# A Step Forward: Investigating Dynamic Stability in a Low-Cost Quadruped Robot

Honors Thesis
by
Gabrielle Grace Conard

Advisor: Professor Alexander Brown
Department of Mechanical Engineering
Lafayette College
Easton, PA 18042
May 2021

# A Step Forward: Investigating Dynamic Stability in a Low-Cost Quadruped Robot

by

Gabrielle Grace Conard

May 2021

## Abstract

As robots become more integrated into life outside of the flat, predictable domains of the laboratory, platforms need to be able to navigate uneven terrain and respond to unexpected obstacles and impacts. Quadruped robots, equipped with legs, are particularly well-suited for these tasks. While much progress has been made in the realm of the state-of-the-art, the cost and complexity of these robots prevent students and researchers with limited budgets from accessing these platforms. In response, researchers and hobbyists alike have begun developing low-cost quadruped robots, but their capabilities have been mostly limiting to walking. To address the need for a low-cost quadruped robot that can be used for education and research, MicroDog, a $160 platform consisting of a small set of 3D printed parts and a custom printed circuit board, was developed. This thesis describes this platform's design in detail, as well as the implementation of obstacle avoidance to demonstrate performance on par with current low-cost quadruped technology as well as disturbance recovery to test its usefulness as a research tool.

For my advisor, Professor Brown, who believed in me every step of the way.

# Contents

# Chapter 1

# Introduction

Robots are becoming increasingly woven into the fabric of everyday life. From Roombas that zip around our living room floors to robotic arms that assemble our cars in manufacturing plants, they are shaping how our society operates. However, most successful commercial and consumer robots robots have been mostly limited to the flat, predictable realms of the lab, factory, or home. Venturing outside has proved much more challenging, but is necessary if robots are to be used in applications such as search and rescue [1], planetary exploration [2], and construction site inspections [3]. While wheeled robots are advantageous in certain applications, biologically-inspired legged robots have proved to be superior in traversing uneven terrain. However, since animal locomotion is complex, developing legged robots is no simple task [4]. As a result, the development of legged robots, and especially quadruped robots, have been the subject of extensive research.

One of the primary challenges is developing a robot that can respond to external stimuli, such as unexpected obstacles presented by rough terrain or falls. A number of state-of-the-art quadruped robots, such as Boston Dynamics' commercially available Spot [5], use advanced vision and LIDAR systems to map the surrounding terrain and identify potential footholds. Another method circumvents the need to "see" the environment altogether by sensing the ground reaction forces acting on the legs to inform the gait and react to unexpected obstacles [6, 7]. While a number of quadrupeds use the motors themselves to detect and respond to impacts, such as the MIT Cheetah 3 [6, 8] and Ghost Robotics' Minitaur [9], or motor encoders, such as ATRIAS [7] and Cassie [10], other robots use force sensors. In the case of quadrupeds, these are often installed in the feet, allowing the robot to detect when a foot is in contact with the ground. For example, Biodog uses force sensing resistors to compare the signals produced by the Central Pattern Generator (CPG) to the actual motion of the legs in order to better control Biodog's gait pattern as well as characterize the contact surface friction [11]. Another a quadruped robot developed by the National University of Defense Technology in China uses a com-

bination of contact force sensors (though not described in the paper) and joint position sensors to implement locomotion by evaluating the desired force distribution and changing its behavior when the actual force exceeds the desired force [1].

However, developing proprioceptive sensing and rough terrain navigation in small scale robots, such as those using hobby servo motors, has been largely overlooked. While rapid-prototyped robots are not as powerful as their larger companions, they are important for miniaturizing and reducing the cost of quadruped technology. The complexity and cost of developing quadruped robots can be prohibitive to researchers entering the field [12]. In order to promote progress in the realm of legged robotics, it is imperative that quadruped technology becomes more accessible, allowing earlier access to quadruped robots through inclusion in undergraduate curricula to more ready access to quadrupeds at institutions without the means to purchase an expensive platform.

Recently, some progress has been made to meet this need. Low-cost quadruped robots ranging from over $1000 to as little as $200 have been developed, many of them open-source to encourage others to push the designs even further. However, as reviewed in Section 1.1, the vision for the majority of them was to implement gait patterns such as walking and trotting and to navigate their environments by avoiding obstacles rather than investigating methods of overcoming unexpected obstacles and disturbances. Only one low-cost quadruped robot, which is lacking documentation other than a YouTube video, has successfully implemented proprioceptive sensing to develop active compliance, or programmatic "flexibility" in the joints [13]; another using foot force sensors is still in development [14].

To address the lack of low-cost research-grade quadrupeds, the Lafayette MicroDog was developed: a $160 quadruped robot that is capable of avoiding obstacles autonomously, responding compliantly to vertical ground reaction forces, and using force sensing to maintain dynamic stability. In order to provide better context for this new robot, however, a detailed review of the other low-cost quadruped robots as well as their more expensive counterparts is presented in Section 1.1.

## 1.1 Literature Review

Several other robots have been developed in an attempt to make legged robots more accessible. In the realm of research, a simple insect quadruped robot called robot-K was designed to be more accessible to students [15], and another group developed an amphibious open source robot with the goal of providing a platform for other researchers that would be simple to operate and modify [12]. Neither of these provide the total cost of their

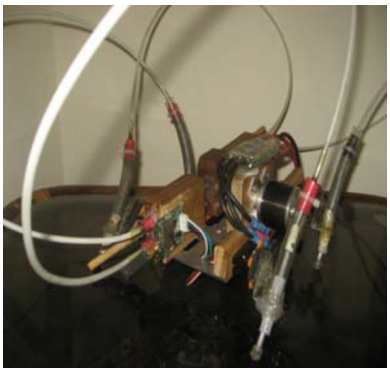


Figure 1.1: robot K [15]

robots, but from the materials described, both likely cost under $200. However, these two examples have few, if any, sensors to observe the environment around them and do not include force sensors on the limbs, making it difficult to navigate unpredictable environments. The robot-K has no sensors [15], which may be one of the reasons why the robot is limited to walking over gaps less than 2 cm deep and on an incline of no more than 5 degrees. The amphibious robot does have an ultrasonic sensor [12], but this still does not provide adequate information about the surface terrain or external disturbances.

MUTT, a $600 quadruped robot developed by undergraduate students at WPI, also attempts to meet the need for low-cost quadrupeds by relying primarily on rapid-prototyping and off-the-shelf parts for constructing the robot [16]. In order to overcome rough terrain, the robot walks around obstacles using a camera and mapping software, but does not have a means to respond to unexpected disturbances. Similarly, another team designed the arachnid-like quadruped, Charlotte, to make research-grade quadruped robots more accessible to researchers with smaller budgets [17]. However, the goal of this $540 robot was to map and navigate its environment using LIDAR, and it does not appear that the group addressed the issue of rough terrain or disturbances.

One of the fastest trotting low-cost robots developed is the Cheetah-cub quadruped robot [18]. This 1.1 kg robot can travel up to 6.9 body lengths per second and traverse a step-down perturbation without external sensors. However, it does this by using passive compliance, a method of creating soft joints by using physical springs or elastic components, rather than artificially creating softness in the motors using software as in active compliance. Passive compliance has proven effective at improving the robustness of quadruped locomotion under specific circumstances. However, since changing the stiffness of the joint involves replacing the spring or changing the design, it is not as useful if one needs different spring constants for different applications, such as a robot that both trots and jumps. In addition, [18] notes that to trot in rough terrain, future designs of the

robot will need to use feedback from sensors to operate.

Figure 1.2: Cheetah-cub [18]

Outside of academia, it appears work on low-cost quadrupeds has been quite active, perhaps motivated by the budgets constraints faced by most hobbyists. A quick survey of products from RobotShop shows that most quadruped kits range from $65 to over $900 and are marketed as an introduction for students students in elementary school up through undergraduate studies to robotics [19]. However, they are not quite as cost effective as they might appear. For example, while the motion-tracking Adeept DarkPaw Quadruped Spider Robot Kit is sold at a reasonable price of $115, it does not come with the Raspberry Pi that it needs to operate and it is difficult to add sensors or make other modifications to the chassis or electronics [20]. As a result, it is not the most practical platform to use for research. On the other hand, the Lynxmotion Phoenix 3DOF Hexapod platform includes only the chassis and legs for about $250 [21], which costs more than a number of the fully-equipped robots investigated in this review.

However, many more capable and cost-effective alternatives to the products of robotics stores have been developed. For example, the Stanford Robotics Club and Extreme Mobility group has developed several quadruped robots, including Stanford Doggo and two versions of the Stanford Pupper platform [22]. At the time of publication, Doggo achieved the second highest vertical jump by a legged robot and set a record for vertical jumping agility, a measure of how quickly an animal (or robot) can change its energetic state [23], but costs nearly $3000. Pupper V1 is an impressive open-source small robot that can walk, creep, and jump for a total cost of $700 to $1250, depending on whether builders buy a kit or buy the parts themselves [24]. In addition to those capabilities, a second version can also scramble over some rough terrain. While not much information has been provided yet as the paper detailing this work is awaiting publication, the group's 2021 International Conference on Robotics and Automation submission video demonstrates Pupper V2 climbing a grassy incline of 20 degrees and scrambling over 2 inch tall beams [25]. Changes in the surrounding terrain are likely detected by measuring the excess torque being applied to its high-power, torque-controllable actuators,

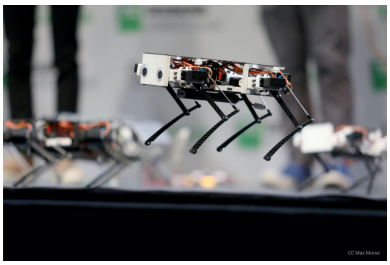


Figure 1.3: Stanford Pupper V1 [24]

a technique used by several state-of-the-art platforms. However, these actuators drive the robot's total cost to nearly $2000 [19]. Both lack sensors (other than an optional inertial measurement unit) and are marketed as inexpensive "hackable" platforms for testing control algorithms.

Another hackable robot, a kitten-inspired quadruped called OpenCat Nybble, has been developed by Petoi LLC [26], and the company has recently released a second quadruped kit, Bittle [27]. Nybble features an ultrasonic sensor for rudimentary vision, and it appears that vision sensors can be added to Bittle, indicating room for some expansion of these platforms. These partially open-source robots are equipped with springs to achieve passive compliance and are controlled using a remote control. The kits for Nybble and Bittle are being sold for $225 and $250, respectively [27, 28].

A community of electrical, computer, and robotics engineers have been developing the SpotMicro, an open-source, low-cost version of the Boston Dynamic's Spot [29]. One of the more recent versions can walk and trot untethered both forwards, backwards, and sideways and is remotely controlled via a laptop [30]; other versions sport LIDAR sensors or robotic arms. From the parts listed on this version's GitHub page, the base platform has an estimated cost of $250 [31]; however, this price does not reflect the cost of additional sensors (such as the LIDAR sensor mentioned above) to meet the goal of autonomous navigation and obstacle avoidance. To aid in traversing rough terrain, they are training the robot in simulation using augmented random search [29]. While this is useful in that it allows the robot to learn how to react appropriately to what its sensors detect, this strategy is only useful in navigating the changes in terrain that the sensors can see, which can be fairly limited if one is using inexpensive IR sensors or cameras as MicroDog does to reduce the cost.

It appears that only one low-cost quadruped has the capability of sensing force and reacting using active compliance [13]. First released in July 2020 by Martin Triendl, there is no other documentation other than several YouTube videos, making it difficult to know how he implemented force sensing. From the parts listed in the description of the videos, the robot costs upwards of $150 [32]. While this is by far one of the most inexpensive robots produced, it does not appear that the platform can be expanded upon without major design

Figure 1.4: SpotMicro [31]

changes. Another quadruped robot is being developed with the goal of incorporating force sensors to respond to its environment as well, developed by Miguel Ayuso Parrilla [14]. At present, this approximately $400 robot is roughly 6 inches tall and 10 inches long. In addition to being capable of walking in all directions, the robot is also has rudimentary disturbance recovery from an impact from the side using its onboard IMU. It does not appear to sense the direction of the disturbance as it simply trots forward in response to an impact, limiting its effectiveness at recovering from the impact.

As demonstrated, work on force-sensing low-cost quadrupeds is just beginning, and there is much to be done to meet the need for inexpensive robots that can recover from unexpected disturbances like they would face outside the laboratory. While a number of larger, more expensive quadrupeds have accomplished this feat, not everything can be exactly replicated due to sensor and actuator cost limitations. For example, using inexpensive hobby servos significantly aids in the reduction of the cost and complexity of the robot (see Section 2.3), a strategy employed by several of the examples above. However, unlike other electric motors, hobby servos cannot be controlled by simply regulating current; rather, they are controlled by sending position commands. This makes it difficult to use current to sense motor torque, from which the external forces can be estimated, and respond appropriately. As a result, using direct drive to detect and respond to impacts, as it was used for controlling the MIT Cheetah 3 [6, 8] and the Ghost Robotics' Minitaur [9], cannot be used in this application; some sort of force sensor is necessary to take in external stimuli. In addition, unlike [33], which aimed to

Figure 1.5: Martin Triendl's "DIY Quadruped" [32]

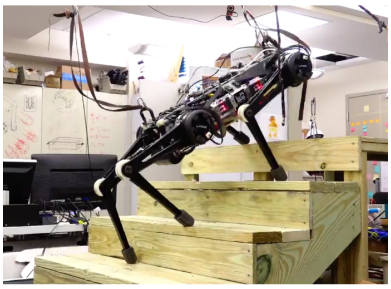


Figure 1.6: MIT Cheetah 3 [8]

mitigate the impact experience by a one-legged parachuting robot while landing by using servo gain to control the stiffness of the joint [34], one is not able to control gain directly on inexpensive hobby servos. In fact, it is not possible even to directly measure the positions of the legs and infer the state of the robot to compare them to the expected positions at a given moment since the potentiometers in hobby servos are not accessible to the user without taking them apart and rewiring them.

However, investigating how state-of-the-art quadrupeds have approached this problem is still likely to be informative when searching for solutions that can be applied to low-cost robots. Several have implemented proprioceptive sensing, some so successfully that other sensors, such as LIDAR or cameras, were not necessary for navigating their environments. For example, the MIT Cheetah 3 quadruped operates under the assumption that the terrain is flat and unobstructed and uses torque control to detect excess torque created by an unseen obstacle and make appropriate adjustments [6]. Much work on dynamic balancing and proprioceptive sensing has also been done in the realm of bipedal robotics. Like MIT Cheetah 3, the passively-compliant bipedal robot ATRIAS [7] and its successor Cassie [10, 35] are effectively blind and rely solely on a gyroscope to determine their orientation and encoders to track joint angles and spring displacements, which provide proprioceptive sensing. Though neither torque control or encoder measurements can be used with low-cost servo motors as discussed before, these robots demonstrate the power of force sensing in navigating uneven terrain navigation and recovering from disturbances. Thus, implementing this capability on an inexpensive platform would be an important step forward in the field of low-cost quadruped robotics.

## 1.2 Methodology and Overview of Thesis

In order to meet the need for low-cost quadruped robots that can be used as research tools, MicroDog was developed over the past year through a series of four design iterations. Three major principles guided this process: staying below a maximum cost of $200 to meet or exceed the current low-cost technology, ensuring reliability

of the platform, and demonstrating that implementing "advanced" features such as force sensing is indeed possible on an inexpensive platform while also leaving room to expand the platform for future research. The final result is a $160 platform equipped with force sensors that can walk, turn, avoid obstacles, and dynamically recover from an outside disturbance.

This thesis is arranged as follows. First, a detailed explanation of the mechanical, electrical, and software design is provided in Chapter 2. In Chapter 3, the development of walking gaits and obstacle avoidance to demonstrate capability on par with that of other robots in this price range. Next, the two components of disturbance recovery, leaning against a force and scrambling to regain balance, will be investigated in Chapters 4 and 5. Finally, the conclusions of this study and suggestions for future work will be given in Chapter 6.

# Chapter 2

# Robot Design

## 2.1 Design Component

The design component of this thesis involved the development of a complex electromechanical robotic platform. This chapter will describe the different facets of the design in detail.

## 2.2 Goals and Constraints

While much progress has been made in quadruped robot research (see Chapter 1), the availability of capable low-cost quadruped robots is still lacking. Thus, a major component of this thesis was the development of such a platform. To meet or exceed the current state-of-the-art, the following constraints and goals shown in Table 2.1 were defined.

To meet these design requirements, MicroDog, shown in Figure 2.1, was developed. The last of four prototypes, this version is approximately $160 and consists of twelve RC servos, a 7.4V 1500mAH LiPo battery pack for untethered walking, and several 3D-printed components. In addition, the custom printed circuit board that serves as the robot's chassis features an integrated ATmega32U4, a Raspberry Pi Zero W, four IR sensors, an Adafruit BNO055 IMU, and a number of power and communication breakout pins and mounting holes to make this an expandable platform. The details of the mechanical and electrical design are discussed in depth below.

Table 2.1: Design Constraints and Resultant Specifications

| **Constraint** | **MicroDog Design Specification** |
|---|---|
| Cost under $200 | $163 |
| Minimize weight and size for portability | Mass: 0.45 kg<br>Width: 12.5 cm<br>Length: 19.0 cm<br>Height: 17.0 cm |
| Operate untethered | Power: 7.4V 1500mAH Battery<br>Battery Life: 27 minutes during walking (assuming current draw of 3.36 A. See Section 2.4.1)<br>Computing: ATmega32U4 and Raspberry Pi Zero W |
| Locomotion: Walk and Turn in all directions | 3 Degrees of Freedom per Leg |
| Obstacle Detection | 4 IR Sensors |
| Proprioceptive Sensing | Hall Effect Feet Force Sensors |
| Room for Expansion | Accessible Power Pins (ground, 3.3V, 5V)<br>6 $I^2C$ Buses<br>2 Digital Pins<br>Screw Holes for Mounting Sensors |

## 2.3 Mechanical Design

One aspect of making a robot more accessible is to reduce not only its cost but also the complexity of the design in order to make it easier to manufacture and assemble. As a result, the robot was designed to require as few mechanical parts as possible. As seen in Figure 2.2, the robot consists of seventeen Polylactic Acid (PLA) 3D-printed components for the legs and battery tray which can be easily produced on a hobby-level 3D printer. To eliminate the need to print and assemble a set of large components for the robot's chassis, a custom printed circuit board was designed (which will be discussed in Section 2.4) to supply not only the robot's electrical needs but also its structural needs as the robot's chassis.

Twelve MG90s 9g micro servos were used to actuate the legs. While these servos are not the most reliable, they were among the cheapest small metal-gear servos available on the market, and investing in more expensive servos would have made it difficult to meet the cost and weight constraints as noted in Table 2.1. For example, Triendl's robot (see Section 1.1) uses MG92B servos [32], which have approximately the same dimensions and mass as MG90s servos but cost nearly three times as much per servo [36]. As a result, the twelve MG92B servos

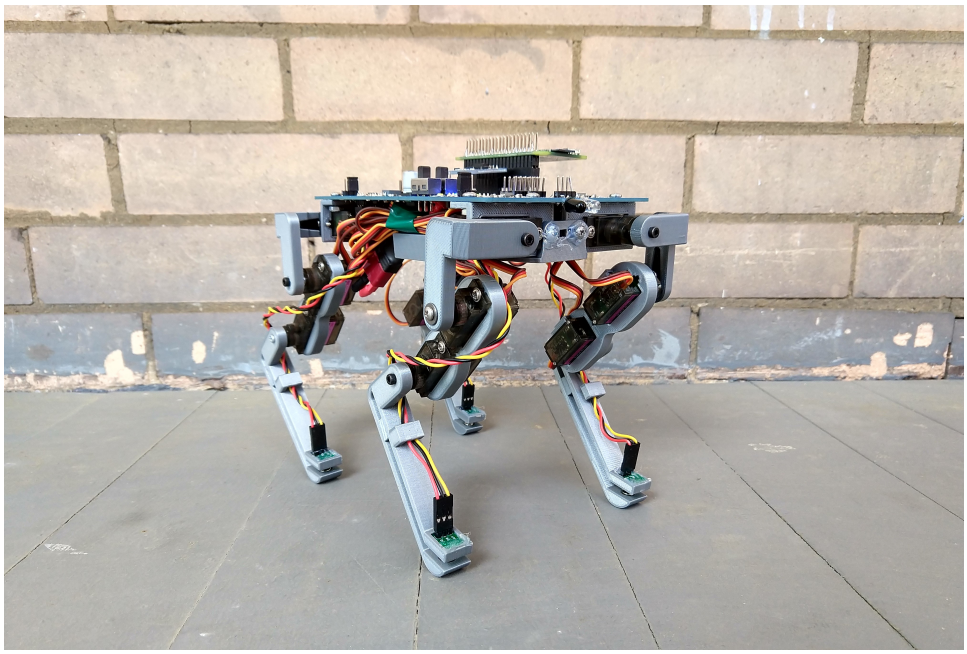


Figure 2.1: MicroDog, a low-cost quadruped robot

alone (at approximately $8.50 each) would cost over $100. While the listed price of the MG90s servo on the same hobby store's website is higher than that of those purchased for MicroDog, MG90s servos are incredibly popular and are widely available through Amazon and other sellers at much lower prices. Looking at Parrilla's robot, the DS3218 servos used not only cost not only cost over four times as much as MicroDog's MG90s servos (even on Amazon), but also have a mass nearly five times that of a MG90s [37]. The minimal cost and weight and wide availability of the MG90s servo made it the optimal choice for this platform. However, their low cost also leads to reduced quality, meaning that special care was needed in operating them to avoid burning them out (see Section 2.4.1 for further discussion).

Three servos are needed to control each leg, which consists of the tibia, the femur, and the hip. This allows each foot to have 3 degrees of freedom (forward/backwards, up/down, abduction/adduction) in order to meet the constraint of being able to walk and turn in all directions. The leg is mounted to the PCB chassis by means of a hip servo mount (see Figure 2.2). In addition, a tray mounted to the bottom of the chassis supports the battery and aids in wire management. The battery, being by far the heaviest component of the robot, was suspended underneath the chassis as close to the center as possible to keep the center of mass near the middle of the robot.

One drawback of using PLA for the 3D-printed legs was that the feet do not maintain much traction on smooth surfaces. In an effort to counteract this, silicone shoes were designed to slid over the bottom half of

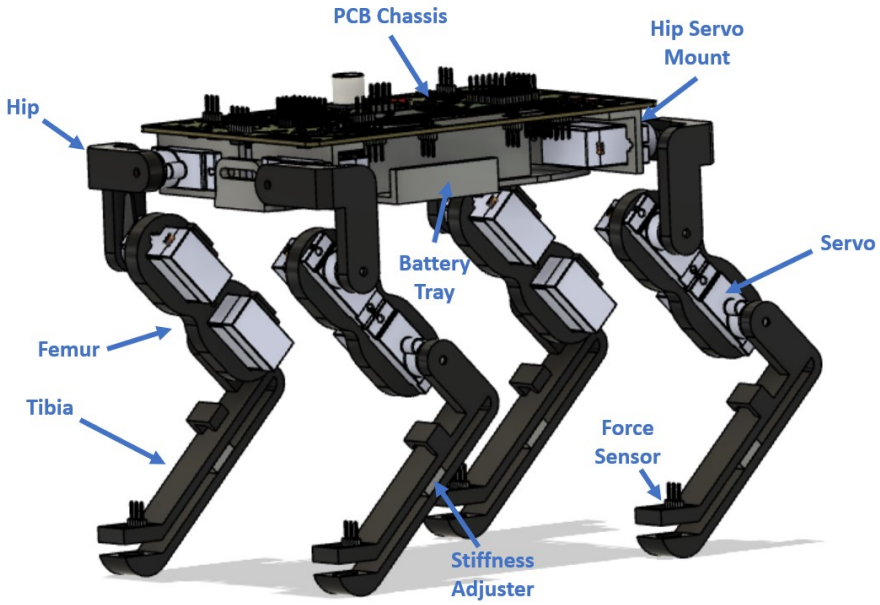



Figure 2.2: CAD Model of MicroDog, with important features noted.

the foot. They were cast in a custom mold, as shown in Figure 2.3, out of Smooth-On Ecoflex 00-50 silicone. While these successfully increased the traction between the feet and the floor, the shoes proved less forgiving of positional errors of the feet. During walking, the inability for the feet to slip actually inhibited locomotion. As a result, the shoes were maintained as an optional feature that can be installed and removed as needed, rather than making them a permanent feature of the robot.

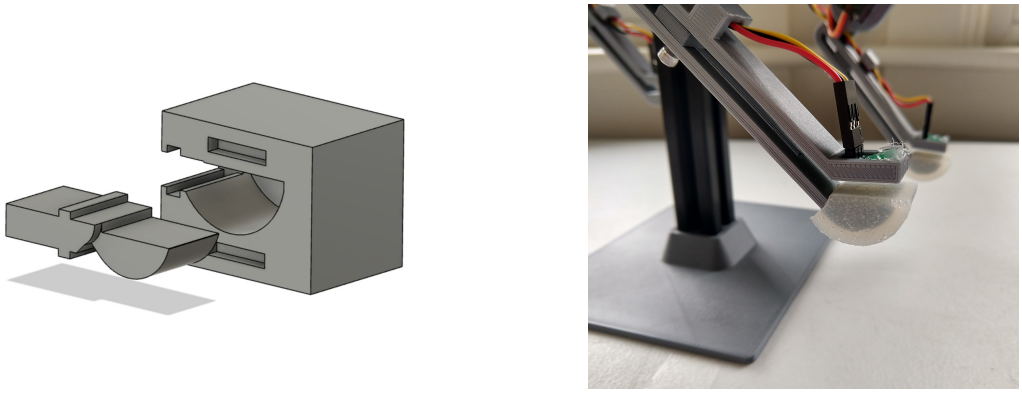


Figure 2.3: Left panel - CAD model of the shoe mold, showing the insert and the outer mold. Not shown are the lid and the bands that secure the lid to the outer mold while the silicone is curing. Right Panel - Finished shoe on MicroDog's foot.

To explore the mechanical aspects of this design further, see Appendix A.3 for a repository of components'

STL files.

## 2.4 Electrical Design

As hinted at in the general design constraints in Table 2.1, the robot required a complex electronic system to operate. The tasks that the electronic system needed to perform included:

- On-board computing
- Regulate battery voltage to 3.3V, 5V, and 6V to power on-board motors, sensors, and computing resources
- Drive 12 servos
- Interface on-board sensors with microcontroller unit (MCU, or the ATmega32U4) and single board computer (SBC, or the Raspberry Pi Zero W)
- Facilitate communication between MCU and SBC
- Allows user to control the robot using buttons and switches
- Provide expansion capability for new sensors, actuators, and/or user interface devices
- Serve as the structural foundation of the robot's chassis

The next two sections will describe in detail how MicroDog's printed circuit board (PCB) meets these needs.

### 2.4.1 Printed Circuit Board: Power, Computing, and Control

While the original prototype of MicroDog featured a collection of separate boards, such as an Arduino Nano, a Raspberry Pi, a voltage regulator, and a servo driver, integrating all of these separate components into one PCB had significant advantages in terms of cost, assembly, and packaging. Table 2.2 demonstrates that the price of just a few of the central components are significantly higher when purchased as separate boards rather than embedded into a single board. In addition, using an integrated PCB reduced the size of the chassis needed to house all of the necessary components and prevented reliability issues that were encountered while working with the first prototype, such as a loose wire causing a board to lose power. Thus, the MicroDog integrated PCB was developed, as shown in Figures 2.4 and 2.5.

Table 2.2: Circuit Board Price Comparison

| **Component** | **Separate Boards** | **Integrated PCB** |
|---|---|---|
| 5V and 6V Voltage Regulators | $48.00 [38] | $7.21 |
| Arduino Nano / ATmega32U4 Circuit | $20.70 [39] | $6.78 |
| PCA9685 Servo Driver | $9.99 [40] | $2.47 |
| **Total Components Cost** | $78.69 | $16.46 |

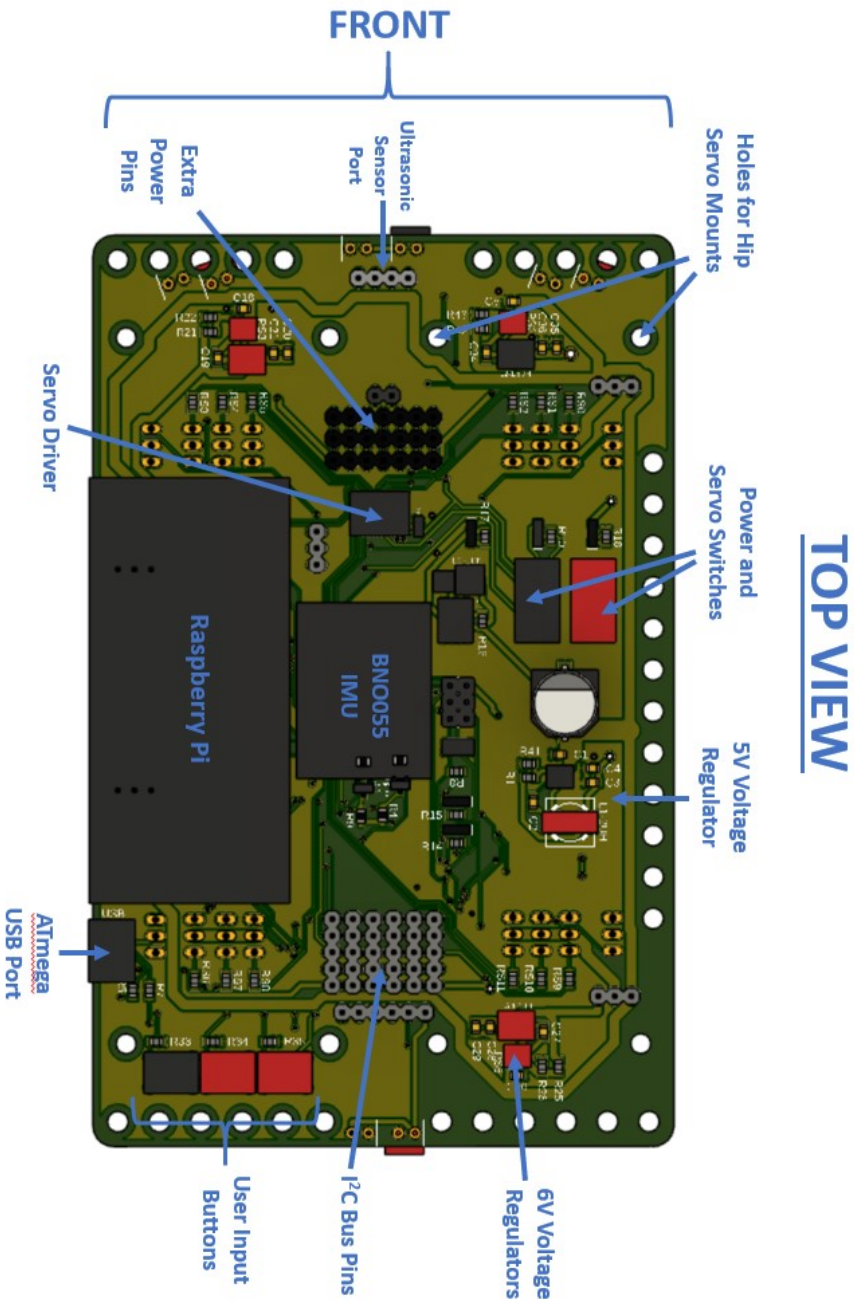



Figure 2.4: Top View of MicroDog's PCB

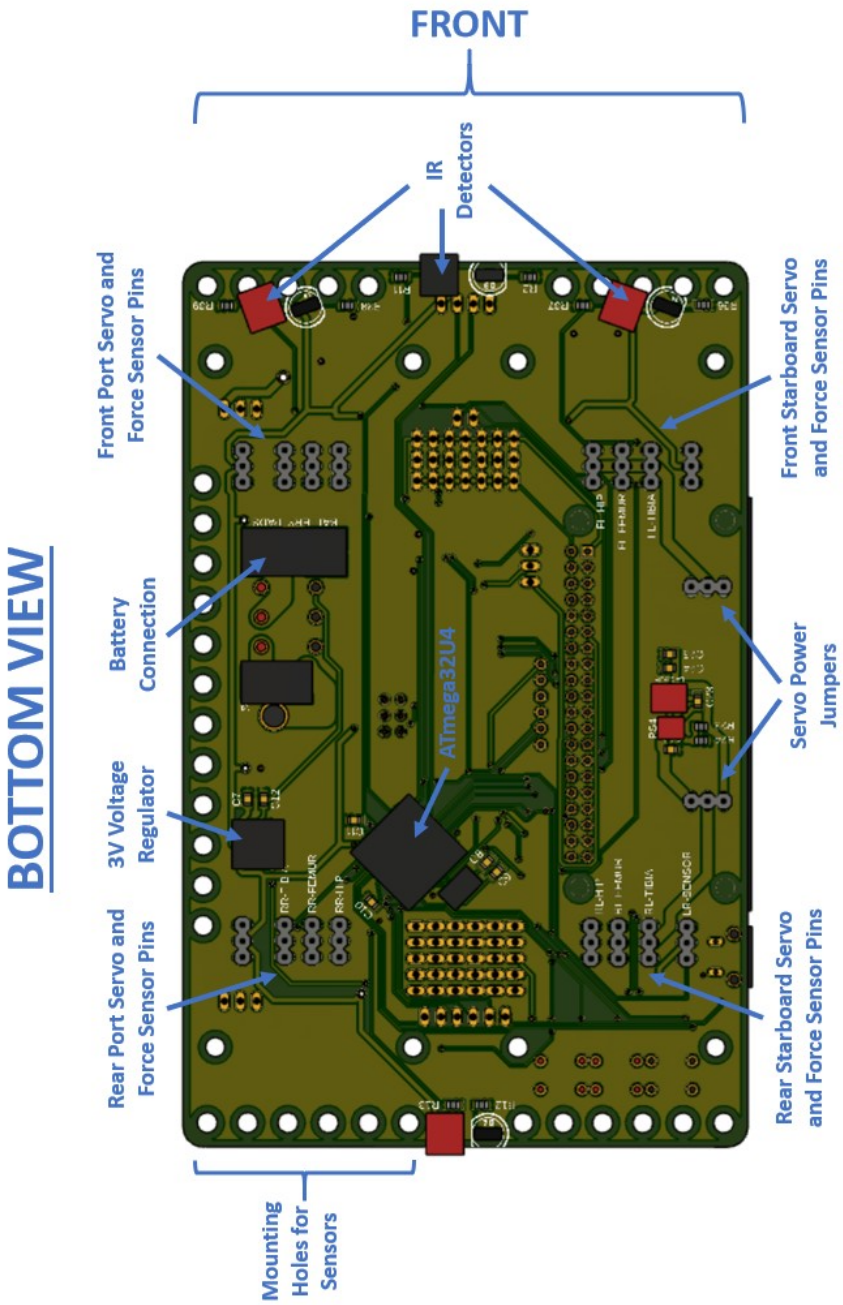



Figure 2.5: Bottom View of MicroDog's PCB

Several principles guided the design of the PCB. In order to make this platform accessible to novice programmers while maintaining high-level computing power, it was decided that both an Arduino-compatible microchip and a Raspberry Pi Zero W would be used. The ATmega32U4, which is the microchip used on several Arduino and Pololu microcontrollers, was integrated into the board and supports Arduino's beginner-friendly subset of the C++ language. However, as a single-board computer, the Raspberry Pi is useful for more complex algorithms and supports Robot Operating System (ROS), which is popular in the robotics field. The Raspberry Pi also allows the robot to have WiFi and Bluetooth connectivity, access the internet in real time, drive a high-quality display, and incorporate vision and LIDAR systems, among other applications. Unlike the ATmega32U4, the Raspberry Pi was not integrated, as its complex design made it more difficult to manufacture and costly to integrate than to simply purchase the board as is. The GPIO breakout headers on the integrated PCB allows the user to easily install the Raspberry Pi if desired.

Since neither the Raspberry Pi nor the ATmega32U4 have the ability to drive the 12 servos, it was necessary to use a separate servo driver. This design uses a PCA9685 chip, which uses pulse width modulation (PWM) to pass position commands to the servos and the Inter-Integrated Circuit ($I^2C$) protocol to communicate with the ATmega32U4 and Raspberry Pi.

As noted in Table 2.1, it was important for the robot to operate untethered to be on par with current technology. This requires that the robot has on-board power and computing. For power, a 7.4 1500mAH LiPo battery was selected, which provides a minimum battery life of approximately 27 minutes if MicroDog is continuously walking during that time [41]. This power source was then regulated down to provide sufficient voltage to each device. The servos required 6V and are sensitive to high voltage. Therefore, four 6V 3A switching regulator circuits (one for each leg) were necessary to protect them from burning out. In addition, a 5V regulator circuit powered the ATmega32U4 and the Raspberry Pi while a subsequent 3.3V regulator circuit powered the PCA9685 servo driver and BNO055 Inertial Measurement Unit. The variety of voltage requirements meant that logic shifters were needed along the $I^2C$ communication lines between the 3.3V and 5V devices.

It was also critical to ensure that sufficient current was available to all the devices, especially when choosing regulators. Thus, an electric current budget was developed, as seen in Table 2.3. Maximum current draw for each device was found from its datasheet or found experimentally, as was the case for the servos. This information informed not only the choice of components but also the width of the traces connecting components on the PCB.

While it would be difficult to measure the current drawn by all of the components to validate the design, it was possible to access the power pins of the servos. As the servos draw far more current than the other

Table 2.3: Electric Current Budget

| Voltage (V) | Device | Quantity | Max Current (A) | Total Device Current (A) | Regulator Current |
|---|---|---|---|---|---|
| 3.3 | PCA9685 Servo Driver | 1 | 0.4 | 0.4 | 0.82 |
| | Blue LED | 1 | 0.003 | 0.003 | |
| | BNO055 IMU | 1 | 0.0123 | 0.0123 | |
| | Additional Sensors | 5 | 0.5 (combined) | 0.5 | |
| 5 | ATmega32U4 | 1 | 0.2 | 0.2 | 2.2 |
| | Raspberry Pi Zero W | 1 | 1.2 | 1.2 | |
| | White LED | 1 | 0.00475 | 0.00475 | |
| | Green LED | 1 | 0.0085 | 0.0085 | |
| | Yellow LED | 1 | 0.009 | 0.009 | |
| | IR Detectors | 4 | 0.03 | 0.12 | |
| | Hall Effect Sensors | 4 | 0.01 | 0.04 | |
| | Additional Sensors | 5 | 1 (combined) | 1 | |
| 6 | MG90s Servos | 12 | 0.7 | 8.4 | 8.4 |
| 7.4 | Green Battery Status LED | 1 | 0.02 | 0.02 | 0.02 |
| **Maximum Total Draw on Battery (A)** | | | | | 7.85 |

components combined, their performance was a good indicator of the validity of the entire current budget. After connecting a power supply to one of the leg's power jumpers, the current drawn by the servos on that leg was observed under three conditions: suspended in air, standing (see Figure 2.6), and walking. As shown in Table 2.4, the maximum required current used by three servos on a leg during walking is far below the stall current of 0.7 A for a single servo, indicating that the power design provides more than sufficient current for all of the components.

Table 2.4: Maximum Current Drawn by Servos

| Condition | Maximum Current of Leg (A) | Maximum Current of All Servos (A) |
|---|---|---|
| In Air | 0.01 | 0.04 |
| Stand | 0.03 | 0.12 |
| Walk | 0.34 | 1.36 |

Thus, the robot's power requirements were validated and its computing chips were ready to communicate with one another and other sensors.

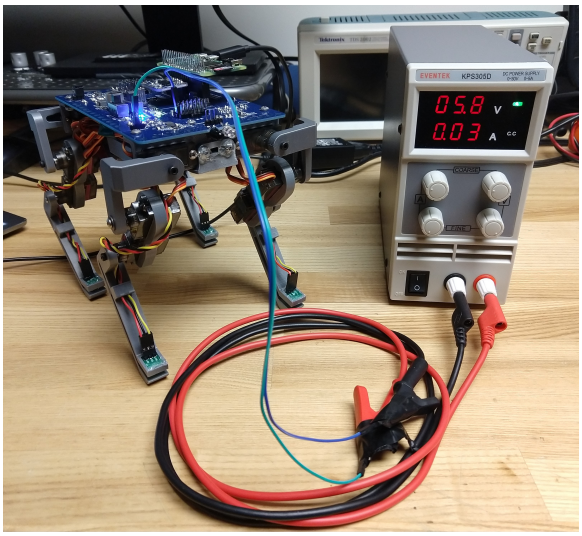



Figure 2.6: Observing the current drawn by one leg while the robot is standing.

### 2.4.2 Printed Circuit Board: User Interface

While the robot's power and computing needs are addressed in the preceding subsection, the PCB also allows a user to interact with the robot. In addition to the power switch, which turns on all of the components, there is a separate Servo Enable switch that enables the servo driver to power the servos (see Figure 2.4). Each switch has a corresponding LED indicator. While the Raspberry Pi can also control this, it was important to maintain this feature should the Raspberry Pi not be used. More importantly, however, the switch serves as an emergency stop should the servos start misbehaving without having to shut the entire robot down. Some other features include a battery status indicating LED that turns off when the battery voltage is low, LED indicators for the ATmega32U4's transmitting and receiving lines, and three buttons that can be read by both the Raspberry Pi and the ATmega32U4.

### 2.4.3 Sensors

Finally, this platform includes several sensors in order to interact with its environment. To implement rudimentary obstacle detection, the robot employs four different IR detectors: one in the rear and three in the front. The three in the front are oriented to allow the robot to detect obstacles directly ahead as well as diagonally to the left and right. (The calibration of the IR sensors will be discussed in Chapter 3.) To detect the robot's orientation and acceleration, an Adafruit BNO005 Inertial Measurement Unit (IMU) is situated near the center of the PCB chassis. This IMU was chosen because it has an advanced on-board sensor fusion algorithm that

provides Euler angle attitude estimates along with raw angular velocity, linear acceleration, and magnetometer measurements.

The feature that sets MicroDog apart from other low-cost quadruped robots, however, is the force sensors, which provide the robot proprioceptive feedback. As seen in Figure 2.8, MicroDog's compressible feet were each equipped with a magnet in the bottom half and a PCB containing a Hall effect sensor in the top half. When force is applied, the foot compresses and the Hall effect sensor measures the relative position of the magnet. The stiffness of the foot, and thus the range of the force sensor, can be increased or decreased by moving the stiffness adjuster down or up the leg, respectively. In Chapter 4, the calibration of these sensors will be discussed.

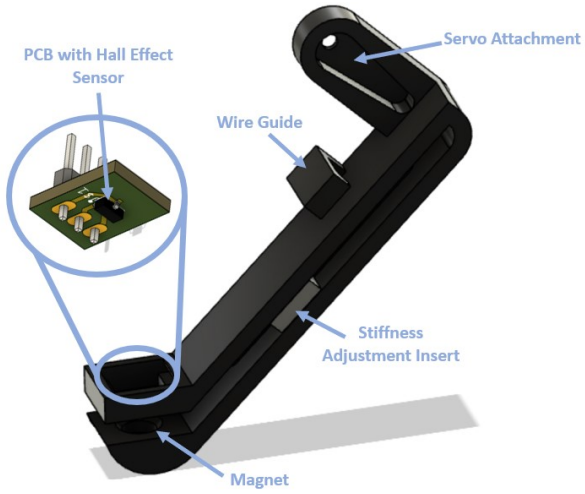



Figure 2.7: Model of tibia with force sensing foot. Inset is a closer look at the force sensor PCB, which includes a Hall effect sensor.

While this platform boasts a number of sensors already, making this board expandable is crucial to making this a useful research tool. Currently, there are designated ports for an ultrasonic sensor, another vision sensor, in the front and an optical flow sensor, which can measure transverse motion, in the rear. In addition to this, there are two digital pins available that connect to the ATmega32U4, three rows of seven pins for 3.3V, 5V, and ground, respectively, and room to connect six additional $I^2C$ devices. Potential peripherals include an LCD screen, an inexpensive camera, LIDAR, additional proximity sensors, and additional analog sensors for measuring limb position, among others.

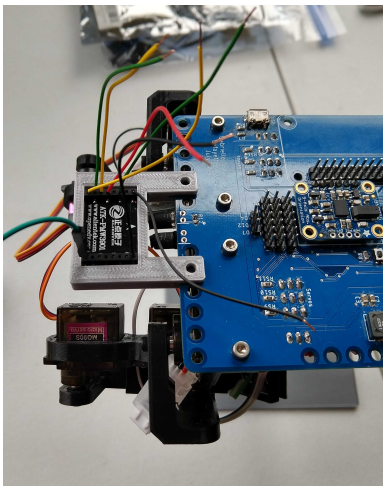


Figure 2.8: As an example of this platform's expansion capabilities, this optical flow sensor was attached using a custom-designed mount and connected to its designated port on the PCB.

## 2.5 Software Design: Inverse Kinematics

The electromechanical design of the robot is vital to its function, but the robot cannot do much without software to coordinate the motion of its twelve joints. While the software and controller designs for different actions such as walking or leaning will be discussed in later chapters, the foundation of every motion the robot makes is the inverse kinematics (IK) software, which transforms requests for the position of an individual foot into requests for the angular positions of each of the three servos comprising each leg. Microdog's IK software was written separately as a library for both Arduino and Python (for use if the Raspberry Pi controls the servos). The libraries can be found here. (See Appendix A.2 for full link.) The layout of the inverse kinematic model for MicroDog is illustrated in Figure 2.9.

For the derivation of the robot's IK equations, based on the angles in Figure 2.9, it was assumed that the femur and hip joint were located at the same point. In the early design, this was physically justifiable because the distance between the two joints was small. Thus, the origin of the leg coordinate system was placed at the femur joint, as indicated by the green axes in Figure 2.9. In order to describe position and motion mathematically, the directions of movement were cast into Cartesian coordinates. In this model, the robot foot moves forward and backward along the $x$ axis, abducts and adducts along the $y$ axis, and lifts up and down along the $z$ axis. The foot position is defined as $P = [x_P, y_P, z_{cmd}]$ relative to the origin at the femur joint, where $z_{cmd}$ is the commanded vertical position of the foot. However, when the foot has lateral displacement as seen in front view of Figure 2.9, $z_{cmd}$ is actually shorter than the distance that the foot needs to be extended to,

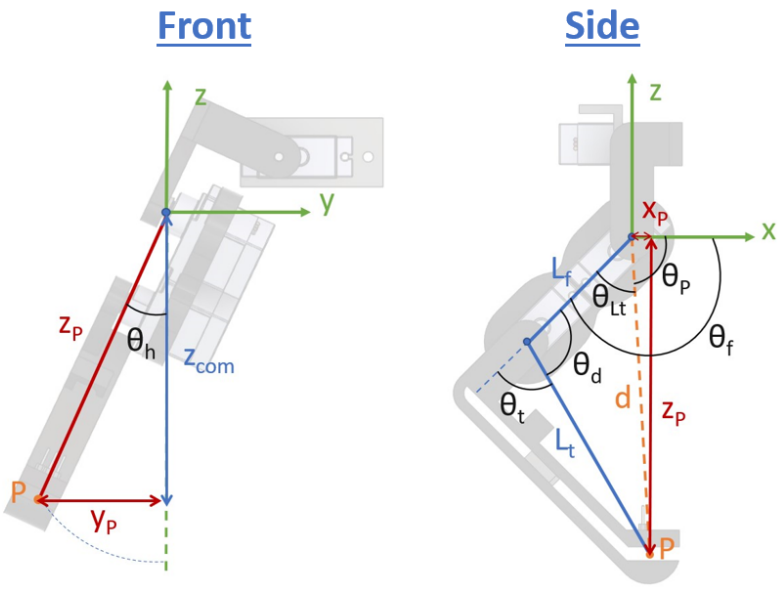



Figure 2.9: Inverse kinematics model for the MicroDog. Note that the separation between the femur joint and the hip joint was neglected in this derivation.

$z_P$. Thus, a coordinate transform is applied to get the actual leg length using the right triangle in the left panel of Figure 2.9.

$$z_p = \frac{z_{cmd}}{\cos(\theta_h)} \tag{2.1}$$

Here, the angle of the hip is defined as:

$$\theta_h = \text{atan2}(\frac{y_P}{z_{cmd}}) \tag{2.2}$$

With this information, the angles of the femur and tibia in the $x_P$-$z_P$ plane (i.e. the side view) can be calculated. First, the distance between the foot and the femur joint, $d$, is given by

$$d = \sqrt{z_P^2 + x_P^2} \tag{2.3}$$

and its angle with respect to the $x$ axis is

$$\theta_P = \text{atan2}(\frac{z_P}{x_P}) \tag{2.4}$$

From the Law of Cosines,

$$\theta_{Lt} = \arccos(\frac{d^2 + L_f^2 - L_t^2}{2dL_f}) \tag{2.5}$$

where $L_f$ and $L_t$ are the modeled lengths of the femur and tibia, as shown in Figure 2.9. With this information, the angle of the femur can be computed.

$$\theta_f = \theta_{Lt} + \theta_P \tag{2.6}$$

To find the tibia angle, one can again use the Law of Cosines to first find its supplementary angle $\theta_d$:

$$\theta_d = \arccos(\frac{L_f^2 + L_t^2 - d^2}{2L_f L_t}) \tag{2.7}$$

Finally,

$$\theta_t = \pi + \theta_d \tag{2.8}$$

In summary, the required servo angles can be found from the requested foot position using the following equations.

$$\theta_h = \text{atan2}(\frac{y_P}{z_{cmd}}) \tag{2.9}$$

$$\theta_f = \arccos(\frac{d^2 + L_f^2 - L_t^2}{2dL_f}) + \text{atan2}(\frac{z_P}{x_P}) \tag{2.10}$$

$$\theta_t = \pi + \arccos(\frac{L_f^2 + L_t^2 - d^2}{2L_f L_t}) \tag{2.11}$$

These equations are implemented in both the Arduino and Python libraries such that if the robot's control system sends a foot command in $x, y, z$ coordinates, the library computes the servo angles required to achieve this foot position. Slight modifications to servo angles and directions are required based on which leg the command is sent to, as the orientation of each servo's 180 degree range must be accounted for. The library's constructor takes the form:

$$\text{leg}_n = \text{Leg3d(side, diagonal, face)}$$

where $n$ indicates the leg. For inverse kinematics, the legs are enumerated as $n = [1, 2, 3, 4] =$ [Front Starboard, Front Port, Rear Port, Rear Starboard]. (Note that the numbering scheme of the legs for the gait

design as presented in Chapter 3 is different than what is used for the inverse kinematics calculations.) The values of the other parameters are either 1 or 2, where side indicates if the leg is on the port or starboard side of the robot, face indicates if it is a front or rear leg, and diagonal, used particularly for the hip servos, indicates if the hip is part of the front starboard or rear port leg ***or*** a part of the front port or rear starboard leg.

With this foundation laid, the robot's feet could now be controlled for a variety of motions, such as walking, turning, and active compliance.

## 2.6 Robot Cost and Thesis Expenditures

Throughout this design process, the key guiding principle was to make a functional platform at a very low price. As discussed in Section 1.1, quadruped robots advertised as "low-cost" can range anywhere from $200 to thousands of dollars. By choosing to use extremely inexpensive hobby servos, 3D-printed parts, custom force sensors, and an integrated PCB, the price of MicroDog was kept below the $200 constraint established at the beginning of this project, as summarized in Table 2.5.

Table 2.5: Robot Cost Break Down

| **Category** | **Cost** |
|---|---|
| Mechanical | $9.53 |
| Servos | $37.49 |
| Electronics - Power | $19.40 |
| Electronics - Connectors | $9.71 |
| Electronics - Sensors | $46.30 |
| Electronics - Computing | $25.63 |
| Electronics - PCB and User Interface | $15.55 |
| **Total Cost** | $163.59 |

In the development of this final platform, four different prototypes were produced since the spring of 2020; the final iteration was developed over the course of this thesis. Two copies of each design were developed so that this project could be run efficiently in a remote learning environment. A brief overview of the expenditures over the course of the 2020-2021 school year is outlined below.

For a complete bill of materials for the robot, as well as a more detailed look at total expenses, please reference this spreadsheet. (The full link is available in Appendix A.1.)

Table 2.6: Total Thesis Budget

| **Category** | **Cost** |
|---|---|
| Mechanical | $214.47 |
| Electronics - Power | $101.48 |
| Electronics - Connectors | $52.72 |
| Electronics - Sensors | $447.51 |
| Electronics - Computing | $127.14 |
| Electronics - PCB and User Interface | $176.82 |
| **Total Cost** | $1120.14 |

## 2.7 Conclusion

While this design met the established constraints, as demonstrated in this chapter, it was necessary to investigate whether this platform can truly be a useful research tool. The next chapters demonstrate MicroDog's range of capabilities and topics for further exploration, starting with locomotion and obstacle avoidance.

# Chapter 3

# Autonomous Navigation: Obstacle Avoidance

Central to a quadruped robot is its ability to walk and navigate its environment. All of the quadrupeds reviewed in Chapter 1 have some ability to locomote, whether that entails a "simple" motion such as walking or more advanced maneuvers such as galloping or jumping. In addition, a few of the low-cost quadruped robots had some means, such as IR detectors or even LIDAR, to detect and avoid obstacles in their environment [16, 29]. Thus, the next step was to develop algorithms that would enable MicroDog to do the same in order to demonstrate performance on par with its peers.

## 3.1 Methodology

### 3.1.1 Software Design: Walking and Turning

As discussed in Section 2.5, the first step towards locomotion was deriving the inverse kinematics model of MicroDog. This model converted given foot position commands into the corresponding servo angle commands. However, the appropriate position trajectories of the feet necessary for different maneuvers still needed to be determined.

Looking to nature for inspiration, roboticists have attempted to replicate the gait patterns of legged animals on legged robots, which describe the foot swing and placement patterns. For the purpose of this study, it was decided that the walking gait would be sufficient for implementing autonomous navigation. During walking, at least three feet are on the ground at any given time. Each foot, then, experiences two phases: stance phase, in which it is on the ground, and swing phase, in which it is swinging through the air. The gait cycle in Figure 3.1, which was developed for Boston Dynamics' Little-Dog by researchers at the University of Southern California, illustrates the order in which each leg swings forward and includes time between each swing phase for all four

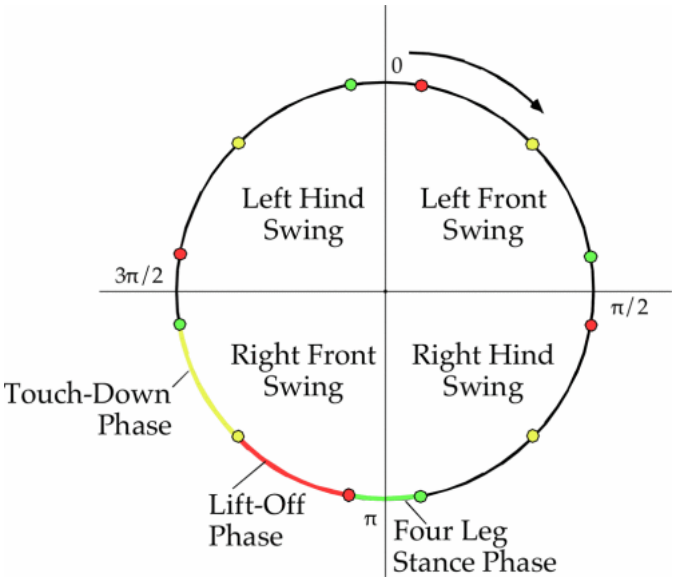



Figure 3.1: The gait cycle for Little-Dog's walk, which illustrates the swing leg order [42]

feet to be on the ground [42]. While the feet trajectories were not discussed in [42], the timing and order of the each foot's swing phase served as a good starting point for developing MicroDog's gait. Other papers do discuss methods of developing foot trajectories, but oftentimes use complicated methods such as neural networks or central pattern generators. While it may be possible to implement these on MicroDog, the purpose of this investigation was to implement a simple walking gait to demonstrate the capabilities of this platform. As a result, those methods were not used.

The "Four Leg Stance Phase" in which all four feet were in stance phase was necessary for Little-Dog to adjust its center of gravity as it climbed extremely uneven terrain; however, it is not needed for walking over flat ground or mildly rough terrain, which is what MicroDog would be starting on. As a result, MicroDog's walking gait would involve each leg's swing phase lasting a quarter of the total gait cycle, while each leg's stance phase would last 3/4 of the cycle. Quantifying the duration of the gait cycle and the two phases each foot experiences during the gait cycle in terms of phase angle,

$$\phi_{gait} = 2\pi \tag{3.1}$$

$$\phi_{swing} = \frac{1}{4}\phi_{gait} = \frac{1}{2}\pi \tag{3.2}$$

$$\phi_{stance} = \frac{3}{4}\phi_{gait} = \frac{3}{2}\pi \tag{3.3}$$

Since elapsed time can be measured readily, it is useful to describe the gait in terms of time and frequency.

From the phase angles of swing and stance above, the length of time in each phase can be described as:

$$T_{swing} = \frac{1}{2} T_{gait} \tag{3.4}$$

$$T_{stance} = \frac{3}{2} T_{gait} \tag{3.5}$$

Thus, the frequencies of these phases are given by:

$$\begin{aligned} \omega_{swing} &= \frac{2\pi}{T_{stance}} \\ &= \frac{2\pi}{\frac{1}{2} T_{gait}} \\ &= \frac{2\pi}{\frac{1}{2} \frac{2\pi}{\omega_{gait}}} \\ &= 2\omega_{gait} \end{aligned} \tag{3.6}$$

$$\begin{aligned} \omega_{stance} &= \frac{2\pi}{T_{stance}} \\ &= \frac{2\pi}{\frac{3}{2} T_{gait}} \\ &= \frac{2\pi}{\frac{3}{2} \frac{2\pi}{\omega_{gait}}} \\ &= \frac{2}{3} \omega_{gait} \end{aligned} \tag{3.7}$$

To determine when each leg would start swinging, a threshold was developed in terms of the phase angle. Swing threshold is defined as:

$$\phi_{thres} = \frac{3\pi}{2} + n\frac{\pi}{2} \tag{3.8}$$

where $n$ indicates the swinging leg. In the case of walking, $n = [1, 2, 3, 4]$ = [Front Port, Rear Starboard, Front Starboard, Rear Port].

For each leg, the gait cycle is divided into three parts, as illustrated in Figure 3.2: stance phase before swinging, swing phase, and stance phase after swinging. This last part must transition smoothly back to stance before swinging to ensure a smooth walking gait. To implement this, the phase at a given moment in time, $\omega_{gait} t$, was compared to the swing threshold $\phi_{thres}$ and swing duration $\phi_{swing}$. This determined which sinusoidal trajectory to use.

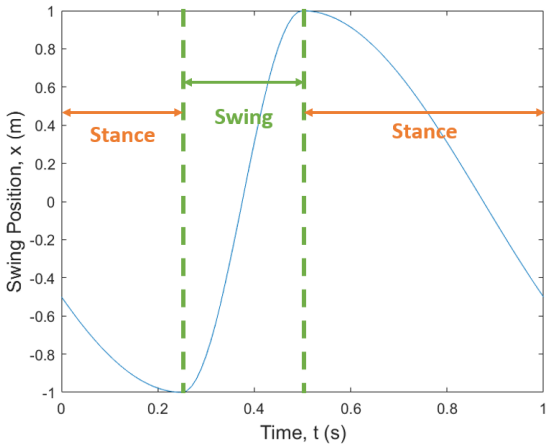



Figure 3.2: Trajectory of Rear Starboard Foot

For forward or backwards walking, the foot position in $x$ was defined by the following piecewise function:

$$x = \begin{cases} \cos(\frac{2}{3}\omega_{gait}t - \frac{2}{3}(n\phi_{base} + pi)) & \omega_{gait}t \leq \phi_{thres} \\ \cos(2\omega_{gait}t - 2n\phi_{base}) & \phi_{thres} < \omega_{gait}t < \phi_{thres} + \phi_{swing} \\ \cos(\frac{2}{3}\omega_{gait}t - \frac{2}{3}(n\phi_{base})) & \phi_{thres} + \phi_{swing} \leq \omega_{gait}t \end{cases} \tag{3.9}$$

For turning or sideways walking, the lateral foot trajectory was defined similarly.

$$y = \begin{cases} \cos(\frac{2}{3}\omega_{gait}t - \frac{2}{3}(n\phi_{base} + pi)) & \omega_{gait}t \leq \phi_{thres} \\ \cos(2\omega_{gait}t - 2n\phi_{base}) & \phi_{thres} < \omega_{gait}t < \phi_{thres} + \phi_{swing} \\ \cos(\frac{2}{3}\omega_{gait}t - \frac{2}{3}(n\phi_{base})) & \phi_{thres} + \phi_{swing} \leq \omega_{gait}t \end{cases} \tag{3.10}$$

Finally, the height of the foot, $z$, was found using the following equations. Note that while the foot is in stance phase, the height is set to 0 to achieve a flat trajectory along the ground and prevent the foot from attempting to push through the ground in an attempt to follow the sinusoid, potentially destabilizing the robot.

$$z = \begin{cases} 0 & \omega_{gait}t \leq \phi_{thres} \\ \cos(2\omega_{gait}t - 2(n+3)\phi_{base}) & \phi_{thres} < \omega_{gait}t < \phi_{thres} + \phi_{swing} \\ 0 & \phi_{thres} + \phi_{swing} \leq \omega_{gait}t \end{cases} \tag{3.11}$$

An illustration of the sinusoidal trajectories for the front port foot during forward walking is presented in

Figure 3.3.

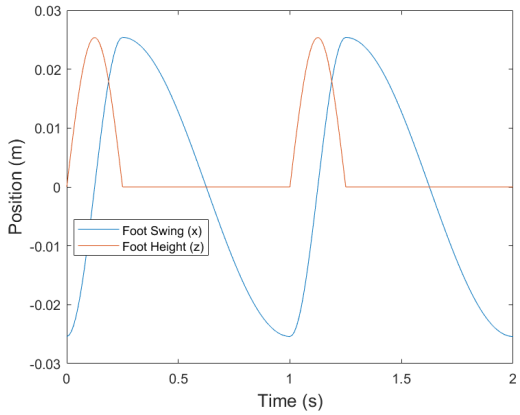
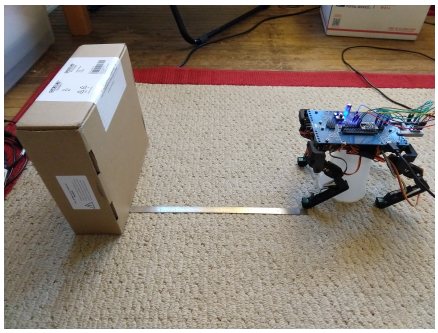



Figure 3.3: Modified sinusoidal commands for forward walking.

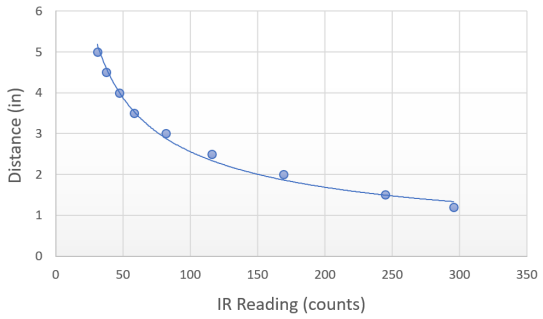
To implement this, the $[x, y, z]$ position at a given moment in time passed into the inverse kinematics (IK) software, which then converts the position to servo angles. The walking and turning gaits described above allowed for autonomous navigation using the robot's IR sensors.

### 3.1.2 IR Sensor Calibration

In order to detect obstacles in its environment, MicroDog is equipped with four IR detectors. However, these needed to be calibrated to convert the sensor counts into meaningful units of distance. This was done by setting a box at a different known distances from MicroDog and recording the corresponding IR sensor reading. The results of this calibration experiment are shown in Figure 3.4.



(a) Taking IR sensor data at set distances from a box.




(b) Calibration curve translating the IR sensor readings into distance.

Figure 3.4: IR Calibration

This gave the relationship,

$$d = 40.571i^{-0.6} \tag{3.12}$$

where $i$ is the IR sensor reading in counts and $d$ is the distance in inches.

### 3.1.3 Obstacle Avoidance

When navigating its environment, there are two major categories of obstacles that MicroDog must avoid: a large drop-off, such as the edge of a table, or a step or wall that is too large to climb over. While it is hoped that MicroDog will eventually be able to recognize obstacles that it can climb over, as shown in Figure 3.5, for the purpose of demonstrating capabilities on par with current low-cost quadruped technology, only avoidance of the edge of a table was investigated.

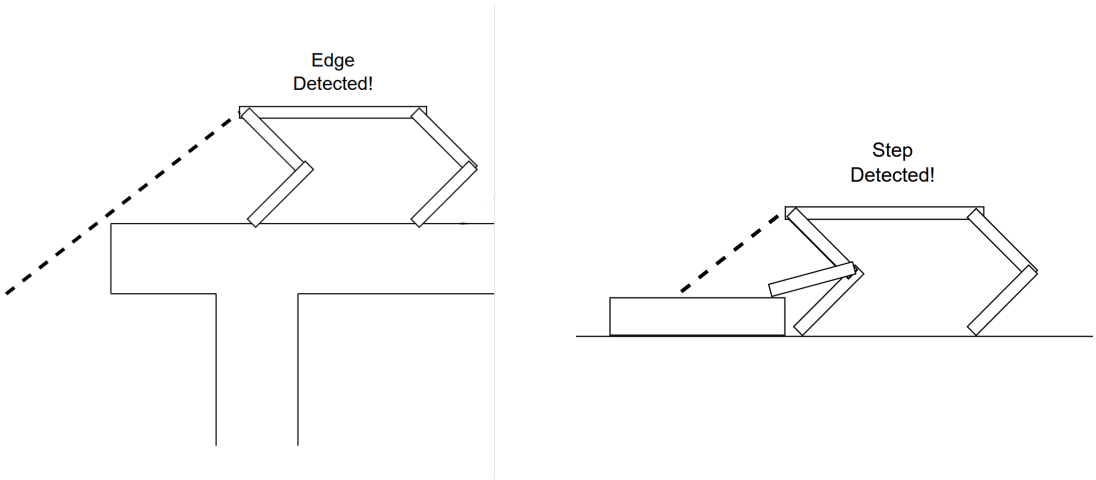



Figure 3.5: Examples of Obstacle Detection

To implement this, the state machine shown in Figure 3.6 was developed, and its corresponding table detailing the transitions is presented in Table 3.1. MicroDog is typically in the Fwd state, which represents forward walking. When the IR sensors, which are aimed at a 45 degree angle towards the table surface, detect that the distance has suddenly increased beyond some threshold, MicroDog interprets that as a table edge. In response, MicroDog backs away from the edge for a period of time before turning right or left to begin walking in a new direction.

At the time this state machine was implemented, the design of the MicroDog platform was only in its third iteration, which was equipped with only two IR sensors: one in the front and one in the rear. This means that

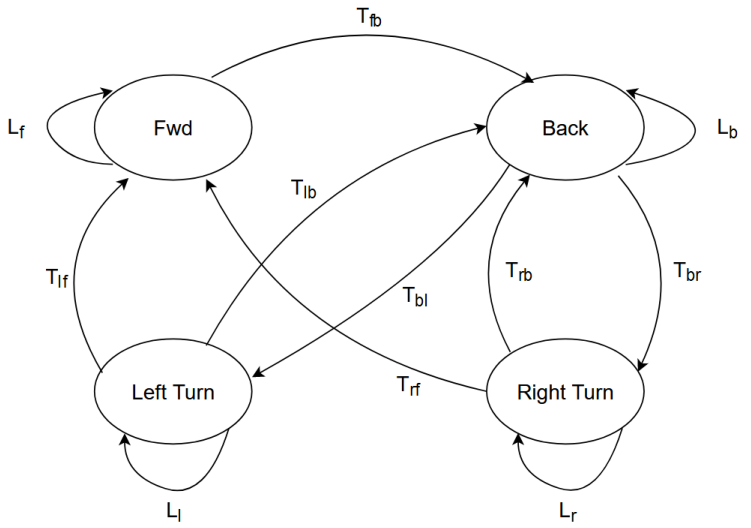



Figure 3.6: Diagram of Obstacle State Machine

| Transition | Start State | Condition | End State |
|---|---|---|---|
| $L_f$ | Forward (Fwd) | No Obstacle Detected | Fwd |
| $T_{fb}$ | Fwd | Obstacle Detected | Back |
| $L_b$ | Back | Back Timer Incomplete | Back |
| $T_{br}$ | Back | Back Timer Complete and Chose Right | Right Turn |
| $T_{bl}$ | Back | Back Timer Complete and Chose Left | Left Turn |
| $L_r$ | Right Turn | Turn Timer Incomplete and No Obstacle Detected | Right Turn |
| $L_l$ | Left Turn | Turn Timer Incomplete and No Obstacle Detected | Left Turn |
| $T_{rb}$ | Right Turn | Obstacle Detected | Back |
| $T_{lb}$ | Left Turn | Obstacle Detected | Back |
| $T_{rf}$ | Right Turn | Turn Timer Complete | Fwd |
| $T_{lf}$ | Left Turn | Turn Timer Complete | Fwd |

Table 3.1: Obstacle Avoidance State Transition Table

it could not make an informed choice about what way to turn to best avoid the edge of the table. Thus, the direction was chosen randomly. Now that the final design has two additional IR sensors in the front directed towards the right and left, further work could include the modification of the state machine to better navigate its environment by avoiding obstacles on its sides.

### 3.1.4 Validation

To validate the obstacle avoidance state machine, MicroDog was placed on a table and allowed to roam about freely. During testing, MicroDog successfully detected the table edges and triggered backwards walking. However, the robot tended to drift during walking, and turning was not the most effective. MicroDog exhibited much

instability as it tended to tilt towards the corner unsupported by its swinging leg. Further work is needed to identify methods for making the walking and turning gaits more stable, such as investigating the effects of walking speed and sinusoid amplitudes (as discussed in regards to scrambling in Chapter 5.

Another challenge that was faced was defining the threshold distance. The IR sensors are dependent upon lighting conditions and the reflectivity of the surface, which was not accounted for during calibration. Ideally, MicroDog would respond to the obstacle about 3 inches away from the edge. Under the lighting conditions similar to those used during calibration, this corresponded to approximately 100 counts. However, under brighter conditions and operating on a smoother surface, the count threshold had to be decreased to 5. As a result, further work is needed then to allow MicroDog to navigate environments with different lighting and surface conditions.

# Chapter 4

# Disturbance Recovery: Lean Control

The ability for quadruped robots to recover from an unexpected disturbance, such as a sideways push, and to maintain dynamic stability more generally, has been an important focus in robotics research. Though not much as been published on its control software, perhaps one of the most popular examples demonstrating this capability is Spot from Boston Dynamics, which can quickly side-step to regain its balance after being kicked [43]. While a large force such as a kick does require some amount of side-stepping to recover (which will be discussed in Chapter 5), tipping due to smaller disturbances can be counteracted by having the robot lean into the force. Thus, the first step towards disturbance recovery is to control the robot's lean in response to a force.

A common method that has been used to maintain dynamic stability in general, such as while walking, is Zero Moment Point Control [44–46]. The Zero Moment Point (ZMP) is an imaginary point on the ground about which the moments applied to the robot sum to 0. If the ZMP goes outside of the support polygon (that is, the area enclosed by the feet on the ground), then the robot will begin to tip. Under the assumption that the robot is well-approximated by a point mass at a constant height $L$ above flat terrain, it can be shown that the $x$ and $y$ positions of the ZMP can be found using the position and acceleration of the center of gravity (CG), as given by [44]:

$$P_{ZMP,x} = x_{CG} - \frac{L}{g}\ddot{x_{CG}} \tag{4.1}$$

$$P_{ZMP,y} = y_{CG} - \frac{L}{g}\ddot{y_{CG}} \tag{4.2}$$

where $L$ is the height of the CG above the ground (assumed to be constant) and $g$ is the acceleration due to

gravity.

However, as [46] explains well, the CG position and acceleration terms are "at odds" with one another due to the difference in sign. As a result, a classical controller could become unstable when high gains are used. To solve this problem, many researchers (including all the ones that use ZMP noted) implement preview control by using a reference trajectory.

Though preview control is a powerful tool, this will not suffice for push recovery, since the unexpected disturbance means that there is not a useful trajectory to follow. Thus, a different method was used. The relative readings between MicroDog's port and starboard force sensors give insight into the robot's state at any given moment. Focusing specifically on the case of a horizontal disturbance (that is, a force applied on the robot's side), a controller was designed to respond to changes in the force read by the feet.

## 4.1 Methodology

Several steps were taken to design the controller. First, a physics-based model of the robot was made to understand its behavior in response to disturbances. This model was validated, when possible, by observing this open loop response. Based on this model, the controller was designed and its theoretical response was simulated in MATLAB. Before implementing it on the physical robot, however, the controller was also validated in Webots, an open-source robot simulation environment, which minimized the risk of damaging the physical robot during controller testing. Using Webots also allowed for faster debugging and iterative testing. Once this validation was performed, the controller was tested on the real MicroDog, and its response was compared to those of the MATLAB and Webots simulated models.

### 4.1.1 Mechanical and Electrical Design: Force Sensors

In order to use the force sensors to measure the disturbance and implement the lean controller, the sensors needed to be characterized to convert their output from counts to units of force (Newtons). This calibration was performed by applying known weights to one corner of the robot, observing the corresponding sensor reading in counts, and curve fitting to find the relationship between counts and force, as seen in Figure 4.1.

This produced the relationship,

$$F = -0.0005c^2 + 0.2042c - 18.365 \tag{4.3}$$

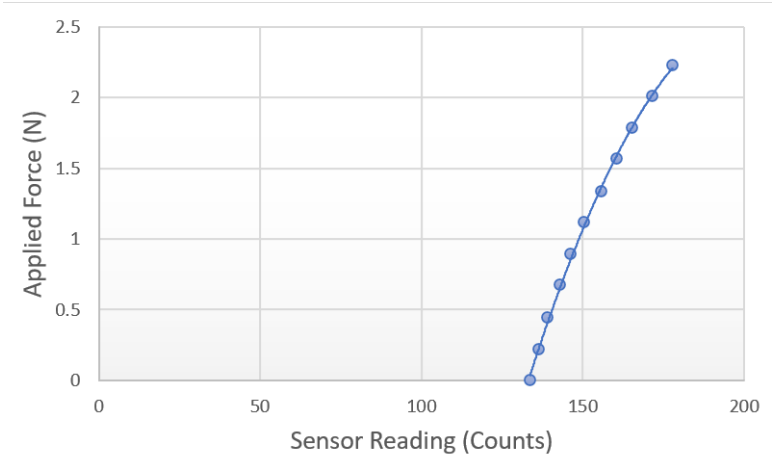



Figure 4.1: Calibration of a force sensor

where $c$ represents the sensor counts and $F$ is the corresponding applied force in Newtons.

### 4.1.2 Model Construction

#### Webots Simulation

To validate the controller designs in a semi-realistic environment without risk to the robot, they were tested on a simulated model of the robot in Webots. In order to accurately represent the behavior of the physical robot in the simulation, however, it was necessary to characterize important dynamics that were not captured by a simple physical model (as discussed in 4.1.2), as these play an important role in the robot's response to a command. For example, the servos change the dynamics of the robot because each have an internal proportional-integral-derivative (PID) controller that enables it to achieve the correct angle in response to a command. In Webots, the programmer can simply add a servo to the simulated robot and input the PID gains. However, since the gains used by the MG90s servos are not publicly available and cannot be changed easily by the user, it was necessary to characterize the servo controller's effects on the robot's open loop behavior to estimate the values of the gains. This was done by commanding a step in the horizontal position of the physical robot, observing its response using slow-motion video capture, and analyzing the video using Physlets Tracker.

After measuring the period of the oscillation and the overshoot, the following second order plant model

Figure 4.2: Real robot (left) and its Webots model (right) leaning due to a step in commanded horizontal position.

was developed for the robot. Its response is shown in Figure 4.3.

$$
\begin{aligned}
P(s) &= \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \\
\omega_n &= 21.75\frac{rad}{s} \\
\zeta &= 0.14
\end{aligned}
\tag{4.4}
$$

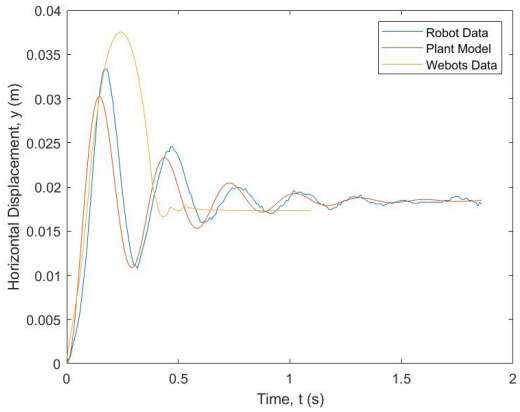



Figure 4.3: Step response of the physical robot to a horizontal position command of 2 cm, with the corresponding plant model developed from it. Using this response, the servo PID gains in the Webots simulations were adjusted to produce a similar response, as shown.

It is important to note that while the dynamics of the servos contribute to this second order behavior, much of the oscillatory behavior is due to the flexibility of the 3D-printed legs and small amount of backlash in the servos, neither of which can be replicated easily in the simulation. As a result, the simulated servo control gains were tuned to make the simulated robot's response match the real robot's rising time. The chosen simulated

servo gains were $K_p = 40$, $K_i = 0$, and $K_d = 0.05$.

The comparatively large overshoot of the Webots model was caused by the robot moving so quickly that it caused itself to tip slightly, and this tipping motion also caused the simulated robot to slide, leading to some steady state error at the end of the test. During testing, the real robot's feet were secured to prevent tipping and sliding, which limited the overshoot and steady state error. The lack of oscillations can be explained by the model's stiff limbs. While the simulated robot's open loop behavior does not exactly match that of the real robot, estimating the servo gains was an important step towards making the simulation more representative of the real robot, as this would provide more accurate information during controller and other software validation.

### The Cart-Table Model

While the plant model developed from the robot's open loop step response could be used to construct the Webots robot, this was not useful for describing the robot's response to an impact. In order to predict how the robot will respond to a disturbance and to facilitate controller design, a simple physics-based model of the robot was necessary. One common model used for quadrupedal [46–48] and especially for bipedal robots [44, 49] is the cart-table model, in which mass is concentrated in a cart that can move freely on the surface of a tilting massless table. Typically, the base of the table represents the dimensions of the biped's foot in contact with the ground. However, in the case where MicroDog is tilting about the $x$ axis, the robot has two "points" of contact (the port feet and starboard feet), assuming that the feet on the side against which the force is being applied will lift off the ground at the same time. It is also assumed that both contacts can be approximated as points since the feet are very small compared to the chassis. Thus, the table was modified to include two contact points on its base, as shown in Figure 4.4.

If the table (and thus the robot) does not tilt, then there is no angular acceleration of the table, and the moments about any point on the table should sum to 0. When a disturbance force, $F_d$, is applied to the side of the table (or chassis), the cart (center of mass) needs to shift in the opposite direction in order to counteract the tipping moment caused by $F_d$. This is accomplished by having the robot lean into the force.

$$\Sigma M_P = 0 = F_{zr}w - F_{zl}w - mgy + m\ddot{y}L - F_dL \tag{4.5}$$

From this, the expression for the difference in the magnitudes of the ground reaction forces applied to the

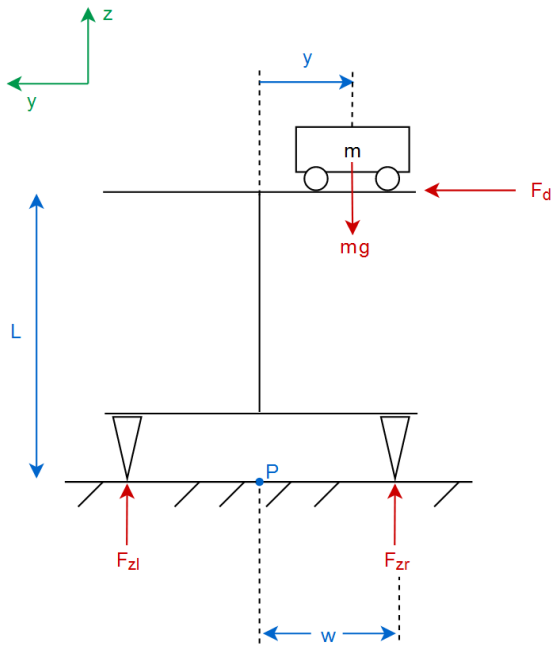



Figure 4.4: Cart-Table model, modified to account for the two rows of support legs.

starboard and port feet (noted as right and left in the diagram and equations) can be expressed as:

$$\Delta F_z = F_{zr} - F_{zl} = \frac{1}{\omega}(mgy - m\ddot{y}L + F_d L) \tag{4.6}$$

To develop a controller based on the model, it needs information about the robot's states; that is, its position, velocity, and acceleration. It is useful to formulate this using state space. It is assumed that the robot's mass is concentrated in the cart as a point mass, that the table has no mass, and that the cart remains at a constant height relative to the ground. Neglecting external disturbances such as $F_d$, the only input is the robot's jerk or change in lateral acceleration, $u = \dddot{y}$.

$$\dot{\vec{x}}_o = \frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{A_o} \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{B_o} u \tag{4.7}$$

$$\vec{y}_{out} = \begin{bmatrix} \Delta F_z \\ y \\ \ddot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{mg}{w} & 0 & -mL \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{C_o} \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{D_o} u \tag{4.8}$$

### 4.1.3 Disturbance Force Control Design

Since the difference in force between the port and starboard feet is a measure of the disturbance force, the controller was designed around controlling this difference, or $\Delta F_z$. To do this, however, $\Delta F_z$ needed to be included as a state by transforming the state vector, as seen in Equation 4.9.

$$\vec{x}_i = \begin{bmatrix} \Delta F_z \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{mg}{w} & 0 & -mL \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix}}_{\vec{x}_o} \tag{4.9}$$

This required the state space matrices to be transformed as well.

$$\dot{\vec{x}}_i = \underbrace{TA_oT^{-1}}_{A_i} \vec{x}_i + \underbrace{TB_o}_{B_i} u = \underbrace{\begin{bmatrix} 0 & \frac{mg}{w} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{A_i} \begin{bmatrix} \Delta F_z \\ \dot{y} \\ \ddot{y} \end{bmatrix} + \underbrace{\begin{bmatrix} -mL \\ 0 \\ 1 \end{bmatrix}}_{B_i} u \tag{4.10}$$

$$\vec{y}_{out} = \underbrace{C_oT^{-1}}_{C_i} \vec{x}_i + \underbrace{D_o}_{D_i} u = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ \frac{w}{mg} & 0 & \frac{wL}{g} \\ 0 & 0 & 1 \end{bmatrix}}_{C_i} \begin{bmatrix} \Delta F_z \\ \dot{y} \\ \ddot{y} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{D_i} u \tag{4.11}$$

To minimize steady state error, the integral of $\Delta F_z$ must also be included as a state. Thus, the open loop

state vector and model can be expressed as,

$$\vec{x} = \begin{bmatrix} \int \Delta F_z \\ \vec{x}_i \end{bmatrix} = \begin{bmatrix} \int \Delta F_z \\ \Delta F_z \\ \dot{y} \\ \ddot{y} \end{bmatrix} \tag{4.12}$$

This leads to the following open loop model in state space form:

$$\dot{\vec{x}} = \frac{d}{dt} \begin{bmatrix} \int \Delta F_z \\ \Delta F_z \\ \dot{y} \\ \ddot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{mg}{w} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} \int \Delta F_z \\ \Delta F_z \\ \dot{y} \\ \ddot{y} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ -mL \\ 0 \\ 1 \end{bmatrix}}_{B} u \tag{4.13}$$

With the following output equation describing the choice of vertical force, lateral CG position, and lateral acceleration as outputs:

$$\vec{y}_{out} = \begin{bmatrix} \Delta F_z \\ y \\ \ddot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{w}{mg} & 0 & \frac{wL}{g} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{C} \begin{bmatrix} \int \Delta F_z \\ \Delta F_z \\ \dot{y} \\ \ddot{y} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{D} u \tag{4.14}$$

To control the robot's $\Delta F_z$, a state feedback controller was chosen. In order to identify the optimal control gains $\vec{K}$, a Linear Quadratic Regulator was designed to minimize the objective function $J = \int_0^\infty (\vec{x}^T Q \vec{x} + u^T R u) dt$ using weights $Q$ and $R$, where $u = -\vec{K}\vec{x}$. For this study, more weight was placed on minimizing $\int \Delta F_z$ and $\ddot{y}$ to enable the controller to respond more quickly to steady-state error, as shown:

$$Q = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}, R = 1 \tag{4.15}$$

Using the MATLAB function *lqr*, these weights for $Q$ and $R$ produced the gains $\vec{K}$ = [10.0, 6.13, 177.2, 21.7].

Now considering the lateral step disturbance $F_d$ when the input $u = F_d$, the $B$ and $D$ matrices of the state space and output equations need to be adjusted as follows:

$$B_2 = \begin{bmatrix} -\frac{L}{w} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.16}$$

$$D_2 = \begin{bmatrix} -\frac{L}{w} \\ 0 \\ 0 \end{bmatrix} \tag{4.17}$$

Incorporating this new $B_2$ into the state space equation to include $F_d$ gives:

$$\begin{aligned} \dot{\vec{x}} &= A\vec{x} + Bu + B_2 F_d \\ &= A\vec{x} - B\vec{K}\vec{x} + B_2 F_d \\ &= \underbrace{(A - B\vec{K})}_{A_c}\vec{x} + \underbrace{B_2}_{B_c} F_d \end{aligned} \tag{4.18}$$

Thus, the lean controller's closed-loop dynamics can be described using the following state space model.

$$\dot{\vec{x}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ mLK_1 & mLK_2 & mLK_3 & mLK_4 \\ 0 & 0 & 0 & 0 \\ -K_1 & -K_2 & -K_3 & -K_4 \end{bmatrix}}_{A_c} \underbrace{\begin{bmatrix} \int \Delta F_z \\ \Delta F_z \\ \dot{y} \\ \ddot{y} \end{bmatrix}}_{\vec{x}} + \underbrace{\begin{bmatrix} -\frac{L}{w} \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{B_c} F_d \tag{4.19}$$

$$\vec{y}_{out} = \begin{bmatrix} \Delta F_z \\ y \\ \ddot{y} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{w}{mg} & 0 & \frac{wL}{g} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{C_c} \underbrace{\begin{bmatrix} \int \Delta F_z \\ \Delta F_z \\ \dot{y} \\ \ddot{y} \end{bmatrix}}_{\vec{x}} + \underbrace{\begin{bmatrix} -\frac{L}{w} \\ 0 \\ 0 \end{bmatrix}}_{D} F_d \tag{4.20}$$

When implemented in code, $\dddot{y}$, or the commanded CG position, is used as the control input, and servo dynamics are neglected. Because the LQR is a state feedback controller, information is needed about each of the states. However, while $\Delta F_z$ and the lateral acceleration $\ddot{y}$ can be measured using the force sensors

and IMU, respectively, there is no means to directly measure the position $y$, the velocity $\dot{y}$, or $\int \Delta F_z$ on the real robot. Thus, $\ddot{y}$ and $\Delta F_z$ were integrated to find the missing values, which means that commanded servo positions and actual servo positions were assumed to be the same.

### 4.1.4 Software Design

The control software was written in Python (see Appendix A for repositories) and implemented on the Raspberry Pi. As shown in Figure 4.5, however, the Raspberry Pi also communicated with the ATmega32U4 to gather force sensor data (since these sensors were tied to the analog pins of the ATmega32U4, and thus could not be read directly by the Raspberry Pi) and gathered acceleration data over $I^2C$ from the IMU before passing the servo position commands to the PCA9685 servo driver.

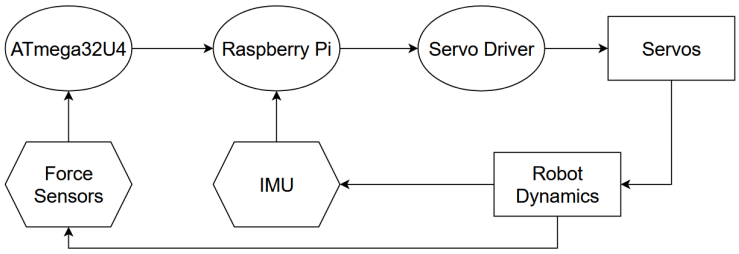



Figure 4.5: Architecture of Lean Control Software

### 4.1.5 Experiments

To test this controller, a small pulse force $F_d$ was applied to the side of the robot, first in the Webots simulation and then on the physical robot. Data about all the states that could be measured or calculated was collected. As noted previously, the lack of motor encoders means that the robot has no direct way of measuring its actual horizontal position. To collect this information in Webots, the GPS tool was used to track the robot's center of mass (CM). To find the actual position of the real robot, the motion was video-recorded and analyzed in Physlets Tracker, as was done to derive the servo dynamics (see Section 4.1.2).

## 4.2 Results and Discussion

Figure 4.6 illustrates the controller responses of the MATLAB simulated design, the Webots simulation, and the robot. The commanded position (found through integration) and actual position (measured using the GPS

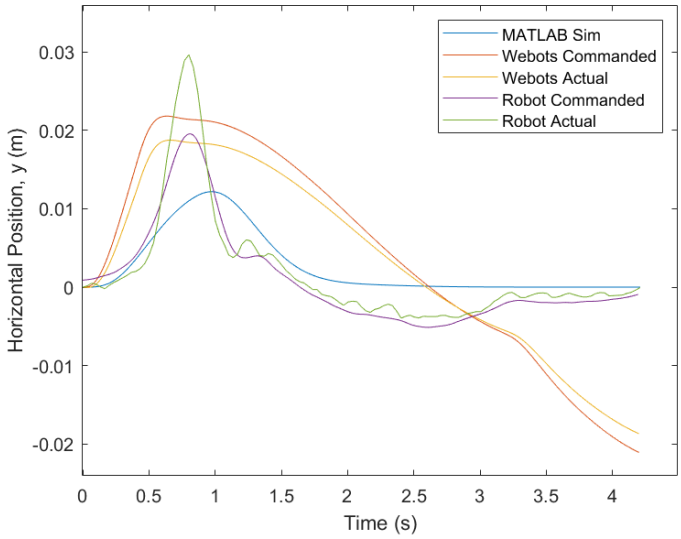



Figure 4.6: Comparison of controller responses

tool in Webots or video for the robot) of the Webots model and robot are also compared.

One issue in precisely comparing the models and the actual robot was that it was difficult to apply a consistent and known pulse force to the robot by hand. Thus, the magnitude of the disturbance and period of time that it was applied was estimated using data from the force sensors and the video, and these approximate values were then implemented in MATLAB and Webots. This leads to some inaccuracy in the comparison, but such validation is still useful for understanding general trends in behavior. The rise times of the MATLAB simulation and robot to reach the peak lean position are similar, and the larger actual position magnitude of the robot can be explained by the fact that unmodeled dynamics (such as flexibility in the legs and servo dynamics, which were not accounted for in the MATLAB design) caused the robot to experience overshoot. While there is some drift in the controller as the robot settles back to no lean after the force is removed, both the commanded and actual position do reach approximately 0.

More concerning, however, is the behavior of the Webots model. While the magnitudes of the commanded and actual position reflect the robot's behavior well, the model experiences faster rise time, slow settling time, and drift at the end. An explanation of these trends was not discovered during the course of this thesis. However, while this made it difficult to verify the design, the Webots model was still useful in checking for stability and approximating the required lean magnitude in the safety of the simulation environment.

# Chapter 5

# Disturbance Recovery: Scramble

While the lean controller enables MicroDog to counteract small disturbance forces, the robot can only lean so far. If the moment caused by a disturbance is larger than what the robot can match by shifting its center of mass, the disturbance will cause the robot to tip over. Thus, MicroDog needed a method to side-step, or scramble, away from the force to maintain its balance in response to large side loads, just as Spot does in response to a strong kick [43]. In a state machine combining the lean controller of Chapter 4 with the more aggressive scramble motion described in this chapter, MicroDog can react to both large and small lateral disturbances.

## 5.1 Methodology

### 5.1.1 State Machine Design

To implement this two-part disturbance recovery scheme, a simple state machine was developed to switch between the lean controller and scramble gait, as illustrated in Figure 5.1.

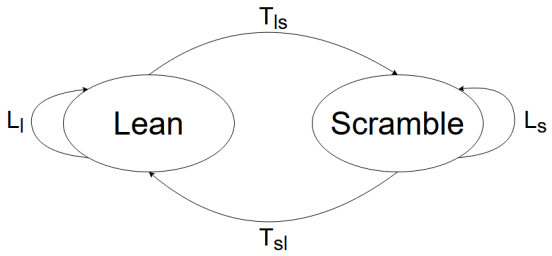



Figure 5.1: State machine diagram, corresponds with Table 5.1

As detailed in Table 5.1, two possible conditions were used to trigger the transition from lean to scramble.

First, the position of the Zero Moment Point (or ZMP, which will be discussed in 5.1.2) was evaluated. If this was beyond the support polygon (or "out of bounds") for several consecutive loops, then the Scramble state was triggered. The other condition that may cause the transition $T_{ls}$ to become true was if more than two feet are off the ground for several consecutive loops. This was needed because the ZMP calculation is only valid if at least three of the feet are on the ground. To return to Lean, the robot needs to complete at least one full gait cycle. This prevents the state machine from switching back and forth between the states every time the ZMP goes in and out of the support polygon.

Table 5.1: Disturbance Recovery State Machine, where outOfBounds is true if the y position of the ZMP is outside of the support polygon, offGround is true if at least 2 feet are lifted off the ground, and gaitComplete is true when one gait cycle has elapsed (i.e. each foot has taken a step).

| **Transition** | **Start State** | **Condition** | **End State** |
|---|---|---|---|
| $T_{ls}$ | Lean | outOfBounds or offGround | Scramble |
| $L_l$ | Lean | !outOfBounds and !offGround | Lean |
| $T_{sl}$ | Scramble | gaitComplete | Lean |
| $L_s$ | Scramble | !gaitComplete | Scramble |

### 5.1.2 Zero Moment Point

Some means of detecting MicroDog's stability was needed in order to determine when to switch out of the lean controller and to begin scrambling. While the Zero Moment Point (ZMP) was not useful for directly controlling MicroDog's lean (as discussed in Chapter 4), detecting when the ZMP went outside of the support polygon was as a useful method for checking if the robot was about to tip. Determining the location of the ZMP usually involves using a robot's motor encoders to determine the current position of the robot. Since MicroDog's servos do not have encoders, this method is not useful. What MicroDog does have, however, are force sensors, so if there was some means to relate the position of the ZMP to ground reaction forces (also known as contact forces) instead of the robot's inertia (CG), then the force sensors can be used to measure the ZMP.

Referring again to the cart-table model presented in Section 4.1.2, in which a point mass remains at a constant height above flat ground, and neglecting the disturbance force for now, it can be shown that the forces, and thus the moments about any point, from contact with the ground are equal and opposite to the forces and moments from the robot's inertia when the robot is dynamically balanced [50]. Modeling the port and starboard contact forces $F_{zl}$ and $F_{zr}$ as creating a coupling moment about $P_{ZMP}$, the horizontal position of the ZMP can be found as shown in Equation 5.1, where W is the whole width of the robot, or twice the value of $w$ in

Figure 4.4.

$$P_{ZMP,y} = \frac{W(F_{zr} - F_{zl})}{2(F_{zr} + F_{zl})} \tag{5.1}$$

This value can then be compared to half the width of the robot, as a value of zero indicates that the ZMP is centered between the feet. If the position of the ZMP exceeds half the width, then the ZMP is outside of the support polygon, leading to the need to scramble.

### 5.1.3 Scramble Gait

With the state machine having means to trigger scramble using the ZMP position, the scramble gait itself needed to be developed. It was observed that Spot uses a gait similar to a sideways walking gait, where approximately three feet are on the ground at a given moment in time and diagonally opposed feet swing in succession, starting with the front leg on the side experiencing the force. (That is, if a disturbance is applied to the quadruped's port side, the stepping pattern will be front port foot, rear starboard foot, front starboard foot, and rear port foot.) Thus, MicroDog's sideways walking gait discussed in Chapter 3 was used, as reiterated here.

$$y = \begin{cases} \cos(\frac{2}{3}\omega_{gait}t - \frac{2}{3}(n\phi_{base} + pi)) & \omega_{gait}t \leq \phi_{thres} \\ \cos(2\omega_{gait}t - 2n\phi_{base}) & \phi_{thres} < \omega_{gait}t < \phi_{thres} + \phi_{swing} \\ \cos(\frac{2}{3}\omega_{gait}t - \frac{2}{3}(n\phi_{base})) & \phi_{thres} + \phi_{swing} \leq \omega_{gait}t \end{cases} \tag{5.2}$$

$$z = \begin{cases} 0 & \omega_{gait}t \leq \phi_{thres} \\ \cos(2\omega_{gait}t - 2(n+3)\phi_{base}) & \phi_{thres} < \omega_{gait}t < \phi_{thres} + \phi_{swing} \\ 0 & \phi_{thres} + \phi_{swing} \leq \omega_{gait}t \end{cases} \tag{5.3}$$

In this study, only a disturbance on the port side of the robot was considered, leading to a scramble gait that starts with the front port leg. However, this design can be expanded in respond to forces from any direction.

With the state machine developed, disturbance recovery could be validated in Webots and on MicroDog.

## 5.2 Experiments, Results, and Discussion

For initial validation, a pulse force large enough to cause tipping was applied to the Webots model. The simulated robot successfully triggered the scramble gait, took a few steps sideways, and stopped to enter into

lean once more. Once it was determined that state machine worked in simulation, it was tested on the physical robot, with similar results.

Other than simply validating that the robot successfully maintains dynamic stability, however, there are several parameters in the scramble gait that need to be optimized to develop the most reliable gait: the "speed" of the gait ($\omega_{gait}$), the height of the step ($z_{amp}$), and the lateral amplitude of the step ($y_{amp}$). For this study, six different $\omega_{gait}$ ranging from $\pi$ to $6\pi$ were evaluated while $y_{amp} = 2$ cm and $z_{amp} = 1.5$ cm. Since it was difficult to control the applied pulse force on the real robot by hand, the experiments were first performed on the robot. Using the initial change in $\Delta F_z$ (since this term is influenced directly by $F_d$ through the $D$ matrix as shown in Equation 4.20), the magnitude of the force and the time it was applied was found. This information was then used to create a matching pulse force in Webots. After tracking its response, the behavior of the Webots model and MicroDog were compared.

Figure 5.2 illustrates how the horizontal position of the center of mass and the difference in force between the port and starboard legs when $\omega_{gait} = 1\pi$. The robot triggered scramble a bit later than the Webots model did due to the robot slipping more along the smooth floor than the Webots model did. Accounting for this difference, however, the motion matches fairly well. Both experience some oscillatory behavior during scramble due to the large amount of tilting that occurs during slow walking. The robot experienced more oscillation due to unmodeled dynamics, and the combination of these oscillations and the low surface friction prevented the real robot from traveling as far as the Webots model.

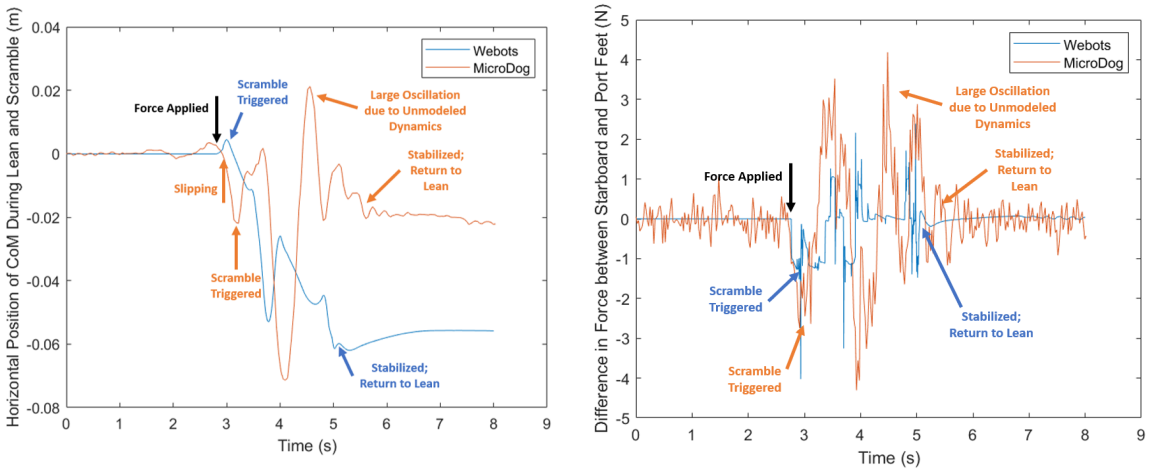



Figure 5.2: Scramble Behavior at $\omega_{gait} = 1\pi$

As $\omega_{gait}$ increases, however, the oscillatory behavior of both the Webots model and the physical robot is drastically diminished, as seen in Figures 5.3 and 5.4. The faster scrambling speeds reduced the tilting that

occurred at slower speeds because the feet moved more quickly than the chassis could tilt. This allowed both the Webots model and the real robot to traverse a greater distance and settle back into lean more quickly.

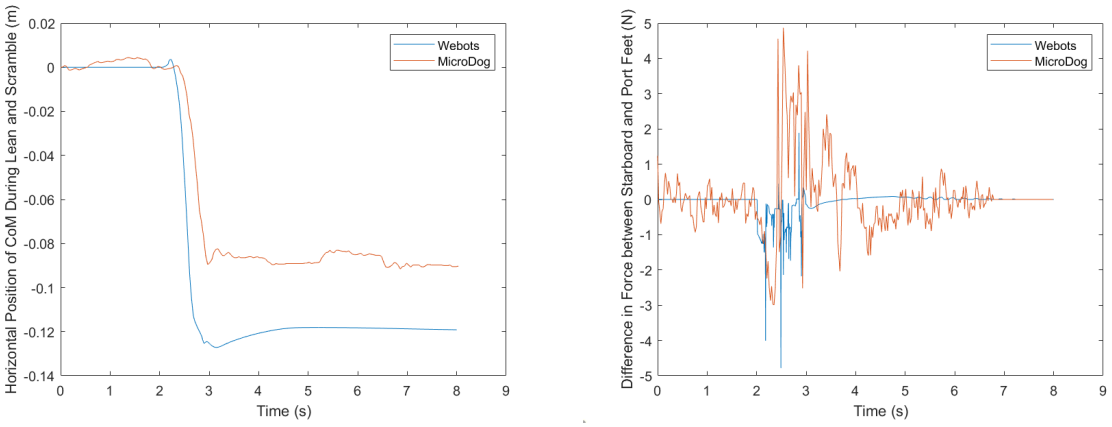



Figure 5.3: Scramble Behavior at $\omega_{gait} = 3\pi$

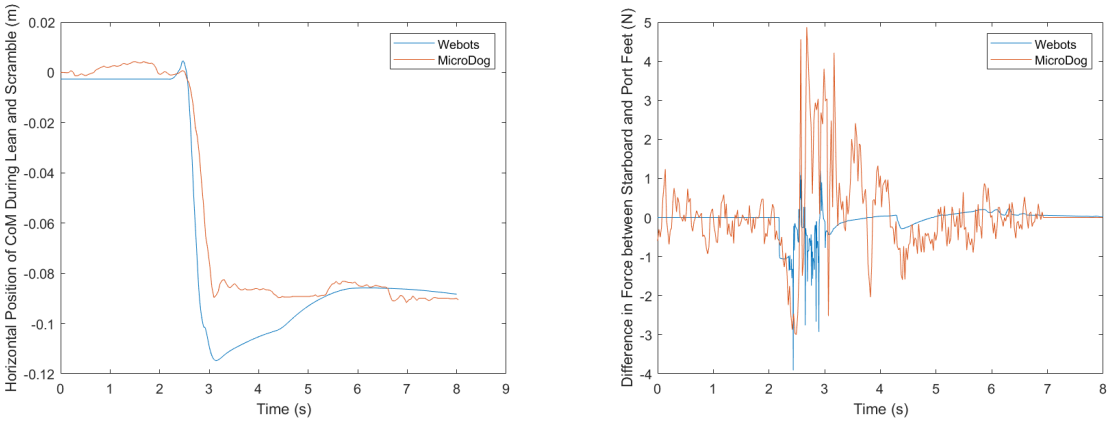



Figure 5.4: Scramble Behavior at $\omega_{gait} = 5\pi$

However, there is a limited range of useful scrambling speeds. What is not reflected in these graphs is the reliability of the gait. Increasing $\omega_{gait}$ too much sometimes causes the real robot to loose its footing and fall. While the Webots model did not initially fall, some trials showed that the model got trapped in the Scramble state because the inertia caused it to continue tipping in the direction of motion, which did not allow the ZMP to be within the support polygon or prevent multiple feet from lifting off the ground. Trials run at $\omega_{gait} >= 4\pi \frac{rad}{s}$ evidenced this behavior. From this very brief investigation, it appears that $\omega_{gait} = 3\pi \frac{rad}{s}$ when $y_{amp} = 2$ cm and $z_{amp} = 1.5$ cm is optimal for reliable scrambling. However, a more thorough investigation is needed.

# Chapter 6

# Conclusions and Future Work

Over the course of this thesis, a low-cost quadruped robot was developed to meet the need for inexpensive and accessible research tools. The result was MicroDog: a $160 platform consisting of 3D printed components as well as a custom PCB and onboard battery to allow for untethered operation. This expandable platform includes IR sensors for obstacle detection, an IMU for measuring orientation and accelerations, and, most importantly, the Hall effect force sensors for proprioceptive sensing.

To demonstrate capability on par with other low-cost platforms, walking and turning gaits were developed, as well as basic obstacle avoidance using the IR sensors. To demonstrate MicroDog's value as a research tool, disturbance recovery was implemented using a lean controller and scrambling.

This work only scratches the surface of what can be done with this platform. In relation to the capabilities discussed in this thesis, some areas for further work include improving the walking and turning gaits, developing a calibration scheme for the IR sensors that accounts for different lighting and surface conditions, testing more controllers for lean, and optimizing the scramble gait. Beyond the specific topics discussed in this thesis, other areas of research include obstacle identification, stair climbing, rough terrain navigation, and so much more. As demonstrated in this thesis, MicroDog is ready to contribute to the fields of low-cost quadruped robotics and robotics education.

# Appendix A

# Appendix

## A.1 MicroDog Bill of Materials and Thesis Expenditures

For a detailed breakdown of MicroDog's bill of materials and of total expenses over the course of this year, please see: https://bit.ly/3fyaTXU.

## A.2 Inverse Kinematics and Walking Libraries

To take a look at the Python and Arduino libraries used for inverse kinematics, different walking gaits, and lean, check out the microdog_v4_libraries repository at:
https://github.com/G-Conard/microdog_v4_libraries.

## A.3 Hardware and Software Developed for Thesis

To see the code developed for obstacle avoidance, disturbance recovery, and other tests, as well as the STL files of the 3D printed components and PCB Gerber files, see the GCThesis_MicroDog_Tests repository at:
https://github.com/G-Conard/GCThesis_MicroDog_Tests.

# Bibliography


[1] J. Xu, L. Lang, H. Ma, and Q. Wei, "Contact force based compliance control for a trotting quadruped robot," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, May 2015, pp. 5144–5149, iSSN: 1948-9447. 3, 4

[2] H. Zhuang, H. Gao, Z. Deng, L. Ding, and Z. Liu, "A review of heavy-duty legged robots," *Science China Technological Sciences*, vol. 57, no. 2, pp. 298–314, Feb. 2014. [Online]. Available: https://doi.org/10.1007/s11431-013-5443-7 3

[3] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "ANYmal - a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 38–44, iSSN: 2153-0866. 3

[4] Y. Li, B. Li, J. Ruan, and X. Rong, "Research of mammal bionic quadruped robots: A review," in *2011 IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM)*, Sep. 2011, pp. 166–171, iSSN: 2158-219X. 3

[5] "Spot® | Boston Dynamics." [Online]. Available: https://www.bostondynamics.com/spot 3

[6] Q. Nguyen, M. J. Powell, B. Katz, J. D. Carlo, and S. Kim, "Optimized Jumping on the MIT Cheetah 3 Robot," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 7448–7454, iSSN: 2577-087X. 3, 8, 9

[7] C. Hubicki, A. Abate, P. Clary, S. Rezazadeh, M. Jones, A. Peekema, J. Van Why, R. Domres, A. Wu, W. Martin, H. Geyer, and J. Hurst, "Walking and Running with Passive Compliance: Lessons from Engineering: A Live Demonstration of the ATRIAS Biped," *IEEE Robotics Automation Magazine*, vol. 25, no. 3, pp. 23–39, Sep. 2018, conference Name: IEEE Robotics Automation Magazine. 3, 9

[8] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 2245–2252, iSSN: 2153-0866. 3, 8, 9

[9] G. Kenneally, A. De, and D. E. Koditschek, "Design Principles for a Family of Direct-Drive Legged Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 900–907, Jul. 2016, conference Name: IEEE Robotics and Automation Letters. 3, 8

[10] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback Control of a Cassie Bipedal Robot: Walking, Standing, and Riding a Segway," in *2019 American Control Conference (ACC)*, Jul. 2019, pp. 4559–4566, iSSN: 2378-5861. 3, 9

[11] X. Li, W. Wang, and J. Yi, "Foot contact force of walk gait for a quadruped robot," in *2016 IEEE International Conference on Mechatronics and Automation*, Aug. 2016, pp. 659–664, iSSN: 2152-744X. 3

[12] K. G. Karwa, S. Mondal, A. Kumar, and A. Thakur, "An open source low-cost alligator-inspired robotic research platform," in *2016 Sixth International Symposium on Embedded Computing and System Design (ISED)*, Dec. 2016, pp. 234–238, iSSN: 2473-9413. 4, 5

[13] "DIY quadruped robot (testing force feedback)," Jul. 2020. [Online]. Available: https://www.youtube.com/watch?v=KRen4yexD4Y&feature=youtu.be 4, 7

[14] M. Ayuso Parrilla, "DIY hobby servos quadruped robot." [Online]. Available: https://hackaday.io/project/171456-diy-hobby-servos-quadruped-robot 4, 8

[15] L.-C. Liao, K.-Y. Huang, and B.-C. Tseng, "Design and implementation of a quadruped robot insect," in *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, Aug. 2015, pp. 269–273, iSSN: 2152-744X. 4, 5

[16] A. L. Bittle, A. Martinez, J. Graff, and K. Maynard, "Low Cost Quadruped: MUTT," p. 97. 5, 27

[17] F. Garcia-Cardenas, N. Soberon, O. E. Ramos, and R. Canahuire, "Charlotte: Low-cost Open-source Semi-Autonomous Quadruped Robot," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. Ponta Delgada, Portugal: IEEE, Apr. 2020, pp. 281–286. [Online]. Available: https://ieeexplore.ieee.org/document/9096210/ 5

[18] 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, Jul. 2013, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/0278364913489205 5, 6

[19] "Quadruped Benchmark — Stanford Student Robotics." [Online]. Available: https://stanfordstudentrobotics.org/quadruped-benchmark 6, 7

[20] "Adeept DarkPaw Quadruped Spider Robot Kit for Raspberry Pi." [Online]. Available: https://www.robotshop.com/en/adeept-darkpaw-quadruped-spider-robot-kit-raspberry-pi.html 6

[21] "Lynxmotion Phoenix 3DOF Hexapod - Black (No Servos / Electronics)." [Online]. Available: https://www.robotshop.com/en/lynxmotion-phoenix-3dof-hexapod---black-no-servos---electronics.html 6

[22] "Robots — Stanford Student Robotics." [Online]. Available: https://stanfordstudentrobotics.org/robots 6

[23] N. Kau, A. Schultz, N. Ferrante, and P. Slade, "Stanford Doggo: An Open-Source, Quasi-Direct-Drive Quadruped," *arXiv:1905.04254 [cs]*, May 2019, arXiv: 1905.04254. [Online]. Available: http://arxiv.org/abs/1905.04254 6

[24] "Stanford Pupper 2020 documentation." [Online]. Available: https://pupper.readthedocs.io/en/latest/# 6, 7

[25] Nathan, "Stanford Pupper - ICRA 2021 Submission Video," Nov. 2020. [Online]. Available: https://www.youtube.com/watch?v=T48Rl8wFT34 6

[26] "Petoi Bittle: A Palm-sized Robot Dog for STEM and Fun." [Online]. Available: https://www.kickstarter.com/projects/petoi/bittle 7

[27] L. Rongzhong, "Bittle: A Palm-sized Robot Dog for STEM and Fun." [Online]. Available: https://www.indiegogo.com/projects/2633743 7

[28] ——, "Nybble - World's Cutest Open Source Robotic Kitten." [Online]. Available: https://www.indiegogo.com/projects/2421126 7

[29] "Home - SpotMicroAI." [Online]. Available: https://spotmicroai.readthedocs.io/en/latest/ 7, 27

[30] "Raspberry Pi Spot Micro Quadruped Project." [Online]. Available: https://www.youtube.com/watch?v=S-uzWG9Z-5E&t=4s 7

[31] mike4192, "mike4192/spotMicro," Sep. 2020, original-date: 2020-03-31T00:54:50Z. [Online]. Available: https://github.com/mike4192/spotMicro 7, 8

[32] Martin Triendl, "DIY quadruped robot (cheap with suspension)," Jan. 2021. [Online]. Available: https://www.youtube.com/watch?v=Kz3gDpoZsoE 7, 8, 12

[33] T. Tsujita, T. Kitahara, R. Tahara, S. Abiko, and A. Konno, "Drop test for evaluating effect of cushioning material and servo gain on parachute landing impact using a small one-legged robot," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2017, pp. 2474–2479. 8

[34] "DYNAMIXEL MX-28AT." [Online]. Available: https://www.robotis.us/dynamixel-mx-28at/ 9

[35] "Cassie: Dynamic Planning on Stairs," Feb. 2019. [Online]. Available: https://www.youtube.com/watch?v=qV-92Bq96Co&feature=youtu.be 9

[36] "TowerPro MG92B Metal Gear Servo." [Online]. Available: https://graysonhobby.com/towerpro-mg92b-metal-gear-servo.html 12

[37] "Pack of 4 DS3218 Digital RC Servo Motor 20KG High Torque Full Metal Gear Waterproof for Baja Cars Robot DIY 270°: Toys & Games." [Online]. Available: https://www.amazon.com/DS3218-Digital-Servo-Torque-Waterproof/dp/B07WYQ9P3F 13

[38] "SMAKN DC 4.5-60V to 1.25-30V Lm2596hv Power Supply Buck Voltage Switching Regulator Module Step Down." [Online]. Available: https://www.amazon.com/4-5-60V-1-25-30V-Lm2596hv-Switching-Regulator/dp/B00W8UTRJA/ref=sr_1_8?dchild=1&keywords=5V+switching+1A+pcb&qid=1621543862&sr=8-8 15

[39] "Arduino Nano." [Online]. Available: https://store.arduino.cc/usa/arduino-nano 15

[40] "SunFounder PCA9685 16 Channel 12 Bit PWM Servo Driver for Arduino and Raspberry Pi." [Online]. Available: https://www.amazon.com/SunFounder-PCA9685-Channel-Arduino-Raspberry/dp/B014KTSMLA/ref=sr_1_2?dchild=1&keywords=16+channel+servo+driver&qid=1621543813&sr=8-2 15

[41] "Battery Life Calculator | DigiKey Electronics." [Online]. Available: https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-battery-life 18

[42] D. Pongas, M. Mistry, and S. Schaal, "A Robust Quadruped Walking Gait for Traversing Rough Terrain," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 1474–1479, iSSN: 1050-4729. 28

[43] "Spot The Robot Dog Can Run, Go Up Stairs And Even Endure Kicks!" Feb. 2015. [Online]. Available: https://www.youtube.com/watch?v=BY-FTldlGA8 35, 46

[44] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, Sep. 2003, pp. 1620–1626 vol.2, iSSN: 1050-4729. 35, 39

[45] T. Akbas, S. E. Eskimez, S. Ozel, O. K. Adak, K. C. Fidan, and K. Erbatur, "Zero Moment Point based pace reference generation for quadruped robots via preview control," in *2012 12th IEEE International Workshop on Advanced Motion Control (AMC)*, Mar. 2012, pp. 1–7, iSSN: 1943-6580. 35

[46] K. Byl, A. Shkolnik, S. Prentice, N. Roy, and R. Tedrake, "Reliable Dynamic Motions for a Stiff Quadruped," in *Experimental Robotics*, ser. Springer Tracts in Advanced Robotics, O. Khatib, V. Kumar, and G. J. Pappas, Eds. Berlin, Heidelberg: Springer, 2009, pp. 319–328. 35, 36, 39

[47] A. W. Winkler, F. Farshidian, M. Neunert, D. Pardo, and J. Buchli, "Online walking motion and foothold optimization for quadruped locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5308–5313. 39

[48] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Fast, robust quadruped locomotion over challenging terrain," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 2665–2670, iSSN: 1050-4729. 39

[49] Shengjun Peng, Haitao Shui, Guang Li, and Hongxu Ma, "Walking gait planning of humanoid soccer robot based on the desired ZMP trajectories," in *2010 The 2nd International Conference on Industrial Mechatronics and Automation*. Wuhan, China: IEEE, May 2010, pp. 127–131. [Online]. Available: http://ieeexplore.ieee.org/document/5538351/ 39

[50] "Zero moment point," Apr. 2020, page Version ID: 950168859. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Zero_moment_point&oldid=950168859 47