Hardware design for an electro-mechanical bicycle simulator in an immersive virtual reality environment

Jaemin Powell
University of Iowa

Follow this and additional works at: https://ir.uiowa.edu/etd
Part of the Industrial Engineering Commons

HARDWARE DESIGN FOR AN ELECTRO-MECHANICAL BICYCLE SIMULATOR IN AN IMMERSIVE VIRTUAL REALITY ENVIRONMENT

by

Jaemin Powell

A thesis submitted in partial fulfillment
of the requirements for the Master of Science
degree in Industrial Engineering in the
Graduate College of
The University of Iowa

December 2017

Thesis Supervisor: Professor Geb Thomas

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

MASTER'S THESIS

This is to certify that the Master's thesis of

Jaemin Powell

has been approved by the Examining Committee for
the thesis requirement for the Master of Science degree
in Industrial Engineering at the December 2017 graduation.

Thesis Committee:

Geb Thomas, Thesis Supervisor

Joe Kearney

Stephen Baek

ACKNOWLEDGEMENTS

There are countless individuals who have supported me throughout my academic career as a graduate student. In the Spring of 2016 of my senior year at Central College, I met with Jane Dorman who knew of my motivations to become an engineer at the University of Iowa. She diligently recruited me since my oldest brother came to Iowa for their engineering program. Without her, I would not have been able to achieve my dream of becoming an engineer at Iowa. She introduced me to my advisor, Professor Geb Thomas. He allowed me to grow into the engineer that I am today by guiding me in the direction to succeed and pushing me to test my abilities. I am beyond grateful for his guidance and lessons he has taught me.

I want to thank everyone who contributed to my graduate school experience at Iowa. Geb was also a part of my defense committee along with Stephen Baek and Joe Kearney. I would like to thank them for agreeing to be a part of my committee as well as the specific roles each of them have taken on throughout my academic career. I want to recognize Jodie Plumert as well, who worked with Joe. I would not have been able to succeed in getting my degree without them collaborating with me throughout the process of constructing my bicycle simulator that fits their specific application. I would also like to acknowledge my lab team of Oliver Stroh, Mitch Fitzpatrick, Marcus Tatum, Jake Uribe, Summer Campbell and Clayton Mills for helping me through the frustrations and giving me unforgettable memories of my time here.

I'd like to express my eternal gratitude toward my friends and family. My friends have always gone out of their way to help me through tough times and create memories that will last forever. My family has always been my foundation that I've built my goals and successes upon because they endlessly support me through any obstacle. Any life goals that I’ve accomplished or life’s frustrations that I’ve been able to overcome are undoubtedly due to my family.

Lastly, I’d like to thank my oldest brother, Addison. My young engineering interests all began with him and me using Legos to create masterpieces. It took hours looking for one specific piece but that never slowed you down. We’d get out a blanket, lay it on the floor, then dumb the Legos into separate sections (based on color) across the blankets. It was a mess that stayed there for weeks at a time and we struggled every second to move in our rooms. Of course, there was a lot of pain throughout those weeks. Maneuvering through the Legos, trying to step in the gaps of free floor space. No matter how much caution we took, it was inevitable that we’d step on a Lego. The amount of pain such a small piece could inflict as our bare feet made contact is a physical phenomenon that I’ll never understand. But through the pain, you always accomplished your goal. Observing your creativity and determination to complete a task as small as that has always motivated me to get through every little Lego that I step on in life. Until we meet again, peace for AJP.

This material is based upon work supported by the National Science Foundation under Grant No. 1305131. None of my research would’ve been able to take place without the generosity of this grant.

ABSTRACT

Virtual reality provides a unique opportunity to safely investigate the interaction of children and traffic. Different methods have been implemented for bicycle simulators in virtual reality environments; however, these methods have failed to validate their theory behind their bicycle simulators. Additionally, their bicycle simulators only use electrical components for the different simulated resistances. These electrical components undoubtedly have an electronic delay that creates an unrealistic instantaneous inertia when the rider initially begins to pedal from a rested position. This is a very important factor when observing riders while they cross the street during heavy traffic. This study seeks to provide the Hank Bicycle Simulator with instantaneous inertia using electro-mechanical components as well as validate the methods implemented in the design.

The time it takes a rider to directly cross a road from a stopped position is almost entirely a function of the energy the rider puts into the pedals and the resistance to rotation provided from the pedals. At low velocities, nearly all of the pedal resistance is related to the effects of inertia, which is a function of the weight of the rider and the bicycle. To simulate the pedal resistance a flywheel was implemented into the system. The more the rider and bicycle weigh the higher rotational velocity the flywheel must produce. To produce a higher or lower rotational velocity an internally geared hub was used to increase or decrease gear ratio throughout the Hank Bicycle Simulator. At higher velocities, wind resistance increases and becomes a larger impediment to further acceleration than the effects of inertia. The wind resistance keeps riders from attaining past a certain velocity, called the terminal velocity. A motor was implemented to simulate the wind resistance effects.

Five experiments were conducted to provide validation in the proof of our method, however most of the emphasis is on the final two experiments. The first emphasized experiment measured the time delay of the electrical component signals traveling to the virtual environment. The time delay was about 20 ms to travel from the steering of the handlebars to the turning of the virtual environment. The final experiment validated the Hank Bicycle Simulator performance compared to the mathematical model of a bicycle in a real environment using a constant propulsive force. The Hank Bicycle Simulator had an initial acceleration better than 0.20% error and a terminal velocity better than 3.73% error for the gears that are used in the virtual reality environment

Overall, it is easy to see how quick the inertial response of the Hank Bicycle Simulator is given any rider compared to the delay of other simulators. The terminal velocity had a higher error; however, it is unlikely that a rider will get to terminal velocity because they are being observed crossing the street from a stopped position. The performance of the Hank Bicycle Simulator allows the rider to cross a street with about a 60 ms time difference between the simulator and a real-life rider pedaling at a constant propulsive force. The mechanical simulation of inertia in the Hank Bicycle Simulator provides an accurate and immediate reproduction of the expected inertia. We hope that the current study will encourage future researchers to report the details of their underlying models and their system performance, particularly those with ties to the physical parameters of the simulation they wish to reproduce, so that the community may move forward together, taking the best elements of each system.

PUBLIC ABSTRACT

Roughly 50,000 people are injured in bicycle collisions with motor vehicles each year. The Hank Bicycle Simulator provides a virtual environment to study and reduce this tragic loss by safely investigating the interaction of bicycle riders and traffic, particularly for bicyclists crossing streets. The bicycle simulator design focuses on the bicycle and rider inertia, the predominant dynamic element for riders moving from a stopped position. The Hank Bicycle Simulator's flywheel provides instantaneous inertial response while a servomotor provides simulated wind resistance to pedaling. This work describes the simulator design and a validation experiment that compares the simulator performance to theoretical predictions. The Hank Bicycle Simulator achieved initial acceleration with less than 0.20% error at realistic rider weights. The observed terminal velocity achieved less than 3.75%, with smaller errors for heavier riders. This allows the rider to cross a street with about a 60 ms time difference between the simulator and a real-life rider pedaling at a constant propulsive force. The Hank Bicycle Simulator was also validated through various physical experiments measuring the system inertia, the time delay of the electrical components, and the overall system performance. Such careful system validation for a mechanical feedback system is relatively rare in simulation research and is unique among previous reports of bicycle simulators.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Of the 50,000 people injured in bicycle collisions with motor vehicles each year, approximately 6,000 involve children younger than 14 years of age. Fifty such collisions each year result in child fatalities [National Center Statistics]. Virtual reality provides a unique opportunity to safely investigate the interaction between children and traffic. The Hank Bicycle Simulator has been instrumental in this type of investigation for over a decade and has provided unique insights into the perceptions and decision-making processes of both children and adults who encounter traffic on a bicycle [Grechkin et al., 2013; Plumert, Kearney and Cremer, 2004; Plumert, Kearney and Cremer, 2007]. The Hank Bicycle Simulator includes an instrumented bicycle fixed at the center of three projection screens and a projection floor [Babu, Grechkin, et al., 2009]. The movement of the bicycle is synchronized with graphics so that the rider is presented with a three-dimensional view consistent with the experiment of riding through a town or countryside. A motor connected through the bicycle's drivetrain provides resistance to pedaling, consistent with the resistance perceived during normal bicycling.

Several researchers have implemented similar devices in numerous ways. Some are most interested in the influence of virtual reality on exercise [Huang et al., 2008; Mestre, Dagonneau and Mercier, 2011]. Leblanc and Sicard (2010) emphasize the mathematical details of the control system. Sari et al. (2009) test the use of a virtual reality bicycling experience as a means for promoting actual bicycle use on the University of Indonesia campus. Our interest differs in that it primarily concerns the construction of a working bicycle with sufficient fidelity to reconstruct the force-displacement relationship in real time in order to serve as a platform for psychological experiments involving decision-making.

Table 1 compares our project with the five other bicycle simulators that feature controlled pedal resistance. The KAIST simulator [Kwon et al., 2001; Kwon et al., 2002] was designed to completely immerse a rider in a bicycle race with other riders on similar bicycles. It features a bicycle mounted on a six-degree-of-freedom Stewart platform, with motions emphasizing the vertical movement (heave), side-to-side (roll), and front-to-rear (pitch) movements. Pedal resistance is provided through a magneto-rheological fluid brake. An AC servomotor simulates acceleration due to gravity as the rider goes down a slope. The rider wears a head-mounted display and graphics are synchronized with a second bicycle over a network.

The Hong Kong simulator [Tang et al., 2007] was designed to promote exercise through cycling and allowed riders to exercise in a confined space. The bicycle rests on a platform with spring supports on the bottom, permitting riders to slightly tilt as they pedal and turn. A damper is attached to the rear wheel to provide resistance from wind and road friction. An AC servomotor produces decelerating or accelerating torque based on the slope of the rider's environment.

The Morics simulator [Miyanoue et al., 2015] was designed for traffic safety education and safety analysis of a wide variety of traffic situations. The rear wheel stand enables the rider to lean slightly to either side but does not provide a natural lean that a rider can feel while turning. An actuator simulates the inertial force of the rider and the load generator simulates idling and other weather conditions of the environment.

The Shanghai simulator [He et al., 2005] features a complex, two-wheeled, bicycle dynamic model that uses Lagrange's equation of motion and the Runge-Kutta method. The model has two sub-models: the stability sub-model and the vibration sub-model. The stability sub-model solves equations for the rider's current movement of the bicycle. The vibration sub-model solves equations for the vibrations that result from the bicycle's current surface. The researchers validated their mathematical sub-models by comparing data of the calculated steering and tilt angle to the measured values. They put forward methods to evaluate the pedal torque, however they never provided any information on the resistances the specific rider must overcome to move the bicycle in the forward direction. Consequentially, the resistances applied to the rider in their bicycle simulator is not validated.

The Japan simulator [Kikuchi et al., 2012] was designed for people with cognitive and physical impairments. The researchers aimed to provide an opportunity for safe exercise without the need to worry about other riders or traffic. Instead of using a servomotor to simulate the resistance of the bicycle, a magneto-rheological fluid brake was custom-designed by their team. The magneto-rheological brake fluid brake was equipped with an enhanced braking torque by increasing the magnetic field's distribution along the area of the cylindrical surface. That allowed all the movements of the bicycle to be restricted using this one movement system, rather than using two different systems like most simulators. They collected data of the pedaling torques while the rider traveled virtually through different slopes.

Table 1 compares several main features of each of the reference systems. The first four features detail the physical and virtual surroundings of the bicycle simulators. *Screen* refers to how the virtual environment is displayed and whether the presentation is 2D or 3D, the number of surrounding screens, or whether the implementation used a head-mounted display. The horizontal and vertical field of view row shows the restrictions on the rider's field of vision. Since many simulators use head-mounted displays, riders see only a portion of their surroundings compared to an unrestricted view. Each of the virtual environments included sound, although only two provided 3D sound.

Table 1. Selected bicycle simulator implementations.

Bike Simulators						
	Thomas, Powell, et al.	Kwon et al.	Tang et al.	Miyanoue et al.	He et al.	Kikuchi et al.
	Hank Bicycle Simulator	KAIST	Hong Kong	Morics	Shanghai	Japan
Screen	4xStereoscopic Display	Mono Display or HMD	Stereoscopic Display	Mono Display or HMD	HMD	Stereoscopic Display or HMD
Horizontal Field of View	270°	45° or less	45° or less	102°	45° or less	45° or less
Vertical Field of View	180°	45° or less (no ground)	45° or less	64°	45° or less (no ground)	45° or less
Sound	✓	✓ (3D)	✓	✓	✓(3D)	✓
Air Resistance	✓	✓	✓	✗	✓	✓
Leaning	✗	✓(6-dof/4-dof)	✓(Springs)	✓(Not full range)	✓(6-dof)	✗
Simulated Inertia	✓	✓	✓	✓	✓	✓
Steering Angle	175°	Max	Max	Max	Max	Max
Front Wheel Velocity	✓	✗(No front wheel)	✗	✗	✗(No front wheel)	✗
Movement System Type	Flywheel & AC Servomotor	MR-Brake & AC Servomotor	Damper & AC Servomotor	Load Generator & Actuator	3 Torque Estimators	MRB
Instantaneous Inertia	✓	Electronic Delay	Electronic Delay	Electronic Delay	Electronic Delay	Electronic Delay

The next five features in *Table 1* compare the physical properties of the instrumented bicycles. All but one simulator implemented the effect of air resistance on bicycle pedal acceleration. Four simulators allowed the bicycle to tilt side to side, though one performs this mechanism with passive springs. All implementations simulate the rider's inertia and the full range of steering angles. Two simulators removed the front wheel, presumably because they used head-mounted displays and the front wheel would not be visible. In the other three simulators, the front wheel did not spin.

The final two features in *Table 1* refer to the mechanism and model of the pedal resistance system. All five of the bicycle systems used motors or brakes to simulate the inertia, air resistance, rolling resistance, and propulsive forces. AC servomotors or actuators were a typical choice for accelerating and decelerating on slopes. The pedaling resistance caused by the remaining factors of air resistance, rolling resistance and bump resistance were supplied by magneto-rheological fluid brakes, dampers, or load generators. Electrical motors and brakes i.e. AC servomotors, actuators, magneto-rheological fluid brakes, dampers and load generators, are widely used to simulate the forces a rider experiences, but this comes at a cost as such devices produce an electronic delay that the rider encounters upon first pedaling. The magneto-rheological fluid brake increases the viscosity fluid when a magnetic field is present, which takes about a millisecond of time to perform [Kikuchi et al., 2012]. Additionally, the majority of the delay is from other electronics used for the bicycle dynamics. The dynamic signals from the components are sent to the system controller, the system controller receives and interprets the signals, then the system controller sends the signals back to the simulator for the components to perform their updated functions. After that process, the overall delay increases to tens of milliseconds. Although this may seem inconsequential, this delay creates an instance of free pedaling as the rider first tries to move forward, in turn causing them to accelerate at a faster rate on the bicycle once the delay has passed. This requires an unrealistic pedal force from the rider as he or she begins to pedal from a stopped position and the magnitude of the instantaneous change requires the system to absorb more instantaneous force. The Hank Bicycle Simulator is particularly interested in the starting acceleration of the bicycle because the riders are crossing the street from a stopped position. Thus, any delay of the inertial system will create differing responses of the virtual world compared to the real world.

The simulators in *Table 1* all have their own unique theoretical background implemented into the simulator. They promoted safe exercise [Tang et al., 2007; Kikuchi et al., 2012], derived mathematical models [He et al., 2005], implemented Stewart platforms [Kwon et al., 2001; Kwon et al., 2002] and analyzed safety [Miyanoue et al., 2015]. Each of the five reference systems provide programmable, dynamic pedal resistance while delivering the steering angle and velocity to the graphical simulator system, but each one failed to compare their theoretical results with the real world.

Because the psychological experiments conducted with the Hank Bicycle Simulator focus on street crossing scenarios from a stopped bicycle, the bicycle design is particularly focused on the starting simulated rider and bicycle inertia, the dominant component of the initial movement dynamics. In the past, we have been satisfied with pedal feedback that was realistic enough to escape the attention of the riders. In this work, we seek to quantify the performance of the Hank Bicycle Simulator.

Also, because of the role of bicycle in formal psychological experiments, we seek to publicly explain the details of the simulator's behavior and describe how the performance was validated in order that other researchers can consider how these details might affect the results of the psychological tests and ultimately improve on our results. To our knowledge, no scientific team has attempted to validate or analyze the performance of their bicycle simulator against their theoretical model, nor have they postulated how the details of their implementation may affect the scientific conclusions reached with their simulator.

In this work, we first describe the physical model we use for the bicycle. Then we describe the simulator's construction and finally multiple tests to validate the performance of the system, emphasizing the last two tests that measure the time delay of the bicycle's response and finally the response of the simulator to a constant torque input.

CHAPTER 2

BACKGROUND

2.1 The Physics of the Bicycle

The time it takes a rider to directly cross a street from a stopped position is almost entirely a function of the energy the rider puts into the pedals and the resistance to rotation provided from the pedals. At low velocities, nearly all of the pedal resistance is related to the effects of inertia, which is a function of the weight of the rider and the bicycle. At higher velocities, wind resistance increases and becomes a larger impediment to further acceleration than the effects of inertia. Since the rider's expectation of how long it will take to cross a street in the virtual world is presumably based on their past experience in the real world, it is important to create a system that accurately reproduces the effects of inertia and wind resistance, so that the relationship between the rider's exerted energy and the time to cross the street matches their past experiences in the real world.

A bicycle transfers torque from the pedal rotation into a propulsive force, F_P in Newtons (N), acting between the rear wheel and the ground. This propulsive force accelerates the rider and bike's inertia when it overcomes the drag, F_D, the slope resistance, F_S, the rolling resistance, F_R, the bump resistance, F_B, and the mechanical inefficiencies, F_M [Wilson et al., 2004].

The acceleration force, F_{Acc}, of the total bicycle system can then be described as:

$$F_P - (F_D + F_S + F_R + F_B + F_M) = F_{Acc} = Ma \qquad (2.1)$$

where M is the mass of the bicycle and rider in kilograms (kg), and a, is the acceleration of the bicycle in meters per second squared (m/s^2).

In this study, the potential effects of slope and bump resistance are ignored because of the assumption that the rider will have a flat smooth surface as he or she crosses the street [Wilson and Papadopoulos, 2004]. F_R is a fraction of a percent of the total system weight due to the rolling resistance coefficient, which can be anywhere from 0.002% to 0.008% [Wilson et al., 2004]. Since F_R is such a small value of the rest of the system, it is considered to be negligible in our calculations.

The mechanical efficiency, F_M, is the ratio of the output power compared to the input power. There can be as low as a 1% loss of energy to as high as a 20% loss of energy in the bicycle system [Kyle & Berto 2001]. This force will be considered in the calculations, though the rider will naturally compensate to the inefficiencies similar to the real-life experiences of pedaling a bicycle. The efficiency of the simulator should be similar to that of a regular bicycle.

Given these parameters, *Equation (2.1)* is simplified to:

$$F_P - (F_D + F_M) = F_{Acc} = Ma \tag{2.2}$$

The inertial force depends mostly on the mass of the rider and the bicycle, M, although the rotational inertia of the rotating components on the bike are also important factors. Since the theater of our simulation presumes that the rider is using an unfamiliar bicycle, it seems reasonable to assume that so long as the bicycle parameters are within the range of similarly sized commercial bicycles, the validity of this is dominated by the mass of the rider and bicycle.

The force of wind drag, F_D, is a function of the square of the velocity:

$$F_D = D\dot{x}^2 \tag{2.3}$$

where D is the air drag constant in kg/m, and $\dot{x}$ is the velocity in m/s, in terms of position. The drag constant is described as follows:

$$D = \frac{1}{2}\rho C_d \mathrm{A} \tag{2.4}$$

where ρ is the air density in kg/m^3, C_d is the drag coefficient (unitless), and A is the area of the rider in m^2. Our simulator treats each of these variables as constants, adjusting the area of the rider with the size of the bicycle. At high speeds, lift will be applied to the bicycle and rider from the head wind. The will effectively produce a lighter overall mass for the bicycle and rider, consequently, making it easier to accelerate. Our simulator will be *Equation (2.3)* and *Equation (2.4)* will be described in greater detail later in the section "Simulating the Air Drag".

For simplicity, we will combine the terms, F_P and F_M into the propulsive force variable, P. Accounting for drag and inertia results in the following second order, partial differential equation:

$$P = D\dot{x}^2 + M\ddot{x} \tag{2.5}$$

where $\ddot{x}$ is the acceleration, in terms of position.

Solving *Equation (2.5)* for $x(t)$ gives the position of the rider, x, at time t:

$$x(t) = C_2 + (\frac{M}{D} ln\left(\cosh\left(\frac{\sqrt{DP}(t+MC_1)}{M}\right)\right)) \tag{2.6}$$

where C_1 and C_2 are constants. Assuming the bicycle initially starts from a resting point, we can apply those boundary conditions of $x(0) = 0$ and $\dot{x}(0) = 0$. From those boundary conditions, we produce the values of the constants to be $C_1 = C_2 = 0$. This produces the fundamental equation of our bicycle traveling in a straight line:

$$x(t) = \frac{M}{D} ln(\cosh\left(\frac{\sqrt{DP}}{M}t\right)) \tag{2.7}$$

Every term in this equation is considered to be constant, even P. Though the propulsive force changes in a real scenario, it is considered to be constant because we want consistent forces to drive the system and work against the air drag. This is intuitive because the larger force that is applied to the bicycle the greater speeds the bicycle will be able to attain. Taking the derivative of this equation will then describe the velocity of the bicycle with t, as follows:

$$\dot{x}(t) = \sqrt{\frac{P}{D}}\tanh\left(\frac{\sqrt{DP}}{M}t\right) \tag{2.8}$$

If we assume that P is constant, that is, the rider applies a constant torque on the pedals, *Equation (2.8)* describes an asymptotic function that converges to $\sqrt{\frac{P}{D}}$, which is the terminal velocity of the bicycle. Near the origin, at $t = 0$, the slope of the line would then be described below by taking the derivative of $\dot{x}(t)$ as follows:

$$\sqrt{\frac{P}{D}}\frac{\sqrt{DP}}{M} = \frac{P}{M} \tag{2.9}$$

Thus, under the special condition of a constant torque on the pedals, we can predict the beginning acceleration and the terminal velocity of the bicycle. Using *Equation (2.8)* we can measure and display the theoretical speed at a given time with a velocity versus time graph (*Figure 1)*, and compare the theoretical results with the experimental results that are measured while driving the bicycle with a unique propulsive force.

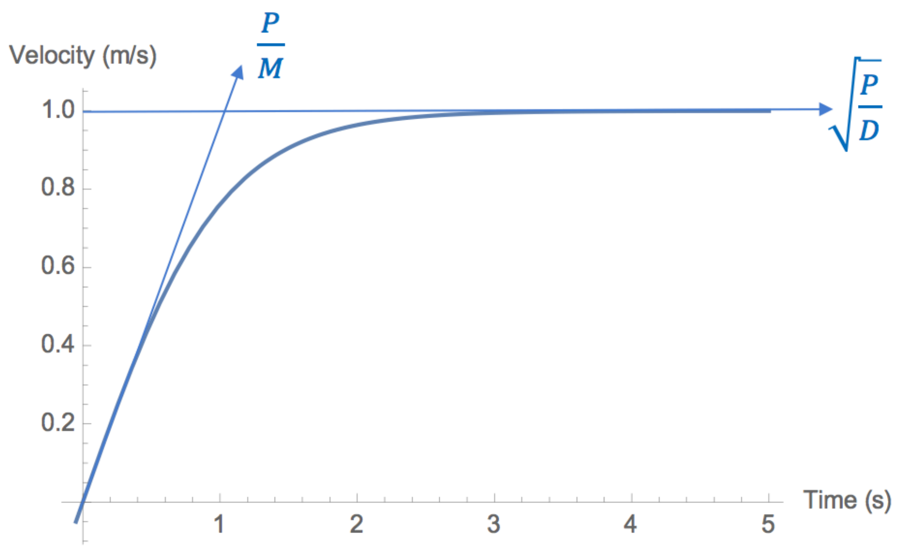


Figure 1: Graphical representation of Equation (2.8) with an asymptote at $\sqrt{\frac{P}{D}}$ *and a starting acceleration at* $\frac{P}{M}$.

2.2 Past Iterations

In the two earlier version of the Hank Bicycle Simulator, the first of which was used for nearly a decade, the rear wheel of the bicycle interacted with a 6.5" diameter, motor-driven aluminum roller. The roller inertia was not precisely integrated into the rest of the program, which was ultimately tuned by hand to provide a subjective sense of realism. Another shortcoming of both systems was that the roller tended to wear away the tire, producing a fine, sooty residue that soiled the projector screens.

Both earlier iterations used a large servomotor and controller in velocity control mode. A computer received the current torque used by the motor to maintain the current velocity, compared it to a model of what torque should be required at the given velocity and then sent a new velocity target to the motor controller. An important challenge in the system was the need to quickly respond to sudden torques, such as when the rider first applied force to the pedals from a stopped position. The noisy response torque signal from the motor necessitated multiple readings and averaging, which caused a noticeable response delay. It also required a substantial motor to resist the initial input torque when a rider aggressively stands on the pedal to begin a ride, particularly when demoing the system to athletic college students eager to test the system limits. The mechanical flywheel provides instantaneous and accurate inertia. The accuracy of the inertia is an important factor in the Hank Bicycle Simulator, so validation of the inertia for a specific rider is a key component of this study. We plan to prove our theory of the implementation of the bicycle through experimentation of the Hank Bicycle Simulator.

CHAPTER 3

GENERAL METHODS

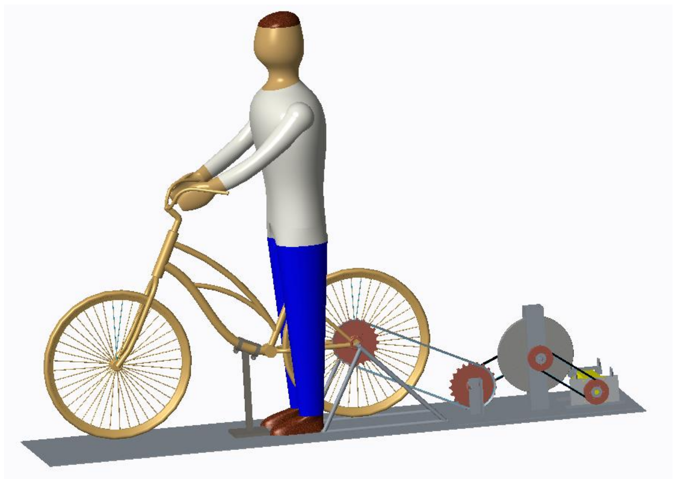


Figure 2: The initial virtual design of the hardware in the Hank Bicycle Simulator.

3.1 Selection of the Bicycles

Because the system was to be used by both children and adults, we expected that it should be capable of accommodating several sizes of bicycles. The design parameters for the bicycles were that they should:

1. fit rider's weight up to 220 lb.;
2. fit riders from age 6 and up;
3. fit rider's height up to 6 ft.;
4. have comfortable seats;
5. have cruiser style handlebars;
6. have a chain guard;

7. have no gears; and,
8. be a step through bicycle.

Typically, children's bicycle sizes are listed by using the diameter of the rear wheel, i.e. they are usually 12 inches (in.), 16 in., 20 in., or 24 in. The 16-in. bicycles are made to fit children between 38 in. and 47 in. tall. Generally speaking, this fits riders between 3 and 8 years of age (*Figure 3)*. The 20-in. bicycles are made to fit children between 44 in. and 52 in. tall, which generally fits riders between 4 and 12 years of age (*Figure 3*). Based on the CDC chart above, the 20-in. bicycle would fit 84% of 6-year olds, while the 16-in. bicycle would fit only the bottom 30% of children around 6 years old. Thus, the 20-in. bicycle was chosen as the smallest bicycle for the Hank Bicycle Simulator.

The 24-in. bicycle was the next option chosen because its lowest height limitation started near the highest height limitation on the small (20-in.) bicycle. This bicycle sizes for children from 54 in. to 61 in. The 52 in. to 54 in. riders that are not included between the small bicycle and medium (24-in.) bicycle can be added to the small bicycle, simply because it is easier to ride a bicycle that is too small than it is to ride one that is too large.

Based on *Figure 3,* there were still some heights that needed to be accounted for that the small and medium bicycles could not accommodate. For this reason, we included one more bicycle in our Hank Bicycle Simulator. The large (26-in.) bicycle was chosen because it was the next size of bicycle after the medium bicycle and it covered heights from 62 in. to 68 in. Using these three bicycle sizes, we can fit riders with heights of 44 in. to 68 in., covering the ages of 4 years to the 25th percentile of 20-year old males, and the 90th to 95th percentile of 20-year old females.

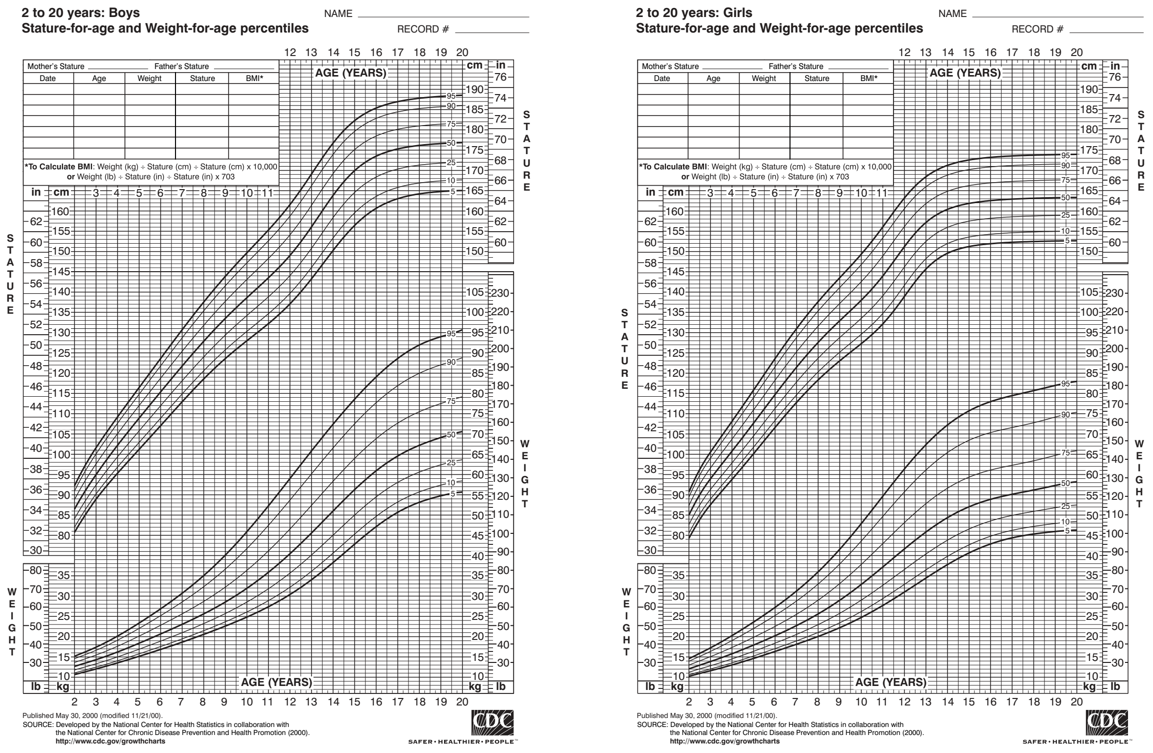


Figure 3: CDC Charts showing the stature-for-age and weight-for-age percentiles for children from the ages 2-20 years old based on their sex.

3.2 Designing the Flywheel

The goal of the design is to accurately simulate a road crossing, or to accurately reproduce the rider's position in time, given the rider's mass and torque input. The flywheel simulates the inertia of the bicycle and the rider, and the motor simulates resistance from the wind drag. Since the drag increases with the square of the relative velocity, in this application it has a relatively small effect; the resistance of the rider pedaling is mostly related to overcoming the inertia of the rider and bicycle.

Neglecting the rotational inertia of the bicycle wheels, the linear kinetic energy, in joules (J), of a bicycle and the rider is:

$$KE_{rider} = \frac{1}{2}M\vartheta^2 \tag{3.1}$$

where M is the combined mass of the bicycle and the rider, and ϑ is the velocity of the bicycle. The rotational kinetic energy of the inertial system is:

$$KE_{flywheel} = \frac{1}{2}I\omega^2 \tag{3.2}$$

where ω is the angular velocity of the flywheel in radians per second (rad/s), and I is the moment of inertia in kg(m^2). Equating *Equation (3.1)* and *Equation (3.2)* gives:

$$\frac{1}{2}I\omega^2 = \frac{1}{2}M\vartheta^2 \tag{3.3}$$

Solving for M produces:

$$M = \frac{I\omega^2}{\vartheta^2} \tag{3.4}$$

Thus, if the gear system can be designed to control the ratio between ω and ϑ, then using a certain inertia from a flywheel, a range of rider and bicycle masses, M, can be simulated. The only unknown variable remaining in *Equation (3.4)* is I. I is the moment of inertia of the flywheel and the rear wheel of the bicycle.

The inertia of a rotating cylinder with mass, m_F expressed in kg, and radius, r_F in *m,* is:

$$I_F = \frac{1}{2}m_F {r_F}^2 \tag{3.5}$$

The inertia of the rear wheel of the bicycle was also included in the calculations because the rider is driving the rear wheel as well as the flywheel of the system. The inertia of the rear wheel of a bicycle with mass, m_R, and radius, R, is:

$$I_R = m_R R^2 \tag{3.6}$$

I_F and I_R are additive, however, I_R mechanically transfers to the flywheel as a squared inverse function of the gear ratio factor, G. Thus, the total inertia is:

$$I_{total} = I_F + \frac{I_R}{G^2} = \frac{1}{2} m_F r^2 + \frac{m_R R^2}{G^2} \tag{3.7}$$

The bike velocity and the flywheel rotation rate are related by G and the inverse of R, producing the *Equation (3.8)* below.

$$\omega = G\vartheta / R \tag{3.8}$$

The bicycle pedals are connected to the chain ring, which is connected by a chain to the bicycle's cassette on the rear wheel. The gear ratio between the chain ring and the cassette typically provides the rider with a mechanical advantage. In this case, the bicycle cassette has been removed. A new chain is connected from the back wheel to the input of an internally geared hub. The output of the hub is connected to the flywheel, which is connected to the motor.

The gear ratio of the whole system, from the pedals to the motor driving the flywheel, is the product of the individual gear ratios across the system, G. Thus, for a cylindrical flywheel, a rider's mass can be simulated with an appropriate choice of flywheel mass, radius, and gear ratio. Combining *Equation (3.4)*, *Equation (3.7)* and *Equation (3.8)* produces:

$$M = \frac{G^2 m_F r^2}{2R^2} + m_R \tag{3.9}$$

The best dimension of the flywheel was determined by the range of gear ratios available in the drivetrain, particularly in the gear hub. An internal gear hub was selected because it provided the freewheel capability necessary to protect riders from any error involving accidentally overdriving the motor in reverse, which could potentially spin the pedals backwards into a rider's shin. The other advantage of the internal gear hub is that it can be shifted when it is not in motion, which simplifies the experimental preparation between participants. We selected an 11-speed, internally geared hub that could shift between gear ratios from 0.527 to 2.153. When coupled with the other fixed gears in the drivetrain, the overall gear ratio ranged from 2.306 to 9.419.

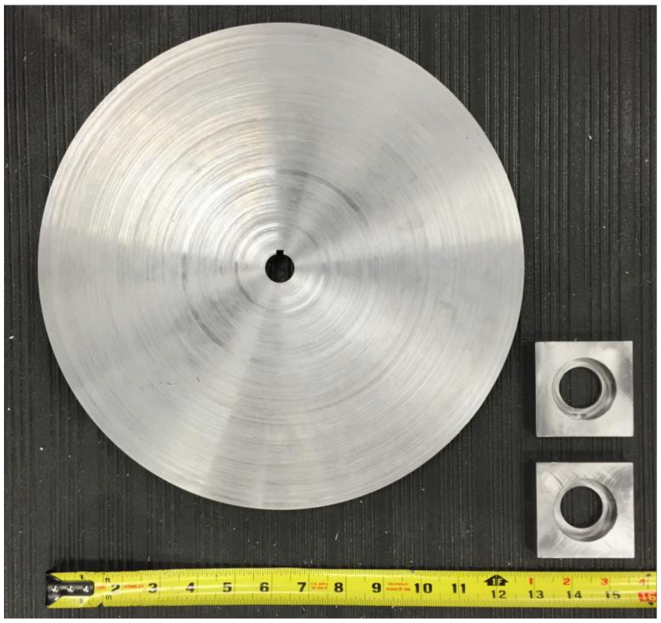


Figure 4: Flywheel with the mounting brackets for the driveshaft.

The flywheel was then designed to provide the correct range of inertias for the expected weight of riders on each of the three bicycle sizes. This resulted in a steel flywheel with a thickness of 0.709 in. and radius of 6.339 in. shown in *Figure 4* above. We cut the flywheel from a ¾" steel plate and turned it. The final weight was approximately 26 lbs. *Figure 5* displays the final design of the inertial system. With this design, the Hank Bicycle Simulator can accommodate riders as small as 10 lbs. on the small bicycle, using gear 1 of the internally geared hub, and riders as large as 253 lbs. on the large bicycle, using gear 11 on the internally geared hub. *Table 2* shows the size of the bicycles as well as the weights of the rider with the corresponding gear ratio. There are a range of weights for each different bicycle with the small bicycle having 4 different simulated weights and the medium and large bicycles having 5 different simulated weights. There are other weights that can be simulated using all 11 gears, however, they are considered to be out of the range of a typical rider's weight given the bicycle size.

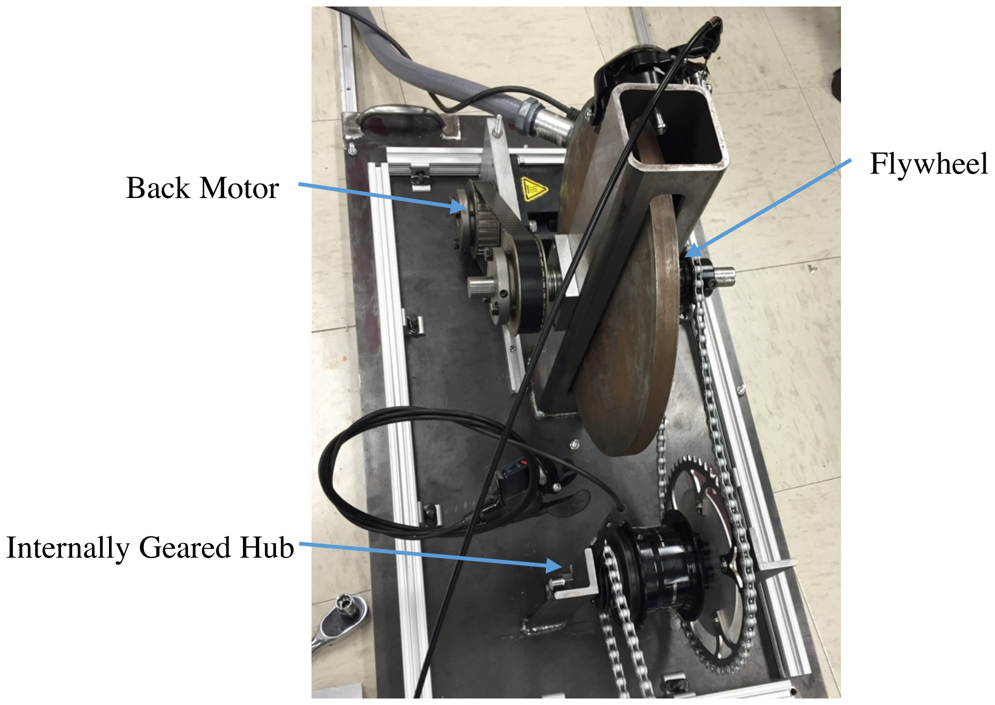


Figure 5: The flywheel and the internally geared hub connected by chain in the inertial system before the painting of the steel components.

Table 2: Chart of the bicycle size and the weight corresponding to the gear ratio.

Bicycle Size/Gears Chart

Bike Size:	Weight(lbs):	Range (lbs):	Gear #:
Small(20")	40	<51	3
	60	51 to 71	4
	85	71 to 98	5
	115	>98	6
Medium(24")	50	<58	5
	70	58 to 82	6
	95	82 to 112	7
	130	112 to 151	8
	175	>151	9
Large(26")	75	<88	7
	100	88 to 121	8
	140	121 to 164	9
	190	164 to 220	10
	255	>220	11

Gear	Ratio
1	2.306
2	2.979
3	3.369
4	3.841
5	4.353
6	4.961
7	5.653
8	6.396
9	7.293
10	8.260
11	9.419

3.3 Simulating the Air Drag

If a cyclist provides a constant torque to the pedals while riding on a flat, smooth surface, the bicycle will accelerate until the air drag increases to the point that all of the cyclist's work is spent overcoming the air drag. A motor (AKM53K-BKCNC-00, Kollmorgen, Radford, VA) and motor controller (AKD-P01206-NBAN-0000, Kollmorgen, Radford, VA) resist the flywheel rotation to simulate wind resistance as described in *Equation 2.3*.

The velocity, ϑ, of the bicycle relative to the ground in *Equation (3.10)* was expressed in terms of the derivative of a position given a certain time (*Equation (2.3)*); however, using the motor output, we can also find the velocity of the rider using the gear ratio from the motor to the rear wheel. The drag is based on the following assumptions: (a) air density, ρ, is 1.184 kg/m^3, (b) the area facing the wind, A, is 0.4 for 20-inch bicycle, 0.5 for the 24-inch bicycle, and 0.6 for the 26-inch bicycle, and (c) the drag coefficient, C_D, is 1.1 for each bicycle [Gross, A., et al.].

A microcontroller (Uno, Arduino, Ivrea, Italy) receives the velocity signal from the motor controller, interprets the velocity, and calculates the corresponding air drag, F_d, according to:

$$F_d = \frac{1}{2}\rho C_D A \vartheta^2 \tag{3.10}$$

The microcontroller then sends a new torque command to the motor as an analog voltage input. The motor's response is then transferred through the gear train to the rider's feet to produce the sensation of increasing air drag as the rider increases pedaling velocity.

3.4 Manufacturing the Inertial System Box

The flywheel, motor, and internally geared hub are housed in a case. The case sides are made of ¼ in. ABS plastic supported by an extruded aluminum frame. The case helps to dampen the gear train noise from the system and protect experiment participants from the moving parts.

The motor is connected to the flywheel by a belt. The flywheel was supported using a ¼ inch-thick steel square tube welded in the center of the base plate. Two slots milled in parallel sides of the square tube contain the flywheel. Ball bearings and mounts support the driveshaft and allow low-resistance rotation. Spacers on the driveshaft hold the flywheel in the center.

The motor is mounted to the plate of the inertial system box through two T-slot tracks screwed to the plate to facilitate belt-tension adjustment. Custom clamps on the top of the motor and the driveshaft housing secure the motor to the T-slot with long hexagon-headed screws. The

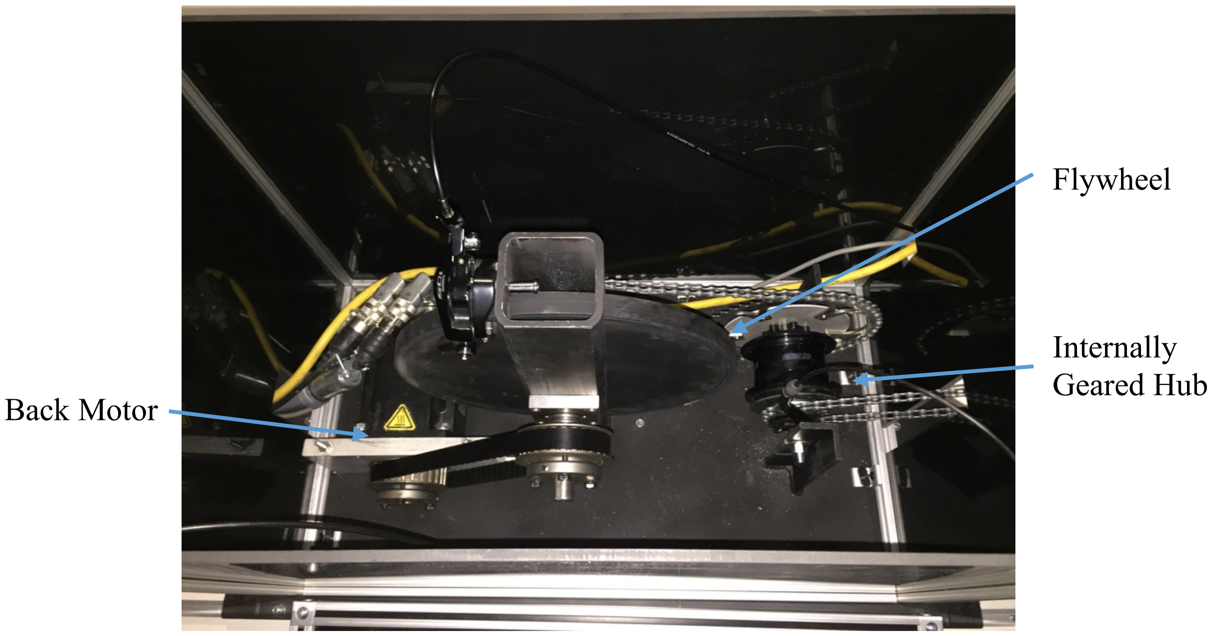


Figure 6: The complete inertial system, consisting of a motor for air drag, a flywheel for kinetic energy and an internally geared hub for different weights of riders.

motor feedback and motor power cables exit the back of the inertial system box and are connected to the motor controller in the electronics box.

A chain and two sprockets connect the flywheel and the internal gear hub. The internal gear hub was designed to use only one sprocket, which was located on the drive side, going towards the bicycle in *Figure 6*. Thus, we designed and built an adapter to connect a 49t sprocket to the side of the internal gear hub intended for a disc braking system. The internally geared hub is held above the bottom plate by two welded L-brackets. Slots in the brackets facilitate chain-tension adjustment.

The internal gear hub control exits the enclosure at the top so users can easily shift the gears, changing the simulated weight of the rider. A caliper brake at the top of the flywheel is connected with a brake cable to the bicycle handle bars, enabling the rider to slow the flywheel and reduce their speed. Slowing down the flywheel slows down the rear bicycle wheel.

3.5 Combining the Mechanical Components with the Bicycles

The inertial system box is mounted on three parallel T-slot tracks screwed into a 24 x 96 in., 1 in. thick white melamine board. Four welded handles near the corners of the bottom plate of the inertial system box facilitate shifting the plate forward and backward to adjust the tension of the chain connecting the internal gear hub to the bicycle's rear wheel.

The bicycle is mounted in front of the inertial system box on the melamine board so that the rear and front wheels are supported above the melamine surface. The back wheel is supported with a commercial stand (Rock-the-Bike, Oakland, CA) with pipe routing clamps. A flip-flop hub (Surly Track Cog 1/8" X 20t, Geoff's Bike & Ski, Iowa City, IA) on the rear wheel of each bicycle (*Figure 7)* allows the rear bicycle wheel to spin faster than the pedals, which often happen when the rider coasts.

After failing to find a commercially available stand that would allow the front wheel to turn freely, we designed and built the one show in *Figure 8*. A steel disc was welded to a long, round, hollow tube, with a thickness about equal to a bicycle frame. The top of the tube was cut at an angle to match each bicycle's downtube. The bicycle downtube rests on a long section from a round tube that was welded into the notches at the top of the vertical square tube. A screw passes through the tube section into the bicycle downtube, using a rivet nut we installed in the downtube. The disk at the bottom of the stand is screwed into the melamine board.

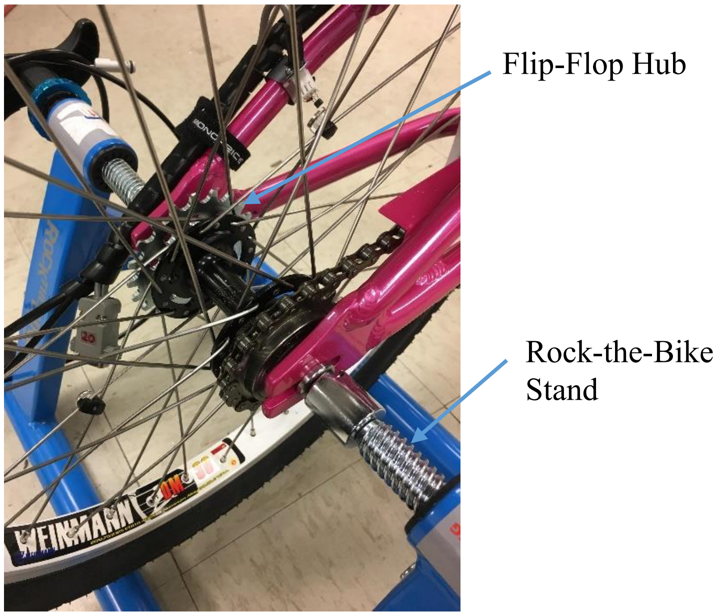


Figure 7: Rear wheel of large bicycle with flip-flop hub on far side of the wheel and the Rock-the-Bike attached to the axle of the bicycle.

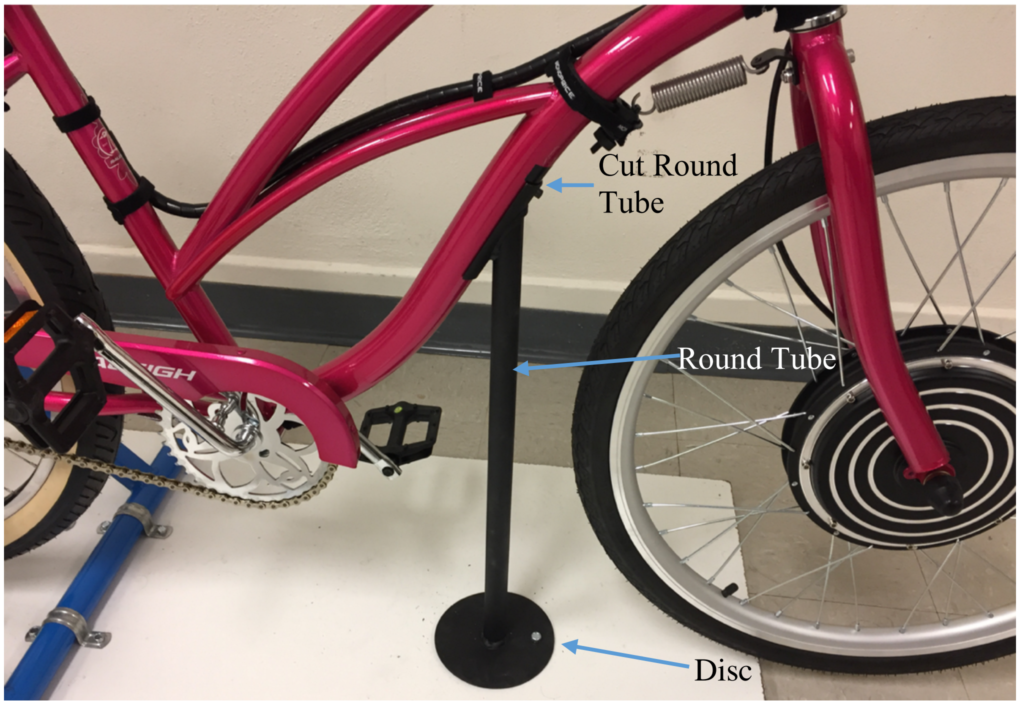


Figure 8: The downtube stand that sustained the front of the bicycle above the ground.

3.6 Front Wheel Angular Velocity

A front hub motor (model and manufacturer) rotates the front wheel. This motor was designed for an electric bike kit and was driven by proprietary controller to be actuated by a thumb dial mounted on the bike handlebars. It was necessary to reverse engineer the control signals to use the motor in this context. After some experimentation, we learned to control the input motor with an analog signal produced by the microcontroller.

3.7 Measuring the Steering Angle

The steering angle is measured by a potentiometer mounted in front of the head tube. A custom pulley was 3D-printed to key into the topmost part of the fork at the top of the head tube. A keyed spacer squeezed the pulley against the fork so that it turned with the fork and handlebars. A small belt connects this pulley to a potentiometer pulley, so that the potentiometer turns with the handlebar movement. The potentiometer was clamped onto the head tube (*Figure 9)*. Accounting for the pulley ratios (0.583), the mechanism measures 198 degrees of steering angle for the rider. The zero-degree line was set manually when the potentiometer was at 2.5V, so no bias was needed for the simulators. This allowed 99-degree turns of the handle bars in either direction. A cover was 3D-printed for the potentiometer, providing it greater protection from the environment surrounding the bicycle (*Figure 10).*

3.8 Microcontroller Electrical Communication

As *Figure 11* illustrates, a microcontroller coordinates the front hub motor controller, the back motor controller, the potentiometer for the steering angle, the potentiometer for the gear

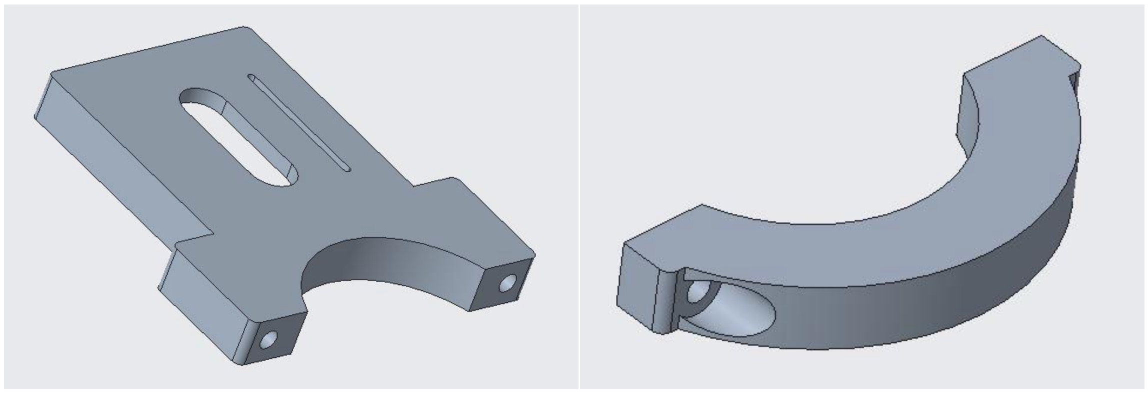


Figure 9: Potentiometer mount to connect to the head tube.

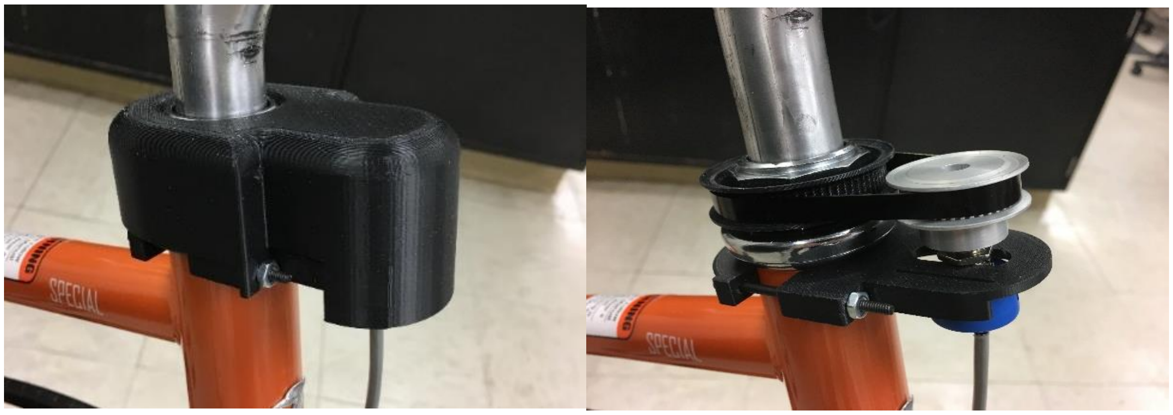


Figure 10: Potentiometer mount with and without the cover (left and right, respectively).

ratio, and sends velocity and steering angle updates to the virtual environment computer over a USB connection.

The microcontroller reads the analog 0-5V input from the gear ratio potentiometer and the steering angle potentiometer. The gear ratio potentiometer provides ten positions corresponding to ten different gear ratios. Although the internal gear hub has 11 gears, the first gear is never used. The steering angle potentiometer is described in Section 3.7.

The motor controller provides the microcontroller with an analog signal corresponding to the velocity of the back motor. The microcontroller provides the motor controller with an analog signal corresponding to the torque with which the back motor should resist the flywheel motion. The microcontroller also provides an analog signal corresponding to the front hub motor's velocity to the front hub motor controller. The front hub motor controller is powered by 36V or 48V, depending on the size of rear wheel of the bicycle.

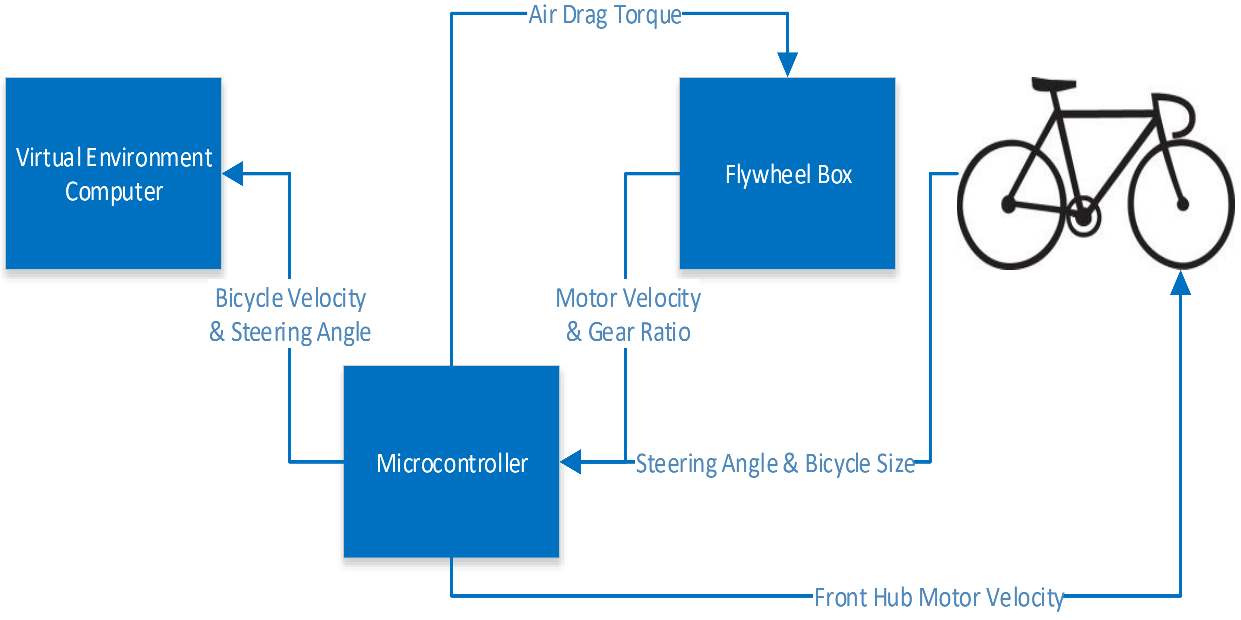

Air Drag Torque
Virtual Environment Computer
Flywheel Box
Bicycle Velocity & Steering Angle
Motor Velocity & Gear Ratio
Microcontroller
Steering Angle & Bicycle Size
Front Hub Motor Velocity

Figure 11: Diagram of the microcontroller functions throughout the Hank Bicycle Simulator.

3.9 Electrical Box

The electrical box houses the control systems for the electrical components of the bicycle. It contains the microcontroller that coordinates the input and output signals corresponding to the parameters of the rider. This includes the signal from that back motor that controls the wind drag as the rider's velocity changes. The back motor controller also needs a 24V power source for the motor controller to function. The 240V signal would go from the power source of a standard wall outlet into the electrical box, thereby powering a smaller 4-outlet electrical box. Then a 240V/24V AC/DC voltage converter from a typical printer power supply was used to power the motor controller. The back motor controller is on the top left of *Figure 12* and the voltage converter is directly beneath it.

The motor controller was also connected to the microcontroller because of the electrical messages that had to be relayed back and forth between those two components. The microcontroller was powered by a USB 2.0 cable that fed into the computer controlling the virtual reality environment. This is how the microcontroller communicated with the virtual reality environment. A 9-pin serial cable was soldered onto the open side of the microcontroller with the other end extended to the bicycle to communicate with the steering angle potentiometer and the front hub motor controller. For safety, an emergency-stop button was added to the motor controller. This would immediately cut any electrical connection to the power source upon pressing the button, thereby shutting down the entire system. An electrical schematic can be seen in *Appendix A* showing the connections throughout the electrical box as well as where the cables extend throughout the Hank Bicycle Simulator. This schematic shows how the microcontroller is interconnected with every component of the simulator. It is the central source that coordinates the electrical systems to recreate the physics of a bicycle ride.

The electro-mechanical design of the Hank Bicycle Simulator allows the rider to feel the appropriate inertial acceleration and terminal velocity that they would feel on a bicycle in a real-life scenario. The flywheel simulates the exerted kinetic energy of the bicycle rider pedaling on a level surface crossing the street from a stopped position and the back motor simulates the air

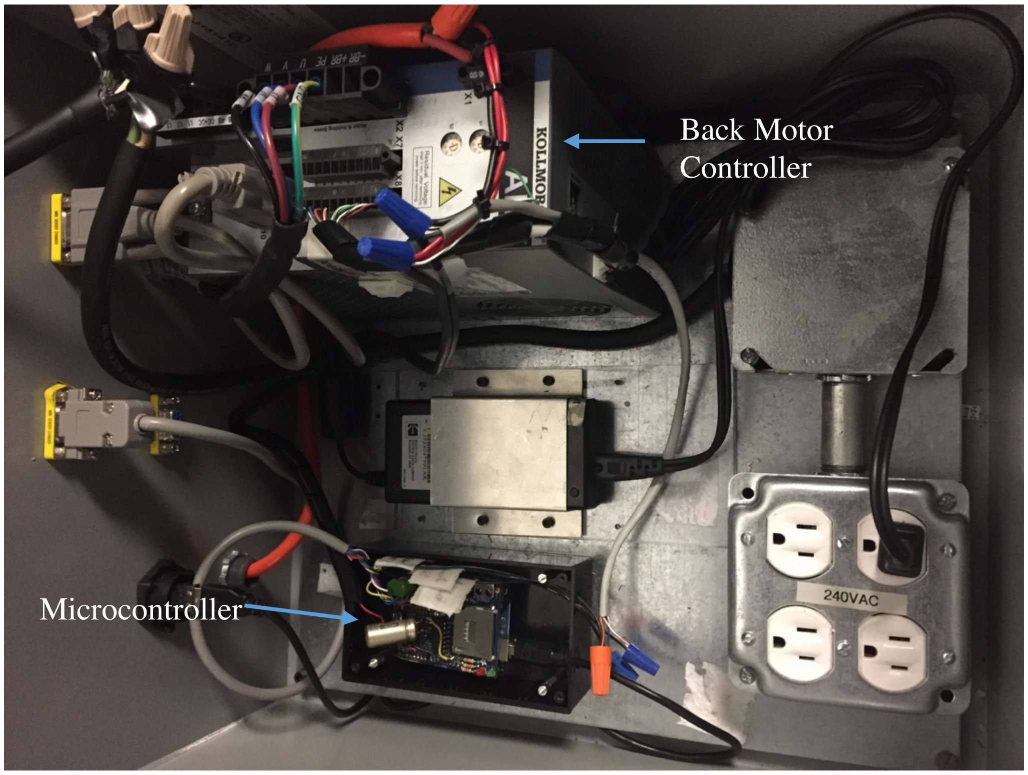


Figure 12: Electrical box for the Hank Bicycle Simulator with the microcontroller and back motor controller.

drag force that eventually holds the rider at a terminal velocity. The inertial acceleration and the terminal velocity will be validated in the experiments presented in the next section. These experiments will begin by validating the velocity output of the bicycle, the inertia of the simulator and the torque transfer through the gear ratio of the simulator. Validating those factors will verify our understanding of the Hank Bicycle Simulator. The next experiment will measure the time delay from the signals of the electrical components. The electrical components send the bicycle velocity and steering angle to the virtual reality environment. The final experiment will validate the system performance of the inertial acceleration and the terminal velocity given a specific rider and propulsive force that drives the bicycle.

CHAPTER 4

EXPERIMENTS

4.1 Experimental Set-Up

Most of the following experiments rely on a second motor (AKM53K-BKCNC-00, Kollmorgen) replacing the pedals in order to provide a consistent input, which is referred to as the input motor. This motor was temporarily mounted onto T-tracks affixed to a melamine board directly below the pedals. A keyed adaptor was milled to mount a 14t sprocket to the motor shaft. The chain between the pedals and rear wheel was replaced with a chain to connect the input motor to the 20t sprocket on the rear wheel, properly aligning the input motor to allow free chain movement (*Figure 13*). The input motor was controlled by its own motor controller (AKD-P01206-NBAN-0000, Kollmorgen, Radford, VA).

4.2 Experiment 1: *Velocity validation*

The microcontroller receives an analog signal from the back motor corresponding to the motor velocity. This is converted to the bicycle velocity with the following equations:

$$\vartheta_B = 2\pi \frac{R\vartheta_M}{G} \tag{4.1}$$

$$\vartheta_M = gV_M - b \tag{4.2}$$

where ϑ_B is the bicycle velocity in m/s, ϑ_M is the motor rotation in revolutions per second (rps), R is the radius of the rear wheel in m, G is the gear ratio from the rear wheel to the back motor, V_M is the input voltage corresponding to the velocity of the motor in volts (V), g is the gain in rps/V, and b is the velocity bias in rps based on the voltage output of the back motor at rest.

The first experiment validated this relationship by measuring the motor rotation and wheel rotation with a tachometer at various velocities and gear settings.

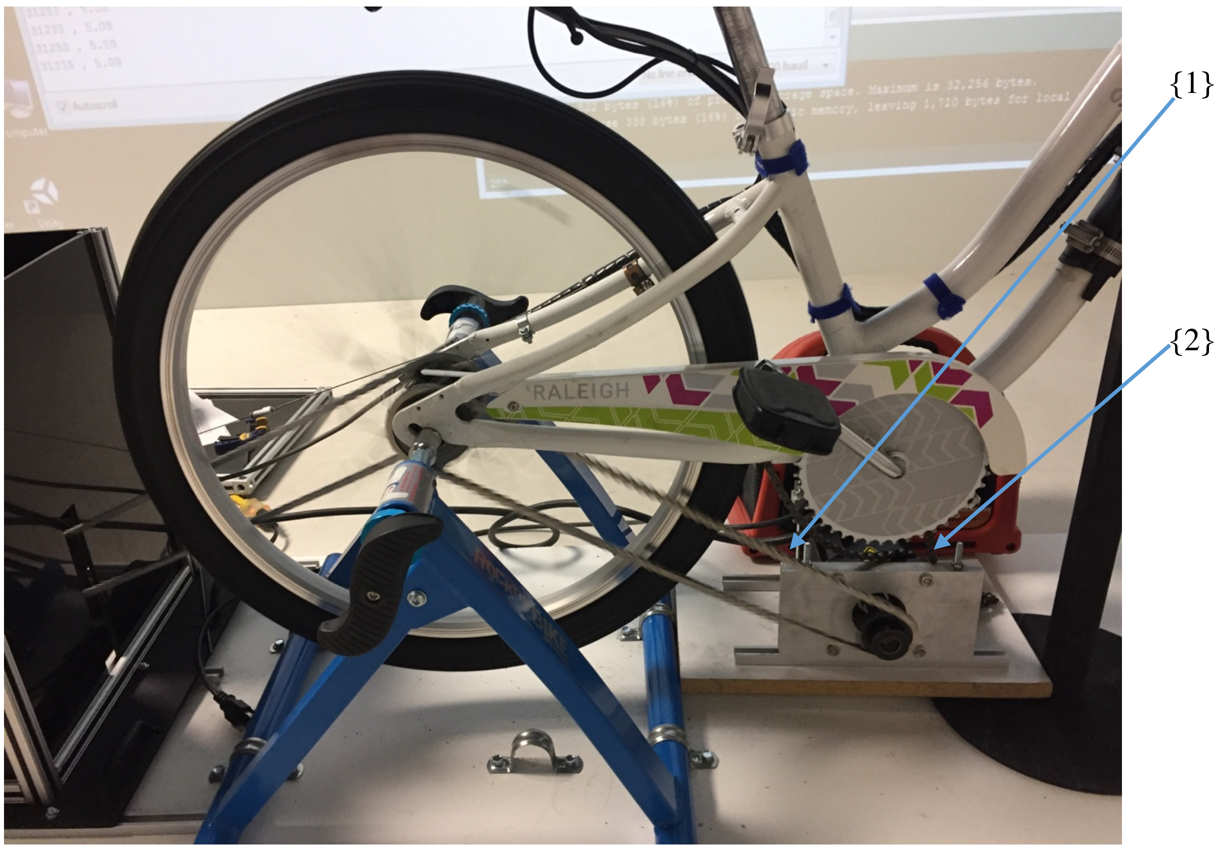


Figure 13: Experimental set-up for Experiment 1, Experiment 2, and Experiment 5. {1} Chain connecting the input motor sprocket and the rear wheel sprocket {2} input motor.

4.2.1 Methods

We applied small square, pieces of reflective tape to the timing belt pulley of the flywheel and the rear tire of the bicycle. We chose to place the reflective tape on the flywheel rather than motor because it was too difficult to fit the tachometer in to read the motion of the motor direction, and a timing belt connected the motor to the flywheel with a gear ratio of 1 that did not allow slippage.

The input motor controller was programmed to provide a constant torque and the system was allowed to accelerate until the simulated air drag from the back motor came into equilibrium with the input torque to achieve a terminal velocity. Then we used the tachometer (model, company) to measure the rotational velocity of the flywheel and the rear wheel. The microcontroller also reported the value of the flywheel and rear wheel used in its calculation. Finally, the velocity of the input motor was also recorded from the input motor controller. The process was repeated for gears 2, 4, 6, and 8 on the internal gear hub.

4.2.2 Results

Table 3 presents the experimental results. The microcontroller velocity estimates depend on the bias and gain of the analog signal. These were taken to be 2.60 rps and 9.333 rps/volts based on an empirical measurement at zero velocity and the controller's setting, respectively. The % error was calculated as (estimated – actual)/actual, with the tachometer measurement treated as the actual. The microcontrollers calculation of the rear wheel velocity also depends on the combined gear ratios of the flywheel to the internal gear hub, the gear ratio inside the internal gear hub, and the two chain sprockets between the internal gear hub and back wheel. These were calculated as 2.979, 3.841, 4.961, and 6.396 for gears 2, 4, 6 and 8, respectively. The calculation of the rear wheel velocity from the input motor controller velocity depends on the gear ratio of the chain sprockets between the input motor and the back wheel, which was taken to be 14:20.

Table 3: Flywheel and rear wheel velocities at steady state for several gears.

		Flywheel Rotational Velocity (rps)		Rear Wheel Rotational Velocity (rps)		
Gear	**Input Motor**	**Tacho-meter**	**Micro-controller (error)**	**Tacho-meter**	**Micro-controller (error)**	**Input Motor (error)**
2	2.38	4.93	4.76 (3.45%)	1.66	1.60 (3.61%)	1.67 (0.36%)
4	1.94	5.07	4.90 (3.35%)	1.32	1.28 (3.03%)	1.36 (2.88%)
6	1.84	6.43	6.20 (3.58%)	1.30	1.26 (3.08%)	1.29 (0.92%)
8	3.20	14.46	13.92 (3.73%)	2.26	2.18 (3.54%)	2.24 (0.88%)

4.2.3 Discussion and Conclusion

The microcontroller underestimates the flywheel velocity between 3.35% and 3.73%. The microcontroller underestimated the rear wheel velocity by very similar factors ranging from 3.03% to 3.61%, which seems reasonable and suggests that the gear ratios are calculated correctly and that there is no slip in the system. If the error in the microcontroller was caused by an incorrect setting in the bias, we would expect the error to be smaller with faster speeds, which was not the case. The error may then be in the motor controller, suggesting that the assigned gain and the actual gain used by the controller were not exactly the same. The input motor controller was more accurate, overestimating the velocity between 0.36% and 2.88%. Together, the results suggest that the velocity values used by the microcontroller are reasonably consistent for the range of velocities expected to be used in this application, though they may overestimate the rider velocity by approximately 3.5%.

4.3 Experiment 2: *Torque Transfer*

The next experiment validates the static and dynamic torque transfer through the mechanical system. The torque from the input motor should be reduced by the gear ratio to the back motor. The first part of the experiment measures the input motor torque required to hold the system in static equilibrium. The second part of the experiment validates that the system responds to pedal velocities with the expected torque predicted by our model of the air drag force.

4.3.1 Methods

For the first part of the experiment, the back motor was programmed to provide a constant torque. The input motor torque was set to balance that torque so that the two motors were in static opposition, then the torque of the input motor was increased until the pedals just started to move. This process was repeated for 4 levels of torque for each of gears 1, 3 and 5.

For the second part of the experiment, the input motor was set to produce a constant velocity and the response torque representing the wind drag was observed. After a brief acceleration period, we recorded the microcontroller's commanded torques at the back motor, and the resulting torque that should be perceived at the input motor controller. The experiment was repeated at 3 velocities at each of 4 gears.

The commanded torque to the back motor and the observed torque at the input motor were compared to the theoretical values calculated from *Equations (4.3)* and *(4.4)*. τ_{IM} is the torque of the input motor, τ_{BM} is the torque of the back motor, and (14/20) is the gear ratio between the input motor and the rear wheel.

$$\tau_{IM} = \frac{14}{20} F_D R \tag{4.3}$$

$$\tau_{BM} = F_D R/G \tag{4.4}$$

4.3.2 Results

Table 4 and *Figure 14* present the results of the static equilibrium portion of the experiment. In *Table 4*, the first column represents the commanded torque at the back motor. The second column represents the input motor torque required to barely move the pedals. The third column presents the predicted input value torque calculated from the gear ratio and the back motor torque that would theoretically be required to maintain the static equilibrium. The last column is the percentage error between the actual and observed values.

In *Figure 14*, the observed and theoretical values of the input motor torque are plotted as a function of the back motor torque. The three pairs of lines correspond to the three gears. A fit of the empirical observations provide an empirical estimate of the gear ratio which are 2.26, 3.42 and 4.34, compared to the theoretical values of 2.31, 3.37 and 4.35, for gears 1, 3 and 5, respectively. The R^2 value of the fit was better than 0.96 for each gear.

Table 4: Torque transfer during static equilibrium.

Gear	Back Motor Torque (Nm)	Experimental Input Motor Torque (Nm)	Theoretical Input Motor Torque (Nm)	% Error
1	1.00	2.28	2.31	1.13%
	1.11	2.51	2.56	1.94%
	1.22	2.75	2.81	2.25%
	1.33	3.00	3.07	2.18%
3	0.84	2.90	2.83	2.47%
	0.91	3.14	3.07	2.42%
	0.99	3.37	3.34	1.04%
	1.06	3.60	3.57	0.81%
5	0.76	3.28	3.31	0.85%
	0.82	3.63	3.57	1.70%
	0.88	3.75	3.83	2.11%
	0.94	4.09	4.09	0.04%

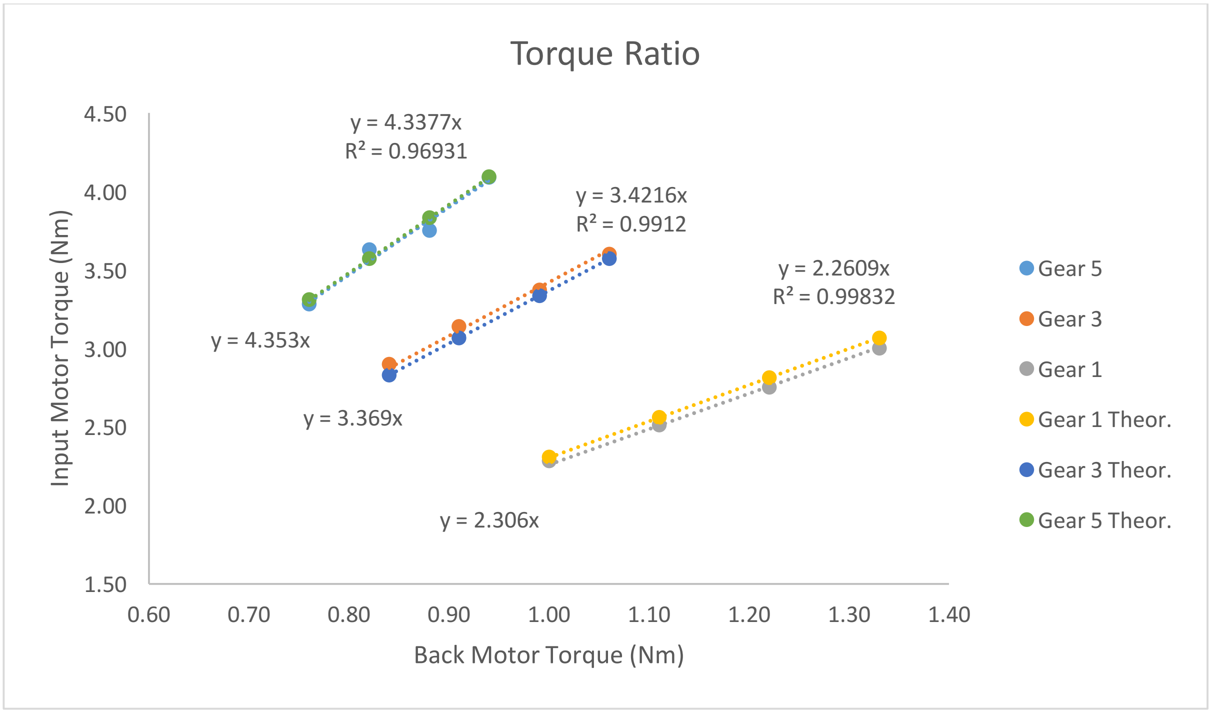


Figure 14: Graph of the torque ratio through the static system.

Table 5 presents the results of the dynamic experiment. The input velocities at each of the 4 gears are listed in the second column, followed by the commanded and theoretically desired torques at both the back and input motors.

Table 5: Dynamic response to input velocity.

Gear	Input Motor Velocity (m/s)	Input Motor Torque (Nm)		Back Motor Torque (Nm)	
		Commanded by Microcontroller	Theoretical	Commanded by Microcontroller	Theoretical
1	4.87	1.64	1.65	1.02	1.02
	5.35	1.98	1.99	1.23	1.23
	5.81	2.36	2.35	1.45	1.45
3	4.85	1.64	1.63	0.69	0.69
	5.33	1.97	1.97	0.84	0.84
	5.81	2.36	2.35	1.00	0.99
5	4.85	1.63	1.63	0.54	0.54
	5.33	1.97	1.97	0.65	0.65
	5.80	2.34	2.34	0.77	0.77
7	4.84	1.63	1.63	0.41	0.41
	5.32	1.97	1.97	0.50	0.50
	5.80	2.34	2.34	0.59	0.59

4.3.3 Discussion and Conclusion

The first part of the experiment demonstrates that the ratios between the input and back motor are what we expected, producing an error of less than 2.5% throughout. The fact that the theoretical value is higher than the experimental value suggests that the static friction in the system is negligible. Since dynamic friction is typically less than static friction, this suggests that our neglecting friction is unlikely to cause a problem, relative to the other sources of error in the system.

The second part of the experiment shows that the system responds with the expected torque and is consistent with our model of air drag. This indicates the input signals and microcontroller software all behave as expected.

4.4 Experiment 3: *Validating the Flywheel Inertia*

The third experiment validated the inertia of the flywheel and the value of the commanded torques. Based *Equation (3.5)*, the flywheel's theoretical inertia is 0.152 kg(m^2). We validated this by setting the back motor to produce a constant torque and observing how fast the flywheel accelerated, according to the equation:

$$\tau = I\alpha \tag{4.5}$$

which relates torque, τ in Nm, to angular acceleration, α.

4.4.1 Methods

The back motor was commanded to a fixed torque in the backward direction. The freewheel in the internal gear hub prevents the other portions of the transmission from moving when driven backward, so that the motor was driving only the flywheel, the flywheel shaft and the output of the internal gear hub. The back motor was commanded with four evenly-spaced torques from 0.412 Nm to 1.590 Nm. We recorded the angular velocity and time reported by the microcontroller as the flywheel accelerated. For each trial, we plotted the velocity versus time to find the angular acceleration.

4.4.2 Results

The resulting angular accelerations corresponding to each commanded torque are plotted in *Figure 15*. This line has a slope of 6.420, with an R^2 value of 0.9999.

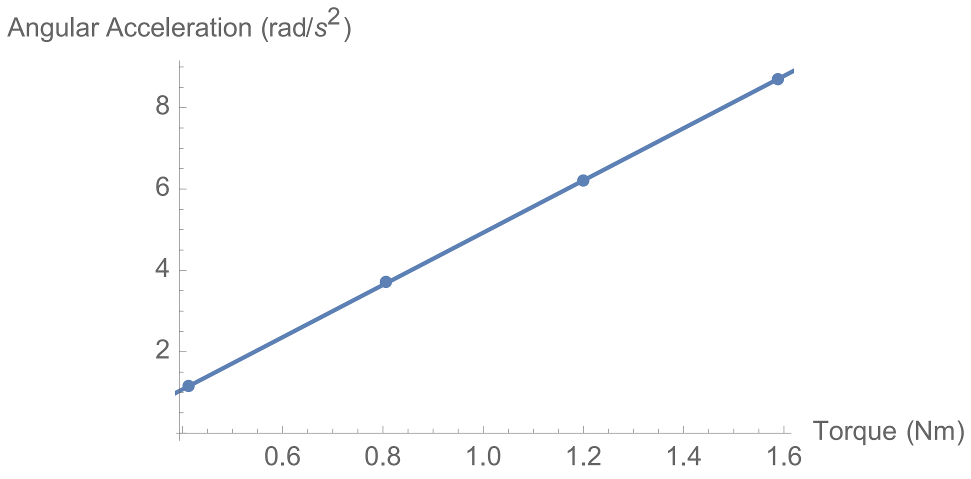


Figure 15: Torque vs. angular acceleration graph of the back motor and flywheel with a linear fit of -1.493+6.420x.

4.4.3 Discussion and Conclusion

The inverse of the slope in *Figure 15* implies that the system inertia is 0.156 kg(m^2), based on *Equation (4.5)*, which is just slightly larger than the theoretical value, which is consistent with our expectations. This suggests that the system will reproduce the rider's inertia accurately.

4.5 Experiment 4: *Time Delay*

System lag, the time between when the user takes an action and the system responds, is very important in an immersive virtual environment because lags of more than approximately 100 ms have been identified as a factor contributing to motion sickness. The overall system lag includes the time to send the signals from the bicycle hardware to the computer rendering the graphics plus the time to actually render the graphics. The following experiment measures only the time to send the signal to the computer.

4.5.1 Methods

When displaying the microcontroller output directly on a text terminal, it appears that the microcontroller produces a new output every 21 ms. That estimate includes the time to transmit the serial information to the next terminal, however. However, the lag between the microcontroller and the rendering computer may be more accurately estimated by writing a simple program on that computer to echo messages from the microcontroller back to the microcontroller and using the echoed signal to toggle the output voltage on one of the microcontroller's pins. This approach avoids the often time-consuming process of writing to a monitor. The handlebar movement provided an appropriate input signal. We programmed the microcontroller to toggle one output pin between 0 and 5V whenever the handlebars were turned past the zero-degree position. It also sent a single character "H" or "L" to the rendering computer over the USB connection. Python code on the rendering computer echoed the signal back to the microcontroller, which then toggled a second output pin between 0 and 5V. An oscilloscope connected to the two output pins could then measure the time interval between the two signals. *Figure 16* below shows a diagram of the set-up of the delay experiment.

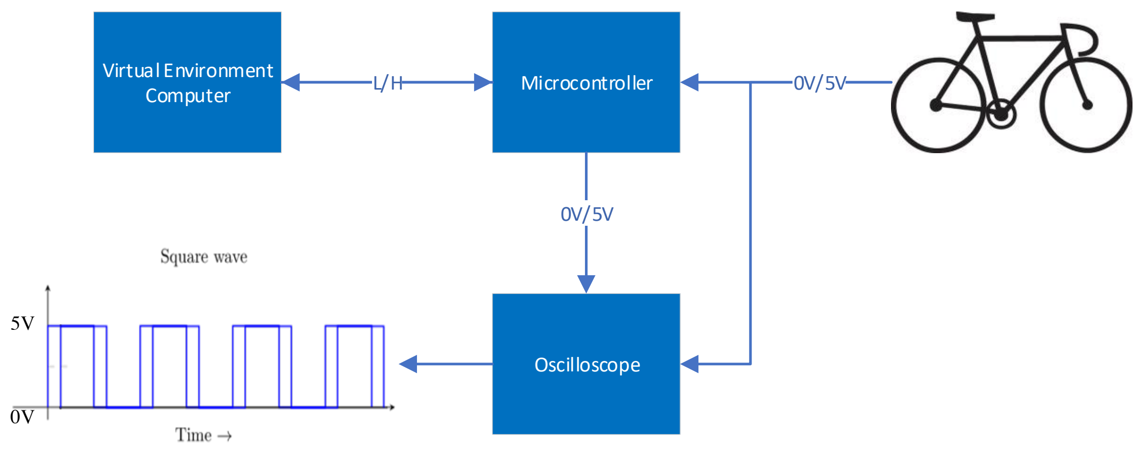


Figure 16: Diagram of the time delay experiment.

4.5.2 Results

The oscilloscope indicated that the time between the handlebar movement through the zero-degree position and the echo from the rendering computer was 40 ms. This is the round-trip time, so the lag to the rendering computer is half of that, or 20 ms.

4.5.3 Discussion and Conclusion

The microcontroller appears to loop through its code every 21 ms and another 20 ms is required to send the signal to the rendering computer. The difference between the moment at which the microcontroller samples the handlebar signal and the moment when that processed signal begins its trip to the rendering computer is some fraction of the loop time. In this experiment, the output pin designating the initiation of the signal was set right after the signal was sampled, so only a few microcontroller clock ticks expired between the sampling time and the changing of the pin value. Thus, the round-trip estimate is likely to be an accurate measure of the whole lag, and it is not necessary to add the loop time plus half the round-trip time to estimate the lag. The net lag between the user's movement of the handlebars and the arrival of the signal at the rendering computer is just 20 ms.

4.6 Experiment 5: *System Performance*

The final experiment is a summative evaluation, to determine whether, given a constant input torque, the system will respond as predicted by the solution to the differential equation presented in *Equation (2.8)*.

4.6.1 Methods

The system was configured with the input motor as in Experiment 1. Three constant torques space 0.157 Nm apart were applied for each of gears 1, 3, and 5. As the system accelerated towards the terminal velocity associated with each torque input, the microcontroller reported the time and the velocity of the rear wheel.

4.6.2 Results

Figure 17, 18 and *19* display the velocity and time reported by the microcontroller for each of the three gears and three constant input torques. The theoretical curves are provided for each condition.

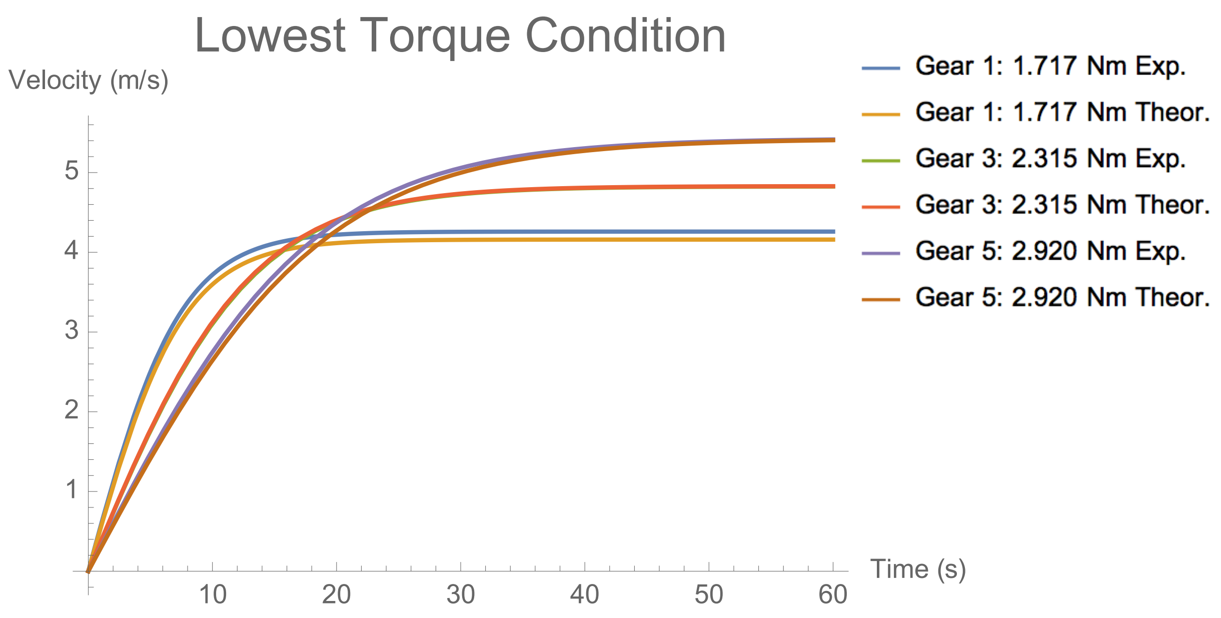


Figure 17: Plot of the system velocity under three torques applied in gear 1.

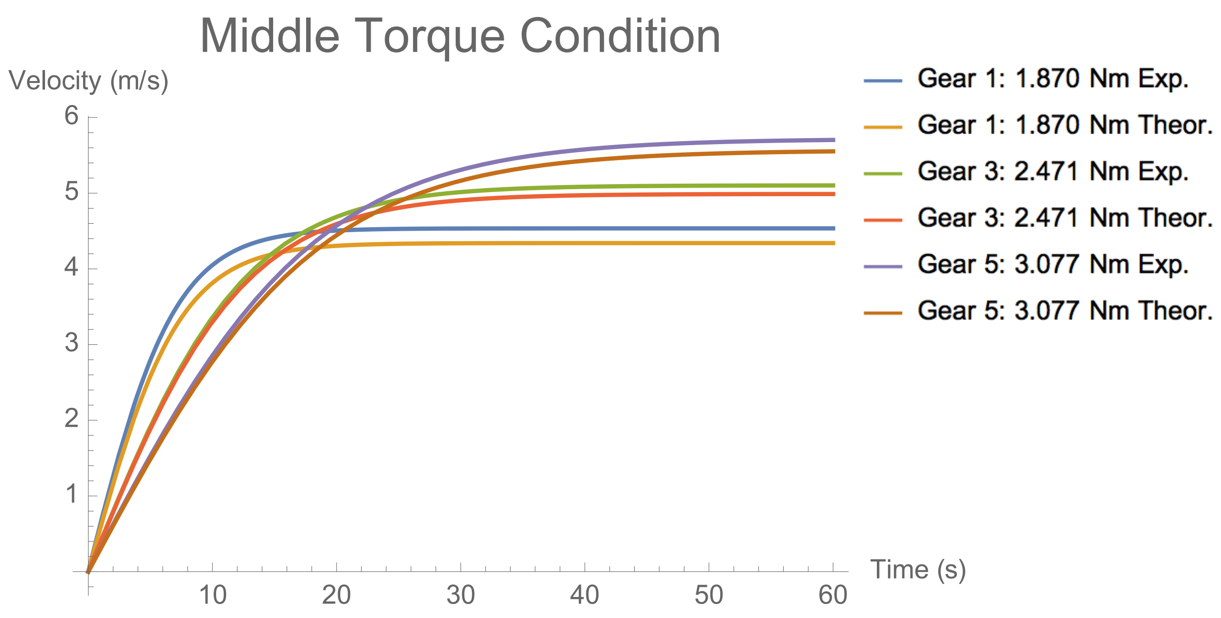


Figure 18: Plot of the system velocity under three torques applied in gear 3.

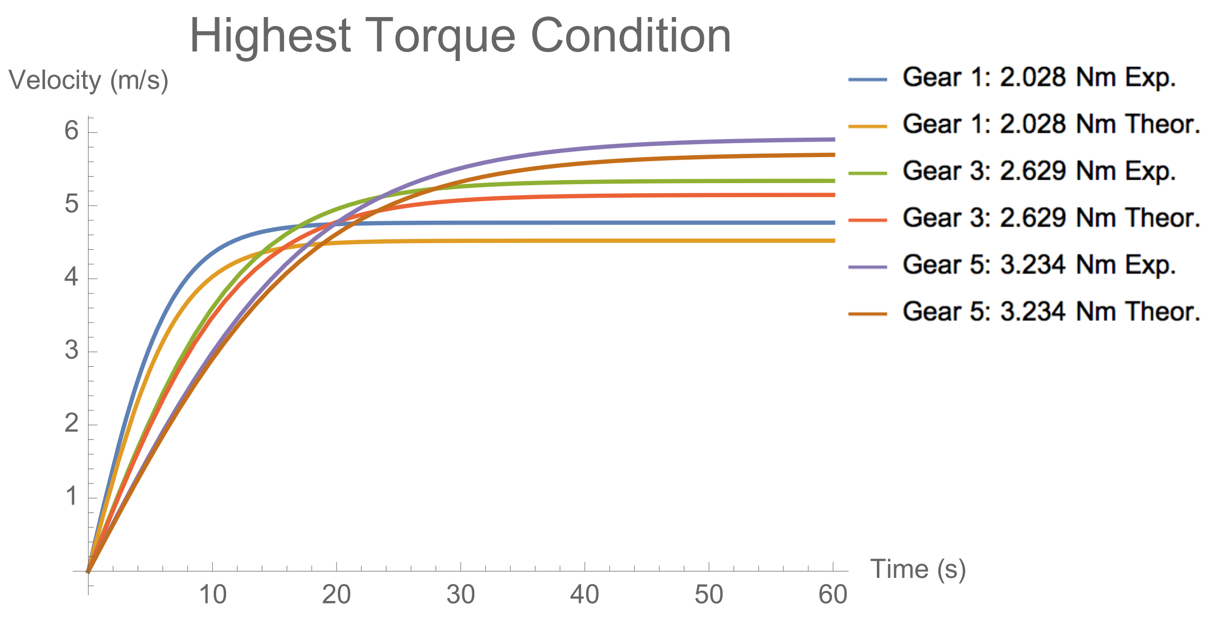


Figure 19: Plot of the system velocity under three torques applied in gear 5.

Table 6 presents the input torque, the initial acceleration and the terminal velocity, both experimentally and theoretically predicted, for each of the three gears. The estimates of the initial acceleration were derived from a nonlinear fit model using the following equation:

$$a(Tanh[bt]) \tag{4.6}$$

This equation describes the velocity of the Hank Bicycle Simulator at a given time, t. The derivative of the nonlinear fit model is the acceleration of the Hank Bicycle Simulator at any point in time. The initial acceleration is the acceleration of the simulator at $t = 0$, which simplifies the initial acceleration results to be ab.

Table 6: Theoretical and actual initial acceleration and terminal velocity for a constant input torque in several gears.

Gear	Torque	Initial Acceleration (m/s^2)			Terminal Velocity (m/s)		
		Theoretical	Measured	Error (%)	Theoretical	Measured	Error (%)
1	1.717	0.54	0.57	0.62%	4.16	4.26	2.38%
	1.870	0.59	0.65	1.24%	4.34	4.54	4.47%
	2.028	0.65	0.73	1.94%	4.52	4.77	5.46%
3	2.315	0.37	0.37	0.07%	4.83	4.83	0.08%
	2.471	0.40	0.40	0.02%	4.99	5.10	2.26%
	2.629	0.42	0.43	0.20%	5.15	5.34	3.73%
5	2.920	0.29	0.30	0.20%	5.42	5.43	0.07%
	3.077	0.31	0.31	0.05%	5.57	5.72	2.71%
	3.234	0.32	0.33	0.10%	5.71	5.92	3.70%

4.6.3 Discussion and Conclusion

The initial accelerations across the three gears was better than 2%, and much better than 1% for most cases that would be encountered for heavier riders. The terminal velocities were better than 5.5%, again most accurate for heavier riders. Given the mechanical and electrical complexity of the system, this is quite good and most likely sufficient for the application scenario.

CHAPTER 5

DISCUSSION & CONCLUSION

The purpose of the Hank Bicycle Simulator is to observe the behavior of children and adults riding across a street on a bicycle, with particular emphasis on how participants judge the gaps between passing cars when choosing their moment to cross. The participants' decisions are likely to be dominated by their estimates of how long it will take to cross the street. This estimate is likely to be a function of their past experience with bicycles, an experienced dominated by the challenge of physically overcoming the mechanical inertia of the system.

Thus, the most important factor in the Hank Bicycle Simulator is the simulation of the inertial force. As seen throughout the results of the experiments, the flywheel produces a sufficient amount of kinetic energy compared to the kinetic energy produced by the rider. This can be observed by the starting acceleration results in experiment 5. The error on the starting acceleration is 0.20% or under for gears 3 and 5. Gear 1 has a higher error, but this gear produces too low of a simulated weight for any rider to use (gear 1 is for a 10 lb. rider). Thus, gear 1 will not be used at all in the virtual environment, and neither will gear 2 for the same issue.

The air drag produced by the back motor of the system gives the rider a specific terminal velocity based on the force they use for pedaling. The terminal velocity will rarely be a factor in the application of the Hank Bicycle Simulator because the riders in the experiment are simply crossing the street. This gives little time to accelerate to terminal velocity, making terminal velocity of the bicycle unlikely. The results for the terminal velocity in the system performance experiment were favorable, however. Just like the starting acceleration, gear 1 produced the highest error in terminal velocity. Since gear 1 is not used, the most important results of the terminal velocity are for gears 3 and 5. The highest error was 3.73%, meaning the rider had a terminal velocity that was about 0.20 m/s faster than the theoretical terminal velocity. The lowest error was 0.07%, meaning that the terminal velocity of the rider was under 0.01 m/s faster than the theoretical value.

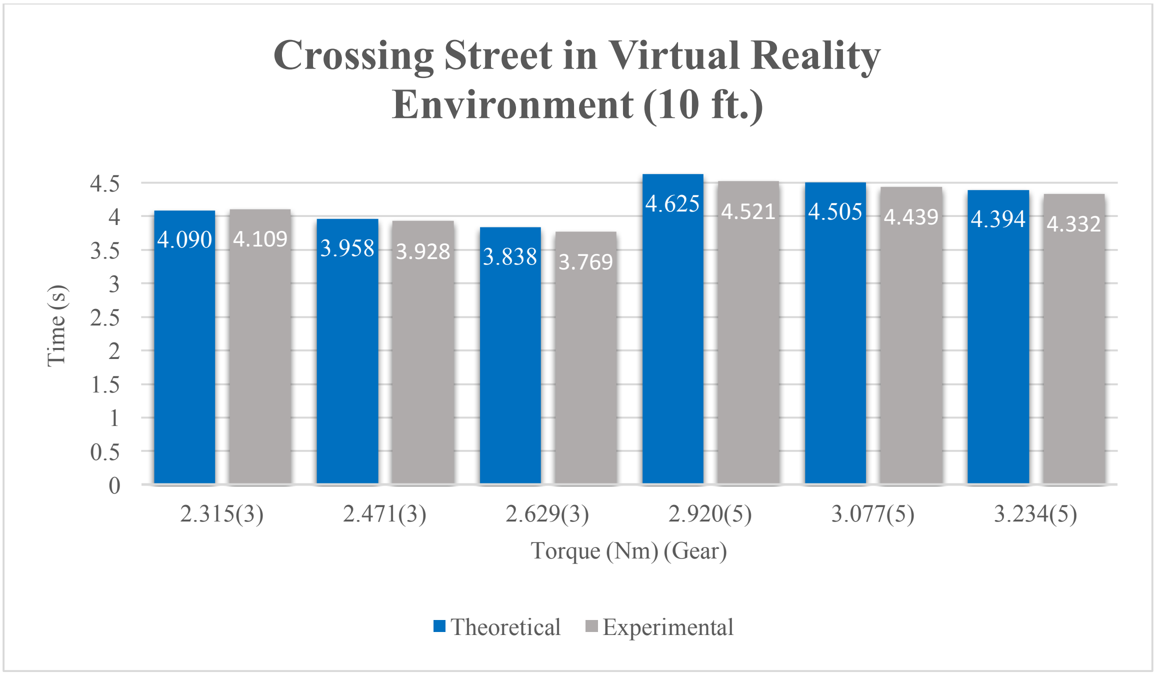


Figure 20: Graph showing the time difference between the experimental rider and the theoretical rider crossing the street in a virtual reality environment.

Based on the results from the system performance experiment, given a constant propulsive force the time to cross a street can be found (*Figure 20)*. The street that the Hank Bicycle Simulator is crossing in the virtual reality environment is about 10 ft. wide or 3.06 m. From the data of the experiments, we can calculate the time it takes the rider to move 10 ft.

starting from rest and pedaling at a constant propulsive force. The average time difference between the predicted and observed time to cross the street in gears 1, 3, and 5 is about 80 ms. For the more commonly used gears 3 and 5, the average time difference goes down to about 60 ms with the Hank Bicycle Simulator typically being faster. When a real rider uses the Hank Bicycle Simulator they will use higher propulsive forces that will in turn increase the velocity of the system. Increasing the propulsive force will decrease the time it takes to cross the street, as well as decrease the time difference between the theoretical and experimental performance. These times produced by the Hank Bicycle Simulator are very close to the theoretical time in a real environment. Thus, the Hank Bicycle Simulator has proven that it can produce accurate performances given 33 different simulated weights.

A study by Plumert et al. (2004) observed 10-year olds, 12-year-olds and adults crossing a two-lane intersection using a bicycle simulator. When the rider stopped at the intersection traffic would approach the intersection from the left of the rider in the lane closest to them. The rider would then have to choose an appropriate gap width to safely cross the intersection. On average, all participants chose to cross the intersection during the 3.5 s gap size between vehicles. They found that the average time left to spare when the rider cleared the lane of the approaching vehicle decreased with the younger riders. The average time left was 1.13 s for the 10-year-old riders, 1.49 s for 12-year-old riders and 1.98 s for the adult riders. The 60 ms time difference between the predicted and observed time will have little to no effect on the performance of the riders in the Hank Bicycle Simulator. The riders will typically be around 60 ms ahead of where they may predict, however that 60 ms results in a displacement difference of only a few inches. For example, in the worst case with the widest time difference of 104 ms, if we assume a constant propulsive force of 2.920 Nm in gear 5, this will result in an observed displacement of about five inches ahead of the predicted displacement.

In future iterations of the Hank Bicycle Simulator, there are various ways that this simulator can continue to improve for better performance and quality. The gear ratio switch could be changed to be automatic. When the gear is set it can measure the velocity of the rear wheel and compare it to the velocity of the motor. This will give the gear ratio of the system. Reduction of the steering delay could always be improved, as well as the mechanics of the bicycle. Further iterations will be able to refine the mechanical model for better performance.

The next step for the Hank Bicycle Simulator is to compare the behavior of riders as they use it. By observing and testing the riders in their natural environments, we can record their behavior and other factors that might be useful for future changes to the Hank Bicycle Simulator. For instance, we could observe the rider's steering performance and the resistance they feel from steering at a given velocity in a natural environment. Using that information, we can compare the Hank Bicycle Simulator to the natural environment and implement a steering damper that produces realistic steering resistance dependent on the velocity. These comparisons would help enhance the Hank Bicycle Simulator experience.

Overall, it is easy to see how quick the inertial response of the Hank Bicycle Simulator is given any rider compared to the delay of other simulators. The other simulators mentioned in *Table 1* all use electrical systems that have inevitable electronic delays. The bicycle simulators, as well as other simulators from the literature review, lack validation of the theory behind their structure. They produced well-made bicycle simulators, but did not compare their simulator's system performance to the real-life riders. The mechanical simulation of inertia in the Hank Bicyce Simulator provides an accurate and immediate reproduction of the expected inertia. How this compares with other simulators, particularly those relying on electrical motors and magneto-rheological fluid brakes is unclear, because previous studies have not provided enough performance detail to be able to make a valid comparison. We hope that the current study will encourage future researchers to report the details of their underlying models and their system performance, particularly those with ties to the physical parameters of the simulation they wish to reproduce, so that the community may move forward together, taking the best elements of each system.

REFERENCES

Babu, S. V. *et al*. (2009) A Virtual Peer for Investigating Social Influences on Children's Bicycling. *2009 IEEE Virtual Reality Conference*, Lafayette, 91-98. DOI: 10.1109/VR.2009.4811004

Babu, S. V. *et al*. (2011). An Immersive Virtual Peer for Studying Social Influences on Child Cyclists' Road-Crossing Behavior. *IEEE Transactions on Visualization and Computer Graphics*, *17*(1), 14-25. DOI: 10.1109/TVCG.2009.211

Deligiannidis, L., & Jacob, R. (2006). The VR Scooter: Wind and Tactile Feedback Improve User Performance. *3D User Interfaces (3DUI'06),* 143-150. DOI: 10.1109/VR.2006.131

Grechkin, T. Y., Chihak, B. J., Cremer, J. F., Kearney, J. K., & Plumert, J. M. (2013). Perceiving and acting on complex affordances: How children and adults bicycle across two lanes of opposing traffic. *Journal of Experimental Psychology: Human Perception and Performance, 39*(1), 23-36. DOI:10.1037/a0029716

Gross, A., Kyle, C., & Malewicki, D. (n.d.). "Human Powered Vehicle Performance" [Digital image]. Retrieved from http://2.bp.blogspot.com/-6rUkPBorBnY/UKGkyUzCMII/AAAAAAAABVA/3KflV7aANbg/s1600/Human Powered Vehicle Data.jpg

He, Q., Fan, X., & Ma, D. (2005). Full bicycle dynamic model for interactive bicycle simulator. *Journal of Computing and Information Science in Engineering*, *5*(4), 373-380. DOI: 10.11115/1.2121749

Huang, S. F. *et al.* (2008). The Comparisons of Heart Rate Variability and Perceived Exertion During Simulated Cycling with Various Viewing Devices. *Presence*, *17*(6), 575-583.

Kikuchi, T., Kobayashi, K. & Sugiyama, M. (2012). Development of virtual reality bike with cylindrical MR fluid brake. *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, Guangzhou, 1753-1758.

Kwon, D. *et al.* (2001). KAIST interactive bicycle simulator. *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, *2001,* 3, 2313-2318. DOI: 10.1109/ROBOT.2001.932967

Kwon, D. *et al.* (2002). KAIST interactive bicycle racing simulator: the 2nd version with advanced features. *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on, 2002, 3*, 2961-2966. DOI: 10.1109/IRDS.2002.1041722

Kyle, C. *et al.,* (2001). International Human Powered Vehicle Association (D. G. Wilson, Ed.). *Human Power*, *52*(Summer 2001). Retrieved from http://www1.bhpc.org.uk/Data/Sites/1/Uploads/humanpower/PDF/hp52-2001.pdf

Leblanc, M. & Sicard, P. (2010). EMR and inversion-based control of a virtual reality bicycle trainer. *2010 IEEE Vehicle Power and Propulsion Conference*, 1-7. DOI: 10.1109/VPPC.2010.5728993

Mestre, D., Dagonneau, V., & Mercier, C. (2011). Does Virtual Reality Enhance Exercise Performance, Enjoyment, and Dissociation? An Exploratory Study on a Stationary Bike Apparatus. *Presence*, *20*(1), 1-14.

Miyanoue, K., Suzuki, M., & Yai, T. (2015). *Journal of Japan Society of Civil Engineers, Ser. D3 (Infrastructure Planning and Management), 71*(5), I_859-I_604. DOI: 10.2208/jscejipm.71.I_589

National Center for Statistics and Analysis. (2016, May). Bicyclists and other cyclists: 2014 data. (Traffic Safety Facts. Report No. DOT HS 812 282). Washington, DC: National Highway Traffic Safety Administration.

Plumert, J. M., Kearney, J. K. & Cremer, J. F. (2004). Children's Perception of Gap Affordances: Bicycling Across Traffic-Filled Intersections in an Immersive Virtual Environment. *Child Development, 75*, 1243–1253. DOI:10.1111/j.1467-8624.2004.00736.x

Plumert, J. M., Kearney, J. K. & Cremer J. F. (2007). Children's Road Crossing a Window into Perceptual–Motor Development. *Current Directions in Psychological Science*, *16(5)*, 255-258. DOI: 10.1111/j.1467-8721.2007.00515.x

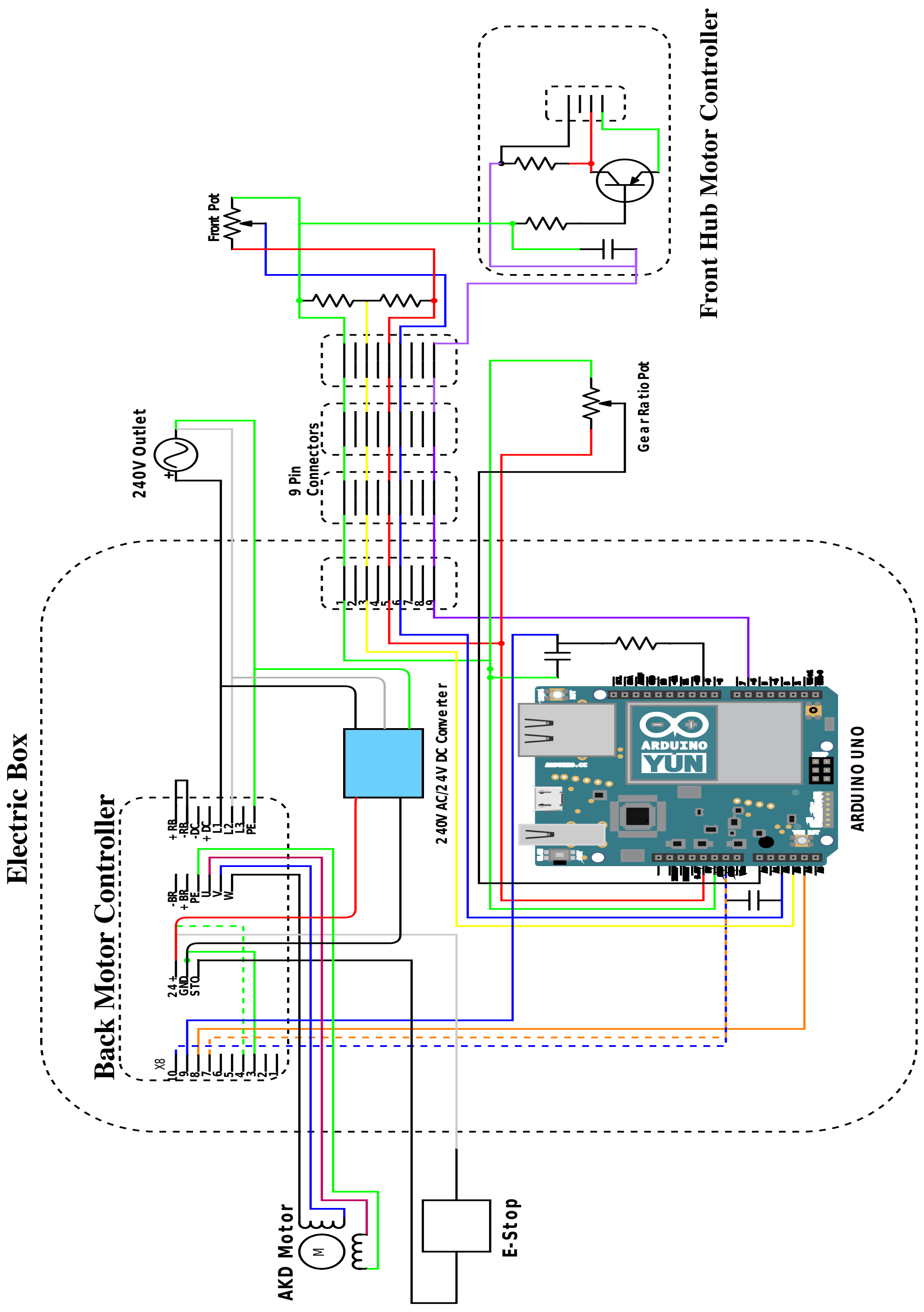
Sari, R. F. *et al.* (2009). 3D object implementation on bicycling at ui virtual reality application based on 3D-Gamestudio," *2009 International Multiconference on Computer Science and Information Technology*, 509-515.

Tang, Y *et al.* (2007). The Development of a Virtual Cycling Simulator. *Technologies for E-Learning and Digital Entertainment, 2007. Second International Conference, Edutainment 2007*, 162-170. DOI: 10.1007/978-3-540-73011-8_18

Wilson, D. (2004). Power and speed. In *Bicycling science*. (pp. 123-172), Cambridge, MA: MIT Press.

APPENDIX A

HANK BICYCLE SIMULATOR ELECTRICAL SCHEMATIC



APPENDIX B

MICROCONTROLLER CODE

```
/* Full_bike_program:  An arduino sketch by Jaemin Powell
Inputs:
        A potentiometer input from 0-5V, with approximately 2.5 V straight ahead, larger turns to right.
        A hall effect sensor that starts high and goes low each time a magnet passes the sensor. Assume
        two magnets on the wheel. A velocity input from the motor controller, 0-5V. Scale depends on
        motor controller and must be compiled into this sketch. Bike size voltage:  0V for 20" diam
        wheel, 2.5V for 24" diameter wheel and 5V for 26" diameter wheel.
Outputs:
        Front wheel motor controller, 0-5V.
        Send torque command to back motor controller, 0-5V.
        Text read by Hank Bicycle Simulator (velocity (m/s), steering angle (degrees))*/
const short buffLength = 32;
unsigned long buff[buffLength];
int pBuff = 0;
float TorqueConstant = 1 / 1.468; // 1 / 1.242 for old motor, 1 / 1.468 for new motor Arms / Nm
float VoltsPerARMS = 1 / 1.580; // V / Arms, value found from motor controller under Motor
                                    input
float RPSperVolt = 9.3333; // Value found from motor controller under Motor output
float VoltsPerFrontVelocity = 5 / 11.176; // 5 V / 11.176 m/s (max speed)
//short samplingIntervalSecs = 5; // Seconds to intergrate hall effect sensor clicks
float SteeringZeroOffset = 2.5; // Volts
float SteeringVoltsToDegrees = 340 / 5; // Degrees / Volts Figure this out from datasheet
//pins 3, 4 don't work
float SteeringGearRatio = 0.583; // 42 / 72 ratio between steering angle and pot
float MotorRPSOffset = 2.60;
short HallEffectInputPin = 3;
short SteeringAngleInputPin = 2; // A2
short FrontMotorOutputPin = 6;
short MotorTorqueOut = 9;
short VelocityInputFromMotorController = 4; // A4
short BikeSelectionInput = 3; // A3
//short LedPin = 7;
short GearRatioPin = 0; // A0
/////////////////////////////////////////////////////////////////////////////////
void setup() {
        pinMode(HallEffectInputPin, INPUT);
        pinMode(SteeringAngleInputPin, INPUT);
        pinMode(FrontMotorOutputPin, OUTPUT);
        pinMode(MotorTorqueOut, OUTPUT);
        pinMode(BikeSelectionInput,INPUT);
        pinMode(VelocityInputFromMotorController, INPUT);
        // pinMode(LedPin,OUTPUT);
        pinMode(GearRatioPin, INPUT);
```

```
	Serial.begin(19200);
	attachInterrupt(digitalPinToInterrupt(HallEffectInputPin), interrupt_function,FALLING);}
///////////////////////////////////////////////////////////////////////////////
/*int countRecentHallClicksInBuffer(int samplingIntervalSecs) {
	int counter = 0;
	unsigned long currentTime;
	int ptr = pBuff - 1;
	if (ptr < 0) {ptr = buffLength - 1;}
	currentTime = millis();
	while ((buff[ptr] > currentTime - samplingIntervalSecs * 1000) && ptr != pBuff) {
		counter++;
		ptr = ptr - 1;
		if (ptr < 0) {ptr = buffLength - 1;}}
	  return (counter);}
///////////////////////////////////////////////////////////////////////////////
void HallEffect(int *HallInput,int *TotalTime, int *counter, float *RPM, int *OldVal) {
	if (*HallInput == 0) {if (*OldVal!= 0) { *counter=*counter + 1;}}
	*RPM = *counter / (*TotalTime / 1000);}
///////////////////////////////////////////////////////////////////////////////
float getRearWheelRPS(int samplingIntervalSecs) { // rps
	int magnetsPerWheel = 2;
	float rotations = countRecentHallClicksInBuffer(samplingIntervalSecs) / magnetsPerWheel;
	float rotationsPerSecond = rotations / samplingIntervalSecs;
	return (rotationsPerSecond);}*/
///////////////////////////////////////////////////////////////////////////////
float getBikeWheelCircumference() { // m, from A3
	float circ;
	int val;
	int maxAnalog = 1023;
	val = analogRead(BikeSelectionInput);
	if (val < (maxAnalog / 3)) {circ = 1.5959; // approx. from 1.5959 m}
	else if (val < ((2 * maxAnalog ) / 3)) {circ = 1.9151; // approx. from 1.9151 m}
	else {circ = 2.0747; // approx. from 2.0747m}
	return (circ);}
///////////////////////////////////////////////////////////////////////////////
float getMotorVelocity() { // rps
	float val;
	int maxAnalog = 1023;
	int secsPerMinute = 60;
	float volts;
	int counterLoop = 0;
	float valAvg = 0;
	float total = 0;
	long Start = millis();
	long Stop = millis() - Start;
	while (Stop < 20) {val = analogRead(A4); total = total + val; counterLoop++;
		Stop = millis() - Start;}
	val = analogRead(A4);
	valAvg = total / counterLoop;
	volts = 5.0 * valAvg / maxAnalog;
	float velocityRPS = RPSperVolt * volts - MotorRPSOffset;
```

```
        return (velocityRPS);}
/////////////////////////////////////////////////////////////////////////////
float calcBikeVelocity(float motorVelocity, float wheelCircumference, float gearRatio) {
        float bikeVelocity;
        bikeVelocity = motorVelocity * wheelCircumference / gearRatio; //go from motor rps to bike m/s
        return (bikeVelocity);}
/////////////////////////////////////////////////////////////////////////////
float calcAirDrag(float bikeVelocity, float wheelCircumference){ // in N
        float massDensityAir = 1.184; // kg/m^3.  Air at 25 degrees C
        float dragCoefficient = 1.1;
        float area; /* 0.4 m^2 - 0.7 m^2 typical http://www.analyticcycling.com/ForcesPower_Page.html
        */
        /* estimated from http://2.bp.blogspot.com/-
        6rUkPBorBnY/UKGkyUzCMII/AAAAAAAABVA/3KflV7aANbg/s1600/Human+Powered+Ve
        hicle+Data.jpg */
        if (wheelCircumference == 1.5959) {area = 0.40; // m^2}
        else if (wheelCircumference == 1.9151) {area = 0.50; // m^2 .45}
        else {area = 0.60; // m^2 .5}
        return (0.5 * massDensityAir * bikeVelocity * bikeVelocity * dragCoefficient * area);}
/////////////////////////////////////////////////////////////////////////////
float getSteeringAngle(void) { // degrees
        int val = analogRead(SteeringAngleInputPin);
        float volts = 5.0 * val / 1023.;
        float angle = SteeringVoltsToDegrees * (volts - SteeringZeroOffset) * SteeringGearRatio;
        return (angle);}
/////////////////////////////////////////////////////////////////////////////
float getTorqueOutput(float airDragForce, float wheelCircumference, float gearRatio) { // Nm
        float wheelRadius = wheelCircumference / (2 * PI); // m
        float desiredTorque = airDragForce * wheelRadius; // Nm of the backwheel
        float torqueOutput = desiredTorque / gearRatio; // Nm of the motor
        return (torqueOutput);}
/////////////////////////////////////////////////////////////////////////////
float getGearRatio() {
        float gearRatio;
        int val = analogRead(GearRatioPin);
        float volts = 5.0 * val / 1023.;
        if (volts < 0.3) {gearRatio = 2.306;}
        else if (volts < 0.8) {gearRatio = 2.979;}
        else if (volts < 1.4) {gearRatio = 3.369;}
        else if (volts < 1.9) {gearRatio = 3.841;}
        else if (volts < 2.5) {gearRatio = 4.353;}
        else if (volts < 3.0) {gearRatio = 4.961;}
        else if (volts < 3.6) {gearRatio = 5.653;}
        else if (volts < 4.1) {gearRatio = 6.396;}
        else if (volts < 4.7) {gearRatio = 7.293;}
        else {gearRatio = 8.260;}
        return gearRatio;}
/////////////////////////////////////////////////////////////////////////////
void sendTorqueToMotorController(float torqueOutput) {
        float ARMS = torqueOutput * TorqueConstant;
        float volts = ARMS * VoltsPerARMS;
```

```
            int val = (int) ((255 * (volts / 5.0)));
            if (val < 3) {analogWrite(MotorTorqueOut, 0);}
            else {analogWrite(MotorTorqueOut, val);}}
////////////////////////////////////////////////////////////////////////////////
void sendVelocityToFrontMotor(float bikeVelocity) {
            float volts = bikeVelocity * VoltsPerFrontVelocity;
            int valOffset = 93;
            int val = (int) (valOffset + (255 * (volts / 5.0)));
            if (bikeVelocity < 0.1) {analogWrite(FrontMotorOutputPin, 0);}
            else if (val < 255) {analogWrite(FrontMotorOutputPin, val);}
            else {analogWrite(FrontMotorOutputPin, 255);}}
////////////////////////////////////////////////////////////////////////////////
void printResults(float velocity, float steeringAngle) {
            Serial.print(velocity); // bike m/s
            Serial.print(" , ");
            Serial.println(steeringAngle); // degrees}
////////////////////////////////////////////////////////////////////////////////
/*void updateLED(float steeringAngle) {
            if (steeringAngle > 0) {digitalWrite(LedPin, HIGH);}
            else {digitalWrite(LedPin, LOW);}}*/
////////////////////////////////////////////////////////////////////////////////
void loop() {
            float gearRatio = getGearRatio();
            float wheelCircumference = getBikeWheelCircumference(); // m
            float motorVelocity = getMotorVelocity(); // rps
            float bikeVelocity = calcBikeVelocity(motorVelocity, wheelCircumference, gearRatio); // m/s
            float airDragForce = calcAirDrag(bikeVelocity, wheelCircumference); // N
            float torqueOutput = getTorqueOutput(airDragForce, wheelCircumference, gearRatio);   // Nm
            sendTorqueToMotorController(torqueOutput);
            sendVelocityToFrontMotor(bikeVelocity);
            float steeringAngle = getSteeringAngle();
            printResults(bikeVelocity, steeringAngle);
            //  updateLED(steeringAngle);}
////////////////////////////////////////////////////////////////////////////////
void interrupt_function() {buff[pBuff] = millis(); pBuff = (pBuff + 1 )% buffLength; }
```