Honors Program Capstone

PID Control Learning Device

Scott Andermann

Dr. Brianno Coller

Dr. Jenn-Terng Gau

Northern Illinois University

5/3/2013

Table of Contents

List of Figures

List of Tables

List of symbols

a_p - Acceleration of person

a_b - Acceleration of wheel

$a_{p/b}$ - Acceleration of person relative to wheel

$\dot{\theta}$ – Angular velocity

$\ddot{\theta}$ – Angular Acceleration

$\ddot{x}$ – Linear Acceleration

L – Length from pivot to center of gravity

Θ – Angle relative to vertical

T – Tension in rod

τ – Applied torque

m – Mass of the person

M – Mass of the wheel

g – Acceleration due to gravity

F – Force due to friction

R – Radius of wheel

I_w – Moment of inertia of the wheel

k_d – Derivative Gain

k_p – Proportional Gain

k_i – Integral Gain

Acknowledgement

I would like to thank Dr. Coller for his guidance on the control theory regarding this project. I would also like to thank Northern Illinois University for use of the facilities and knowledge to be able to create this project. I would also like to thank Dr. Gau for his encouragement and for keeping me on track. Finally, I would like to thank Michael Marolda for all of the work he put in this semester toward the preliminary steps of this project. I would have no hope to finish this project without their help.

Abstract

PID control theory has been used to solve simple and complex problems. Currently the best way of teaching PID control is describing a system and determining the outputs based on charts and graphs. A self balancing robot can be controlled by a PID controller. If the PID gains can be changed in real time students will gain a deeper understanding of how a PID controller works and how to adjust the gains to improve the performance of the controller. This project was approached analytically to find the gains that balance the robot. A prototype was built to prove that this idea is plausible. The prototype balanced well using the calculated gains which verify the mathematical model. A robot was created that allowed the properties of a PID controller to be changed in real time and the results can be seen and felt in a more meaningful way that simply looking at response graphs. A simple controller that can be investigated and developed further by students was also created using a simple to use software and computer code.

Chapter 1.1: Background

Segway Human Transporters have fascinated many people since they were introduced. In its simplest element, it is the common control theory problem, an inverted pendulum. A simple control law can be developed to balance the pendulum and has been many times. The Segway adds another aspect of complexity in that the base of the inverted pendulum is not simply moved, but the rotation of the wheels also contributes to the dynamics of the system. This control theory problem has been tackled many times, but usually the development ceases at that point.

For this project to be successful a reliable controller for balancing in the forward and backward directions and being able to adjust the gains while riding the vehicle. The turning will be accomplished by turning handlebars. This project will utilize control theory, design processes, and other engineering principles to create a safe and operational final product.

Chapter 1.2: Initial Thoughts

A difficult concept in control theory is how the controller gains affect the overall dynamics of the system. For example, how does increasing the proportional gain affect the reaction of the vehicle along with the balancing characteristics. While there is a simple answer to this, to more fully understand this phenomenon being able to alter the gain in real time will give a new perspective on this concept. The goal is to be able to demonstrate, in real time, how the system is affected when each gain is adjusted.

Much of the undergraduate control theory focuses on decaying oscillations. The most common way of displaying is this is through plots and charts. While this is a functional way of introducing the concept to students it is narrow minded and if a student cannot understand the concept through plots there is no other way to demonstrate the concept. This project will bridge that gap. Some students learn more effectively hands on when they can experiment with real things. A robot that can demonstrate how these gains affect the system will allow these students to change the gains and make interpretations of their own not based on plots that may not mean anything to them.

Chapter 1.3 Professional Components

The design of this project included many different aspects including manufacturability, economic, and social. Manufacturabilty was a huge component because the scope of the project included the creation of a prototype. Since this is an undergraduate project with a very limited budget, the overall design will focus on a simple yet effective design. There are many individual parts and in an attempt to keep costs down as well as keep the overall simplicity, sacrifices in cosmetics will be made. However, there are bound to be some parts that must be complex and will require more advanced manufacturing processes.

The economic aspect of this project will focus on staying on budget. Top of the line balancing robots cost upwards of $6500, but there are many features that will be sacrificed on this design project because of the astronomical cost of some of them. The budget for this project is $600 from start to finish.

The potential social impact of this project is large. This project could revolutionize how PID control is taught. Allowing students to experiment with a real robot could lead to more passion towards control theory. Also, this project will demonstrate to high school students that there are interesting things in engineering and may convince more people to study engineering.

Chapter 1.4: Contributions

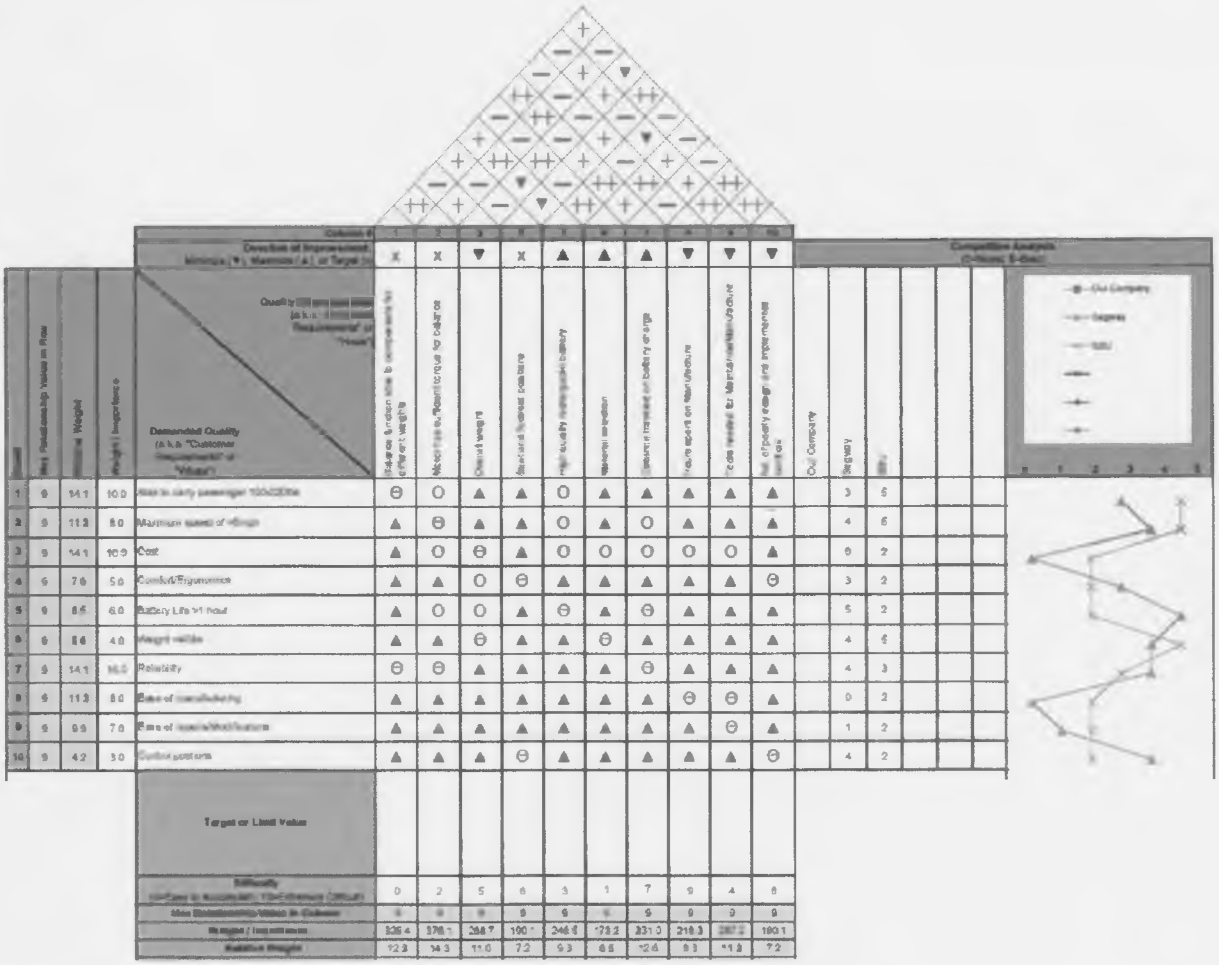
With the help of Michael Marolda in the fall semester the equations of motion for a Segway like robot were developed. Together we determined the transfer function, used Matlab to create a root locus plot and built a small scale model of Legos.

From there the development fell on my shoulders. Adjusting the code to be compatible with the Arduino UNO proved the most time consuming. Also, I had to improve the angle finding method because of the limitations of the Lego sensors. Two different angle finding techniques were investigated, and both work well on the full scale prototype. I also sourced and purchased all of the parts for the prototype. Building the prototype was fairly simple and with the help of my father we got it done in a couple of Saturdays. Testing and debugging the code took the longest. Bad connections and dead batteries were only some of the problems that I encountered throughout this build.

This paper was written and put together by me as well, that's what happens when one works alone. I enjoyed this project and it was very cool to ride my balancing robot around my house.

Chapter 2.1 Design Specifications

In order to make this project competitive economically, many of the performance specifications from the Segway Human Transporter are adopted for this project. Overall weight is targeted at being less than 40 lbs when it is operational. The battery must be rechargeable and able to power the vehicle at full power for over an hour. The top speed of the vehicle is targeted at 10 mph so that if the rider falls off, the risk of injury is low. The vehicle must be able to operate with riders ranging from 100-200 lbs. The electronics system will consist of a Arduino microcontroller, a gyro sensor, and two motors. Additionally, there must be a dead man switch so that if the rider falls off, the vehicle will not continue to try to balance as this could lead to a condition that is unsafe for the rider. Additional design constraints will come from manufacturing limitations, available materials, and the budget for the project. These specifications are reflected in the house of quality, Figure 1.



Our Company
Segway

Row #	Max Relationship Value in Row	Relative Weight	Weight / Importance	Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	1	2	3	4	5	6	7	8	9	10	Our Company	Segway	[illegible]
				Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x)	x	x	▼	x	▲	▲	▲	▼	▼	▼			
				Quality Characteristics (a.k.a. "Functional Requirements" or "Hows")	[illegible]	Motor has sufficient torque for balance	Overall weight	[illegible]	High quality rechargeable battery	[illegible]	[illegible] impact on battery charge	Hours spent on manufacture	Tools needed for [illegible]	[illegible]	Our Company	Segway	[illegible]
1	9	14.1	10.0	Able to carry passenger 100-200lbs	Θ	O	▲	▲	O	▲	▲	▲	▲	▲		3	5
2	9	11.3	8.0	Maximum speed of 10mph	▲	Θ	▲	▲	O	▲	O	▲	▲	▲		4	6
3	9	14.1	10.0	Cost	▲	O	Θ	▲	O	O	O	O	O	▲		0	2
4	9	7.0	5.0	Comfort/Ergonomics	▲	▲	O	Θ	▲	▲	▲	▲	▲	Θ		3	2
5	9	8.5	6.0	Battery Life >1 hour	▲	O	O	▲	Θ	▲	Θ	▲	▲	▲		5	2
6	6	5.6	4.0	Weight <40lbs	▲	▲	Θ	▲	▲	Θ	▲	▲	▲	▲		4	5
7	9	14.1	10.0	Reliability	Θ	Θ	▲	▲	▲	▲	Θ	▲	▲	▲		4	3
8	9	11.3	8.0	Ease of manufacturing	▲	▲	▲	▲	▲	▲	▲	Θ	Θ	▲		0	2
9	9	9.9	7.0	Ease of repairs/Modifications	▲	▲	▲	▲	▲	▲	▲	▲	Θ	▲		1	2
10	9	4.2	3.0	Control positions	▲	▲	▲	Θ	▲	▲	▲	▲	▲	Θ		4	2
				Target or Limit Value													
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)	0	2	5	6	3	1	7	9	4	6			
				Max Relationship Value in Column	9	9	9	9	9	9	9	9	9	9			
				Weight / Importance	326.4	378.1	284.7	190.[illegible]	246.8	172.2	331.0	218.3	[illegible]	190.1			
				Relative Weight	12.3	14.3	11.0	7.2	9.3	6.6	12.6	8.3	11.3	7.2			

Figure 1: House of quality for PID Control Learning Robot

Chapter 2.2 Concept Generation

The overall function that is going to be accomplished with this project is to demonstrate how a control system is affected when the controller gains are adjusted. This will help students to more fully understand control theory and help to solidify the concepts related to controller gains.

In order to accomplish this, there are many systems that must work together seamlessly. First of all, there must be a control law that uses inputs from gyro sensors and accelerometers to a microcontroller to balance the vehicle. Next, the motors will need to be reliable so that the vehicle does not just stop working while it is moving as that could be a dangerous situation. The battery must be sufficient to power both of the motors and the control system as well. Finally, the vehicle must be sturdy enough to support a large man and be built in such a way that it is ergonomic to ride.

This project was first postulated by Dr. Brianno Coller of Northern Illinois University. This project will bring together many engineering concepts. Dynamics of systems will be necessary to derive the physics of the system. System control theory will be necessary to develop the necessary equations that govern the motion of the vehicle. Using both computer programming and basic electrical engineering knowledge, the circuits and programs that are used to control the vehicle will be created. Finally, the design of the vehicle will use materials science and designing machine elements to ensure a safe, reliable vehicle.

Chapter 2.3 Evaluation

Table 1: Decision matrix for various components

	Cost	Reliability	Manufacturability	Availability of Materials	Feasibility	Compatibility	Total
Motor							
New	-1	1	1	1	-1	1	8
Used	0	0	1	0	1	0	7
Repurpose	1	0	1	0	1	0	11
Chassis							
Aluminum	0	1	1	1	0	1	16
Steel	0	1	1	1	0	1	16
Wood	1	0	1	1	1	1	18
Carbon Fiber	-1	0	-1	-1	-1	-1	-18
Electronics							
Arduino Uno	0	1	1	1	1	1	20
Texas Instruments LaunchPad	1	0	0	1	1	0	13
Weight	4	6	3	5	4	2	

In order to develop a set of criteria to build the vehicle a decision matrix Table 1, was used. The largest cost for this project is projected to be the motor and battery. These must be compatible with one another so it was determined that when deciding whether to buy new, used, or repurpose from another source that they would both come from the same place. Clearly the repurposing option is the best choice based on the above decision matrix followed by buying new. However, buying new is almost out of the question based on the budget that this project has. It would be good to avoid buying used because there is no way to test that the motor is operational before purchase.

The next decision to be made was what the chassis would be built of. The options were aluminum, steel, wood, or carbon fiber. Wood seems to be the best choice based on these criteria. Weight was determined to not be a determining factor in the decision making process, therefore steel is a close second. However, the ease with which wood can be worked with puts it at the top of the list. Additionally, since this is a prototype, being able to build multiple iterations if needed is necessary.

The electronics for the vehicle are what will separate this vehicle from others on the market. The two options are quite similar but it seems that the Arduino Uno will be a better solution. This is due in part to the fact that the Arduino has a very large community online that can be tapped into when problems arise when programming or building circuits. This will allow the code to be expanded if the customer wants to alter the vehicle to demonstrate other control theory concepts.

Overall the choices were made based on a combination of the cost, reliability, manufacturability, availability, feasibility, and compatibility with other systems.

Chapter 3.1 Cost Analysis and Patentability

The PID Control Learning Robot is a control theory problem that has been tackled numerous times. The most famous use is by Segway Inc, the manufacturer of the famous Segway PT. The Segway PT is a more refined product with numerous safety features to protect the user. The price tag on these features is large, the base model is $6500. While there are many improvements compared to the PID Control Learning Robot, there is a steep price penalty involved. The uniqueness of this project is that the balancing gains can be changed in real time. The Segway models do not allow this because they are designed to transport humans not to help teach control theory. The build price of the PID Control Learning Robot is <$600 and the performance is comparable to the Segway PT. This is very competitive even though the technology is not as advance and there are many safety features on the Segway PT that were deemed not necessary for this prototype.

If this product were to go into production, there would need to be numerous upgrades to both the chassis and the layout of parts to improve mass manufacturability. Additionally, the material choice would be different because wood is not a confidence inspiring material when used as the decking. These aspects of the new design would increase the initial cost of the robot. One aspect that would lower costs is integrating all electronics into one circuit board. This would not only simplify wiring but condense the electronics to increase available space.

It is unclear what patents Segway Inc. has on this type of robot. Most likely everything is confidential within the company; therefore there are no legal restrictions on this type of development. There are patents on the parts that Segway Inc. uses on its robots, however with a new design the parts while similar would not infringe on the current patents. A final design that is patentable would require more development and more safety features to ensure that the user is not in danger at any time.

Chapter 4.1 Balancing Theory

A balancing robot in the simplest form is an inverted pendulum. While a normal pendulum has a mass hanging from a pivot point, an inverted pendulum has a mass supported above a pivot point by a rod. A regular pendulum is a stable system as natural state is for the mass to hang downward. An inverted pendulum is inherently unstable as the mass will always tend toward a position where there is less potential energy. A simple model of an inverted pendulum is shown in Figure 2. In order to balance the pendulum in an upright position one of two things must happen. The pivot point of the pendulum must be accelerated or a torque must be applied at the pivot point. A feedback control law can be used to balance an inverted pendulum.

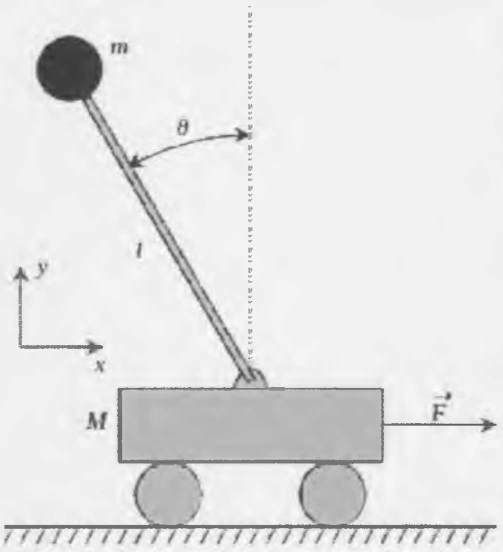


Figure 2: Simple model of an inverted pendulum

The inverted pendulum is a widely used model in dynamics and control theory. The simplest example is balancing a broomstick upright on a finger or palm. However, there are some very complex and interesting uses of this theory. NASA and other organizations use the inverted pendulum model for missile guidance when the center of pressure is above the center of gravity [4].

For a robot that moves on wheels, both concepts of balancing are needed. When the robot moves, the wheels exert a torque on the vehicle while the pivot point is accelerated. When power is applied to the motors, an equal and opposite amount of torque is applied to the wheels and the robot. This torque helps to balance the robot, but if it is ignored, the correct controller gains will be impossible to determine analytically.

Chapter 4.2 PID Control Theory

PID control is used throughout industry and hobbies for simple and complex control theory problems. PID controllers operate on feedback of the system. The current state is fed back to before the controller and added to the desired state to determine the controller output. The flow diagram of this is shown in Figure 3. PID control relies on three different gains to adjust the dynamics of the system, these gains, proportional, integral, and derivative give rise to the name of this control law. In order to have a robust controller, the first step is to understand the dynamics of the system. This is accomplished by developing a mathematical model that describes the motion of the system when there are no outside forces. The next step is to reliably measure the system or have a way of determining the current state of the system. Finally, the control law can be applied and the gains can be tuned to achieve the desired, altered dynamics of the system.

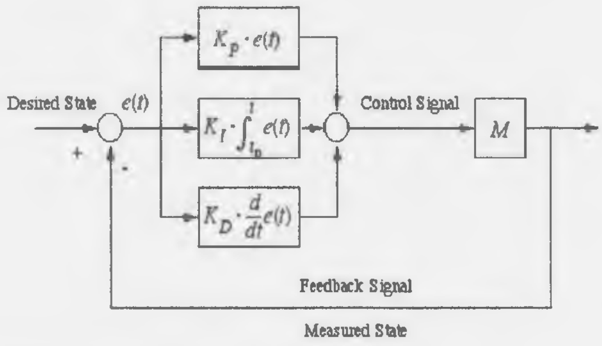


Figure 3: Block diagram of PID controller

The proportional gain of the PID controller depends on the instantaneous error of the system. For the PID Control Learning Device this is the angle of tilt from vertical. Based on this angle and the value of the proportional gain, a value is calculated. This value will be added to the values from the integral and derivative gains before being sent to the motors. The proportional gain is the largest contributor toward altering the dynamics of the system. However, a purely proportional controller usually has some sort of steady state error. The steady state error can be minimized by using an integral term.

The integral gain acts on the cumulative error of the system. This term relies on the integration of the instantaneous error over time. The integral gain attempts to correct system error that should have been corrected previously. As the vehicle is away from equilibrium, the integral term continues to increase in an attempt to regain this equilibrium point. Again, the value calculated is added to the proportional and derivative gains before being sent to the motors. Too high of an integral term results in a large overshoot which can be disastrous for a balancing robot.

The derivative gain of the PID controller acts based on the change in error. The main benefit of this gain is to anticipate the changes in the system. This helps to decrease settling time and improve controller performance by improving system stability.

These three gains working together help to solve many control theory problems. The PID Control Learning Device uses this law by measuring the angle and computing the necessary error terms to apply this type of controller. The PID controller was chosen for this project because it is a relatively simple yet powerful control law. Additionally, since PID control is very common it seems intuitive to design a learning device to incorporated a commonly used control law.

Chapter 4.3 Finding Gains

The first step in analytically finding controller gains is developing an analytical model of the entire system, Figure 5. This model is used to create the mathematical model of the system. The goal of the mathematical model is to relate the input torque, adjusted by the control law, to the angle of tilt of

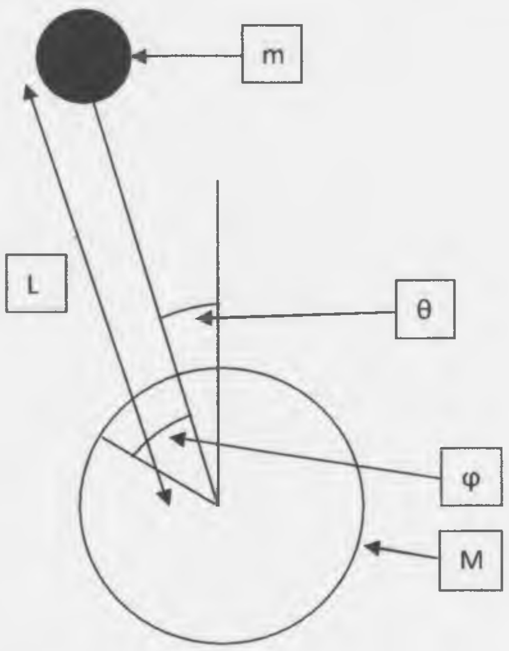


Figure 5: Analytical model of balancing robot

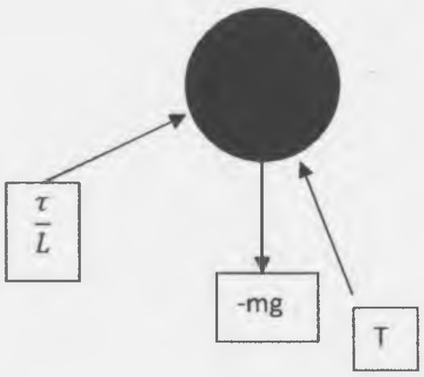


Figure 4: Free body diagram of person

the person. The mathematical model is important because it is a concise function containing the dynamics of the system. In order to generate a mathematical model of any dynamic system, the starting point is a free body diagram. For the inverted pendulum, and more specifically a self balancing robot, the starting point will be the free body diagrams of the wheel and person independently, Figure 8 and Figure 4 respectively.

The mass acceleration diagrams are also needed to find the equations of motion that govern the dynamics of the vehicle. The mass acceleration diagrams are shown in Figure 6 and Figure 7. Notice

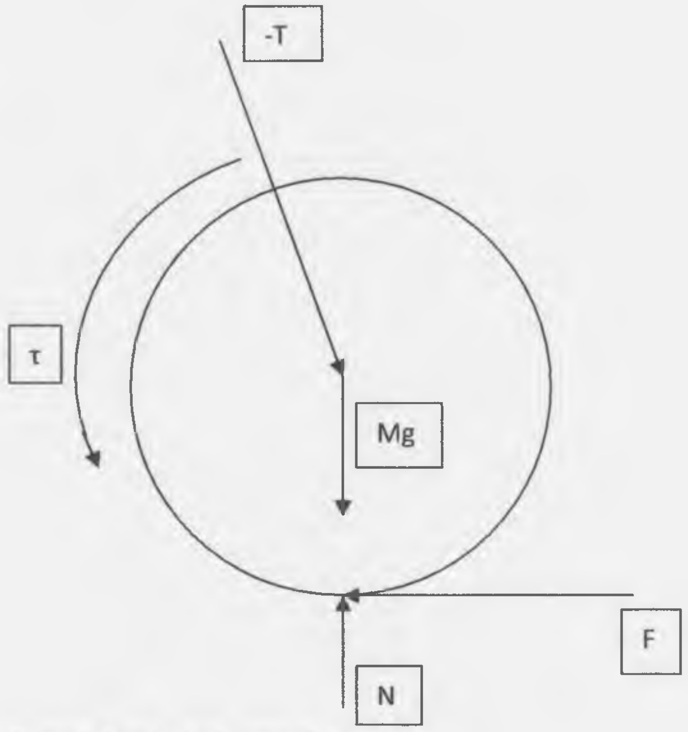


Figure 8: Free body diagram of the wheel

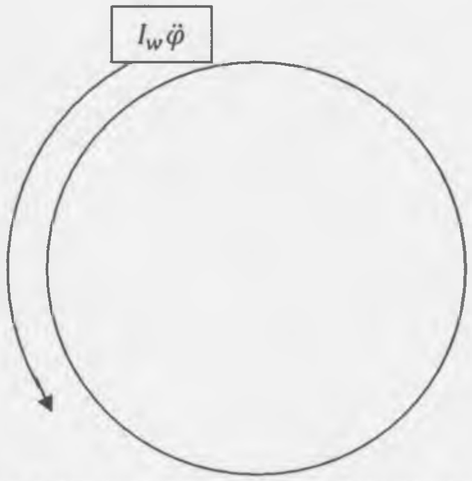


Figure 7: Mass acceleration diagram of the wheel

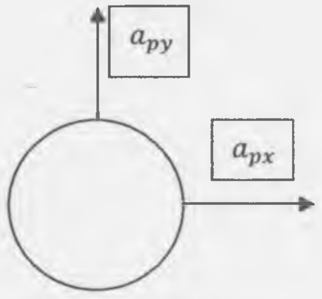


Figure 6: Mass acceleration diagram of person

that the acceleration of the wheel is in a circular coordinate system. This simplifies the equations of motion because torque is being adjusted by the control law. Also, there is no vertical acceleration of the wheel since it is assumed that the vehicle is traveling over flat land.

Using the mass acceleration diagram and free body diagram of each the person and the wheel, the equations of motion that govern how the vehicle moves can be found using Newton's second law and relating force to accelerations. The first step is to relate the accelerations of both bodies together. The control law will be acting on the wheel, therefore the acceleration of the person will be related to the base by Equation 1. Since the person is constrained to move in a circle about the base, $a_{p/b}$ can be expanded, resulting in Equation 2. This is further broken down using trigonometry into the two dimensions of interest, the *i* and *j* directions, Equation 3. The equations of the person and the wheel, given by Equation 4 through Equation 7, are related through *T*, or the tension in the rod connecting the two bodies.

Equation 1

$$a_p = a_b + a_{p/b}$$

Equation 2

$$a_p = a_b + L\ddot{\theta}\widehat{e_\theta} - L\dot{\theta}^2\widehat{e_r}$$

Equation 3

$$a_p = \left[a_b - L\ddot{\theta}\cos(\theta) + L\dot{\theta}^2\sin(\theta)\right]i + \left[-L\ddot{\theta}\sin(\theta) - L\dot{\theta}^2\cos(\theta)\right]j$$

Equations of motion – Person

Equation 4: Sum of forces in i direction

$$-Tsin(\theta) + \frac{\tau}{L}\cos(\theta) = m\left[a_b - L\ddot{\theta}\cos(\theta) + L\dot{\theta}^2\sin(\theta)\right]$$

Equation 5: Sum of forces in j direction

$$Tcos(\theta) + \frac{\tau}{L}\sin(\theta) - mg = m[-L\ddot{\theta}\sin(\theta) - L\dot{\theta}^2\cos(\theta)]$$

Equations of motion – Wheel

Equation 6: Sum of forces in i direction

$$Tsin(\theta) - F = Ma_b$$

Equation 7; Sum of forces in k direction

$$\tau - FR = I_w\ddot{\varphi}$$

Equation 8

$$\ddot{\varphi} = \frac{\ddot{x}}{R}$$

Equation 9

$$F = \frac{\tau}{R} - \frac{I_w}{R}\ddot{x}$$

Equation 10: Sum of forces in i direction

$$Tsin(\theta) - \frac{\tau}{R} - \frac{I_w}{R}\ddot{x} = M\ddot{x}$$

The angular acceleration of the wheel can be directly related to the acceleration of the base by Equation 8. This allows a substitution for *F* in Equation 6 resulting in Equation 10. Equation 4 and Equation 10 can be added together thus eliminating the last unknown, the tension in the rod.

After some basic algebra a two transfer functions can be identified, one for the *i* direction and one for the *j* direction given by Equation 11 and Equation 12 respectively. These equations will be referenced as TF1 and TF2. Clearly the applied torque has a significant effect in both of the transfer functions. This is intuitive because if torque did not play a role, balancing this robot would be impossible.

Equation 11: Transfer function 1

$$\tau - mgLsin(\theta) = mL\ddot{x}\cos(\theta) - mL^2\ddot{\theta}$$

Equation 12: Transfer function 2

$$\left(m + M + \frac{I_w}{R^2}\right)\ddot{x} + \tau\left(\frac{1}{R} - \frac{\cos(\theta)}{L}\right) = -mL\ddot{\theta}\cos(\theta) + L\dot{\theta}^2\sin(\theta)$$

The next step in finding the open loop transfer function is to linearize the two transfer functions. Assuming that the angle of tilt remains fairly small, sin(θ) can be approximated as θ. Similarly cos(θ) is assumed to be 1. The final step in linearization is to assume that all derivative terms that are raised to a power are negligible and approximated as 0. These steps result in Equation 13 and Equation 14 for TF1 and TF2 respectively.

Equation 13: Linearized transfer function 1

$$\tau - mgL\theta = mL\ddot{x} - mL^2\ddot{\theta}$$

Equation 14: Linearized transfer function 2

$$\left(m + M + \frac{I_w}{R^2}\right)\ddot{x} + \tau\left(\frac{1}{R} - \frac{1}{L}\right) = -mL\ddot{\theta}$$

The end is in sight, from here, the two transfer functions are converted to the Laplace Domain. The Laplace Domain is where most control law is focused because derivatives are noted as simply *s* rather than mathematical statements. Once in the Laplace domain, the equations are rearranged to isolate a variable, θ for TF1 and X for TF2. These steps are shown in Equation 15 through Equation 18. TF2 can then be substituted into TF1, resulting in the open loop transfer function of the system, Equation 19.

Equation 15: Transfer function 1 in Laplace domain

$$\tau - mgL\Theta = mLXs^2 - mL^2\Theta s^2$$

Equation 16: Transfer function 2 in Laplace domain

$$\left(m + M + \frac{I_w}{R^2}\right)Xs^2 + \tau\left(\frac{1}{R} - \frac{1}{L}\right) = -mL\Theta s^2$$

Rearrange

Equation 17: Transfer function 1

$$\Theta = \frac{mLs^2X - \tau}{mL^2s^2 - mgL}$$

Equation 18: Transfer function 2

$$X = \frac{-mLs^2\Theta - \tau\left(\frac{1}{R} - \frac{1}{L}\right)}{\left(m + M + \frac{I_w}{R^2}\right)s^2}$$

Open Loop Transfer Function (Referenced as T in future equations or N/D)

Equation 19: Open loop transfer function, referenced as T in future equations

$$\Theta = \frac{\left[-mL\left(\frac{1}{R} - \frac{1}{L}\right) - \left(m + M + \frac{I_w}{R^2}\right)\right]s^2}{\left[mL^2s^4 + \left(\frac{m^2L^2}{\left(m + M + \frac{I_w}{R^2}\right)} - mgL\right)s^2\right]}\tau = \mathrm{T}\tau$$

The open loop transfer function is necessary for finding the analytical gains. The easiest method is a root locus plot. For the root locus of this system, MatLab's SISOTool was used. By plugging in the above open loop transfer function, the theoretical gains can be found. A plot of the root locus used to find these gains is shown in Figure 9. The root locus for a PID controller consists of two open loop poles from the open loop transfer function, shown by blue crosses, two open loop zeros, the red circles and one more open loop pole, shown by the red cross. The latter three components are adjustable by the user and are used to control the shape of the root locus plot. The pink squares represent the gain of the loop. Obviously different orientations of the loop and squares give different control characteristics. The PID Control Learning Robot helps to demonstrate how changing the control characteristics affects the overall system.

The last step involving the gains is implementing them into a control law. For this a close loop transfer function is needed. The closed loop transfer function relates the current angle of tilt to the desired angle of tilt and the torque applied at the pivot point. The closed loop transfer function is given in Equation 20 with a more useful, rearranged transfer function given by Equation 21.

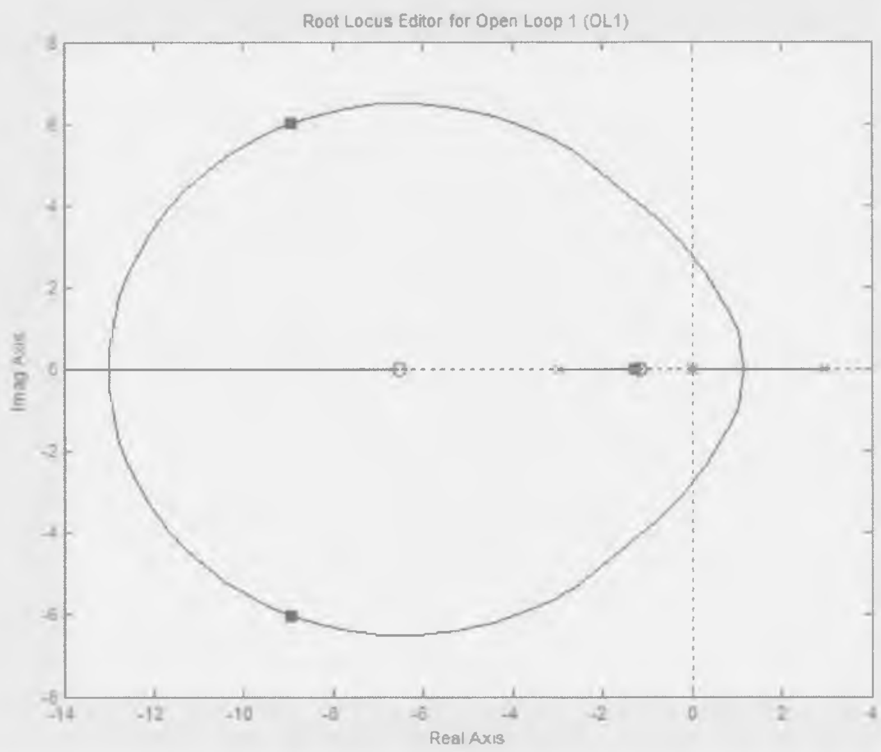


Figure 9: Root locus plot for balancing robot

Equation 20: Closed loop transfer function

$$\Theta = \mathrm{T}\tau\left(k_d s + k_p + \frac{1}{s}k_i\right)(\Theta_{des} - \Theta)$$

Equation 21: Rearranged closed loop transfer function

$$\Theta = \frac{\mathrm{T}\tau\left(k_d s + k_p + \frac{1}{s}k_i\right)}{1 + \mathrm{T}\tau\left(k_d s + k_p + \frac{1}{s}k_i\right)}\Theta_{des}$$

This means that if an angle of tilt and rate of rotation can be measured, the control law will be able to adjust the power of the wheels to the motor based on these ever changing parameters. With properly tuned gains and a reliable algorithm, the inverted pendulum will no longer be an unstable system.

Chapter 5.1 Components

The brain of the robot is an Arduino UNO microcontroller board. There are numerous benefits of using this platform. The Arduino brand has grown into the standard for prototyping electronics projects. This is in part due to the easy to use interface that has been developed. Based on C++ language, coding the Arduino UNO is straightforward and intuitive. Additionally, there is a multitude of information in various forums on the internet with regards to troubleshooting, debugging, and general coding advice.

The nervous system of the PID Control Learning Robot is the internal measurement unit, or IMU. In order to build a robust controller, there must be a reliable way of finding the angle that the vehicle is at in real time. Two sensors were used in conjunction to find the angle, a gyro sensor to measure the rate of angular rotation and an accelerometer to approximate the actual angle. These two sensors must be oriented parallel to one another so that the data can be combined into one estimated angle. For this project an IMU was used so that the sensors are orientated perfectly with respect to one another.

The IMU used for this project is an MPU-6050 triple axis breakout board. This product was chosen for many reasons. For the price, this IMU has many features that are beneficial for this project. First and foremost the sensor range for the gyro and accelerometers are ±250 dps and ±2g respectively. These ranges are sufficient for this project. While it was not necessary the full-scale range can be increased for both sensors. Increasing the range however lowers the resolution and will thus lower the accuracy of the angle calculation.

Two angle finding algorithms were investigated for the PID Control Learning Robot, a complimentary filter and a Kalman filter. The complimentary filter uses a ratio of the two measurements to calculate the angle of the vehicle. By applying a ratio to both the sensor measurements, a low-pass filter is applied to the accelerometer data and a high-pass filter is applied to the gyro sensor. The low-pass filter only allows long term changes to be seen in the data. This allows the accelerometer to be used to measure angle even though there will be variations due to the movement of the vehicle. The high-pass filter on the gyro sensor is necessary to compensate for the drift of the gyro sensor. While at rest, the gyro sensor will give consistently varying readings due to sensor drift present in all gyro sensors. A high-pass filter allows short term changes to affect the final angle calculation while filtering out the trends that are steady. Figure 10 shows the final combination of the sensors along with the raw sensor data.

The complimentary filter is very easy to put into code. Shown in Equation 22 is the code form of a complimentary filter.

Equation 22

$$\theta_i = ((1-\alpha) * (\theta_{i-1} + GyroReading * \Delta t) + \alpha * AccelReading$$

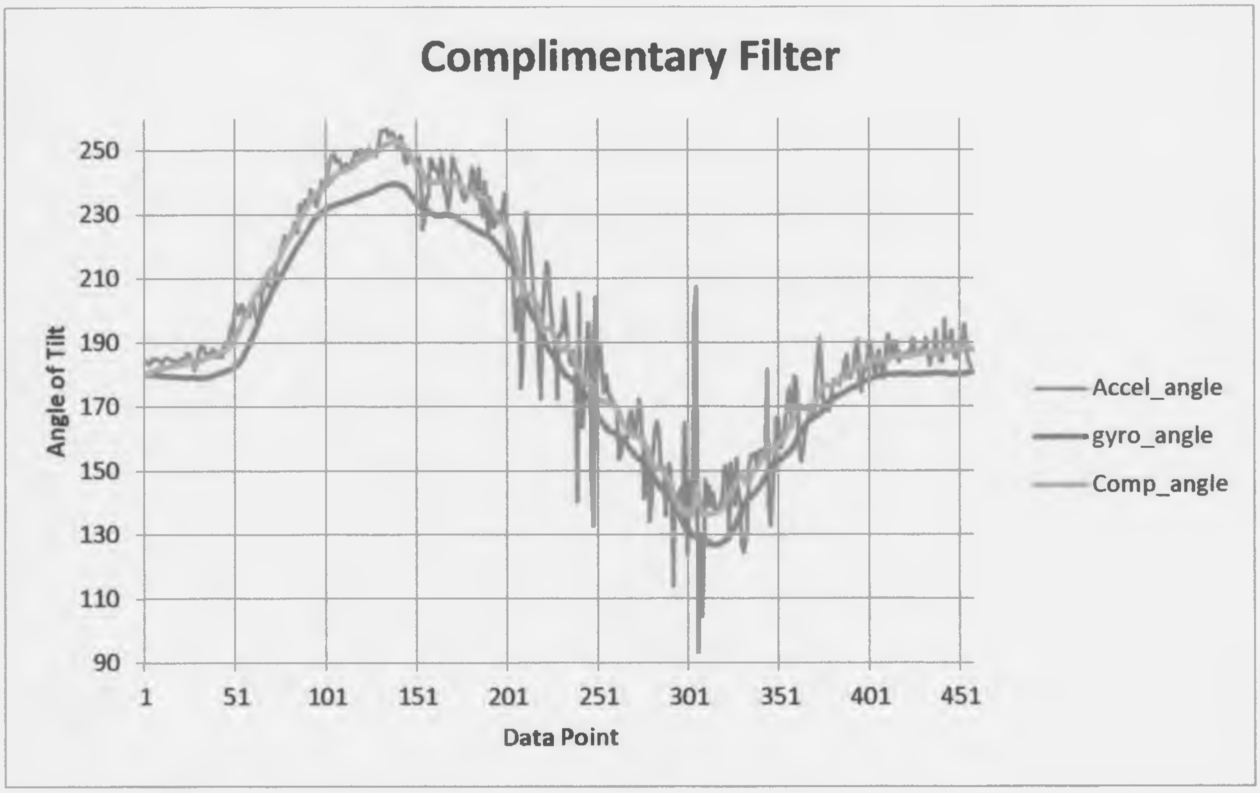


Figure 10: Combination of signals using a complimentary filter

A Kalman filter uses a more complex method of combining the sensor data. By using a recursive algorithm, the Kalman filter can predict the past, present, and future states of a system. Welch discusses more fully the aspects of a Kalman filter along with a derivation of the algorithm. [5] Figure 11 shows how the data is combined with a Kalman filter.

Figure 12 demonstrates that both filters work well. The Kalman filter generates a slightly smoother curve. The complimentary filter is more of a brute force filter while the Kalman offers a more elegant calculation. While the robot will operate properly with a complimentary filter a Kalman filter will be used for the final project because of the smoother curve. For this project the smooth angle curve is important to ensure that the drive of the robot is not jumpy and uncomfortable for the user. Also, a smoother motor drive will lead to a more confident rider.

The muscle of the vehicle comes from two 350W motors and two 12V batteries. The motors are controlled by a Sabertooth 2x25 motor driver. These motors were chosen because they have an internal gearbox that eliminates the need for a chain drive that would add complexity to the final product. Also, these motors are capable of handling 24V, which is supplied by the two 12V batteries wired in series.

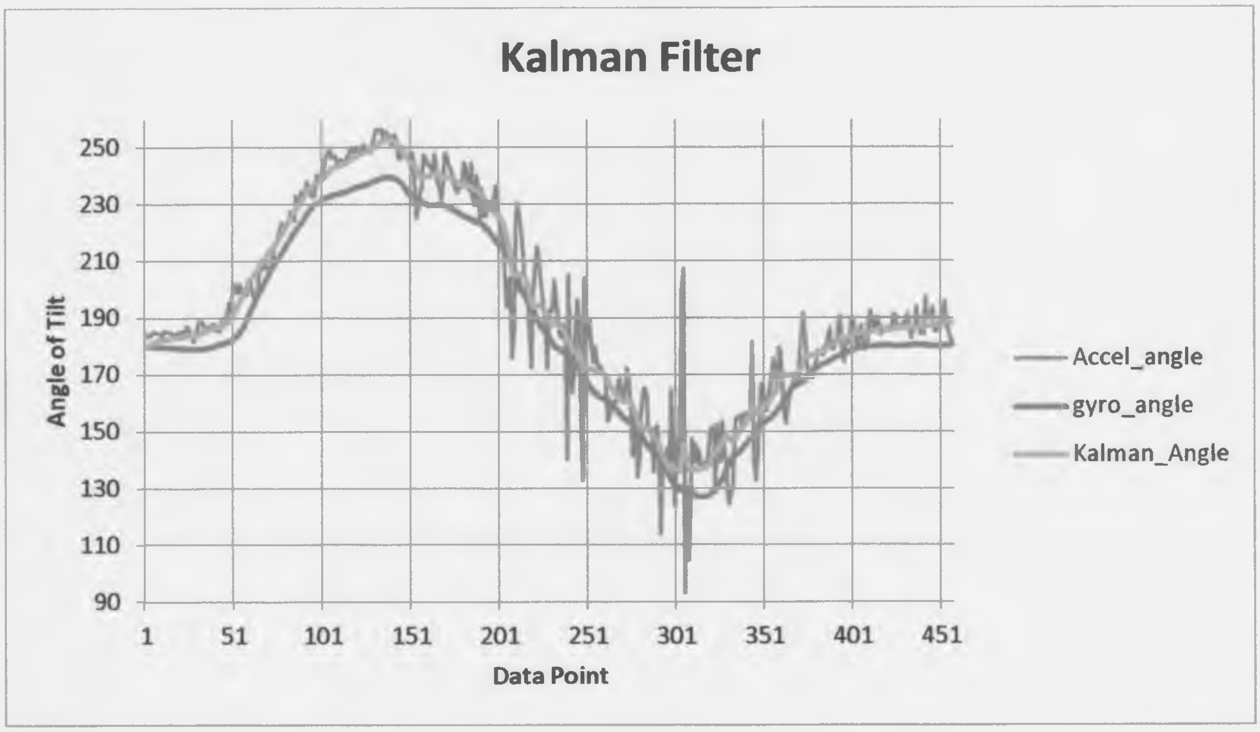


Figure 11: Combination of signals using a Kalman filter

Four 10K Ohm potentiometers are used on the robot for user input. Three of the potentiometers adjust the proportional, integral, and derivative gains of the robot. The fourth potentiometer is used for steering. There is also a push button switch on the handle bars so that if the rider gets into trouble, he or she can simply let off the button and power is no longer sent to the motors.

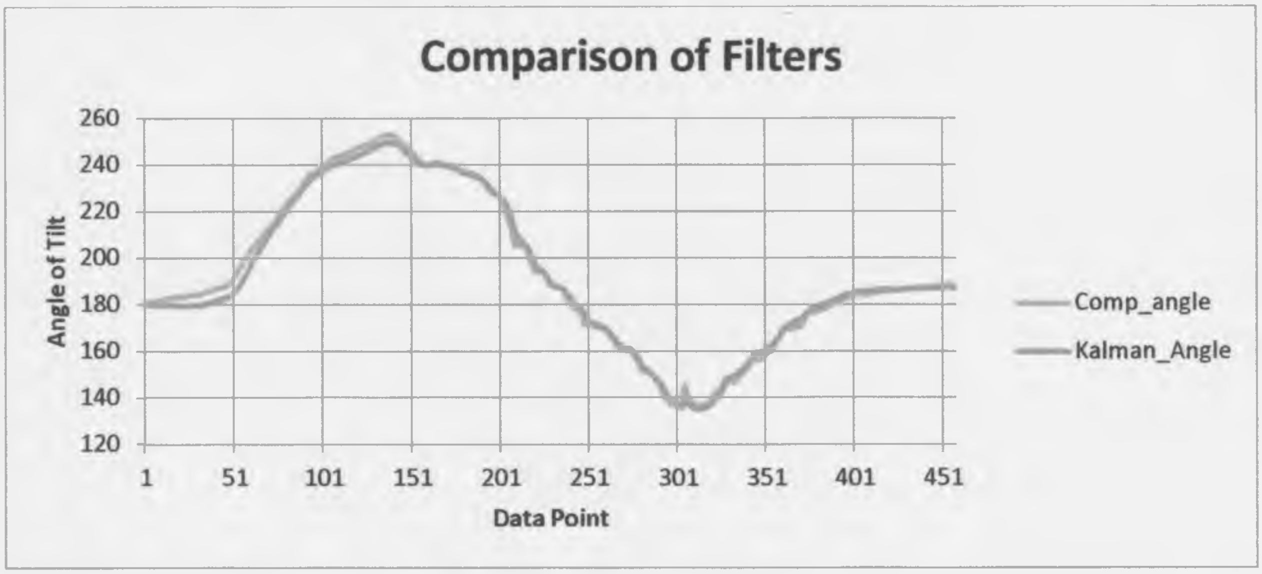


Figure 12: Comparison of Complimentary and Kalman filters on same data set

Chapter 6.1 Testing

Testing for this project consisted of empirical test relying on human senses. In order to determine that the algorithm worked correctly and reliably, the gains were adjusted manually using the potentiometers until a reliable balancing controller was found. This is far from a perfect test because humans automatically balance themselves and therefore impose another degree of freedom on the system. Even with this shortcoming, this test allowed testing of the coding and to understand how each gain affects the system.

The next step was to test the mathematical model. A correct mathematical model will allow an engineer to reliably choose PID controller gains based solely on the static properties of the system, mass of the person, mass of the wheel, moment of inertia, along with other parameters based on the materials and geometry of the robot. The model was tested by using gains determined through the root locus plot to try to balance the robot. It was found that these gains did balance the robot, although they were not tuned as finely as the gains found experimentally.

The testing method was simply to ride the robot. Unfortunately, there is no way of gathering data from the Arduino UNO while the robot is operational. The Arduino UNO does allow the user to read values through a serial port when a computer is connected. Therefore, testing of the individual sensors was conducted before they were installed in the robot to develop the code and ensure proper operation. This testing involved data collection and analysis to determine how to change the code for correct measurements. This analysis is shown in Chapter 5.1.

To test the robot, the robot was simply ridden and the gains were adjusted. As expected, when the proportional gain is increased, the vehicle begins to oscillate about the balance point due to small variations in the calculated angle. When the proportional gain is decreased, the robot falls over because the wheels do not have enough speed to keep the vehicle from falling. By increasing the integral term, more oscillating occurs before becoming stationary and decreasing the integral term results in a robot that is slower to react. The derivative term when increased caused the system to become very unstable due to the increase of power when the robot is rotating. Decreasing the derivative term made the robot slower to reach a balance point. However it did not cause more oscillating.

Due to the shortcomings of the Arduino UNO the testing of the full scale robot is purely opinion. This makes it impossible to prove that the vehicle is operating in the manner it was designed. In theory, when the gains are changed, the response of the system should also change. To eliminate bias from these findings the gains were tuned high and low for someone who is not familiar with PID control and the rider was asked for a description with relation to the first set of gains that they were given to ride.

One of the problems with testing the balancing of the vehicle is that the human body automatically balances itself. Different people have different capacities for balance, for example, a gymnast who walks on a balance beam has much better balance than a child. This means that a specific set of gains may work for one person and not for another if the latter's skill of balance is less refined. To try to counteract this, a person with bad balance was chosen to test the final product and tune the final gains. If a person with even worse balance can still not ride the robot, the gains can be adjusted dynamically to hopefully find a balancing algorithm for every user.

The theoretical gains along with the experimental findings are shown in Table 2. It is clear that the mathematical model closely reflects the actual dynamics of the system. One aspect that could contribute to the variation is that the motor controller operates on an 8-bit scale. This means that the value at full stop is 0, and full forward is 127. For the PID calculations it is assumed that the scale is from 0 to 100. The theoretical gains did provide balance, so to save computing space on the microcontroller another calculation was deemed as unnecessary.

Table 2: PID gain values

	Gain Values		
	P	I	D
Calculated	6.02	8.62	0.11
Experimental	6.96	8.0	0.32

With a little bit of balance skill from the rider many different gains were able to be ridden. These gains had the smoothest feel to them and allowed a degree of confidence in the robot.

Chapter 7.1 Discussion and Conclusion

The aim of this project was to create a product that allowed the user to investigate the effect that PID controller gains have on a simple system. The focus of this investigation is an inverted pendulum and more specifically a self balancing robot. Perhaps the most difficult aspect of control theory is how to tune the gains of a system to improve the performance once the required task is reliably completed. Hopefully this project lends some insight as to how each gain affects the dynamic response of the system.

There were many difficulties involved with this project. Initially the dynamics of the system had to be investigated and a mathematical model had to be created. From there, a small scale test and proof of concept was pursued. This also helped to understand how the code had to operate and many iterations of code were used to improve both the angle finding and validate the mathematical model. Next, the correct equipment for the full size prototype had to be found at a low price. In order to have a working prototype many different parts had to be compatible. These parts included the sensors, microcontroller, motor controller, batteries, and motors. Additionally, the specifications of these parts had to be sufficient to carry a human being.

The coding of the full size prototype turned out to be the most difficult part of the entire project. The basic algorithm is common throughout PID controllers but in order to have the PID algorithm mesh smoothly with the sensors and vehicle a lot of trial and error was needed. The approach taken was to create programs to make sure every sensor operated correctly and then integrate the programs together. The Arduino UNO allows the user to monitor the program through a serial port. This serial connection can be programmed to send any variable the user wants. The individual sensor programs are contained in Appendix I.

To test the prototype initially, the potentiometers corresponding to the gains were adjusted until a reliable balance was found. This is a way of determining gains experimentally. This test also served to confirm that the algorithm worked properly and reliably. The second test was to validate the mathematical model. This was accomplished by using values gathered from the root locus plot. The final results are purely opinion because the Arduino UNO has no onboard storage and it is unsafe to have a laptop connected while the vehicle is in operation.

In conclusion, this project was a success. The effects that the gains of a PID controller have on a simple system were demonstrated. Hopefully this project will allow future students to more fully understand not only the capabilities of a control system but how to more effectively tune a control law in order to achieve the performance desired. A full scale prototype was constructed that can be ridden forward and backward and turned left and right. The mathematical model was validated and can be used for future projects. Also, the intricacies of control theory and how to apply a control law were necessary for this project to be successful. This is a very intriguing project and the final product is not only a fun toy but also a marvel of control theory. Hopefully this project will increase the desire to learn control theory.

References

[1] Bageant, Maia R. "Balancing a Two Wheeled Segway Robot." *MIT*. N.p., 6 May 2011. Web. 15 Feb. 2013.

[2] Chi Ooi, Rich. *Balancing a Two-Wheeled Autonomous Robot*. *University of West Alabama*. N.p., 2003. Web. 4 Apr. 2013.

[3] Musser, Nick. "Modeling and Control of a Segway." *NickMusser.Weebly*. N.p., 8 June 2010. Web. 3 Feb. 2013.

[4] "Rocket Stability." *Rocket Stability*. Ed. Tom Benson. N.p., n.d. Web. 4 Apr. 2013.

[5] Welch, Greg, and Gary Bishop. *An Introduction to the Kalman Filter*. Tech. University of North Carolina at Chapel Hill, 24 July 2006. Web. 6 Apr. 2013.

Appendix I: Test Functions

Angle Test:

```
#include <Wire.h>

#include "Kalman.h"

Kalman kalmanX;

//Kalman kalmanY;

uint8_t IMUAddress = 0x68;


/* IMU Data */

int16_t accX;

int16_t accY;

int16_t accZ;

int16_t tempRaw;

int16_t gyroX;

int16_t gyroY;

int16_t gyroZ;


double accXangle; // Angle calculate using the accelerometer

double accYangle;

double temp;

double gyroXangle = 180; // Angle calculate using the gyro

//double gyroYangle = 180;

double compAngleX = 180; // Calculate the angle using a Kalman filter

//double compAngleY = 180;

double kalAngleX; // Calculate the angle using a Kalman filter

//double kalAngleY
```

```
uint32_t timer;


void setup() {
 Serial.begin(38400);
 Wire.begin();
 i2cWrite(0x6B,0x00); // Disable sleep mode
 if(i2cRead(0x75,1)[0] != 0x68) { // Read "WHO_AM_I" register
  Serial.print(F("MPU-6050 with address 0x"));
  Serial.print(IMUAddress,HEX);
  Serial.println(F(" is not connected"));
  while(1);
 }
 kalmanX.setAngle(180); // Set starting angle
 //kalmanY.setAngle(180);
 Serial.println("Accel_angle \t gyro_angle \t Comp_angle \t Kalman_Angle");
 timer = micros();
}


void loop() {
 /* Update all the values */
 uint8_t* data = i2cRead(0x3B,14);
 accX = ((data[0] << 8) | data[1]);
 accY = ((data[2] << 8) | data[3]);
 accZ = ((data[4] << 8) | data[5]);
 tempRaw = ((data[6] << 8) | data[7]);
 gyroX = ((data[8] << 8) | data[9]);
```

```
  //gyroY = ((data[10] << 8) | data[11]);

  //gyroZ = ((data[12] << 8) | data[13]);

  /* Calculate the angls based on the different sensors and algorithm */

  accYangle = (atan2(accX,accZ)+PI)*RAD_TO_DEG;

  accXangle = (atan2(accY,accZ)+PI)*RAD_TO_DEG;

  double gyroXrate = (double)gyroX/131.0;

  //double gyroYrate = -((double)gyroY/131.0);

  //gyroXangle += gyroXrate*((double)(micros()-timer)/1000000); // Calculate gyro angle without any
filter

  //gyroYangle += gyroYrate*((double)(micros()-timer)/1000000);

  gyroXangle += kalmanX.getRate()*((double)(micros()-timer)/1000000); // Calculate gyro angle using the
unbiased rate

  //gyroYangle += kalmanY.getRate()*((double)(micros()-timer)/1000000);

  compAngleX = (0.93*(compAngleX+(gyroXrate*(double)(micros()-timer)/1000000)))+(0.07*accXangle);
// Calculate the angle using a Complimentary filter

  //compAngleY = (0.93*(compAngleY+(gyroYrate*(double)(micros()-
timer)/1000000)))+(0.07*accYangle);

  kalAngleX = kalmanX.getAngle(accXangle, gyroXrate, (double)(micros()-timer)/1000000); // Calculate
the angle using a Kalman filter

  //kalAngleY = kalmanY.getAngle(accYangle, gyroYrate, (double)(micros()-timer)/1000000);

  timer = micros();

  temp = ((double)tempRaw + 12412.0) / 340.0;

  /* Print Data */

  /*

Serial.print(accX);Serial.print("\t");

Serial.print(accY);Serial.print("\t");

Serial.print(accZ);Serial.print("\t");

Serial.print(gyroX);Serial.print("\t");
Serial.print(gyroY); Serial.print("\t");

Serial.print(gyroZ);Serial.print("\t");

*/

 //Serial.print(accXangle);Serial.print("\t");

 Serial.print(accXangle);Serial.print("\t");

 Serial.print(gyroXangle);Serial.print("\t");

 //Serial.print(gyroYangle);Serial.print("\t");

 Serial.print(compAngleX);Serial.print("\t");

 //Serial.print(compAngleY); Serial.print("\t");

 Serial.print(kalAngleX);Serial.println("\t");

 //Serial.print(kalAngleY);Serial.print("\t");

 //Serial.print(temp);Serial.print("\t");

 //Serial.print("\n");

  delay(1); // The accelerometer's maximum samples rate is 1kHz

}

void i2cWrite(uint8_t registerAddress, uint8_t data){

 Wire.beginTransmission(IMUAddress);

 Wire.write(registerAddress);

 Wire.write(data);

 Wire.endTransmission(); // Send stop

}

uint8_t* i2cRead(uint8_t registerAddress, uint8_t nbytes) {

 uint8_t data[nbytes];

 Wire.beginTransmission(IMUAddress);

 Wire.write(registerAddress);

 Wire.endTransmission(false); // Don't release the bus
 Wire.requestFrom(IMUAddress, nbytes); // Send a repeated start and then release the bus after
reading
 for(uint8_t i = 0; i < nbytes; i++)
  data[i] = Wire.read();
 return data;
}
```

Dead man Test:

```
#define BUTTON_PIN      10  // Button
#define DELAY           20  // Delay per loop in ms
void setup()
{
 pinMode(BUTTON_PIN, INPUT);
 digitalWrite(BUTTON_PIN, HIGH); // pull-up
 Serial.begin(9600);
}
boolean handle_button()
{
 int button_pressed = !digitalRead(BUTTON_PIN); // pin low -> pressed
 return button_pressed;
}


void loop()
{
 // handle button
 boolean button_pressed = handle_button();
 // do other things
 if (button_pressed == 1)

 Serial.println(button_pressed);

 // add newline sometimes

 /*static int counter = 0;

 if ((++counter & 0x3f) == 0)

   Serial.println();*/

 delay(DELAY);

}
```

Potentiometer Test:

```
int potPin = 0;    // select the input pin for the potentiometer

int ledPin = 13;   // select the pin for the LED

int val = 0;       // variable to store the value coming from the sensor


void setup() {

 pinMode(ledPin, OUTPUT);  // declare the ledPin as an OUTPUT

 Serial.begin(38400);

}


void loop() {

 val = analogRead(potPin);    // read the value from the sensor

            // stop the program for some time

 Serial.println(val);

}
```

Motor Controller Test:

```
#include <SoftwareSerial.h>
```

```
#include <Sabertooth.h>

SoftwareSerial SWSerial(2, 3); // RX on pin 2 (unused), TX on pin 3 (to S1).

Sabertooth ST(128, SWSerial); // Address 128, and use SWSerial as the serial port.

void setup()

{

 SWSerial.begin(9600);

 ST.autobaud();

}

void loop()

{

 int power;

 // Ramp from -127 to 127 (full reverse to full forward), waiting 20 ms (1/50th of a second) per value.

 for (power = -127; power <= 127; power ++)

 {

  ST.motor(1, power);

  //ST.motor(2, power);

  delay(20);

  Serial.print(power);

 }

 // Now go back the way we came.

 for (power = 127; power >= -127; power --)

 {

  ST.motor(1, power);

  //ST.motor(2, power);

  delay(20);

  Serial.print(power); }}
```

Balance Function:

```
//Initialize all sensors

#include <Wire.h>

#include "Kalman.h"

#include <SabertoothSimplified.h>

#include <SoftwareSerial.h>


Kalman kalmanX;

SoftwareSerial SWSerial(2,3);

SabertoothSimplified ST(SWSerial);

//Kalman kalmanY;

uint8_t IMUAddress = 0x68;

/* IMU Data */

int16_t accX;

int16_t accY;

int16_t accZ;

int16_t tempRaw;

int16_t gyroX;

int16_t gyroY;

int16_t gyroZ;

#define Dead_man 10


const float INIT_ANGLE = 0.0;

const int ANGLE_BUFFER_LENGTH = 1;

float GYRO_X_INIT_VAL = 425.0;
float angleBuffer[ANGLE_BUFFER_LENGTH];

int angleBufferIndex = 0;

const float SumErrMax = 10;

const float SumErrMin = -10;

const int analogInPin1 = 3; //Analog input pin that pot is attached to

const int analogInPin2 = 2;

const int analogInPin3 = 1;

int potValue1 = 0;

int potValue2 = 0;

int potValue3 = 0;

float outputValue1 = 0; //kpt

float outputValue2 = 0; //dit

float outputValue3 = 0; //kdt

const int steering = 0;

int steeringValue = 460;

float turnRate = 0;

float throttle = 0.0;

float leftthrottle = 0.0;

float softStartCounter = 0;

float leftsoftStartCounter = 0;

double accXangle; // Angle calculate using the accelerometer

double temp;

double gyroXangle = 180; // Angle calculate using the gyro

double compAngleX = 180; // Calculate the angle using a Kalman filter

double kalAngleX; // Calculate the angle using a Kalman filter
```

```
float elapsedTimeSec = 0.0;

unsigned long prevTime = 0, currTime = 0, elapsedTime = 0;

float curErr = 0, prevErr = 0, sumErr = 0;

float integralTerm = 0, derivativeTerm = 0;

double To_motor;

double offset;

double thetaDot;

uint32_t timer;

boolean isFirstTime = true;

float CalGyro() {

 uint8_t* data = i2cRead(0x3B,14);

 gyroX = ((data[8] << 8) | data[9]);

 float val = 0.0;

 delay(500);

 for (int i = 0; i < 25; i++) {

  val += gyroX;

  delay(10);

 }

 val /= 25.0;

 return val;

}

float GetAvgAngle() {

 int count = 0;

 float a = 0;

 for (int i = 0; i < ANGLE_BUFFER_LENGTH; i++) {
    if (angleBuffer[i] != 0) {
      count++;
      a += angleBuffer[i];
    }
  }
  if (count > 0)
    return a / float(count);
  else
      return 0.0;
}
float GetAccAngle()
{
  /* Update all the values from sensors */
  uint8_t* data = i2cRead(0x3B,14);
  accY = ((data[2] << 8) | data[3]);
  accZ = ((data[4] << 8) | data[5]);
  accXangle = (atan2(accY,-accZ))*RAD_TO_DEG;
  return accXangle;
}
float KalmanAngle()
{
  uint8_t* data = i2cRead(0x3B,14);
  gyroX = ((data[8] << 8) | data[9]);
  currTime = micros();
  elapsedTime = currTime - prevTime;
  prevTime = currTime;
  elapsedTimeSec = (float) elapsedTime/ 1e6f;
  float angle = GetAccAngle();
  float gyroXrate = -((float)gyroX-GYRO_X_INIT_VAL)/131.0;
  if (isFirstTime){
    kalAngleX = angle;
    isFirstTime = false;
  }
  else
  {
    kalAngleX = kalmanX.getAngle(angle, gyroXrate, elapsedTimeSec);
  }
  //prevAngle = currAngle  //can use for complimentary filter
  return kalAngleX;
}
boolean handle_button()
{
  int button_pressed = !digitalRead(Dead_man); // pin low -> pressed
  return button_pressed;
}
void sendToMotor(float ang, float To_motor_L, float To_motor_R)
{
  boolean deadManPressed = handle_button();
  if (deadManPressed == 1){
  throttle = To_motor_R;
  if ( softStartCounter < 1000 ) {    //Start the motor slow on power up
    softStartCounter++;
```

```
 throttle *= ( softStartCounter / 1000 );

}

if (throttle > 127)

throttle = 127;

else if (throttle < -127)

throttle = -127;

if (abs(ang) > 20.0)

 ST.motor(1, 0);

else

 ST.motor(1, throttle);

// Left motor speed

leftthrottle = To_motor_L;

if ( leftsoftStartCounter < 1000 ) {        //Start the motor slow on power up

 leftsoftStartCounter++;

 leftthrottle *= ( leftsoftStartCounter / 1000 );

}

if (leftthrottle > 127)

leftthrottle = 127;

else if (leftthrottle < -127)

leftthrottle = -127;

if(abs(ang) > 20.0)

 ST.motor(2,0);

else

 ST.motor(2, leftthrottle);

}

else
```

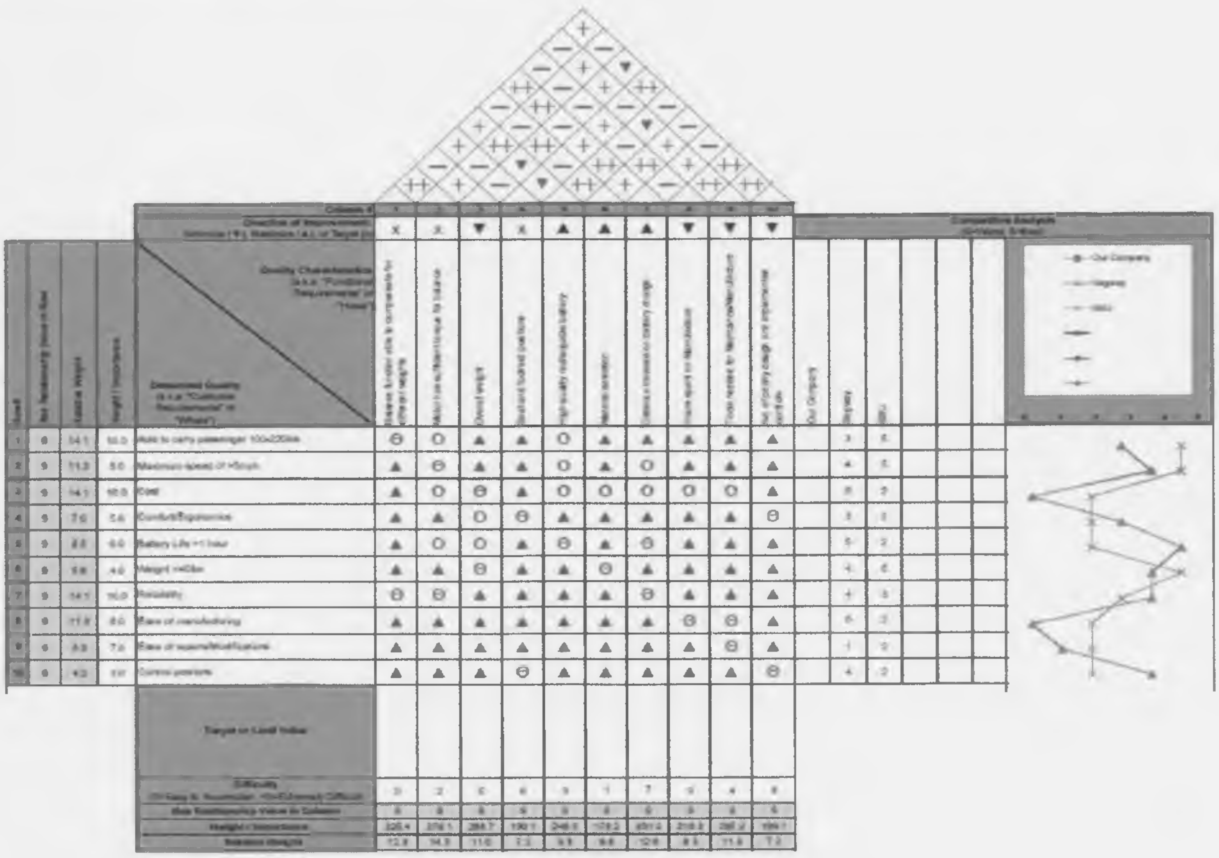
```
 {
   ST.motor(1,0);
   ST.motor(2,0);
 }
}
void setup() {
 Serial.begin(38400);
 SWSerial.begin(38400);
 Wire.begin();
 pinMode(Dead_man, INPUT);
 digitalWrite(Dead_man, HIGH);
 i2cWrite(0x6B,0x00); // Disable sleep mode
 if(i2cRead(0x75,1)[0] != 0x68) { // Read "WHO_AM_I" register
   Serial.print(F("MPU-6050 with address 0x"));
   Serial.print(IMUAddress,HEX);
   Serial.println(F(" is not connected"));
   while(1);
 }
 kalmanX.setAngle(180); // Set starting angle
 GYRO_X_INIT_VAL = CalGyro();
 ST.motor(1 , 0); //set motor 1 to stop
 ST.motor(2 , 0);  //set motor 2 to stop
}
void loop() {


 /* Read Pot Values ******************************************************/
```

```
  potValue1 = analogRead(analogInPin1);

  potValue2 = analogRead(analogInPin2);

  potValue3 = analogRead(analogInPin3);

  steeringValue = analogRead(steering);

  outputValue1 = map(potValue1, 0, 1023, 1000, 0.0)/100.0;

  outputValue2 = map(potValue2, 0, 1023, 10000, 0.0)/100.0;

  outputValue3 = 0;//map(potValue3, 0, 1023, 1000, 0)/100.0;

  turnRate = 0;//(steeringValue - 460)* .1;  //turnRate = ( pot - pot offset ) * turnRateGain

  /* Get Angle **********************************************************/

  float a = KalmanAngle() - INIT_ANGLE;

  angleBuffer[angleBufferIndex] = a;

  angleBufferIndex = (angleBufferIndex + 1) % ANGLE_BUFFER_LENGTH;

  float ang = GetAvgAngle();

  /* PID Calculation ****************************************************/

  curErr = ang - INIT_ANGLE;

  sumErr += curErr;

  if (sumErr > SumErrMax) sumErr = SumErrMax;

  else if (sumErr < SumErrMin) sumErr = SumErrMin;

  integralTerm = sumErr * elapsedTimeSec * float(outputValue2) / float(outputValue1);

  derivativeTerm = curErr - prevErr;

  derivativeTerm = derivativeTerm * 10000 * float(outputValue3) * elapsedTimeSec /
float(outputValue1);

  /* Update value to motors ***************************************/

  float To_motor_R = ((curErr + integralTerm + derivativeTerm) * float(outputValue1)) - turnRate; //this
should balance the robot with no turning

  float To_motor_L = ((curErr + integralTerm + derivativeTerm) * float(outputValue1)) + turnRate; //this
should balance the robot with no turning
```

```
/*Send power value to Sabertooth ********************************/

sendToMotor(ang, To_motor_R, To_motor_L);

prevErr = curErr;

/*

Serial.print (accXangle);

Serial.print (",");

Serial.print (-((float)gyroX-GYRO_X_INIT_VAL)/131.0);

Serial.print (",");

Serial.print (kalAngleX);

Serial.print (",");

Serial.print (a);

Serial.print (",");

Serial.println (ang);

*/

 //Tuning Potentiometer values

Serial.print (outputValue1);

Serial.print (",");

Serial.print (outputValue2);

Serial.print (",");

Serial.println (outputValue3);

  /*

 Serial.print((int)To_motor_R);

 Serial.print(",");

 Serial.print(curErr);

 Serial.print(",");

 Serial.print(prevErr);
```

```
  Serial.print(",");

  Serial.print(derivativeTerm *100);

  Serial.print(",");

  Serial.print(sumErr);

  Serial.print(",");

  Serial.println(integralTerm);

*/

 delay(1);

}

void i2cWrite(uint8_t registerAddress, uint8_t data){

 Wire.beginTransmission(IMUAddress);

 Wire.write(registerAddress);

 Wire.write(data);

 Wire.endTransmission(); // Send stop

}

uint8_t* i2cRead(uint8_t registerAddress, uint8_t nbytes) {

 uint8_t data[nbytes];

 Wire.beginTransmission(IMUAddress);

 Wire.write(registerAddress);

 Wire.endTransmission(false); // Don't release the bus

 Wire.requestFrom(IMUAddress, nbytes); // Send a repeated start and then release the bus after
reading

 for(uint8_t i = 0; i < nbytes; i++)

   data[i] = Wire.read();

 return data;

}
```

Appendix II: Figures and Tables



Row #	[illegible]	Relative Weight	Weight / Importance	Demanded Quality	1	2	3	4	5	6	7	8	9	10	Our Company	[illegible]	[illegible]
Direction of Improvement					x	x	▼	x	▲	▲	▲	▼	▼	▼			
1	9	14.1	10.0	Able to carry passenger 100-220lbs	Θ	O	▲	▲	O	▲	▲	▲	▲	▲		3	5
2	9	11.3	8.0	Maximum speed of 5mph	▲	Θ	▲	▲	O	▲	O	▲	▲	▲		4	5
3	9	14.1	10.0	Cost	▲	O	Θ	▲	O	O	O	O	O	▲		5	2
4	9	7.9	5.6	Comfort/Ergonomics	▲	▲	O	Θ	▲	▲	▲	▲	▲	Θ		3	1
5	9	8.5	6.0	Battery Life >1 hour	▲	O	O	▲	Θ	▲	Θ	▲	▲	▲		5	2
6	9	5.6	4.0	Weight <40lbs	▲	▲	Θ	▲	▲	Θ	▲	▲	▲	▲		4	5
7	9	14.1	10.0	Reliability	Θ	Θ	▲	▲	▲	▲	Θ	▲	▲	▲		4	3
8	9	11.3	8.0	Ease of manufacturing	▲	▲	▲	▲	▲	▲	▲	Θ	Θ	▲		5	2
9	9	9.9	7.0	Ease of repairs/Modifications	▲	▲	▲	▲	▲	▲	▲	▲	Θ	▲		1	2
10	9	4.2	3.0	Control positions	▲	▲	▲	Θ	▲	▲	▲	▲	▲	Θ		4	2
				Target or Limit Value													
				Difficulty (0=Easy to Accomplish, 10=Extremely Difficult)	3	2	5	6	3	1	7	3	4	5			
				Max Relationship Value in Column	9	9	9	9	9	9	9	9	9	9			
				Weight / Importance	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]			
				Relative Weight	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]			

Figure 13: House of quality for PID Control Learning Robot

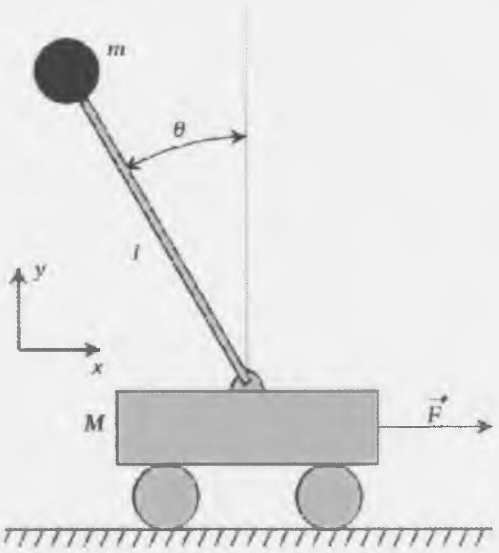


Figure 14: Simple model of an inverted pendulum

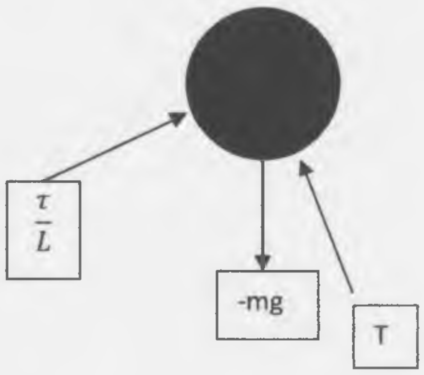


Figure 16: Free body diagram of person

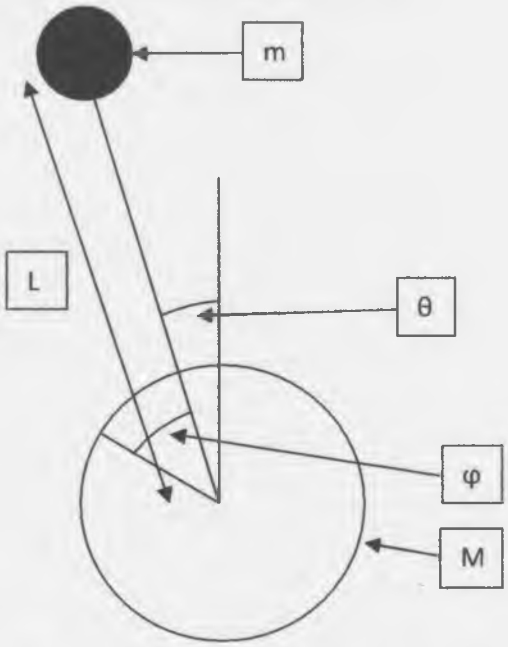


Figure 15: Analytical model of balancing robot

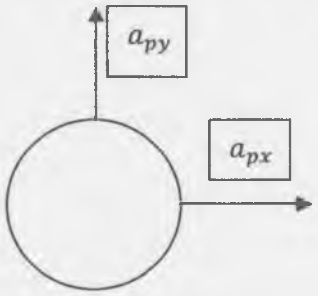


Figure 17: Mass acceleration diagram of person

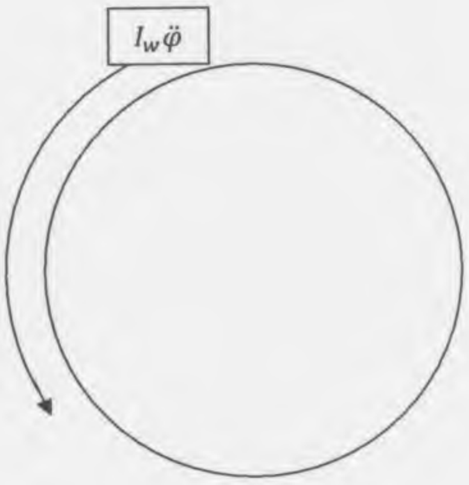


Figure 19: Mass acceleration diagram of the wheel

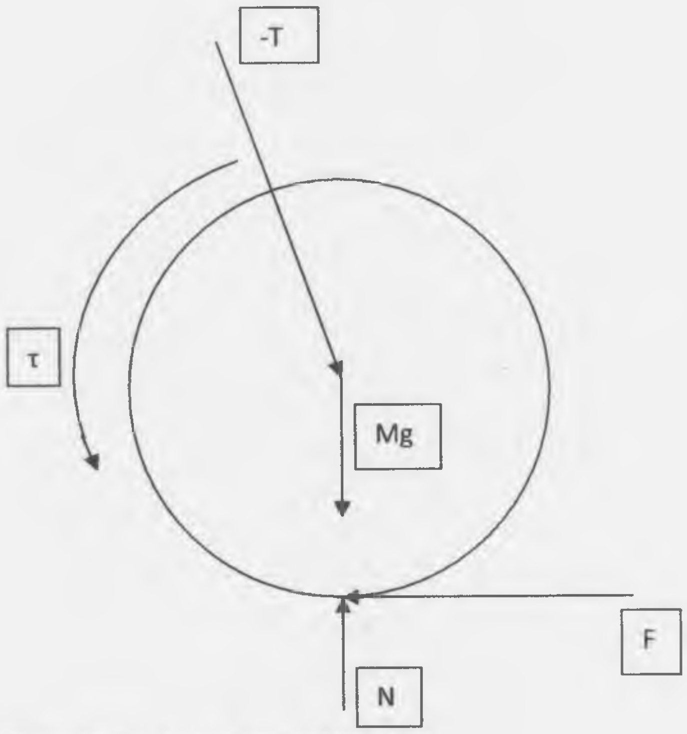


Figure 18: Free body diagram of the wheel

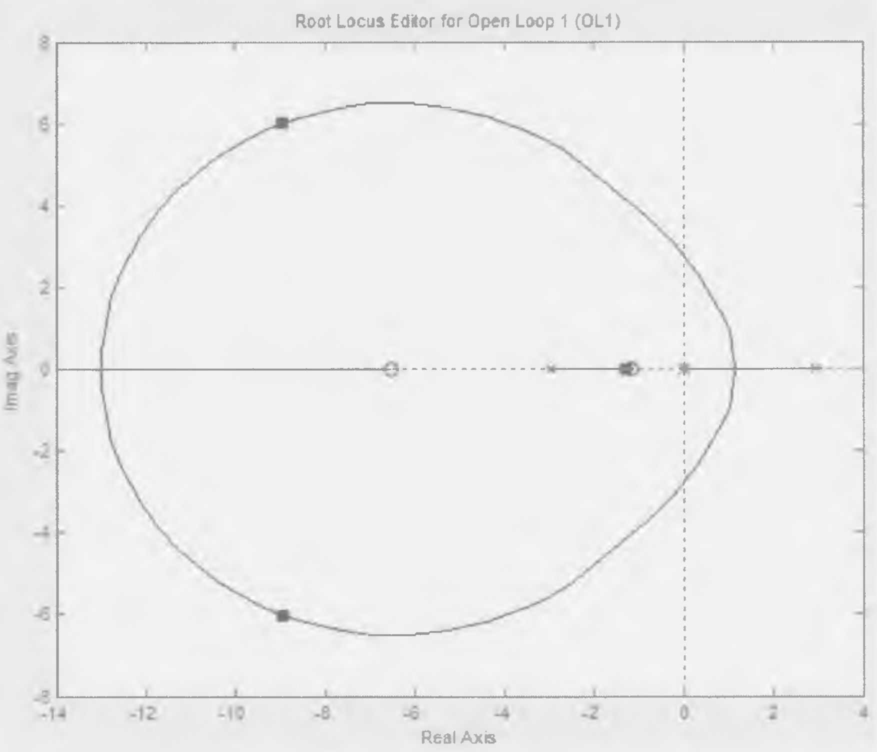


Figure 20: Root locus plot for balancing robot

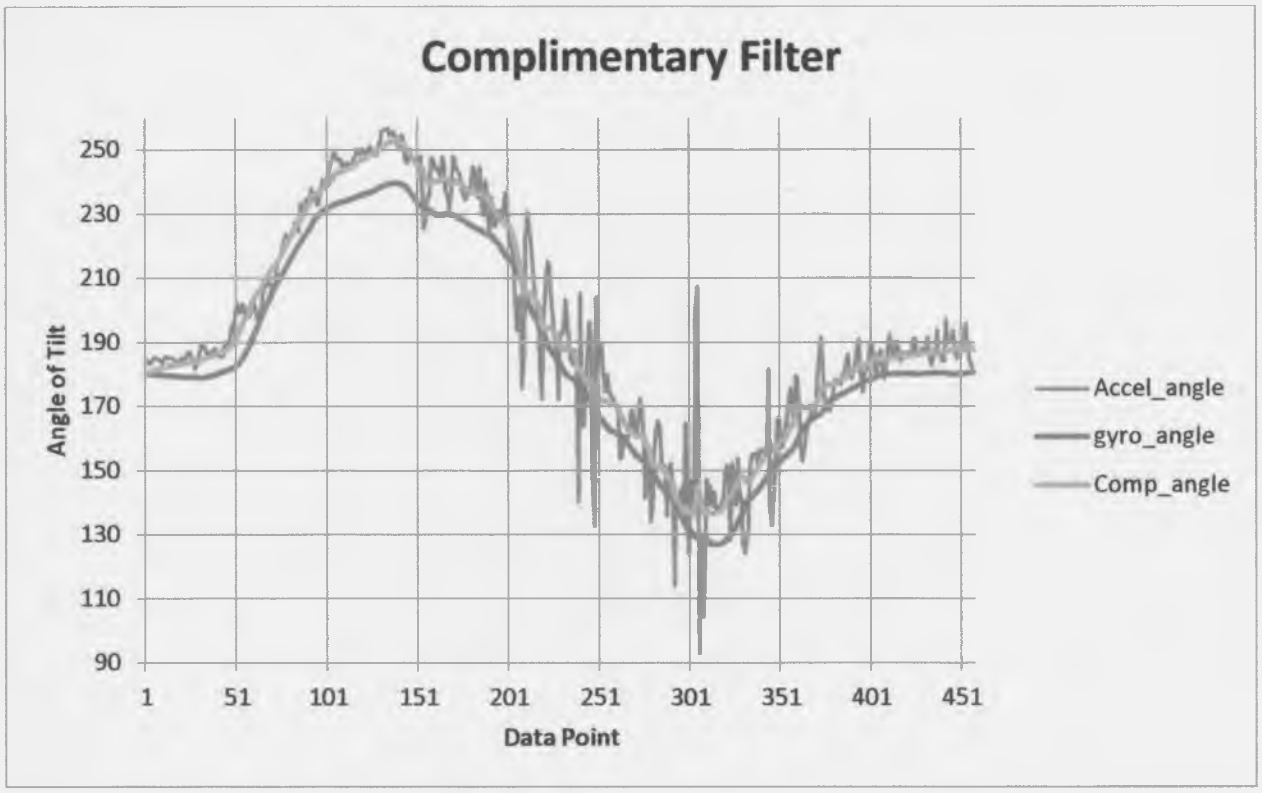


Figure 21: Combination of signals using a complimentary filter

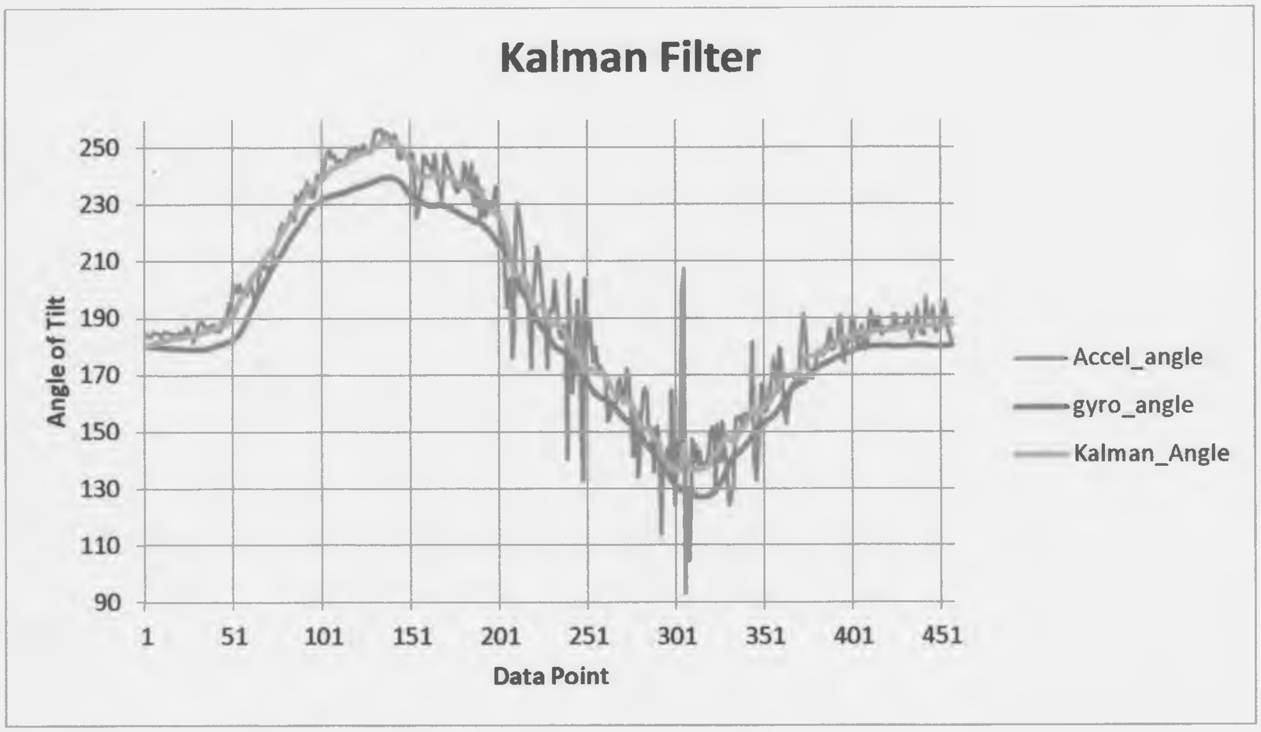


Figure 22: Combination of signals using a Kalman filter

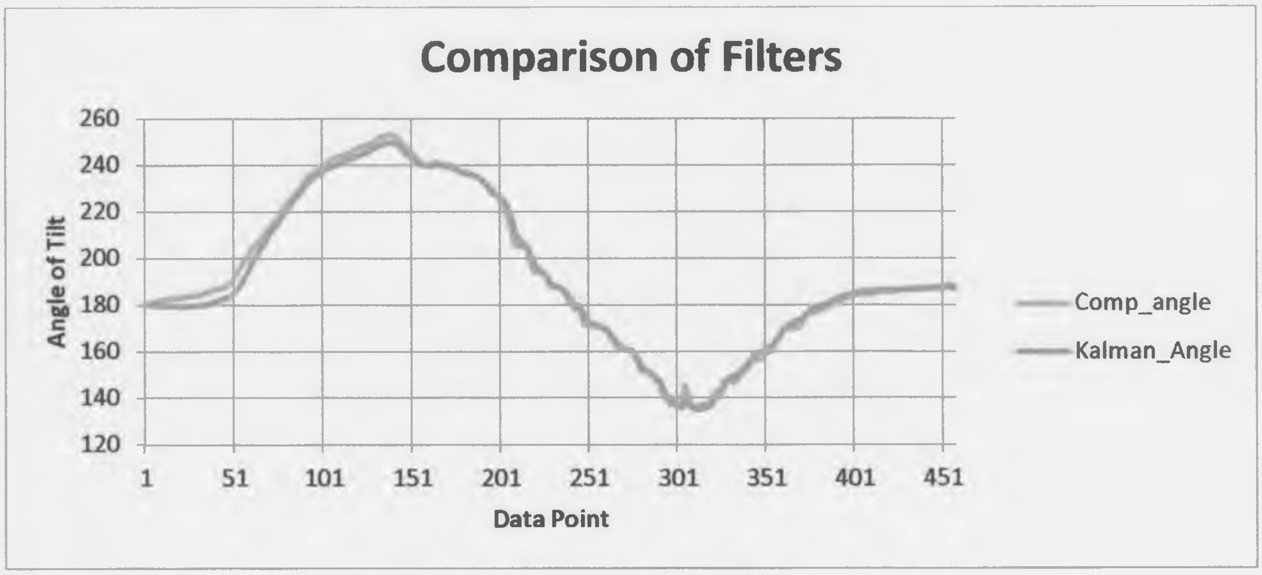


Figure 23: Comparison of Complimentary and Kalman filters on same data set

	Cost	Reliability	Manufacturability	Availability of Materials	Feasibili ty	Compatibility	Total
Motor							
New	-1	1	1	1	-1	1	8
Used	0	0	1	0	1	0	7
Repurpose	1	0	1	0	1	0	11
Chassis							
Aluminum	0	1	1	1	0	1	16
Steel	0	1	1	1	0	1	16
Wood	1	0	1	1	1	1	18
Carbon Fiber	-1	0	-1	-1	-1	-1	-18
Electronics							
Arduino Uno	0	1	1	1	1	1	20
Texas Instruments LaunchPad	1	0	0	1	1	0	13
Weight	4	6	3	5	4	2	

Table 3: Decision matrix for various components

Appendix III: Equations

List of symbols

a_p- Acceleration of person

a_b- Acceleration of wheel

$a_{p/b}$- Acceleration of person relative to wheel

$\dot{\theta}$ – Angular velocity

$\ddot{\theta}$ – Angular Acceleration

$\ddot{x}$ – Linear Acceleration

L – Length from pivot to center of gravity

Θ – Angle relative to vertical

T – Tension in rod

τ – Applied torque

m – Mass of the person

M – Mass of the wheel

g – Acceleration due to gravity

F – Force due to friction

R – Radius of wheel

I_w – Moment of inertia of the wheel

k_d – Derivative Gain

k_p – Proportional Gain

k_i – Integral Gain

Equation 23

$$a_p = a_b + a_{p/b}$$

Equation 24

$$a_p = a_b + L\ddot{\theta}\widehat{e_\theta} - L\dot{\theta}^2\widehat{e_r}$$

Equation 25

$$a_p = \left[a_b - L\ddot{\theta}\cos(\theta) + L\dot{\theta}^2\sin(\theta)\right]i + \left[-L\ddot{\theta}\sin(\theta) - L\dot{\theta}^2\cos(\theta)\right]j$$

Equation 26: sum of forces in i direction

$$-Tsin(\theta) + \frac{\tau}{L}\cos(\theta) = m\left[a_b - L\ddot{\theta}\cos(\theta) + L\dot{\theta}^2\sin(\theta)\right]$$

Equation 27: sum of forces in j direction

$$Tcos(\theta) + \frac{\tau}{L}\sin(\theta) - mg = m[-L\ddot{\theta}\sin(\theta) - L\dot{\theta}^2\cos(\theta)]$$

Equation 28: Sum of forces in i direction

$$Tsin(\theta) - F = Ma_b$$

Equation 29; Sum of forces in j direction

$$\tau - FR = I_w\ddot{\varphi}$$

Equation 30

$$\ddot{\varphi} = \frac{\ddot{x}}{R}$$

Equation 31

$$F = \frac{\tau}{R} - \frac{I_w}{R}\ddot{x}$$

Equation 32: Sum of forces in i direction

$$Tsin(\theta) - \frac{\tau}{R} - \frac{I_w}{R}\ddot{x} = M\ddot{x}$$

Equation 33: Transfer function 1

$$\tau - mgLsin(\theta) = mL\ddot{x}\cos(\theta) - mL^2\ddot{\theta}$$

Equation 34: Transfer function 2

$$\left(m + M + \frac{I_w}{R^2}\right)\ddot{x} + \tau\left(\frac{1}{R} - \frac{\cos(\theta)}{L}\right) = -mL\ddot{\theta}\cos(\theta) + L\dot{\theta}^2\sin(\theta)$$

Equation 35: Linearized transfer function 1

$$\tau - mgL\theta = mL\ddot{x} - mL^2\ddot{\theta}$$

Equation 36: Linearized transfer function 2

$$\left(m + M + \frac{I_w}{R^2}\right)\ddot{x} + \tau\left(\frac{1}{R} - \frac{1}{L}\right) = -mL\ddot{\theta}$$

Equation 37: Transfer function 1 in Laplace domain

$$\tau - mgL\Theta = mLXs^2 - mL^2\Theta s^2$$

Equation 38: Transfer function 2 in Laplace domain

$$\left(m + M + \frac{I_w}{R^2}\right)Xs^2 + \tau\left(\frac{1}{R} - \frac{1}{L}\right) = -mL\Theta s^2$$

Equation 39: Transfer function 1

$$\Theta = \frac{mLs^2X - \tau}{mL^2s^2 - mgL}$$

Equation 40: Transfer function 2

$$X = \frac{-mLs^2\Theta - \tau\left(\frac{1}{R} - \frac{1}{L}\right)}{\left(m + M + \frac{I_w}{R^2}\right)s^2}$$

Equation 41: Open loop transfer function, referenced as T in future equations

$$\Theta = \frac{\left[-mL\left(\frac{1}{R} - \frac{1}{L}\right) - \left(m + M + \frac{I_w}{R^2}\right)\right]s^2}{\left[mL^2s^4 + \left(\frac{m^2L^2}{\left(m + M + \frac{I_w}{R^2}\right)} - mgL\right)s^2\right]}\tau$$

Equation 42: Closed loop transfer function

$$\Theta = T\left(k_d s + k_p + \frac{1}{s}k_i\right)(\Theta_{des} - \Theta)$$

Equation 43: Rearranged closed loop transfer function

$$\Theta = \frac{T\left(k_d s + k_p + \frac{1}{s}k_i\right)}{1 + T\left(k_d s + k_p + \frac{1}{s}k_i\right)}\Theta_{des}$$

Equation 44: Complimentary Filter

$$\theta_i = ((1 - \alpha) * (\theta_{i-1} + GyroReading * \Delta t) + \alpha * AccelReading$$