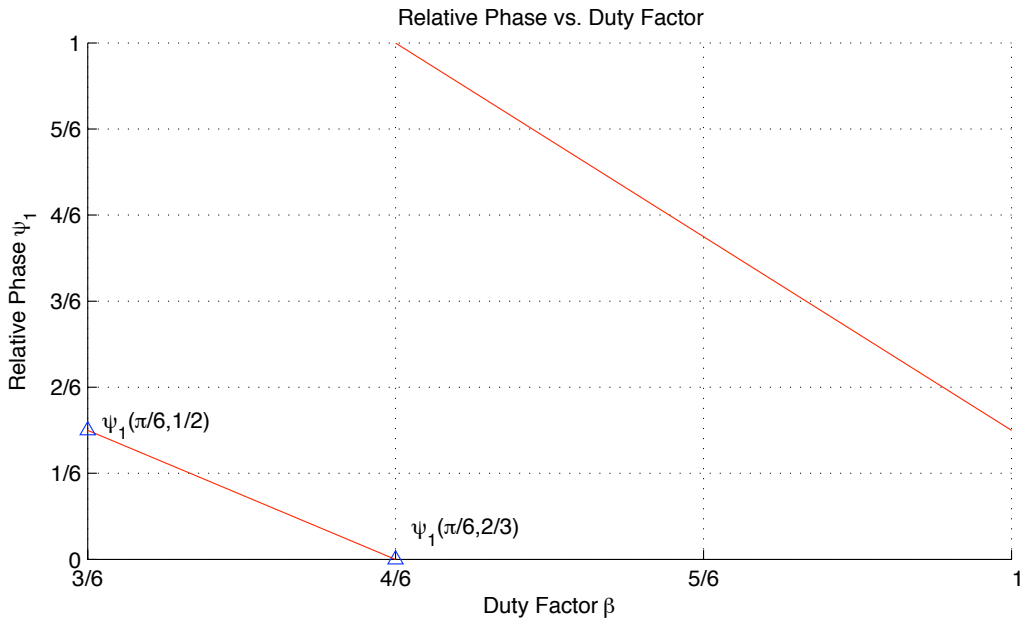


Figure 3.8: Relative phase of leg 1 ψ_1 vs. variable duty factor β with $\alpha = \frac{\pi}{6}$.

$$\begin{aligned} \psi_i\left(\frac{\pi}{6}, \beta(t)\right) &= \left(6\left[\psi_i\left(\frac{\pi}{6}, \frac{2}{3}\right) - \psi_i\left(\frac{\pi}{6}, \frac{1}{2}\right)\right]\beta(t)\right. \\ &\quad \left. - \left[3\psi_i\left(\frac{\pi}{6}, \frac{2}{3}\right) - 4\psi_i\left(\frac{\pi}{6}, \frac{1}{2}\right)\right]\right)_{mod1}. \end{aligned} \tag{3.14}$$

Actually, these results are not surprising as Equation (3.13) is similar to Equation (3.3), which was shown to define a conventional forward wave gait.

Equation (3.14) represents the relative phases of the legs for any time-varying β with a crab angle of $\alpha = \frac{\pi}{6}$. In order to have the relative phase vary with the crab angle as well, the values $\psi_i\left(\frac{\pi}{6}, \frac{2}{3}\right)$ and $\psi_i\left(\frac{\pi}{6}, \frac{1}{2}\right)$ in Equation (3.14), need to be replaced with functions $\psi_i\left(\alpha(t), \frac{2}{3}\right)$ and $\psi_i\left(\alpha(t), \frac{1}{2}\right)$, respectively. These functions must satisfy the conditions for either a conventional forward wave gait or a forward wave gait.

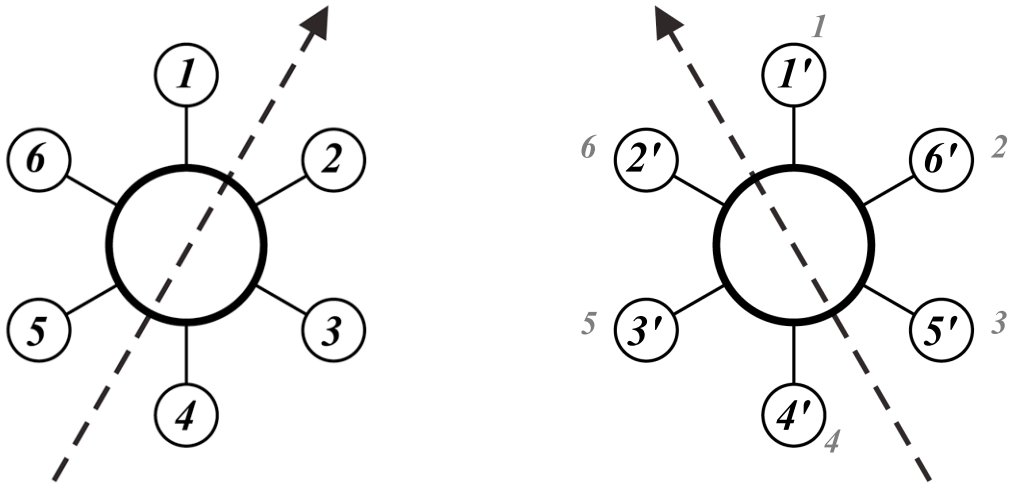
In [53], Yoneda produces a set of relative phases, and using the conditions from the previous section, proves that the set of relative phases can be classified as a forward wave gait. Instead of simply re-proving Yoneda's results, it is instructive to infer how these sets

of relative phases were selected or developed. Two sets of equations will be developed with one set representing a variable crab angle with a duty factor of $\beta = \frac{1}{2}$ and the other set representing a variable crab angle with a duty factor of $\beta = \frac{2}{3}$. In both cases, the procedure to generate the equations will be similar. Using Equation (3.3) and the conditions outlined in the previous section, relative phases for specific crab angles, at the given duty factor, will be determined. These discrete values will be connected using linear interpolation to yield continuous functions, $\psi_i\left(\alpha(t), \frac{2}{3}\right)$ and $\psi_i\left(\alpha(t), \frac{1}{2}\right)$, for the range $\alpha(t) \in [0, 2\pi)$.

First, consider the tripod gait, $\beta = \frac{1}{2}$ with $\alpha = \frac{\pi}{6}$, depicted in Figure 3.9(a). Equation (3.3) and the conditions defining a forward wave gait resolve to the following relative phase values:

$$\begin{array}{rclrcl} \psi_1\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & 0 & \psi_2\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & \frac{1}{2} \\ \psi_6\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & \frac{1}{2} & \psi_3\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & 0 \\ \psi_5\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & 0 & \psi_4\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & \frac{1}{2}. \end{array}$$

Next consider the same tripod gait, with $\alpha = \frac{11\pi}{6}$, depicted in Figure 3.9(b) ($\alpha = \frac{\pi}{6}$



(a) $\beta = \frac{1}{2}$ and $\alpha = \frac{\pi}{6}$ (b) $\beta = \frac{1}{2}$ and $\alpha = \frac{11\pi}{6}$

Figure 3.9: Leg sequencing with $\beta = \frac{1}{2}$ and specific crab angles.

leg numbers are shaded and located next to the feet). For $\alpha = \frac{\pi}{6}$ the two groups were $< 5, 6, 1 >$ and $< 4, 3, 2 >$. By effectively renumbering the feet, the same groups, and ordering within the groups, for $\alpha = \frac{\pi}{6}$ can be achieved for $\alpha = \frac{11\pi}{6}$. The renumbered legs are shown in Figure 3.9(b), with the new numbers located inside the feet and designated with a prime, $'$. By equating the relative phases for group $< 5, 6, 1 >$ where $\alpha = \frac{\pi}{6}$, and group $< 5', 6', 1' >$ where $\alpha = \frac{11\pi}{6}$, the results below are obtained:

$$\begin{array}{rclcrclcl}
\psi_1\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_{1'}\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_1\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & 0 \\
\psi_2\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_{6'}\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_6\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & \frac{1}{2} \\
\psi_3\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_{5'}\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_5\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & 0
\end{array}$$

$$\begin{array}{rclcrclcl}
\psi_6\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_{2'}\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_2\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & \frac{1}{2} \\
\psi_5\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_{3'}\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_3\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & 0 \\
\psi_4\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_{4'}\left(\frac{11\pi}{6}, \frac{1}{2}\right) & = & \psi_4\left(\frac{\pi}{6}, \frac{1}{2}\right) & = & \frac{1}{2}.
\end{array}$$

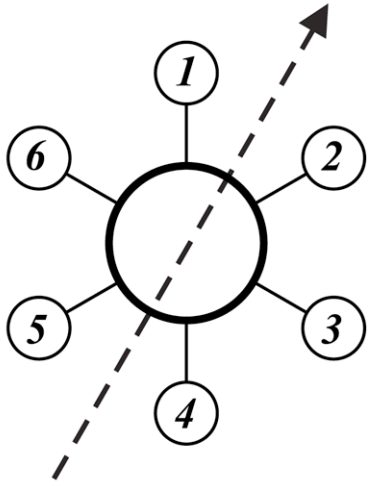
If this procedure is repeated for $\alpha \in \frac{(2i-1)\pi}{6},\ i = 1 \ldots 6$, the relative phases offset by $\frac{1}{4}$ as per the discrete values adapted from Yoneda [52, 53] summarized in Table 3.2, and a line interpolated between these relative phases, Yoneda's results are obtained,

$$\psi_i\left(\alpha(t), \frac{1}{2}\right) = \begin{cases} \frac{1}{4} & \text{if } i = 1, 3, 5 \\ \\ \frac{3}{4} & \text{else} \end{cases} . \tag{3.15}$$

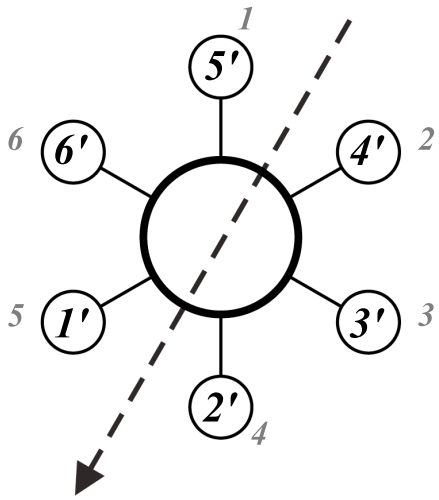
This set of relative phases satisfies conditions (1) and (2) and therefore generates a forward wave gait.

The function $\psi_i\left(\alpha(t), \frac{2}{3}\right)$ can be formulated in a similar manner. In the interest of brevity, only $\psi_1\left(\alpha(t), \frac{2}{3}\right)$ is fully derived. An initial set of relative phases can be determined using Equation (3.3). The results, generating a forward wave gait, correspond to Figure 3.10(a) and are listed below:

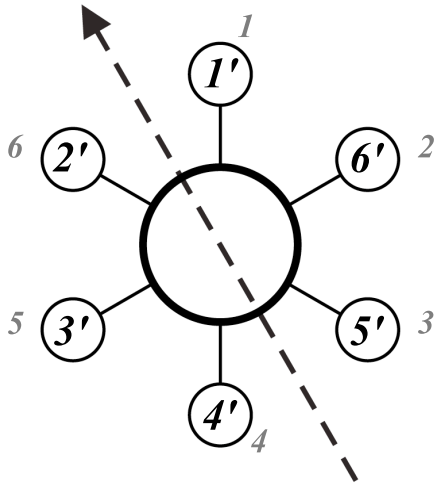
$$\begin{array}{rcl}
\psi_1\left(\frac{\pi}{6}, \frac{2}{3}\right) & = & 0 \\
\psi_3\left(\frac{\pi}{6}, \frac{2}{3}\right) & = & \frac{1}{6} \\
\psi_5\left(\frac{\pi}{6}, \frac{2}{3}\right) & = & \frac{2}{6}.
\end{array}$$



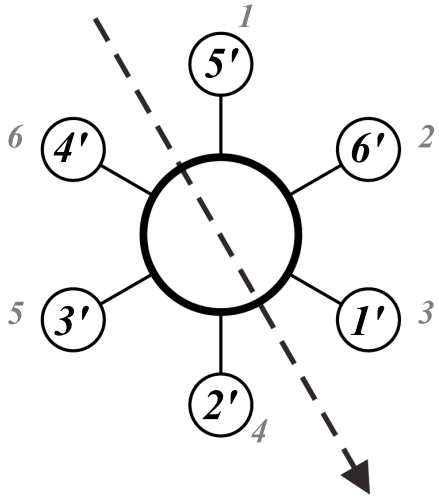
(a) $\beta = \frac{2}{3}$ and $\alpha = \frac{\pi}{6}$



(b) $\beta = \frac{2}{3}$ and $\alpha = \frac{7\pi}{6}$



(c) $\beta = \frac{2}{3}$ and $\alpha = \frac{11\pi}{6}$



(d) $\beta = \frac{2}{3}$ and $\alpha = \frac{5\pi}{6}$

Figure 3.10: Leg sequencing with $\beta = \frac{2}{3}$ and specific crab angles.

Table 3.2: Relative Leg Phases (ψ_i) at Discrete Duty Factors (β) and Crab Angles (α)

Relative Phase $\psi_i(\alpha, \beta)$	Duty Factor (β)											
	1/2						2/3					
Crab Angle (α)	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	ψ_6	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	ψ_6
$\pi/6$	1/4	3/4	1/4	3/4	1/4	3/4	0	3/6	1/6	5/6	2/6	4/6
$3\pi/6$	1/4	3/4	1/4	3/4	1/4	3/4	1/6	3/6	0	4/6	2/6	5/6
$5\pi/6$	1/4	3/4	1/4	3/4	1/4	3/4	2/6	4/6	0	3/6	1/6	5/6
$7\pi/6$	1/4	3/4	1/4	3/4	1/4	3/4	2/6	5/6	1/6	3/6	0	4/6
$9\pi/6$	1/4	3/4	1/4	3/4	1/4	3/4	1/6	5/6	2/6	4/6	0	3/6
$11\pi/6$	1/4	3/4	1/4	3/4	1/4	3/4	0	4/6	2/6	5/6	1/6	3/6

In Figure 3.10(b), the legs have been renumbered to equate relative phases between crab angles of $\alpha = \frac{\pi}{6}$ and $\alpha = \frac{7\pi}{6}$. The relative phase for leg 1 is given below:

$$\psi_1\left(\tfrac{7\pi}{6}, \tfrac{2}{3}\right) = \psi_{5'}\left(\tfrac{7\pi}{6}, \tfrac{2}{3}\right) = \psi_5\left(\tfrac{\pi}{6}, \tfrac{2}{3}\right) = \tfrac{2}{6}.$$

Following this same procedure for $\alpha = \frac{11\pi}{6}$ and $\frac{5\pi}{6}$, shown in Figures 3.10(c) and 3.10(d), respectively, the leg 1 relative phases listed below are obtained:

$$\begin{aligned} \psi_1\left(\tfrac{11\pi}{6}, \tfrac{2}{3}\right) &= \psi_{1'}\left(\tfrac{11\pi}{6}, \tfrac{2}{3}\right) = \psi_1\left(\tfrac{\pi}{6}, \tfrac{2}{3}\right) = 0 \\ \psi_1\left(\tfrac{5\pi}{6}, \tfrac{2}{3}\right) &= \psi_{5'}\left(\tfrac{5\pi}{6}, \tfrac{2}{3}\right) = \psi_5\left(\tfrac{\pi}{6}, \tfrac{2}{3}\right) = \tfrac{2}{6}. \end{aligned}$$

Plotting and linearly interpolating between these values, as shown in Figure 3.11, then yields a continuous equation for the relative phases of leg 1 with a variable crab angle and

a duty factor of $\beta = \frac{2}{3}$. By extending this procedure to the other legs, Yoneda [53] provides

$$\psi_i\left(\alpha(t), \frac{2}{3}\right) = \begin{cases} \begin{cases} \frac{\alpha'}{2\pi} & \text{if} \quad 0 \le \alpha' \le \frac{2\pi}{3} \\ \frac{1}{3} & \frac{2\pi}{3} \le \alpha' \le \pi \\ -\frac{\alpha'}{2\pi} + \frac{5}{6} & \pi \le \alpha' \le \frac{5\pi}{3} \\ 0 & \frac{5\pi}{3} \le \alpha' \le 2\pi \end{cases} & , \quad \text{if } i = 1, 3, 5 \\ \begin{cases} \frac{\alpha'}{2\pi} + \frac{1}{2} & \text{if} \quad 0 \le \alpha' \le \frac{2\pi}{3} \\ \frac{5}{6} & \frac{2\pi}{3} \le \alpha' \le \pi \\ -\frac{\alpha'}{2\pi} + \frac{1}{3} & \pi \le \alpha' \le \frac{5\pi}{3} \\ \frac{1}{2} & \frac{5\pi}{3} \le \alpha' \le 2\pi \end{cases} & , \quad \text{if } i = 2, 4, 6 \end{cases} \tag{3.16}$$

where

$$\alpha' = \left[\alpha - \frac{(2i-1)\pi}{6}\right]_{mod 2\pi}, \tag{3.17}$$

which by inspection satisfies all of the conditions necessary to define a forward wave gait.

The entire purpose of examining a variable crab angle was to develop a set of equations to describe relative leg phases for variable crab angles and duty factors. Incorporating Equations (3.15) and (3.16) into Equation (3.14) provides the desired result in the form of

$$\begin{aligned} \psi_i\left(\alpha(t), \beta(t)\right) = & \left(6\left[\psi_i\left(\alpha(t), \frac{2}{3}\right)_{mod1} - \psi_i\left(\alpha(t), \frac{1}{2}\right)_{mod1}\right]\beta(t)\right. \\ & \left. - \left[3\psi_i\left(\alpha(t), \frac{2}{3}\right)_{mod1} - 4\psi_i\left(\alpha(t), \frac{1}{2}\right)_{mod1}\right]\right)_{mod1}. \end{aligned} \tag{3.18}$$

3.1.5 Gait Planner

Equation (3.18) and Figure 3.7 form Yoneda's gait planner. Based on a set of inputs $\mathcal{I}(t)$ and the current state of the robot, the gait planner generates a set of touch-down timings to sequence the legs. This sequence has been carefully designed to produce a modified forward wave gait that is dynamically optimal first for balance and then for speed. In order to make use of these touch-down timings, they are converted to represent and quantify the states of a leg. Afterwards, based on the states of the legs, the foot motion planner selects an appropriate course of action.

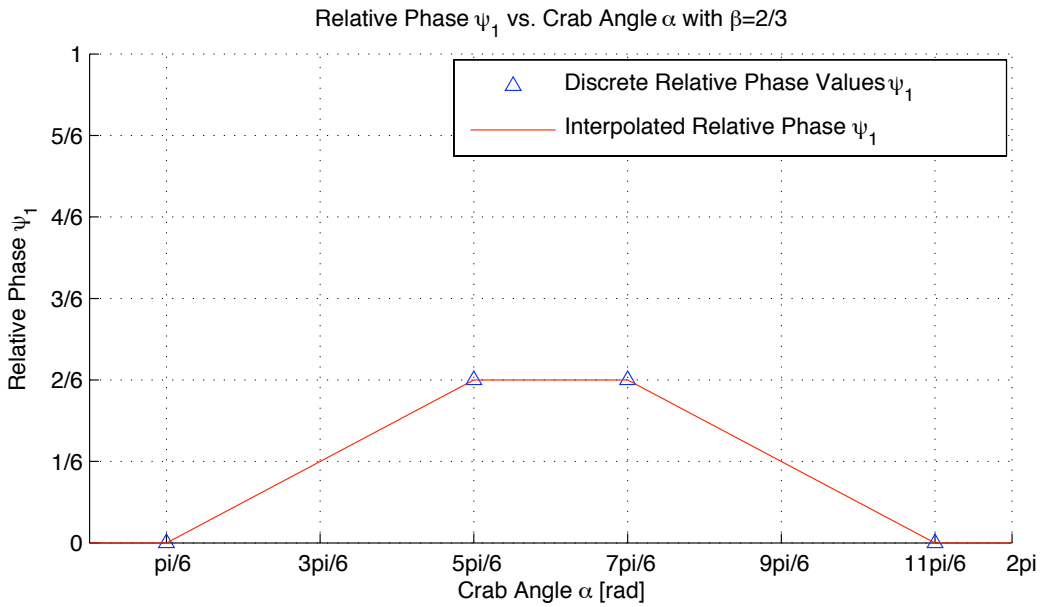


Figure 3.11: Relative phase of leg 1 ψ_1 vs. variable crab angle α with $\beta = \frac{2}{3}$.

Leg and Body States

In this section, touch-down timings are combined with the overall state of the robot to generate a set of variables that determine and describe the state of a leg. Very simply, the state of leg i, ϕ_{Li}, refers to whether the leg is in its support or transfer phase as well as the completion of the current state. With this information, support legs can be driven such that the correct motion of the body of the robot is achieved, and transfer legs can determine whether they should aggressively raise their feet from the ground or prepare to become support legs, ready to gently place a foot on the ground.

The most important phase variable and state is the kinematic phase of the robot ϕ. Based on this value, supervisory inputs, and relative leg phases, all of the necessary gait information can be generated. As defined earlier, the kinematic phase of the robot represents the percent completion of a gait cycle. It essentially measures how far the robot has moved in the present cycle compared to the total movement the robot can achieve in a single cycle. The kinematic phase is calculated based on the instantaneous kinematic

period [28], τ, as

$$\phi(t) = \int_0^t \frac{1}{\tau(\eta)} d\eta, \tag{3.19}$$

where η is a dummy integration variable. The leg phase variable indicates the current state of the leg. It is calculated with

$$\phi_{Li} = \left[\phi(t) - \psi_i\left(\alpha(t), \beta(t)\right)\right] \mod 1, \tag{3.20}$$

and its values are interpreted as

$$\begin{array}{rcccl} 0 & \leq & \phi_{Li} & \leq & \beta \quad \text{being in the support phase and} \\ \beta & < & \phi_{Li} & \leq & 1 \qquad\quad \text{being in the transfer phase.} \end{array} \tag{3.21}$$

By normalizing the leg phase using β, degree of completion of the support phase is measured as a value between 0 and 1, as per the definition of support phase. The support phase of leg i is calculated as

$$\phi_{Si} = \frac{\phi_{Li}}{\beta}. \tag{3.22}$$

Similarly, normalizing the leg phase remaining after the support phase with the fraction of a cycle spent in the transfer phase, completion of the transfer phase is measured as a value between 0 and 1, as per the definition of transfer phase. The transfer phase of leg i is calculated as

$$\phi_{Ti} = \frac{1 - \phi_{Li}}{1 - \beta}. \tag{3.23}$$

The phase values, ϕ_{Li} and, ϕ_{Si} or ϕ_{Ti}, together, form the inputs to the foot motion planner.

3.1.6 Foot Motion Planning

Yoneda defines a foot motion planner which, depending on the state of the leg, prescribes the motion of a foot relative to the reference frame of the leg. Relatively simple in structure, the logic of the foot motion planner is depicted in Figure 3.12.

If a leg is in its support phase, then the motion of the foot relative to the reference frame of the leg must occur in such a way that the motion of the body of the robot matches the desired body motion. Goulet explains this constraint using Wilson's Law [17],

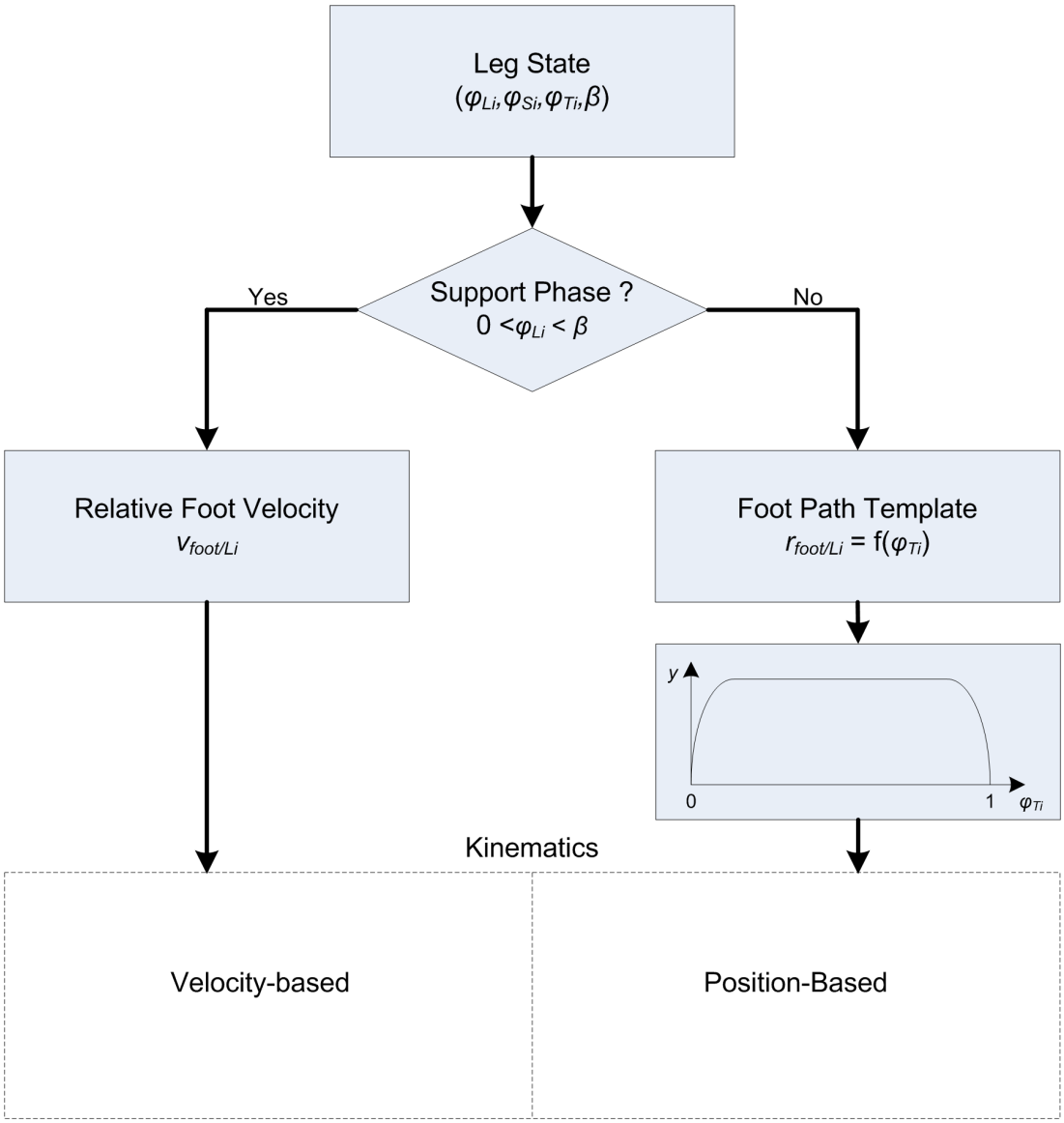


Figure 3.12: Outline of Yoneda's foot motion planner.

stating that the polygon formed by the feet acting as the based of support cannot deform. Using either of these explanations, the velocity of the feet with respect to the leg or body can be determined using kinematics, in addition to realizing that support feet are in contact with the ground, and therefore should have an absolute velocity of zero. By carefully selecting and defining reference frames, identical code can be implemented on all of the DSPs responsible for controlling Hexplorer. Eight reference frames are defined in Figure 3.13. The world reference frame $\{W\}$ is an inertial reference frame. The body

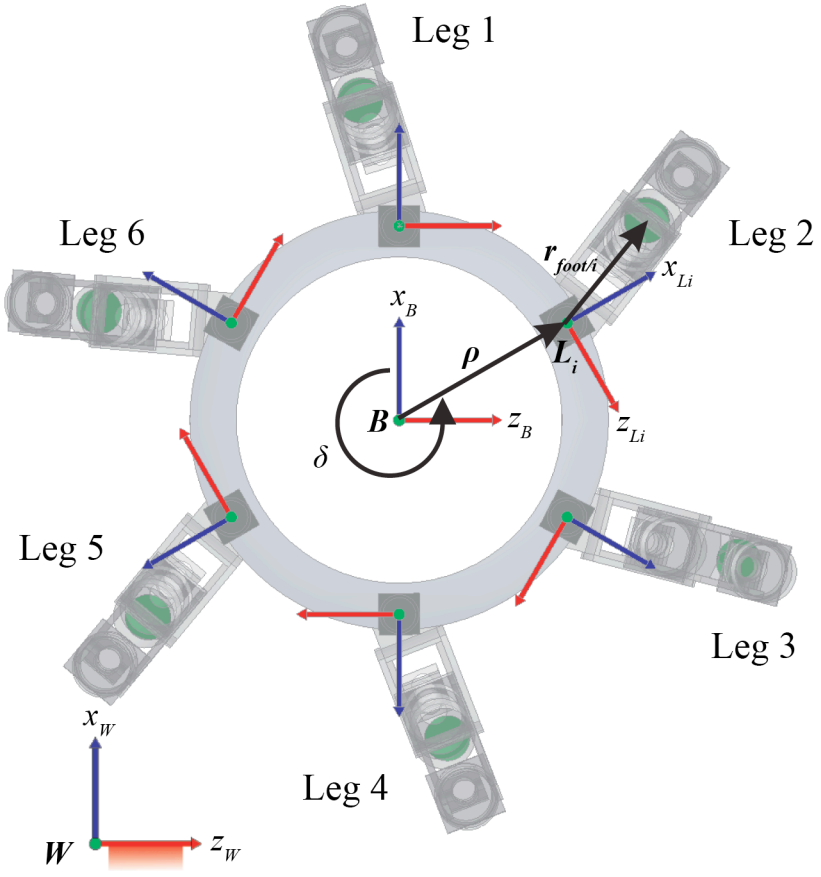


Figure 3.13: Cartesian reference frames.

reference frame $\{B\}$ is a reference frame attached to the body of the robot, with x_B aligned

with leg 1 as shown. The input velocities and angular velocities are defined in this frame. The remaining six frames are local leg frames. Attached to the body of the robot, these frames are oriented such that the leg workspaces all share identical coordinates in their local frames. The origins of these local leg frames are located at the hip joint connecting the legs to the body of Hexplorer and designated as $\{L_i\}$. The constant angle δ measures the rotation of the leg frames with respect to the body frame and the radius of the body is given by ρ. Using Equation (3.9), the equation governing the relative velocity of foot i with respect to a local leg frame i is

$$\mathbf{v}_{foot_i} = \mathbf{v}_{L_i} + \mathbf{\Omega} \times \mathbf{r}_{foot/L_i} + \mathbf{v}_{foot/L_i}. \tag{3.24}$$

In Equation (3.24), $\mathbf{v}_{foot_i}$ is the velocity of foot i which is zero, $\mathbf{\Omega}$ is the rotation of local frame $\{L_i\}$ which is the yaw rate ω_y, $\mathbf{r}_{foot/L_i}$ is the displacement of foot i with respect to the local leg frame, and $\mathbf{v}_{foot/L_i}$ is the relative velocity of foot i with respect to local leg frame $\{L_i\}$. The components of $\mathbf{r}_{foot/L_i}$ and $\mathbf{v}_{foot/L_i}$, respectively, as measured from the local frame $\{L_i\}$ resolve into

$$\mathbf{r}_{foot/L_i} = \left\{ \begin{array}{c} r_{x_{foot/L_i}} \\ r_{y_{foot/L_i}} \\ r_{z_{foot/L_i}} \end{array} \right\}_{L_i} \quad \text{and} \quad \mathbf{v}_{foot/L_i} = \left\{ \begin{array}{c} v_{x_{foot/L_i}} \\ v_{y_{foot/L_i}} \\ v_{z_{foot/L_i}} \end{array} \right\}_{L_i}. \tag{3.25}$$

Substituting the inputs, $\mathcal{I}(t)$, angular offset of the leg frame, δ, displacement of the leg frame from the body frame, ρ, and the column matrix components of Equation (3.25) into Equation (3.24), results in the set of equations:

$$\left\{ \begin{array}{c} v_{x_{foot/L_i}} \\ v_{y_{foot/L_i}} \\ v_{z_{foot/L_i}} \end{array} \right\}_{L_i} = \left\{ \begin{array}{c} v_z \sin\delta - v_x \cos\delta - \omega_y \left(r_{z_{foot/L_i}} \right) \\ 0 \\ -v_x \sin\delta - v_z \cos\delta + \omega_y \left(\rho + r_{x_{foot/L_i}} \right) \end{array} \right\}_{L_i}. \tag{3.26}$$

On the other hand, if the leg is in its transfer phase, it can move freely without affecting the motion of the robot. The only critical requirement is that the foot cannot be in contact with the ground. Yoneda uses a set of velocity templates that when integrated, define a path of travel for the foot. Hexplorer uses a simpler technique based on position control that will be fully discussed in Section 4.2.3. The freedom of motion of the foot while in

its transfer phase, introduces another problem. Where should the foot move within its workspace, such that it can maximize the distance traveled through its workspace during its next support phase? In more colloquial terms, where should the foot-hold be located so that the biggest step can be taken? If the distance traveled in the support phase (step size) is maximized, then the number of gait cycles (steps) required to travel a certain distance will be minimized. Yoneda selects a foot-hold located on a circle that has its centre located in the middle of the workspace of the leg. The position of the foot-hold on the circle is selected based on the crab angle. This technique is simple and computationally efficient. It also provides larger step sizes than simply returning the foot to the centre of the workspace. The implementation on Hexplorer, however, takes advantage of the existing functionality of the temporal kinematic margin. Instead of limiting the foot-hold position to occur on a circle of a pre-determined radius, the foot-hold can occur anywhere in the workspace. By reversing the direction of the inputs $\mathcal{I}(t)$ and solving for the temporal kinematic margin, from the centre of the workspace, a foot-hold near the workspace boundary is selected. Like Yoneda's technique, this one is dynamic, and new foot-hold locations are recalculated, until the foot is placed on the ground. But unlike Yoneda's technique, the one implemented on Hexplorer takes advantage of the existing temporal kinematic margin functionality, is based solely on local leg frame coordinates, and exploits the shape of the workspace more fully.

Two main ideas are important in the foot motion planner. If a leg is in its support phase then velocity is the parameter to control. Otherwise, if a leg is in its transfer phase, the position of the foot is critical.

Kinematic Phase and Temporal Kinematic Margin

The temporal kinematic margin possesses two advantages that are related to one another. The temporal kinematic margin is used to update the state of the kinematic phase, ϕ, by means of the kinematic period, τ. It is also able to adjust the kinematic period to dynamically shrink or expand the workspace of a leg, within its physical limits.

Lee and Orin [28] define the temporal kinematic margin as

$$t_{Si} = \frac{d}{\left|\mathbf{v}_{foot/L_i}\right|}, \tag{3.27}$$

where d is the distance between the current position of the foot and the workspace boundary

(real or effective) and $\mathbf{v}_{foot/body}$ is the relative velocity of the foot with respect to the body. It represents a first order approximation of the time until a support foot reaches its workspace boundary.

The temporal kinematic margin is converted into a kinematic period by considering the state of the support legs. First, the support period, which indicates the amount of time required to complete the support phase, must be determined. If the temporal kinematic margin represents the time remaining in the support phase, then dividing this by the completion of the support phase will provide the support period [28]:

$$\tau_{Si} = \frac{t_{Si}}{1 - \phi_{Si}}. \tag{3.28}$$

The support period can then be converted to the kinematic period by considering the fraction of a cycle the leg is in its support phase (*i.e.* duty factor). This kinematic period should ideally be equivalent for all legs; however, since it is a first order approximation, support legs closest to their workspace boundaries will provide the most accurate estimate of the kinematic period. Accuracy increases because as a leg approaches its workspace boundary, the linear approximation of time until impact better reflects the non-linear motion of the robot, including variable speeds, crab angles, and yaw rate. A kinematic period, τ_i, is calculated for each leg in a support phase with

$$\tau_i = \frac{\tau_{Si}}{\beta}. \tag{3.29}$$

The minimum value is selected to represent the kinematic period, and used to update the kinematic phase. This will increase the chance that a leg close to its workspace boundary will finish its support phase and begin its transfer phase because the kinematic phase is advanced more by a smaller kinematic period. The instantaneous estimate of the kinematic period is calculated,

$$\tau = \min_{\substack{i=1\ldots6,\\ \phi_{Li} \leq \beta}} (\tau_i), \tag{3.30}$$

which in turn is used to update the kinematic phase [28].

In Equation (3.27) the distance between the foot and the workspace boundary is measured by d. This distance may reflect the real workspace boundary, or a boundary within

the workspace itself. Lee and Orin [28] introduced this method as constrained working volume (CWV). It is particularly useful if the workspaces between legs on the robot overlap with one another. By artificially limiting the size of the workspace, a robot designed for a follow-the-leader gait can execute a wave gait without the concern of workspace interference. However, usefulness of the CWV method is limited on Hexplorer because the leg workspaces do not overlap.

3.2 Other Modifications and Enhancements

While the contribution of the variable duty factor wave gait represents a significant contribution to the research community, Yoneda does not fully address a few technical implementation issues in his team's papers. These issues involve the sign of the kinematic period, handling the temporal kinematic margin, unexpected contact events, and the rate of change of inputs.

Temporal Kinematic Margin

In Orin's original derivation of the kinematic phase of the robot, he takes into account the direction of travel [28] by changing the sign of the kinematic period. If Orin's robot traveled forward, the kinematic phase increased. If it traveled backwards, the kinematic phase decreased. The benefit of calculating kinematic phase in this respect is that if the robot moves forwards for a half cycle and then backwards for a half cycle, it does not necessarily require new foot-holds as the support legs may not change. If, on the other hand, the kinematic phase only increases, and the robot moves forwards for a half cycle and then backwards for a half cycle, the kinematic phase would indicate that the robot has completed a full cycle. This implies that the robot would have changed support legs even though a tiny distance was traveled and the total displacement of the robot is zero. Thus, using an unsigned kinematic period to calculate kinematic phase can, in some instances, cause unnecessary changes in support legs and consequently foot-holds. Yoneda likely excluded the sign of the kinematic phase because of the omnidirectional nature of his algorithm and the ability of the robot to yaw. Realistically, if selecting new foot-holds is a delicate activity because of inhospitable terrain, then the forward wave gait itself is likely

a bad choice of gait for the terrain.

The temporal kinematic margin is an estimate that affords the robot the ability to effectively shrink the workspace of the leg. However, Yoneda does not address the proper procedure to deal with a leg that has reached its physical workspace limit prematurely. If a support leg reaches its workspace boundary prematurely, its foot cannot move in such a way to prevent the base of support polygon to deform. In simulation of Hexplorer, this is handled by increasing the kinematic phase so that the delinquent leg becomes a transfer leg. Although this technique introduces a slight discontinuity into the kinematic phase, the temporal kinematic margin should be reasonably accurate, so that any legs instantaneously changed from support to transfer or vice versa, should not do so very abruptly.

Ground Contact

In [53], Yoneda briefly mentions that phase changes depend on a foot sensor, but he does not elaborate. Based on experience with Hexplorer, two important scenarios can occur based on unexpected foot sensor readings. It is possible for a foot sensor to indicate that a leg has lost contact with the ground, during the beginning or middle of its support phase. In this case, heuristics are likely to solve the problem best because the robot has effectively broken its gait and needs some method to recover. The robot could activate an E-stop, halting all movement before executing a search algorithm to find another foot-hold location. This scenario only occurs on uneven or inclined terrain. It was assumed at the outset however, that the robot would only be exposed to smooth level terrain, making this problem beyond the scope of the research for this thesis. It is something to be aware of, should Hexplorer become fully autonomous. The second scenario involves having a foot make contact with the ground earlier than predicted. In general, this is likely an indication of an obstacle or uneven terrain. However, due to the smooth terrain assumption, obstacle avoidance is not discussed. Therefore, the only case where the foot acquires contact with the ground prematurely, is at the end of the transfer phase. This situation can be handled by advancing the kinematic phase in order to reclassify the leg as a support one. This does create a discontinuity in the kinematic phase, but as in the case of the inaccurate temporal kinematic margin, the discontinuity is small and is likely dissipated by the compliance of the joints.

Input Rate of Change

A larger issue, involving discontinuities in the relative leg phases, can be caused by large instantaneous changes in crab angle or yaw rate. Consider an example, where a fast moving transfer leg is supposed to instantaneously become a support leg. Let the body of the robot move at a fixed speed with a duty factor of $\beta = \frac{2}{3}$. At $t = 0$ let the kinematic phase be $\phi(0) = \frac{3}{4}$ and the crab angle be $\alpha(0) = \frac{\pi}{6}$. Accordingly, the phase of leg 1 is $\phi_{L1}(0) = \left(\frac{3}{4} - 0\right)_{mod1} = \frac{3}{4}$, meaning the leg is in the middle of its transfer phase and moving quickly high above the ground. At the next instant in time, $t = 0^+$, $\phi(0^+) = \frac{3}{4}^+$, and the crab angle changes to $\alpha(0^+) = \frac{5\pi}{6}$. Accordingly, the phase of leg 1 is $\phi_{L1}(0^+) = \left(\frac{3}{4}^+ - \frac{1}{3}\right)_{mod1} = \frac{5}{12}$, meaning the leg is now supposed to be nearing the end of its support phase and moving fairly slowly on the ground in the opposite direction. This type of instantaneous change cannot occur because the legs have mass and are subject to the laws of physics. If such an input were applied to the robot, severe synchronization problems between legs would likely result. Furthermore, this de-synchronization could cause the robot to lose its balance and tip over.

To counteract the possibility of such an instantaneous change in crab angle or yaw rate, a second-order Butterworth filter is applied to the inputs, $\mathcal{I}(t)$. The cut-off frequency of the filter is tuned based on the physical properties of the motors. Assuming that the leg can only travel at $U_{max} = 5cm/s$ and its foot travels $5cm$ above the ground in the transfer phase, it should take at least 1 second to change its crab angle from $\frac{\pi}{6}$ to $\frac{5\pi}{6}$ and consequently switch phases. In a second-order system, the rise-time is defined as the time it takes for a system to reach the vicinity of its new set-point based on a step input [11]:

$$t_r \approx \frac{1.8}{\omega_{\text{cut-off}}}. \tag{3.31}$$

Using a rise-time of $t_r = \frac{1}{\frac{5\pi}{6} - \frac{\pi}{6}}$ seconds for a unit step input, the cut-off frequency of the Butterworth filter can be calculated. Thus, the digital implementation is based on a 10Hz sampling rate with a cut-off frequency of 0.6Hz.

3.3 Summary

In this chapter, the concept and details of a variable duty factor wave gait were introduced. Based on a desired heading, speed, and yaw rate, the robot is able to sequence its legs properly. When supporting the robot, legs must move at an appropriate velocity to propel the robot at the desired heading, speed, and yaw rate. Supporting legs are also used to update the overall state of the robot. During the transfer phase, the motion of the leg can be arbitrary, provided the foot is not in contact with the ground. The next stage involves realizing the motions of the feet and legs by actuating the leg joints, and is accomplished with a kinematic analysis of the legs of the robot.

Chapter 4

Kinematic Model and Simulation

In Yoneda's algorithm [52, 53], input variables $\mathcal{I}(t)$, processed by gait and foot motion planners, produce values that determine the desired motion of the legs of the robot. This motion is generated by moving the joints of the leg, but the leg joints must move in such a way that the overall motion of the body, specified in a Cartesian coordinate system, is achieved. The relationship between these two descriptions of motion is determined by analysing the kinematic configuration of the leg. This chapter provides an in-depth mathematical description of the mechanical configuration of Hexplorer. With this mathematical description defined, a full kinematic simulation of Yoneda's algorithm on Hexplorer is presented. The simulation includes a number of details, such as the workspace of a leg, results of using a higher-order approximation to estimate the temporal kinematic margin, and a comparison between two horizontal terrain foot-hold selections.

4.1 Kinematic Model

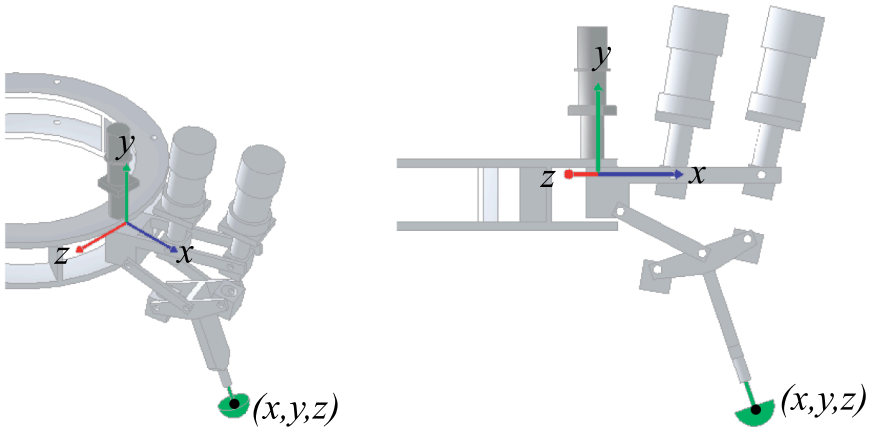
Before the kinematic model of the robot is discussed, the four coordinate systems used in conjunction with the kinematics are presented.

The Cartesian coordinates of a foot are based on a reference frame attached to the *body* of the robot, not the *leg*, and located as shown in Figure 4.1. The Cartesian coordinates

of foot i are:

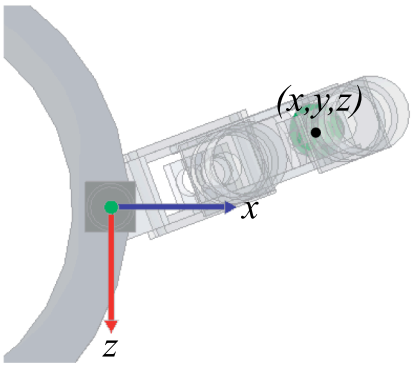
$$\mathbf{x}_{i_{Cartesian}} = \{x_i, y_i, z_i\}. \tag{4.1}$$

The joint coordinates of a leg are the two lead screw lengths and the rotation of the leg



(a) Isometric view.

(b) Planar view.

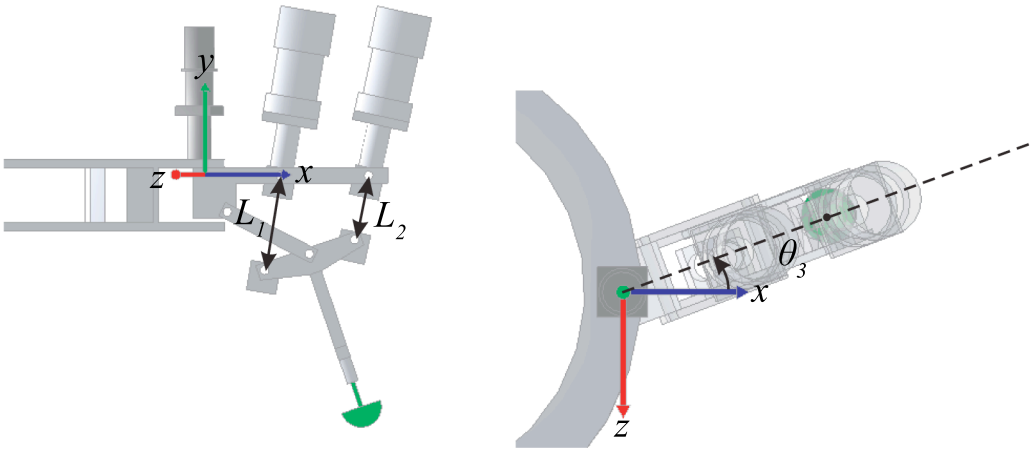


(c) Top view.

Figure 4.1: Cartesian coordinate system of a leg.

about the hip as shown in Figure 4.2. The joint coordinates of leg i are:

$$\mathbf{x}_{i_{Joint}} = \{L_{1_i}, L_{2_i}, \theta_{3_i}\}. \tag{4.2}$$



(a) Planar view. (b) Top view.

Figure 4.2: Joint coordinate system of a leg.

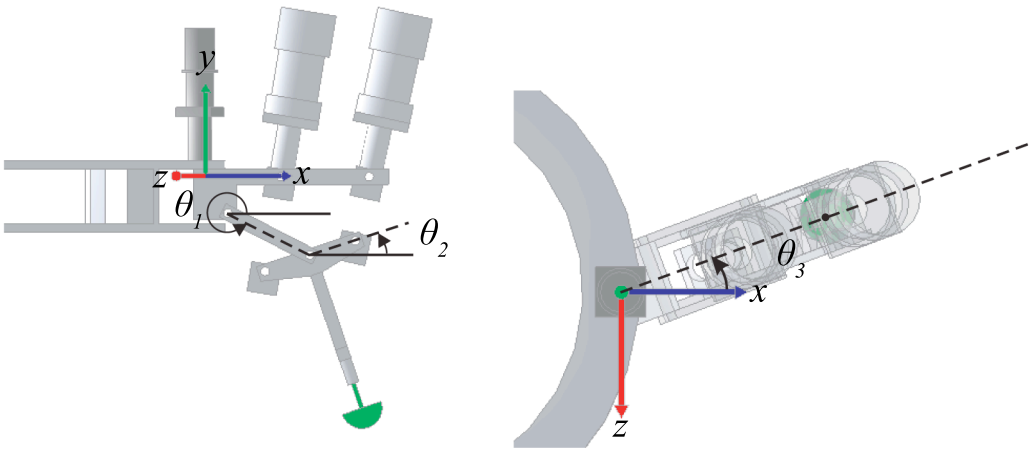
The modeling coordinates of a leg are based on rotations of the two serially connected links, depicted in Figure 4.3, relative to the body-fixed xyz frame. The modeling coordinates of leg i are:

$$\mathbf{x}_{i_{Model}} = \{\theta_{1_i}, \theta_{2_i}, \theta_{3_i}\}. \tag{4.3}$$

Natural coordinates of a planar multibody system describe the configuration of the system using two Cartesian points to describe each joint [12]. In the plane of the RPR mechanism, one point at each end of each lead screw defines the natural coordinates of the leg, shown as points c, e, f, and h in Figure 4.4. Since the positions of points f and h are fixed within the plane, only variables representing points c and e are required. The scalar coordinate components of points c and e are:

$$\mathbf{x}_{i_{Nat}} = \{v_{c_i}, v_{e_i}, w_{c_i}, w_{e_i}\}. \tag{4.4}$$

The kinematic model of the robot has two components, forward kinematics and inverse kinematics. A forward kinematic analysis is used to determine the Cartesian position of an end effector of a mechanism based on the joint positions [7]. In this instance, the



(a) Planar view. (b) Top view.

Figure 4.3: Modeling coordinate system of a leg.

mechanism is a leg attached to Hexplorer, while the end effector is the foot of the leg. The forward kinematic analysis is used for a number of purposes, especially during the support phase of a leg, including estimating the temporal kinematic margin and calculating joint velocities. Inverse kinematics, on the other hand, are used for the opposite task. An inverse kinematic analysis converts the Cartesian position of a foot, belonging to Hexplorer, into joint positions of the leg to which the foot is attached [7]. The inverse kinematics are useful during the transfer phase of a leg, where the desired position of the foot is given in Cartesian coordinates, but where control of the foot is accomplished at the joint level.

Two additional sets of coordinate systems are used to relate the Cartesian and joint coordinates. Referred to as modelling and natural coordinates in this thesis, these coordinate systems are required because of difficulties arising from determining the forward kinematics of the RPR mechanism. Figure 4.1 describes the relationships between these four sets of coordinates.

4.1.1 Forward Kinematics

The legs of Hexplorer were designed to allow the robot to be passively supported and sturdy. These goals were indeed met; however, they were met at the expense of an analytical

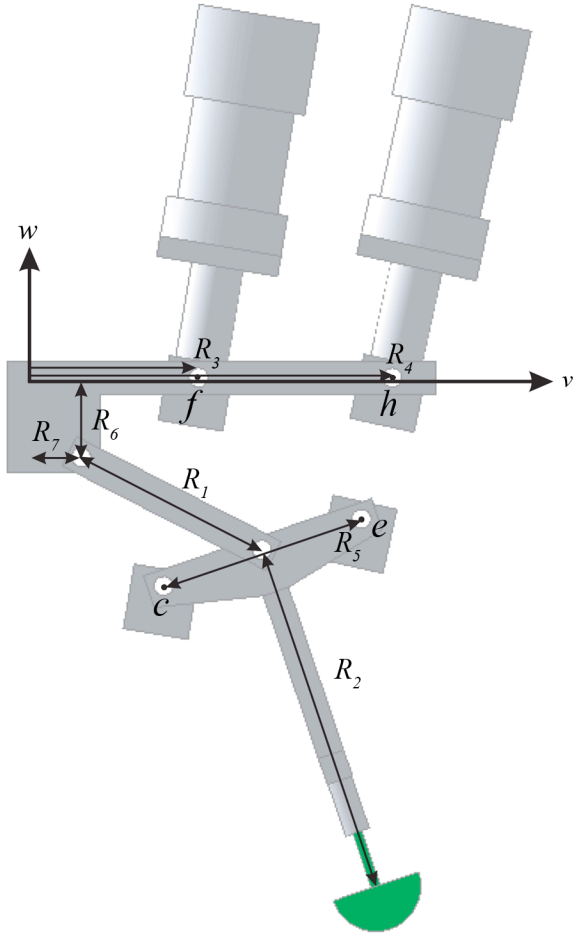


Figure 4.4: Natural coordinate system of a leg.

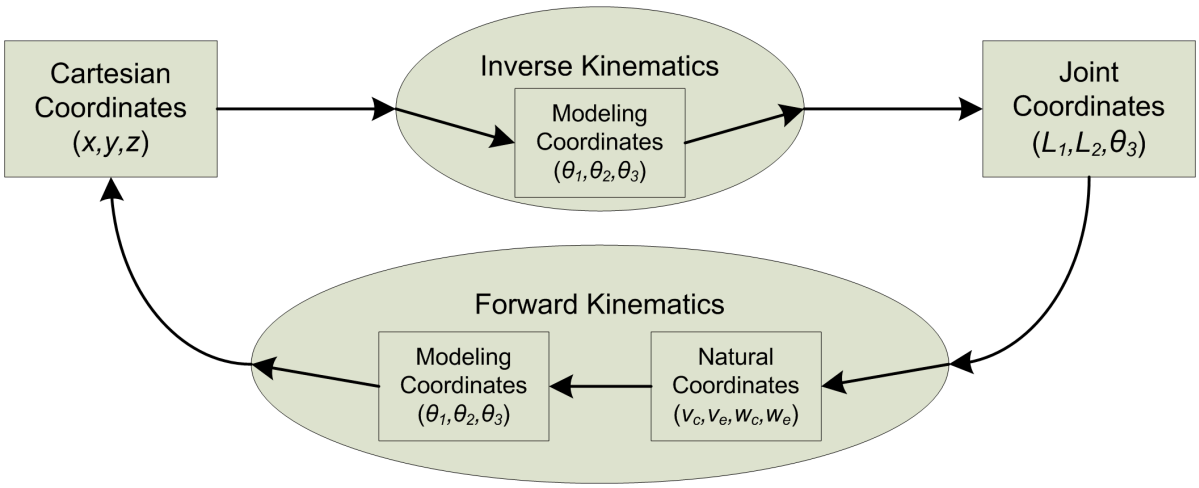


Figure 4.5: Transformation between coordinate sets.

solution to the forward kinematic equations. As mentioned in previous sections, the legs of the robot form a planar parallel revolute-prismatic-revolute (RPR) mechanism.

Kong and Gosselin [22] performed an extensive analysis of planar parallel mechanisms and the associated forward kinematics. They show that in general, a closed-form forward kinematic solution does not exist for a planar parallel RPR mechanism. In special circumstances, such as having three collinear revolute joints ($R_6 = 0$ in Figure 4.4), the forward kinematics are simplified, and a closed-form solution exists. Unfortunately, because of the vertical offset, $R_6 \neq 0$ and the legs of Hexplorer form a general planar parallel RPR mechanism. The forward kinematic solution hinges on one equation consisting of a sixth-order polynomial, the roots of which can only be determined iteratively [22]. A number of iterative techniques, such as Laguerre's method [13], could be used to solve this equation. However, implementing Laguerre's method requires that several divisions and square roots be calculated each iteration. As explained earlier, when precision is required for fixed-point numbers, the range of representation is sacrificed, making division a computationally risky operation. Alternatively, a system of four non-linear equations with four unknowns can be used to determine the forward kinematic configuration of a leg. Similar to the sixth-order polynomial, the non-linear system can be solved using an iterative technique. The Newton-Raphson algorithm is the preferred method for solving a system of non-linear equations.

In general, iterative techniques begin with an initial guess and improve the guess until it provides a solution within some tolerance.

Iteratively solving equations is an unfavourable proposition in real-time control [7]. Problems with convergence, the rate of convergence, and convergence to an incorrect solution, limit the appeal of iterative techniques. However, a good initial guess can address some of these concerns. The Newton-Raphson algorithm features quadratic convergence, meaning that convergence accelerates as the guess approaches the correct solution [13].

To ensure good guesses, Hexplorer frequently executes the Newton-Raphson algorithm using the results from the last solution as the initial guess for the next. The initial guess for the first execution of the algorithm is calculated offline for the home position of the legs. During its homing sequence, each leg reaches a configuration where all the joints reach their under-travel limit switches. In this configuration, the solution to the forward kinematics is known and that result is applied to the initial guess, solving the first execution of the Newton-Raphson algorithm in one iteration.

The Newton-Raphson iterative technique [13] is used in conjunction with natural coordinates to solve the forward kinematics of a leg. Given the two lead screw lengths, the coordinates of points c and e are determined from the kinematic constraints of the RPR mechanism. The coordinates of points c and e form the column matrix $\{\mathbf{x}_{Nat}\}$ where

$$\{\mathbf{x}_{Nat}\} = \begin{Bmatrix} v_c \\ v_e \\ w_c \\ w_e \end{Bmatrix}. \tag{4.5}$$

Two of the four kinematic constraints are found by equating the distances between points c and f, and between points e and h to lead screw lengths L_1 and L_2, respectively. The other two kinematic constraints are found by expressing R_1 and R_5 in $\mathbf{x}_{Nat}$. These constraints form the column matrix $\{\boldsymbol{\varphi}\}$ where

$$\{\boldsymbol{\varphi}\} = \begin{Bmatrix} \left(\frac{1}{2}\left(v_c + v_e\right) - R_7\right)^2 + \left(\frac{1}{2}\left(w_c + w_e\right) - R_6\right)^2 - R_1^2 \\ \left(v_c - R_3\right)^2 + \left(w_c\right)^2 - L_1^2 \\ \left(v_e - R_4\right)^2 + \left(w_e\right)^2 - L_2^2 \\ \left(v_c - v_e\right)^2 + \left(w_c - w_e\right)^2 - R_5^2 \end{Bmatrix}. \tag{4.6}$$

The Jacobian of $\{\boldsymbol{\varphi}\}$ with respect to the vector $\{\mathbf{x}_{Nat}\}$ is designated as $[\boldsymbol{\varphi}]_{\mathbf{x}_{Nat}}$ and is calculated as

$$
\begin{aligned}
[\boldsymbol{\varphi}]_{\{\mathbf{x}_{Nat}\}} &= \frac{\partial\{\boldsymbol{\varphi}\}}{\partial\{\mathbf{x}_{Nat}\}} \qquad (4.7)\\
&= \begin{bmatrix}
v_c + v_e - R_7 & v_c + v_e - R_7 & w_c + w_e - R_6 & w_c + w_e - R_6 \\
2(v_c - R_3) & 0 & 2w_c & 0 \\
0 & 2(v_e - R_4) & 0 & 2w_e \\
2(v_c - v_e) & 2(v_e - v_c) & 2(w_c - w_e) & 2(w_e - w_c)
\end{bmatrix}.
\end{aligned}
$$

One of the reasons for selecting natural coordinates is that the matrix $[\boldsymbol{\varphi}]_{\mathbf{x}_{Nat}}$ is composed entirely of linear combinations of the natural coordinates. In turn, calculating the values in the matrix is very predictable and numerically well-conditioned. The only operators are addition, subtraction, and multiplication by 2. Since all of the variables or constants are measuring values between $\pm 0.5m$, multiplication by 2 results in numbers that can be represented by the smallest fixed-point range [43]. The same logic applies to addition and subtraction. The Newton-Raphson algorithm works as follows:

1. An initial guess, $\{\mathbf{x}_{Nat}\}_k$ is selected.
2. Equation (4.8) is evaluated, where $\Delta\{\mathbf{x}_{Nat}\}$ represents the error between the true solution and $\{\mathbf{x}_{Nat}\}_k$.
3. Using the error $\Delta\{\mathbf{x}_{Nat}\}$, the guess is updated by Equation (4.9), resulting in a better guess, $\{\mathbf{x}_{Nat}\}_{k+1}$.
4. Repeat the process again from step 2 using $\{\mathbf{x}_{Nat}\}_{k+1}$, and repeat until the error $\Delta\{\mathbf{x}_{Nat}\}$ is within tolerance.

$$
\begin{aligned}
\left(\boldsymbol{\varphi} + [\boldsymbol{\varphi}]_{\{\mathbf{x}_{Nat}\}}\Delta\{\mathbf{x}_{Nat}\}\right)\Big|_{\{\mathbf{x}_{Nat}\}=\{\mathbf{x}_{Nat}\}_k} &= 0 \qquad (4.8)\\
\{\mathbf{x}_{Nat}\}_{k+1} &= \{\mathbf{x}_{Nat}\}_k + \Delta\{\mathbf{x}_{Nat}\} \qquad (4.9)
\end{aligned}
$$

Once complete, the natural coordinates are converted into into intermediate modelling

variables where

$$\theta_1 = \tan^{-1}\left(\frac{\frac{1}{2}(w_c + w_e) - R_6}{\frac{1}{2}(v_c + v_e) - R_7}\right) \text{ and} \tag{4.10}$$

$$\theta_2 = \tan^{-1}\left(\frac{w_e - w_c}{v_e - v_c}\right). \tag{4.11}$$

Simple 'forward kinematics' are applied to the modeling variables in order to obtain the Cartesian coordinates

$$x = (R_7 + R_1\cos\theta_1 + R_2\sin\theta_2)\cos\theta_3, \tag{4.12}$$

$$y = R_6 + R_1\sin\theta_1 - R_2\cos\theta_2, \text{ and} \tag{4.13}$$

$$z = -(R_7 + R_1\cos\theta_1 + R_2\sin\theta_2)\sin\theta_3. \tag{4.14}$$

By formulating the Newton-Raphson approach with natural coordinates, solving a linear system of 4 equations and 4 unknowns lies at the heart of the computation. Solving this system of linear equations can be performed two reasonable ways: an ad-hoc approach, or a numerical methods based approach. An ad-hoc approach refers to solving a linear system of equations by hand. This approach takes advantage of the structure of the Jacobian $[\boldsymbol{\varphi}]_{\{\mathbf{x}_{Nat}\}}$, especially entries of 0. However, the ad-hoc approach is not well-conditioned numerically, because of a large number of multiplications and divisions and consequently emulated floating-point operations must be used. The other option is to use a numerical technique such as LU decomposition. In LU decomposition, a matrix is decomposed into the product of a *Lower* triangular matrix and an *Upper* triangular matrix [13]. Because of the well-conditioned Jacobian $[\boldsymbol{\varphi}]_{\{\mathbf{x}_{Nat}\}}$ and partial pivoting, the LU decomposition algorithm solves this system accurately using fixed-point representation.

Table 4.1 shows the results of profiling the ad-hoc approach using emulated float-point operations versus LU decomposition and fixed-point representation. The results clearly show that the IQmath library coupled with LU decomposition is much faster than the ad-hoc approach.

The Newton-Raphson implementation used to determine the forward kinematics for Hexplorer works well. LU decomposition with partial pivoting using fixed-point numbers handles the computationally intensive task of the Newton-Raphson algorithm well. The algorithm easily achieves sampling rates of 100Hz, keeping guesses and solutions accurate enough that only 1 or 2 iterations are required to achieve errors less than $10^{-5}m$.

Table 4.1: Forward Kinematics Computation Performance Comparison: Fixed-Point LU Decomposition vs. Emulated Floating-Point Ad-Hoc

Technique	Average Operations	Maximum Sample Rate
Ad hoc	$1,843,558$	81 Hz
LU Decomposition	$467,838$	320 Hz

4.1.2 Inverse Kinematics

The inverse kinematic equations convert the Cartesian coordinates of a foot into leg modeling variables, and finally into joint variables. Closed-form inverse kinematics were originally developed in [6] and later corrected in [9]. Neither [6] nor [9] considered a case where a foot may be directly under the body rings. Adding the additional case, the closed-form inverse kinematics are given by

$$\begin{aligned}\theta_1 &= \cos^{-1}\left(\frac{R_1^2 - R_2^2 + \left(\sqrt{x^2+z^2} - R_7\right)^2 + (y-R_6)^2}{2R_1\sqrt{\left(\sqrt{x^2+z^2} - R_7\right)^2 + (y-R_6)^2}}\right) \\ &\quad -\text{atan2}\left(R_6 - y, \sqrt{x^2+z^2} - R_7\right), \end{aligned} \tag{4.15}$$

$$\begin{aligned}\theta_2 &= \frac{\pi}{2} - \cos^{-1}\left(\frac{R_2^2 - R_1^2 + \left(\sqrt{x^2+z^2} - R_7\right)^2 + (y-R_6)^2}{2R_2\sqrt{\left(\sqrt{x^2+z^2} - R_7\right)^2 + (y-R_6)^2}}\right) \\ &\quad -\text{atan2}\left(R_6 - y, \sqrt{x^2+z^2} - R_7\right), \text{ and} \end{aligned} \tag{4.16}$$

$$\theta_3 = -\tan^{-1}\left(\frac{z}{x}\right), \tag{4.17}$$

followed by a conversion to joint values

$$L_1 = \sqrt{\left(R_7 + R_1\cos\theta_1 - \frac{R_5}{2}\cos\theta_2 - R_3\right)^2 + \left(R_6 + R_1\sin\theta_1 - \frac{R_5}{2}\sin\theta_2\right)^2} \tag{4.18}$$

and

$$L_2 = \sqrt{\left(R_7 + R_1\cos\theta_1 + \frac{R_5}{2}\cos\theta_2 - R_4\right)^2 + \left(R_6 + R_1\sin\theta_1 + \frac{R_5}{2}\sin\theta_2\right)^2} \tag{4.19}$$

As with all kinematic equations, the fixed-point implementation of the inverse kinematics was examined. Equations (4.15) through (4.17) were optimized by hand. Repeated expressions such as $\sqrt{x^2 + z^2} - R_7$ are computed and stored in temporary variables. These temporary variables are then used to build Equations (4.15) through (4.17). The resulting source code is similar to optimized code generated by *Maple*,[1] only far fewer temporary variables are used and calculations of some expressions are duplicated. Equations (4.18) and (4.19) are well-conditioned for a fixed-point implementation. Neither equation has any division operators, recalling that $\frac{R_5}{2} = 0.5R_5$. Most of the multiplication operators involve squaring a number between ± 1 or multiplying by sine or cosine, which both ensure that the resultant is within the range of the selected fixed-point representation. Multiplication by sine and cosine have this property because they evaluate to numbers between ± 1.

4.1.3 Joint Velocities

The joint velocities are used to realize the Cartesian motion of the robot. Calculating the desired velocities of the leg joints is based on the desired Cartesian velocity of the foot, as well as the state of the leg itself. Mathematically, joint velocities are calculated using the forward or inverse kinematic equations and calculating the corresponding Jacobian matrix. Joint velocities can be calculated using forward kinematics:

$$\begin{Bmatrix} \dot{L}_1 \\ \dot{L}_2 \\ \dot{\theta}_3 \end{Bmatrix} = \begin{bmatrix} \frac{\partial x(L_1,L_2,\theta_3)}{\partial L_1} & \frac{\partial x(L_1,L_2,\theta_3)}{\partial L_2} & \frac{\partial x(L_1,L_2,\theta_3)}{\partial \theta_3} \\ \frac{\partial y(L_1,L_2,\theta_3)}{\partial L_1} & \frac{\partial y(L_1,L_2,\theta_3)}{\partial L_2} & \frac{\partial y(L_1,L_2,\theta_3)}{\partial \theta_3} \\ \frac{\partial z(L_1,L_2,\theta_3)}{\partial L_1} & \frac{\partial z(L_1,L_2,\theta_3)}{\partial L_2} & \frac{\partial z(L_1,L_2,\theta_3)}{\partial \theta_3} \end{bmatrix}^{-1} \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix}. \tag{4.20}$$

Or, joint velocities can be calculated using inverse kinematics:

$$\begin{Bmatrix} \dot{L}_1 \\ \dot{L}_2 \\ \dot{\theta}_3 \end{Bmatrix} = \begin{bmatrix} \frac{\partial L_1(x,y,z)}{\partial x} & \frac{\partial L_1(x,y,z)}{\partial y} & \frac{\partial L_1(x,y,z)}{\partial z} \\ \frac{\partial L_2(x,y,z)}{\partial x} & \frac{\partial L_2(x,y,z)}{\partial y} & \frac{\partial L_2(x,y,z)}{\partial z} \\ \frac{\partial \theta_3(x,y,z)}{\partial x} & \frac{\partial \theta_3(x,y,z)}{\partial y} & \frac{\partial \theta_3(x,y,z)}{\partial z} \end{bmatrix} \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix}. \tag{4.21}$$

Calculating joint velocities is typically done using the forward kinematics [7] because the Jacobian produced is based on the joint state of the leg, not the Cartesian state of the

[1] *Maple* is a software package produced by Waterloo Maple Inc.

foot. Since sensors usually measure joint variables, joint states are readily available for use in the Jacobian. It is also important to note that the invertability of this Jacobian matrix has an important physical meaning. When a singular Jacobian matrix is encountered, it means that the mechanism has reached a lock-up condition [18]. As such, the Jacobian is an important tool when examining the workspace of a mechanism. Alternatively, calculating joint velocities can be done using the inverse kinematic equations instead, but the Jacobian matrix is formed using the Cartesian position of the foot, which is not usually directly available from a sensor.

Using inverse kinematics to determine joint velocities was considered because of a lack of closed-form forward kinematic equations. However, in addition to being too cumbersome to analyse for numeric stability of the equations in a fixed-point environment, the *Maple* optimized source code to calculate joint velocities required more variables to calculate the Jacobian matrix than the DSP was able to allocate. Instead, the joint velocities are calculated using the 'forward kinematics' of the modeling variables (Equations (4.12) to (4.14)). Since the natural coordinates are updated frequently, they are used to estimate the state of the modeling variables (Equation (4.10) to (4.11)). The modeling variable velocities are then translated into joint velocities using Equations (4.19), (4.18), and their resulting Jacobian.

Finally, the fixed-point performance of the joint velocity equations was examined. The critical calculation involved solving the inverse of the 3×3 Jacobian. The inverse was solved using an ad-hoc approach with emulated floating point numbers and again using fixed-point LU decomposition. Just as before, the ad-hoc approach refers to a hand solution where one variable is the system is calculated. Then, using back-substitution with equations developed for the variable solved first, the remaining variables are calculated. As with the 4×4 case, the fixed-point LU decomposition of a 3×3 system out-performed the ad-hoc approach. The results are shown in Table 4.2.

Table 4.2: Joint Velocity Computation Performance Comparison: Fixed-Point LU Decomposition Forward Kinematics vs. Emulated Floating-Point Inverse Kinematics

Technique	Average Operations	Maximum Sample Rate
Emulated Floating-Point using Inverse Kinematics	416,866	359 Hz
Fixed-Point LU Decomposition using Forward Kinematics	156,486	958 Hz

4.2 Kinematic Simulation

The kinematic simulation was completed using the *MATLAB*[2] software package. It was used to implement Yoneda's algorithm, including the gait equations, and foot motion planning discussed in Section 3.1.6. In simulation, the robot can be controlled using equations or a joystick input to specify both horizontal velocities and the yaw rate of the robot. The body of the robot is assumed to be at a fixed height with neither any pitch or roll motion, *i.e.* $v_y = \omega_x = \omega_z = 0$. The graphical output is shown in Figure 4.6.

The simulation includes a number of visual representations, in addition to numerical data. The following can be shown visually during the simulation: the configuration of each leg, the phase of each leg (support feet are coloured red), the support polygon, the stability margin (S_M) calculated and normalized between the centre of the robot projected on to the ground and the nearest edge of the support polygon, and the height-dependent resizing of the horizontal workspaces.

4.2.1 Leg Workspace

In order to solve for or estimate the temporal kinematic margin, the workspace of the robot must be described in Cartesian coordinates. It is the joint coordinates, however, that define the workspace of a leg. The workspace boundary was determined by holding one lead screw at its minimum or maximum length while the other lead screw length varied between its minimum and maximum. The hip joint was held constant. Using the forward

[2] *MATLAB* is a software package produced by The MathWorks.

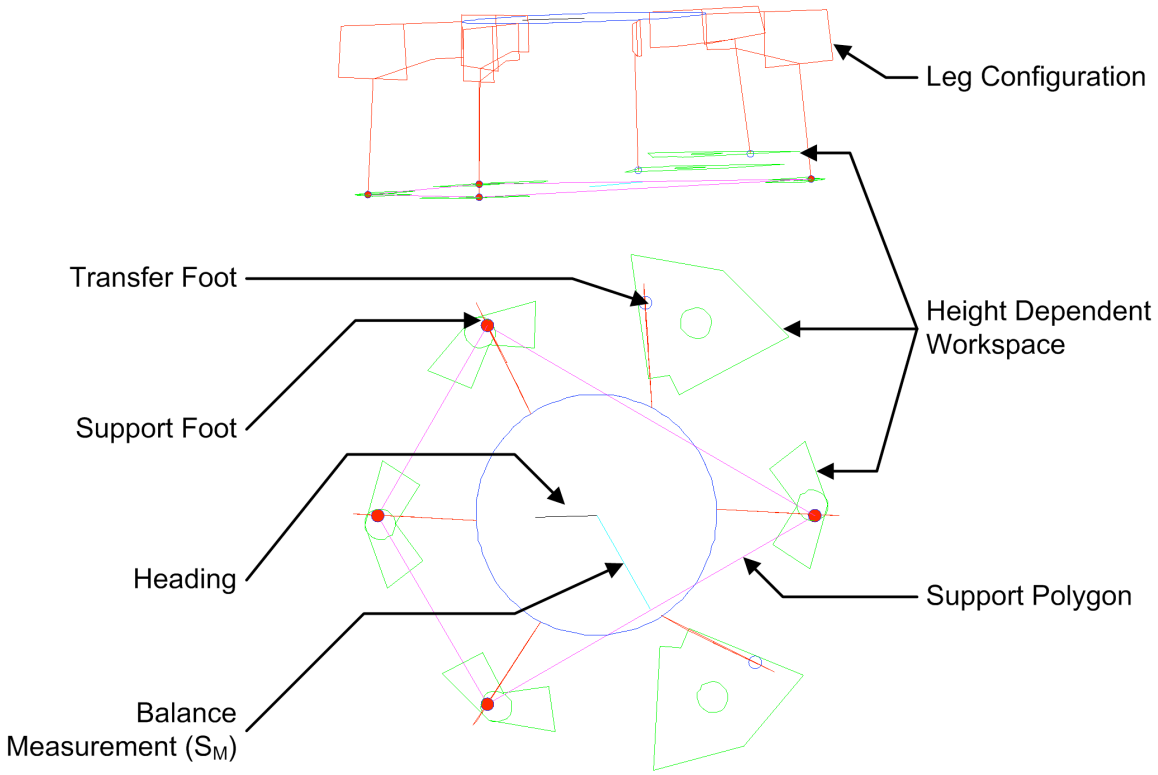
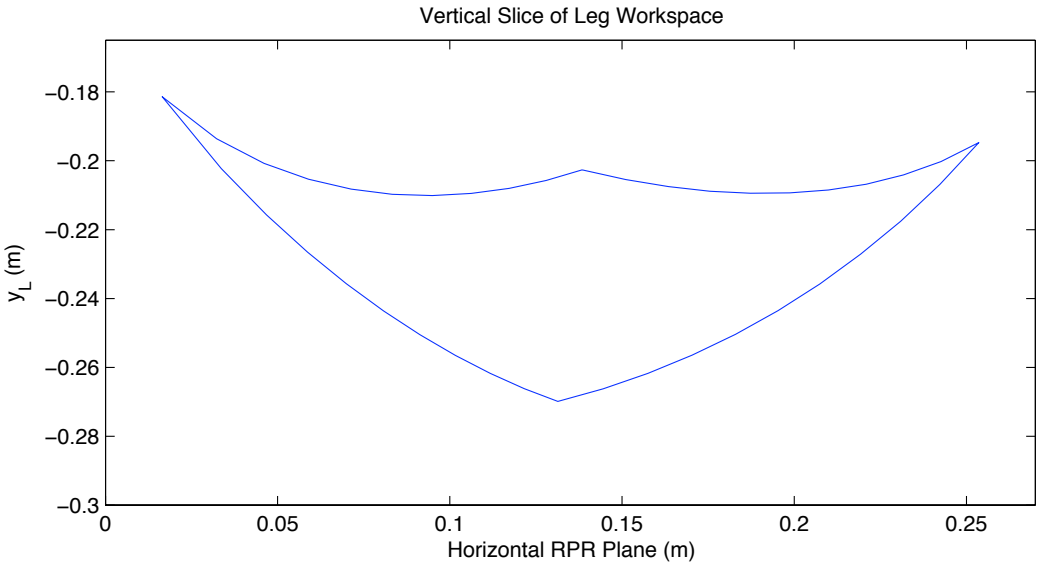


Figure 4.6: Hexplorer simulation graphics.

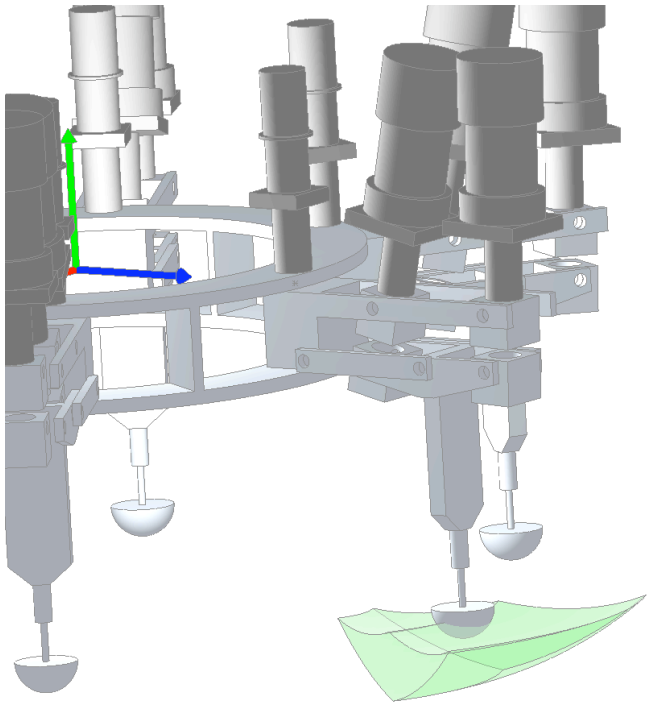
kinematics developed in Section 4.1, these lead screw lengths that traced out the workspace envelope, were converted into Cartesian coordinates. The resulting planar shape is shown in Figure 4.9(a). This shape was then revolved about the y-axis of the leg between the minimum and maximum hip rotations. The resulting workspace is shown in Figure 4.9(b).

The vertical planar slice of the workspace in Figure 4.9(a) indicates that the horizontal workspaces shrink as the foot moves downward with respect to the body. The horizontal workspace is important because the height of the robot is fixed, and support feet must therefore travel on the horizontal plane. The horizontal workspaces at $y_L = -0.24cm$ and $y_L = -0.22cm$ are shown in Figure 4.8 and demonstrate the reduced workspace at lower foot positions. The trade-off between ground clearance of transfer feet and workspace size is evident. As the size of the horizontal workspace increases, the amount of clearance afforded to transfer feet is reduced. The spring-loaded feet of the robot compound this problem. Gravity causes the spring loaded feet to be fully compressed when in the support phase; thus to clear the ground, the spring must uncompress. This requires that the foot be raised by an additional $1cm$ to achieve the desired ground clearance. Removing the springs from the feet would provide a much larger vertical workspace.

In order to calculate the temporal kinematic margin, the distance between the foot and the workspace boundary in the direction of travel, must be determined. By approximating the workspace of a leg as a set of piece-wise linear functions, the distance between a point in the workspace and a line on the boundary is determined using elementary algebra. In addition, by adding a scaling factor to the linearized workspace, the steps taken by the robot can be modified. Using the full linearized workspace results in large steps, while reducing the size of the linear workspace results in smaller steps. This is actually the basis for the constrained working volumes (CWVs) introduced by Lee and Orin [28]. The linearized vertical planar workspace is shown in Figure 4.9(a). The linearized horizontal workspace for $y = -24cm$ is shown in Figure 4.9(b). Careful examination of Figure 4.9(b) shows that for $y \leq -23.5cm$ the linearized workspace provides a conservative estimate of the actual workspace. On the other hand, for values of $y \geq -23.5cm$ the estimate of the linearized workspace is relaxed. This overestimation does not pose a problem because the linearized workspace is only used with the height of the body from the ground, not the height of the body from a foot. Because of the spring-loaded feet, in order to achieve a



(a) Vertical leg workspace.



(b) Workspace volume.

Figure 4.7: Leg workspace.

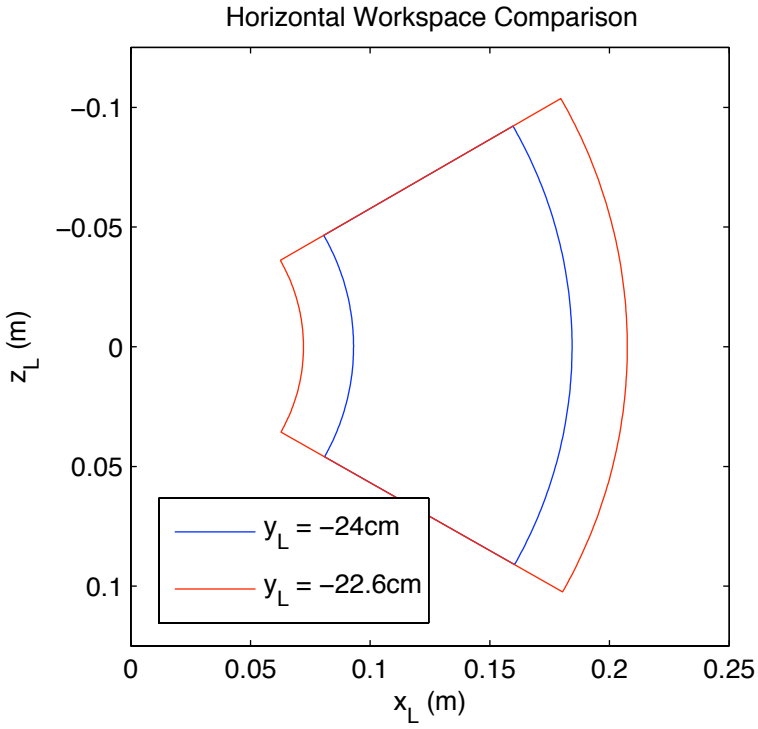


Figure 4.8: Comparison of horizontal leg workspaces at different distances below the body of the robot.

ground clearance of $y \leq -25cm$, the conservative linearized estimate of the workspace is used.

4.2.2 Improved Temporal Kinematic Margin

The temporal kinematic margin, as proposed by Lee and Orin, is a first-order approximation of the amount of time it will take until a foot reaches its workspace boundary. It is referred to as a first-order approximation in this thesis because it does not take into account the curvature of the trajectory between calculations, and it assumes that the velocity is constant in magnitude and direction. To improve the distance travelled in a fixed number of steps, taking into account the curvature of the trajectory between calculations was considered.

The motion of the foot relative to the leg i frame, was given by Equation (3.26). Considering only the horizontal plane formed by (x_i, z_i) from any leg i reference frame $\{L_i\}$, substitute the following values:

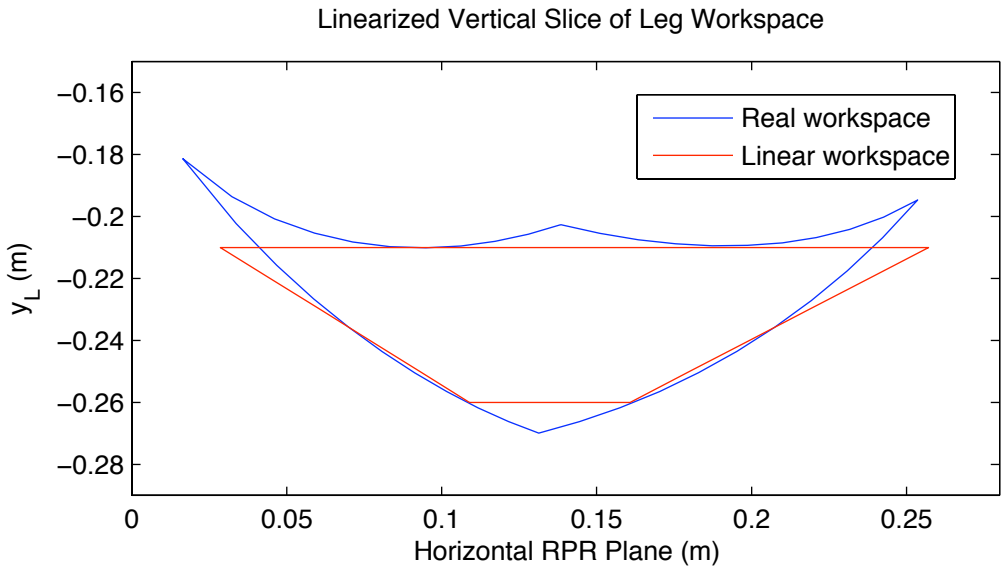
$$v_{foot/L_i} = \left\{ \begin{array}{c} \dot{x_i} \\ \dot{z_i} \end{array} \right\}, \tag{4.22}$$

$$r_{foot/L_i} = \left\{ \begin{array}{c} x_i \\ z_i \end{array} \right\}. \tag{4.23}$$

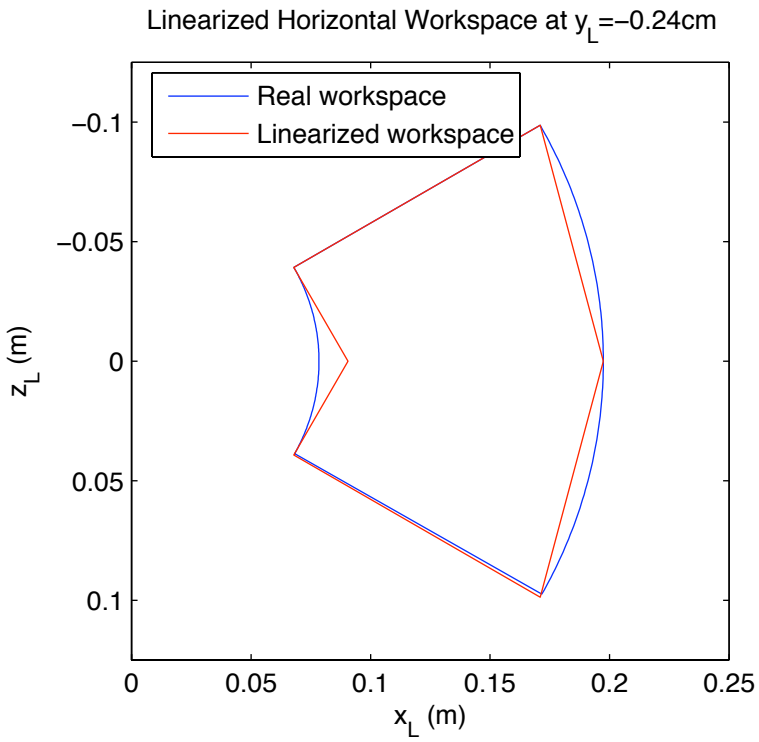
Assuming a constant velocity and yaw rate, the velocity of the foot relative to the leg is given by

$$\left\{ \begin{array}{c} \dot{x_i} \\ \dot{z_i} \end{array} \right\} = \left\{ \begin{array}{c} v_z \sin\delta - v_x \cos\delta - \omega_y z_i \\ -v_z \cos\delta - v_x \sin\delta + \omega_y(\rho + x_i) \end{array} \right\}. \tag{4.24}$$

This forms a set of linear ordinary differential equations (ODEs). Solving this set of ODEs assuming v_x, v_z, and ω_y are constant and applying initial conditions $x_i(0)$ and $z_i(0)$, which represent the initial position of the foot, expressions as functions of time for the relative



(a) Linearized vertical leg workspace.



(b) Linearized horizontal leg workspace.

Figure 4.9: Linearized leg workspace.

position of the foot with respect to the leg reference frame are found to be

$$\left\{ \begin{array}{c} x_i(t) \\ \\ z_i(t) \end{array} \right\} = \left\{ \begin{array}{c} -\sin(\omega_y t) z_i(0) + \cos(\omega_y t)\left(x_i(0) + \rho\right) - \rho + \\ \frac{v_x}{\omega_y}\left[\sin\delta - \sin\delta\cos(\omega_y t) - \cos\delta\sin(\omega_y t)\right] + \\ \frac{v_z}{\omega_y}\left[\cos\delta + \sin\delta\sin(\omega_y t) - \cos\delta\cos(\omega_y t)\right] \\ \\ \cos(\omega_y t) z_i(0) + \sin(\omega_y t)\left(x_i(0) + \rho\right) + \\ \frac{v_z}{\omega_y}\left[\sin\delta - \sin\delta\cos(\omega_y t) - \cos\delta\sin(\omega_y t)\right] + \\ \frac{v_x}{\omega_y}\left[\cos\delta - \sin\delta\sin(\omega_y t) + \cos\delta\cos(\omega_y t)\right] \end{array} \right\}. \quad (4.25)$$

Using these expressions and the linearized workspace of the robot, the amount of time it takes to reach the nearest boundary (the temporal kinematic margin) can be estimated. The sinusoidal functions in the expression prevent a perfectly accurate closed-form solution from being determined. Thus, an approximation is required to determine a closed-form solution. The approximation is made by approximating sine and cosine functions as second order polynomials like

$$cos(\omega_y t) = c_2(\omega_y t)^2 + c_1(\omega_y t) + c_0 \text{ and} \quad (4.26)$$

$$sin(\omega_y t) = s_2(\omega_y t)^2 + s_1(\omega_y t) + s_0. \quad (4.27)$$

The constants $c_i, i = 0 \ldots 2$ and $s_i, i = 0 \ldots 2$ are determined by fitting each quadratic to its sinusoid using a least-squares estimate. The accuracy of the sinusoid approximation depends on which part and how much of each sinusoid is fitted. Consequently, the temporal kinematic margin is also affected by this decision. Based on experimentation, the largest distances travelled in a fixed number of steps were found when sine and cosine were approximated between $0 \leq \omega_y t \leq 0.6\ rad$.

For completeness, if the yaw of the robot is fixed, $\omega_y = 0$, the ODEs in Equation (4.24) solve to:

$$\left\{ \begin{array}{c} x_i(t) \\ z_i(t) \end{array} \right\} = \left\{ \begin{array}{c} \left(v_z \sin\delta - v_x \cos\delta\right) t + x_0 \\ \left(-v_z \cos\delta - v_x \sin\delta\right) t + z_0 \end{array} \right\}. \quad (4.28)$$

These equations are used when the robot is in pure translation. Since all of the input velocities are assumed to be constant, this equation is the first-order approximation used by Yoneda.

To examine the effectiveness of the higher-order temporal kinematic margin (t_{Si}), the system was simulated using two different sets of inputs. Each simulation was executed until the robot had completed five gait cycles, or steps, and the performance was evaluated by the total distance travelled in the five steps. In the first set the robot undergoes pure rotation while in the second set the rotation is reduced and the robot has translational velocity components. A case with no rotational velocity was not considered for two reasons. Firstly, the higher-order equation, Equation (4.25), is only defined when some rotational input exists. Secondly, Equation (4.28), used for pure translation, is the same first-order approximation used by Yoneda, and consequently the results would be identical.

The inputs and results for the first simulation are summarized in Table 4.3. For a

Table 4.3: Travelling 5 Gait Cycles in Pure Rotation

Input $v_x(t) = 0.0m/s$, $v_z(t) = 0.0m/s$, $\omega_y(t) = 0.15rad/s$		
Measure	First-order t_{Si}	Higher-order t_{Si}
Distance (m)	0.0	0.0
Rotation (rad)	2.42	3.52
Minimum S_M	0.33	0.32
Mean S_M	0.52	0.50

pure rotation, the higher-order temporal kinematic margin improved upon the first-order approximation by rotating the body of the robot 45% further. The reason for such a dramatic improvement is because the higher approximation incorporates the curvature of the path of the foot through the workspace and the oblong shape of the workspace. Without considering curvature, the first order approximation frequently ran into the boundaries along the narrow part of the workspace, which in turn forced the robot to advance its kinematic phase too quickly. Advancing the kinematic phase too quickly prematurely ends one step and begins the next, explaining the above results. It should also be noted that the mean and minimum measures of balance (S_M) only decreased 3% using the higher-order approximation.

The inputs and results for the second simulation are summarized in Table 4.4. In this simulation the yaw rate was significantly lower than in the previous simulation and therefore the paths travelled by the feet in the workspace were less curved. Recall that

Table 4.4: Travelling 5 Gait Cycles with Little Rotation

Input $v_x(t) = -0.01m/s$, $v_z(t) = -0.01m/s$, $\omega_y(t) = 0.05rad/s$		
Measure	First-order t_{Si}	Higher-order t_{Si}
Distance (m)	0.37	0.38
Rotation (rad)	1.32	1.34
Minimum S_M	0.34	0.4
Mean S_M	0.50	0.50

the curvature of the path travelled, or rather, the relative path between the foot and the local leg reference frame is governed by Equation (4.25). Consequently the higher-order approximation of the temporal kinematic margin was less beneficial and its performance was on par with the first-order approximation. It is interesting to note however, that although the mean stability margins were identical, the minimum stability margin of the higher-order temporal kinematic margin improved by 17%. This means that when using the higher-order temporal kinematic margin the robot is theoretically less likely to tip over than if the first-order approximation was used.

Based on the results of these simulations, the higher-order approximation of the temporal kinematic margin improved the distance travelled in a fixed number of steps as the yaw rate of the robot increased. This is due to the improved accuracy of the calculation as well as the oblong shape of the leg workspace. While this approximation yields at least a marginal improvement over the first order temporal kinematic margin, the added computational complexity of solving it does not warrant its use on Hexplorer because of limited CPU resources.

4.2.3 Horizontal Foot-Hold Selection

As discussed in Section 3.1.6, Yoneda selects a foot-hold position for a transfer leg based on a circle located in the horizontal plane. Instead of using a circle, a scalable linear piecewise scalable approximation of the workspace is used in conjunction with the temporal kinematic margin (t_{Si}) in order to determine a suitable foot-hold location. The proposed technique determines the foot hold position by determining t_{Si} when the supervisory inputs

are reversed, $-\mathcal{I}(t)$. The temporal kinematic margin is then applied to the trajectory of the foot to find the foot-hold location. In order to calculate t_{Si}, the position of the foot is required. When using the t_{Si} to calculate the foot-hold location, the centre of the workspace is used, as opposed to the actual position of the foot. This prevents the foot-holds from getting locked into the corners of the workspace.

Similar to the previous section, a series of simulations were executed to determine the total distance travelled and rotation achieved in five gait cycles using Yoneda's foot-hold selection and the proposed foot-hold selection technique. It is important to realize that Yoneda may have selected the circular foothold due to the shape of the workspace of his robot. The leg workspaces of Hexplorer on the other hand, are oblong and Yoneda may have suggested a different shape in this situation.

The inputs and results for the first simulation are summarized in Table 4.5. In this

Table 4.5: Travelling 5 Gait Cycles in Pure Translation

Input $v_x(t) = 0.02cm/s$, $v_z(t) = 0.0cm/s$, $\omega_y(t) = 0.0rad/s$			
Measure	Yoneda	Proposed (with 1^{st} order t_{Si})	Proposed (with higher-order t_{Si})
Distance (m)	0.32	0.33	N/A
Rotation (rad)	0.0	0.0	N/A
Minimum S_M	0.43	0.30	N/A
Mean S_M	0.48	0.46	N/A

simulation the only command input was a translational velocity in the x_{L_1} direction. As it happens, in this direction of travel, the size of Yoneda's circular subset is equivalent to the linearized workspace subset for legs 1 and 4. This can be seen in the results, as the distance travelled by the robot between Yoneda's foothold selection technique and the proposed foothold selection technique is identical. However, Yoneda's foot hold selection balances the robot slightly better. This occurs because using the first-order technique, legs other than 1 and 4 are attempting to maximize the distance travelled along the oblong direction of the workspace; whereas using Yoneda's technique these legs are constrained within a small circular region and consequently more centred, ultimately resulting in better balance.

The inputs and results for the second simulation are summarized in Table 4.6. Due to

Table 4.6: Travelling 5 Gait Cycles in Pure Rotation

Input $v_x(t) = 0.0m/s$, $v_z(t) = 0.0m/s$, $\omega_y(t) = 0.15rad/s$			
Measure	Yoneda	Proposed (with 1^{st} order t_{Si})	Proposed (with higher-order t_{Si})
Distance (m)	0.0	0.0	0.0
Rotation (rad)	2.26	2.42	3.52
Minimum S_M	0.40	0.33	0.32
Mean S_M	0.52	0.52	0.52

the oblong workspace, the simulations using both the first order and higher order temporal kinematic margins and the proposed foothold selection technique outperformed Yoneda's technique by rotating 7% and 56% more.

The inputs and results for the third simulation are summarized in Table 4.7. The

Table 4.7: Travelling 5 Gait Cycles with Rotation and Translation

Input $v_x(t) = -0.01m/s$, $v_z(t) = -0.01m/s$, $\omega_y(t) = 0.15rad/s$			
Measure	Yoneda	Proposed (with 1^{st} order t_{Si})	Proposed (with higher-order t_{Si})
Distance (m)	0.26	0.37	0.38
Rotation (rad)	0.93	1.32	1.34
Minimum S_M	0.40	0.34	0.40
Mean S_M	0.51	0.50	0.50

proposed foothold selection technique improved the distance travelled and rotation each by approximately 40% for both calculations of the temporal kinematic margin.

The overall pattern of these results, is that larger steps can be taken when larger workspaces are available. In the first simulation, Yoneda's foot-hold selection technique provided a larger effective workspace. In the second and third simulations, the proposed method, with its larger oblong workspace, outperformed Yoneda's foot-hold selection technique. Based on these results, the best solution would be to increase the number of nodes

used to linearize the workspace, such that the resulting workspace would be as large, and usually larger than Yoneda's circular workspace, while retaining its oblong shape. However, increasing the number of piece-wise linear functions to describe the workspace also significantly increases the computations required to determine t_{Si}. Since floating-point computing power is fairly limited on the DSPs, the simple linearized oblong workspace shown in Figure 4.9 was implemented on Hexplorer.

4.3 Summary

The kinematics required to implement Yoneda's algorithm on Hexplorer were investigated in this chapter. The forward kinematics are calculated using a Newton-Raphson iterative algorithm and natural coordinates. By selecting the initial guess to be the previous solution, only one iteration of the algorithm is required to solve the forward kinematics. In the support phase, the joint velocities of a leg are calculated based on the results of this algorithm and the commanded inputs $\mathcal{I}(t)$. In the transfer phase, the joint positions are calculated based on the inverse kinematics and a horizontal foot-hold selection. All of these equations are implemented using the high performance floating-point library IQmath and were carefully evaluated to provide accurate fixed-point results at reasonable sampling rates. With a full kinematic simulation of the robot complete, the implementation on the robot is discussed in the next chapter.

Chapter 5

Implementation and Results

Having already developed the necessary theoretical and mathematical background, this chapter deals with the actual implementation of Yoneda's algorithm on Hexplorer and the corresponding results.

Although Hexplorer was able to move its legs in the prescribed motion, with its body atop a platform and its legs in mid-air in both the stance and transfer phases, it was not able to walk on the ground for more than a step or two. The hip encoder on Leg 5 failed when the leg was in its stance phase. As per the recommendations in the next chapter, moving the encoder from the joint to the motor powering the joint will address this problem, and provide better resolution resulting in better velocity control.

5.1 Gait Algorithm Implementation

The block diagram in Figure 5.1 provides an overall perspective of the gait implementation. Control of the robot is accomplished using a set of finite state machines that communicate with one another. Each leg and the brain have their own state machines that are interlocked at a few locations.

The leg DSP is responsible for servicing a 100Hz control loop to maintain either the desired velocity or position of the foot with respect to the leg. 100Hz represents the maximum control loop frequency. This is governed by the speed of the DSP and complexity of the kinematic equations as discussed in Chapter 4. As part of this routine, the forward

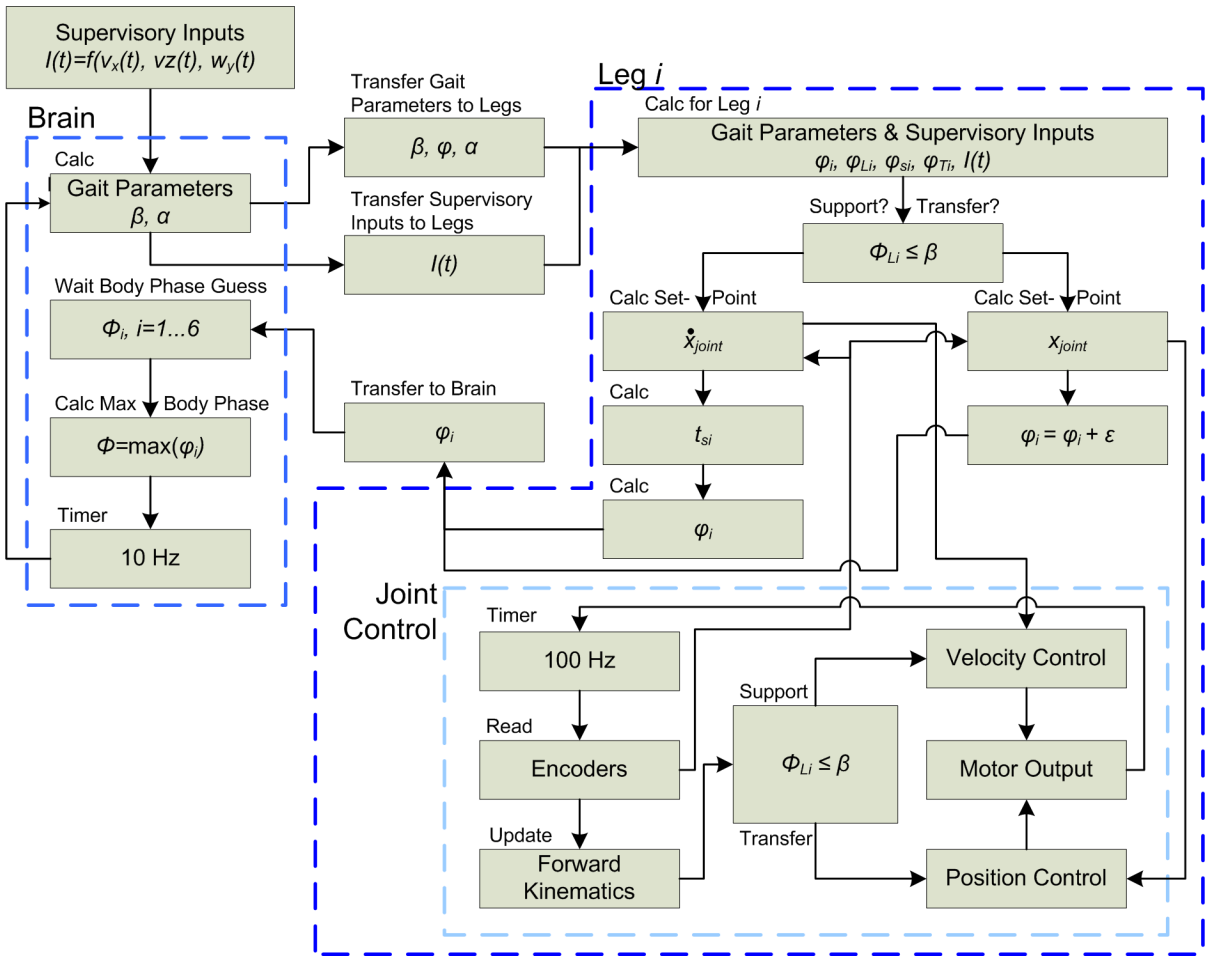


Figure 5.1: Overall block diagram of gait implementation.

kinematics are updated using the Newton-Raphson technique. If the leg is in its support phase, then the desired joint velocities are calculated and provided to the joint controllers. Otherwise, if the leg is in its transfer phase, then the desired joint position based on the selected foot-hold is provided to the joint controllers.

The leg and brain also have an interlocking 10Hz control loop that services the gait of the robot. This gait control loop is slower than the motor control loop because of the data transmission times required between the legs, brain, and supervisory software interface. Initially, the brain requests supervisory inputs from the user. The duty factor, β, and crab angle, α, are calculated and, along with the inputs $\mathcal{I}(t)$ and the kinematic phase ϕ of the body, are then transferred to all of the legs. The brain now awaits the new kinematic phase estimates from each leg.

Upon receiving these inputs, each leg calculates its gait parameters and executes one iteration of the control loop. Then, if the leg is in its support phase, one of two calculations will occur. If the leg has reached its workspace boundary, the kinematic phase ϕ will be incremented until the leg is to begin its transfer phase. This value is sent to the brain as ϕ_i. If the leg has not reached its workspace boundary, the kinematic phase ϕ will be updated using the temporal kinematic margin, and sent to the brain as ϕ_i. However, if the leg is in its transfer phase, the estimate of the kinematic phase, ϕ_i, sent to the brain will be calculated as, $\phi_i = \phi + \epsilon$ where ϵ is a small value (typically equal to 0.005). After transferring its estimate ϕ_i to the brain, the leg will enter the 100Hz control loop, keeping either the velocity of the foot constant or moving the foot to its desired foot-hold. The leg will remain in the 100Hz control loop, until new supervisory inputs are received from the brain. Having collected all estimates for the kinematic phase, the brain selects the maximum estimate to become the new kinematic phase. The brain then waits until the next iteration of the 10Hz control loop to request the supervisory inputs and repeat the process.

As discussed in Section 2.2.4, the robot must first home each leg to calibrate the absolute measurements of the encoders. The homing sequence is initiated by the user through the control software. This signal is received by the brain. The brain then initiates the homing sequence for the first leg. Upon completion of its homing sequence, the leg indicates that it has homed successfully. The brain then initiates the homing sequence for the next leg,

until all legs have been homed. Because the positions of the leg joints are unknown at start-up and several legs may be off the ground, the legs are homed individually to try and prevent the robot from tipping over during its homing sequence. Once all legs are successfully homed, the robot enters its normal operating mode.

The graphical user interface to the robot control software is used to supply supervisory inputs to the robot. The interface is shown in Figure 5.2. This software was written

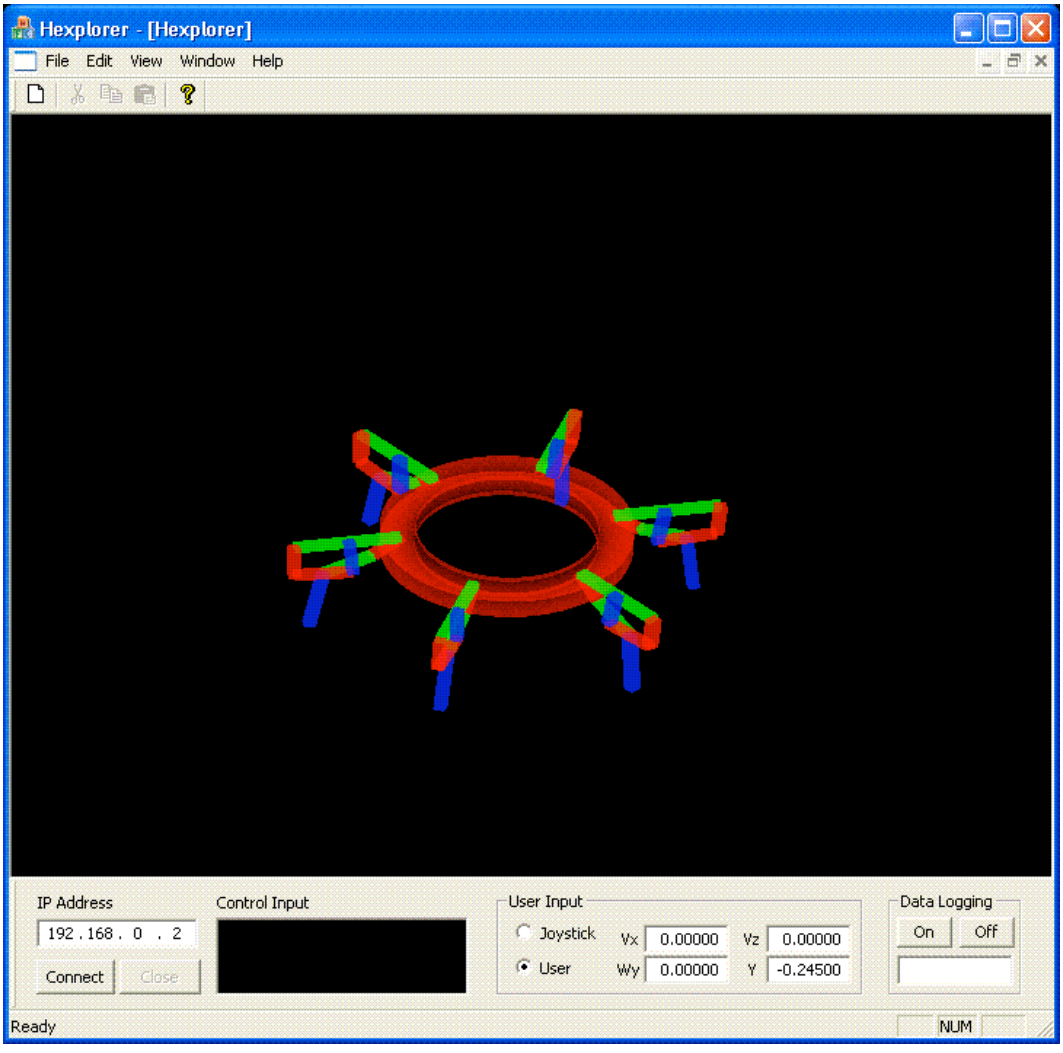


Figure 5.2: Graphical user interface to control software for Hexplorer.

using Microsoft Foundation Classes (MFC) in C++. It is a multi-threaded application that supports TCP/IP communication with the robot. An OpenGL representation of the robot, shown in Figure 5.2, mimics the leg configurations on the robot. The OpenGL window can be zoomed, rotated, or panned. The control software also supports very rudimentary data logging that records the Cartesian position of the feet compared to their respective leg frames. The supervisory inputs can be supplied using a numeric constant or a 4-axis joystick. Another control on the joystick is used to adjust the heave of the robot, $y(t)$.

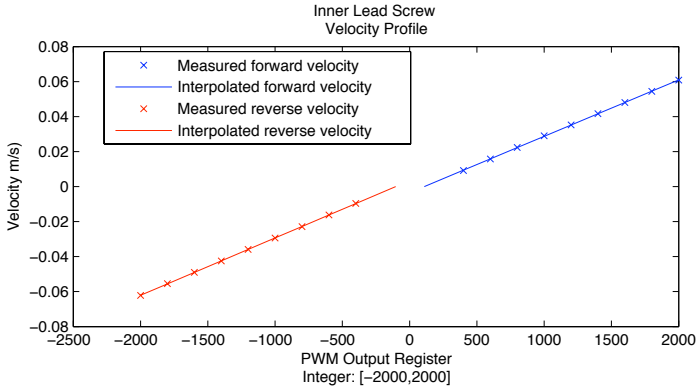
With the overall implementation of the gait algorithm defined, the details of the joint controllers are discussed, as are modifications to the Leg DSP board circuitry introduced in Section 2.2.4.

5.2 Joint Controllers

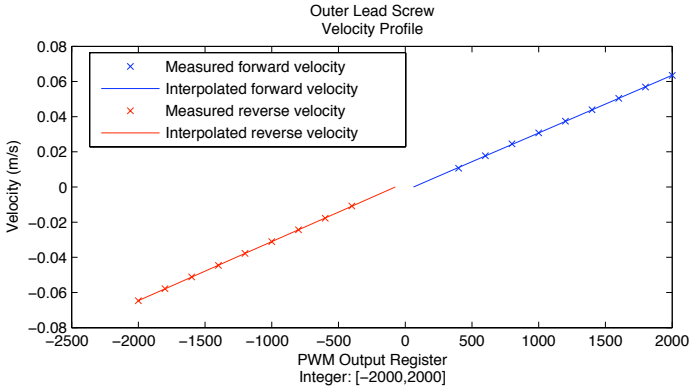
Two different joint controllers are used to implement the gait. In its support phase, the leg uses a joint velocity controller to maintain the correct motion of the body of the robot, whereas in its transfer phase, the leg uses inverse kinematics and a joint position controller.

5.2.1 Velocity Control

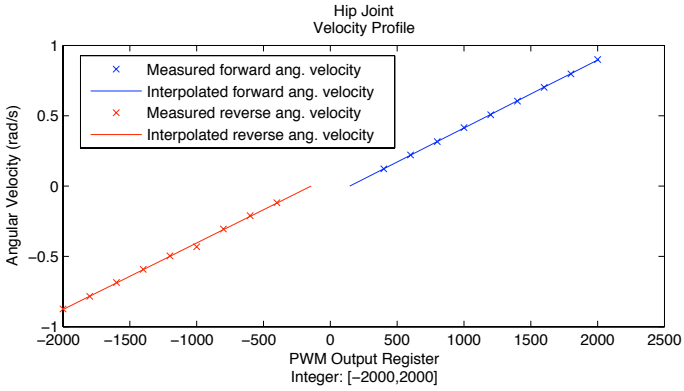
The desired instantaneous velocity of the foot with respect to its leg frame is given in Cartesian coordinates. These Cartesian velocities are then converted into joint velocities and applied to the joint controller as the desired input. The entire gait algorithm is based around planar motion with no body motion along the vertical axis. This means that during the support phase, the Cartesian velocity in the vertical direction is zero. Ideally, the robot would implement a position-based controller on a vertical joint to maintain the body height. The legs, however, do not have such a joint. Instead, a position controller is placed on the vertical position of the robot. The error generated by this controller is converted into a vertical velocity, appended to the supervisory inputs $\mathcal{I}(t)$, and finally converted into desired joint velocities. Operating the joints at the specified velocities permits the feet to travel according to the inputs and allows the height of the robot to be maintained at a near-constant value.



(a) Inner lead screw velocity profile.



(b) Outer lead screw velocity profile.



(c) Hip joint angular velocity profile.

Figure 5.3: Leg joint velocity profiles

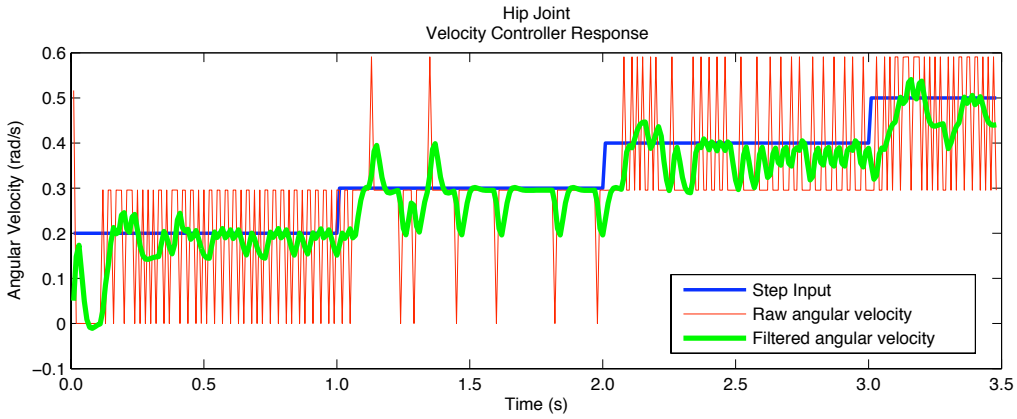
The joint velocity controllers are simple Proportional-Integral (PI) controllers with a feed-forward term. PI controllers possess the benefit of eliminating steady-state error. The feed-forward terms for the controllers are linear piece-wise functions that were determined empirically. Each joint was run forward or in reverse at specific pulse-width modulated (PWM) outputs for a set distance. Using the total time for each joint to travel a fixed displacement, an average velocity was calculated and finally mapped back to the PWM input using linear interpolation. This open-loop feed-forward term is supplemented with a PI controller having relatively small gains. This allows each joint to compensate for external disturbances and non-linearities excluded by the linear interpolation. Figure 5.3 shows plots of the feed-forward terms. The linear relationship between the PWM output and joint speed are quite evident from these plots.

The hip joint has especially low PI gains and functions in nearly an open-loop manner. For stability reasons, low gains allow some disturbance rejection and a minor reduction in steady-state error while the overall control signal is dominated by the feed-forward term. Velocity joint control for the hip is extremely difficult because of the poor resolution from the optical encoders. Consider the minimum non-zero angular velocity that the DSP is capable of computing, based on encoder measurement. To measure the minimum non-zero angular velocity, the encoder must change by one pulse within the sample time. Each hip encoder count corresponds to $0.0031rad$ and the sample time is $10ms$. Given

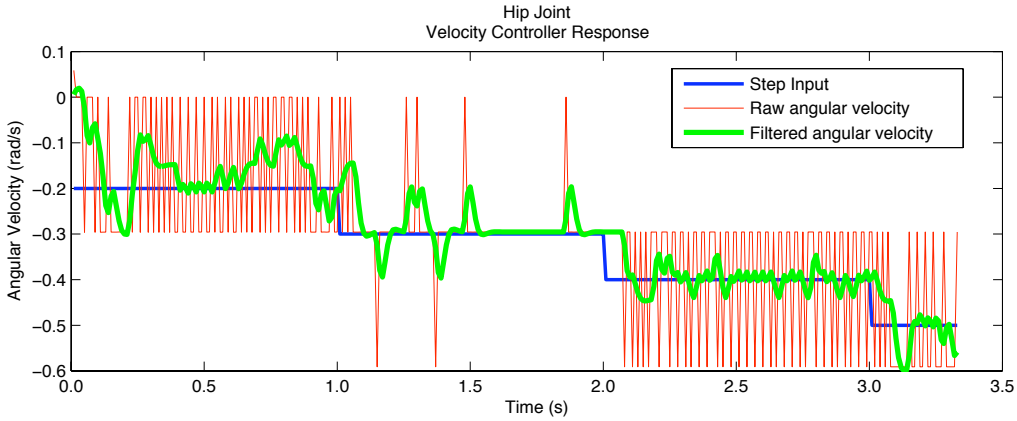
$$\dot{\theta_3} \approx \frac{\theta_{3_k} - \theta_{3_{k-1}}}{\Delta t}, \tag{5.1}$$

the minimum non-zero angular velocity measured by the encoder is $0.307rad/s$. However, the hip joint typically operates between 0 and 0.2 rad/s.

Figure 5.4 shows the results of the hip velocity controller responding to a set of step inputs. As can be seen, the raw velocity estimate of the hip is very noisy, and consequently provides a poor feedback signal to a controller. In order to counteract the fluctuations between zero and non-zero velocities, a second-order Butterworth filter is used with a cut-off frequency of 6Hz. Applying the second-order Butterworth filter to this raw signal permits a reasonable feedback signal to be generated. Although the Butterworth filter introduces a delay in the system response, the benefit of creating a reasonably smooth feedback signal far outweighs any minor delays that are introduced. Likewise, the lead



(a) Hip joint response with velocity controller to forward step input.



(b) Hip joint response with velocity controller to reverse step input.

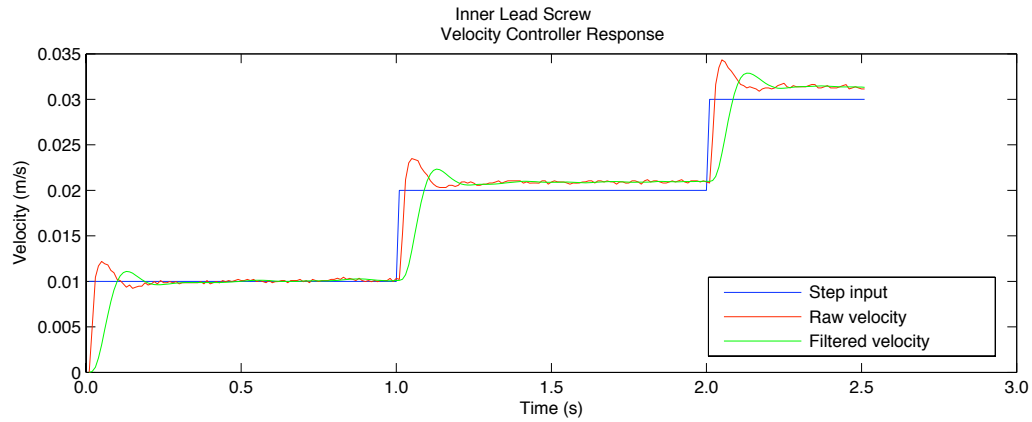
Figure 5.4: Hip joint response with velocity controller.

screw velocity controllers also used Butterworth filters; however, these filters have large bandwidths because their presence is primarily for flexibility. Figures 5.5 and 5.6 show the performance of the inner and outer lead screw velocity controllers. The contrast between the unfiltered velocities of the lead screws and hip motor could not be more prominent. With a far better resolution, the raw velocities calculated from the lead screw encoders are far more stable and uniform. Simply relocating the hip joint encoder to the other side of the gearbox should transform the current raw hip velocity, Figure 5.4, into a far smoother and more stable signal, much like those of the lead screws in Figures 5.5 and 5.6. Relocating the encoder is not a simple task, and is unfeasible in the project at hand. In addition to the mechanical changes required for mounting new motor with built-in encoders, the circuitry would require substantial changes. As discussed in Section 2.2.4, the quadrature decoding required for the hip motor is emulated in the DSP. Relocating the encoder improves the resolution of the hip position, and in so doing, increases the number of pulses to be decoded by the DSP by a corresponding magnitude. In other words, the DSP is required to process an increased number of pulses without affecting the overall performance of the controllers and gait algorithm. It is unlikely that the DSP will be able to handle such a processing load and consequently, the quadrature decoding for the hip will have to be moved off-chip and on to the circuit board. This in turn requires new circuitry on a new circuit board, making the hip encoder relocation infeasible within the time constraints for the task at hand, but necessary for future work.

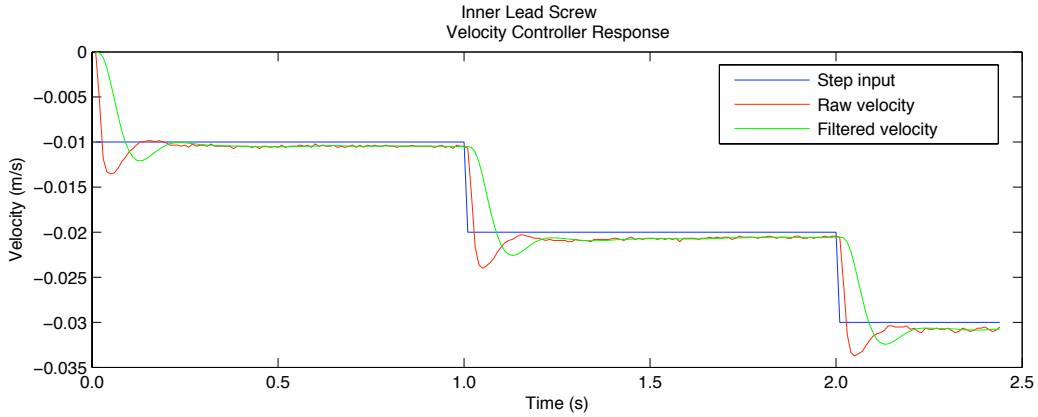
5.2.2 Position Control

Position control is used to place the foot of a leg at a desired location during the transfer phase of the leg. Specified in Cartesian coordinates, the position of the foot is then determined in terms of its joint position using the inverse kinematics developed earlier in Section 4.1.2. In order to arrive at the joint position specified, one Proportional-Integral (PI) controller is used on each joint until the actual joint position is close enough to the desired joint position. Again, a PI controller is used to eliminate steady-state error that may occur. The position controller in the transfer phase is set to the desired targets as follows:

1. if $\phi_{T_i} \leq 0.3$, then the relative x and z coordinates of the foot are held fixed, while

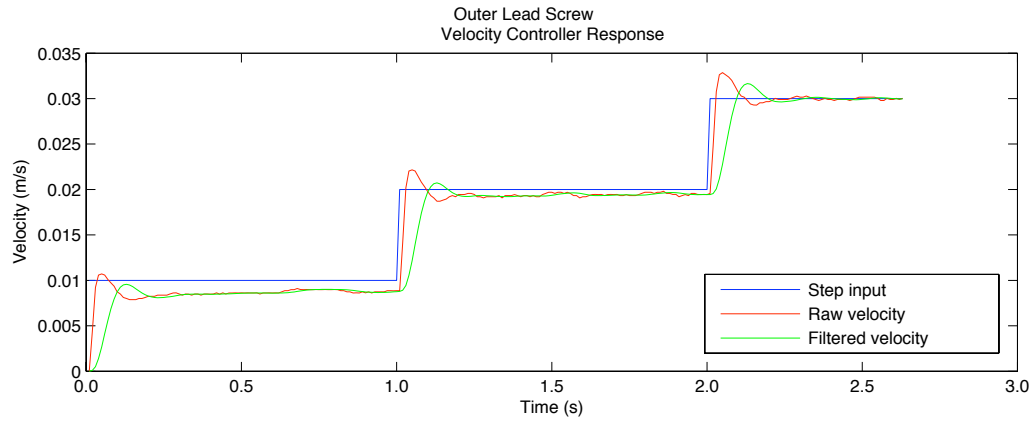


(a) Inner lead screw response with velocity controller to forward step input.

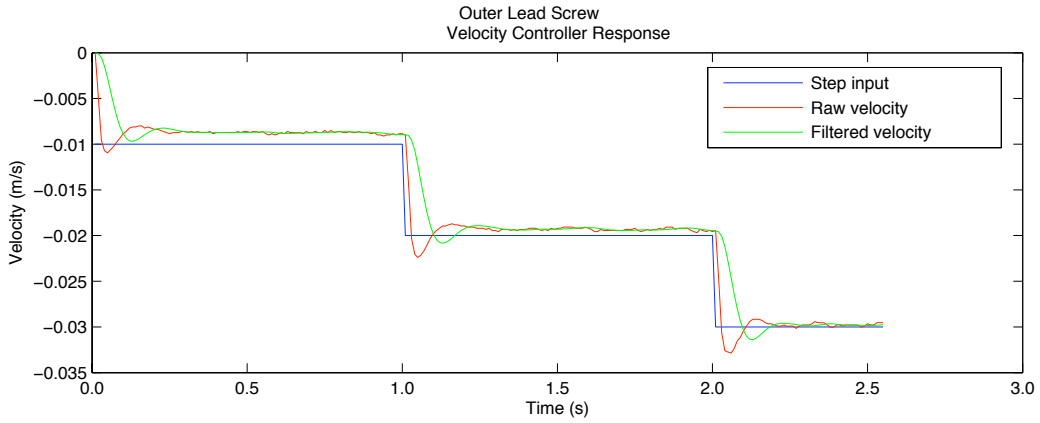


(b) Inner lead screw response with velocity controller to reverse step input.

Figure 5.5: Inner lead screw response with velocity controller.



(a) Outer lead screw response with velocity controller to forward step input.



(b) Outer lead screw response with velocity controller to reverse step input.

Figure 5.6: Outer lead screw response with velocity controller.

the foot is lifted to achieve the desired ground clearance

2. if $0.3 < \phi_{T_i} \leq 0.7$, then the y coordinate of the foot is held fixed, while the foot is relocated above its new foot-hold location (as per Section 3.1.6)

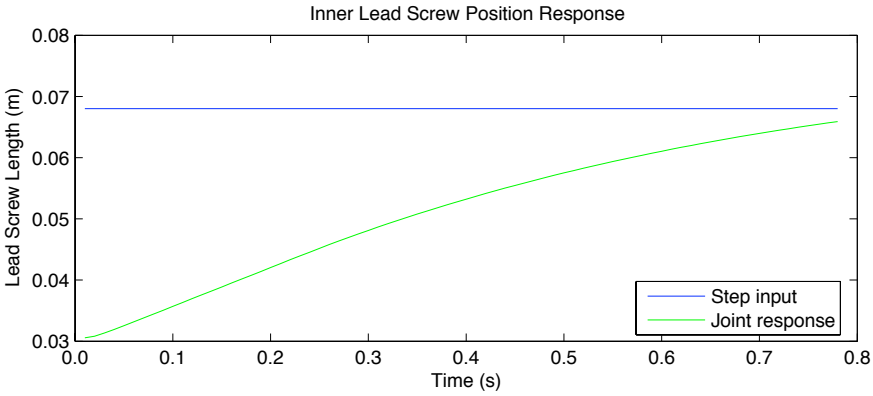
3. if $\phi_{T_i} > 0.7$, then the relative x and z coordinates of the foot are held fixed, while the foot is lowered to achieve the correct body height of the robot

Very precise position control was deemed too difficult for the purposes of the research for this thesis because of the large amount of backlash and flexibility within the joints, especially the hip joint. The only critical property of the controller involved overshoot. When the initial position of a joint is at one extreme of the workspace and the desired position is at another, overshoot caused by the position controller could cause the joint to activate an E-Stop. In order to prevent this, the PI controller parameters were tuned such that no overshoot occurred when a joint was forced to traverse its entire workspace. Tuning the PI controller was done by adjusting the proportional gain to achieve maximal performance without overshoot, and then the integral gain was adjusted in the same way. Figure 5.7 shows the results for the PI controllers on each joint responding to a step input causing the joint to traverse a significant portion of its workspace.

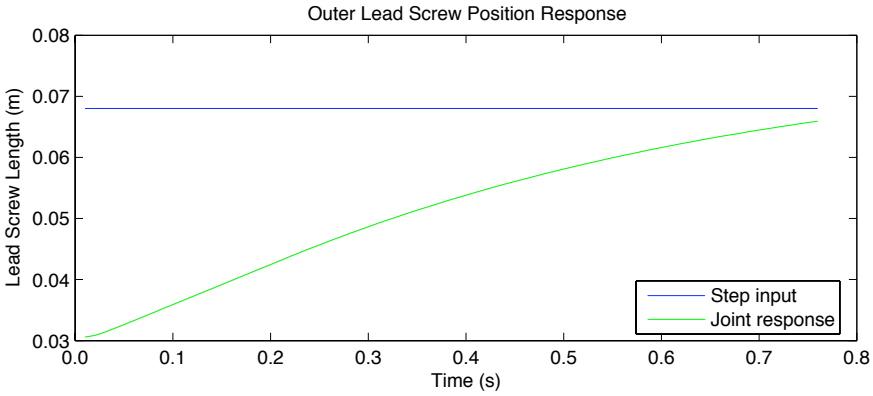
None of the joint position controllers cause any overshoot. The two lead screws, in Figure 5.7(a) and 5.7(b), traverse approximately $3.5cm$ of the $7cm$ workspace without overshoot. Likewise, the hip motor, in Figure 5.7(c) traverses approximately $1rad$ of the workspace without overshoot.

5.3 Revised Electronics

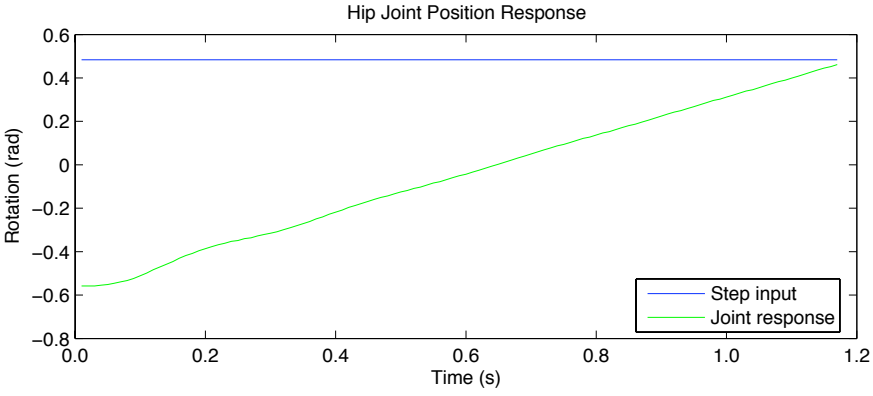
Only a few errors in the electronics of the robot presented themselves. Overall, the electronics worked well without a single DSP 'flashing out'. Two DSPs did however require replacement. One DSP was destroyed by an electrostatic discharge due to improper handling. The other DSP was damaged when the robot stepped on and pierced the power tether. The metal legs and feet conducted electricity and short-circuited the entire robot. Fortunately, only a single DSP was affected. Even though the DSP sockets facilitated the



(a) Inner lead screw response with position controller to step input.



(b) Outer lead screw response with position controller to step input.



(c) Hip response with position controller to step input.

Figure 5.7: Joint responses with position controllers.

replacement of the damaged DSPs, interestingly enough, they actually caused a few problems. The DSP sockets are spring loaded, using the spring force to create a mechanical connection between the DSP pins and traces on the circuit board. Unfortunately, with use and vibration the spring was no longer able to maintain a mechanical connection with all DSP pins. Solving this problem involved bending the pins of the DSP before placing it in the socket.

The only other component that did not function properly was the latching circuitry powering the emergency stop. The logic gates were sensitive enough and fast enough, that rapid changes in the direction of the motors triggered the E-stop. To correct this, the latch circuitry was removed so that E-stops were only triggered when the E-stop limit switches remained depressed. In the future, increased filtering to de-bounce the switch may help to solve the problem.

5.4 Gait

The modest success achieved in implementing Yoneda's algorithm on Hexplorer precludes a comprehensive discussion of the results for two different reasons. The first reason is that the failure of the hip encoder on Leg 5, during its support phase, causes Leg 5 to active its E-Stop. As a result, this prevented any further updates to the kinematic phase ϕ, which in turn caused the remaining legs to wait for a kinematic phase update and incidentally caused these legs to trigger their respective E-Stops. Since the possible fault-tolerance of the gait is beyond the scope of the research, it was not investigated further. The second reason is that the difficulty in controlling the speed of the hip joints caused the body phase to be increased faster than necessary, resulting in a reduced step size and effectively eliminating portions of the support and transfer phases.

The results of the implementation and simulation are presented below. The full simulation of the robot serves as a benchmark for the implementation results. The first set of implementation results discussed are obtained from a rather contrived execution of Yoneda's algorithm for a single leg. The second set of results discussed are obtained by having the robot swim. Swimming refers to operating the robot on a stand such that its legs do not make ground contact at any point in time. This allows the Leg 5 hip encoder to

function properly which in turn permits Yoneda's algorithm to work correctly. Results of both the simulation and implementation are evaluated based on the calculated Cartesian coordinates of Foot 1 with respect to the Leg 1 reference frame.

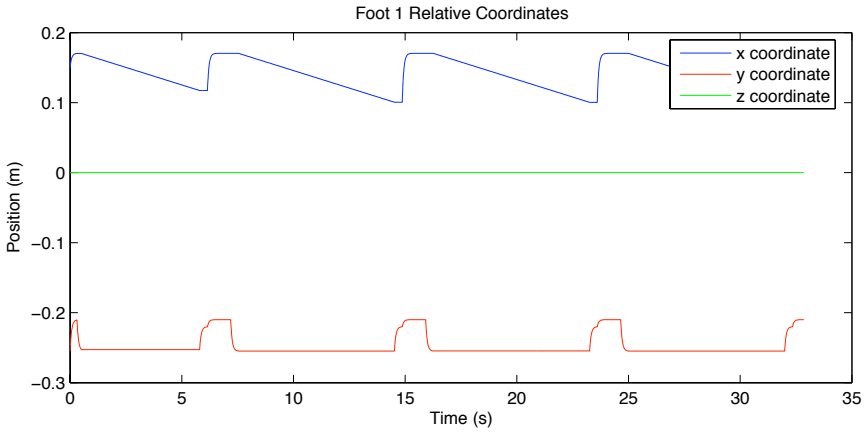
Figure 5.8 shows the simulation results of the robot with respect to three different supervisory inputs.

In Figure 5.8(a), the supervisory inputs are: $v_x = 0.01m/s, v_z = \omega_y = 0$ and the platform is kept $25.5cm$ above the ground. The appropriate lift-off and touch down motions of the foot can be observed from the y coordinate of Figure 5.8(a). The x coordinate of the foot shows a linearly increasing value during the support phase, consistent with the input $v_x = 0.01m/s$. During the transfer phase of the leg, the x coordinate indicates the foothold selection for the next step. Since $v_z = \omega_y = 0$ there is no change in the z coordinate, as shown.

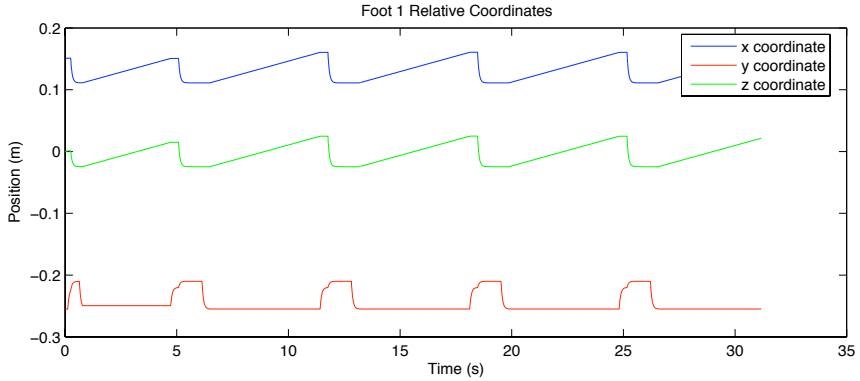
In Figure 5.8(b), the supervisory inputs are: $v_x = v_z = -0.01m/s, \omega_y = 0$ and the platform is kept $25.5cm$ above the ground. The y coordinate of the foot reflects the appropriate lift-off and touch down motions. The motion of the foot in the x and z directions is identical, as it should be.

In Figure 5.8(c), the supervisory inputs are: $v_x = v_z = 0, \omega_y = 0.15rad/s$ and the platform is kept $25.5cm$ above the ground. Since the platform of the robot is yawing, each foot should move through its workspace resulting in a large displacement along the local z axis, and this displacement is reflected in the figure. Changes in the x coordinate reflect that the arc traced by rotating the hip is different than the arc required to achieve a perfect yawing motion.

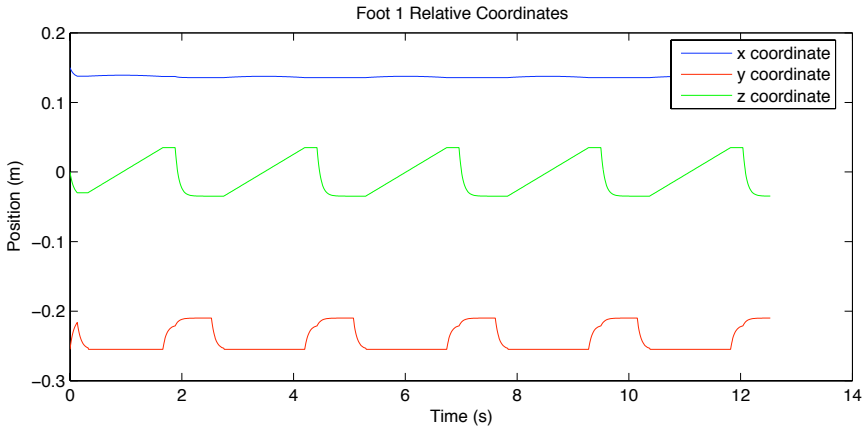
Figure 5.9 shows the experimental results when running only a single leg of Hexplorer using Yoneda's algorithm. In this case, the kinematic phase cannot be incremented by the other legs while the single leg is in its transfer phase. To solve this, upon entering its transfer phase, the kinematic phase is updated by a fixed amount until the next support phase is reached. This partially accounts for the differing values between the simulation and the implementation results. What is important to notice in these results is that the general motion obtained in the simulation is obtained on the leg itself by virtue of the fact that the position curves in Figures 5.8 and 5.9 share a similar shape. The curves resemble each other, but are not identical because of the imprecise control of the hip motor introducing



(a) Simulation of Hexplorer with inputs: $v_x = 0.01m/s, v_z = \omega_y = 0$.



(b) Simulation of Hexplorer with inputs: $v_x = v_z = -0.01m/s, \omega_y = 0$.



(c) Simulation of Hexplorer with inputs: $v_x = v_z = 0, \omega_y = 0.15rad/s$.

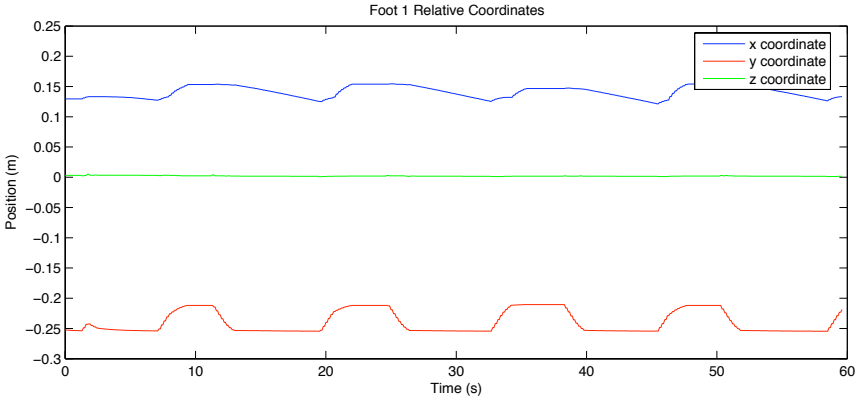
Figure 5.8: Foot 1 position with respect to Leg 1 reference frame in simulation.

error into the temporal kinematic margin calculation that in turn prematurely advances the kinematic phase resulting in smaller steps. The similar shapes formed in both the simulated and actual coordinates indicates the successful implementation of the kinematics and mathematics behind Yoneda's algorithm on the robot itself. Only Figure 5.9(c) bears less correlation to its simulated counterpart than the others. In this test the robot is yawing. By watching the robot execute a constant yaw, it would seem, qualitatively, that the numerical results should follow accordingly. Unfortunately that is not the case, likely because of the large dependence of the yawing motion on successful control of the hip motor.

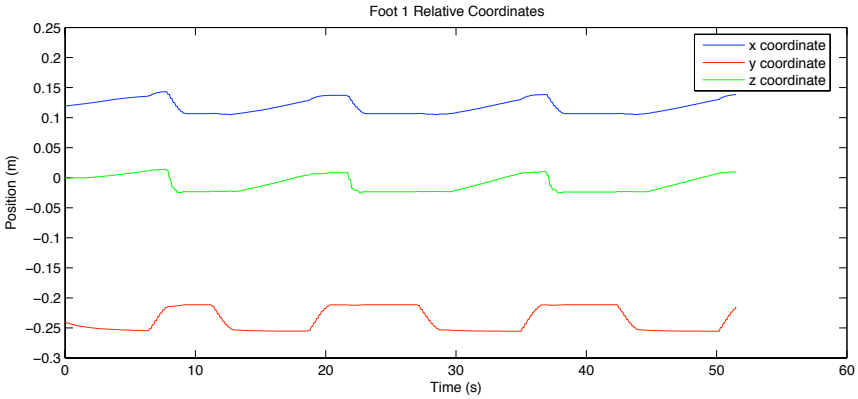
Finally, Figure 5.10 shows the results when all legs are operational and the supervisory inputs are $v_x = 0.01m/s, v_z = \omega_y = 0$ and the platform is kept $25.5cm$ above the ground. The abbreviated nature of the steps is quite evident, especially when viewing the reciprocating nature of the y coordinate. Still, the trajectory does qualitatively reflect that the kinematics and mathematics are functioning correctly and that the discrepancy is largely caused by imperfect control.

5.5 Summary

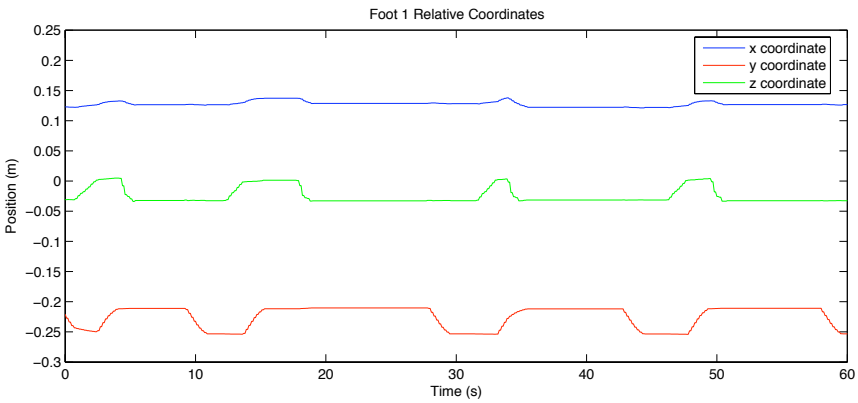
In conclusion, Yoneda's algorithm appears to be well-suited to walking robots with large workspaces and precise motor control. In its present state, Hexplorer is unable to obtain the necessary motor control precision because of problems with the hip encoder design. Using its current hardware and electronics, it is unlikely that Hexplorer will be able to obtain a smooth continuous gait. Instead, a smooth, yet discontinuous gait may be implemented with some success.



(a) Hexplorer output with inputs: $v_x = 0.01m/s, v_z = \omega_y = 0$.



(b) Hexplorer output with inputs: $v_x = v_z = -0.01m/s, \omega_y = 0$.



(c) Hexplorer output with inputs: $v_x = v_z = 0, \omega_y = 0.15rad/s$.

Figure 5.9: Single leg algorithm results: Foot 1 position with respect to Leg 1 reference frame.

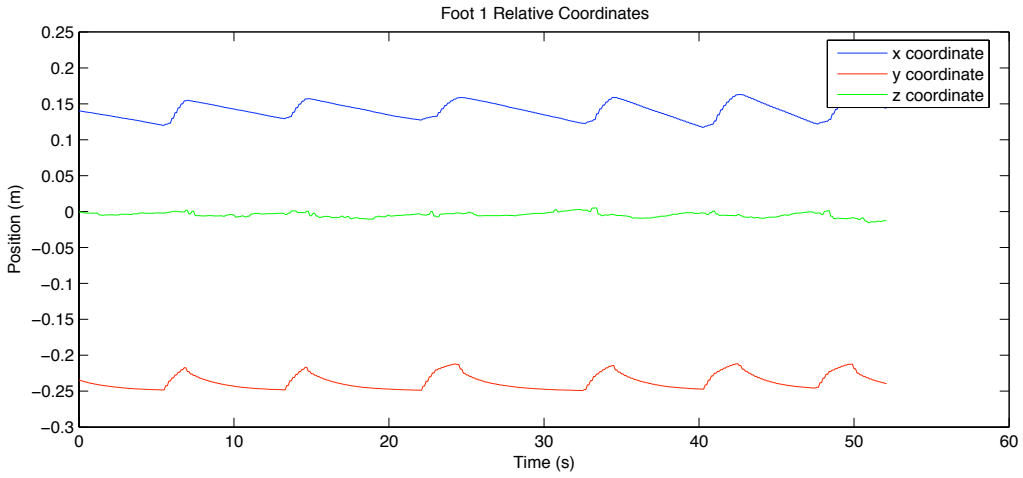


Figure 5.10: Full implementation results: Foot 1 position with respect to Leg 1 reference frame with inputs: $v_x = 0.01m/s, v_z = \omega_y = 0$.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The algorithm introduced by Yoneda has been successfully implemented in simulation and somewhat successfully on Hexplorer. Because of encoder problems caused by the hip joint, Hexplorer cannot take more than a step or two on land. Hexplorer does however, swim in mid-air with Yoneda's algorithm when suspended in the air with some success. It can be said that Hexplorer now swims with a variable duty factor gait that maximizes balance. This fits well into the general autonomous model as supervisory inputs from the autonomous system can be applied directly to the robot that will in turn swim and with some hardware modifications, walk.

Two modifications to the gait algorithm suggested by Yoneda were investigated. The temporal kinematic margin, used to estimate the time until a foot reaches its workspace boundary, was calculated using a higher-order approximation that included the curvature of the path of the foot. In simulation, it reduced the number of steps necessary to travel a certain distance, or rotation, without significantly reducing the balance of the robot, when the yaw rate of the body increased. The increased computational complexity of this technique prevented its implementation on Hexplorer. The other modification involved selecting a foot-hold position. In the proposed method implemented on Hexplorer, the temporal kinematic margin functionality introduced by Yoneda was exploited. The benefits of this method were apparent when the path of the foot traversed the oblong workspace of

the robot.

The architecture of the robot presented unique challenges in terms of kinematic analysis and motion control. The parallel mechanism allows the robot to be balanced when de-energized but at the expense of closed-form solutions to the forward kinematic equations. Due to the number and speed of the DSPs, the DSPs were able to handle the rational number computations required by an iterative Newton-Raphson technique to solve the forward kinematics. In fact, sampling rates of 100Hz for the joint controllers and 10Hz for the gait controllers were achieved. Implementation of the controllers, however, was complicated due to the poor resolution of the hip encoder. Strong filters and feed-forward terms, based on open-loop experiments, were used to deal with the deficiency of the hip encoder.

In conclusion, although the goal to *make Hexplorer walk* was not entirely achieved, Hexplorer successfully executed Yoneda's algorithm using a distributed DSP architecture when raised above the floor.

6.2 Future Work

With Yoneda's algorithm fully implemented on Hexplorer, and a few mechanical corrections, it can continue its journey to one day walk and be endowed with full autonomy. Proceeding with such a challenge will require improvements to mechanical, electrical, and computing aspects of the robot.

Mechanically, the legs of the robot are well built and allow the robot to be passively balanced. The heave provided by the legs, however, is relatively small considering the overall size of the robot. If the future intentions involve rugged terrain where ground clearance of transfer feet is important, then the leg may need to be redesigned.

The hip connection between the leg and the body of the robot, however, must be redesigned. The press-fit bearings and washers should be replaced with thrust bearings and supports. Supports added to the top of the top ring and bottom of the bottom ring would hold the thrust bearings in place while allowing the hip to rotate freely. The location of the bottom support would require that the hip encoder be replaced or relocated, which is actually required.

The current location of the hip encoder, is also inadequate. As shown in Chapters 2 and 5, having the encoder measure the output shaft of the gearbox reduces its resolution, increasing the difficulty of control. Replacing the hip motors with ones that have a built-in encoder, like the lead screw motors would solve this problem.

In terms of the forward kinematics of the robot, two other techniques could be investigated. A lookup table containing predetermined mappings between joint and Cartesian coordinates combined with linear interpolation could be used to solve the forward kinematics very quickly. However, such a table listing n joint positions for a lead screw, would require a table of n^2 entries to determine the full forward kinematics of the leg. Limiting the size of the table or storing the table on a EPROM would require further investigation. A redundant encoder or potentiometer could also be used to measure the angle θ_1 or θ_2, resulting in a closed-form solution to the kinematics.

Depending on the future purpose of the robot, an electronic hardware revision may prove beneficial. The existing distributed processing architecture has its advantages. Parallel processing increases the computational power available to the robot. Each leg is able to make its own low-level decisions regarding its state. These decisions could correct small errors or minor unexpected environmental interactions that would otherwise unnecessarily interrupt higher priority issues being dealt with by the brain. For example, if a foot encounters an obstacle, the leg could activate an obstacle avoidance algorithm without disturbing a computationally-intensive localization algorithm being executed by the brain.

However, without being sensor-laden and programmed with such high-level cognitive abilities, Hexplorer does not have any higher priority tasks with which to deal. Instead of being beneficial, the additional complexity created by the distributed architecture unnecessarily confuses and complicates debugging tasks. By migrating to a new design based on a single processor, debugging source code would be greatly simplified. Cost could also be reduced or money that would have otherwise been used to purchase the remaining five DSPs could instead be used to upgrade the single DSP to a floating-point capable model. Having a floating-point capable DSP would improve the portability of the source code and accuracy of the arithmetic. Portability would be improved because the same source code could be written, compiled, and executed on a PC as well as the floating-point DSP. As mentioned in Chapter 4, the fixed-point DSP required special libraries to perform fast

emulated floating-point operations. Since these libraries are not readily available for PC compiled software, portability is limited. Accuracy of the arithmetic would be improved because fixed-point numeric representation would no longer be necessary. If a new electronic hardware revision is undertaken, a single floating-point DSP architecture should be considered because of simpler debugging, improved source code portability, and numerical accuracy.

The future direction of research on Hexplorer may include rugged terrain or closed-loop supervisory control. In either case, the mechanical and electrical improvements suggested above will help to achieve future research goals.

References

[1] M.B. Binnard. *Design of a small pneumatic walking robot.* Master's thesis, Massachusetts Institute of Technology, 1995.

[2] W.D. Bishop, R.B. Gorbet, C.C.W. Hulls, and W. Loucks. *ECE 324 and ECE 325 Microprocessor Systems and Interfacing Lecture Notes.* University of Waterloo, Waterloo, Ontario, 2004.

[3] BlueRadios, Inc. *Bluetooth®Module: BR-C11 Class1*, 2005.

[4] I. Chen, S.H. Yeo, and Y. Gao. Locomotive gait generation for inchworm-like robots using finite state approach. *Robotica*, 19(5):535–542, 2001.

[5] S. Chitta, F. W. Heger, and V. Kumar. Design and gait control of a rollerblading robot. In *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, pages 3944–3949, New Orleans, Louisiana, April 26 - May 1, 2004.

[6] C. Collins and D. Orr. *Intelligent Motion Control Using The TMS320LF2407 Applied To A Six-Legged Walking Robot.* Systems Design Engineering 4th Year Undergraduate Workshop Project, University of Waterloo, 2001.

[7] J.J. Craig. *Introduction to Robotics: Mechanics and Control.* Pearson Prentice Hall, Upper Saddle River, New Jersey, third edition, 2005.

[8] S. Cutler and R. Tien. *Hexplorer: Hexplorations in Walking (H^2IW).* Systems Design Engineering 4th Year Undergraduate Workshop Project, University of Waterloo, 2005.

[9] N. Cristello and D. Kwok. *How 2 Hex*. Systems Design Engineering 4th Year Undergraduate Workshop Project, University of Waterloo, 2003.

[10] M. R. Fielding, R. Dunlop, and C. J. Damaren. Hamlet: Force/position controlled hexapod walker - design and systems. In *Proceedings of the 2001 IEEE International Conference on Control Applications*, pages 984–989, Mexico City, Mexico, September 5 - 7, 2001.

[11] G.F. Franklin, J.D. Powell, and M. Workman. *Digital Control of Dynamic Systems*. Addison-Wesley, Menlo Park, California, third edition, 1998.

[12] J. Garcia de Jalon and E. Bayo. *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge*. Springer-Verlag, 1994.

[13] C.F. Gerald and P.O. Wheatley. *Applied Numerical Analysis*. Addison-Wesley, Boston, Massachusetts, seventh edition, 2003.

[14] J.H. Ginsberg. *Advanced Engineering Dynamics*. Cambridge University Press, Cambridge, second edition, 1998.

[15] Y. Go, X. Yin, , and A. Bowling. A navigable six-legged robot platform. In *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, pages 5105-5110, New Orleans, Louisiana, April 26 - May 1, 2004.

[16] D Golubovic and H. Hu. An interactive software environment for gait generation and control design of sony legged robots. In *Proceedings of the 6th International Symposium on RoboCup*, Fukuoka, Japan, June 24 - 25, 2002.

[17] M. Goulet. *Hexapode : Developpement mecatronique d'un robot marcheur*. Master's thesis, Universite Laval, 2006.

[18] E.J. Haug. *Computer-Aided Kinematics & Dynamics of Mechanical Systems - Volume 1: Basic Methods*. Allyn and Bacon, Needham Heights, Massachusetts, 1989.

[19] G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata. Autonomous evolution of gaits with the sony quadruped robot. In *Proceedings of the Genetic and*

Evolutionary Computation Conference, pages 1297–1304, Orlando, Florida, July 13 - 17, 1999.

[20] D. Horvath, J. Lee, S. Williams. *Hexotica: Design and Implementation of a Small Walking Robot*. Systems Design Engineering 4th Year Undergraduate Workshop Project, University of Waterloo, 1997.

[21] K. Iagnemma, A. Rzepniewski, S. Dubowsky, P. Pirjanian, T. Huntsberger, and P. Schenker. Mobile robot kinematic reconfigurability for rough terrain. In *Proceedings of the Sensor Fusion and Decentralized Control in Robotic Systems III Conference*, pages 413–420, Boston, Massachusetts, November 6 - 8, 2000.

[22] X. Kong and C. M. Gosselin. Forward displacement analysis of third-class analytic 3-RPR planar parallel manipulators. *Mechanism and Machine Theory*, 36(9):1009–1018, 2001.

[23] R. Kurazume, K. Yoneda, and S. Hirose. Feedforward and feedback dynamic trot gait control for quadruped walking vehicle. *Autonomous Robots*, 12(2):157–172, 2002.

[24] A. Kwan and E. So. *Hexplorer: Hex to Walk*. Systems Design Engineering 4th Year Undergraduate Workshop Project, University of Waterloo, 2004.

[25] Lantronix, Inc. *XPort Data Sheet*, 2005.

[26] K. Larochelle, S. Dashnaw, and G. Parker. Gait evolution for a hexapod robot. In *Proceedings of the Fourth International Symposium on Soft Computing and Intelligent Systems for Industry*, Paisley, Scotland, June 26 - 29, 2001.

[27] National Semiconductor. *Linear and Switching Voltage Regulator Fundamentals*, 2006.

[28] W. J Lee and D. E. Orin. Omnidirectional supervisory control of a multilegged vehicle using periodic gaits. *IEEE Journal of Robotics and Automation*, 4(6):635–642, 1988.

[29] M. Lewis, A. Fagg, and G. Bekey. Genetic algorithms for gait synthesis in a hexapod robot. In *Recent Trends in Mobile Robots*, World Scientific, 1993.

[30] R. McGhee. Some finite state aspects of legged locomotion. *Mathematical Biosciences*, 2:67–84, 1968.

[31] D. McMillen. *Kafka: A hexapod robot*. Master's thesis, University of Toronto, 1995.

[32] G. Parker, D. Braun, and I. Cyliax. Evolving hexapod gaits using a cyclic genetic algorithm. In *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*, pages 141–144, Banff, July 27 - August 1, 1997.

[33] PNI Corporation. *MicroMag3: 3-Axis Magnetic Sensor Module*, 2005.

[34] J.M. Porta and E. Celaya. Efficient gait generation using reinforcement learning. In *Proceedings of the 4th International Conference on Climbing and Walking Robots*, pages 411–418, Karlsruhe, Germany, September 24 - 26, 2001.

[35] Robert Bosch GmbH. *CAN Specification 2.0*, 1991.

[36] U. Saranli, M. Buehler, and D. E. Koditschek. Rhex – a simple and highly mobile hexapod robot. *The International Journal of Robotics Research*, 20(7):616–631, 2001.

[37] S.V. Shastri. A biologically consistent model of legged locomotion gaits. *Biological Cybernetics*, 76:429–440, 1996.

[38] C.L. Shih and C. A. Klein. An adaptive gait for legged walking machines over rough terrain. *IEEE Transactions on Systems, Man and Cybernetics*, 23(4):1150–1155, 1993.

[39] R. Siegwart and I. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, Cambridge, Massachusetts, 2004.

[40] S. M. Song and B. S. Choi. The optimally stable ranges of 2n-legged wave gaits. *IEEE Transactions on Systems, Man and Cybernetics*, 20(4):888–902, 1990.

[41] S.M. Song and K.J. Waldron. *Machines That Walk: The Adaptive Suspension Vehicle*. The MIT Press, Cambridge, Massachusetts, 1989.

[42] Spectrum Digital Incorporated. *eZdsp^{TM} F2812 Technical Reference*, 2003.

[43] Texas Instruments. *IQmath Library: A Virtual Floating Point Engine*, 2002.

[44] Texas Instruments. *TPS79633 Ultralow-Noise, High PSRR, Fast, RF, 1A, Low-Dropout Linear Regulator Datasheet*, 2004.

[45] Texas Instruments. *PTN78020W 6-A Wide-Input Voltage Adjustable Switching Regulator Datasheet*, 2006.

[46] US Digital. *E2 Optical Encoder Kit*, 2006.

[47] S.T. Venkataraman. A model of legged locomotion gaits. In *Proceedings of the 1996 IEEE International Conference on Robotics & Automation*, pages 3545–3550, Minneapolis, Minnesota, April 22 - 28, 1996.

[48] C. Villard, P. Gorce, J.G. Fontaine, and J. Rabit. Ralphy: A dynamic study of a quadruped robot. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 106–111, Le Touquet, France, October 17 - 20, 1993.

[49] J. Yang and J. Kim. A strategy of optimal fault tolerant gait for the hexapod robot in crab walking. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, pages 1695–1700, Leuven, Belgium, May 16 - 20, 1998.

[50] T. Yee. *Gait Planning and Transitions of Walking Robots on Smooth and Rough Terrains*. PhD thesis, University of Illinois, 2003.

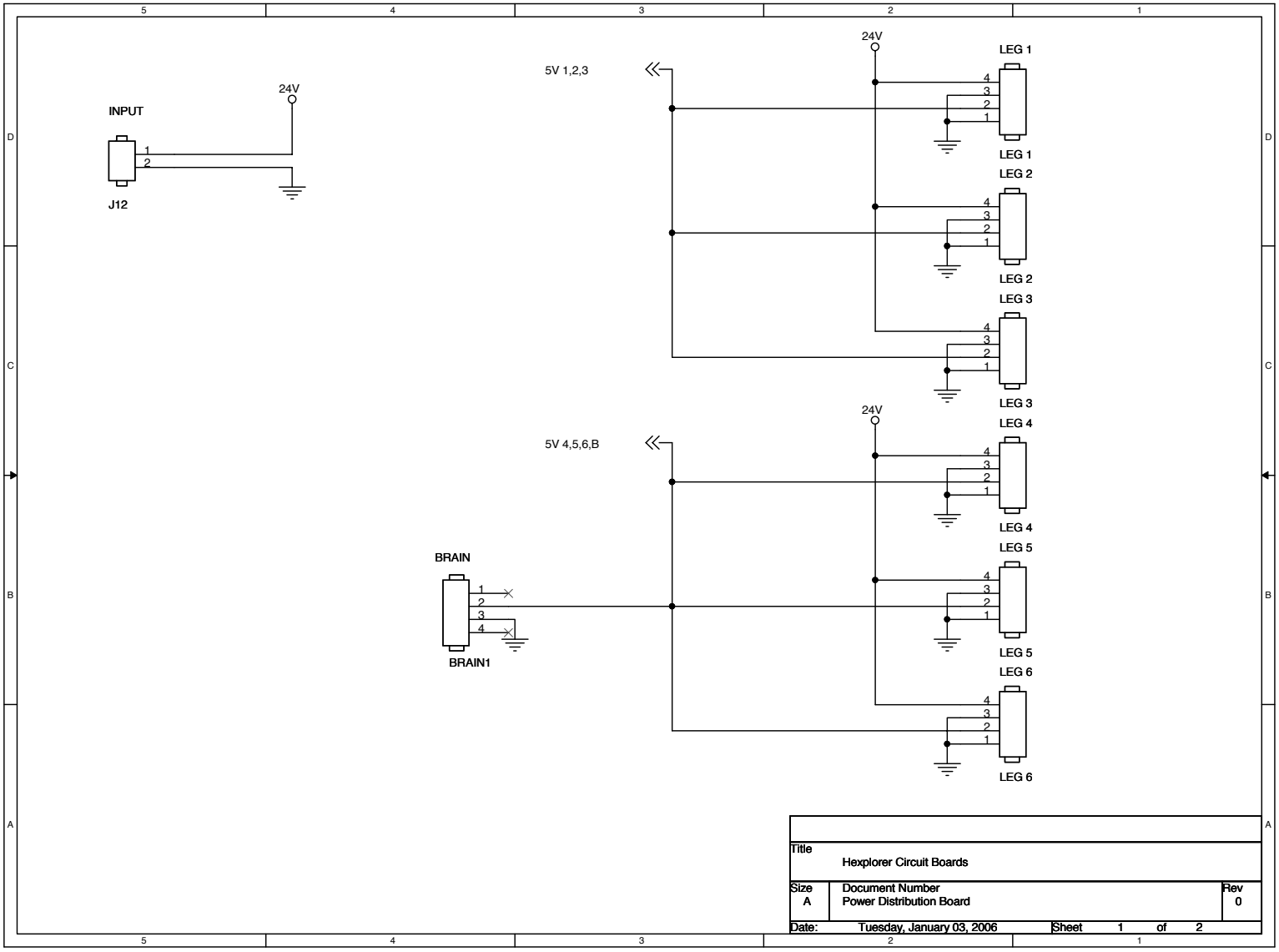
[51] K. Yoneda and S. Hirose. Dynamic and static fusion gait of a quadruped walking vehicle on a winding path. *Advanced Robotics*, 9(2):125–136, 1995.

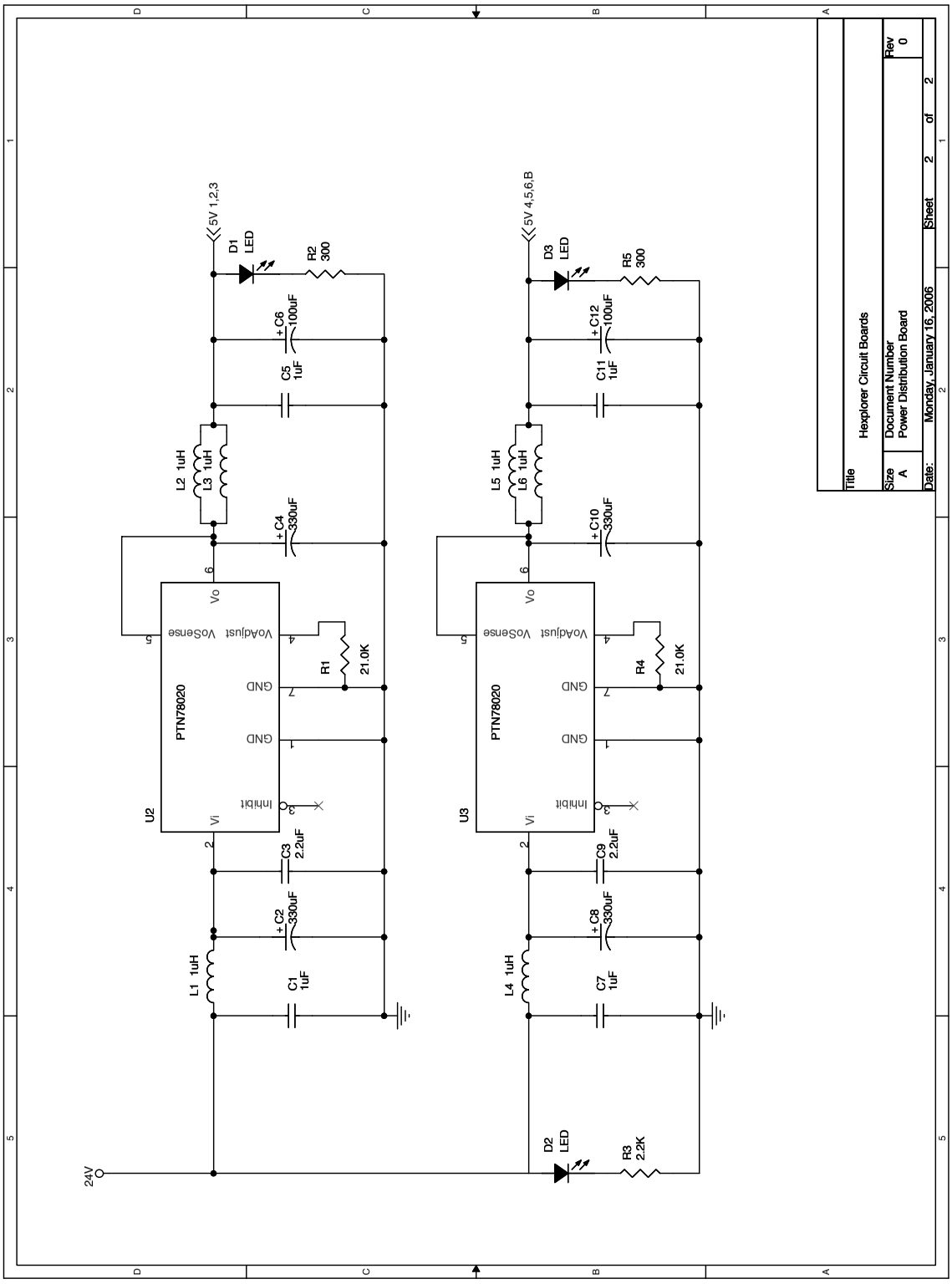
[52] K. Yoneda, K. Suzuki, and Y. Kanayama. Gait planning for versatile motion of a six legged robot. In *Proceedings of the 1994 IEEE International Conference on Robotics & Automation*, pages 1338–1343, New Orleans, Louisiana, April 26 - May 1, 2004.

[53] K. Yoneda, K. Suzuki, Y. Kanayama, H. Takahashi, and J. Akizono. Gait and foot trajectory planning for versatile motions of a six-legged robot. *Journal of Robotic Systems*, 14(2):121–133, 1997.

Appendix A

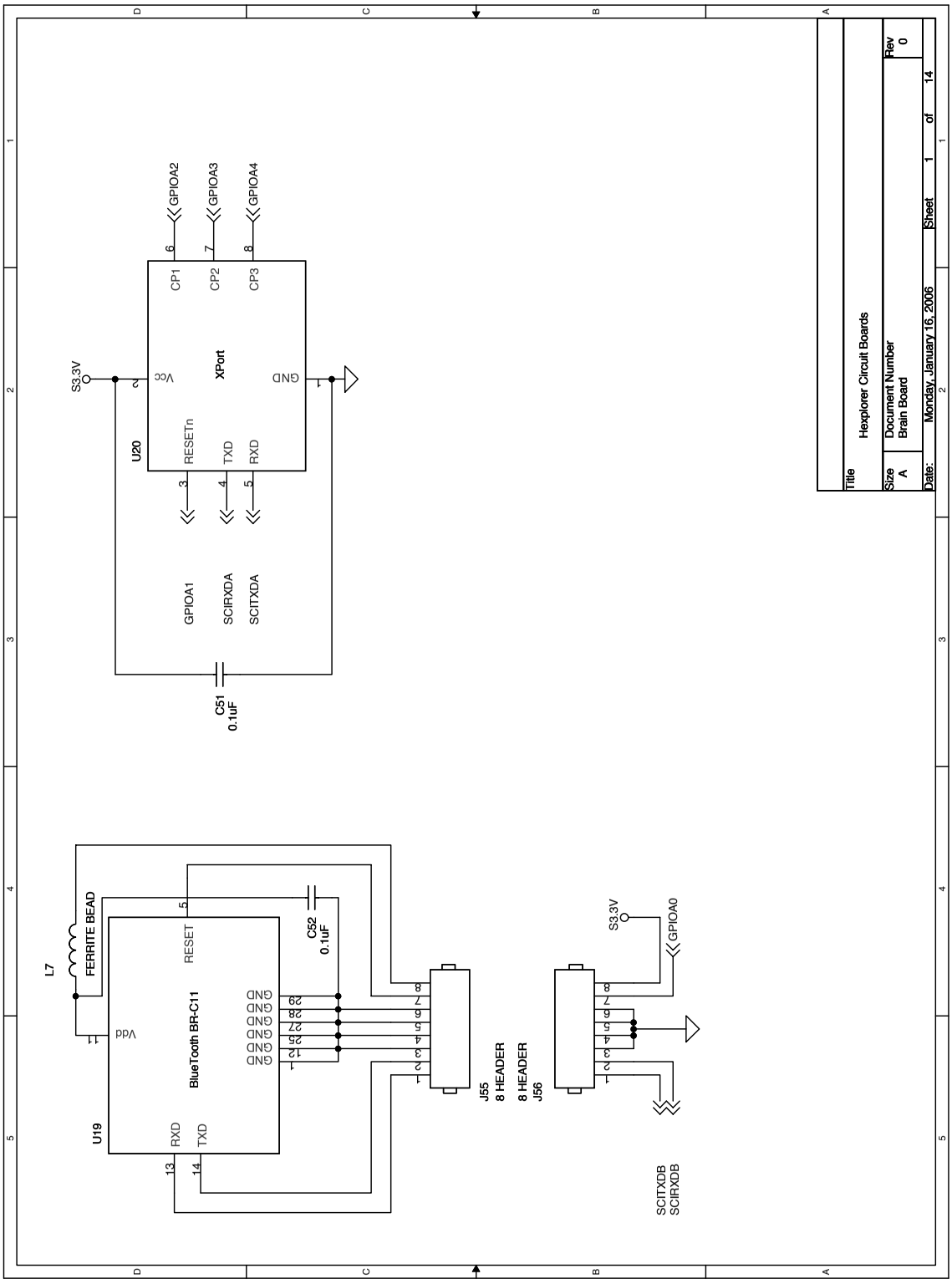
Power Distribution Board Schematics

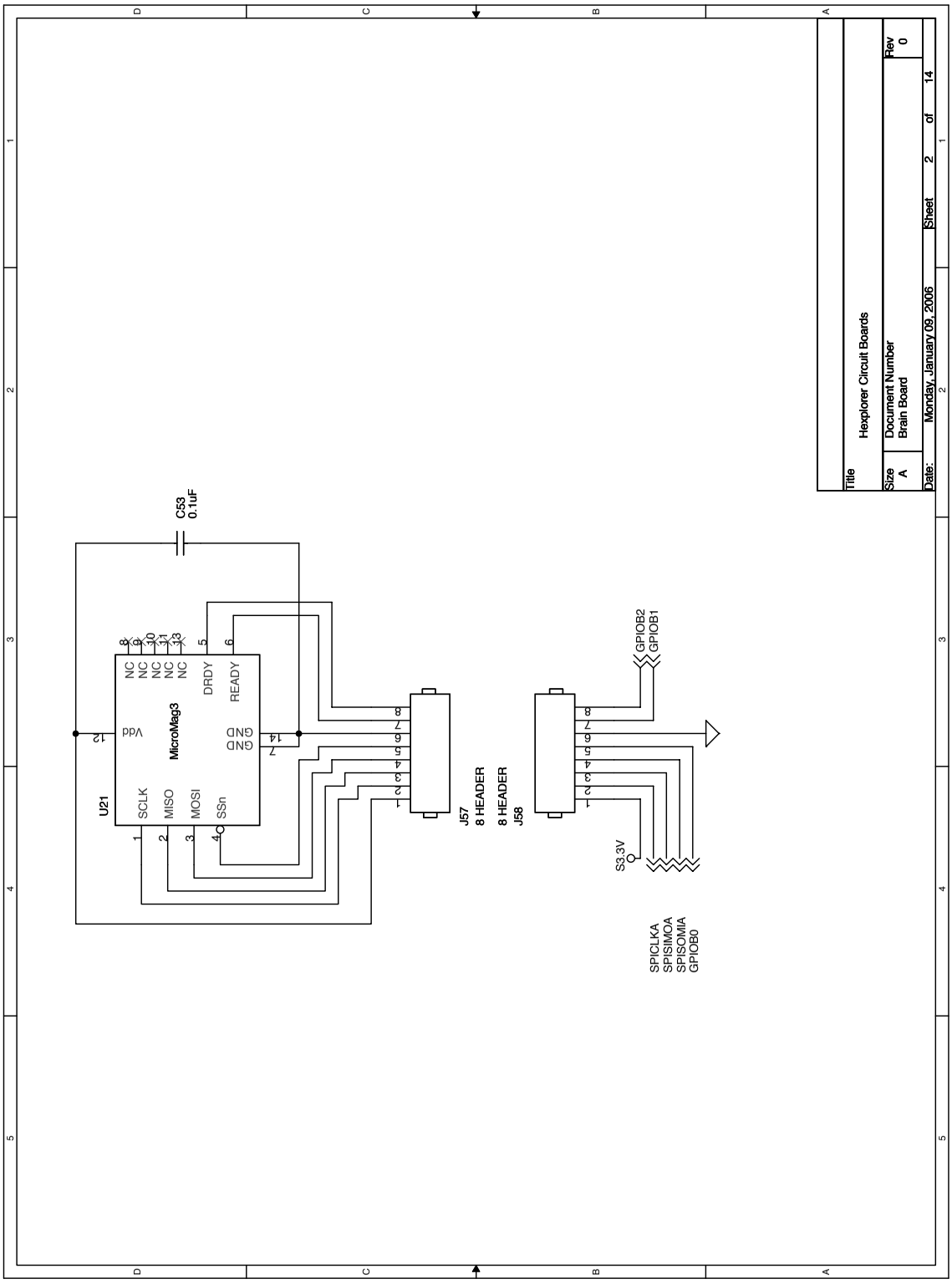

INPUT
24V
1
2
J12
5V 1,2,3
24V
LEG 1
4
3
2
1
LEG 1
LEG 2
LEG 2
LEG 3
LEG 3
5V 4,5,6,B
24V
LEG 4
LEG 4
BRAIN
1
2
3
4
BRAIN1
LEG 5
LEG 5
LEG 6
LEG 6
Title
Hexplorer Circuit Boards
Size
A
Document Number
Power Distribution Board
Rev
0
Date:
Tuesday, January 03, 2006
Sheet 1 of 2

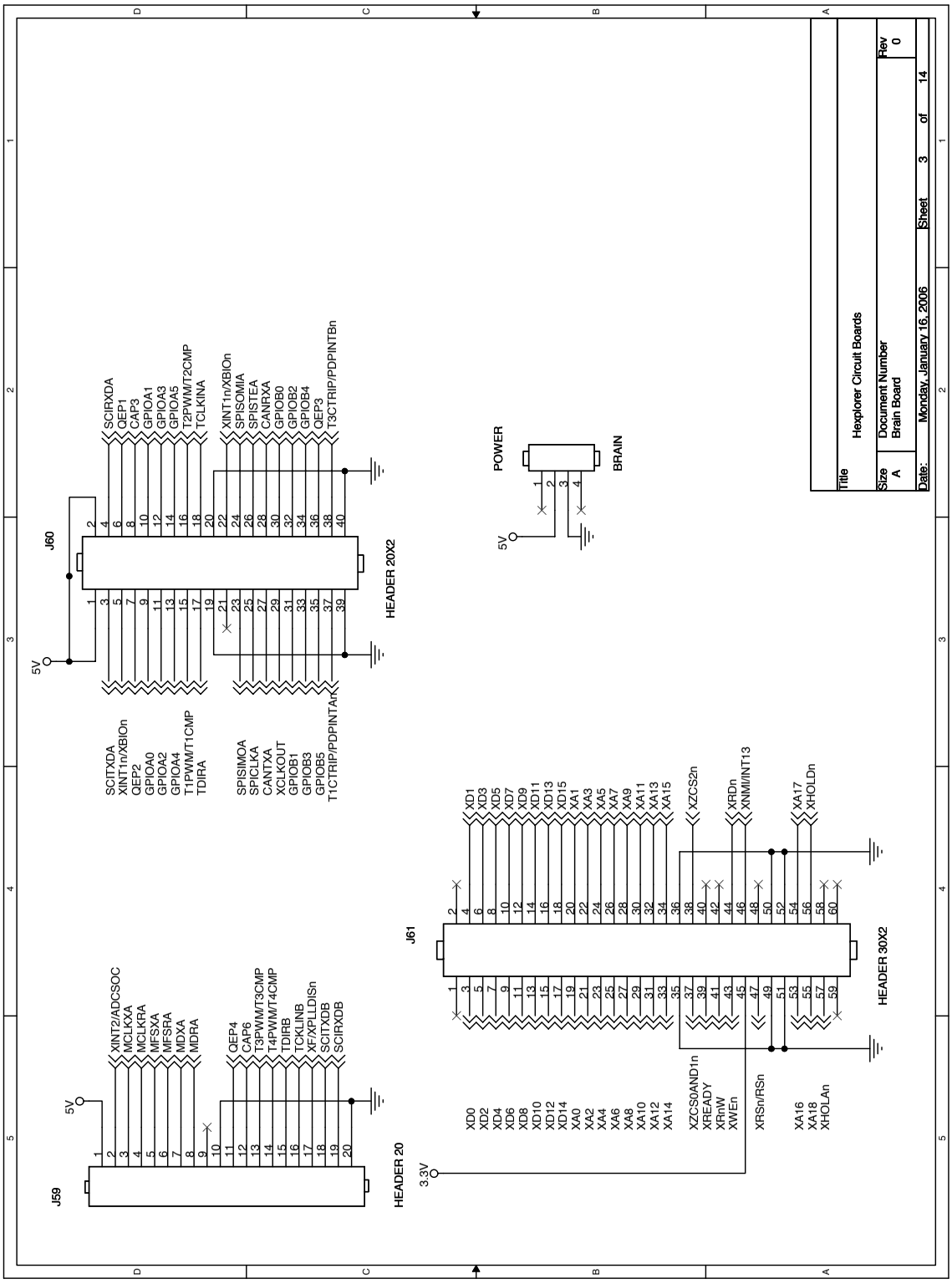

5V 1,2,3
D1
LED
R2
300
C6
100uF
C5
1uF
L2 1uH
L3 1uH
C4
330uF
U2
PTN78020
Vo
VoAdjust
VoSense
GND
GND
Inhibit
Vi
R1
21.0K
C3
2.2uF
C2
330uF
L1 1uH
C1
1uF
5V 4,5,6,B
D3
LED
R5
300
C12
100uF
C11
1uF
L5 1uH
L6 1uH
C10
330uF
U3
PTN78020
R4
21.0K
C9
2.2uF
C8
330uF
L4 1uH
C7
1uF
24V
D2
LED
R3
2.2K
Title
Hexplorer Circuit Boards
Size
A
Document Number
Power Distribution Board
Rev
0
Date:
Monday, January 16, 2006
Sheet
2
of
2

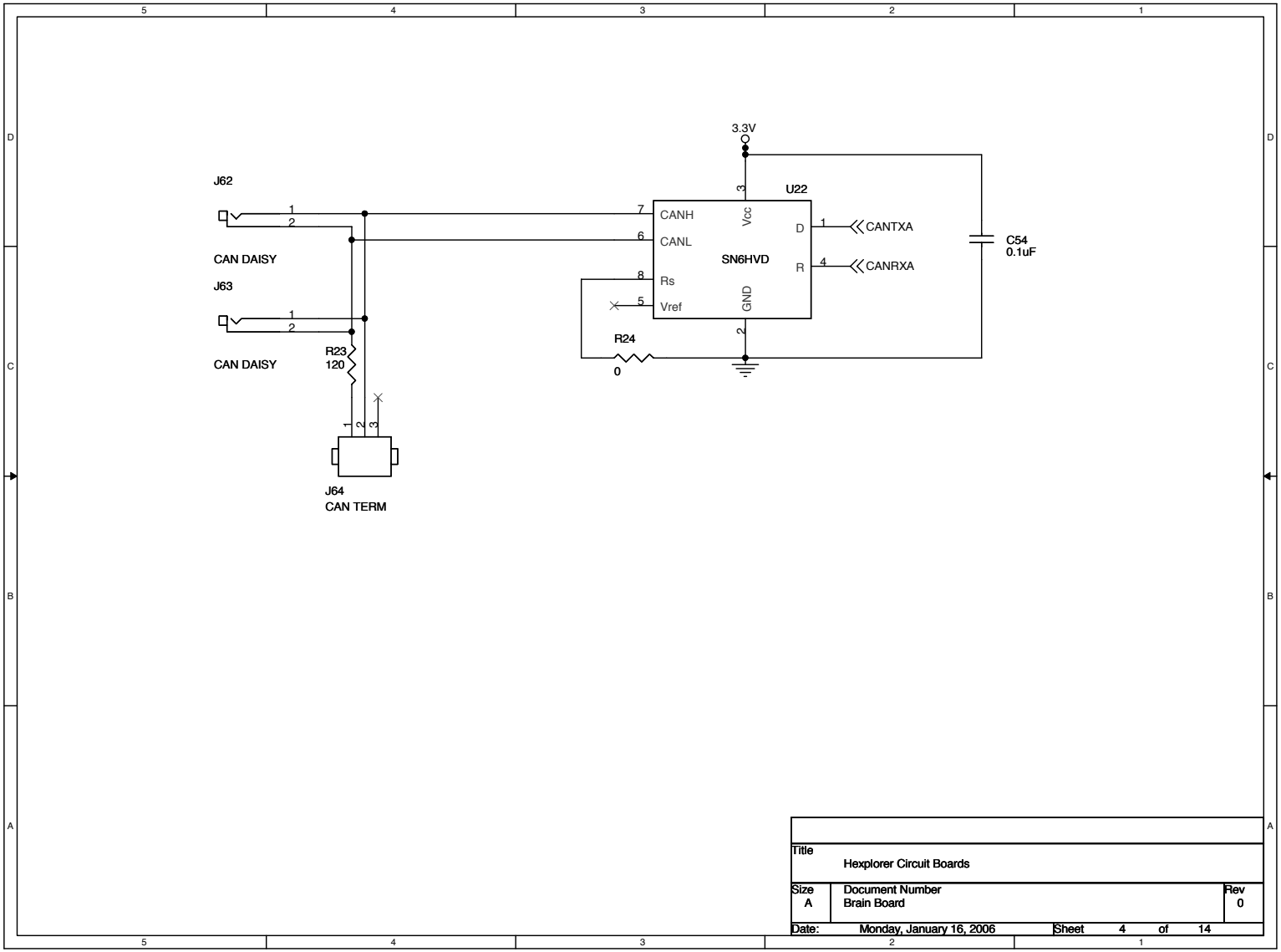
Appendix B

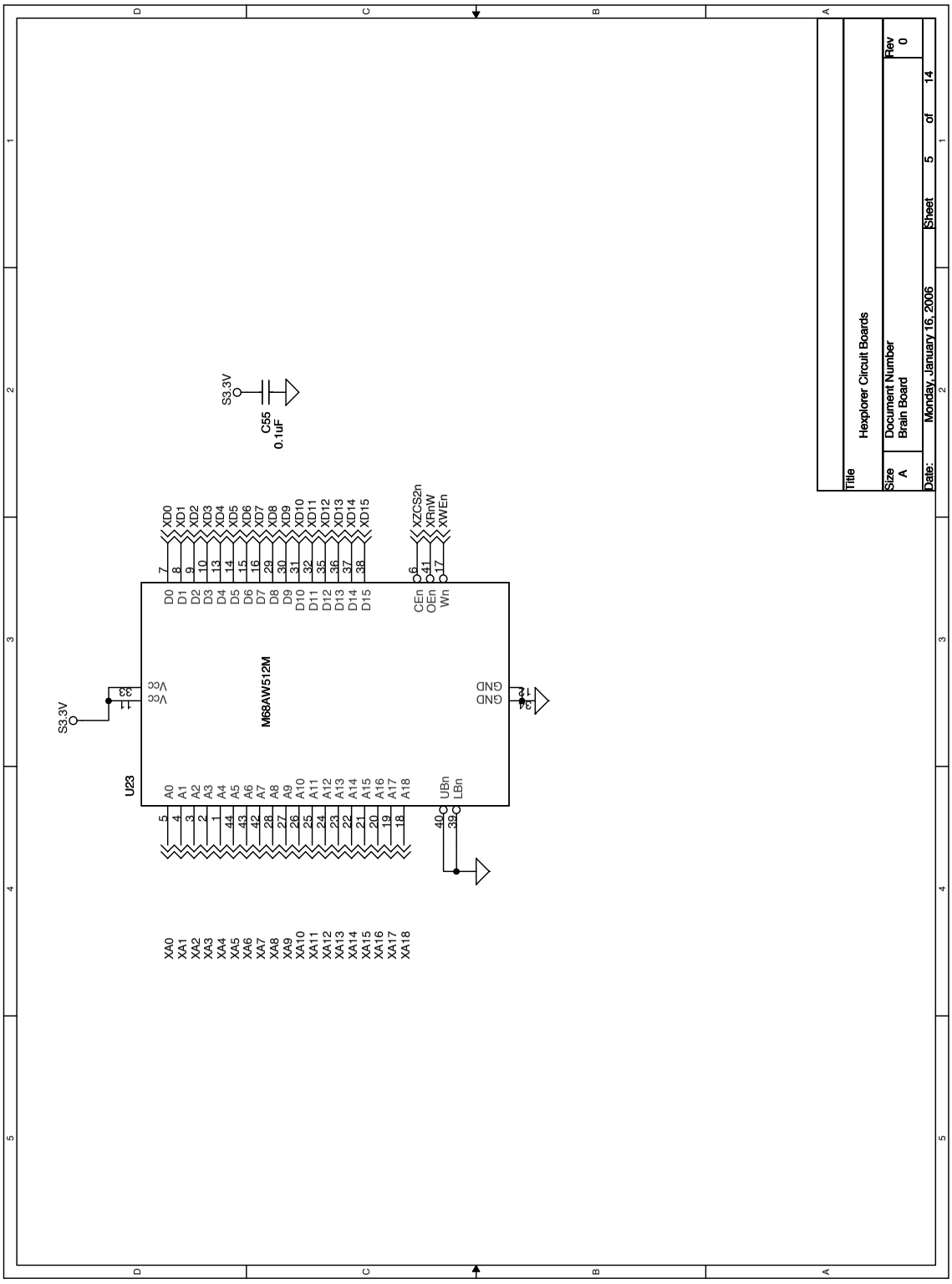
Brain Daughter Board Schematics

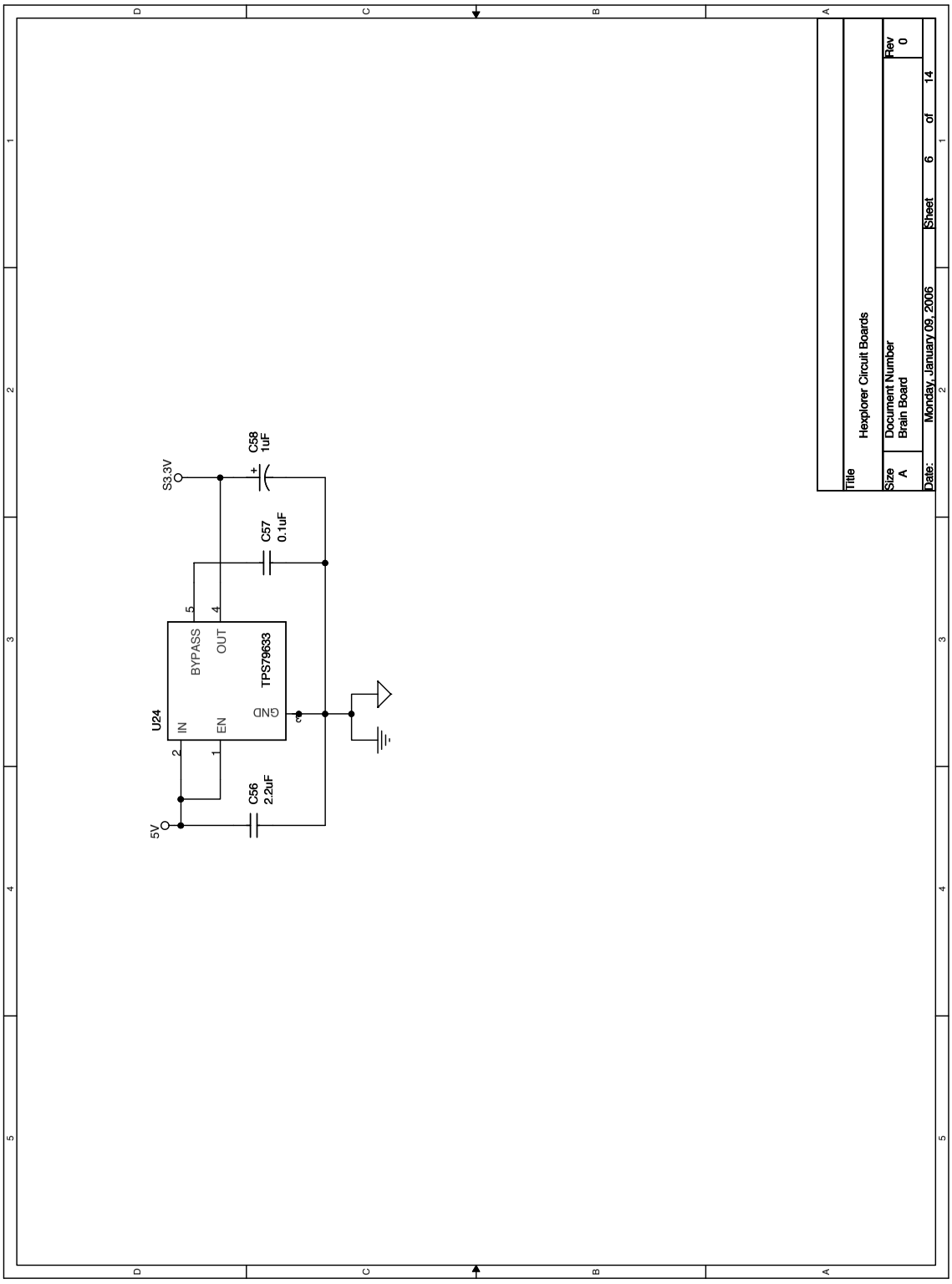

Title
Hexplorer Circuit Boards
Size
A
Document Number
Brain Board
Rev
0
Date:
Monday, January 16, 2006
Sheet
1
of
14
U20
XPort
CP1
CP2
CP3
GND
RESETn
TXD
RXD
GPIOA1
SCIRXDA
SCITXDA
GPIOA2
GPIOA3
GPIOA4
S3.3V
C51
0.1uF
L7
FERRITE BEAD
U19
BlueTooth BR-C11
RESET
Vdd
RXD
TXD
C52
0.1uF
J55
8 HEADER
J56
8 HEADER
GPIOA0
SCITXDB
SCIRXDB


Title
Hexplorer Circuit Boards
Size
A
Document Number
Brain Board
Rev
0
Date:
Monday, January 09, 2006
Sheet
2
of
14
C53
0.1uF
U21
MicroMag3
NC
NC
NC
NC
DRDY
READY
Vdd
GND
GND
SCLK
MISO
MOSI
SSn
J57
8 HEADER
J58
8 HEADER
GPIOB2
GPIOB1
S3.3V
SPICLKA
SPISIMOA
SPISOMIA
GPIOB0


J59
HEADER 20
J60
HEADER 20X2
J61
HEADER 30X2
POWER
BRAIN
5V
3.3V
Title
Hexplorer Circuit Boards
Size
A
Document Number
Brain Board
Rev
0
Date:
Monday, January 16, 2006
Sheet
3
of
14

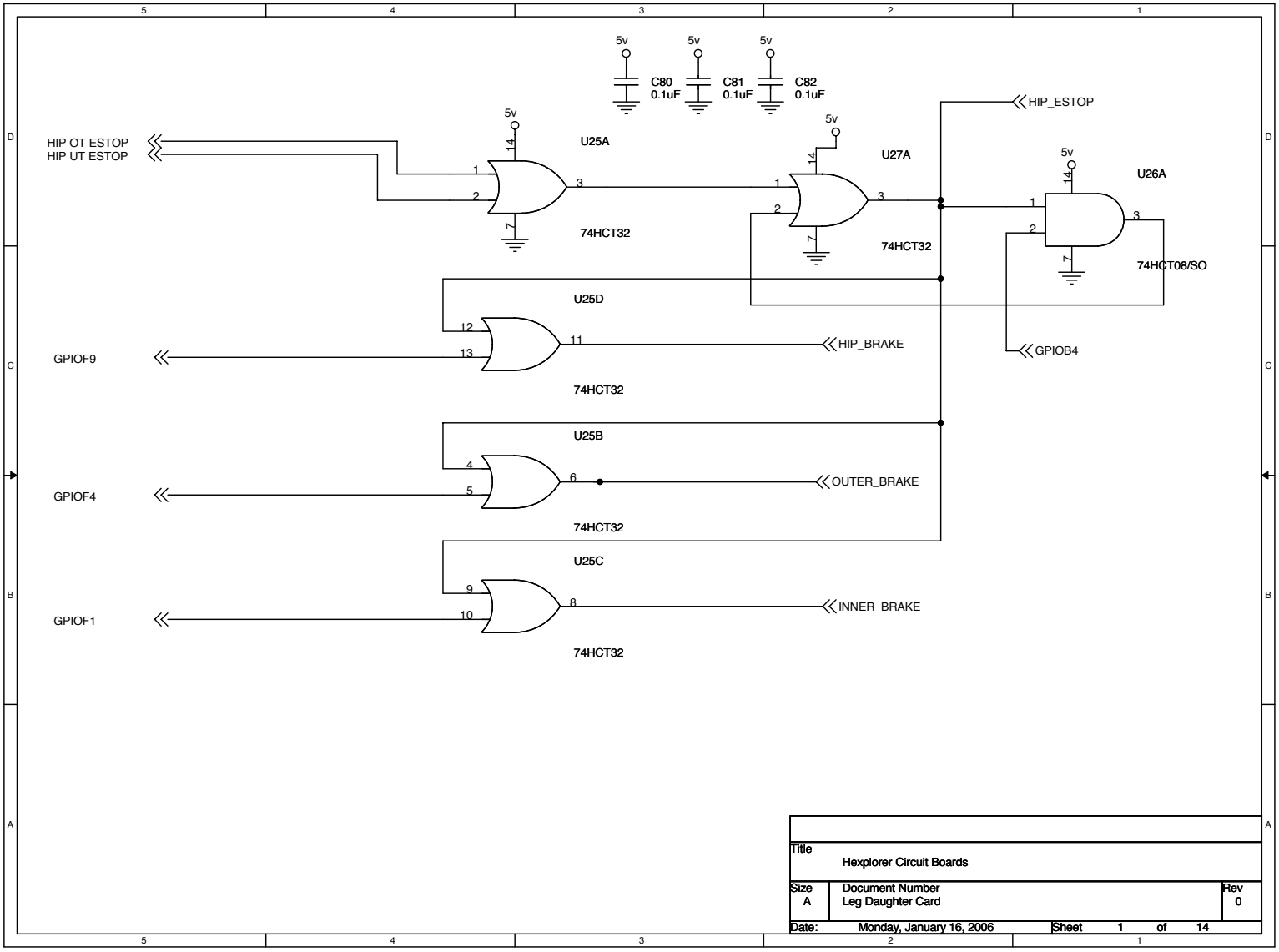

3.3V
J62
CAN DAISY
J63
CAN DAISY
R23
120
J64
CAN TERM
U22
CANH
CANL
Rs
Vref
Vcc
GND
D
R
SN6HVD
CANTXA
CANRXA
C54
0.1uF
R24
0
Title
Hexplorer Circuit Boards
Size
A
Document Number
Brain Board
Rev
0
Date:
Monday, January 16, 2006
Sheet 4 of 14

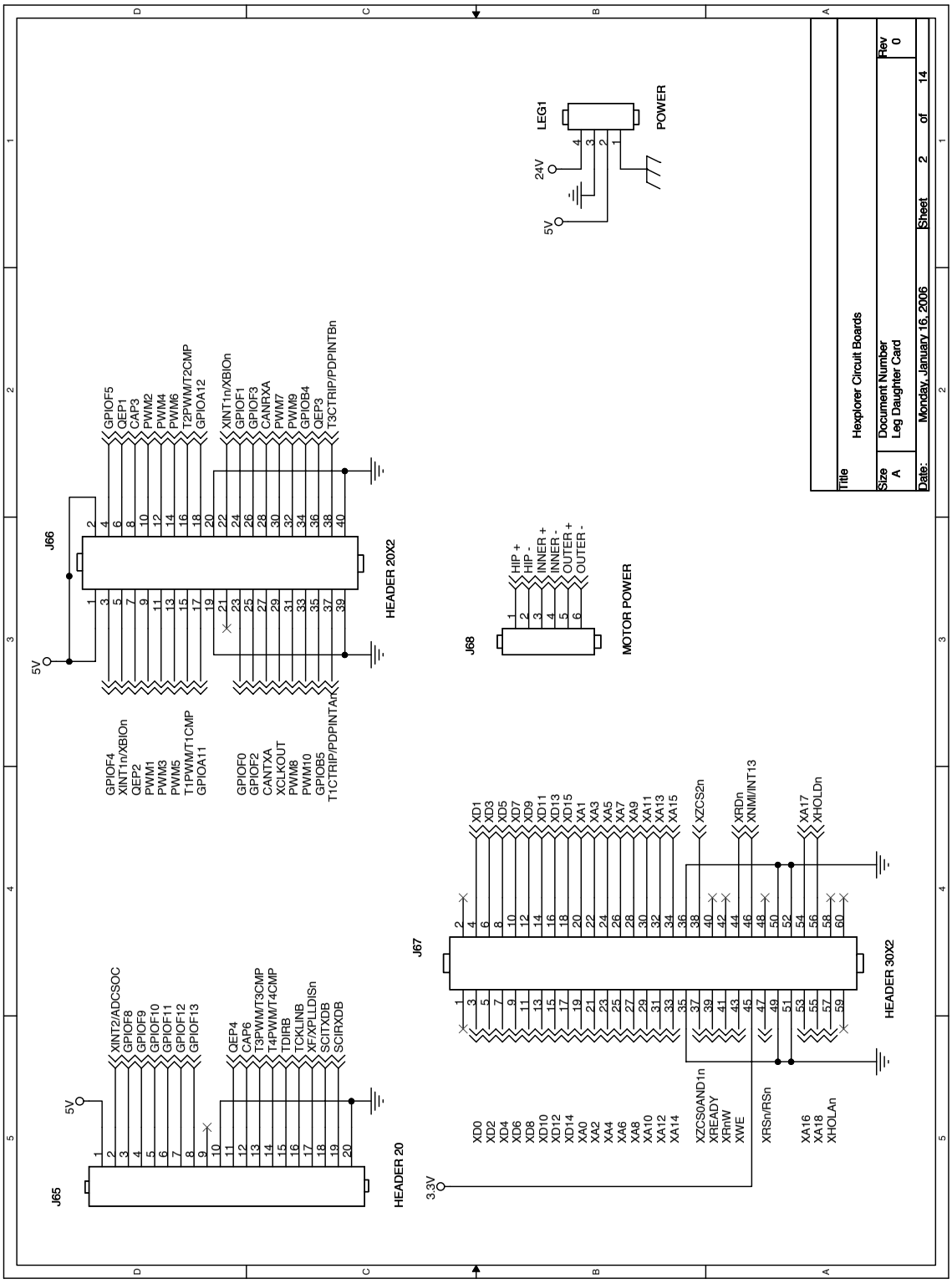

Title
Hexplorer Circuit Boards
Size
A
Document Number
Brain Board
Rev
0
Date:
Monday, January 16, 2006
Sheet
5
of
14
S3.3V
C55
0.1uF
U23
M68AW512M
VCC
GND
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
CEn
OEn
Wn
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
UBn
LBn
XD0
XD1
XD2
XD3
XD4
XD5
XD6
XD7
XD8
XD9
XD10
XD11
XD12
XD13
XD14
XD15
XZCS2n
XRnW
XWEn
XA0
XA1
XA2
XA3
XA4
XA5
XA6
XA7
XA8
XA9
XA10
XA11
XA12
XA13
XA14
XA15
XA16
XA17
XA18

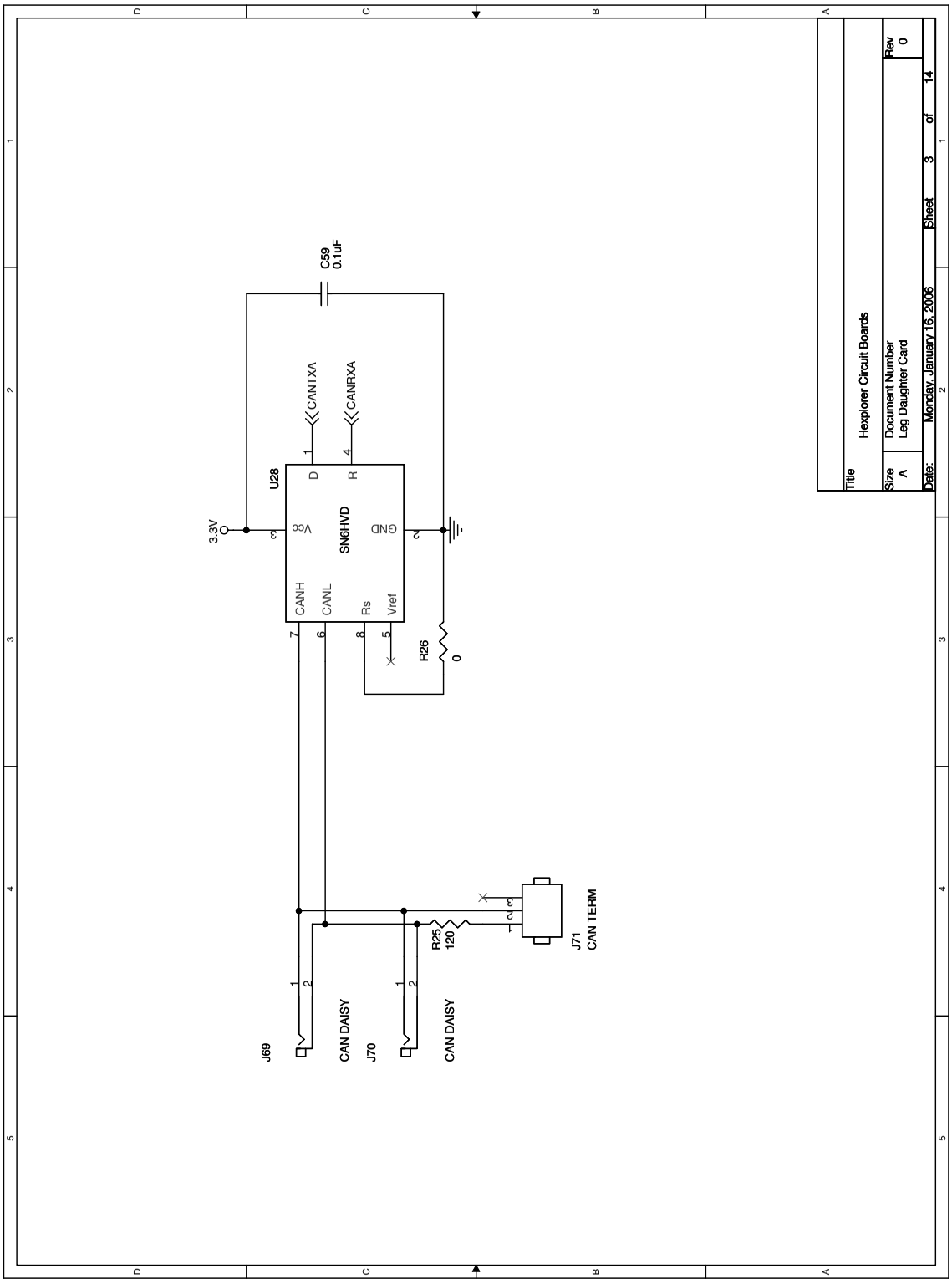

Title
Hexplorer Circuit Boards
Size
A
Document Number
Brain Board
Rev
0
Date:
Monday, January 09, 2006
Sheet
6
of
14
3.3V
C58
1uF
C57
0.1uF
U24
TPS79633
BYPASS
OUT
IN
EN
GND
5V
C56
2.2uF

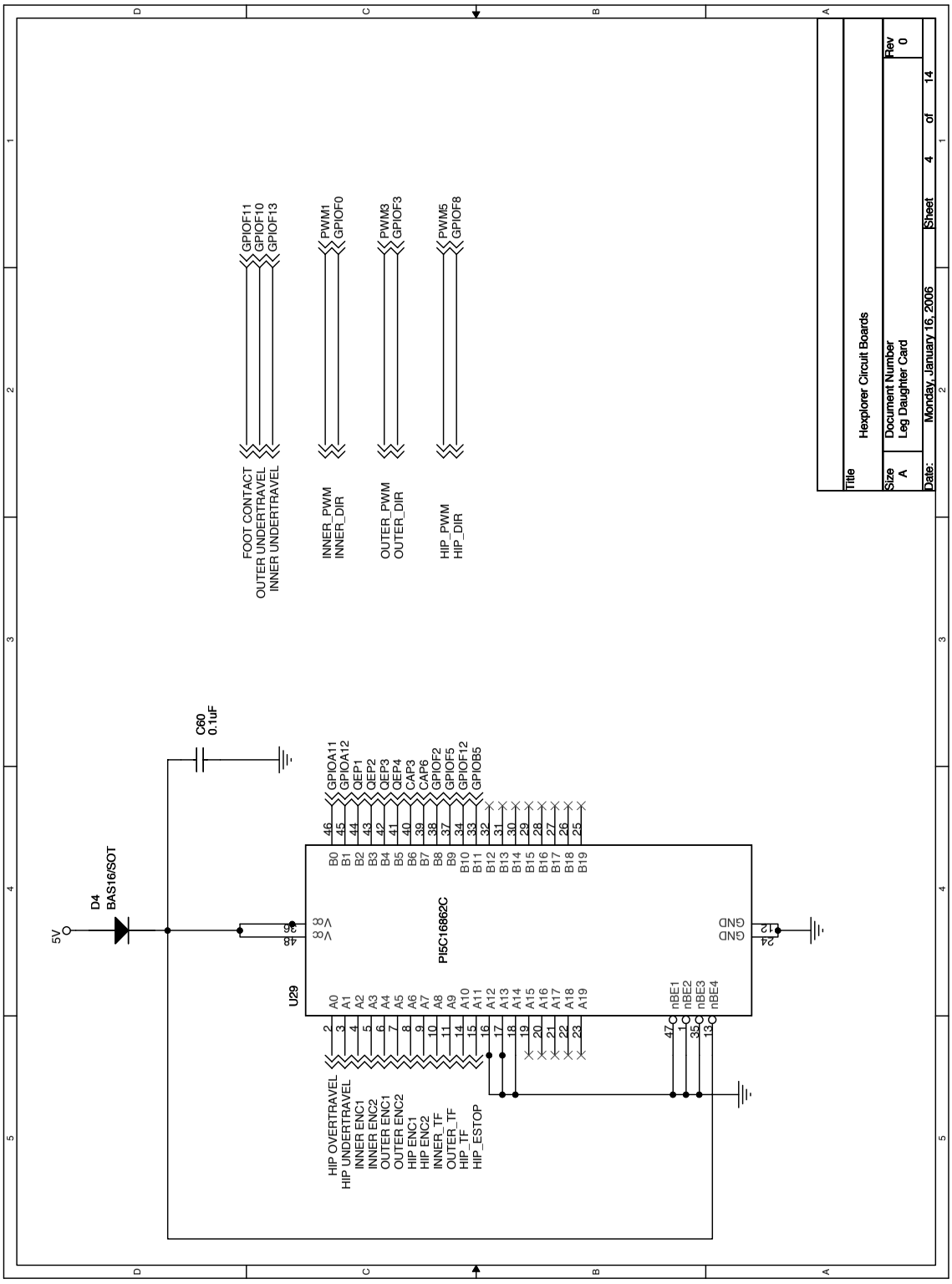
Appendix C

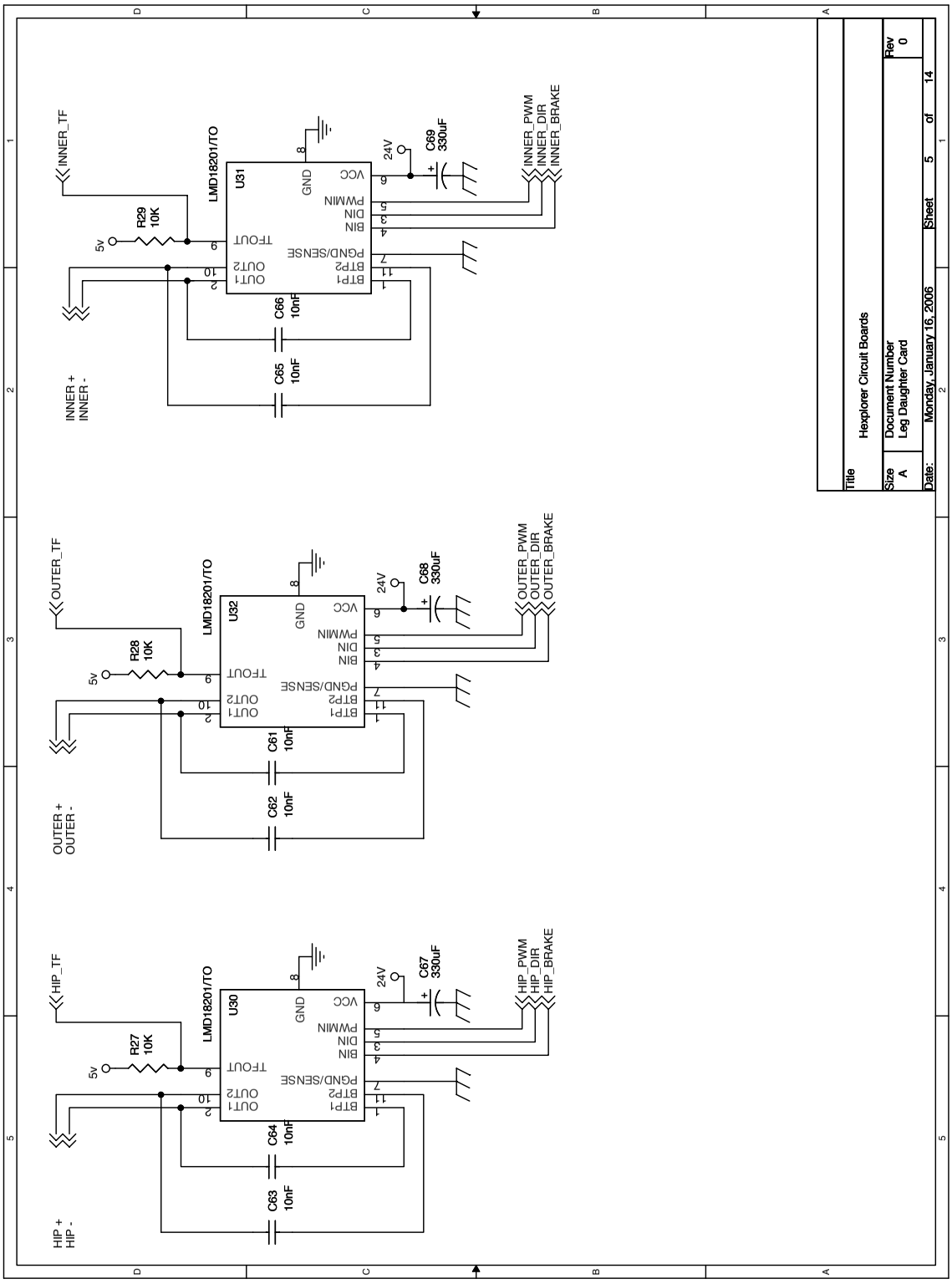
Leg Daughter Board Schematics

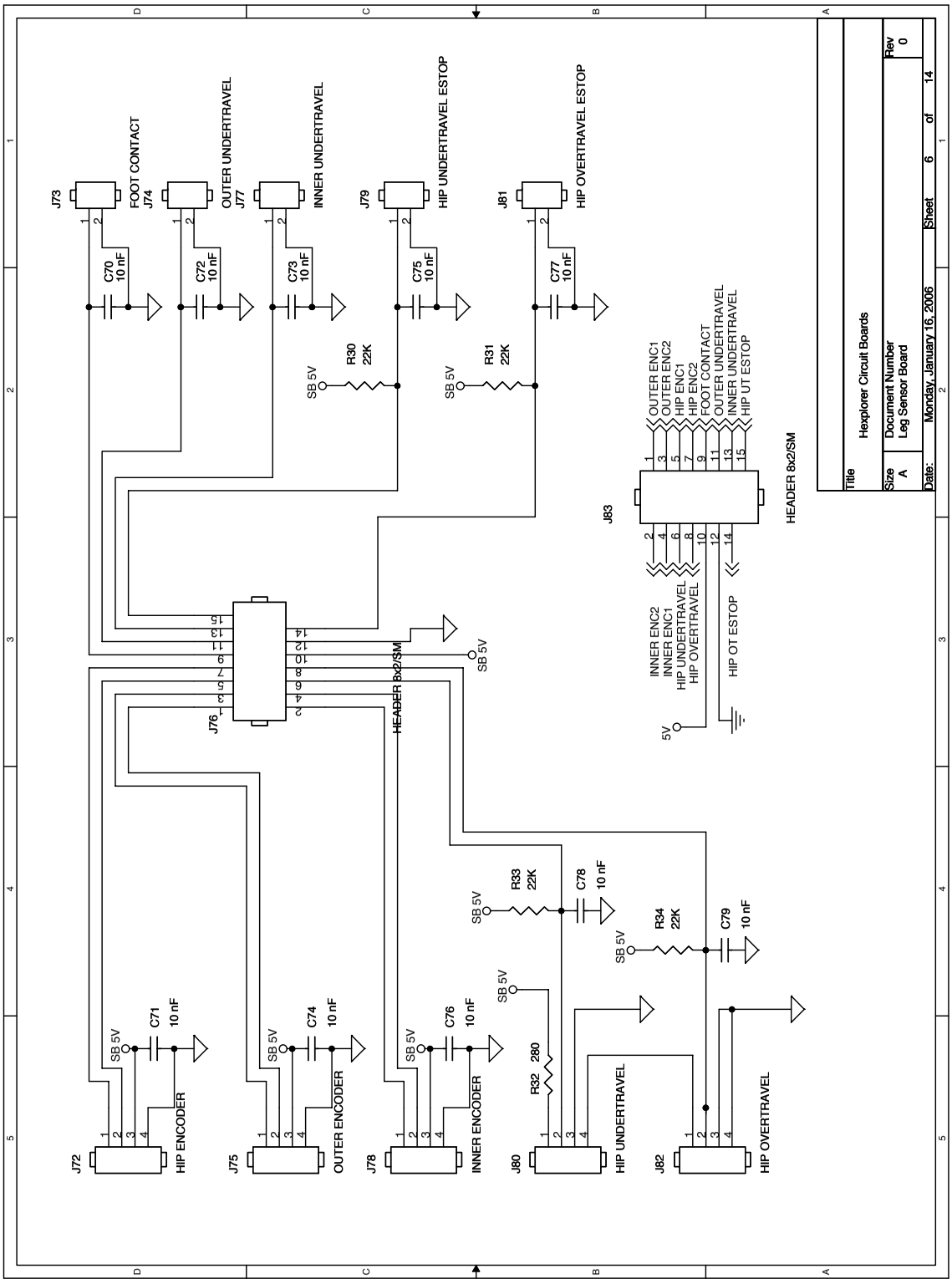

5v
C80
0.1uF
C81
0.1uF
C82
0.1uF
HIP_ESTOP
HIP OT ESTOP
HIP UT ESTOP
U25A
74HCT32
U27A
74HCT32
U26A
74HCT08/SO
GPIOB4
U25D
GPIOF9
HIP_BRAKE
74HCT32
U25B
GPIOF4
OUTER_BRAKE
74HCT32
U25C
GPIOF1
INNER_BRAKE
74HCT32
Title
Hexplorer Circuit Boards
Size
A
Document Number
Leg Daughter Card
Rev
0
Date:
Monday, January 16, 2006
Sheet 1 of 14


Title
Hexplorer Circuit Boards
Size
A
Document Number
Leg Daughter Card
Rev
0
Date:
Monday, January 16, 2006
Sheet
2
of
14
J65
HEADER 20
J66
HEADER 20X2
J67
HEADER 30X2
J68
MOTOR POWER
LEG1
POWER
5V
3.3V
24V
GPIOF4
XINT1n/XBIOn
QEP2
PWM1
PWM3
PWM5
T1PWM/T1CMP
GPIOA11
GPIOF0
GPIOF2
CANTXA
XCLKOUT
PWM8
PWM10
GPIOB5
T1CTRIP/PDPINTAn
GPIOF5
QEP1
CAP3
PWM2
PWM4
PWM6
T2PWM/T2CMP
GPIOA12
XINT1n/XBIOn
GPIOF1
GPIOF3
CANRXA
PWM7
PWM9
GPIOB4
QEP3
T3CTRIP/PDPINTBn
XINT2/ADCSOC
GPIOF8
GPIOF9
GPIOF10
GPIOF11
GPIOF12
GPIOF13
QEP4
CAP6
T3PWM/T3CMP
T4PWM/T4CMP
TDIRB
TCKLINB
XF/XPLLDISn
SCITXDB
SCIRXDB
HIP +
HIP -
INNER +
INNER -
OUTER +
OUTER -
XD0
XD2
XD4
XD6
XD8
XD10
XD12
XD14
XA0
XA2
XA4
XA6
XA8
XA10
XA12
XA14
XZCS0AND1n
XREADY
XRnW
XWE
XRSn/RSn
XA16
XA18
XHOLAn
XD1
XD3
XD5
XD7
XD9
XD11
XD13
XD15
XA1
XA3
XA5
XA7
XA9
XA11
XA13
XA15
XZCS2n
XRDn
XNMI/INT13
XA17
XHOLDn


C59
0.1uF
CANTXA
CANRXA
U28
SN6HVD
D
R
Vcc
GND
CANH
CANL
Rs
Vref
3.3V
R26
0
R25
120
J69
CAN DAISY
J70
CAN DAISY
J71
CAN TERM
Title
Hexplorer Circuit Boards
Size
A
Document Number
Leg Daughter Card
Rev
0
Date:
Monday, January 16, 2006
Sheet
3
of
14


FOOT CONTACT
OUTER UNDERTRAVEL
INNER UNDERTRAVEL
GPIOF11
GPIOF10
GPIOF13
INNER_PWM
INNER_DIR
PWM1
GPIOF0
OUTER_PWM
OUTER_DIR
PWM3
GPIOF3
HIP_PWM
HIP_DIR
PWM5
GPIOF8
Title
Hexplorer Circuit Boards
Size
A
Document Number
Leg Daughter Card
Rev
0
Date:
Monday, January 16, 2006
Sheet
4
of
14
C60
0.1uF
D4
BAS16/SOT
5V
U29
PI5C16862C
GPIOA11
GPIOA12
QEP1
QEP2
QEP3
QEP4
CAP3
CAP6
GPIOF2
GPIOF5
GPIOF12
GPIOB5
HIP OVERTRAVEL
HIP UNDERTRAVEL
INNER ENC1
INNER ENC2
OUTER ENC1
OUTER ENC2
HIP ENC1
HIP ENC2
INNER_TF
OUTER_TF
HIP_TF
HIP_ESTOP


INNER_TF
R29
10K
5v
LMD18201/TO
U31
TFOUT
OUT1
OUT2
PGND/SENSE
BTP1
BTP2
BIN
DIN
PWMIN
GND
VCC
24V
C69
330uF
INNER_PWM
INNER_DIR
INNER_BRAKE
C66
10nF
C65
10nF
INNER +
INNER -
OUTER_TF
R28
10K
5v
LMD18201/TO
U32
C68
330uF
OUTER_PWM
OUTER_DIR
OUTER_BRAKE
C61
10nF
C62
10nF
OUTER +
OUTER -
HIP_TF
R27
10K
5v
LMD18201/TO
U30
C67
330uF
HIP_PWM
HIP_DIR
HIP_BRAKE
C64
10nF
C63
10nF
HIP +
HIP -
Title
Hexplorer Circuit Boards
Size
A
Document Number
Leg Daughter Card
Rev
0
Date:
Monday, January 16, 2006
Sheet
5
of
14


J73
FOOT CONTACT
J74
OUTER UNDERTRAVEL
J77
INNER UNDERTRAVEL
J79
HIP UNDERTRAVEL ESTOP
J81
HIP OVERTRAVEL ESTOP
C70 10 nF
C72 10 nF
C73 10 nF
C75 10 nF
C77 10 nF
SB 5V
R30 22K
R31 22K
J76
HEADER 8x2/SM
J83
HEADER 8x2/SM
OUTER ENC1
OUTER ENC2
HIP ENC1
HIP ENC2
FOOT CONTACT
OUTER UNDERTRAVEL
INNER UNDERTRAVEL
HIP UT ESTOP
INNER ENC2
INNER ENC1
HIP UNDERTRAVEL
HIP OVERTRAVEL
HIP OT ESTOP
5V
R33 22K
R34 22K
C78 10 nF
C79 10 nF
C71 10 nF
C74 10 nF
C76 10 nF
R32 280
J72
HIP ENCODER
J75
OUTER ENCODER
J78
INNER ENCODER
J80
HIP UNDERTRAVEL
J82
HIP OVERTRAVEL
Title
Hexplorer Circuit Boards
Size A
Document Number
Leg Sensor Board
Rev 0
Date: Monday, January 16, 2006
Sheet 6 of 14