Final Design Report

Project Title: *Ghost Rider*

Course: EE 327 - Electronic System Design

Team Number: *13*

Quarter: Spring 2025

Instructor: Prof. Ilya Mikhelson

Submitted by:

Michael Kim	michaelkim2026@u.northwestern.edu
Milan Shah	milanshah2026@u.northwestern.edu
Michael Jenz	michaeljenz2026@u.northwestern.edu

Contents

List of Figures

List of Tables

Abstract

This quarter we worked to develop the GhostRider, a self-balancing unicycle. The initial goal of this project was for the unicycle to be able to stabilize itself against outside disturbances, as well as move to arbitrary locations around a room based on controls from a website. The GhostRider has a DC motor driving its wheel, as well as a servo-motor-controlled counterweight. Together, these two motors can drive the unicycle forward and backward and tilt the unicycle left and right. Midway through the quarter we decided to scale our project goals back and focus solely on achieving stability with the unicycle. In order to obtain stability, proportional-integral-derivative (PID) controls were used to actuate both of the motors. The PID controls were fed information from an inertial measurement unit (IMU), from which we could discern the angle and angular velocity of the unicycle. Throughout the design process, we faced numerous setbacks, including short circuits, component malfunctions, and mechanical faults. Additionally, we discovered that PID controls, which try to force the error to be zero at all times, may not be sufficient to control a dynamic and nonlinear system such as a unicycle. Ulitmately, the GhostRider prototype is not able to remain upright on its own. However, with light support, it is able to remain upright and demonstrate its self-stabilizing capabilities. In the future, a more robust nonlinear control method could be used to further improve its capabiltiies.

1 Introduction

The GhostRider is a self-balancing unicycle developed to investigate the control and dynamic challenges associated with balancing a single-wheeled vehicle. Beyond its technical significance, the device also serves as an engaging demonstration platform for showcasing dynamic balancing systems at public and private venues. Its visually striking motion and engineering novelty make it well-suited for attracting attention at restaurants, concerts, and similar events.

This project draws inspiration from the classic control systems problem of stabilizing an inverted pendulum—a canonical example used to evaluate and implement control strategies. In theory, such systems can be stabilized using conventional techniques, most notably Proportional-Integral-Derivative (PID) control. PID controllers continuously adjust system inputs based on current, accumulated, and predicted future error, and are widely used in industrial automation and robotics.

To estimate the unicycle's orientation and angular velocity, the system employs an Inertial Measurement Unit (IMU), which includes both a gyroscope and an accelerometer. When at rest, the accelerometer provides an estimate of tilt based on gravitational acceleration. The gyroscope, through integration over time, also provides orientation data, but with greater susceptibility to drift. By fusing the outputs of both sensors, we obtain a more accurate and less noisy estimate of the unicycle's state, which is then used to drive the PID control loop.

Translating these control principles into a real-world self-balancing unicycle, however, introduces significant challenges. Unlike the fixed inverted pendulum, a unicycle must maintain continuous motion to generate gyroscopic forces essential for stability. This requires carefully modulated oscillations in the forward and backward direction. Furthermore, balancing in the lateral (side-to-side) direction is particularly difficult due to the unicycle's narrow base of support and lack of inherent lateral stability. Real-world systems must also contend with a range of issues absent in simulation, including sensor noise, latency, and imperfect measurements. Our system attempts to replicate the reactive control behavior of a human rider through real-time sensor fusion and software-driven PID feedback.

Although commercial self-balancing unicycles exist [1], they typically feature design optimizations—such as low centers of gravity and wide wheels—that simplify the balancing task. In contrast, our design retains the structure of a standard children's unicycle, resulting in a significantly more complex control problem. By retrofitting a conventional unicycle using off-the-shelf components, our approach serves as a novel and exploratory investigation into the practical limitations of control algorithms when applied to underactuated, dynamically unstable systems in real-world conditions.

2 Design Constraints and Requirements

Since the GhostRider project involved modifying a full size consumer product with motors and electronics, we required large amounts of power output. In this section, we discuss the specifications that we developed for our system based on our estimations at the start of the project. These specifications for our components used in this project (motors, controllers, batteries) played a large role in the success of our project later on. Therefore, it is worthwhile to examine the constraints and requirements that we set for ourselves at the start of the project.

2.1 Constraints

2.1.1 Budget

The main constraint in this project was the budget. Our project was given a budget of $100 per student, totaling $300. This magnitude of this constraint was first realized during our component selection phase of the project. Without getting bogged down into the details, our project required a motor, motor controller, IMU, and a unicycle. If we were to get all of the best of these components, we would far exceed our budget. Given this situation, we were forced to balance our design needs and purchase higher quality components while compromising quality for others. In the end, our team was able to keep within budget, spending a total of $219.45. We were able to do this by choosing a child-size unicycle and using cheap motors while splurging on the IMU and motor controller.

2.2 Requirements

The first step of the design process is to specify the design requirements. For the GhostRider, this meant analyzing the dynamics of the system to estimate motor and battery requirements. This section provides an overview of our method.

2.2.1 Wheel Motor Torque

The wheel motor torque required should be at least equal to the torque that the unicycle will produce as it falls down. As you can see in 2.3 this force can be analyzed in the side plane or the wheel plane by estimating the height of the center of gravity, and multiplying by the weight of the unicycle.

Figure 2.1: Torque in front plane

Using the specifications provided on the unicycle manufacturer's page, as seen in 2.1, we were able to estimate this maximum torque around the wheel. We estimated the center of mass to be at half the height of the unicycle which is an overestimation to ensure proper motor specification. Then, we subtracted the height of the wheel axle from this and used the resulting length as that of the torque around the wheel. This calculation can be seen below:

$$\tau_{1max} = l \times m = 20cm \times 2.5Kg = 50Kg * cm$$

With this specification now know what our motor's peak torque needs to be.

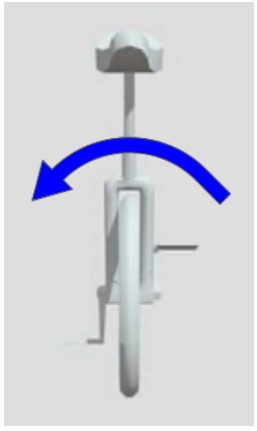

Figure 2.2: Torque in front plane

Figure 2.3: Torque in side plane

2.2.2 Wheel Motor Speed

For our motor selection, it was also helpful to estimate a maximum speed output that the motor would need to output. We approached this problem by estimating the distance that the unicycle might need to travel in a second. Using this approach, we were able to over estimate the maximum speed necessary from the motor and thereby provide a safe specification for the wheel motor speed output.

We estimated the maximum distance that the wheel would need to travel in half a second to be the length of the unicycle since traveling this length would be able to save the unicycle seat from hitting the ground in a worst case scenario. Another way of saying this is that the wheel should spin twice the height of the unicycle in one second. Since the wheel diameter is $16in$ this gives the circumference as roughly $50in$. With this information, we can estimate the maximum motor speed output:

$$\omega_{max} = ((2 \times c) \div h) \div 1sec = ((2 \times 50in) \div 32in) \div 1sec = 1.28RPS$$

This result is not as it may appear at face value. Since this is calculated within a worst case scenario situation, the maximum load is likely applied to the motor at the same time as this maximum speed requirement is used. Therefore, this maximum speed requirement cannot be used as a benchmark for the motor no-load speed. Instead, we should ensure that the motor is capable of moving at $1.28RPS$ while it is outputting the maximum torque requirement.

2.2.3 Counterweight Total Weight

In addition to balancing the unicycle in the side plane, we also need to balance the unicycle in the front plane as seen in Figure **??**. Since the wheel does not move in this front plane direction to balance the unicycle, we decided to use a counterweight to balance the unicycle in this plane. The first task with this design was to estimate the weight of the counterweight required such that the shift in the center of mass is effective. While this might seem complicated, all that is necessary for this condition to be fulfilled is to move a mass that is a majority of the weight of the unicycle past the actual center of mass of the unicycle. Therefore, we can set out specified weight for the counterweight as 51% of the weight of the unicycle, or approximately $2.26Kg$.

2.2.4 Counterweight Motor Torque

To get a exact number for the length of the moment arm for this counterweight, we can examine an edge case. The case we can examine is the maximum recoverable angle. We set this angle as 45 since we cannot expect the unicycle to recover after reaching any incline lower than this. As you can see in Figure 2.4, the length of the moment arm for the counterweight in this case is exactly equal to the height of the counterweight motor divided by the square root of two.

Using this equation we can estimate the length of the moment arm, as seen below.

$$L = h/\sqrt{2} = 71cm/\sqrt{2} \approx 50cm$$

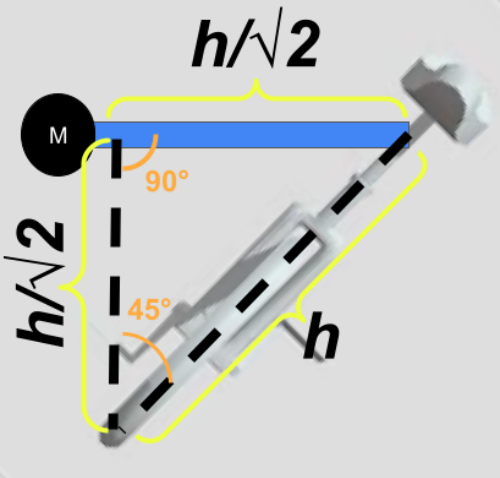

Figure 2.4: Counterweight moment arm length diagram

Now with the counterweight weight and the length of the counterweight moment arm we can calculate the maximum torque required by the counterweight motor, as seen in the calculation below.

$$\tau_{2max} = l \times m = 50cm \times 2.26Kg = 113Kg * cm$$

This specification can serve as the starting place for the peak torque required by the counterweight motor.

2.2.5 Battery Capacity

To estimate the size of the battery that we will need, we used characteristics of the motor current consumption to specify the battery capacity. The only other parameter that we needed to specify was the lifetime of the battery. Since we were only interested in having the unicycle run for short tests, we set the lifespan as 30 minutes. Then, we estimated the average current draw of the motor by using its peak current ($7.4A$) and rated current ($2A$) to derive the average. We estimated that the motor would only spend 10% of its time outputting the maximum torque/current. With this estimation, we found the average current consumption of the motor to be $6.86A$. With this information, we could then calculate the battery capacity:

$$Capacity = I_{avg} \times lifespan = 6.86A \times 0.5h = 3.43Ah$$

2.2.6 Battery C Rating

Another aspect of battery selection is the C rating, which is essentially the rate that the battery can be discharged relative to its maximum capacity. We need to establish a floor for this value to ensure that the motor can draw enough current. The minimum C rating that we require can be calculated as seen below:

$$C_{min} = I_{max} \div Capacity = 7.4A \div 3.43Ah \approx 2.2(1/h)$$

This C rating is low, especially for Lithium Polymer batteries, therefore, it was not much of a concern for us during battery selection and the project overall.

3 Engineering Standards

3.1 Programming

While PID controllers can theoretically stabilize a unicycle, this is only feasible under ideal conditions. One such condition is that the PID loop operates instantaneously. In other words, we assume that the control system is not band-limited. However, due to sensor sampling, signal processing, and actuation delay, our PID loop is band-limited. Thus, minimizing this latency is critical to improving control performance. The lower the system's total group delay, the broader the class of dynamic plants (including unstable ones like a unicycle) that can be effectively stabilized using PID control.

To address this, we focused on optimizing the computational efficiency of the signal processing pipeline. For example, in our implementation of the Madgwick filter, we replaced the standard square-root operation with Quake's fast inverse square root. Furthermore, we deployed all control and sensing routines using a real-time operating system (FreeRTOS) to make sure that everything runs as fast as possible. We also maximized the IMU sampling rate.

Another significant constraint was our limited budget. Due to cost limitations, we were unable to acquire motors, batteries, and structural components capable of supporting the torque required to balance a human rider. Based on our estimates, achieving the necessary torque and responsiveness would require motors in the $300+ range, which exceeded the budget for this project. As a result, our system was designed to have a "ghost" or massless object as the rider.

3.2 Safety

Safety was a central consideration in the design and implementation of our self-balancing unicycle. Since the system operates at currents up to 5 amps and involves fast-moving mechanical components, there was considerable potential for electrical and physical hazards. To minimize these risks, we adopted a number of precautionary measures throughout the development process.

Initial system testing was performed with the unicycle suspended using harnesses. This setup allowed us to evaluate control behavior and sensor performance without risking damage to mechanical components or nearby objects. Electrically, we ensured that all subsystems were properly isolated. We also selected components that were significantly overspecified relative to their expected loads. This provided a critical safety margin against unexpected

current spikes or component failures.

During extended testing, we observed overheating in the voltage regulators. To address this, we added heat sinks to improve heat dissipation, as the voltage regulars could get over 160 degrees F. For high-current power lines, we wanted to use low-gauge wiring to increase safety.

We also took precautions to protect our development equipment. After experiencing a pretty sizable over voltage to Milan's laptop during early firmware testing, we implemented a policy of programming and debugging the system with laptops disconnected from power lines to prevent backflow that could damage USB ports. Additionally, we incorporated safety switches into the power circuit, allowing us to cut off power quickly in the event of something going wrong. We also considered damage to the LiPo battery as we tested the PID control systems, making sure that we are not taking unnecessary blows to the system.

4 Broader Considerations

We chose this project, fitting a self-balancing system onto a pre-existing unicycle, as a novel exploration of control systems. Our project highlights the limitations of the PID control algorithm and improves upon existing sensor fusion techniques.

For example, throughout school, engineers are taught that PID control "just works" in 99% of cases. However, this is one of the rare cases where PID control does not work effectively. We must look at how humans tackle the problem of unicycle balance and find an algorithm that better matches that approach. The iLQR algorithm (see Section 5.3) takes an ergodic control approach, where instead of greedily trying to keep the unicycle upright at every moment, we attempt to keep the unicycle's time-averaged position upright. Just like we need to sway to ride a bike, we also need to sway to ride a unicycle.

This is the key technical insight of our project, thinking about a given problem less greedily and more ergodically.

From an ethical and safety perspective, any technology that involves autonomous vehicles should consider safety risks for both the user and bystanders. While our current prototype is not intended for use, any future iterations or commercial products need a variety of safety regulations. For example, we propose they use fail-safe mechanisms for both electrical components and the underlying control system. An important thing to consider is to monitor battery stability in the event of crashes (LiPo batteries are known to be dangerous) and make it easy for the user to bail or take manual control of the unicycle in case the control system becomes unstable. For example, a high-frequency, rapidly oscillating wheel is not ideal for safety.

We currently collect only Euler-angle data from the IMU, which is transmitted via a web interface. For privacy reasons, it is important that this data is not stored in production. Additionally, it would be beneficial to implement over-the-air (OTA) update capability as well as on-device telemetry logging in case a fault occurs.

5 Design Description

5.1 System Overview

Our system revolves around data from the IMU, which provides information about the angle and angular velocity of the unicycle. The algorithms and code used for filtering and sensor fusion will be described further in section 5.3. For now, the IMU provided us with accelerometer and gyroscope data, which identifies how much the unicycle is tipping over in either the sagittal or coronal plane. This information feeds two separate PID controllers – one for each plane – which attempt to correct for the error. We define error as the difference between the unicycle position and perfectly upright, essentially trying to force the shaft to be vertical. The control signals generated by the PID controllers are actuated by the wheel motor and the servo arm counterweight.

5.2 Block Diagrams

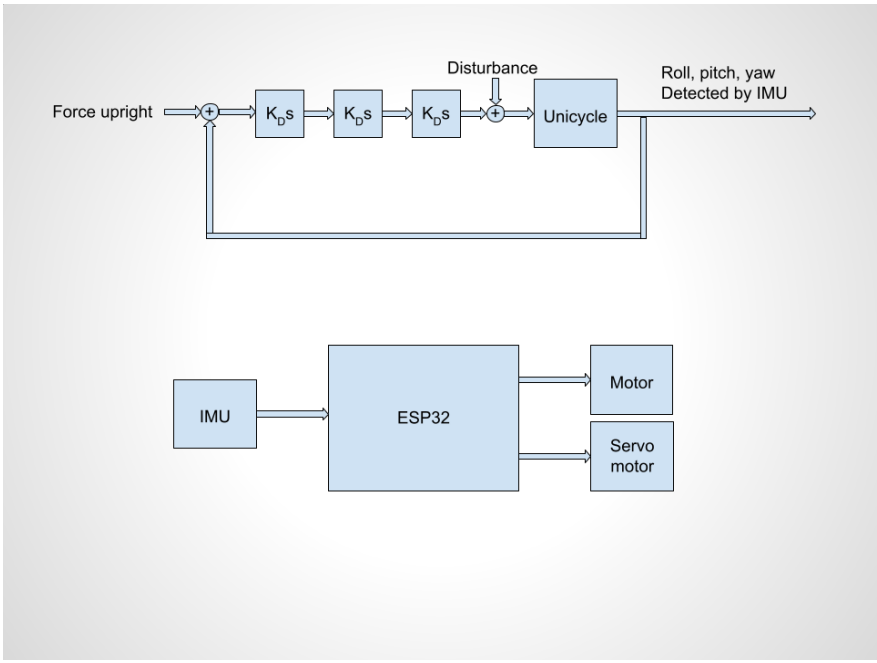

Figure 5.1: System Block Diagram

5.3 Algorithms and Code

Our firmware is developed using the Arduino framework and runs on an ESP32 microcontroller utilizing the FreeRTOS real-time operating system. The primary entry point is the main.cpp file, where multiple tasks are initialized and scheduled. Each task is encapsulated within its own module and follows a singleton design pattern to ensure that only a single instance of each task can be instantiated, thereby avoiding duplication and ensuring consistency across the system.

Each task module includes a dedicated header file that defines a Config struct, which centralizes all task-specific parameters. Tasks are initialized via a standardized getInstance() →startTask() interface, promoting a modular codebase.

To ensure thread safety in this concurrent system, we make extensive use of xSemaphore, queues, and mutexes provided by the FreeRTOS framework. These synchronization primitives are managed centrally through the SyncObjects.cpp file, which provides static access to shared mutexes, queues, and other inter-task communication mechanisms. This architecture allows us to maintain clean task separation while ensuring reliable and safe access to shared resources.

Table 5.1: Overview of Codebase Files and Their Functions

Filename	**Description / Function**
main	Main program entry point, initializes tasks
BikeMotor	Motor control for unicycle drive
Constants	Contains constant values used throughout system
Encoder	Reads motor encoder for velocity estimation
IMU	Reads and processes IMU sensor data
ServoArm	Controls sideways balancing servo motor
SyncObjects	Manages FreeRTOS queues, semaphores, mutexes
WebServer	Hosts web interface for system monitoring

5.3.1 IMU Sensor Fusion

To estimate the orientation of the unicycle in real-time, IMUs that provides raw accelerometer and gyroscope data. However, both sensor modalities are subject to different types of noise: accelerometers are susceptible to high-frequency vibration noise, while gyroscopes suffer from integration drift over time. To obtain a reliable estimate of orientation, we apply sensor fusion using the Madgwick filter.

The Madgwick filter is an efficient quaternion-based algorithm designed for real-time attitude and heading reference systems (AHRS) [2]. It combines gyroscope, accelerometer, data to estimate orientation by minimizing the error between predicted and measured direction of gravity. The filter uses a gradient descent optimization algorithm to compute the direction of the gyroscope error and apply corrections accordingly.

The steps in the filtering process are as follows:

1. Normalize the accelerometer measurement to isolate the gravity vector.

2. Estimate the quaternion derivative from gyroscope readings.
3. Compute the error between the measured gravity vector (from accelerometer) and the predicted gravity vector (from the current orientation estimate).
4. Use a gradient descent algorithm to minimize this error and correct the orientation.
5. Integrate the corrected quaternion derivative to update the orientation estimate.

Mathematically, the orientation quaternion q is updated as:

$$q(t + \Delta t) = q(t) + \frac{1}{2} q(t) \otimes \omega(t) \cdot \Delta t - \beta \cdot \nabla f(q)$$

where $\omega(t)$ is the gyroscope-derived quaternion derivative, $\nabla f(q)$ is the gradient of the objective function derived from the accelerometer error, and β is a tuning parameter that controls convergence speed versus sensitivity to noise.

The theoretical underpinning of this algorithm assumes the noise and tilt are uncorrelated. However, empirically, this is not true.

Thus, when running a pure implementation of this algorithm, we found that the system was noisy. To help denoise this, we found that the accelerometer noise is related to the tilt of the system. For this reason, we added hyperparameter to beta that scales it according to the root-mean-square value of the accelerometer. If RMS of the accelerometer is higher than 1g, then the system is being affected by more than just acceleration due to gravity to the gyroscope should be prioritized. Thus, beta should be higher. Implementing this change produced much more stable IMU sensor values.

5.3.2 Control Systems

We implemented PID control loops for both the core unicycle wheel system and the lateral stabilization arm. The control objective for both subsystems was to maintain an upright orientation, defined by a roll and pitch angle of zero. Orientation feedback was obtained from the output of a Madgwick filter, which provided real-time estimates of Euler angles derived from the IMU data.

Two independent PID controllers were deployed:

- The **wheel PID controller** regulated pitch (forward-backward tilt) by adjusting the main drive motor.
- The **side arm PID controller** regulated roll (side-to-side tilt) using an actuated balancing arm.

Each controller used the Madgwick filter derived Euler angles as feedback and aimed to minimize the deviation from the upright setpoint. This basic control structure provided a stable starting point for balance but did not emulate how humans typically ride unicycles.

Given more time, we would explore trajectory-based nonlinear control using the Iterative Linear Quadratic Regulator (iLQR) algorithm. Unlike PID control, iLQR optimizes over a finite time horizon by simulating forward system dynamics and computing a control sequence that minimizes a quadratic cost function.

The process would involve:

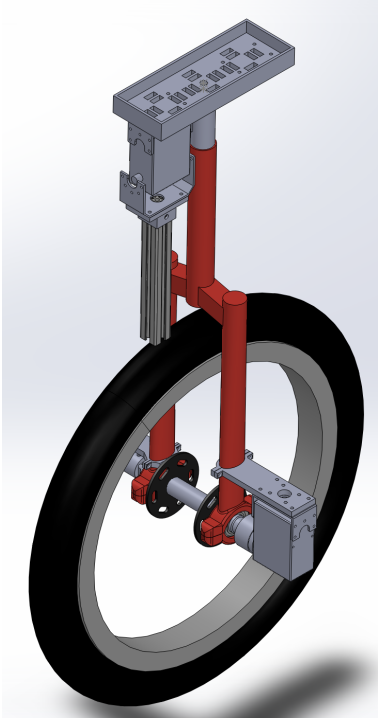

Figure 5.2: Complete assembly

- Defining a desired back-and-forth trajectory for the unicycle.
- Estimating linearized versions system dynamics by computing the impulse response plant
- Using iLQR to compute a feasible and optimal control policy that respects system constraints and dynamics

The unicycle is an extremely hard problem to think about in terms of PID control. It's nature follows more of an ergodic control problem, we we attempt to keep the time-averaged tilt close to zero, rather than the instantaneous tilt. Human riders maintain balance on a unicycle not by holding a static position, but by continuously adjusting their forward and backward motion. This generates the necessary gyroscopic effects and momentum-based corrections to stay upright. Thus, the iLQR algorithm poses a potential solution that may work. Our PID implementation assumes a static setpoint and attempts to minimize deviation from zero roll and pitch. As such, it lacks the ability to produce or follow dynamic trajectories that are critical for realistic unicycle balance.

5.4 Mechanical Design and 3D Printing

This project involved a significant amount of mechanical design work, as the motors were interfaced with an existing unicycle. For this reason, the main focus of the mechanical design and 3D printing for this project was the coupling and mounting between the electrical

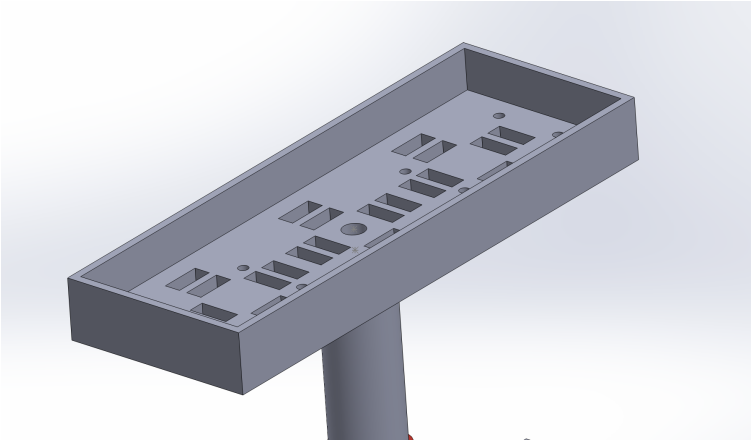

Figure 5.3: Seat and seat shaft

components and the existing unicycle. Another focus of the design was mounting our microcontroller and other inactive electrical components. This section presents our work on developing and manufacturing this design.

5.4.1 Seat

The seat that came with the unicycle, while well suited for a person to use, was not well designed to mount electronics and motors to. Therefore, as part of our mechanical design we created our own seat for the unicycle. The goals for this element are:

- Interface with the unicycle chassis
- House and protect the electronics
- Provide mounting for the counterweight

Seat Shaft

The seat shaft is the part which interfaced directly with the unicycle chassis. Therefore, its diameter matched that exactly of the original seat shaft. Additionally, to reduce weight in the upper part of the unicycle, the inside of the shaft was bored out. And finally, to mount the new seat a hole was drilled and tapped on the top end to allow for this interface. The part was manufactured out of aluminum on the lathe.

Seat

The seat we designed was optimized for customization. Since we were not sure where or how many motors we would mount to the seat we provided multiple options for where to mount the motor. Similarly, we designed many slots to mount the harness to the unicycle. The harness is essentially just a strap that we used to hold the unicycle during testing and tuning of the control system. The seat and shaft can be seen below in Figure 5.3.

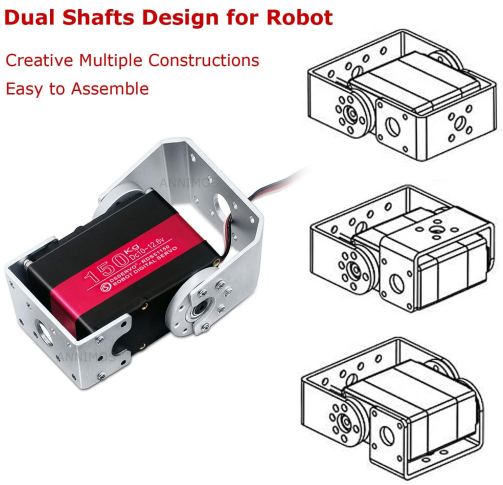

Figure 5.4: Servo motor used in design

5.4.2 Counterweight Motor

Motor Selection

After analyzing the motor requirements as discussed in the requirements and constraints section, we decided to purchase a motor that had as much torque output as possible. This was motivated by the fact that our estimations for the required torque of the motor were elemental, and that having a higher torque motor would allow for more weight to be added to the counterweight in the future. Additionally, we did not want to have additional motor controllers if possible so an easy to control motor would be best. Given these constraints we purchased and used a $150Kg * cm$ peak torque output motor as seen in 5.4. Other helpful features of this motor were the pre-designed brackets that it came with which allowed for easy mounting and coupling with the unicycle and counterweights respectively.

Mounting

The motor was mounted using screws and nuts through the brackets and into the 3D printed seat that we discussed previously. This design worked well, except for the fact that the vibrations caused my actuating the counterweight loosened the nuts on these screws. This was a problem throughout the entire design, and in future work we would need to correct this design flaw by using nuts with vibration protection. Another note is that since our design only used one counterweight motor, there was an inequal moment around the seat shaft and after time this would loosen the single screw holding the seat into the seat shaft. This design flaw would either be addressed by adding a second screw to constrain rotation about the seat shaft, or by adding another counterweight motor to balance out the moment. Of these two options, adding a second motor is probably better, as it would also balance the overall weight of the unicycle and make the design more balanced.

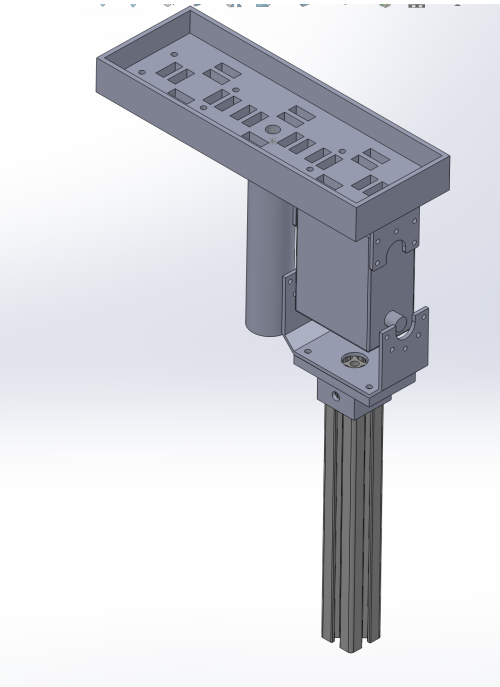

Figure 5.5: Counterweight motor coupling and mounting

Coupling

The coupling between the counterweight and the counterweight motor was difficult to design. This was in part due to the fact that the motor brackets were not designed to actuate a pendulum arm that is coupled perpendicular to the face of the bracket. Even so, the coupling between this motor and the counterweight successfully transfers the direction of coupling and extends the counterweight out six inches from the bracket of the motor. This design uses 80/20 aluminum extrusion and a 3D-printed adapter that couples this extrusion to the motor bracket. Using screw inserts into the 80/20 extrusion, we fixed the adapter to the extrusion and created a sturdy coupling between the counterweight and the motor. See the design in Figure 5.5.

Counterweight

The counterweight was designed to be a heavy object that is swung by the counterweight motor. We decided to manufacture the counterweight out of low carbon steel based on its availability and high density. In our final design, the counterweight was two 3 inch in diameter disks of 0.5 inches of thickness, weighing a little more than $2lbs$ total. This final weight was below our target, however, we realized that not only would such a large weight be difficult to manufacture and mount without more dense materials, but it also shifts the center of mass of the unicycle higher making the system more difficult to balance. See the counterweight in Figure 5.6 below.

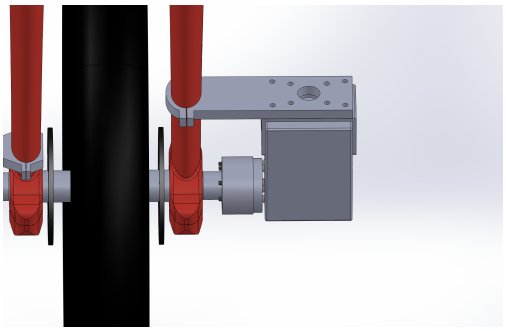

Figure 5.6: Wheel motor coupling and mounting

5.4.3 Wheel Motor

Motor Selection

After difficulties in sourcing a worm gear motor for this project, our team decided to use the same motor, as seen in 5.4 that we had chosen for counterweight actuation. We modified the motor to use a DC motor controller by taking out the servo controller and wiring it directly to the other controller. We wanted to build a more robust position and current controller for this motor, however, we were not able to integrate current or position sensing during this quarter. Part of the future work would be designing and tuning a more sophisticated motor controller.

Mounting

The wheel motor was mounted to the unicycle chassis via a 3D printed arm that wrapped around one of the arms of the unicycle fork. To keep it from slipping on the fork, we used a compression coupler fixtures with screws and nuts. Then, the motor was mounted to this arm using screws through the bracket and arm. See the mounting in Figure 5.6.

Coupling

Finally, the motor needed to be coupled to the shaft of the unicycle wheel. We achieved this coupling by modifying the wheel shaft. Usually, the linkages used for bike petals are press fit onto a drafted square shape on the end of the shaft and fixtured using a nut at the end of the shaft. We wanted to take advantage of this square shaft shape to transmit torque, however, we did not need the screw at the end of the shaft so we cut it off. Then, we designed the coupling to fit snugly onto the shaft, and to interface using screws to one of the pre-designed couplings that came with the motor. The design worked well, although it was not nearly as durable as we would have liked it to be. If we were to continue working on this project, we would need to re-manufacture this part out of a stronger material such as steel. See the final coupling below in Figure 5.6.

6 Final Product

6.1 Initial Goal vs. Final Product

Initially, we had hoped to build a unicycle capable of keeping itself upright with no outside assistance, in addition to being able to move in arbitrary directions based on control inputs from our website. In the end, we fell quite short of that ambitious goal. Around the midpoint of the quarter, we decided that based on our progress up to that point, it would be best to focus on simply getting the unicycle to stand up on its own without trying to add remote controls. Ultimately, we were not able to entirely achieve this desired outcome.

6.2 Performance and Limitations

The final prototype of the GhostRider contains mechanisms to stabilize itself in both the coronal and sagittal planes. Despite this, it is not yet able to remain standing without external support.

6.2.1 Wheel Motor

The wheel motor and its control system are the most functional part of the GhostRider prototype. With little to no external support, the unicycle is able to correct for disturbances in pitch without falling over. For example, if the unicycle begins to tip forward, the wheel will quickly spin until it is once again directly below the shaft. This control system is quite robust and can correct itself even for relatively large disturbances. It does, however, have some overshoot, which in some cases can lead to growing oscillations, leading the system to become unstable.

6.2.2 Counterweight Motor

While the wheel motor and its PID control system is somewhat stable, the servo arm counterweight and its PID controls are not able to stabilize the unicycle. While some of the malfunction can be attributed to PID controls not being sufficient to control a unicycle, there are also several issues with the PID control itself. For the servo arm counterweight, the control system is far too jittery, mainly due to noisy signals from the IMU. This causes it to jerk back and forth quickly, which damages the mechanical parts of the system. It is also only able to successfully counterbalance the unicycle for very small disturbances and is not robust.

6.2.3 Other Limitations

The other major limitation of our system is its general fragility. Because of the aforementioned jitter of the servo arm counterweight, as well as the haphazard mounting of the protoboard and voltage regulator to the unicycle shaft, the whole system is quite fragile and not able to sustain the damage from falling over repeatedly. Additionally, the high torque being driven by the servo arm counterweight is able to loosen nuts and screws in our mounting. The mechanical mountings and couplings would have to be improved in order to remain strong under the high tensions generated by the motor.

7 Challenges Encountered

7.1 Mechanical Challenges

Wheel Motor Ordering

One challenge that had lasting impacts was our wheel motor ordering issue. This challenge was the result of very specific requirements for our wheel motor that lead us to non-traditional motor sales websites. The first motor that we selected was a worm gear motor and was from a vendor that Northwestern could not pay. Then, we found a similar motor on Ali Express which we ordered and which was soon after removed from the website. After these issues with finding a worm gear motor, we decided to pivot and use a servo motor that we had used for the counterweight motor. This resulted in a several week delay getting our motor and delayed progress on the mechanical design further than was ideal for the project.

7.2 Electrical Challenges

ESP32 Flashing

Early on in the design process, we encountered repeated issues with uploading code to the ESP32. At one point we had several different ESP32 chips and none of them were able to be flashed properly. The underlying problem seemed to be that none of our devices could detect them as a device, and therefore there was no COM port through which to upload code. This issue persisted despite troubleshooting efforts including updating drivers. Ultimately, obtaining a new chip solved this problem.

Wiring

Because of the high currents running through some of the wires to drive the motor (maximum 5 amps), we needed lower gauge wires, specifically about 18 gauge. Unfortunately, the only thick wires that we had access to were frayed rather than solid core. This resulted in wires snapping or coming loose from the protoboard almost hourly when we were testing and tuning the system. As a result, we decided to simply tie together several thin wires and solder the ends, creating a makeshift low gauge wire.

Component Burnout

Throughout testing, several of our components were damaged. Because of the many moving parts of the unicycle, especially when it is on and running, it was difficult to pinpoint exactly what was causing the damage. The biggest loss we encountered was the motor driver malfunctioning just a few days before the project fair. One of the enable pins on the motor driver was being pulled low internally, indicating that something within the chip had malfunctioned. We ultimately had to get a new motor driver from Ilya and add it to our circuit.

7.3 Firmware Challenges

Developing the firmware was a pretty smooth process. The biggest issue we had was sampling the IMU real-time, however, that was a problem with the Arduino library we were using. To get around that, we went into the library code and deleted locking mechanisms. The control + sensor fusion algorithm development also posed a challenge. However, those challenges are detailed in section 5.3.

8 Planning and Organization

8.1 Communication Among Team Members

Communication was facilitated largely via a text group between the teammates named 'Silly Unicycle Boys'. All communication was performed via this group message, as well as all progress updates and requests for help, if it was not in person. Our team also had meetings with one another nearly every other week to discuss design updates and each members progress. These meetings were much more frequent during the start of the quarter, and as each team member found their specific niche within the group they became less necessary.

8.2 Splitting Tasks Among Team Members

8.2.1 Mechanical

Michael Jenz took on the mechanical work for this project, since he was the only mechanical engineer on the team. Much of this work was completed in the first half of the quarter, and involved interfacing the electrical components with the stock unicycle. More specifically, this was CAD modeling and designing the system, and then manufacturing all components used in the project. Later in the quarter, it turned into more of a support role maintaining the mechanical design and assisting in other areas where needed.

8.2.2 Electrical

Michael Kim mainly worked on the electrical components of the project, much of which took place in the second half of the quarter. Specifically, this involved putting together the circuitry for components like the motor, motor driver, servo motor, voltage regulators, microcontroller, and more. This included soldering components to the final protoboard, revising and improving the circuitry for better performance and safety, and debugging a myriad of issues while trying to integrate all of the different parts onto the unicycle itself. Debugging became the priority of all three group members toward the end of the quarter as something seemed to malfunction quite often during testing. It also included tuning PID controllers for both the motor and the servo arm to optimize the unicycle's stability.

8.2.3 Firmware

Milan Shah focused on the firmware. He first focused on getting the peripherals coded in RTOS as fast as possible so that there would be no major blockers for when the code was ready. Michael Kim helped with writing the PID loops and he helped debug a lot of the electronics and with the initial bread boarding and during the latter stages where we wired the whole system.

9 Market Research

9.1 Expert Interviews

We conducted several interviews with experts in the field of electronics and electrical engineering to learn about tools and techniques with which we were not familiar.

Alan Sahakian

First we spoke with Professor Alan Sahakian of the ECE department. With him, we discussed topics involving power electronics, including batteries, voltage regulators, and wires. Some important points included different types of batteries (for example, LiPo) and the advantages and disadvantages of each type. We discussed charging and discharging mechanisms for different types of batteries and safety protocols to ensure that the battery is not damaged during charging. Additionally, we reviewed the parts that we had selected to ensure that they were all compatible with each other (for example, the H-bridge driving enough current to run the motor).

Royce Morris

Next we spoke with Royce Morris, the electronics specialist and ECE lab director. Given his background, we focused our conversation on safety considerations, both for us working with the electronics and the components themselves. Our conversation touched on power supply separation for the motor (12V) and the logic (3.3V), proper wiring for high current components, and more on battery charging and discharging. We also discussed the best ways to test our device in the lab; because of the many components mounted to the unicycle, we felt that it would be undesirable to let it fall against the hard floor repeatedly. Some potential solutions he suggested included finding a soft mat or landing pad on which the unicycle could fall without sustaining damage, or using a harness to suspend the unicycle in the air during testing.

9.2 Non-Expert Interviews

In addition to our expert interviews, we also spoke with several of our peers for more insights and advice.

Kiran & Nikhil

First, we spoke with our classmates Kiran and Nikhil who were working on the large drone project. Their project had many similarities to ours; specifically, it involved several motors, PID controls, higher voltage power supplies, and a free-roaming system. One of their suggestions was to debug using a websocket rather than a USB cable. This would not only make debugging easier when the unicycle is up and running, but it would also protect our laptops against potential voltage surges like the one we encountered midway through the quarter. They also recommended keeping the logic voltage and the motor voltage as separate as possible to protect the circuitry.

Griffin & Caroline

Finally, we spoke to some of our fellow engineering students not in this class for insights about marketability. Because we decided early on that our prototype would not be human-ridable, we needed other ways to make it appealing to potential consumers. Since we initially planned to make the unicycle remotely controllable, our classmates suggested taking it a step further and potentially making it something that uses computer vision to follow around a user. A user could then use it to carry around a backpack or other loads.

10 Conclusion

10.1 What We Learned

Throughout the quarter we learned from many important experiences while working on this project.

10.1.1 Control Systems

One of the reasons we initially chose this project was to gain experience working with control systems. We learned a lot about the strengths and limitations of PID control, as well as things to consider when implementing PID control using digital microcontorllers. For example, we had to proceed with caution when adding integral and derivative control. With the integral control, we found that because of the colored noise and slight drift that the IMU presented, an integrator would often cause the set point to drift slightly off from vertical. For derivative control, the gain had to be kept very low to avoid extreme jerking, again a result of the noisy signals outputted by the IMU.

10.1.2 Circuits & Electronics

Most of the problems and debugging we faced when putting together the final prototype were related to our circuitry. For example, the big accident that happened midway through the quarter taught us the importance of keeping circuitry clean and free of stray wires that could potentially touch unwanted parts of the circuit. We also learned the importance of including safety features and properly rated components. This included adding a shutoff switch, properly thick wire, and bypass capacitors for stability.

10.2 What You Would Do Differently

As a group, we had hoped to get more in-depth experience in the areas of focus of our project, especially in designing control systems. However, because of the scope and size of our project, we ended up spending much more time putting together all the components and debugging the various issues that came up in the circuitry. If we had the chance to start over, we might choose an even smaller design, such as a 3D-printed model, so that we could use smaller motors, electronics, and circuitry. This would allow us to focus more on problems like filtering data from the sensors and implementing nonlinear control systems.

10.3 Possible Next Steps

If we were to continue working on the GhostRider, the most important next step we would take would be to switch from PID control to a more robust nonlinear control method, such as ILQR control. As previously described, PID control is not theoretically sufficient for this problem, as it tries to force the error to be zero at every timestep rather than tracking the trajectory of the unicycle through time.

Bibliography

[1] Guangchao Liang, Lingjian Ye, and Minfeng Zhu. Self-balance control of electric unicycle based on active disturbance rejection control. In *Proceedings of the 2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 237–242, Dali, China, May 2019.

[2] Sebastian O.H. Madgwick. An efficient orientation filter for inertial and inertial/magnetic sensor arrays. Technical report, University of Bristol, 2010. https://courses.cs.washington.edu/courses/cse466/14au/labs/l4/madgwick_internal_report.pdf.

A Appendix A: Additional Figures

Class Feedback

Learning Outcomes

Suggestions for Class Improvement

ESP32 Experience

Final Reflections

Additional Comments