Wheeled Bipedal Mobile Robot

A Major Qualifying Project Report
Submitted to the Faculty of
Worcester Polytechnic Institute
In Partial fulfillment of the requirements for the
Degre of Bachelor of Science

Submitted by:
Brian Boxell

Date: April 27th, 2023

Submitted to:
Professor Mohammad Mahdi Agheli Hajiabadi
Professor William Michalson

Abstract

Mobile robotics is a growing field that focuses on developing platforms that can move throughout an environment to accomplish several tasks. Traditionally, mobile robotics consists of a payload attached to a wheeled base, like what can be seen on anything from iRobot's Roomba robotic vacuum to NASA's Mars Rovers. However, wheeled robots are largely restricted to smooth and flat surfaces. In many cases, it is useful for robots to be able to climb obstacles like stairs or jump up to higher elevations to improve their mobility. Recently, there has been a push to recreate the adaptability of humans and animals' legs to make a robot that can truly go anywhere we can. Robots like Boston Dynamics' Spot and Atlas, Agility Robotics' Digit, and the hundreds of quadruped robots across the world push the boundaries of where robots can go. The limitless applications of legged robotics make them extremely attractive for working alongside humans since, in theory, there is no obstacle that we could traverse that they cannot. However, legged robots are incredibly inefficient in comparison to wheeled robots, and in general, complex obstacles that require legs to traverse only make up a small fraction of the total terrain that a mobile robot can expect to encounter. For example, if a robot is attempting to navigate a warehouse, it will likely spend most of its time driving around a smooth floor and only occasionally need to climb a staircase or traverse rough terrain. A legged robot would accomplish this navigation, but would not be the most efficient design. A wheeled robot would be highly efficient for traditional floor navigation, but unable to traverse complicated obstacles. This project explores the effectiveness of combining wheels and legs to create a highly robust mobile platform that can accomplish many types of locomotion as efficiently as possible.

Acknowledgements

I would like to thank Professor Agheli and Professor Michalson for their support and guidance throughout this project. Additionally, for facilitating me in being able to pursue this difficult and rewarding research project.

I would like to thank Analog Devices for donating the IMU that serves as the primary sensor input.

I would like to thank William Stanley and Seems Reasonable Robotics for their assistance in creating a custom release of the VESC firmware.

I would like to thank Evelyn Maude for assisting in debugging SPI communication with the IMU.

And finally, I would like to thank the WPI Combat Robotics Club for inspiring this creative project and pushing me to fulfill its requirements to the best of my ability.

Contents

List of Figures

1 Introduction

Mobile robotics is a subset of robotics that is continually increasing in popularity as robots become more complex and attempt to become more integrated into our lives. Mobile robots have the ability to move around and perform tasks in different places, which increases the versatility and usefulness of a single robot. Mobile robots have a variety of use cases; anything from indoor factory work to outdoor search and rescue, and each terrain likely comes with a different form of type of robotic locomotion that would be ideal to handle it.

The majority of mobile robots work in indoor environments, which primarily means that they are traveling along smooth surfaces like concrete, tile, or hard wood. For smooth surfaces, the natural choice for locomotion is wheels for a number of reasons. First and foremost, wheels are mechanically simple and are power efficient. Engineers have studied for centuries on how to implement robust and reliable powertrains for wheeled locomotion. Wheels are also easy to model kinematically, which simplifies the effort necessary to develop trajectory generation and motion planning algorithms. Additionally, wheels provide a stable base on which to place other components like manipulation elements and payloads, since three or more static points of contact on the ground provides mechanical stability. And finally, the combination of power efficiency and stability lends itself well to traveling at high speeds for long periods of time. However, most wheeled robotic platforms, unless specifically designed to do so, are unable to handle abstract terrain like rocks, rubble, and especially stairs.

The current popular solution to the deficiencies of wheels is legged robotics, a division of mobile robots that aims to replicate the versatility of animals by developing complex robotic legs. Legged robots are kinematically similar to humans and animals, which means that in theory, legged robots are the ideal choice to exist in a world that has been developed by humans. However, legged robots come with several important weaknesses. Firstly, they are extremely complex from a mechanical perspective. It is difficult to design leg linkages that are robust, accurate, and repeatable over extended periods of use. Further, it is excessively difficult to develop control algorithms to reliably maintain balance, especially while traversing the complex terrain that legged robots promise to overcome. The result of this is software that requires a lot of overhead to run, which makes robots intensely computationally expensive, which taxes the battery and power overhead of the whole robot. And finally, practical legged robots are historically very slow. Industrial legged robot technology is simply just now reaching a point where they can achieve the same walking pace as a human, and are far from reaching the speed and efficiency of wheels.

When examining the use cases for mobile robots, most applications require the robot to travel on a smooth surface for a strong majority of the run time. Very few applications require robots to continually do obstacles like stairs, and even when they do have to, the frequency with which they do can often times be optimized down using clever path planning algorithms. Therefore, ideal platform for most indoor mobile robotic applications would be a combination of

the efficiency of wheels and the versatility of legs. This platform would be able to provide the mechanical simplicity and efficiency of wheels, while also providing the ability to traverse stairs without requiring computationally expensive control algorithms.

This project attempts to achieve this ideal mobile platform by combining wheels and legs into a platform that is simple from both a mechanical and controls perspective. This robot will pursue attaching wheels onto a one degree of freedom leg to create a bipedal frame that navigates like a wheeled robot and balances like an inverted pendulum. This 1-DOF leg should provide the ability to dynamically balance in both roll and pitch, as well as the theoretical ability to jump and allow the robot to climb stairs. This combination of wheels and legs should provide a versatile dynamic platform that accomplishes all navigational tasks of an industrial mobile robot. This unique mobile robot shall be called “Scout”, a nod to one of several possible applications for it.

2 Background

While a combination of wheels and legs on a mobile robot is unique and still rare in the modern climate of robotics, this project is not the first to attempt this combination. There are a few platforms that have explored this combination in the past and served as inspiration for some of the design decisions made throughout the course of this project.

2.1 Boston Dynamics "Handle"

Boston Dynamic's groundbreaking robot handle served a number of purposes throughout its lifetime. Originally beginning life as a an offshoot research platform from their humanoid robot Atlas, Handle sought to explore the kinematics of combining wheels and a multi-DOF legged robot. Eventually, Boston Dynamics attempted to pursue a practical application with this robot of manipulating packages in a warehouse environment, demonstrated in Figure 1, before eventually sunsetting the robot to make way for its successor, Stretch. This history made Handle a large inspiration to this project, as prior to the start of this project, Brian Boxell, the engineer behind Scout, worked on the Strech project at Boston Dynamics. [2]

Figure 1: Boston Dynamics Handle as it attempts to pick up an example payload.

2.2 ETH Zurich "ANYmal"

Researchers at ETH Zurich in Switzerland developed a quadruped robot called ANYmal, a research platform to explore the kinematics of legged robots in an academic environment. While ANYmal is primarily a standard quadruped with rubber knobs as feet, it also has a configuration

where the feet can be replaced with wheels. The ANYmal team has published a series of videos that demonstrate its unique abilities to combine wheels and legs to travel quickly over complex terrain like hills, while moving smoothly similar to how a human on roller skates would act. This robot is also capable of a few novel behaviors like shifting from standing like a quadruped to standing on its hind legs in bipedal mode, even with wheels as end-effectors on the legs, as seen in Figure 2. [3]

Figure 2: ANYmal as it stands in bipedal mode, assisted by wheels as feet

2.3 Ascento Robotics "Ascento"

"Ascento" by Ascento Robotics is the one robot of this bunch that is a commercial platform. Ascento combines a four bar linkage with large wheels to create a versatile mobile platform that can transport the variety of sensor packages that the robot can be equipped with to perform various tasks. Ascento is the primary inspiration for this project, and many of the mechanical design decisions made throughout this project were influenced by those made by the Ascento team. [1] The Ascento robot can be seen standing up on two wheels in Figure 3.

Figure 3: Ascento Robot, Version 1

3 Design Decisions

This section will discuss the preliminary design decisions that were made to shape the robot at a high level. This will include discussion over the high-level linkage design and its interaction with the wheels, the electronics hardware structure, and the software architecture behind the various functions of the robot.

3.1 Why a Balancing Robot?

As shown by previous robots in industry that combine wheels and legs, there are many ways to accomplish this unity. Robots like ANYmal integrate wheels onto four legs to improve stability, while Ascento accomplishes the combination by balancing on two legs like an inverted pendulum. To understand which configuration makes the most sense for this project, let us define a few major design criteria for this project:

- Mechanically Simple
- Kinematically Simple
- Able to transport payloads
- Able to achieve speeds higher than a legged robot
- Minimal subsystems as to lower the scope of this project
- Cost effectiveness

Firstly, let us examine how a legged quadruped robot aligns with this design criteria. The mechanical complexity of each leg is going to be slightly more complex because if this robot is going to be able to do stairs, then each leg needs to be able to articulate laterally in the forward direction to be able to take a step up a stair. This means that a leg first must be able to move up and down, but also forward and backwards, thereby requiring at least two degrees of freedom. Two degrees of freedom is still kinematically trivial for an individual leg, but motions that combine all four of these legs would be very complex. In theory, a wheeled quadruped robot platform would be very stable, especially statically. This makes it a good choice for supporting heavy payloads. However, this design would require the manufacturing of four legs, which drives up the cost of this project, and increases the minimum amount of work required for completion.

Next, let us examine how a bipedal balancing robot compares to a quadruped in terms of this design criteria. The mechanical design of the legs is much simpler, because the robot would be theoretically able to accomplish behaviors involving hopping that would be more complex for a quadruped. A balancing robot could change its pitch and hop to accomplish stairs, which alleviates the need for a second degree of freedom. This keeps the robot more kinematically simple. Additionally, this platform requires only two legs, which minimizes the work required for hardware bring-up and reduces the scope of the project. However, this platform will not be as

stable as it will be constantly balancing due to being in unstable equilibrium. This means that this design would not be as good for transporting heavy payloads.

After examining the design requirements for this project, it was clear that a balancing robot would be able to satisfy all the requirements of this robot and would be easier to implement within the one-year scope of the project.

3.2 Size and Weight

One of the key design requirements of this project was to maintain a low but flexible budget since this robot was entirely self-funded without support from WPI. Because of this, an estimated size of 15 inches tall and weight of 5lbs was chosen after approximating the costs of electronics and materials that would be necessary to facilitate a robot of this size. This form factor, though small, would be significant enough to convey the results of the kinematic exploration intentions of the project.

3.3 Linkage Design

One of the focal points of this robot was the linkage design that should facilitate the legged robotic behaviors that the robot should accomplish. The high-level tasks that the linkage needed to accomplish were:

- Individually articulate the length of each leg.
- Raise and lower the robot's body.
- Be kinematically simple as to reduce the complexity required by the control algorithm.

Conceptually, the kinematically simplest solution to the leg linkage would be a linkage that moved the center of the mass of the robot perfectly linearly straight up and down. This reduces the disturbance that would be imparted on the balance controller and would help maintain stability throughout the body's vertical range of travel. However, perfect linearity is not a strict requirement because the balance controller is able to compensate for some lateral shifts in the COM so long as they are not too drastic. An implementation of a linear slide would work for this project, but the power train requirements and mechanical complexity was not favorable since it is easier to design a linkage with revolute joints as opposed to linear slides. A linear linkage like the Peaucellier mechanism was considered but rejected do to the complexity in having so many joints.

The simple choice for this problem was a four-bar linkage that was designed to approximate linearity of the end effector through its range of motion. Though a four bar can never be perfectly linear, there exist link lengths and pivot points that can achieve desirable output paths. Thus, a four-bar linkage was the decision for the linkage of this robot.

3.4 Electronics Architecture

One of the objectives of this project was to minimize cost, and electronics made up a large portion of the total cost of the project. Because of this, budget electronics were chosen where ever possible while attempting to not sacrifice the performance of the robot.

3.4.1 Computational Processing

The primary computational process for this robot is the balance control algorithm. One of the hallmarks of a good control algorithm is executing the loop as quickly as possible to decrease response latency. This means that the optimal choice for the primary microcontroller would be something without too much system overhead to dedicate all the resources towards running the control algorithm. Thus, the optimal choice would be either a single board computer (SBC) running a real time operating system (RTOS) or a microcontroller that natively runs an RTOS. Because of this, this project used an ESP32 as its primary microcontroller, which natively runs a release of FreeRTOS. This microcontroller would handle all inputs from sensor data, including input from the human operator, and would communicate with the brushless motor controllers.

3.4.2 Brushless Motor Control

This project uses brushless motors to actuate the wheels because brushless motors are lightweight and power dense, and also provide a form factor that is useful for packaging. The primary market for brushless motors of the size and power that was applicable for this project is drones, which means that most brushless motors and motor controllers are designed to operate at high RPMs and are not designed around low RPM fine control, which was integral for this project.

The most common solution for brushless motor controllers in robots of this size is a motor controller from ODrive Robotics, which produces motor controllers specifically designed to tackle the fine control issues of brushless motors and also come with a convenient API for sending and receiving data to and from the motor controller from the microcontroller. However, the current options that ODrive produces are all too large of a footprint to fit in the size of robot that was decided upon. Additionally, the cost of $149 for a single ODrive S1 was too expensive for this project's budget. The ODrives that were examined can be found here: https://odriverobotics.com/shop/odrive-s1

Another alternative was a VESC-based brushless motor controller. VESC is an open-source brushless motor control firmware, on which several motor controllers are built. The primary market for VESC-based speed controls is electric skateboards, which although they do not come

with the convenient API of the ODrive, are able to produce the low RPM fine control that this project would require. Additionally, VESC being open source means that this project was able to dig into the source code and modify the firmware to tailor it to this hardware platform wherever necessary. After looking into hardware options, the VESC A50S from TeamTriforce UK were selected. VESC supports multiple types of input, including PWM and DAC, which would make communicating with the ESP32 very easy. More information on these motor controllers can be found here: https://teamtriforceuk.com/a50s-v2/

3.4.3 SBC for Future Peripheral Integrations

Given that this platform was a mobile robot, the intention is to be able to attach useful peripheral devices like a camera or microphone to the robot. Though outside the scope of this project, the intention is to eventually have a standalone single board computer that handles the complex input from these peripherals and can translate them to useful behaviors that would be sent to the primary controls microcontroller. For example, a LiDAR could be attached to the top of the robot and feed data to a Raspberry Pi running ROS on Ubuntu that uses the data for SLAM. The Raspberry Pi could then generate a list of simpler commands for the robot to carry out for navigation and send them to the main ESP32 that handles the controls for how to accomplish those tasks.

3.4.4 Battery and Voltage Regulation

Being a mobile robot, the ability to move under its own power without the necessity of a tether to a power supply was paramount. Because of this, the robot needed its own battery power. The optimal choice for a battery would be lightweight and able to run for more than 30 minutes, a number chosen based on the predicted length of a demonstration of the robot. 16.8V (4S Lipo) was a natural decision for the operating voltage of the robot because it is a common voltage used in drones, which is what the motors and motor controllers of this size are typically designed for. The exact power draw requirements of the robot was difficult to estimate because it is heavily dependent how the robot is being driven. For example, the robot would be able to run for much longer if it was just idly balancing instead of driving around. Fortunately, this project had access to a wide range of LiPo batteries available for testing that were sized for 3lb combat robots, which enabled a guess and check philosophy to figure out what capacity battery was appropriate. The final decision was the GNB 4s 850mah LiPo found here: https://www.amazon.com/GAONENG-850mAh-15-2V-Battery-Range/dp/B097SJTNL4

However, not all components run on 16.8V and therefore, some voltage regulation was required. The ESP32 operates on 5V in, and the servos that would be chosen for the leg linkage operate on 6V. To facilitate this voltage regulation, Battery Eliminating Circuits (BEC) were used to step down the voltage from 16.8V. For the 5V processing line, a 2A BEC was chosen because of the

low current draw of the ESP32. For the servos, a 10A BEC was chosen because the servos chosen can draw upwards of 3A each.

4 Implementation

4.1 Hardware Implementation

Due to the nature of this robot being a balancing robot, designing and manufacturing a robust and durable robot was paramount to the success of the project. The robot had to be able to survive any failure mode without causing significant disruption to the development process.

4.1.1 Leg Linkage

The leg linkage decided upon was a four-bar linkage designed so that the path of the wheel throughout the range of motion would be as close to linear as possible. When designing the linkage, it was discovered that shortening the top link caused the driven link to approach a toggle point towards the limit of the travel. When the link approaches the toggle point, the horizontal displacement of the wheel is minimal. Using this concept, a close to linear four bar was produced according to the dimensions shown in Figure 4.

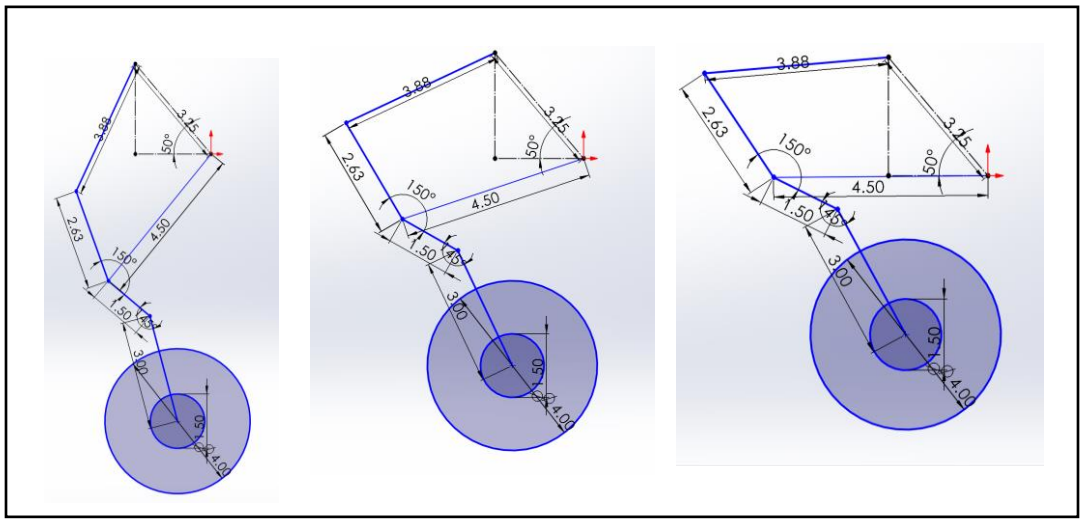

Figure 4: 4 Bark Linkage Diagram at Various Heights

The leg linkage is one of the most exposed parts of the robot, which makes it especially vulnerable to damage when the robot falls over. To accommodate this, materials were chosen that would far exceed the minimum requirements of a functional system. The legs are entirely machined out of aluminum 6061, chosen because of its availability and machinability. The joints of the legs are constructed with ball bearings on alloy steel shoulder bolts, both sourced from McMaster-Carr. Shoulder bolts and ball bearings were chosen because they can create robust,

low slop joints, which was critical for the leg linkage. The cross section of the linkage joints can be seen in Figure 5. The final physical implementation of the leg linkage can be seen in Figure 6.

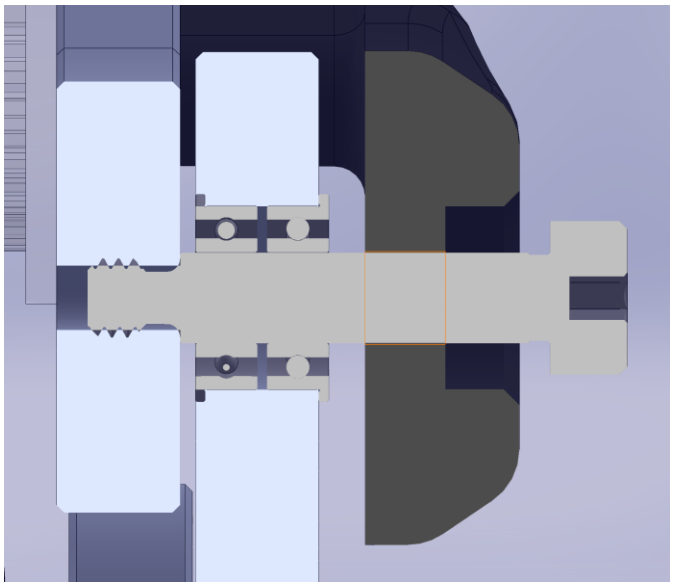

Figure 5: Linkage Joint Cross-Section

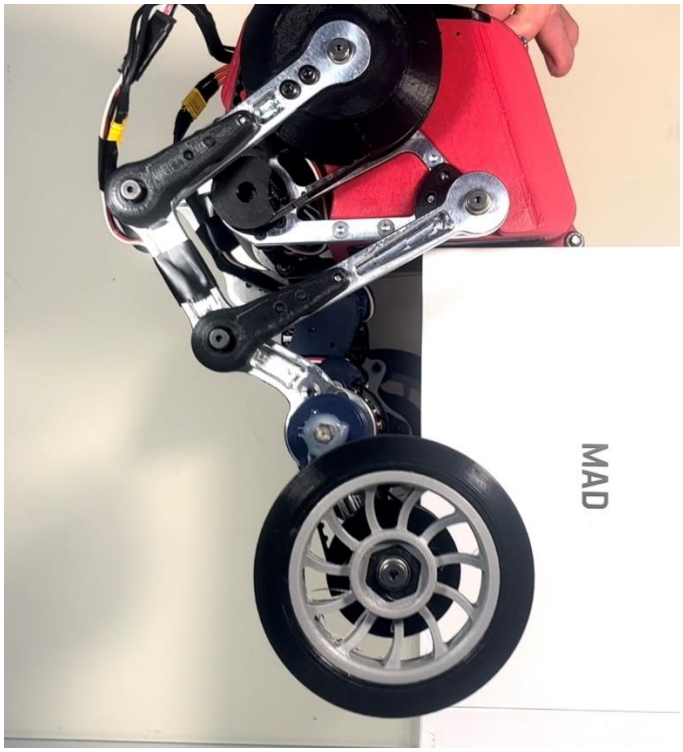

Figure 6: Completed Leg Linkage Profile View

4.1.2 Leg Linkage Power Transmission

As discussed in the design decision section, brushless motors were intended to power both the wheels on the legs and leg joints themselves. This decision was made such that the robot would have the power necessary to perform more complex behaviors like jumping. However, the complexities involved in implementing positional control of a brushless motor due to low rpm commutation issues made this design decision difficult to implement in real life and therefore out of the scope of the project. Instead, servos were chosen to control the legs because servos handle their own positional control, which made it easier to focus on the more interesting controls aspects of the legs. To transmit power to the legs, XL timing belts were used because belted transmissions tend to have low backlash and the XL tooth pattern interfaces well with 3D printed pulleys.

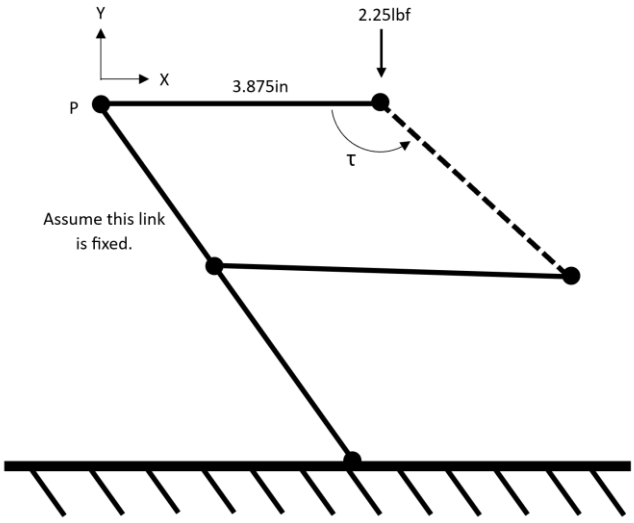

Figure 7: Leg Linkage Free Body Diagram

$$\sum \tau_P = 3.875 * 2.25lbf - \tau_{servo} \qquad [1.1]$$

$$\tau_{servo} = 8.71in * lbf = 140oz * in$$

To maintain a safety factor of 2x, an output torque of 280oz*in of torque as necessary. Additionally, an external ratio of 1:3 was used to change the range of travel from 200 degrees of the servo to 66 degrees so that the entire range of travel of the servo could be used. The resulting servo of choice was the EcoPower 110T, a servo that was already readily available to this project. The integration of these servos can be seen in Figure 8.

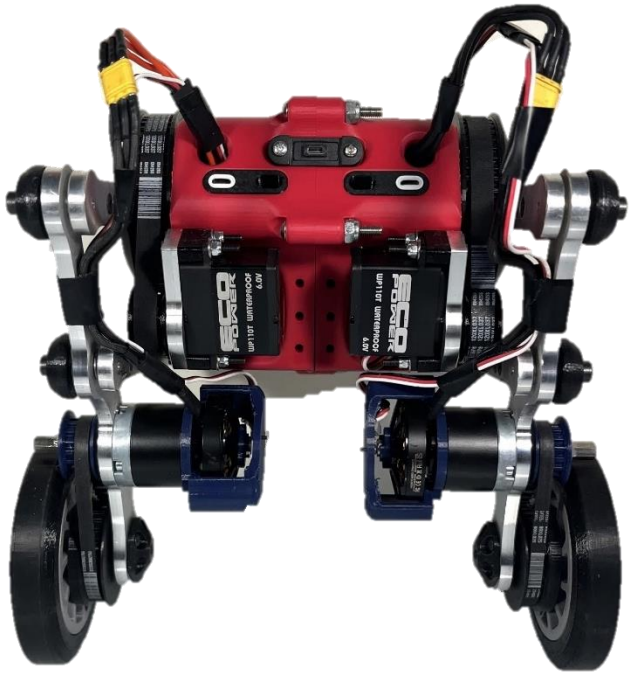

Figure 8: Back View of Robot

4.1.3 Wheel Power Transmission

It was decided that brushless motors would be used to power the wheels of this robot because brushless motors are lightweight and very powerful. However, brushless motors tend to struggle with low RPM commutation. Because of this, calculating the output torque of a brushless motor is not as simple as just reading a data sheet because it heavily depends on the motor controller tuning that it is coupled with. For industrial sized brushless motors, there are off the shelf solutions that do have these ratings very well documented, but since this project uses motors that were aimed towards drones, the advertised specifications were not to be trusted.

Based on prior experience with motors of this size, it was clear that whatever motor we chose would be able to provide significantly more torque than what was necessary for the balance of the robot. The factor that was more important in this decision was the speed of the motor and what the maximum speed of the wheel should be. It was decided that the target maximum speed of the robot should be 3 feet per second. It was clear that there would have to be a significant gear reduction to match the maximum speed of the brushless motor to the desired output speed of the wheel. An off the shelf planetary gearbox was a simple way to add a reduction to the power train, and after doing some market research, the Long Robotics 19:1 planetary gearbox was chosen. Given the target maximum speed, an estimate of the reduction, and the operating voltage of the motor, the kv of the motor can be calculated.

$$RPM_{motor} = kv * volts \quad [2.1]$$

$$RPM_{wheel} = RPM_{motor} * \frac{1}{19_{gearbox}} * 1.2_{belt} \quad [2.2]$$

$$V_{linear} = RPM_{wheel} * 2\pi * \frac{1_{sec}}{60_{min}} * 2_{inch} * 1/12_{in/ft} \quad [2.3]$$

Therefore,

$$kv = \frac{3\frac{ft}{sec} * \frac{1}{12} * \frac{1}{2} * \frac{1}{2\pi} * 60 * 19 * 1.2}{16.8} = 255$$

Thus, a roughly 250kv brushless motor would satisfy the speed requirements of this robot. The MAD Components 4008 250kv brushless pancake motor was selected. A custom titanium shaft for this motor was machined to interface the motor with the Long Robotics gearbox.

Additionally, the material chosen for the tires was 85A Ninjaflex TPU, a flexible 3D printing filament that would provide adequate grip. The implementation of the motors and wheel power train can be viewed in Figure 8.

4.1.4 Body Design

The body requirements of this robot were abstract and not well defined since the only major requirements of it were to house the electronics, connect the two legs together, and provide an attachment point for payload devices. Because of this, some artistic liberties were taken to design a body that was visually appealing, and much of the shape stems from this.

The body consists of two primary areas: an upper and lower compartment. The upper compartment houses the motor controllers, the power distribution, and voltage regulation components. The lower compartment houses the microcontroller, IMU, receiver, and the battery. The body itself is a clamshell design made of two halves, which makes the body easier to be 3D printed on an FDM printer. It also has a lid that provides access to the power distribution compartment and has mounting holes for payloads. On the front, there is a panel to access the battery. These components can all be seen in the cross section view shown in Figure 9.

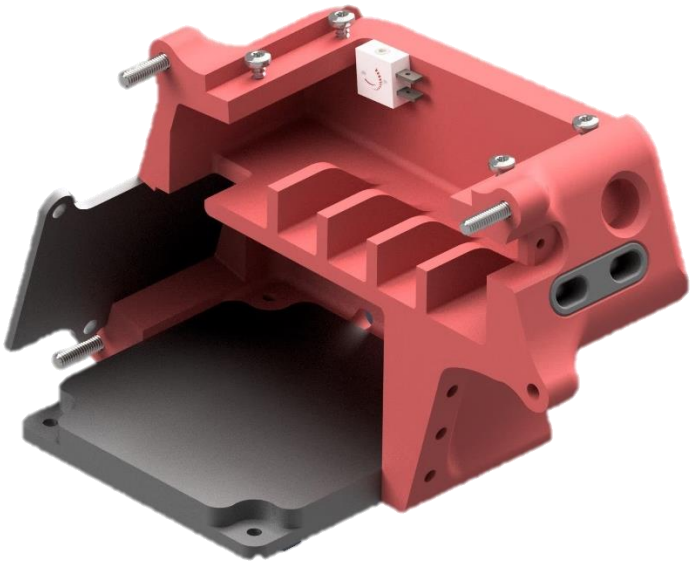

Figure 9: Section View of Body

The bottom of the body is a separate tray that holds all the components of the lower compartment. This tray can be easily removed from the robot for easy of maintenance. The back of the body consists of multiple holes that are filled with Micro USB ports that provide convenient access to the microcontroller and the A50S VESCs. Figure 10 below shows the physical implementation of the electronics tray.

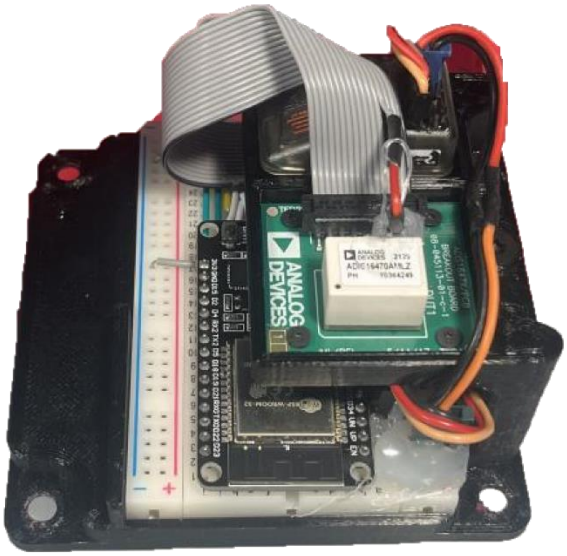

Figure 10: Electronics Tray

The primary material for the body was PLA+, which was chosen for its affordability and printability.

4.2 Electronics Implementation

4.2.1 Brushless Motor Control

As previously discussed, VESC is an open-source brushless motor control firmware that is supported on several hardware platforms and is primarily aimed towards electric skateboards. The advantage to the VESC platform is that it is highly configurable, which means that it can be adapted to fit the needs of any hardware platform. However, this advantage comes with the huge disadvantage that nothing works out of the box and there is a lot of work necessary to reach even a minimally viable setup.

At a high level, a brushless motor controller uses an estimation of the rotor position to determine which coils to charge, which produces a magnetic field that is locally optimal for the position of all the magnets and thereby produces the maximum torque. This closed-loop feedback controller is used until a certain ERPM threshold, where the motor switches to a timing-based controller. However, position estimation of the rotor can be a challenging task. The VESC platform supports both sensorless and sensored control to achieve commutation at low RPM. Sensorless mode with high frequency injection (HFI) enabled continuously measures the inductance of the motor to estimate the position of the rotor, whereas sensored mode uses an encoder to measure the actual position of the rotor.

Originally, the plan was to use the VESCs in sensorless mode to simplify the mechanical and electrical requirements of the system. However, the results of early testing proved that this plan was not going to work. The VESC simply lacked the fine control at low RPMs that was necessary for the balancing algorithm. The motor output was inconsistent and not responsive. It would take anywhere between 50ms and 200ms for the motor to respond, and when it did, it did so with very little torque. Occasionally, when it started spinning, it would rotate in the other direction before taking off in the correct direction. This behavior was unacceptable for the balance controller.

The solution was to use the VESCs in sensored mode with an inductive encoder. VESCs natively supports communication with encoders over SPI, I2C, and ABI hall effect encoders. However, all of these communication algorithms imply one-to-one communication between the encoder and the motor controller. For this project, it was useful to be able to measure the motor position from the microcontroller as well, which meant that these one-to-one communication protocols would not suffice. Instead, a simpler form of communication like PWM is preferred since it can easily be listened to by multiple devices. However, VESCs do not natively support PWM encoders.

To accommodate these issues, a custom release of the VESC firmware was written that supports

PWM inductive encoders. This custom firmware supported the A50S VESCs and the AS5048A inductive encoders that were selected to measure the rotor position.

This firmware release was based on the public VESC fork by Seems Reasonable robotics found here: https://github.com/seems-reasonable/bldc

4.2.2 VESC Signal

VESC supports multiple forms of communication, including PWM and DAC. At the time of this project's conclusion, a 50hz PWM signal is generated to communicate between the ESP32 and the VESC motor controllers. However, this signal frequency is not ideal because it limits the update rate of the motors to only 50hz. Though slow, the effect of this is not noticeable because the brushless motor is not able to respond at this rate anyways.

4.2.3 IMU Communication

Analog Devices donated an ADIS16470 IMU to this project to be used as the primary input into the balance controller. The ADIS16470 only supports SPI communication, so that is the protocol that exists between the ESP32 and the IMU to communicate. The pins that need to be connected and actively controlled are: 3.3V, GND, MISO, MOSI, CS, RST, DR.

4.2.4 ESP32 Pinout

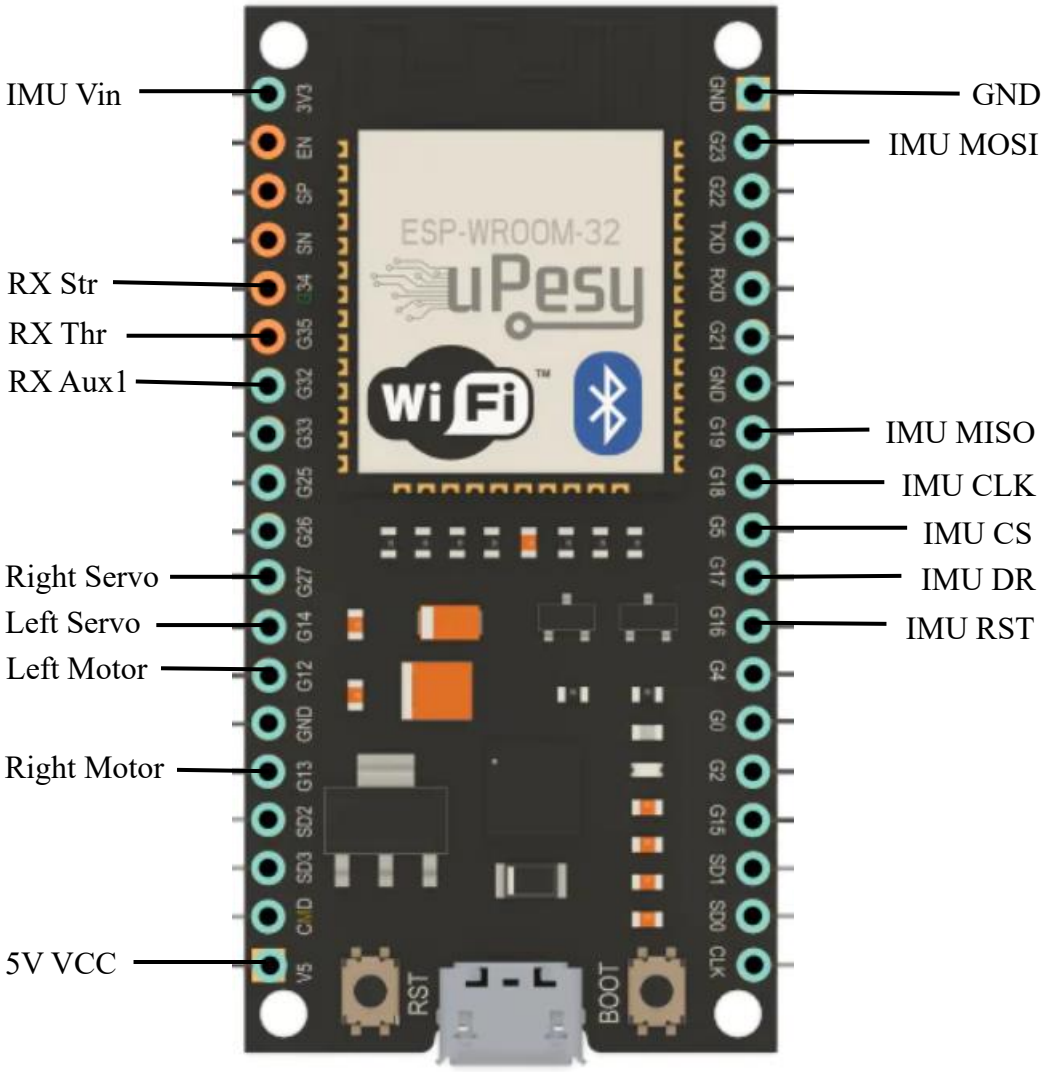

Figure 11: ESP32 Pin Diagram

4.3 Software Implementation

Though controls was the focal point of this project, there is a significant amount of software necessary to create an overarching framework that interacts with all sensors and actuators.

4.3.1 PlatformIO

All software for this project was written in C++ using the PlatformIO extension for VSCode using the Arduino framework. PlatformIO provided a convenient way to manage building a large project, and made it very easy to interact with the ESP32 given its built in ESP32 integrations and serial monitor.

4.3.2 Software Architecture by Class

Main
This is the primary runner class that contains the highest level instructions for setup and loop. This class houses the instances of the control algorithm classes, the controller class, and input from the keyboard is handled here.

Controller
This class manages input from the RC controller that is used to remotely control the robot and drive it around. It is able to read PWM input from three channels of the RC receiver but could easily be configured to handle more PWM inputs.

PWM Monitor
This is a patternable class used to read PWM signals at the lowest level. It manages an interrupt that is attached to a specified pin and continuously updates the value of that PWM signal in the background. This provides a time efficient method of reading a PWM signal that does not need to be managed after instantiation.

IMU
The IMU class is a wrapper class for interacting with various IMUs. This codebase supports both the ADIS16470 and the MPU6050, a common hobbyist IMU. Each IMU communicates in a different way, the ADIS16470 over SPI and the MPU6050 over I2C. The existence of this wrapper class makes it easy from a software perspective to switch back and forth between either IMU.

ADIS16470
This class manages the SPI transfer to communicate with the ADIS16470 IMU. On startup, an interrupt is attached to the data ready pin of the IMU, which then raises a flag

when active. During the main loop execution, the code checks this flag, and if it is active, performs an SPI transfer to read data from the IMU.

Motor Interface

This wrapper class serves as an interface between controller classes and motors. After instantiating this class, controller classes can simply send a desired power to the motors and this class handles the conversion to PWM wavelengths and performs checks on the validity of the given input to make sure that it is within a preset minimum and maximum.

Data Packet

One of the background capabilities of this codebase is real time datalogging. This class represents a packet of data that contains a timestamp and infinitely many datapoints. The intention is that other areas of the codebase have a reference to the data packet that is associated with each time stamp and are able to write data to that packet. The packet is then logged at the end of every loop iteration.

Math Utils

This file is included at the root of most files throughout the codebase. It includes useful calculation functions like bounding functions, dot product and vector multiplication. Importantly, it also contains a struct that represents a pose in 3D space.

PID

The PID file contains a struct that defines a set of PID constants and variables necessary to execute a PID controller. This is a patternable class that makes it easy to implement several PID controllers across the robot without repeating the core PID calculations.

Balance

This is the primary class that contains all code regarding the controls of balancing the robot. In addition to containing the balance controller, it also contains a feed forward velocity approximation algorithm and several functions that allow other classes to interact with the balance controller.

Leg Control

This is the class that deals with control over the leg linkages. This class implements the PID controller for the legs, handles leg velocity control, and contains several functions that allow other classes to interact with the legs.

4.4 Controls Implementation

The heart of this project was the control algorithm used to maintain balance in both pitch and roll. To accomplish this, two separate PID controllers were implemented: one based on pitch and controlled by the wheels, and one based on roll and controlled by the legs.

4.4.1 Pitch Controller

The first step in implementing a PID controller is to identify what the control variable, the setpoint, and outputs should be. Additionally, since this robot is more complex than a simple implementation of an inverted pendulum, this controller had to be agnostic of the height of the robot. This means that the controller had to have built in integrations for the center of mass of the robot changing. The final block diagram of the pitch controller can be seen in Figure 13

Control Variable

At first, it seemed like the logical control variable for this controller would be the angular offset from the vertical. That is, the angle at which the robot is leaning. Conceptually it makes sense that an algorithm that controls the angle of the robot directly could work. However, in practice, this control variable turned out to not be the optimal choice. Since the robot acts like an inverted pendulum as shown in Figure 12, the result of a controller with pitch as the control variable has non-linear output. In theory, this can be compensated for with careful PID constant tuning, but this controller will never succeed when the robot changes heights.

After further inspection, it became clear that the variable that matters for balancing is the displacement of the projection of the center of mass onto the horizontal as shown by Figure 12. This creates a linear input to the PID controller, which makes tuning more forgiving, and makes the controller agnostic of the height of the robot.

Thus, the control variable chosen was $h * \sin(\theta)$, where h is the height of the robot and θ is the angular offset from the vertical.

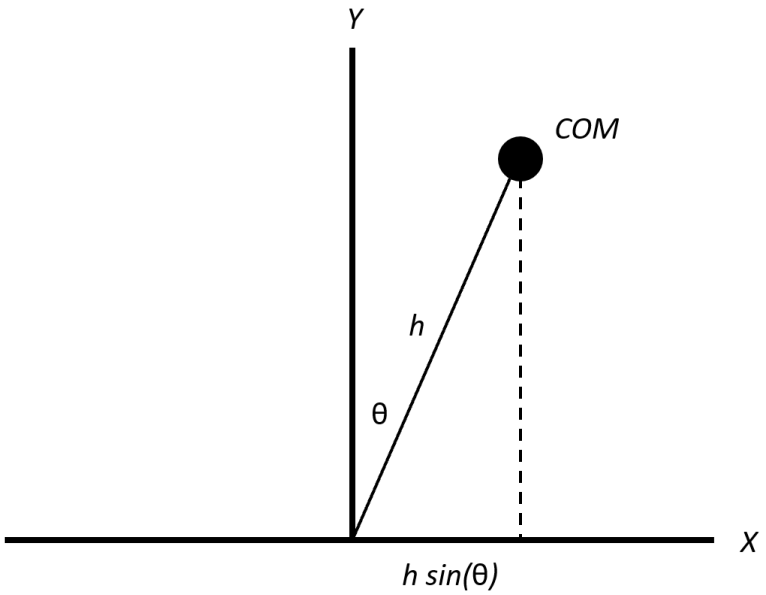

Figure 12: Projection of Center of Mass

Setpoint

Conceptually, it makes sense that if the robot was leaning forward and maintaining equilibrium, then it must be accelerating. Similarly, if it leans backward, then it must be decelerating. This leaning can be linearized similar to what is shown in Figure 12 to calculate the horizontal displacement of the center of mass. This idea indicates a relationship between the displacement of the center of mass of the robot and the desired acceleration. As shown in the control block diagram in Figure 13, the primary outside input to the controller is the desired velocity for the robot. If the robot knows its current velocity, then the error in velocity can be calculated to determine how much the robot should accelerate. Thus, there exists a relationship between the error in velocity and the displacement of the center of mass of the robot that can be used to calculate the setpoint of the PID controller. The exact nature of this relationship was never explicitly modeled, it is only known that one exists. To simplify the work required for this project, the relationship was assumed to be linear. This allowed an implementation of another separate P controller that uses the error in velocity as the control variable, 0 as a setpoint, and outputs the desired horizontal displacement of the center of mass.

Output

The output of the balance PID controller is simply a percentage motor power, where -100 represents full backwards and 100 represents full power forward. The output of this controller is then bounded within a maximum absolute value to help prevent runaway issues.

The Feed Forward Term

A traditional PID controller works well for balancing statically. However, when the robot attempts to maintain dynamic equilibrium, the model is slightly different because there is an additional torque required to maintain a non-zero angle. If, for a moment, we ignore the PID controller and attempt to balance the robot by just sending an open loop value to the motors, we can conceptualize that there are different ideal powers to send to the robot at different angles. For example, if the robot was attempting to balance statically, that is, maintain balance standing still, then ideally no active control is required to balance. However, if the robot was attempting to maintain an angular offset of say, 5 degrees, then there is some torque that needs to be continuously applied to the motors to apply a reaction torque to the whole robot. This torque must exist for every ΔT, which means that the ideal power to send to the motors sums over time. While this summing sounds suspiciously similar to the intention of the integral term of the PID controller, just using the I term to account for this is not a good solution because the I term is useful for reducing steady state error when the robot is balancing in static equilibrium. Thus, a feed forward term can be summed with the output of the PID controller that provides another cumulative term to account for the angular offset of the robot while balancing in dynamic equilibrium.

Integral Runaway and Reset

One of the classic issues with PID controllers is buildup of the integral term over time. Since the integral term is an accumulation of the error over time, residue from early in the run can persist and contribute to the behavior of the controller as much as 10 minutes later when that information is no longer relevant. This causes the behavior of the controller to vary over time. The solution to this problem was to define criteria for when to reset the integral term, as well as when to bound it to a specific value. The implementation of this PID controller specified a range in which the integral term would reset if both the error in position and change in error over time fall beneath a certain threshold. Additionally, it also allowed for the definition of a maximum possible value of the integral term to prevent integral runaway.

Control Block Diagram

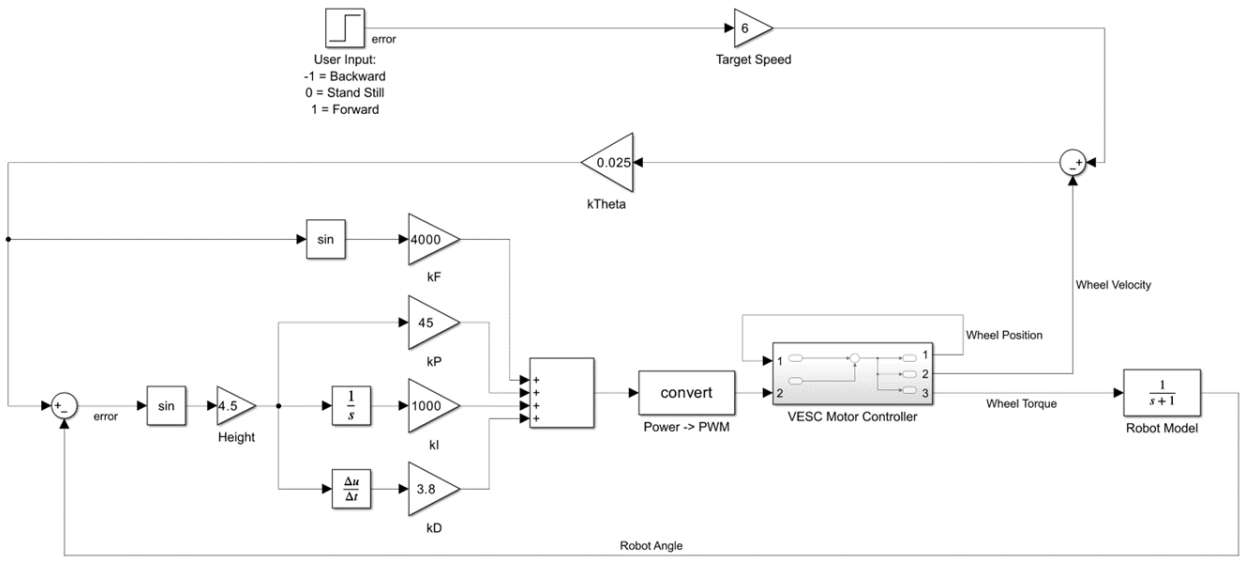

Figure 13: Pitch Controller Block Diagram

4.4.2 Roll Controller

An auxiliary behavior of this robot is to maintain balance in roll as well as balance in pitch. This is accomplished by actively actuating the legs to respond to external stimuli like driving over uneven surfaces. The idea is that if the robot drives over non-level surfaces or is pushed from the side, the legs can adjust accordingly to maintain balance. This was accomplished using a very rudimentary P controller, with the control variable being the roll of the robot, the setpoint being 0, and the output being a desired height of each leg. The desired height of the legs were then converted to a signal to send to the servos, which natively handled the position control.

5 Results and Discussion

5.1 Hardware Discussion

5.1.1 Leg Linkage

The implementation of the four-bar leg linkage overall went very well. The legs were robust and durable, and throughout the course of the project, never physically failed. The legs achieved their original intention of achieving a linear range of travel for the wheels which simplified the amount of work that the balance controller had to compensate for.

However, there are a few deficiencies of this leg design, mostly that reside in the implementation of the leg joints. The leg joints consisted of 3/16” shoulder bolts and ball bearings, which proved to have more slop than originally intended. This caused the body of the robot to be able to wiggle and shift side to side, which introduced noise into the roll readings. There was only roughly 1 degree of wiggle, which was mostly insignificant but still not ideal.

Overall, the hardware implementation of the legs was a resounding success.

5.1.2 Wheel Design

The electromechanical design of the wheels proved to be the most challenging design problem of the entire project. The result of this design was a power train that successfully enabled the robot to drive and balance, however there was significant room for improvement in the drive. The insufficiency mostly comes from the gearbox that was chosen, as there is roughly 2 degrees of slop between the input and output of the gearbox. This introduced a challenge for the balance controller to overcome, which while surmountable, introduced a ceiling to how stable the robot could possibly be.

Additionally, it seems that the initial estimation of 3ft/s maximum speed was too slow. When the robot is balancing dynamically, it needs to be able to accelerate past its cruising speed on occasion to maintain balance. As discussed later in the results section, the maximum stable speed of the robot proved to be about 1.4ft/s, which indicates that it needed 1.6ft/s worth of speed overhead to facilitate the balance controller.

However, the drive train was overall a success as it was able to facilitate the balancing and driving of the robot. Additionally, the TPU wheels provided more than enough grip for most surfaces that were tested.

5.2 Controller and Software Discussion

At the end of this project, a robot that was able to balance both statically and dynamically was successfully produced. However, there were a number of unforeseen challenges that had to be overcome and some that still persist.

5.2.1 Deficiencies of PID Control

PID control is an elementary controller that is designed to work on linear systems with a clearly defined model. This makes PID control a great choice for controls of systems like robot arms where the conditions of the model are relatively consistent. However, for this project, there are various external influences that cause the controller to behave in ways beyond what it was initially designed for.

As discussed in the Feed Forward section of the Pitch Controller implementation, traditional PID control struggles with compensation for cases where the theoretical resting power of the system is not zero. This implies that situations like dynamic balancing and climbing hills are difficult for the controller to accommodate. While it is able to accommodate some disturbances like this, the robot is unable to climb slopes of more than roughly 5 degrees.

One way that the controller was designed to compensate for this issue was by relying heavily on the integral term. This meant that the robot would try to reduce the steady state error that would be introduced by slopes and dynamic balancing, but in turn, it would cause heavy oscillations while balancing statically.

5.2.2 Balance Controller Results

Despite the deficiencies of PID control discussed in the previous section, it did prove to be sufficient to facilitate balancing. In fact, it overall worked well enough to allow for stress testing and performance analysis in several tests. One of the tests that was conducted was a roughly 0.75 mile walk around the WPI campus that required the robot to traverse several different environments like sidewalks, roads, hills, indoor floors, and doorways. In that test, the following results were measured:

Maximum Speed: 1.4ft/s

Battery Life: >1hr

Average Time Between Falls: 14 min

Maximum Recoverable Angular Displacement: 3.5 degrees

Over the course of a 1 hour 10 minute run, the robot experienced 5 falls which were related to accidentally exceeding the maximum speed of the robot, difficulties with handling slopes, or crossing bumps in doorways. Despite this, a failure rate of only 1 fall per 14 minutes is exceptional given the scope of this project. Figure 14 below shows an example of the robot driving past the landmark fountain on WPI's campus.

Figure 14: Scout Driving Past the Fountain

5.2.3 Controller Response Example

Throughout the development of the control algorithm, it was useful to be able to graph the pitch of the robot over time, as well as the output of the PID controller. Figure 15 shows an example transient response of the system in response to an impulse input that was imparted by poking the robot while it was statically balancing. Here, the system was able to respond by oscillating back and forth until once again reaching equilibrium with low steady state error.

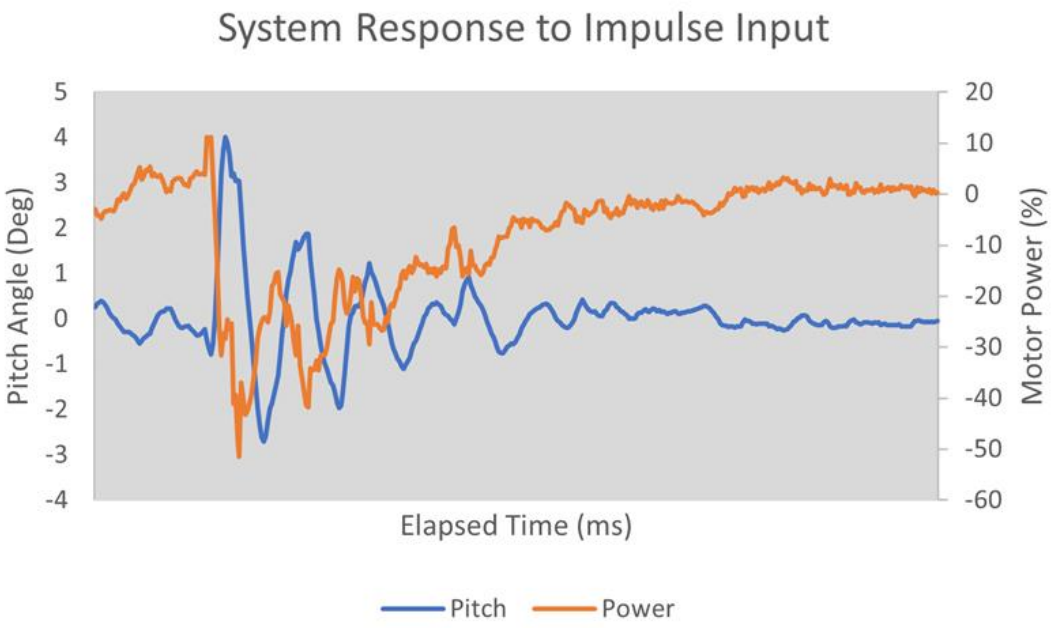

Figure 15: Example Transient Response

6 Recommendations for Future Work

One of the primary goals of this project given that this was its first year was to develop a hardware platform that could support complex control algorithms for later research. As such, the bulk of this project went into hardware development and left a lot of room for improvement in software.

Broken down into categories, here are some recommendations for how the robot can be improved by future project groups.

6.1 Hardware

- Redesign the drive subsystem of the robot to something that does not use planetary gearboxes. One possible idea is to pursue a hub motor design where the motor is directly inside of the wheels, since this would most effectively reduce the slop of the power train.
- Develop a new method for joining the links of the leg linkages that has less slop and is less susceptible to loosening than the current ball bearing and shoulder bolt solution.
- Develop a better system for mounting peripheral electronics and payloads to the top of the robot. Eventually, the goal of this project is to support several attachments like cameras and microphones, which is not currently possible on this platform.
- Implement brushless motors on the leg linkage. Currently, servos are used to power the legs, which allows for basic functions like balancing in roll. However, if the robot is ever to be able to hop and do stairs, then a much stronger actuator is necessary for this.

6.2 Electronics

- To go along with the hardware requirements of supporting peripheral devices, adding another processor to the electronics stack would be necessary for these peripherals. Future project groups could develop an architecture for running complex and slower algorithms like computer vision on an external SBC like a RaspberryPi and implement a communication line between the SBC and the ESP32 used for controls.
- Design a PCB for integrating all the electronics hardware. Currently, the microprocessor, sensors and actuators are all joined on a breadboard. This is not an elegant or permanent solution, since wires frequently come loose when the robot falls over.

6.3 Software

- Experiment with control algorithms besides PID. As discussed previously, PID falls short in a lot of scenarios that this robot could find itself in. Other more advanced control algorithms like MPC or LQR could provide a more powerful and robust method of controlling the balance of the robot.
- Determine a better way of measuring the robot's average velocity. Currently, velocity is measured by applying a moving average filter to the output power to the wheels. However, this method is not sufficient because the PID controller that outputs this power can oscillate significantly. Measuring the robot's velocity goes hand in hand with tuning the balance controller.
- Apply a rotation to the IMU data. Currently, measurements for roll, pitch, and yaw are not mutually exclusive. For example, rotating the IMU in pitch by 90 degrees could cause a 1 degree change in the reading of roll. This is likely because the IMU does not physically align with the casing that it is in. This can be adjusted for in software by applying a transformation matrix to the IMU data.

7 Conclusion

This project successfully produced a hardware platform that could perform the basic tasks of a wheeled bipedal mobile robot. It began to explore some of the complex kinematics involved in balancing. This robot serves as a solid proof of concept for the validity of such a design as it shows that a balancing robot can be robust and stable. This project produced a relatively small robot as to fit within budget constraints, however the kinematics of such a robot would clearly scale up to a larger, similar robot.

Within the scope of a one-year project completed by only one engineer, the resulting robot exceeded expectations and should provide a promising platform to be further developed in future projects. This project was never intended to be completed within the scope of only one year, and as such, it will take time for the full potential of this platform to become reality. This project provided a strong basis for all future work and should enable future researchers to skip a few of the low-level steps that were accomplished in this project.

The final physical product of this project can be seen below in Figure 16.

Figure 16: Scout Final Product

Appendix A: Bill of Materials

Item	Description	#	Source	Unit Cost	Ext. Cost
A50S VESC	Brushless Motor Controller	2	TeamTriForceUK	$ 99.99	$ 199.98
MAD 4008	Brushless Drive Motor	2	MAD Components	$ 68.00	$ 136.00
AS5048A	Inductive Encoder	2	Amazon	$ 16.10	$ 32.20
Diametric Magnet	MAGNET 0.315"DIA X 0.098"H CYL	2	Digikey	$ 0.82	$ 1.64
Planetary Gearbox	Long Robotics 19:1	2	Long Robotics	$ 18.32	$ 36.64
3/16" Shoulder Bolts, 1"	Linkage Joint Bolts	2	McMaster	$ 3.02	$ 6.04
3/16" Shoulder Bolts, 0.75"	Linkage Joint Bolts	6	McMaster	$ 2.49	$ 14.94
1/4" x 1.25" Shoulder Bolt	Drive Shaft	2	McMaster	$ 1.76	$ 3.52
Fasteners	Assorted #6 and #8 Plastite Screws	1	McMaster	$ 25.00	$ 25.00
3/16" x 3/8" Bearing (10pk)	Linkage Bearings	2	Amazon	$ 12.00	$ 24.00
1/4" x 1/2" Bearing	Wheel Bearings	1	Amazon	$ 10.00	$ 10.00
ESP32	Microcontroller	1	Amazon	$ 17.99	$ 17.99
XL Timing Belts (120 and 90)	Power transmission	4	McMaster	$ 7.50	$ 30.00
Red PLA+	Body Filament	1	Amazon	$ 24.99	$ 24.99
Black PLA+	Miscellaneous Part Filament	1	Amazon	$ 24.99	$ 24.99
Black Ninjaflex	Wheel Filament	1	Amazon	$ 58.99	$ 58.99
6" MicroUSB Extension	ESP32 Extender	1	Amazon	9	$ 8.99
3" MicroUSB Extension	VESC Extender	2	Amazon	$ 6.99	$ 13.98
SR315	Spektrum Receiver	1	Amazon	$ 54.99	$ 54.99
ADIS16470 IMU	Gyroscope/Accelerometer	1	Analog Devices	$ 400.00	$ 400.00
GNB 4s 850mah	LiPo Battery	1	Amazon	$ 38.99	$ 38.99
EcoPower110T	Leg Linkage Servos	2	Amazon	$ 39.99	$ 79.98
Castle 10A BEC	6V Voltage Regulation	1	Amazon	$ 24.99	$ 24.99
iFlight 2A BEC	5V Voltage Regulation	1	Amazon	$ 12.99	$ 12.99

XT30, MR30 Connectors	Wiring	1	Amazon	$ 30.00	$ 30.00
				Total:	$1311.83

References

[1] Ascento Robotics | automated Outdoor Security Patrolling. “Ascento Robotics | Automated Outdoor Security Patrolling,” n.d. https://www.ascento.ch/.

[2] Boston Dynamics. “Research | Boston Dynamics,” n.d. https://www.bostondynamics.com/research.

[3] ANYmal on Wheels – Robotic Systems Lab | ETH Zurich. “ANYmal on Wheels,” n.d. https://rsl.ethz.ch/robots-media/anymal-wheels.html.