ABSTRACT

THE DEVELOPMENT OF HARRY, A QUADRUPED ROBOT WITH DYNAMIC OBSTACLE AVOIDANCE

Work done in this thesis pertains to the design, development, fabrication, assembly, and testing of a quadruped robot. The leg design is inherited from Baris Alp's robot. This unique leg design allows for the lower leg actuator, which is most commonly located at the knee joint, to be positioned at the hip joint by transferring motion through a kinematic linkage system. This configuration allows for the robot's legs to have lower rotational inertia when compared to configurations which have knee actuators. The body frame design is novel. Software (including the gait theory, lidar interface, networking, etc.) utilizes many existing frameworks, several of which are under a ROS (robot operating system) environment. The gait theory uses Bézier curves for foot trajectories, and is inspired by a model used by MIT's Cheetah Bot I. A Rplidar A1 360° scanning module is used in conjunction with Python and ROS for interfacing and data parsing. Networking and communication are done with the robot via secure shell and secure file transfer protocol. Twelve MG996R servos are used to achieve locomotion. Pulse width modulation and inter-integrated circuit communication are used via a PCA9685 board to control the servos, with power coming from an external RS310P DC power supply and requiring up to approximately 25 watts. The robot's onboard computer is a Raspberry Pi 4B with two gigabytes of RAM running an Ubuntu 20.04 long term support server operating system with ROS Foxy installed. Fabrication was done primarily with 3D printing. Quick and easy to use attachable/detachable panels utilizing Velcro were designed and included in the robot's frame with the goal of having a modular design. Testing was performed in simulated ROS and SolidWorks environments, as well as in the real world. The product is capable of walking along a linear path with detecting and avoiding moving obstacles. The robot is novel by being to the best of the author's knowledge, the first open hardware and software quadruped which uses a high-level computer for computation that doesn't require an actuator located at the knee joints.

Noah William Haworth
May 2022

THE DEVELOPMENT OF HARRY, A QUADRUPED ROBOT WITH DYNAMIC OBSTACLE AVOIDANCE

by

Noah William Haworth

A thesis
submitted in partial
fulfillment of the requirements for the degree of
Master of Science in Engineering
in the Lyles College of Engineering
California State University, Fresno
May 2022

APPROVED

For the Department of Mechanical Engineering:

We, the undersigned, certify that the thesis of the following student meets the required standards of scholarship, format, and style of the university and the student's graduate degree program for the awarding of the master's degree.

Noah William Haworth
Thesis Author

The Nguyen
Thế Nguyen (Chair) Mechanical Engineering

Daming Zhang
Daming Zhang Industrial Technology

Gemunu Happawana
Gemunu Happawana Mechanical Engineering

For the University Graduate Committee:

Dean, Division of Graduate Studies

ACKNOWLEDGMENTS

I would like to thank my advisor Prof. The Nguyen for his mentorship, caring patience, supplies, and invaluable insights he provided me with through this work. I would also like to thank Prof. Daming Zhang for introducing me to robotics, and for sharing his extensive knowledge of hardware, software, and the robotics community in general, and for also sharing his supplies with me as well. I would like to thank Prof. Happawana for his instruction in Lagrangian mechanics, dynamics, and control theory.

I owe much thanks to Prof. Sankha Banerjee for his continued mentorship, care, and shared academic wisdom. I am also extremely grateful for Prof. Aaron Hoskins mentorship and advice for programming and curve fitting problems. Dr. Ram Nunna’s encouragement and support in my research endeavors have been extremely helpful as well.

Numerous fellow students including but not limited to Anderson Medina, Tanner Meeks, Faheem Shaikh, Mandeep Singh, Riley Wells, Timofey Volkov, and Jesus Gutierrez have provided me with invaluable technical knowledge, emotional support, and friendship.

Finally, I owe a huge thanks to my family for their unconditional love and support throughout my life, including during my academic studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Scientists and engineers have often gained inspiration from nature, and this is the case with developing legged robots. There is great diversity in the number of legs animals and robots have. Two and four are common numbers in both nature and robotics [1, 2, 11, 15]. Examples which have eight or more legs can also be found in nature [3]. Examples with six, three, or even one leg can be found in robotics [4, 5, 6]. Four is the most common number of legs for both legged animals and robots. These animals and robots are referred to as quadrupeds. This thesis documents the work done to develop Harry; the quadruped robot seen below in Figure 1.

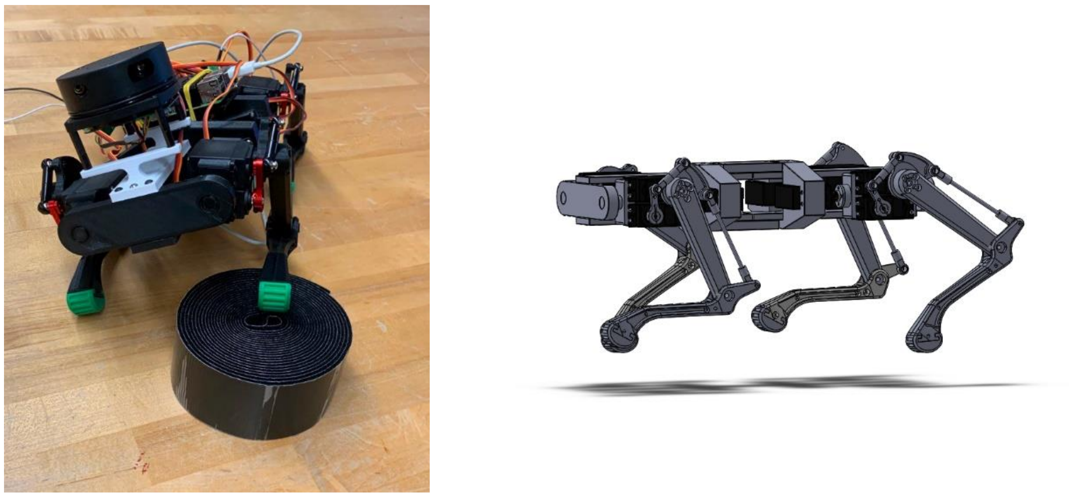


Figure 1: (a) Harry in real-life and (b) Harry in rendered in SolidWorks

The main advantage of legged robots over their wheeled counter parts is their potential to navigate in more complex terrain. Legged robots have several downsides: their kinematics and dynamics are more complex, they tend to have lower velocities, and lower carrying capacities compared to wheeled robots. However once solved, the kinematics and dynamics issues are no longer inherit disadvantages. Also, there are many applications where high velocities are not required such as site inspections and most agricultural applications. Lower carrying capacities are also not always problematic depending on the application, and there are legged robot examples that do have impressive carrying capacities such as Boston Dynamic’s BigDog (50lb or more) [7].

Many quadruped robots have already been created, such as Spot, ANYmal C, Champ, MIT’s Cheetah Bot, SILO4, Astro, Stanford Pupper, and many others [8, 9, 10, 11, 12, 13, 14]. Most of these robots have actuators located at the knee joints to control the motion of the lower leg. To the best of the author’s knowledge, only three quadruped robots do not have actuators located at the knee joints. These three robots are Alp’s quadruped [16], Zenichowski’s quadruped [17], and Harry (this work). The lack of an actuator at the knee joint is advantageous because it allows for each leg to have a comparatively lower rotational inertia when considering robots that do have these actuators.

The scope of topics covered in this thesis are numerous, including gait theory, inverse kinematics, mechatronics, and lidar. Thus, much of the work done involved researching these various topics and exploring existing frameworks within each topic. Existing frameworks were modified to yield the specific output required or desired by and for this work for each topic. The combination of all combined frameworks yielded Harry as the product of this thesis. Most frameworks not implemented directly still contributed indirectly, most often through the underlying theory and concepts learned during their investigations.

LITERATURE REVIEW

Bézier curves are used extensively in path planning for autonomous vehicles and robots. Bézier curves have several properties which make them convenient for use in this context. Firstly, Bézier curves can be easily expressed in the form of polynomials, specifically as an extension of Bernstein polynomials. Secondly, a series of Bézier segments can be made continuous to an arbitrary degree due to the calculation of derivatives of Bézier curves being extremely straightforward thanks to their polynomial representation.

Ji-wung Choi and associates proposed two algorithms which can be used to generate a planned path for autonomous vehicles with obstacle constraints and waypoints (sometimes referred to as control points) as input [18]. These waypoints can either give the general direction of a curve in a certain region, or a destination point through which the curve passes. Yu presents a means of producing temporal spatial Bézier curves for the arrival of multiple unmanned vehicles with various metaheuristics including PSO (particle swarm optimization) and genetic algorithms. Bézier curves are used in gait planning by Jimeno's framework, in a similar method proposed by Lee, and in both Zenichowski's and Elarabawy's inverse kinematics models [19]. MIT's Cheetah bot's use of Bézier curves can be seen in Figure 2.

Lee's hierarchical control algorithm allows MIT's cheetah robot to trot at an impressive speed of 6 meters per second stably by utilizing equilibrium point hypothesis for each leg, impedance control to coordinate between each leg, and a gait foot-fall planner. This is the fastest known speed obtained by any quadruped robot. For reference, the average sprinting speed of a human is approximately 7 meters per second. Lee also developed a dynamics simulator using MATLAB to supplement the results of his real-world experiments performed with Cheetah Bot [11]. Cheetah Bot utilizes legs with four segments. Different leg configurations can be seen in Figure 3.

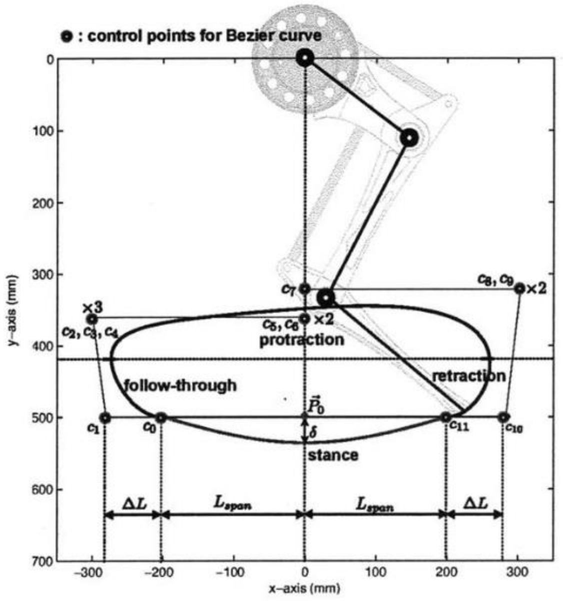


Figure 2: Lee’s Bézier curve with control points tracing foot trajectory [11]

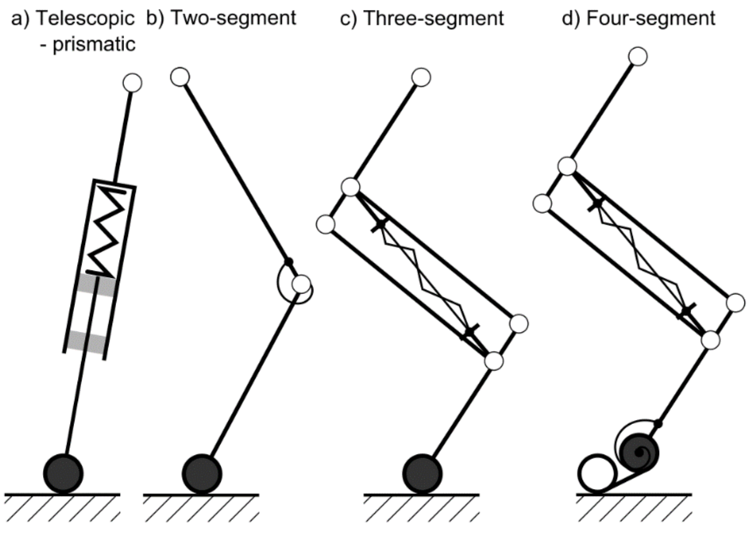


Figure 3: Leg configurations by Sprwötiz [20]

Harry uses a modified form of the two-segment configuration. Of the various quadrupeds investigated in this thesis, two-segment legs are the most common. Biswal and Mohanty, in their survey of quadrupeds, found that both animals and robots which utilize three joints: hips, knees, and ankles, perform best at running at high speeds and are energy efficient [21]. Animals such as tigers, cheetahs, wolves, and leopards have this leg structure. While Cheetah Bot has the highest overall speed amongst quadrupeds, Cheetah cub has the highest recorded speed in terms of Froude number [20], which takes a robot's stance height into consideration. The equation below can be used to calculate Froude number.

$$f = \frac{v}{\sqrt{gL}} \tag{1}$$

Where v is the robot's velocity, g is the acceleration objects experience due to gravity near earth's surface, and L is the characteristic length, or stance height in this case. This is a unitless value, and in other contexts, the square root of this value may be used instead. Another metric to compare velocities or speeds more fairly between robots is using robot lengths per second as the unit of measurement or magnitude.

Ridderström also conducted a survey of legged robots, some of which were quadrupeds [22]. Ridderström's primary interest with his survey was to map out the gait control systems used by various legged robots and use some of these systems to control WARP 1, the quadruped seen in Figure 4.

Ridderström presents a control system in detail which uses inverse dynamics and rigid body dynamics with Kane's equations. At the end of his thesis, Ridderström shares limit cases where "statically stable stances" may in fact not be stable.

Elarabawy's inverse kinematics model is the first known public model that considers offsets common at hip or shoulder joints in quadrupeds. These offsets can be seen in Figure 5.

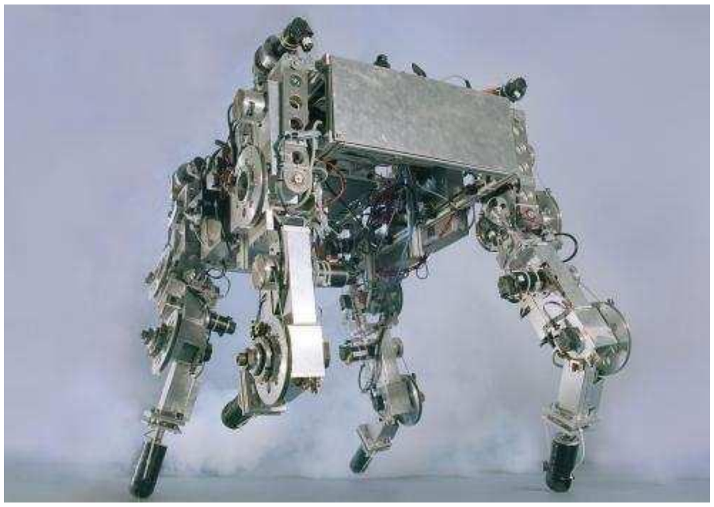


Figure 4: WARP 1 [22]

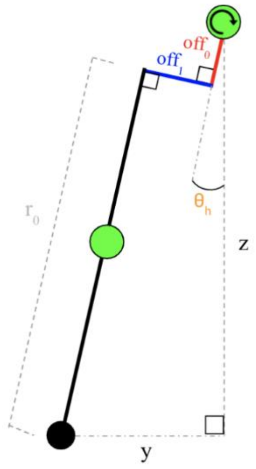


Figure 5: Elarabawy’s hip offset considerations [19]

As Harry's vertical offset goes up instead of down as depicted above, Elarabawy's equations were modified to account for this difference. The resulting equations are used by Harry. These resulting equations can be found in Appendix A.

Two hard problems legged robots face is having a sufficiently powerful onboard power supply, and the ability to navigate over rough terrains. BigDog was designed by Boston Dynamics to make progress on both fronts [7]. BigDog is powered by a two-stroke internal combustion engine which can power all the onboard computers, sensors, and actuators. BigDog's actuators are hydraulic pistons. BigDog uses the information collected from approximately 50 onboard sensors, along with its low- and high-level control systems which allows it to navigate in terrains which are uneven, in mud and in snow. An image of BigDog navigating in one such environment can be seen in Figure 6.

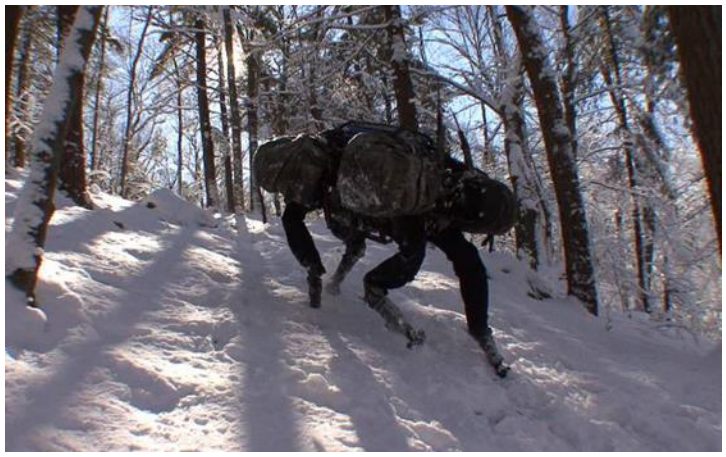


Figure 6: BigDog Navigating on a snowy hill [6]

Kimura and associates used biomimicry to develop several new capabilities for their robot, Tekken2 [23]. Tekken2 utilizes PD-controllers at its hips to function as a virtual spring-damper to replicate leg muscles acting in a visco-elastic manner. Tekken2 also has a CPG (central pattern generator) trained as a NNM (neural network model, or neural system model) to plan the gait phase and respond to uneven terrain. Some of

Tekken2's novel capabilities include the ability to react to tripping over small bumps or pebbles, side stepping with roll motion when going up slopes for stability, and stability control in the event of falling due to loss of contact with ground. Functionalities such as these are often referred to as "reactions" in the context of autonomous robots.

There are many existing open-source and non-open-source quadruped robots. To the best of the author's knowledge, Harry is the first open-source quadruped that combines the alternate leg configuration with a high-level software interface and autonomous avoidance of dynamic obstacles. Baris and Zenichowski [16, 17] have produced the only other found open or partially open-source quadrupeds with leg configurations similar or identical to Harry's. Harry's leg configuration was inherited from Baris's open-source robot. Harry's gait theory was heavily influenced by Lee's and Jimeno's work [10, 11]. Other existing quadrupeds include Champ, Aliengo, ANYmal B, ANYmal C, Astro, D'kitty, Dream Walker, Mini Cheetah, Mini Pupper, Open Quadruped, Opendog, Spot Micro, Stanford Pupper [24, 25, 26, 27, 28, 29].

Champ was created by Jimeno with inspiration from MIT's cheetah robot. Jimeno has produced very valuable material for the open-source robotics community by producing and releasing software which automates the creation of many of the files required to run custom quadruped designs with ROS Melodic or Kinetic [47, 48]. Much time was spent experimenting with Jimeno's framework. The gait theory is based on Lee's work with the goal of producing both a stable and agile gait model. An inertial measurement unit (IMU) sensor provides data to aid in the control system, with the objective of keeping the body link parallel and stable with respect to ground. Jimeno's framework is not currently directly utilized by Harry for three reasons. Firstly, Harry does not have an IMU sensor. This reason should be simple to resolve as a Raspberry Pi 4 compatible and inexpensive six degrees of motion IMU sensor is available. Secondly, Harry's leg configuration is not considered by Jimeno's general framework. Future work can be conducted to incorporate Harry's leg configuration as an option for Jimeno's general framework. It is believed this would be beneficial to Harry to allow an alternative gait theory which is powerful, and beneficial to Jimeno's framework by adding to the scope of possible quadrupeds it can cater to. Figure 7 depicts some of the basic gait parameters this framework uses.

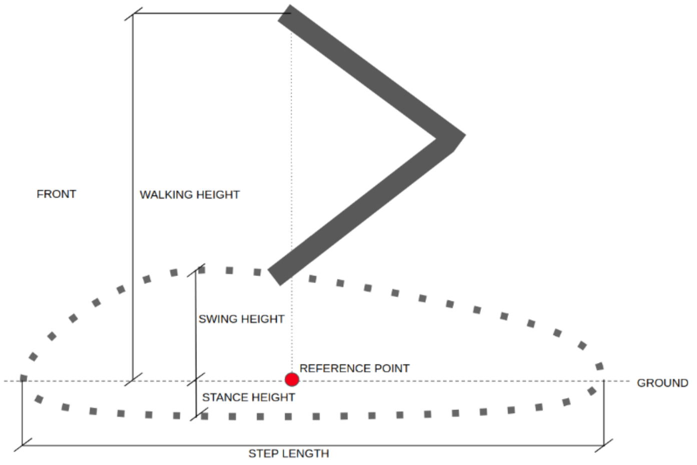


Figure 7: Jimeno's gait parameters [10]

Thirdly, Jimeno's program creates files for simulating and running a quadruped in ROS Melodic which runs natively under Ubuntu 18.04 LTS and on a Raspberry Pi 3. Converting this system to run under a more current ROS distribution such as ROS Foxy on a Raspberry Pi 4 would require much work. Jimeno's and Lee's work is still implemented in directly in Harry; this process is described in the methodology gait section in detail.

Aliengo is a closed source quadruped developed by Unitree as a highly agile robot capable of performing a backflip. It is currently possible to preorder an Aliengo unit for $38,000 [24].

ANYmal B is a quadruped made by ANYbotics through research at Oxford that is intended to be used commercially as a working robot. ANYmal B is capable of walking over obstacles. ANYmal C is the next evolutionary step up from ANYmal B. ANYmal C is capable of walking up and down stairs and navigating autonomously in rough terrains.

This robot is currently marketed as a robot capable of performing site inspections. Figure 8 shows ANYmal C [9].

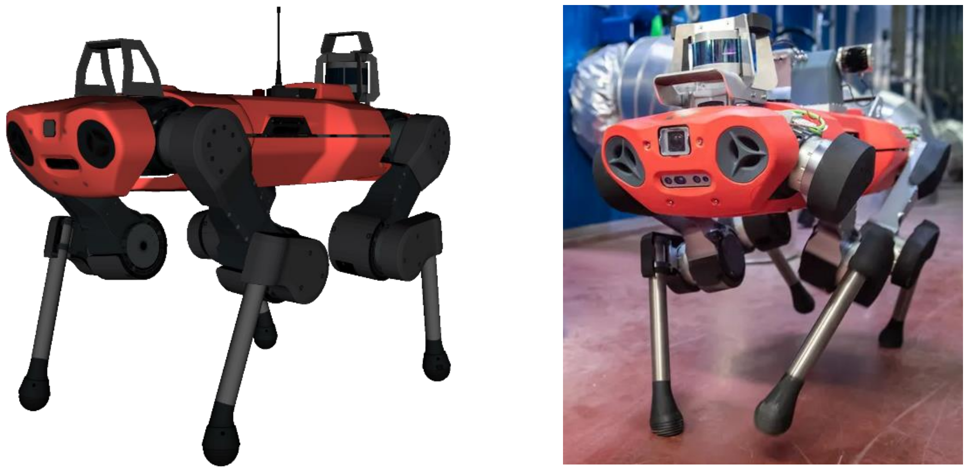


Figure 8: (a) ANYmal C rendered in RViz (b) and in real life [9]

Astro was created by the Cognitive Robotics Laboratory at Florida Atlantic University. Astro heavily inspired and modeled after a Doberman pinscher. Astro can follow simple voice commands such as “sit”, “get up”, and “go forward” [13].

D’kitty is a quadruped intended strictly for academic research of locomotion of quadrupeds. For a time, it was possible to order a D’kitty unit for approximately $3,700 [25].

Dream Walker is an open-source quadruped developed by Uehrara. Dream walker is still in the development stage and is utilizing Jimeno’s framework for gait control [26].

LittleDog is a quadruped developed by Kalakrishnan et al. that combines several advanced sub-systems in its gait planning [30]. LittleDog is capable of navigation on uneven terrains that are not included in its training set.

Open Quadruped is a truly inspirational quadruped created by Elarabawy when he was still in high school. Elarabawy’s gait control is like Harry’s gait control as both use inverse kinematics and Bézier curves for trajectory planning [28]. Open Quadruped also uses ROS. Elarabawy's work is open sourced here on GitHub. Elarabawy's robot leg follows the normal convention of having the lower actuator mounted at the knee joint. Elarabawy's Open Quadruped can be seen in Figure 9.

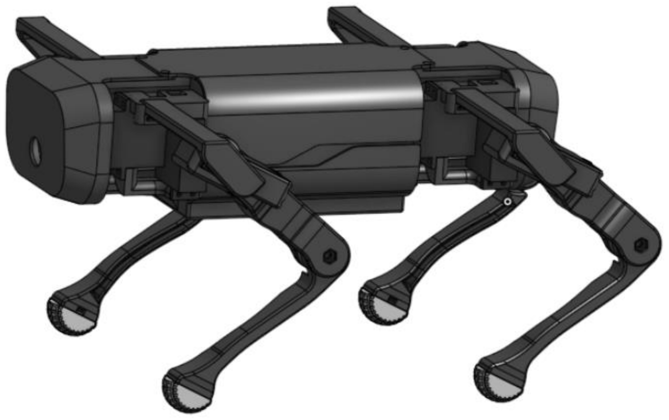


Figure 9: Open Quadruped by Elarabawy [28]

Spot is a quadruped famously developed and produced by Boston Dynamics with roots at MIT [8, 21]. Spot is likely both the most sophisticated and well-known quadruped currently in existence. Spot can be seen in Figure 10. Spot's yellow upper leg panels alone cost $250 each. An entire Spot unit is sold for approximately $75,000.

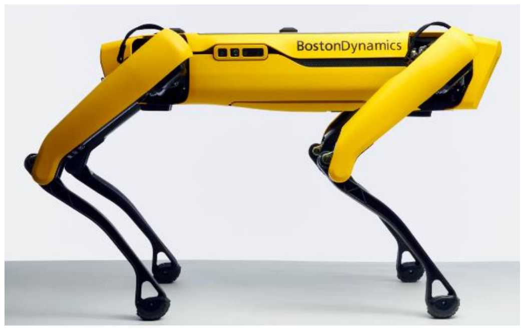


Figure 10: Boston Dynamics's Spot [8]

Spröwit developed cheetah cub, a relatively small and light robot capable of the highest speed in terms of body lengths per second under 30 kgs [20]. Cheetah cub is also unique in that it utilized a control system with an open loop, like Harry.

McClean warns of possible cyber security vulnerabilities robots using ROS may have [31]. One of McClean's primary concerns is that communication between nodes takes place via plain text messages instead of encrypted messages. The use of these unencrypted messages makes for a simpler interface between ROS nodes, and for developer debugging; however, makes it much easier for an unintended audience or listener to also intercept and log information flowing in a ROS network. One individual was able to remotely tap into a device running ROS and remotely control one of the robot's actuators by inserting their own ROS compatible messages. Events such as these should be of concern in applications where ROS is to be used in information sensitive environments. In such a context, it is highly recommended to take cyber security minded steps such as encrypting messages and storage devices which is not done by default in most ROS versions. Harry does not currently have such security concerns. It should also be noted that the Ubuntu operating system is quite secure in terms of cyber security, partly due to the hierarchy of user permissions structure used by default. ROS does not run with root or superuser permission by default, and it is advised to not run ROS with these permissions to reduce risk. Rivera proposes modern methods to greatly increase cyber security in ROS [32].

Gargulak provides different classification of gaits based on each leg's phase seen in Figure 11 [33].

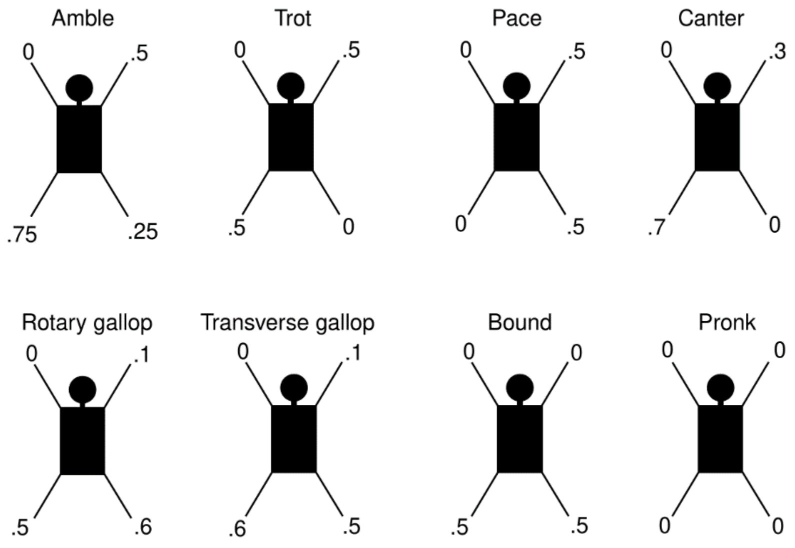


Figure 11: gait type based on leg phase [33]

Harry has access to both amble and trotting gaits. Amble gaits are used in nature for speeds just faster than walking, and trotting is a symmetric gait and used often in nature for slower to medium speeds with high stability [21].

JinYe He and associates conducted a modern survey of various quadruped's ability to react to complex terrains [34]. He's conclusion is that quadrupeds are not yet ready to interact in real-world environments; however, the ability of some quadruped's reflexes to slopes, uneven terrains, and dynamic terrains is quite impressive and has come a long way in the past ten years. These quadrupeds largely rely on extremely precise sensors and actuators. Also, quadrupeds must move more slowly through uneven terrains and require much more complex control systems when compared to navigating in simpler environments.

Liu and Ben-Tzvis' research investigated the use of a tail for stability control in the context of quadrupeds [35]. Like the research of many, Liu found inspiration by looking in nature. Liu's theoretical framework is unique in that the virtual work principle is utilized instead of the more common Lagrangian or Newton-Euler methods. The basic model considered in Liu's work can be seen in Figure 12.

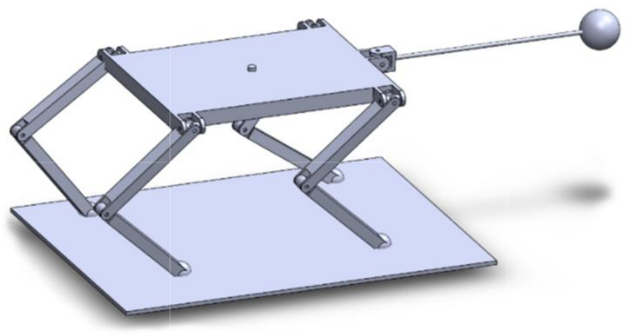


Figure 12: Liu’s model with a pendulum tail for stability [35]

Ganjare and associates proposes a beautifully simple quadruped gait framework which uses circular arcs for the feet trajectories [36]. Zenichowski also uses circular arcs for his feet trajectories.

Ferrolho and associates investigated the performance of contemporary rigid body dynamics libraries solving systems in a forwards or inverse manner [37]. Using inverse dynamics was found to yield superior results in terms of speed of convergence and tended to be more robust than forward dynamics methods. Ferrolho utilized Julia’s rigid body dynamics package. Julia’s combination of ease of use and high performance is an extremely desirable and powerful combination.

Anand developed an inverse kinematics model for quadrupeds with six degrees of freedom per leg using a method presented by Pieper [38, 39]. Anand’s model is capable of fully determining such a robot’s position and orientation on rough or uneven terrain.

Pieper’s method includes both numerical and analytical solutions to the inverse kinematics of both new and existing manipulators [39]. Pieper also presents several heuristics which can navigate a six degree of freedom manipulator from some starting location to an ending location while avoiding known obstacles.

Muhammed and associates developed a kinematics model for a quadruped which has three degrees of freedom per leg [40]. Muhammed’s forward model utilizes Denavit-

Hartenberg's method. The inverse model uses simple geometric principles. The combination of these models was implemented in MATLAB for graphical simulation.

Semini and Wieber present a concise summary of the history of legged robots, including mono, bi, quad, and hex legged configurations [41]. Basic principles of gait, central pattern generators, and some of the basic mathematical methods commonly used such as Lagrangian mechanics are presented.

McRae developed a gait model for SILO4, which can also be used for other quadrupeds with insect type legs [12]. SILO4 with its insect type legs can be seen in Figure 13.

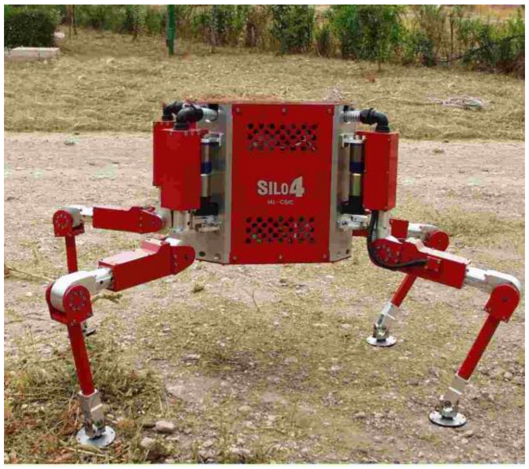


Figure 13: SILO4 with Insect Type Legs [12]

This framework ensures stability by checking that the quadruped's center of gravity remains within the confines of the support polygon defined by the location of the feet. McRae's framework uses both inverse kinematics and dynamics of the insect leg configuration.

Zhou and associates provide a simple method to turn a quadruped with twelve degrees of freedom at a specified angle with experimental results being within 3% of the theoretical values [42]. Figure 14 shows a quadruped implementing Zhou's turning method.

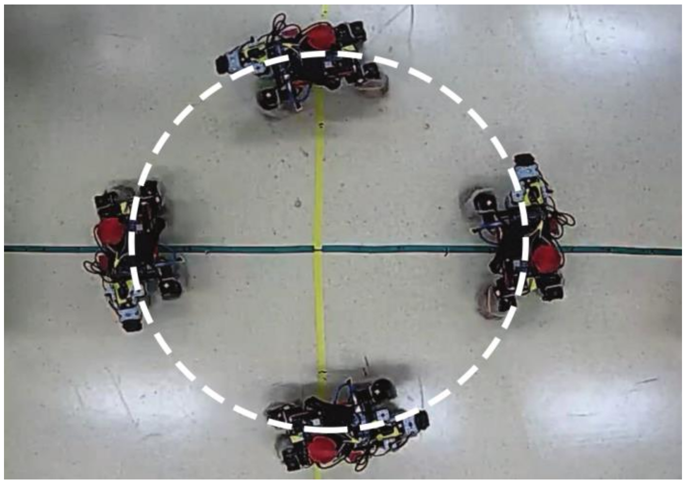


Figure 14: Zhou’s circular turning method [42]

Loakeimidis used a Rplidar A2 module in their thesis robot for SLAM via ROS Ingoo’s official rplidar, amcl, and move_base packages [43].

Xuexi and associates’ experiments with a Rplidar module using ROS’s gmapping, karto SLAM, and hector SLAM demonstrated that producing a precise two-dimensional map on an indoor environment is feasible with the previously mentioned tools [44]. Xuexi found that the hector slam algorithm is often sufficient at tracking a robot’s movements by comparing a sequence of 2D lidar scans. Hector SLAM also requires less computational resources than other localization algorithms such as traditional particle filter algorithms. Karto SLAM can at times be a more efficient algorithm as all that is kept in memory is a Cholesky matrix; however, this benefit tends to dissipate in complex environments as more information is required for the algorithm to converge on a solution. Gmapping is not well suited for mapping large areas. Gmapping, hector slam, and karto slam all yielded similar real-world results when tested on the same robot. Using an imu sensor in conjunction with the rplidar module led to more accurate results.

METHODOLOGY

This section is broken down into five subsections: physical design & hardware, software, networking, and gait theory, and lidar.

The physical design and hardware subsection discusses Harry's structural components, including how the designs were obtained, and the software and file types these parts are available in. The methods by which these parts were physically created is also discussed. This section also lists the various components bought as-is. Physical characteristic such as principal lengths and weights are also presented. The software subsection discusses the operating system and middleware used. The gait theory subsection discusses various gait models either designed from scratch, or previously implemented and invested by this work. The final implementation of Lee's and Jimeno's work is also discussed. The lidar subsection shares about the hard and soft interface between the lidar module and the Pi, and how the lidar's information is used by Harry's gait system.

Physical Design & Hardware

Design of Harry's body was done with SolidWorks. The lower and upper leg segments from Alp's work were imported into SolidWorks as step (.stp) files, and converted into solid-part (.sldprt) files. The linkage system was created in SolidWorks with characteristic lengths measured from Alp's model. Harry's dimensions are provided in Table 1. A simplified depiction of one of Harry's legs can be seen in Figure 15.

Solid-part, step, and standard triangle language (.stl) files for Harry can be found here on GitHub. Harry's structural parts were mostly 3D printed out of PLA. Harry's feet were printed out of TPU, which is a flexible polymer. Using TPU for the feet allows for a greater surface area to be in contact between the feet and ground; and thus, greater

Table 1. Harry’s Physical Properties

Characteristics	Value
Principle dimensions	<400, 185, 190> mm
Upper leg length	100 mm
Lower leg length	102.55 mm
Hip Roll offset	85.9 mm
Hip Pitch offset	215.7 mm
Hip horizontal offset	42.55 mm
Hip vertical offset	11 mm
Robot mass with lidar	1.78 kg
Mass without lidar	1.55 kg

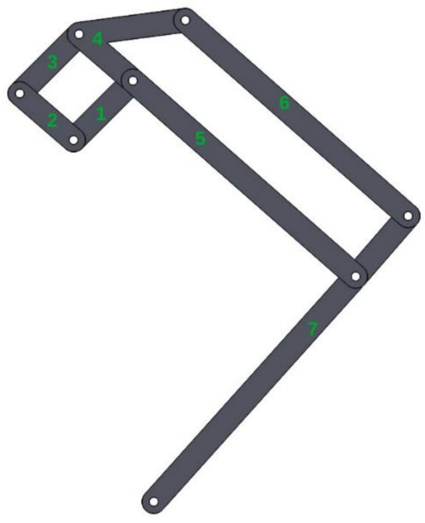


Figure 15: Simple leg linkage system

contact forces be achieved to help reduce slippage. All-thread rods are used in links 3 and 6 (seen in Figure 15 above) to provide the required length between the end rods. These end rods are used to achieve low friction between the various links in the linkage system. The all-thread for links 3 and 6 was cut to lengths of 20 and 84.5 mm respectively. Z390 ball-bearings secured with countersunk screw heads are used at the knee joints to achieve low friction between the upper and lower leg segments. M2.5, M3, and M4 screws are used to fasten most parts together. Harry currently has two Velcro attachment points which are used for the lidar module and PCA board. This can be seen in Figure 16.

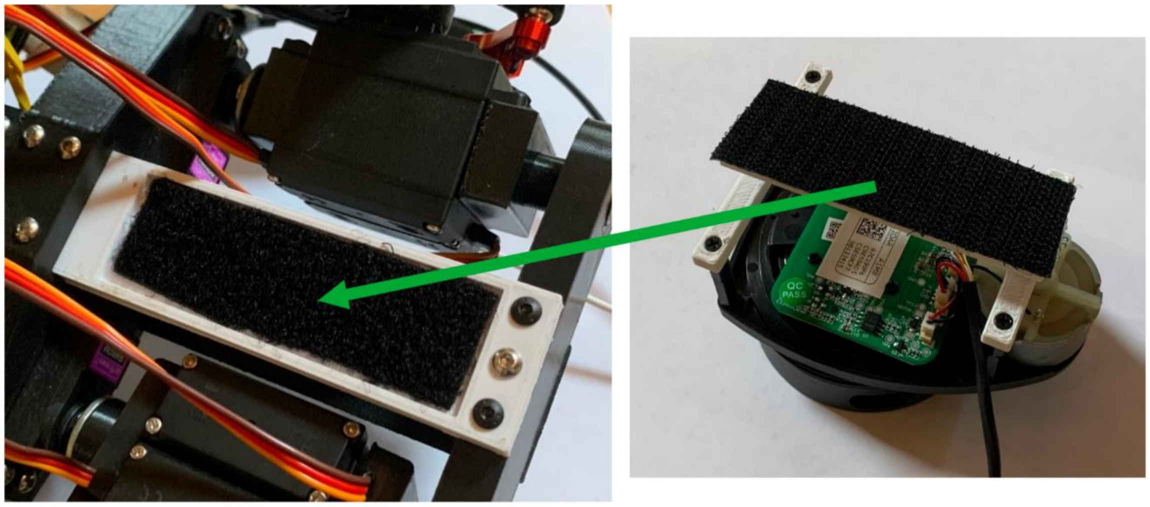


Figure 16: (a) Velcro attachment point (b) Lidar with Velcro

Assembling and tuning each leg required a fair amount of testing and time. Some servos were found to have oversized geared ends which prevented the attachment of servo horns, and thus rendered these servos unusable in this application. The selected servos were then carefully rotated manually to one extreme and the servo horns and upper arms then attached in a uniform orientation to make the calibration of each motor more streamlined after the first servo was tuned. Much time was spent experimenting with each motor to find the desired maximum and minimum angular displacements. Servos and upper legs had to be disassembled and reassembled in different orientations in some cases due to the servos’ limited range of motion.

Twelve MG996R servos are used to control Harry’s 12 degrees of freedom (DOF). These servos are rated to output up to 11 kgf • cm and have a mass of 55 grams each. These servos can rotate 180°. Adafruits’s python motor controller library is used to interface with these motors via pulse width modulation (PWM). The general-purpose input/output pins (GPIO) on the Pi were found to produce signal with too much noise to be useful, thus a PCA9685 16 channel PWM board with powered rails was used. The Pi communicates with this PCA through its serial data (SDA) and serial clock line (SCL) pins via inter-integrated circuit (I2C).

A Rplidar module is used to gather distance information. This lidar module spins around continuously in 360° at a rate of approximately 10 Hz. In this manner, a two-dimensional plane of Harry’s surrounding environment can be measured.

Powering the Pi requires 5 volts and between 2.5 to 3 amps. Voltage regulators were configured to be attached to Harry via the Velcro attachment points. However, it was found that Harry’s onboard lithium-ion batteries were unable to provide sufficient current to adequately supply the Pi or the servos. Thus, the Pi was powered by a 3-amp outlet adapter, and the servos were powered with an external RS310P DC power supply. Harry’s servos were observed to consume upwards of 3.5 amps when operating at 6 volts. The RS310P power supply was configured to output a constant 6 volts and variable amperage with an upper limit of 4 amps. This means Harry’s theoretical maximum power consumption is 24 watts; however, values above 20 were rarely observed.

Software

The most current Ubuntu 20.04 LTS (long term support) server image file tailored specifically to run on Raspberry Pi 4Bs was downloaded from Canonicals official website and flashed to a 32 Gb microSD card with Etcher [45]. This card was then inserted into the Pi. Ubuntu is the most used Linux based operating system for personal computers.

This has led to the development of a large community with good support and documentation. These facts have led the author of this thesis to believe Ubuntu is a good choice of operating system to use for this work.

Much time was spent creating and modifying scripts to give Harry his abilities. Harry's Pi can be thought of as his brain. More accurately, Harry's Pi is the component which runs all software which controls Harry's motion, sensor data collection, and data interpretation. This Pi is also the user interface point for powering Harry on or off, and to give direct directional commands.

ROS (robot operating system) serves as the middleware between the high-level python scripts and user interface and Harry's hardware sensors and actuators. ROS is an open-source project with a large footprint in the hobbyist robotics community. There are many versions of ROS which correspond to specific operating systems. ROS Foxy is the distribution of ROS Harry uses as it currently supports Ubuntu 20.04 LTS [46]. The utilization of widely used software such as Ubuntu and ROS give the benefit of much documentation and a helpful community. ROS Foxy was installed on Harry's Pi by following the Debian binary package installation instructions on the official ROS2 website.

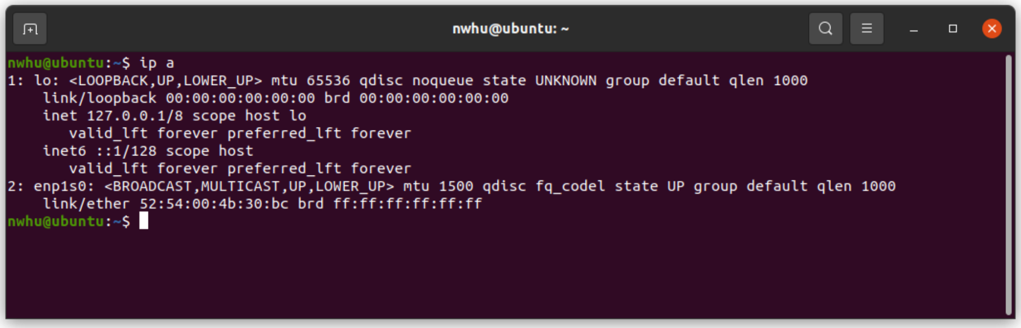
It was required to add the default Pi user account to the dialout and i2c groups, as well as change ownership of the Pi's GPIO pins to allow access without super user permissions.

Networking

Connecting to Harry must be done remotely in almost all cases as Harry has no external user interface built in. The only way to directly give and receive input from Harry is by plugging a keyboard and monitor into Harry's Pi. This method of interface is reserved only for desperate troubleshooting as this is impractical in the context of mobility and convenience. Thus, using remote means such as SSH (secure shell) or SFTP (secure file transfer protocol) are used for all normal operations, including most testing. For the sake of simplicity, Harry’s Pi and the interfacing device were always on the same wireless network. The interface device can be any modern PC or laptop running a typical Windows, Mac, or Linux operating system; however, the input device in this thesis was always a Linux machine for convenience and to allow forwarding the input systems windows manager (X11) to allow for the viewing of graphical user interfaces (GUIs) on Harry remotely while still only running a lightweight server system on Harry’s Pi. It is possible to use GUIs with Harry more directly by installing and running a desktop environment on Harry; however, this isn’t preferred as it requires Harry’s Pi to both store more data and consume considerably more hardware resources that are better allocated on sensor data interpretation, communication, and other processes.

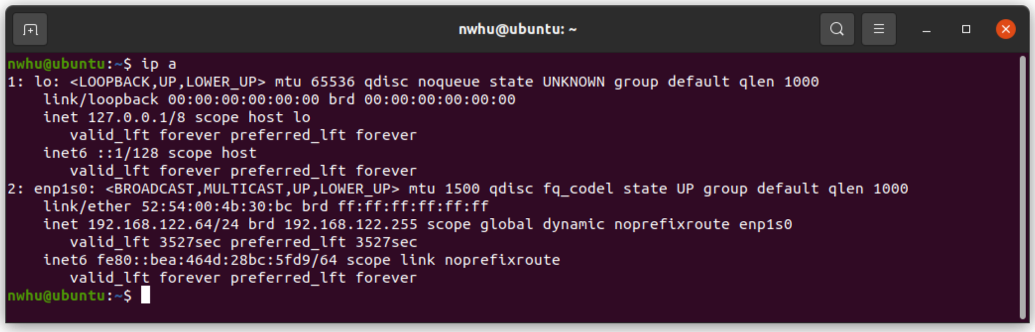
SSH is preferred for issuing shell commands. SFTP is preferred for modifying scripts and moving files around. Both protocols require that Harry's Ubuntu account name, account password, and IP address be known. The username and password are well known and static (“Harry” and “godogs!” respectively). Harry’s IP address, however, is not static across networks as DHCP (dynamic host configuration protocol) is used by default. This can be a slightly tricky problem to solve whenever Harry is connected to a foreign network for the first time. One possible solution to make this step become trivial is to assign a static IP address for Harry’s wireless antenna. Assigning such an address is recommended for any future work performed on Harry. The workaround currently in use with Harry is using a mobile phone as a hotspot to allow for a single IP address to be used across a wide range of locations. The most robust method of finding out the IP address for the first time connecting to a new network is to connect Harry’s Pi to a monitor through the Pi’s left micro-HDMI port (closest to the charging port) and attaching a keyboard via USB. This allows for direct access to the Pi. “ip a " can run in

TTY (teletypewriter) or terminal to view the device's IP address once it is connected to a network. Note that connecting a Pi to a smart tv may not register output; for this reason, it is recommended to connect to a simple monitor which has HDMI ports. VGA adapters can also be used with older monitors. Connecting to a network is done by default by modifying /etc/netplan/50-cloudplan.yaml and generating and applying a new netplan. Other supplicants such as iwd and iwctl could be used as well if the default netplan method fails. Figure 17 shows the output of “ip a” if the device is connected to no networks. Figure 18 shows the output of “ip a” if the device is connected to a network. Pinging a known URL (such as “ping google.com”) can be used to determine if a device is successfully connected to the internet and not just a wireless network. Successful and unsuccessful output using this method can be seen in Figure 19.



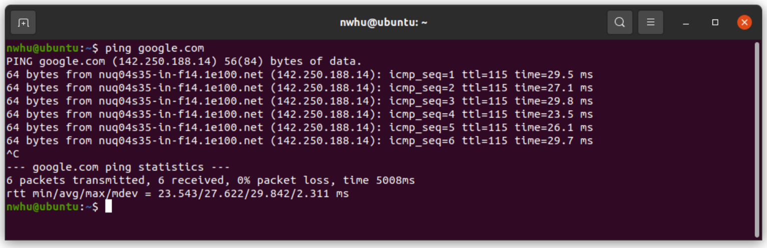
```
nwhu@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 52:54:00:4b:30:bc brd ff:ff:ff:ff:ff:ff
nwhu@ubuntu:~$ 
```

Figure 17: “ip a” output when connected to internet

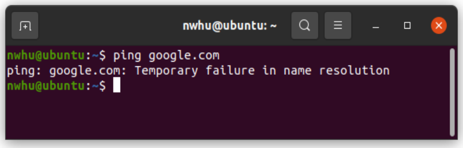


```
nwhu@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 52:54:00:4b:30:bc brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.64/24 brd 192.168.122.255 scope global dynamic noprefixroute enp1s0
       valid_lft 3527sec preferred_lft 3527sec
    inet6 fe80::bea:464d:28bc:5fd9/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
nwhu@ubuntu:~$ 
```

Figure 18: “ip a” output when not connect to internet



```
nwhu@ubuntu:~$ ping google.com
PING google.com (142.250.188.14) 56(84) bytes of data.
64 bytes from nuq04s35-in-f14.1e100.net (142.250.188.14): icmp_seq=1 ttl=115 time=29.5 ms
64 bytes from nuq04s35-in-f14.1e100.net (142.250.188.14): icmp_seq=2 ttl=115 time=27.1 ms
64 bytes from nuq04s35-in-f14.1e100.net (142.250.188.14): icmp_seq=3 ttl=115 time=29.8 ms
64 bytes from nuq04s35-in-f14.1e100.net (142.250.188.14): icmp_seq=4 ttl=115 time=23.5 ms
64 bytes from nuq04s35-in-f14.1e100.net (142.250.188.14): icmp_seq=5 ttl=115 time=26.1 ms
64 bytes from nuq04s35-in-f14.1e100.net (142.250.188.14): icmp_seq=6 ttl=115 time=29.7 ms
^C
--- google.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5008ms
rtt min/avg/max/mdev = 23.543/27.622/29.842/2.311 ms
nwhu@ubuntu:~$
```



```
nwhu@ubuntu:~$ ping google.com
ping: google.com: Temporary failure in name resolution
nwhu@ubuntu:~$
```

Figure 19: (a) Pinging with internet connection and (b) without internet connection

Gait Theory

All existing quadruped gait theories or frameworks, except one, examined for this thesis assume that θ_1 and θ_2 (seen in Figure 20) interact independently with respect to the local leg coordinates.

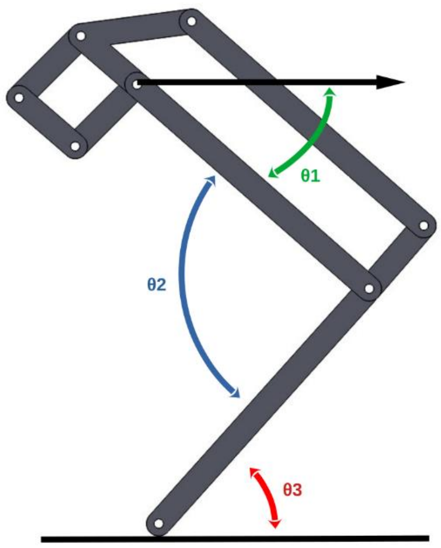


Figure 20: θ1 & θ2 interdependence

This assumption is not met with the leg configuration used by Harry. The angle θ_3 seen in Figure 20 (e.g., the angle between the lower leg segment and ground) is independent of θ1 due to Harry's linkage system. The only previously existing framework found which considers this behavior was implemented by Zenichowski who's robot legs are configured in a similar fashion to Harry's legs. Zenichowski's approach to gait control is similar an approach previously made in the early stages of this thesis. Both approaches use inverse kinematics models and Bézier curves to represent trajectories of the quadrupeds' feet. Zenichowski's inverse kinematics considers the intersection of circles as seen in Figure 21, and the equations can be found in greater detail in Appendix B.

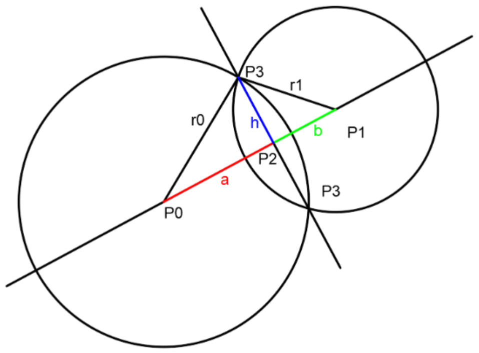


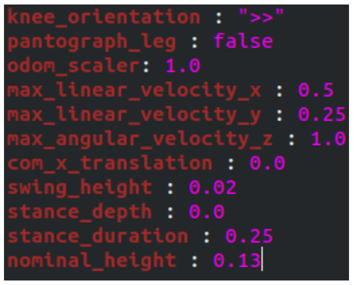
Figure 21: Zenichowski's intersecting circles

This method is based on work presented by Bourke and Voght in both theory and in a C programming language file. Zenichowski repurposed this C code into his Arduino C++ program by considering two circles in the two-dimensional plane in which the lower and upper leg segments rotate in. The hip pitch axis protruded normal from P0 in Figure 21. The foot is located at P1, while r0 and r1 represent the characteristic upper and lower leg lengths respectively. The intersections of these two circles represents the two possible locations for the knee joint. Only one of these branches corresponds to the knee position in real life.

The gait framework developed from scratch early on in this thesis utilizing root finding methods yielded somewhat reasonable results in SolidWorks motion studies when walking in the forwards direction. The equations used in this method can be found in Appendix C. Development of this original gait framework was discontinued after it was decided to search for existing quadruped gait frameworks. Much work was required to make Zenichowski’s framework compatible with Harry. Zenichowski’s framework is specific to a robot using an Arduino as the processing unit, collecting user input from a PSD controller, and is written in C++. Zenichowski's work is highly condensed, and the only documentation comes in the form of C++ comments in the file itself. Making this framework compatible on a Pi using ROS and Ubuntu was trivial. Using C++ as the programming language in the context of Harry should also be trivial, albeit time consuming. It was first desired to switch the input method from a PSD remote (hardware) to ROS’s teleop_twist_keyboard (software). This makes preliminary testing easier by allowing more direct control on the commands being sent to Harry, while still allowing future autonomous directions be sent through twist messages. Using teleop_twist_keyboard (and most other ROS input methods, including autonomous ones) requires both a publisher and a subscriber node. As the names imply, a publisher node issues messages (directions to move in this context), and a subscriber to receive and interpret these messages. The teleop_twist_keyboard acts as the publisher node sending out twist messages containing the next immediate desired direction of motion. The subscriber node should be in the same script that contains the gait framework, so the gait framework has access to the twist messages. Creating a ROS C++ subscriber is possible; however, was not successfully implemented in this thesis. For this reason, rewriting Zenichowski’s C++ gait script in Python was done as creating a subscription node in Python was trivial. The work done based on Zenichowski’s project was not implemented in Harry as the output feet trajectories had many cusps. These cusps are believed to originate from inaccuracies introduced during the Arduino to ROS Python conversion process.

Harry can utilize a variety of gait models. Thus far, the most successful model Harry has implemented is heavily based on Jimeno’s model, which in turn is based on Lee’s gait used by MIT’s cheetah bot. Although not completely integrated in real time, this model allows Harry to transverse on flat surfaces. The process of using Lee's and Jimeno's angles on Harry is described in the following paragraph.

A URDF model with the primary dimensions of Harry was created. These primary dimensions are listed above in Table 1. SolidWorks was used to export .stl files representing each static part of the robot. These .stl files were then imported into Blender for unit scaling and repositioning the origin of each mesh. These mesh files were then exported from blender as .bae files, a common mesh type for URDF models. The main URDF file was modified to combine all parts together in the desired configuration with the appropriate DOFs and inertia values in XML format. Jimeno's automated process was then used to generate the configuration files required to simulate this URDF model in ROS Melodic. The parameters seen in Figure 22 were tuned to yield satisfactory results for Harry. It was then possible to successfully simulate this URDF model (representing Harry) in ROS Melodic with both remote control and autonomous navigation using waypoints. This model can be seen in Figure 23.



```
knee_orientation : ">>"
pantograph_leg : false
odom_scaler: 1.0
max_linear_velocity_x : 0.5
max_linear_velocity_y : 0.25
max_angular_velocity_z : 1.0
com_x_translation : 0.0
swing_height : 0.02
stance_depth : 0.0
stance_duration : 0.25
nominal_height : 0.13
```

Figure 22: Harry's functional parameters

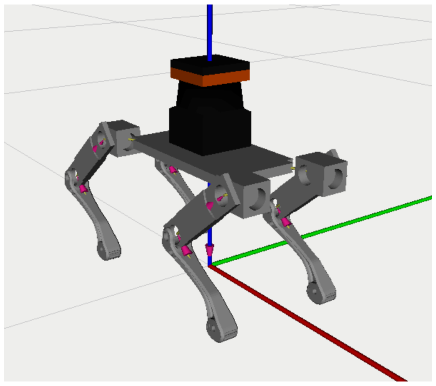


Figure 23: Harry's URDF model in Rviz

A ROS subscription node was created which logs the twelve actuator angles of this simulated model while running it with remote control. The angular values for a single step were then isolated and converted for use in Harry in real life. This was done with Julia scripts for first isolating these angles of interest, finding starting positions, and normalizing the angles to displacement values for use in SolidWorks motion analysis simulations. The required angles for both servos 1 and 2 for each leg were recorded as results of these SolidWorks simulations. This allows for control over the resolution of these stepping angles, uniform coordinate references, and the logging of direct servo 2 angles while avoiding linear approximations or kinematic linkage calculations. Mathematically modeling this kinematics linkage system has also been done but is not required for this application. These equations can be found in Appendix D in the form of a Julia script and may prove useful in future development. The resulting angles collected for the SolidWorks simulations apply directly to the servos; however, mapping must be applied to convert the angles from coordinates referenced in SolidWorks to the unique ranges measured and calibrated for the real-world servos.

Gait testing was completed with Harry’s body mounted upside down in a vise. Walking tests were performed after satisfactory results were observed with Harry’s gait framework running in this orientation.

Lidar

The Rplidar A1 module used collects around 100 data points in polar form per revolution, and spins with a frequency of approximately 10 hertz. Many of these readings fail due to the receiver not detecting the laser signals return; however, a sufficiently clear image is still producible from the data collected from this module. Figure 24 depicts a two-dimensional mapping of a lab room with a rectangular floorplan.

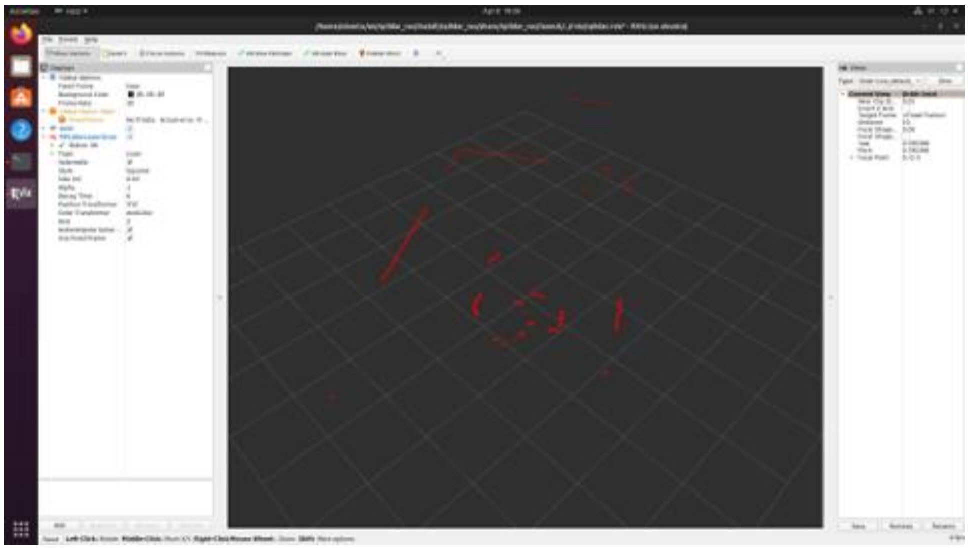
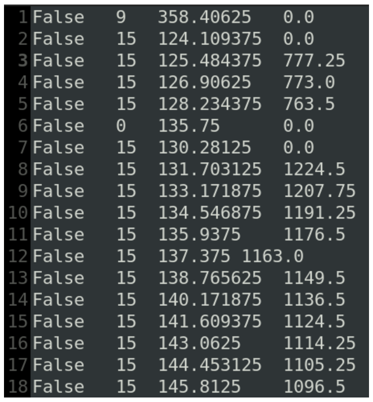


Figure 24: Two-dimensional lidar map in a rectangular room

The center of this mapping is where the Rplidar module is positioned. Approximately half of the room was not visible to the lidar module due to laptops and a portable power supply blocking the line of sight of the module. The range of this module is rated up to ten meters. Interfacing with the lidar was done with a Python module maintained by Robotica, an open robotics collaboration group [49]. A sample of the raw data collected from Rplidar module can be seen in Figure 25.



```
False   9   358.40625   0.0
False   15  124.109375  0.0
False   15  125.484375  777.25
False   15  126.90625   773.0
False   15  128.234375  763.5
False   0   135.75      0.0
False   15  130.28125   0.0
False   15  131.703125  1224.5
False   15  133.171875  1207.75
False   15  134.546875  1191.25
False   15  135.9375    1176.5
False   15  137.375 1163.0
False   15  138.765625  1149.5
False   15  140.171875  1136.5
False   15  141.609375  1124.5
False   15  143.0625    1114.25
False   15  144.453125  1105.25
False   15  145.8125    1096.5
```

Figure 25: Raw Rplidar data

A Python script which collected this data takes only columns 3 and 4 (angular and distance information respectively) and publishes this information as a string message through a ROS publisher. This publisher is subscribed to by a second ROS Python script which translates the string message for each full revolution and determines if an object exists in the region of interest depicted in Figure 26.

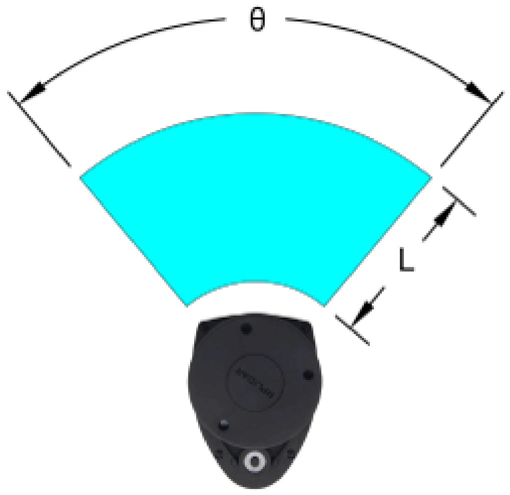


Figure 26: Lidar region of interest

Any objects detected in the region of interest (the annulus segment defined by θ and L) will trigger a message to be sent to the main running ROS Python script to trigger the robot to stop moving until the obstacle is observed to have passed. θ was set to be 90°, and L ranges from (0, 250] millimeters as values of 0 indicated the return signal was not received.

A USB to microUSB cable facilitates communication between lidar module and the Pi. It was found necessary to use a high-quality cable for this purpose as the first two wires utilized proved unfunctional for this application.

RESULTS

SolidWorks motion analysis studies utilizing the first root finding method with Bezerigons yielded somewhat favorable results as seen here.

Simulating Harry’s URDF model in ROS Melodic is successful with both remote control and autonomous movement between waypoints. A video of this can be seen here.

Harry is capable of walking in a linear line while stopping to avoid obstacles in his path detected by his lidar while being powered from external sources. This can be seen here.

CONCLUSION

The work of this thesis has yielded a quadruped robot which utilizes both a unique leg configuration and a currently supported version of ROS. This work is open and accessible on GitHub, and is to the best of the author's knowledge, the first open robot which does not have actuators located at the knee and still utilizes a full computer. This can serve as a platform on which more advanced quadrupeds can be developed which also lack actuators at knee joints.

There are many areas in which vast improvements can be made. Devising an onboard power supply which is capable of fully powering both the robot's servos and onboard circuity to eliminate dependence on external sources to have a fully independent robot would be a great improvement. Configuring a static IP address to allow for simple use of new networks would also be beneficial. Implementing a real-time gait model similar to Cheetah Bot I's instead of using a lookup table method would be beneficial. Investigating other lidar modules or relocating the current attachment point to better position the center of gravity would be beneficial. Adding other sensors such as traditional cameras with object detection, IMUs (inertial measurement unit), and/or accelerometers can be beneficial. Incorporating potentiometers to provide angular displacement for the servos and thus creating a closed loop system would be extremely beneficial.

REFERENCES

REFERENCES

[1] D. D. Chiras, *Human biology*. Burlington, Ma: Jones & Bartlett Learning, 2019.

[2] Klaus-Dieter Budras, *Anatomy of the dog*. Hannover: Schlutersehe, 2010.

[3] C. Beverley and D. Ponsonby, *The anatomy of insects and spiders*. Lewes: Ivy Press, 2007.

[4] D. Mohd Razali, H. Ohroku, and K. Nonami, "1A1-B19 Obstacle Avoidance Control by Laser Range Finder for Six-Legged Robot: SLAM by Laser Range Finder for Six-Legged Robot," *The Proceedings of JSME annual Conference on Robotics and Mechatronics (Robomec)*, vol. 2010, no. 0, pp. _1A1-B19_1_1A1-B19_4, 2010, doi: 10.1299/jsmermd.2010._1a1-b19_1.

[5] S. Kamon, N. Bunathuek, and P. Laksanacharoen, "A Three-Legged Reconfigurable Spherical Robot No.3," *IEEE Xplore*, Aug. 01, 2021. https://ieeexplore.ieee.org/document/9515319

[6] A. Umehara, Y. Yamamoto, H. Nishi, A. Takanishi, and H. Lim, "Jumping pattern generation for one-legged jumping robot," *IEEE Xplore*, Oct. 01, 2017. https://ieeexplore.ieee.org/document/8204211

[7] G. Nelson, K. Blankespoor, and M. Raibert, "Walking BigDog: Insights and challenges from legged robotics," *Journal of Biomechanics*, vol. 39, p. S360, Jan. 2006, doi: 10.1016/s0021-9290(06)84441-6.

[8] "Spot® - The Agile Mobile Robot," *Boston Dynamics*. https://www.bostondynamics.com/products/spot

[9] "Autonomous legged robot for industrial inspection tasks," *Inceptive Mind*, Aug. 24, 2019. https://www.inceptivemind.com/anymal-c-autonomous-four-legged-robot-industrial-inspection/8646/

[10] J. Jimeno, "champ," *GitHub*, Dec. 12, 2021. https://github.com/chvmp/champ

[11] J. Lee, "Hierarchical controller for highly dynamic locomotion utilizing pattern modulation and impedance control: implementation on the MIT Cheetah robot," M.S., MIT, 2013. [Online]. Available: https://dspace.mit.edu/handle/1721.1/85490

[12] S. McRae, "Trajectory Planning For A Quadruped Robot," Thesis, Library and Archives Canada, 2008.

[13] "Astro The Dog-Inspired Quadruped Robot Can Sit, Lie Down, And Learn," *Digital Trends*, Aug. 15, 2019. https://www.digitaltrends.com/cool-tech/astro-dog-robot/

[14] "Pupper — Stanford Student Robotics," *Stanford Student Robotics*. https://stanfordstudentrobotics.org/pupper

[15] Boston Dynamics, "Atlas | Boston Dynamics," *Bostondynamics.com*, 2019. https://www.bostondynamics.com/atlas

[16] B. Alp, "DIY Quadruped Robot," *GrabCAD*. https://grabcad.com/library/diy-quadruped-robot-1

[17] Y. Zenichowski, "Free CAD Designs, Files & 3D Models | The GrabCAD Community Library," *GrabCAD*, 2021. https://grabcad.com/library/walking-quadruped-robot-diy-1

[18] J.-W. Choi and K. Huhtala, "Constrained path optimization with Bézier curve primitives," *IEEE Xplore*, Sep. 01, 2014. https://ieeexplore.ieee.org/abstract/document/6942568

[19] A. Elarabawy, "open-quadruped," *GitHub*, Mar. 30, 2022. https://github.com/adham-elarabawy/open-quadruped

[20] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with Cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 932–950, Jun. 2013, doi: 10.1177/0278364913489205.

[21] P. Biswal and P. Mohanty, "Development of quadruped walking robots: A review," *Ain Shams Engineering Journal*, Dec. 2020, doi: 10.1016/j.asej.2020.11.005.

[22] C. Ridderström, "Legged locomotion: Balance, control and tools — from equation to action," *undefined*, 2003, Available: https://www.semanticscholar.org/paper/Legged-locomotion%3A-Balance%2C-control-and-tools-%E2%80%94-to-Ridderstr%C3%B6m/ec3eeb37b8a7377fccc6f35821f11621b71ae04e

[23] H. Kimura, Y. Fukuoka, and A. H. Cohen, "Adaptive Dynamic Walking of a Quadruped Robot on Natural Ground Based on Biological Concepts," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 475–490, May 2007, doi: 10.1177/0278364907078089.

[24] Z. Chen, "unitree," *unitree*. https://www.unitree.com/products/aliengo/

[25] R. Man, "D'Kitty 12-DOF Quadruped Robot for Studying Locomotion," *Robotic Gizmos*, Mar. 23, 2020. https://www.roboticgizmos.com/dkitty-quadruped-robot/

[26] S. Uehara, "DreamWalker Project," *GitHub*, Mar. 13, 2021. https://github.com/Ohaginia/dream_walker

[27] J. Chu, “Mini cheetah is the first four-legged robot to do a backflip,” *MIT News | Massachusetts Institute of Technology*. https://news.mit.edu/2019/mit-mini-cheetah-first-four-legged-robot-to-backflip-0304

[28] “Open Dog,” *Open Dog*. https://opendog.wordpress.com/

[29] Mike, “Spot Micro Quadruped Project,” *GitHub*, Apr. 12, 2022. https://github.com/mike4192/spotMicro

[30] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, “Learning, planning, and control for quadruped locomotion over challenging terrain,” *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 236–258, Nov. 2010, doi: 10.1177/0278364910388677.

[31] D. D. Mascarenas, J. McClean, C. J. Stull, and C. R. Farrar, “A Preliminary Cyber-Physical Security Assessment of the Robot Operating System (ROS),” *www.osti.gov*, Apr. 30, 2013. https://www.osti.gov/biblio/1078355

[32] S. Rivera and R. State, “Securing Robots: An Integrated Approach for Security Challenges and Monitoring for the Robotic Operating System (ROS),” *IEEE Xplore*, May 01, 2021. https://ieeexplore.ieee.org/document/9463965

[33] Z. Gargulak, “Different Gaits used by Bipeds and Quadrupeds,” *www.ukessays.com*, 2018. https://www.ukessays.com/essays/biology/gaits-bipeds-quadrupeds-8425.php

[34] J. He, J. Shao, G. Sun, and X. Shao, “Survey of Quadruped Robots Coping Strategies in Complex Situations,” *Electronics*, vol. 8, no. 12, p. 1414, Dec. 2019, doi: 10.3390/electronics8121414.

[35] Y. Liu and P. Ben-Tzvi, “DYNAMIC MODELING OF A QUADRUPED WITH A ROBOTIC TAIL USING VIRTUAL WORK PRINCIPLE,” 2018. Available: http://ragazzini.me.vt.edu/pdf/C68_IDETC.pdf

[36] S. Ganjare, V. Narwane, and U. Deole, “Kinematic Modeling of Quadruped Robot | PDF | Kinematics | Velocity,” *Scribd*, 2013. https://www.scribd.com/document/279506868/Kinematic-Modeling-of-Quadruped-Robot

[37] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar, “Inverse Dynamics vs. Forward Dynamics in Direct Transcription Formulations for Trajectory Optimization,” 2021. Available: https://ferrolho.github.io/research/files/ferrolho2021inverse.pdf

[38] R. Mandayam Anand, “Kinematic Analysis Of A Quadruped Robot Capable Of Working As A Machining Tool,” *rc.library.uta.edu*, Nov. 2010, Available: https://rc.library.uta.edu/uta-ir/handle/10106/5157

[39] D. Pieper, “PhD Thesis: The Kinematics Of Manipulators Under Computer Control - Donald L. Pieper,” *www.iri.upc.edu*, 1968. https://www.iri.upc.edu/people/thomas/Collection/details/39172.html

[40] A. Muhammed, V. Sen, M. Bakircioglu, and Kalyoncu, “Inverse Kinematic Analysis Of A Quadruped Robot,” 2017. [Online]. Available: https://www.ijstr.org/final-print/sep2017/Inverse-Kinematic-Analysis-Of-A-Quadruped-Robot.pdf

[41] C. Semini and P.-B. Wieber, “Legged Robots,” *HAL Archives Ouvertes*, 2020. https://hal.archives-ouvertes.fr/hal-02461604

[42] W. Liu, L. Zhou, H. Qian, and Y. Xu, “Turning strategy analysis based on trot gait of a quadruped robot,” *IEEE Xplore*, Dec. 01, 2017. https://ieeexplore.ieee.org/document/8324598

[43] P. Loakeimidis, “Intergration of ROS, Turtlebot, RPLIDAR, RFID technologies and algorithm implementation for navigation and RFID tag detection in a warehouse,” Master’s Thesis, NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS, 2018. [Online]. Available: http://thales-swefs.di.uoa.gr/resources/ThesisIoakeimidisCheck3.pdf

[44] Z. Xuexi, L. Guokun, F. Genping, X. Dongliang, and L. Shiliu, “SLAM Algorithm Analysis of Mobile Robot Based on Lidar,” *IEEE Xplore*, Jul. 01, 2019. https://ieeexplore.ieee.org/abstract/document/8866200

[45] Canonical, “The leading operating system for PCs, IoT devices, servers and the cloud | Ubuntu,” *Ubuntu*, Jun. 19, 2019. https://ubuntu.com/

[46] “ROS 2 Documentation — ROS 2 Documentation: Foxy documentation,” *docs.ros.org*. https://docs.ros.org/en/foxy/index.html

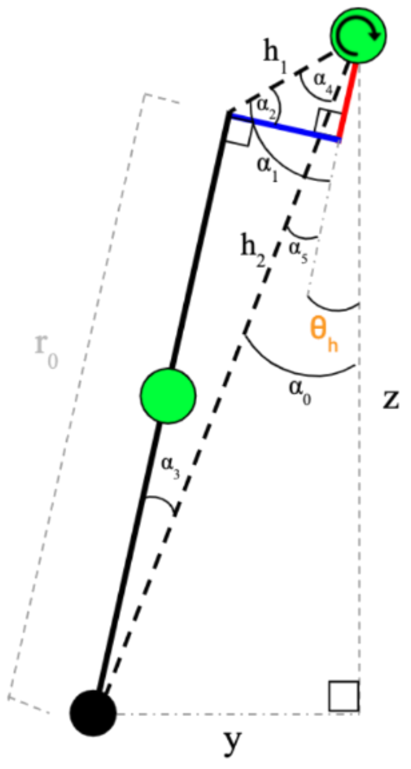
[47] “melodic - ROS Wiki,” *wiki.ros.org*. http://wiki.ros.org/melodic

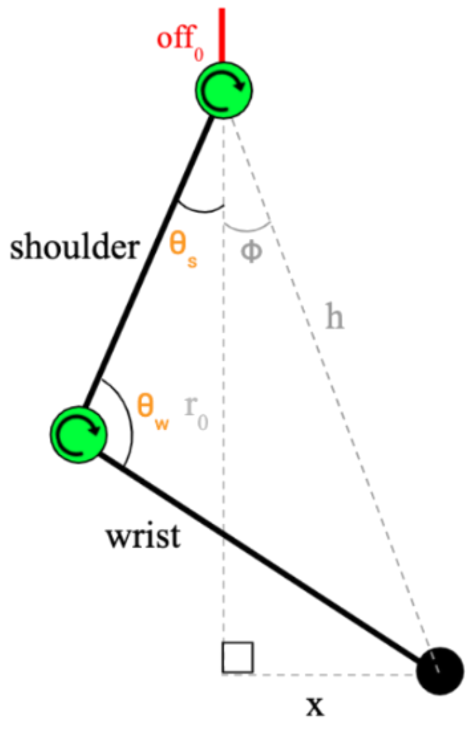
[48] “kinetic - ROS Wiki,” *Ros.org*, 2016. http://wiki.ros.org/kinetic

[49] “Roboticia,” *GitHub*. https://github.com/Roboticia

APPENDICES

APPENDIX A: ELARABAWY’S INVERSE KINEMATICS





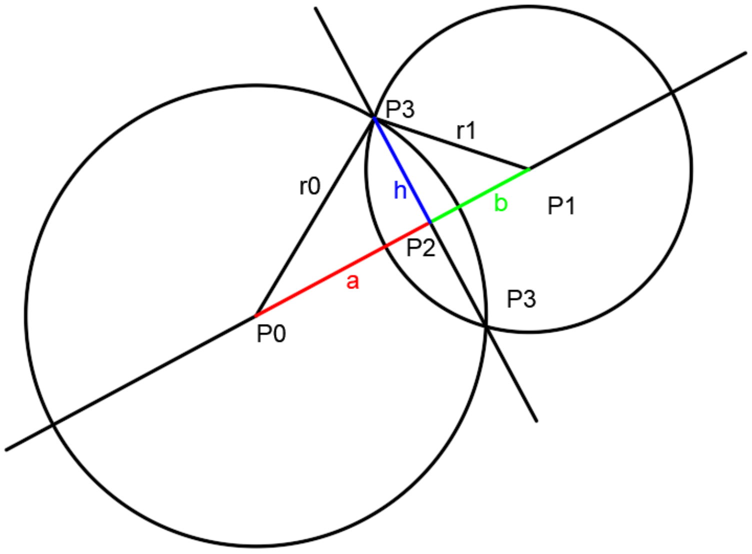
(Ewbarway's figures)

$$\theta_h = \alpha_0 - \alpha_4 \tag{2}$$

$$\theta_s = arccos\left(\frac{h^2 + s^2 - w^2}{2hs}\right) - \phi \tag{3}$$

$$\theta_w = arccos\left(\frac{w^2 + s^2 - h}{2ws}\right) \tag{4}$$

APPENDIX B: ZENICHOWSKI'S INVERSE KINEMATICS



$$d = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \tag{5}$$

$$a = \frac{r0^2 - r1^2 + d^2}{2d} \tag{6}$$

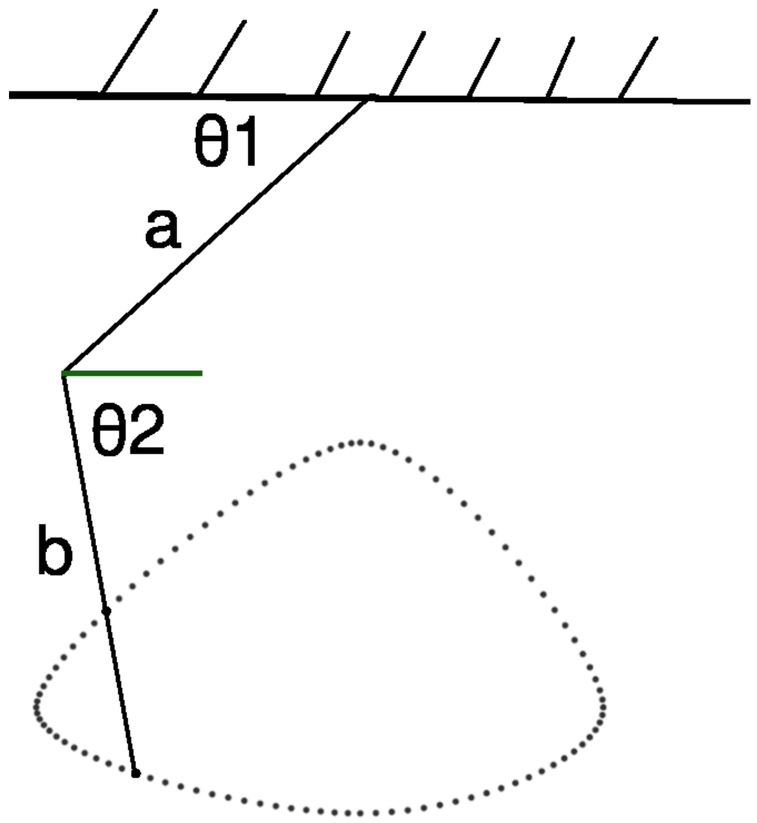
$$h = \sqrt{r_0^2 - a^2} \tag{7}$$

$$P_2 = P_0 + a \times \frac{P_1 - P_0}{d} \tag{8}$$

$$x_3 = x_2 \pm h \times \frac{y_1 - y_0}{d} \tag{9}$$

$$y_3 = y_2 \mp h \times \frac{x_1 - x_0}{d} \tag{10}$$

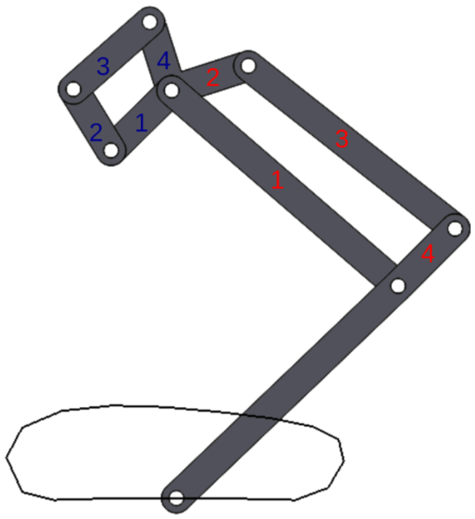
APPENDIX C: ROOT FINDING METHOD



$$f(\theta_1) = a \cdot sin(\theta_1) + b \cdot sin\left[arccos\left(\frac{a \cdot cos(\theta_1) - x}{b}\right)\right] - y \tag{11}$$

$$\theta_2(\theta_1) = arccos\left(\frac{a \cdot cos(\theta_1) - x}{b}\right) \tag{12}$$

APPENDIX D: DOUBLE ROTATING FOUR BAR SYSTEM



```
#!/usr/bin/env julia

struct four_bar
    a
    b
    c
    d
    θ1
    θ2
end

function θ4(bar)
    a,b,c,d,θ2,θ1=bar.a,bar.b,bar.c,bar.d,bar.θ2,bar.θ1
    k1=d/a; k2=d/c; k3=(a^2-b^2+c^2+d^2)/(2a*c)
    A=cosd(θ2)-k1-k2*cosd(θ2)+k3; B=-2*sind(θ2); C=k1-(k2+1)*cosd(θ2)+k3
    return 2atand((-B+sqrt(B^2-4A*C))/(2A))+θ1,2atand((-B-sqrt(B^2-4A*C))/(2A))+θ1
end

function θ3(bar)
    a,b,c,d,θ2,θ1=bar.a,bar.b,bar.c,bar.d,bar.θ2,bar.θ1
    k1=d/a; k4=d/b; k5=(c^2-d^2-a^2-b^2)/(2a*b)
    D=cosd(θ2)-k1+k4*cosd(θ2)+k5; E=-2sind(θ2); F=k1+(k4-1)cosd(θ2)+k5
    return 2atand((-E+sqrt(E^2-4D*F))/(2D))+θ1,2atand((-E-sqrt(E^2-4D*F))/(2D))+θ1
end

system1=four_bar(24,34,24,28.28427125,45,75)
system2=four_bar(27,94,27,100,-45,θ4(system1)[2]-90+45)
println(θ4(system2))

# values for θ2s of both system1 and system2, and θ1 for system1 can be changed for
model other positions

x=system1.d/2*cosd(system1.θ1)+system2.d*cosd(system2.θ1)+102.55*cosd(θ4(system
2)[2]+180)
y=system1.d/2*sind(system1.θ1)+system2.d*sind(system2.θ1)+102.55*sind(θ4(system2
)[2]+180)
println("x = ",x)
println("y = ",y)
```