

Spring 2015

Self-Balancing Two Wheeled Robot

David Laubli

The University Of Akron, dwl10@zips.uakron.edu

Thomas Garabedian

The University Of Akron

Jordan Paul

The University Of Akron

Nikheel Patel


The University Of Akron

Michael Redle

The University Of Akron

Please take a moment to share how this work helps you [through this survey](#). Your feedback will be important as we plan further development of our repository.

Follow this and additional works at: http://ideaexchange.uakron.edu/honors_research_projects

 Part of the [Controls and Control Theory Commons](#), and the [Electrical and Electronics Commons](#)

Recommended Citation

Laubli, David; Garabedian, Thomas; Paul, Jordan; Patel, Nikheel; and Redle, Michael, "Self-Balancing Two Wheeled Robot" (2015). *Honors Research Projects*. 65.

http://ideaexchange.uakron.edu/honors_research_projects/65

This Honors Research Project is brought to you for free and open access by The Dr. Gary B. and Pamela S. Williams Honors College at IdeaExchange@UAkron, the institutional repository of The University of Akron in Akron, Ohio, USA. It has been accepted for inclusion in Honors Research Projects by an authorized administrator of IdeaExchange@UAkron. For more information, please contact mjon@uakron.edu, uapress@uakron.edu.

Self-Balancing Two Wheeled Robot

Senior Design Final Project Report

Team # 10
Thomas Garabedian
David Laubli
Nikheel Patel
Jordan Paul
Michael Redle

Dr. Robert Veillette

April 22, 2015

I) Table of Contents

● List of figures	3
● Abstract	4
● Problem Statement: Need and Objective	5
● Marketing Requirements	5
● Objective Tree	6
● Background	7
● Design Requirements Specifications	12
● Accepted Technical Design	12
● Hardware Calculations	31
● Parts List	33
● Design Team Information	36
● Conclusion	37
● References	38

II) List of Figures (TG)

Figure A1: Objective Tree of Butler Bot

Figure A2: Generic drawing of patent US 8442661 B1 [1.5]

Figure A3: Control system of the first patent [5]

Figure A4: Functionality Loop for the Second Patent

Figure B1: Simulink block diagram of the inner and the outer loops

Figure B2: Root locus plot of the inner loop and the system angular velocity plant.

Figure B3: Root Locus of Control System

Figure B4: Magnitude and phase bode plots.

Figure B5: Linear velocity vs time.

Figure C1: PIC24FJ128GA010 Microcontroller Pin Diagram

Figure C2: Programming Diagram for the PIC

Figure C3: Software Flowchart for Instruction Execution

Figure C4: UART Diagram for PC Output

Figure C5: Interfacing Methods Signal Lines

Figure C6: I2C Timing Diagram

Figure D1: Line sensor array

Figure D2: Wheel Free Body Diagram

Figure D3: Pendulum Free Body Diagram

Figure D4: Root Locus of Longitudinal Controls

Figure D5: Bode Plot of Longitudinal Controls

Figure D6: Longitudinal Controls Loop Simulink Diagram

Figure D7: Step Response of Longitudinal Controls with Velocity as the Output

Figure D8: Step Response with Current as the Output

Figure D9: Step Response with Pendulum Angle as the Output

Figure D10: Zoomed in Step Response with Pendulum Angle at the Output

Figure D11: Step Response with Pendulum Angular Velocity as the Output
Figure D12: Free Body Diagram for the Lateral Controls
Figure D13: Lateral Controls Loop Simulink Diagram
Figure D14: Longitudinal and Lateral Controls Interface
Figure D15: Lateral Controls Step Response
Figure D16: Step Response of Lateral Controls with Current as the Output
Figure D17: Step Response of Lateral Controls with Differential Angle at the Output
Figure E1: Schematic of Motor Driver Circuit
Figure E2: Motor Driver
Figure F1: Step Response of Original Phase-Lag Controller
Figure F2: Root Locus Plot of Original Phase-Lag Controller
Figure F3: Step Response of Modified Phase-Lag Controller
Figure F4: Root Locus Plot of Modified Phase-Lag Controller

III) List of Tables

Table 1: Engineering Tradeoff Matrix of Butler Bot
Table 2: Marketing/Engineering Requirements Trade-off Matrix
Table 3: Gains vs Actual Voltage Values
Table 4: Controls Related Design Specifications
Table 5: Estimated Weight of Robot Butler
Table 6: Motor test results
Table 7: Balance test results
Table 8: Original Gantt Chart
Table 9: Revised/Actual Gantt Chart

Abstract (TG)

Automation is increasingly becoming a larger part of daily life. From automated telephone calls to machines in manufacturing, robots are generally an effective and efficient way to reduce overhead costs, increase consistency in products and services, and perform tasks that may be hazardous to humans. The successful design and building of a two-wheeled balancing robot demonstrates a knowledge of control systems and sensor interfacing that can translate to real world applications. Helping seniors live on their own, performing dangerous mining work, repeatedly screwing the same piece in an assembly line, are great examples of a controls automation system freeing time up for a person to perform more important or more complex tasks, and all of these tasks use design techniques similar to that of a balancing robot. The robot will balance on two wheels and be able to have loads of varying weight and size (up to 5lbs) placed on the top platform. It will be capable of handling disturbances including bumps from humans or running into stationary objects and it can accommodate flooring changes (carpet, tile etc.) while maintaining balance. An accelerometer and a gyroscope feed information back to a pic microcontroller which feeds a PWM signal to two motors that drive the wheels so they stay under the center of mass of the robot.

Key Features: (DL)

- Balance on two wheels
- Avoid spilling load
- Stand roughly 70cm tall
- Accept the weight of food tray and continue to balance
- Light-weight

The following key features were removed from the design due to time and budget constraints:

- Senses when delivery is completed and returns
- Differentiate between different paths
- Follow line to destination

Problem Statement

Need (TG):

For many businesses one of the chief expenses is payroll. Many businesses fail because they cannot afford the costs associated with a high payroll. To improve the success rate and curb the number of failing businesses, there is a need to reduce the costs. A system that can autonomously and repeatedly does the same task can greatly affect the survival of a company. The main goal of the balance bot is to demonstrate the engineering ability to consistently and effectively produce solutions to a wide variety of problems. With the technical experience and know how gained from producing a balance bot engineers can feel comfortable contributing to more advanced real world problems.

Objective (TG):

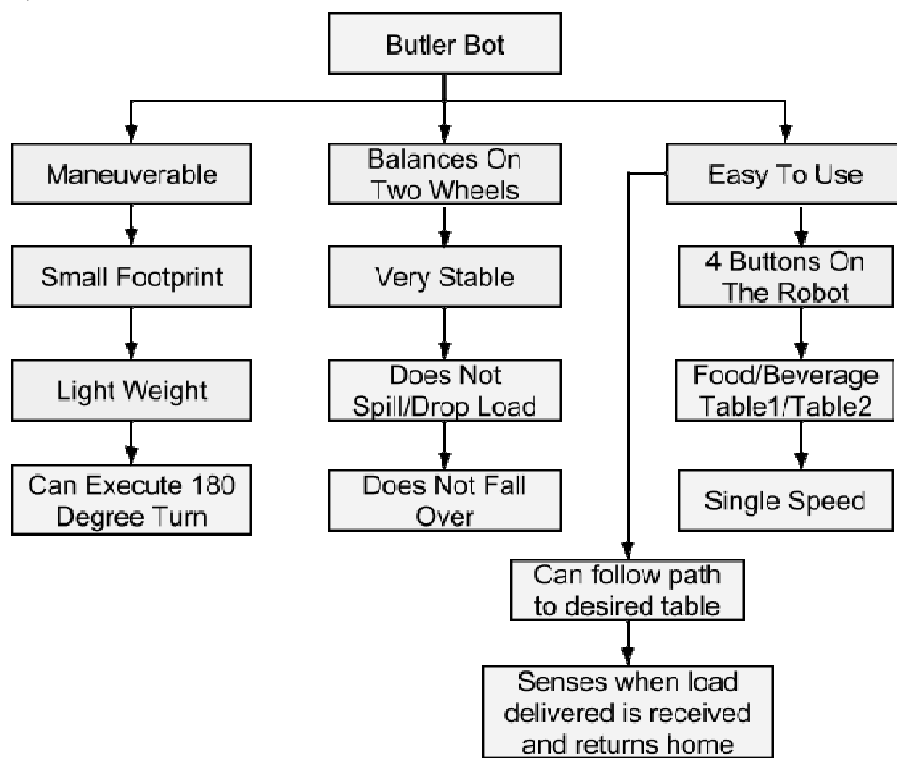


Figure A1: Objective Tree of the Butler Bot

The objective is to develop and build a free standing robot that maintains its balance. The robot will balance on two wheels and use two sensors, a gyroscope and an accelerometer, as feedback for determining the current angular position versus the desired angular position. Figure 1 gives a visual representation of the objectives the robot should accomplish.

Marketing Requirements (DL):

1. Ability to balance on two wheels
2. Must be highly maneuverable, physically accessible, and able to travel at a moderate speed.
3. The system should involve multiple white lines which lead to different tables.
4. The waiter must intelligently select which line to follow based on a command given to reach the target destination.
5. The waiter must not spill drinks poured to within two centimeters of the brim of the glass containing the fluid
6. The robot must be priced reasonably low, and have a decently long lifespan.

Edit: Marketing requirements 3 and 4 no longer apply to this project, as time constraints prevented a line-following system and start/stop command from being implemented on the robot.

	A	B	C	D	E	F	G	H	I
1		Speed and acceleration	Line-following ability	Balancing ability	Small tilt distance	Battery life	Small size/footprint	Low weight	Cost
2	Speed and acceleration	X	-	-	-	-	-	-	-
3	Line-following ability		X	0	+	-	+	+	-
4	Balancing ability			X	+	0	-	0	-
5	Small tilt distance				X	0	0	0	-
6	Battery life					X	+	+	-
7	Small size/footprint						X	+	-
8	Low weight							X	-
9	Cost								X

Table 1: Engineering Tradeoff Matrix of Butler Bot

Shown in Table 1 is the engineering trade-off matrix that describes the relationship among the design requirements. Positive relationships, or relationships where requirements benefit from each other, are indicated by a plus (+), negative relationships, where requirements between relationships hinder one another, by a minus (-), and no relationship by a 0 (or X).

	A	B	C	D	E	F	G	H
1		Speed/Manuverability	Line following capability	Ability to balance	Multiple tables	Line Selection	Food Delivery	Cost
2	Speed and Accerleration	X	-	-	0	-	-	-
3	Line-following ability	+	X	-	+	-	+	+
4	Balancing ability	+	-	X	0	-	-	-
5	Small tilt distance	-	0	+	0	0	+	0
6	Battery Life	-	-	-	-	-	-	-
7	Small size/footprint	+	-	-	0	0	-	+
8	Low Weight	+	+	+	0	0	+	-
9	Cost	-	0	-	+	0	+	x

Table 2: Marketing/Engineering Requirements Trade-off Matrix

BACKGROUND

Patent Search:

There are two patents pertaining to a self-balancing two wheeled robot, the first being US 8442661 B1. “The exemplary robotic system also comprises a first actuator, such as a pneumatic cylinder, configured to change a waist angle defined between the leg segment and the torso segment, a first control system configured to maintain balance of the robotic system on the wheels, and a second control system configured to change a base angle responsive to changing the waist angle. Here, the base angle is defined between a first reference plane having a fixed relationship to the base and a second reference plane having a fixed relationship to an external frame of reference [1].” The given design however does not have a usable platform for the placement of food. Figure 2 shows a schematic of the patented balance bot.

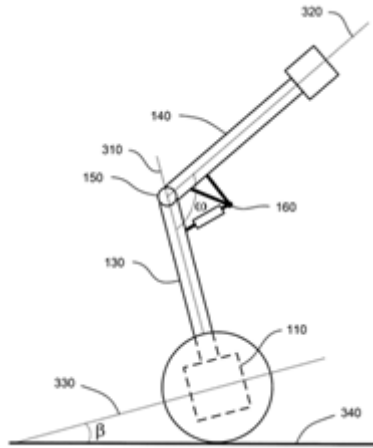


Figure A2: Generic drawing of patent US 8442661 B1 [1.5]

Notice that the top bar in figure 2 is used to change the moment of inertia. The label 160 is of an optical encoder that can determine the angle w . An optical encoder would be just as useful for determining the angle of a pendulum balancing arm.

“The center of gravity of the combined body segments above the waist joint, such as the torso segment and head, can be further than half their overall length from the waist joint, in some embodiments [2].” It is elsewhere suggested that the moment of inertia of the base protruding arm should be as low to the ground as possible and the moment of inertia of the torso or the object labeled 140 should have a moment of inertia that is as far from the base as possible to maximize the effectiveness of the balancing arm.

The robot is also meant to have a head (located at the very top of the robot) that can rotate. A laser that would (ideally) focus parallel to the ground would be mounted on the head. The position of the laser would demonstrate the effectiveness of the robot. A robot waiter would not need such a device however.

“The robotic system also comprises a position sensor configured to detect a position of the base, and movement logic configured to maintain the base at a preferred position responsive to the detected position of the base. The robotic system further comprises a waist angle sensor configured to detect a waist angle between the lower segment and the upper segment, and a base angle calculator configured to calculate a base angle responsive to the detected waist angle, the base angle being calculated to approximately maintain a center of gravity of the system [3].”

Linear position sensors detect the deviation in position from an initialized position. Linear position sensors are limited to a given linear range of deviation in position from the initialized position. Consequently they are not ideal for balance bots that travel. The advantage of this patent is that the robot can maintain balance with a minimal amount of linear motion thanks to the upper arm. But what is necessary is a gyro at the base to sense the acceleration of the base. An integration of the accelerometer output would give the change in position.

An angular position sensor is useful for any balancing robot. This technology senses the change in angle from an initialized position. A gyro merely measures angular velocity. A gyro with a position sensor could provide even more valuable information to the user than a gyro by itself [4]. Figure 3 is a block diagram showing the controls system of a robot.

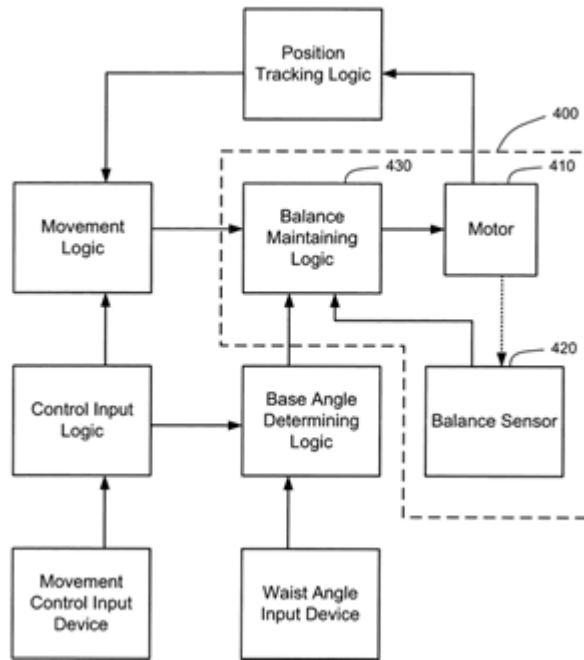


Figure A3: Control system of the first patent [5]

Notice the waist angle input device. This device has the purpose of giving the desired angle in between the two arms of the robot. The relevant part of the patent seen in figure 3 is the upper loop, and it has the Movement Control Device as the input. The waist angle would be a product of the Balance Maintaining Logic block and not a user input for this device. The position tracking logic of the upper loop, could transfer to the butler bot, and rotary encoders would be the best way to implement that logic.

The next patent is US 8478490 B2. This patent is for a controls system of a robot that is one wheeled and has to maintain balance from side to side and forward and backward as well. Figure 4 shows the computer algorithm for patent US 8478490.

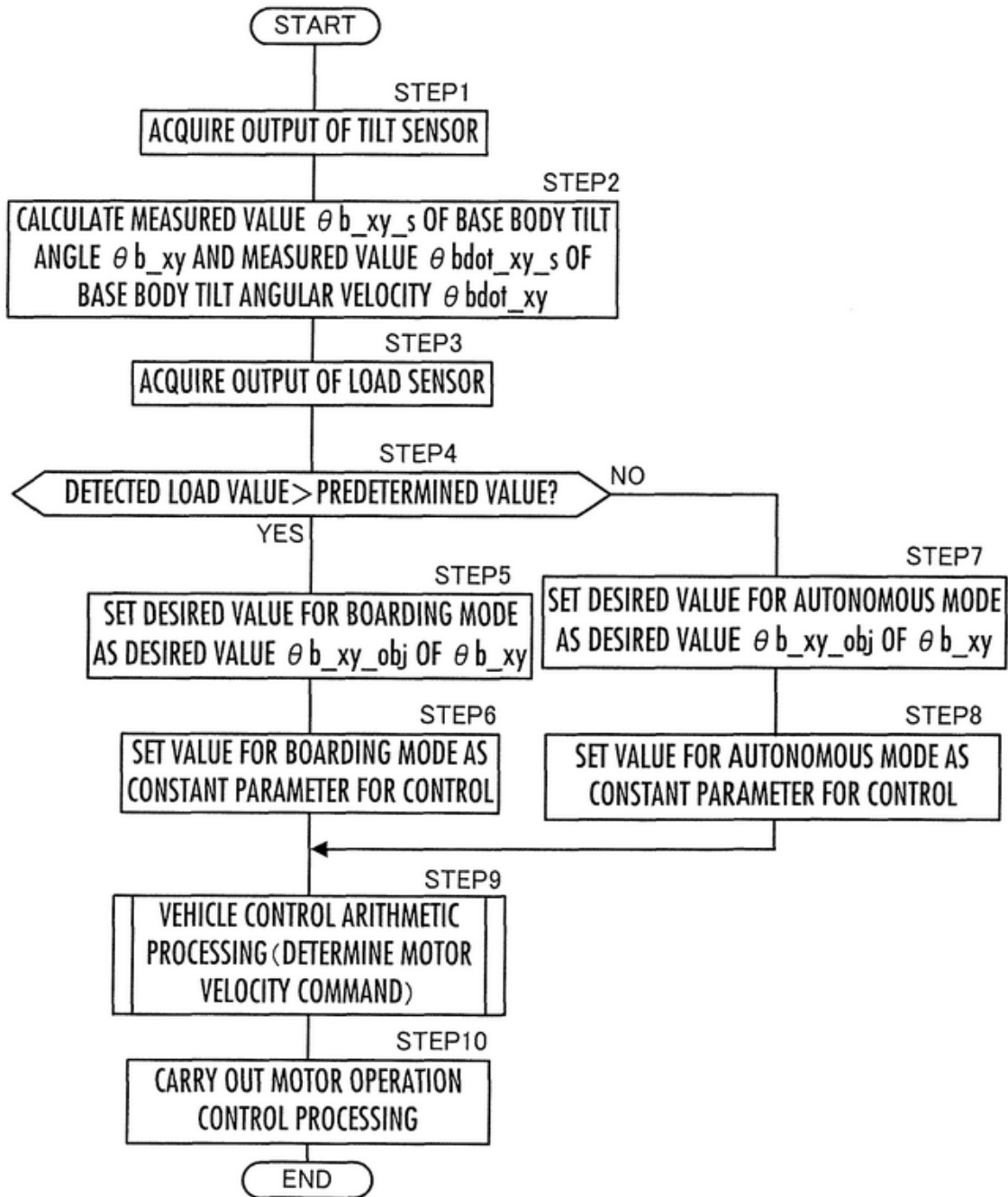


Figure A4: Functionality Loop for the Second Patent

“First, in STEP 1, the control unit 50 acquires an output of a tilt sensor 52 [6].”

“Subsequently, the control unit 50 proceeds to STEP 2 to calculate a measured value $\theta_{b_xy_s}$ of a base body tilt angle θ_{b_xy} and a measured value $\theta_{b\dot{xy}_s}$ of a base body tilt angular velocity $\theta_{b\dot{xy}}$ on the basis of the acquired output of the tilt sensor 52 [7].”

Next, the load sensor determines if there is someone sitting on the seat. In step 4 it is determined whether the load is greater than the predetermined value that was stored. If it is determined that

the load is greater than predetermined load value, the ideal tilt angle is determined. The values of various gains are then set. These gains are multiplied by the various errors tilt and desired velocity to increase the sensitivity of the control loops. In step 5 the ideal tilt angle is set into the controller. In step 6, the desired parameters h_x , h_y , $K_{i_a_x}$, $K_{i_a_y}$, $K_{i_b_y}$, are set where $i=1, 2, 3$. The exact function of these gains however is not relevant to the project because the attempted project is not a replica of the patent. However, if it is determined that the load is less than the predetermined stored value, the vehicle is set for autonomous mode in which there is no rider. The values for the parameters are then determined. The steps in 5 and 6 and 7 and 8 are not necessary for each processing cycle. In all modes the desired value for θ_x and $\dot{\theta}_y$ are both ideally zero according to the patent [8]. Gains can be easily manipulated if the control system is digital.

In step 10, the operating conditions for the motors are carried out which are determined in block 9. The difference between measured and actual velocity goes to zero. Then the control unit insures that the desired torque is applied by the motors [8]. The robot waiter will need to have a base which is stationed on the top of the inverted pendulum which is analogous to the seat. The robot will have a two button interface, one for table 1 and another for table 2, and when the load is placed onto the robot, the user will press the corresponding button and the robot will travel to that table. Pressing the same button again after it reaches the desired table will result in the robot returning to its home position. The desired tilt angle would remain the same due to the careful placing of the food however. The robot would never move to a destination without a package so there would be no need for an equivalent autonomous mode.

Design Requirements Specification (DL)

1. (MR 1, 6) The robot must balance on two wheels.
2. (MR 3) The robot must have the ability to move forward at a speed of 0.3 meters/sec, and it must be able to accelerate to this speed in under 3 seconds.
3. (MR 4, 5, 7) The robot will be semi-autonomous: it must follow a one-inch wide strip of white tape, to turn right or left at a fork as required by its destination and the path it is following, and to travel back in the direction of its intended path if it gets off the tape.
4. (MR 1, 6) The robot should not tilt more than 10 degrees while being compensated by the motors.
5. (MR 8) The robot must have a battery life of at least one-half hour of time in motion. When it is not delivering it should be at its home station, charging as necessary.
6. (MR 3, 8) The robot must be between 70 cm and 90 cm tall, making the items it delivers extremely accessible to seated customers.
7. (MR 3) The robot must have a compact footprint of less than 80x50x50 cm and weigh less than 7 kg. The maximum load the robot will be able to carry is 0.5 kg.
8. (MR 4, 5, 7) The robot must be able to determine when it is carrying a cargo, which will help determine when it starts moving. It must sense when it needs to slow down and stop to allow for cargo delivery, which will be implemented using the line sensors on the robot and line indicators on the floor.
9. (MR 8) Production cost for the robot must not exceed \$600.

Edit: design requirements 3 and 8 no longer apply to this project, as there was not time after the robot was balancing adequately to implement a line-following sensor and load sensors, in addition to the controls theory and algorithms necessary for directional and differential motion of the wheels of the robot. The remainder of these requirements were met

Accepted Technical Design (JP) (MR) (DL) (NP) (TG)

There were 4 design approaches which were experimented with for the controls. The first was classical controls, the second was state variable feedback, the third revision was a reversion to classical controls and the 4th was simple PID control. Only the last two of the control techniques were experimented with.

The classical controls approach was useful for gaining a fundamental understanding of system dynamics by means of root locus. The system dynamics could be understood first through varying the physical variables of the plant model to determine the optimal relationships among the variables. This however, was only done to a small extent. The key objective was to choose realistic values of the physical plant parameters of the system.

The system was first simulated without friction as a parameter. It was found that when the system did not have friction, there was no dc steady state value of the output of the angular velocity of the wheels. It was also desired to have a pendulum angle compensator to stabilize the pendulum in the inner loop and a wheel angular velocity compensator for the outer loop.

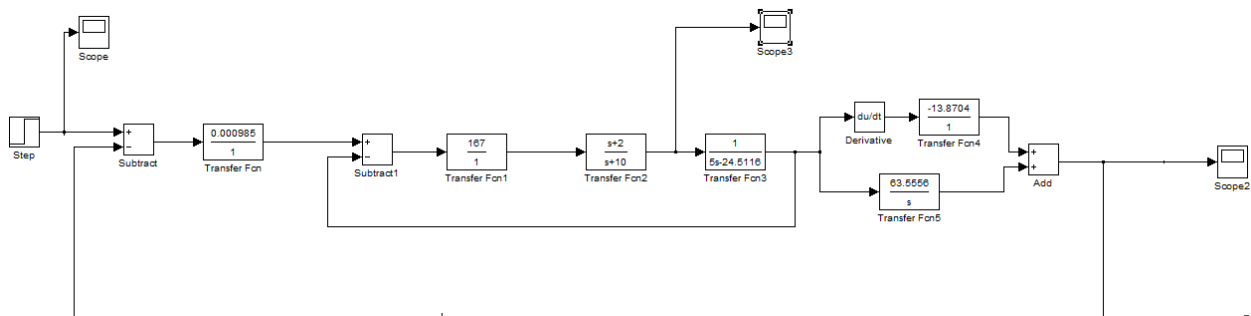


Figure B1 is a Simulink block diagram of the inner and the outer loops.

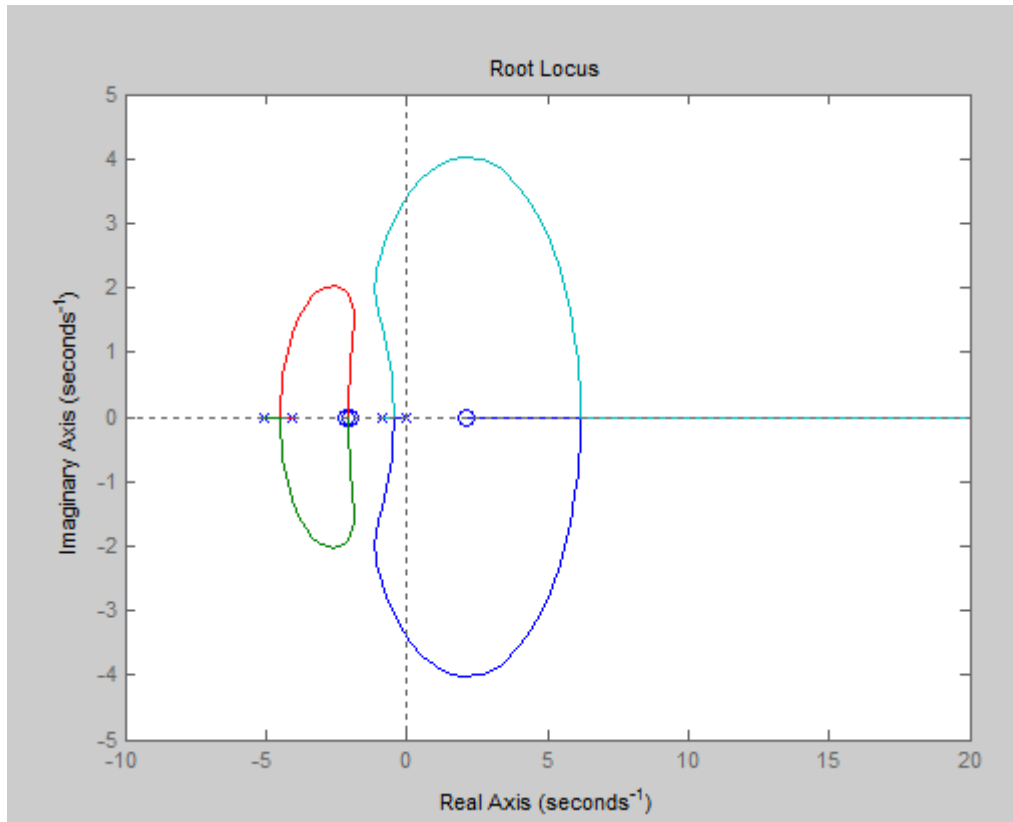


Figure B2: Root locus plot of the inner loop and the system angular velocity plant. It was found that as the gain of the root locus was increased, the poles of the blue and green branches moved towards the left hand plane and the gain margin decreased while the overall outer loop 2 % settling time decreased. A more favorable system was desired so state variable feedback was selected as an option.

The system matrix which is in terms of the system variables is given as follows.

$$\begin{bmatrix} \dot{\theta}_p \\ \ddot{\theta}_p \\ \dot{V}_w \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{T_c}{J_c} & -\frac{2D_f}{J_c} & \frac{2D_f}{J_c R_w} \\ -\frac{J_b T_c R_w}{J_a J_c} & 2\frac{D_f R_w}{J_a} \left(\frac{J_b}{J_c} + 1 \right) & -2\frac{D_f}{J_a} \left(1 + \frac{J_b}{J_c} \right) \end{bmatrix} \begin{bmatrix} \theta_p \\ \dot{\theta}_p \\ V_w \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{2K_a}{J_c} \\ \frac{2K_a R_w}{J_a} \left(1 + \frac{J_b}{J_c} \right) \end{bmatrix} I$$

The wheel velocity is the output of the system and the stabilization of the pendulum angle is what is firstly desired. The secondary priority of the system is to deliver the load to a given location or table

The attractiveness of state variable feedback is in its potential to in a simple manner manipulate the A matrix via three simple gains. The manipulation of these gains alters the system root locus and gives a lot of flexibility with the help of multiple sensors for each of the state variables. Because integral control was used, a root locus plot of the inner plant and controller could be simulated and a root locus plot of the system is Figure B3.

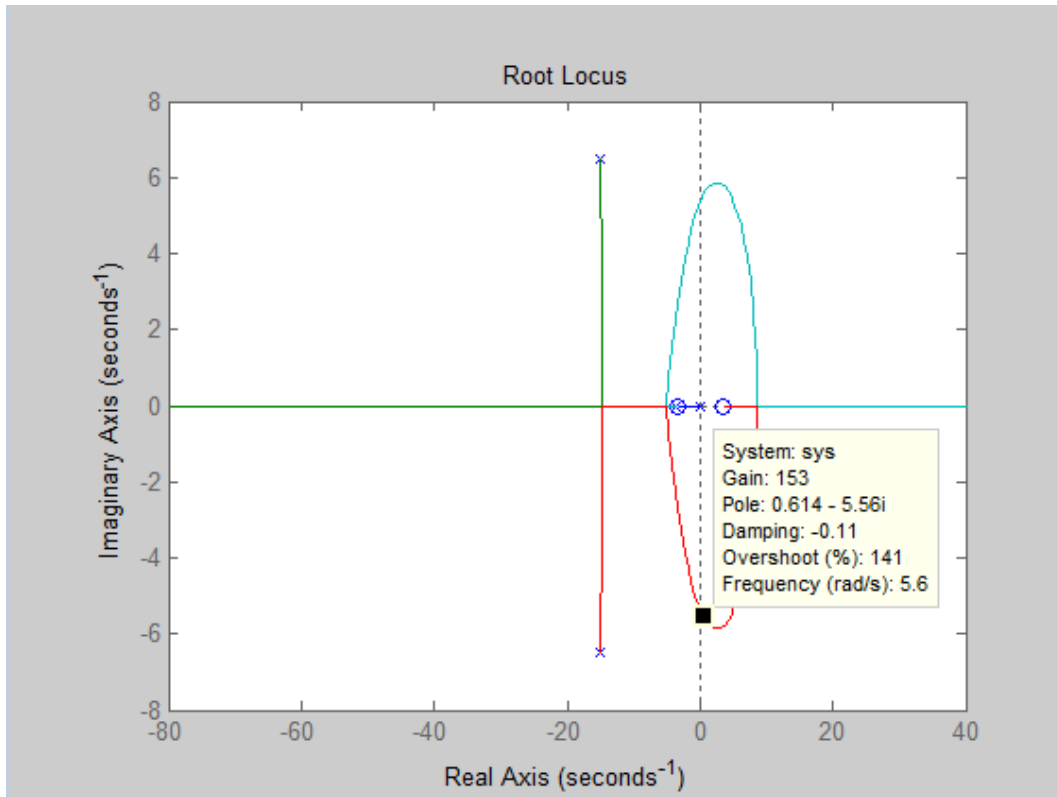


Figure B3: Root Locus of Control System

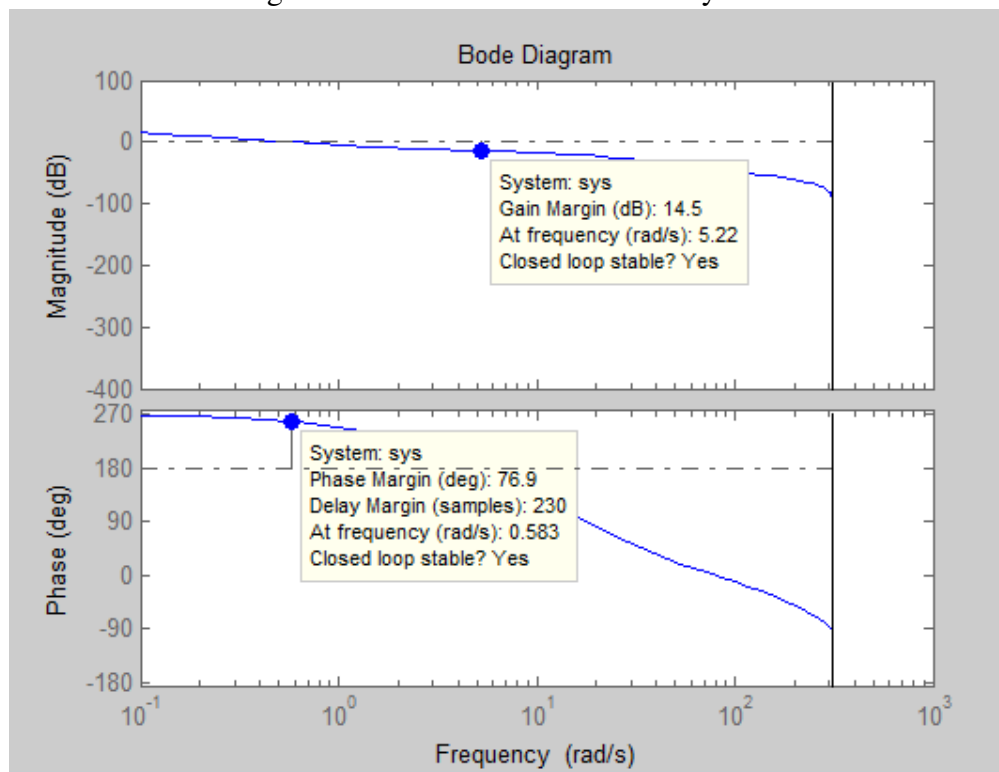


Figure B4: Magnitude and phase bode plots.

The gain and phase margins are given with the system velocity tuning gain chosen. The alleged gain and phase margins imply that stability will be no problem for this naturally unstable system. Figure B5 shows the Robot's linear velocity vs. time.

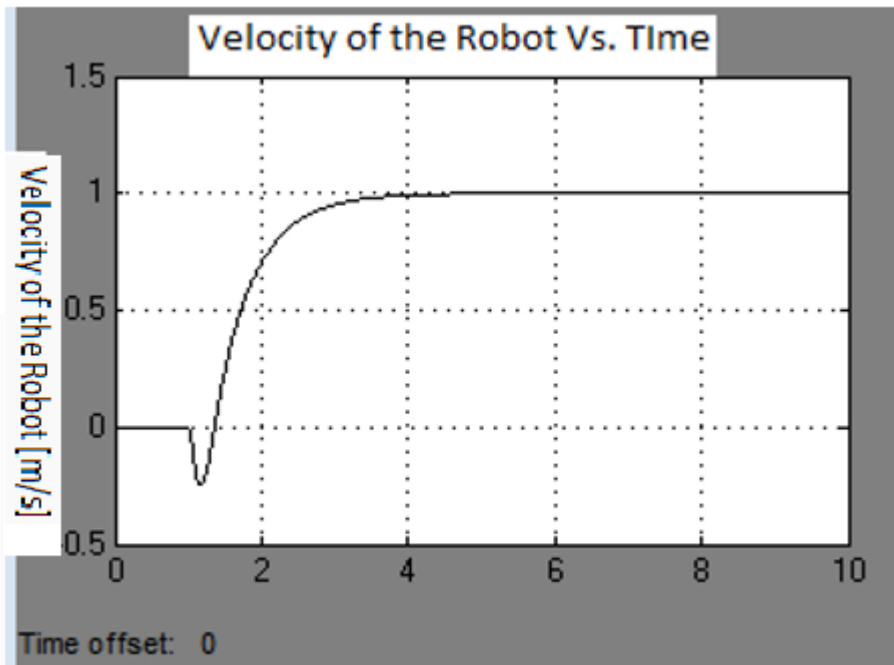


Figure B5: Linear velocity vs time.

Notice that the system has a reasonable settling time is a non-minimum phase system due to the zero in the right hand plane. State variable feedback has the advantages of stability and a reasonable settling time. There are two disadvantages of state variable feedback however. The first disadvantage is the need for multiple sensors. One sensor must ultimately yield, angular position, angular velocity, and wheel velocity. The primary disadvantage is the need for the complex current control circuitry.

The current control circuitry for each motor involves high power op-amps and is complex from a hardware prospective. Design Team 10 decided to go with a different solution. It was found that PWM was widely used and it was desired to use PID control. Design Team 10 compromised and decided to use PWM with equation C1 taken into consideration along with friction

In Equation C1, I is the current which is given to the dc motor and R is the resistance of the DC windings and K_b is the motor constant and w is the speed of the motor. The desired system transfer function has an Analog Voltage which is corresponding to the target PWM in the numerator and the angular position of the pendulum in the denominator. The denominator is third order even if friction is not taken into consideration, so friction was taken into consideration. The following is MATLAB code which demonstrates the variables and the plant numerator and denominator vectors.

```
Mw=0.295; %Mass Wheel [kg]
Mw_t = 2*Mw; %Total Mass of the Wheels [kg]
```



```

Rw=0.068; %Wheel Radius [m]
Mp=3.283; %Mass of the Pendulum [kg]
Plant=9.81; %Gravitational Constant [m/s^2]
Lp_CenterG=0.1936; %Length to Center of Gravity of Robot [m]
Lp=Lp_CenterG*2; %Total Length of the Rod

```

```

%Inertia Calculations

```

```

Jw = 0.00136; %Inertia of the DFRobot Wheel [kg*m^2]
Jw_t = 2*Jw; %Total Inertia of the Wheels [kg*m^2]

```

```

%Governing Equations Constants Moment of Inertia

```

```

Ja = (2*Jw+(2*Mw+Mp)*Rw^2);
Jb = Mp*Rw*Lp;
Jc = (Mp*Lp^2)/3; %Inertia of the Pendulum [kg*m^2]

```

```

%Torque

```

```

Tc=Mp*Plant*Lp_CenterG;

```

```

%Motor Parameters

```

```

Ka = 0.2723; %Torque Constant
R = 3.1; %Resistance of the DFRobot Motor
w_NoLoad = 15.29; %146 RPM in rad/s
I_NoLoad = 0.23; %No Load Current
Dp = 0.004095; %Damping Ratio
A = Ka/R;
B = ((Ka^2)/R+Dp);

```

```

%Transfer Function of the Plant

```

```

NumPlant = [0 0 -A*Ja/B 0];
DenPlant = [Ja*Jc/(2*B) (Ja+Jb+Jc) -Ja*Tc/(2*B) -Tc];

```

Notice that there is a zero at the origin and the system is negative. The Simulink model of the system is as follows in Figure F.

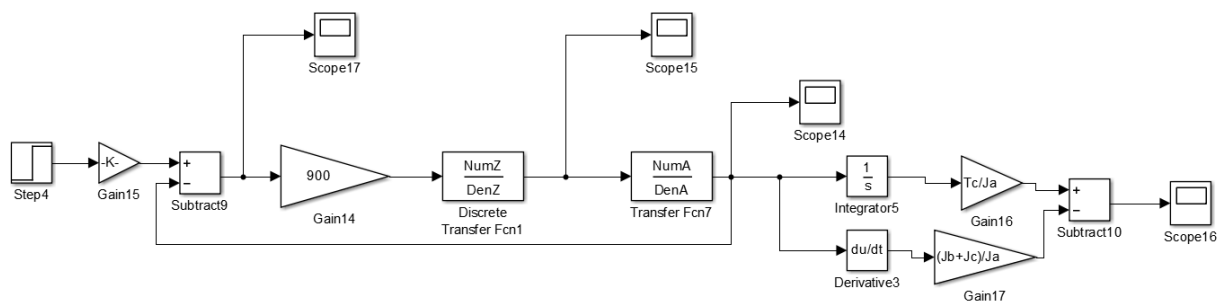


Figure B7: Simulink Model

The NumA and the DenA vectors are plant vectors and are NumPlant and DenPlant as shown in the MATLAB code. NumZ and DenZ is the numerator and denominator vectors of the controller and are [-2.06 1.94] and [1.96 -2.04]. The analog voltage which is the output of the controller was then outputted to PWM. The controller however, did not work upon its immediate implementation, so design team 10 decided to use simple PID control.

The compensator which was derived from the PID approach did not yield good results, so a different and simpler approach was used. Team 10 used a PID controller. Figure sefs shows an example.

The error term is simply the input angle minus the complimentary filter angle value. When the error term is zero for the balancing operation, the error is simply the negative of the complimentary filter output.

Integration with multiplication, differentiation with multiplication, and multiplication are performed on the error term. Integration is performed with the following generic equation

$$Sum = Sum + Error.$$

In Equation A, the above Error term is the Error of the controls loop and sum is the output of the integral. Notice that the above equation is not multiplied by the sampling period. The sampling period is instead taken to the integral gain.

Differentiation is achieved by using the following equation

$$Error_{Slope} = Error - Error_{Last}.$$

In Equation B, the above Error term is the error of the controls loop and the $Error_{Last}$ term is the previous error term and the $Error_{Slope}$ term is the term which is the output of the differentiator block. Equation B would be divided by the period, but the period is instead implicitly inserted into the gain factor.

The proportional part of the controller is a multiplying factor and is analogous to the gains of the integral and proportional parts of the controller. Table 3 shows a listing of the gains.

Gains	Actual Values
Kp Volts/degree	275
Ki Volts/(degree*seconds)	32
Kd Volts*seconds/degree	20

Table 3: Gains vs Actual Voltage Values

D) Software Theory of Operation: Microcontroller Software Theory of Operation (JP):

The development environment for this project is MPLAB X IDE v2.26. The compiler used is the XC16 v1.23. Programming and debugging of the microcontroller were done using the ICD 3. The first step to implementing the software portion of the balancing robot is to include the library for the PIC24FJ128GA010. This is the microprocessor on the Explorer 16. The microcontroller pin layout can be seen in Figure C-1. The programming setup is displayed in Figure C-2.

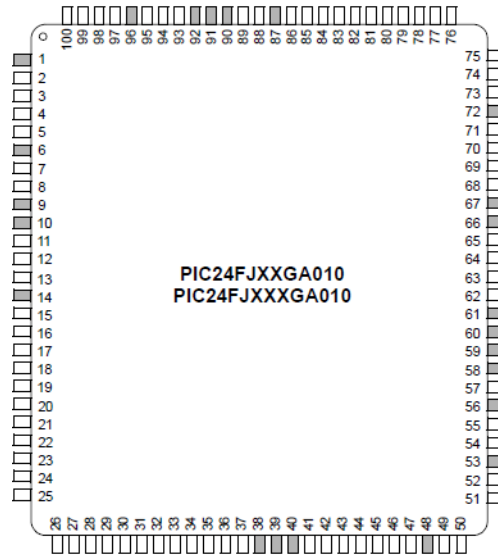


Figure C-1: PIC24FJ128GA010 Microcontroller Pin Diagram

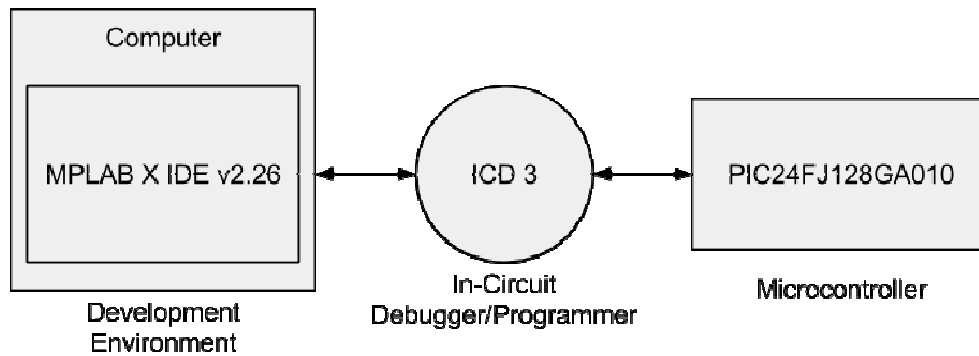


Figure C-2: Programming Diagram for the PIC

The main BalanceBot.c file of the code starts with including all of the libraries necessary for function calls and microcontroller register mapping. The libpic30.h file allows built-in delay functions. Math.h allows the use of the arctangent function necessary for accelerometer data on two axes. The clock frequency is selected to be 16MHz in the config file.

An Explorer 16 macro configuration document was referenced to determine what the macros should be set to. There are two sets of macro configurations for the Explorer 16. In the first set of configurations: the primary oscillator is enabled, JTAG is disabled, code protect is disabled, write protect is disabled, background debug is off, clip-on emulation has been disabled, and the watchdog timer has been disabled. In the second set of macro configurations the following settings were selected: clock switch and monitor is disabled, OSC0 is set to RC15, the HS oscillator is set, and PLL is selected for the primary oscillator. This concluded the configuration macros which are included at the beginning of the main file.

Following the configuration files is the include files. This project is organized as a code library including many source and header files corresponding to interfacing with different sensors and executing functional blocks of code routinely. There are eight included header files. This means that there are eight source files all with their own separate functions which may be called in the main source code.

Global variables play an important role in this project. They are initialized outside of the main function of code. They are used in interrupt service routines for the timer and the rotary encoder external interrupts. In the corresponding rotary encoder source file, these are referenced as extern global variables as they are previously defined in the main file.

The main function of code starts with the initialization of variables of different types for function returns. The IMU returns characters of eight bits. The high and low byte are combined in order to create a short of 16 bits. The PWM duty cycle is always a positive integer and as such it is initialized as unsigned. A forward select Boolean value acts as a flag for calling the PWM source file to determine whether the robot should be moving in the forward or reverse direction. The calculations for this robot were all done in floating point. This simplifies the programming greatly as opposed to fixed point where truncation and round off error are a big concern.

Following variable initialization is the function initialization. Many different functions are called here from their respective source files. The parallel master port is initialized for the LCD, the IMU I2C interface is initialized, the UART2 interface is initialized in order to send serial data to the computer for troubleshooting, and PWM is initialized for setting the correct duty cycle for the motor driver. The IMU registers are then written to. This enables the two axes for the accelerometer and the one axis of the gyro. The other axes are disabled as their data is not important for this particular project. The rotary encoders are then initialized. The external interrupt pins and timer 1 is set for the sampling frequency to be established.

In the infinite while loop of the main code all of the mathematical calculation necessary for achieving dynamic stability is done. The PIC25FJ128GA010 communicates to the accelerometer and the gyroscope in order to receive their measured acceleration and angular velocity. The data manipulation is done in the DA source file. This simply recasts the characters to shorts. After they have been recast the high byte is shifted and added to the low byte. This returns a short from two characters. This is completed for the two axes of the accelerometer and the data for the one axis of angular velocity from the gyroscope. In Figure C-3 is the basic software flow chart for the operation of the code. The right side of the chart contains the infinite while loop which will balance the robot.

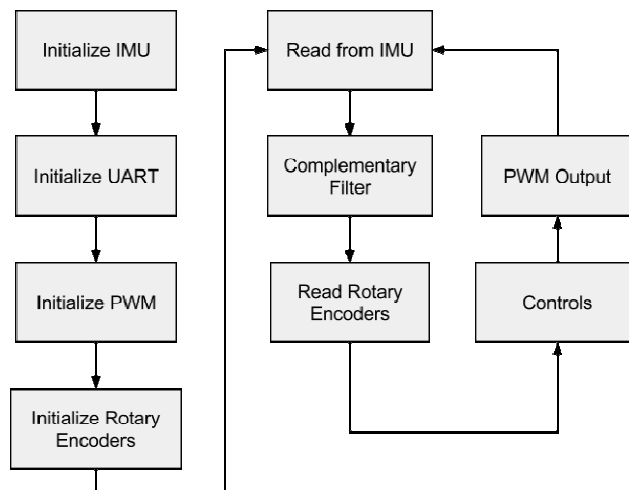


Figure C-3: Software Flowchart for Instruction Execution

Now the microcontroller has raw data. This however is not very useful for a controls implementation. The data is converted into gs and degrees per second. Converting the

accelerometer into gs is completed by dividing the resulting 16 bit binary number by 16384 or 2^{14} . This is because the accelerometer is on a $\pm 2g$ scale. A measured angle is achieved from the two axes of accelerometer data now converted into gs. The atan function return the angle in radians. This value is converted to degrees for use in the controls function.

Following this code is an “if” statement which is only taken when the timer interrupts according to the set sampling frequency. A complementary filter has been added as a source file. This is to combine the measured acceleration value from the accelerometer with the integrated value from the gyroscope. This eliminates high frequency noise values from the accelerometer due to the dynamics of the system. It also eliminates low frequency drift from the gyroscope. The complementary filter requires the previous output for calculation. The controls function follows the complementary filter. This is because once the complementary filter has completed calculating the measured angle, a controls function can be implemented. The controls function takes in a measured angle and returns two variables by using pointers. The two variables the control function returns are a PWM duty cycle and forward flag. These are passed to the PWM function for motor output via the VNH2SP30 motor driver board.

During troubleshooting of the project, different sampling frequencies for interrupt service routine calls were selected. The sampling frequencies were adjusted by changing the register value for timer compare. The three sampling frequencies that were selected were 50Hz, 75Hz, and 100Hz. In the final implementation, 75Hz was chosen as the sampling frequency. This value was chosen experimentally.

Troubleshooting the robot was completed using the UART function. This would pass a redirected printf function to the serial port. Putty was used as a serial monitoring program for the COM1 port to see the printed values. This was done for raw data values from the accelerometer and the gyroscope to verify I2C functionality. This method of troubleshooting was also used for complementary filter returns, controls path testing, forward select, and PWM duty cycles. In Figure C-4, the basic diagram of sending UART messages to the computer is displayed.

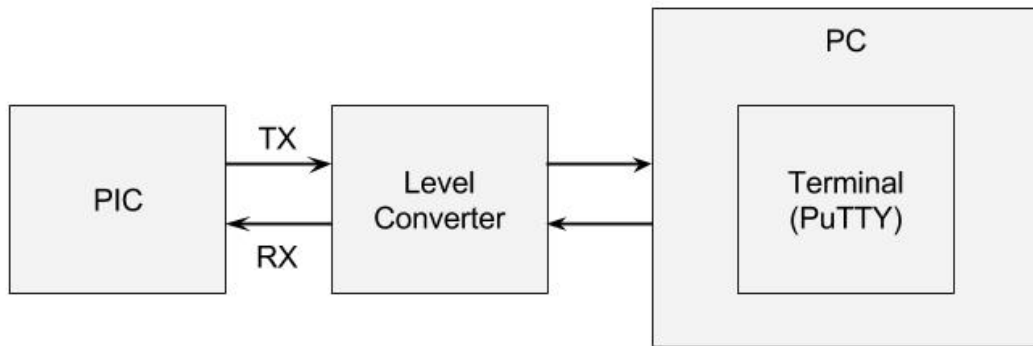


Figure C-4: UART Diagram for PC Output

Originally this project contained additional code for integrated circuits and sensors which were eliminated from the design. Code was written for SPI communication to the accelerometer and the digital to analog converter. The DAC code was replaced by PWM code as the motor driver design for implementation was altered. The DAC code was working successfully to output a corresponding voltage from a given bit input. The inclinometer was also working successfully. SPI communications returned 16 bit words of acceleration data. The inclinometer was deemed unnecessary for the project as the inertia measurement unit already contained an accelerometer.

The repetitive data was unnecessary. Two controls functions were written. One was to implement the pole zero placement transfer function. When testing the bot this implementation proved unsatisfactory and was replaced. The function was completely rewritten for PID control.

Microcontroller Hardware Theory of Operation (JP)

The microcontroller (MCU) chosen for this implementation is the PIC24FJ128GA010. This is a 16 bit MCU. It has one hundred pins. There is a lot of digital I/O pins. Four pins would have been assigned as SPI pins for the inclinometer. Two data pins were assigned for the IMU. The rotary encoders have two data lines each. The rotary encoders take four data pins total. The PIC has 128KB program memory size which will be sufficient to contain all compiled C code instructions. The packaging for this MCU is a 100 pin thin quad flat pack (TQFP) and it is already mounted to the Explorer 16 development board. The interfacing method for the PIC microcontroller to send and receive data to the inclinometer is Serial Peripheral Interface (SPI). To communicate with the IMU and all of the sensor devices on the breakout, the interfacing method will be I2C. Other interfacing methods such as UART, PMP, and external interrupts were used. Input power for the Explorer 16 board is 9 to 15 volts. The board could have been powered from our twelve volt nickel cadmium battery. Instead a common 9 volt battery was used for powering the board. The nickel cadmium battery was solely responsible for providing 12 volt input power to the motor driver board. The general I/O pins are 3.0v - 3.6v. This was a problem for sending logic signals to the motor driver board. The motor driver board operates on 5 volt logic. Logic level shifters had to be used in order to convert the 3.3 volt signals to the 5 volt logic. If the logic level shifters were not used, the threshold values of the motor driver board may not have been reached resulting in undesirable device operation. The microcontroller operates at up to 16 million instructions per second (MIPS). The relevant communication methods the PIC supports I2C, IrDA, LIN, SPI, UART/USART interfacing. This information was obtained from the PIC24FJ128GA010 data sheets as well as the Explorer 16 board documentation and is referenced in the appendix. A basic wire diagram example is given in Figure C-5.

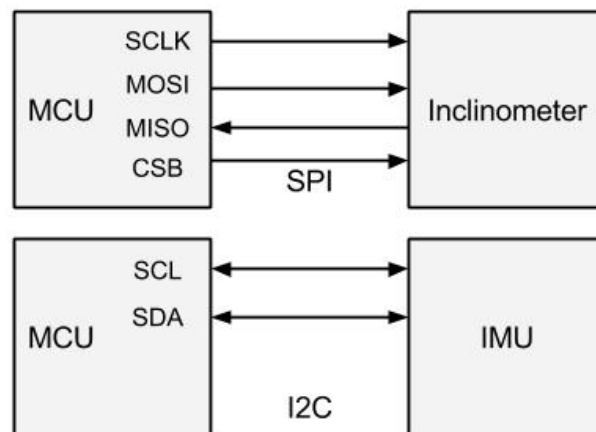


Figure C-5: Interfacing Methods Signal Lines

SPI Bus

- SCLK: Serial Clock (Output from Master)
- MOSI: Master Output, Slave Input
- MISO: Master Input, Slave Output

- SS (CSB): Slave Select

I2C Bus

- SCL: Bidirectional Synchronous Clock
- SDA: Bidirectional Synchronous Data

Rotary Encoder Bus

- INA1 and INB1
- INA2 and INB2

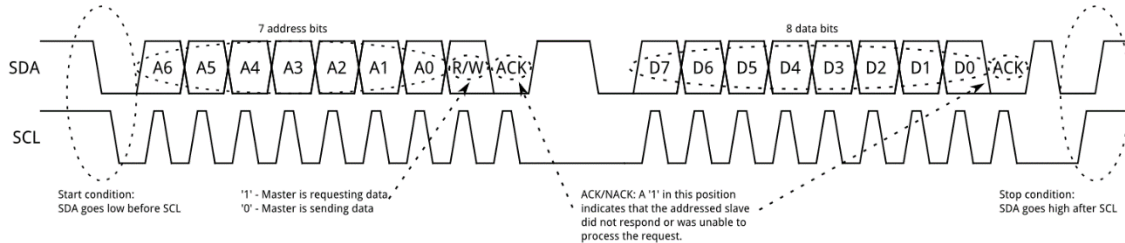


Figure C-6: I2C Timing Diagram

The inertial measurement unit (IMU) uses I2C to interface with the microcontroller while the Inclinometer has a digital output in the form of SPI. For the SPI interface, SCLK is the clock which is an output from the microcontroller to synchronize timings. MOSI is the master output slave input. MISO is the master input slave output. SS (CSB) is the slave select line. When a slave select line is toggled low the peripheral device may begin transmitting data. The select line is then deactivated afterwards and returned to a high state. Only one device may be selected at a time. For the I2C connection, SDA is a bidirectional data path pulled up with a resistor. The level zero block diagram in Figure 12 shows the basic intended operating scheme of the robot.

Hardware Description (DL):

Figure 13 details the sensors that the PIC controller will receive data from. An accelerometer/gyroscope combination, an inclinometer, and a line-sensing array will be connected to the PIC, which will use the information provided to output a voltage-controlled current signal to the DC motors. The inclinometer and line-sensing array that were previously listed here are no longer being used, as the inclinometer was determined to be unnecessary for the controls algorithm that was settled on, and the line-sensing array could not be implemented due to time constraints. A set of rotary encoders was also going to provide feedback with information about the position of the motors, but these were also determined to be mostly redundant, although they would have made the system more precise overall and compensated for some of the drift that the bot experience over time due to back-and-forth motion. However, the inclinometer and rotary encoders were both tested and ensured to be functional, and code was written to enable both to be used and implemented in the project if necessary. Descriptions of the components and their functionality were taken from the respective datasheets, [PIC1], [IMU1], and [Motor1].

Inertial Measurement Unit (IMU):

A L3GD20H 3-axis gyroscope and LSM303DLHC 3-axis accelerometer (as well as a BMP180 barometric/temperature sensor that was not utilized) are combined into one break-out board sourced from Adafruit (product ID 1604). The accelerometer and the inclinometer will be able to counteract drift errors in the gyro when the pendulum is approximately stationary. It is also feared that if the pendulum is tilting by 5 degrees and accelerating at 2.3 m/s^2 , the linear acceleration will tell the inclinometer that it is tilted by significantly more than matches reality. The accelerometer will be used to calibrate the potential tilt error so that the inclination reading is accurate. The gyroscope has different scales of ± 250 , ± 500 , or ± 2000 degrees per second. The minimum scale will be used so that maximum precision is attained. The accelerometer also has different scales which are $\pm 2g/\pm 4g/\pm 8g/\pm 16g$. The sensitivity setting that will be used for the accelerometer is $\pm 2g$, because the robot is not expected to undergo induced acceleration in any direction that amounts to a magnitude close to that of gravity. Communication will be achieved directly over an I2C interface, where both the acceleration and rotational velocity values will be transmitted to the PIC for use in the control algorithm. The IMU will be mounted towards the bottom of the robot so that the accelerometer will not detect the acceleration of the pendulum and only the acceleration of the axis of rotation. The readings of the accelerometer and gyroscope will both be used to compute the current angle of the pendulum using a complementary filter, since the reading from the accelerometer was found to be quite noisy and that of the gyroscope to drift significantly.

Inclinometer:

The inclinometer that was going to be used for this project was the single-axis Murata SCA830-D07-1, which integrates high accuracy micromechanical acceleration sensing with SPI interfacing. It has a $+90$ -degree to -90 -degree measurement range, which will clearly allow for the robot to detect if it has been knocked over. Sensitivity is 0.00179 degrees/count within ± 3 degrees, which should be more than adequate. It is expected that the inclinometer will give identical performance as when the robot is accelerating as when the robot is stationary, but in the event that linear acceleration can throw off the inclinometer readings when accelerating, the accelerometer will be there to calibrate the inclinometer. The inclinometer should be mounted as near the axis of rotation as possible, so as to prevent the readings from being skewed by the acceleration due to the pendulum, given that this sensor measures based on acceleration.

IR sensor array:

The original goal of this project was to implement an infrared combined emitter and sensor line-tracking array $3/4$ " from the surface on which the robot travels, allowing the robot to detect the line it is following. The Pololu QTR-8A Reflectance Sensor Array was going to be utilized, which uses 8 infrared LED/phototransistor pairs spaced 0.375 " apart, and includes a MOSFET to facilitate power-saving. The sensor could be read by applying a $3.3V$ voltage to the input and timing the decay of the voltage output, which is inversely related to the amount of reflection. The signals could be read as a timed high pulse, and no analog-to-digital conversion would be required, which would provide greater sensitivity than if an analog output with a voltage divider (for example) were being used. The analog output voltages would be read as digital inputs. The independent outputs would be fed into the PIC and processed using a line-following algorithm

that would allow the robot to determine if it has started getting off course, if it is at an intersection, or if it encounters a signal to slow down and stop [11]. The supply current is expected to be 100 mA or less. The sensors would not be at their optimum sensing distance, so the output will be skewed so that it will not approach 3.3 V. The output of the sensors at the given distance would be experimentally determined. The internal comparator of the PIC would have one of its pins set to the experimentally determined value and the sensing pin of the internal comparator would be multiplexed to each pin connected to the output sensors at 100 Hz so that each line sensor could be read every 10 ms. When the internal comparator goes high for a given pin, the robot would know which line sensor has been tripped.

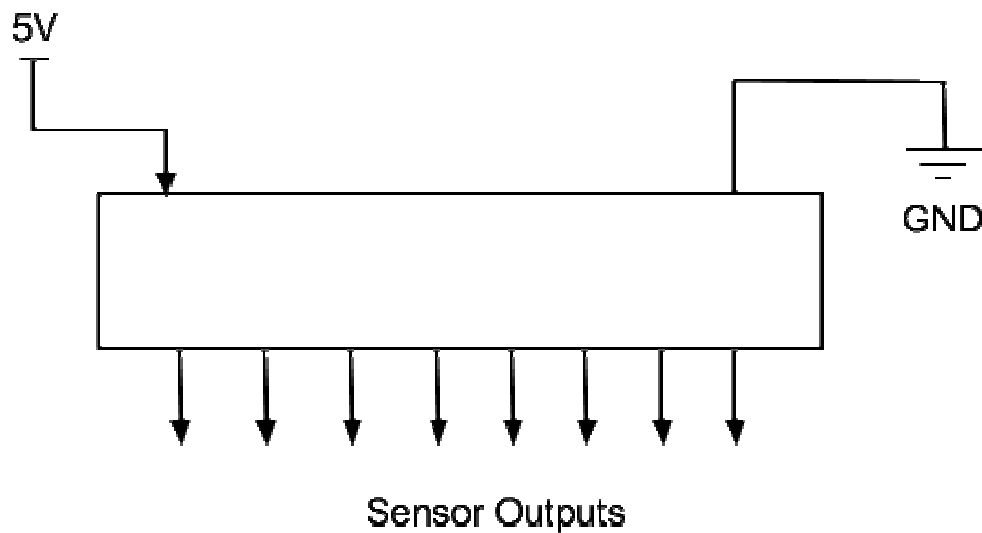


Figure D1: Line sensor array

Optical phototransistor sensor:

A payload sensor was planned to be implemented on the top of the robot in the form of a phototransistor with a 2k Ohm resistor at the output voltage, with a floating base, allowing the sensor to detect the high level of contrast between having an opaque object on the top of the robot and having no payload, and it would also provide a more digital output signal to the PIC. The state of this signal (high or low) would allow the robot to determine whether it is carrying a load or not, and in certain circumstances whether it should be moving or not (for example, if the user wants to use the placement of a payload on the robot to signal it to start the delivery). Given a 5V signal, the current required for this sensor would be 2.5 mA.

Motors:

The robot uses two brushed 12V DC motors from DFRobot that include integrated two-phase hall encoders that can provide a 633-pulse-per-rotation signal (or 0.54-degree sensitivity), which should be a fine enough resolution to meet control requirements if these should need to be utilized for future applications. These motors are extremely compact, and have a weight of 270 grams each, a mere fraction of the total weight specified in the design requirements. The heat

dissipation capacity, reduced power loss, and high output power are expected to be well worth the relatively higher cost. The motors also provide 0.98 N-m of torque, which will be more than sufficient based on the simulations performed, which show that the maximum required torque per motor will be 0.377 N-m (Figure 20). The no-load current is 0.23A, which is significant, and may somewhat compromise efficiency, while the maximum current is 3.6A, which plays a large part in dictating the power and battery requirements for the system. These motors were set aside in favor of either a 30:1 or 50:1 12V motor from Pololu that would have had a bit more speed but somewhat less torque. This was due to the testing of the DFRobot motors, which sometimes made an unsettling clicking noise under load, and did not seem to offer linear torque-speed characteristics that were a high priority when using the more intricate state-space controls system that was also under consideration earlier in the project. Later on, once the controls scheme was redesigned, questions arose over whether sufficient torque could be provided by the Pololu motors, and also over their quality, since one of them broke, and it was decided to begin testing with the DFRobot motors and see how they performed, since one of the noisy motors had been replaced by the supplier.

Motor driver board:

Early in the semester, the decision was made to abandon the motor driver circuitry that had been previously designed, which utilized a DAC followed by two op-amps, as the large op-amps were overly costly, the circuitry unnecessarily finicky and complex, and research into pulse-width-modulation-driven DC motors showed that it would most likely offer more precise and accurate control. Thus, the Dual VNH3SP30 Motor Driver Carrier (MD03A) from Pololu was used, being a low-cost, compact unit that is perfect for driving two high-power motors on a medium-sized, differential drive robot. Hardware is robust, with a maximum current rating of 30 A, plus current-limiting resistors and a FET for reverse battery protection. All that needs to be added is a microcontroller (or similar control circuit) to turn the H-bridges on and off. The board is powered with a +5V input supply, with the control connections being made at the other end of the board. Two large, radial capacitors limit disturbances on the main power line.

Logic level shifters:

To mitigate the challenge of converting the 3.3V to the 5V logic level of the motor driver board, two bi-directional logic level shifters were used. These components simply have four pins on the high side that can be converted to the four pins on the low side, or vice versa, with signal inputs and outputs for each side.

Controls:

Before implementing the system, design specifications which are controls related were made so that the system would perform as it is meant to. The design specifications were chosen so that the only way the design specifications would not be met is if the system would malfunction or perform poorly. Table 4 shows a list of controls related design specifications.

Label	Design Specification	Maximum	Minimum
A	Steady State error	10%	10%
B	2% Settling Time	7s	
C	Percent Overshoot	83.8%	
D	Gain Margin		1.5
E	Phase Margin		15 degrees
F	Maximum Tilt		10 degrees
G	Maximum Line Deviation		8 cm

Table 4: Controls Related Design Specifications

Design specification A deals with the steady state error of the controls system with the robots velocity as the input and the robots velocity as the output. If the robot reaches its final destination 10% slower or 10% faster than expected customers will not be upset and the actuators or motors will have no significant stress put on them. Design specification B deals with the 2% settling time of the robot and is long because, as the settling time gets longer, the system becomes more stable which is evident in figure 21. Design specification C was chosen at a value at which it is anticipated that the actuators would saturate if the robot is traveling at 1 m/s. Design Specification D was chosen so that the system could have a very low gain margin. This was done because the system is highly unstable and a high gain margin might not be possible. The minimum phase margin was also selected at a low value for the same reason. Design specification G deals with how far the center of the balance bot can deviate from the white line. It is anticipated that the line sensor array will be about 4 cm in either direction so the maximum command that the line sensors will give is to go 4 cm towards the line. If the robot is 8 cm from the line and 4 cm from the edge of the line sensor, then the robot will interpret its sensor information that it was reached its destination when it is not close enough to the line to read its position and will thus likely not reach its destination.

In developing the controls for the system, the system had to first be modeled. There are two sets of controls, one of which is the longitudinal controls which give the robot a velocity and the lateral controls which determine the position of the robot. First it was needed to develop the model of the system so a compensation technique could be developed. First a force diagram was given for each wheel and the equations which were developed.

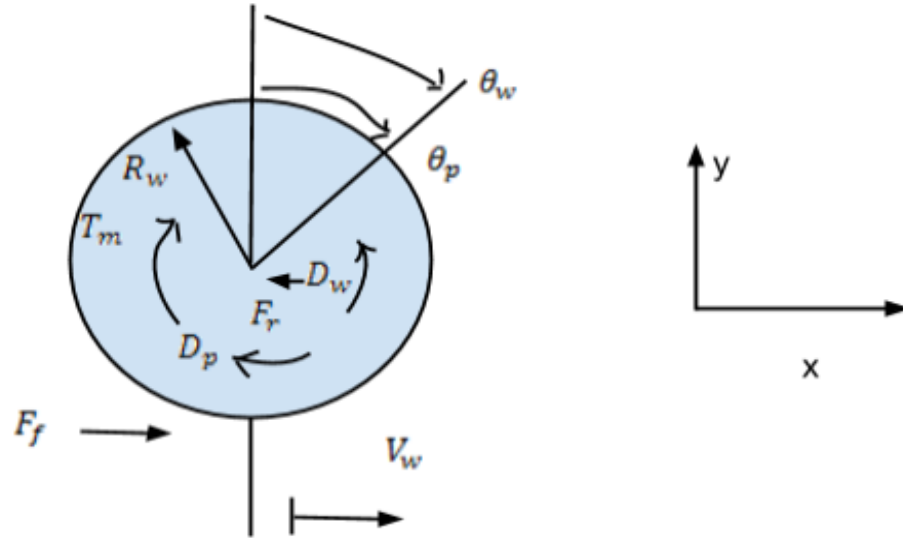


Figure D2: Wheel Free Body Diagram

In the Figure D2, the angular position of the wheel is denoted by θ_w and the angle of the pendulum is denoted by θ_p . The reaction force is denoted by F_r . The pendulum exerts a force, F_r , on the wheel due to its acceleration. V_w is the velocity of the wheel in the x direction, T_m is the torque acting on the wheel from the motor and D_p is the frictional damping due to the velocity and D_w is the frictional damping due to the motion of the wheel. F_f is the frictional force acting on the base of the wheel. R_w is the radius of the wheel.

The equations of motion are as follows and equations below describe the angular forces acting on the wheel.

$$(1)$$

In equations 1, J_w is the inertia of the wheel and will be experimentally determined D_p and D_w are D_f . For the sake of simulations, J_w was calculated as follows.

$$-$$

$$(2)$$

The linear forces that are acting on the pendulum are modeled in equation 3.

$$(3)$$

In equation 3, X_w is the position of the robot in the x direction. To eliminate X_w so that the system could be solved, equations 4 developed the relationship between the angle of the wheel and the position of the robot.

$$(4)$$

Because the lateral controls only serve to add a differential torque to one wheel and subtract that same torque from the second wheel, the average force which the wheels are exerting on the pendulum is the same as if the wheels are both exerting the same force on the wheels. Whatever force is subtracted from one wheel is added to the second wheel so that the force acting on the pendulum is unaltered.

Figure D3 is a diagram of all of the forces which are acting on the pendulum.

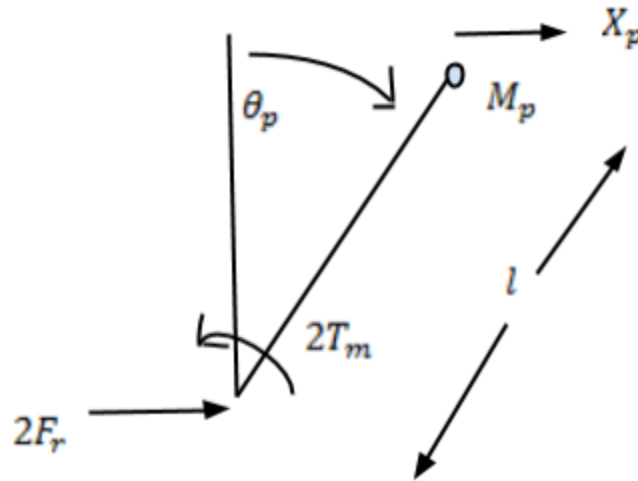


Figure D3: Pendulum Free Body Diagram.

In the figure D3, M_p is the mass of the pendulum and l is the length from the axis to the center of mass of the pendulum. The equation which describes rotational forces about the axis of rotation of the wheels is given in equation 5.

(5)

In equation 5, J_c is the inertia of the pendulum about the axis of rotation, and is given by equation 6. The constant g is the equal to 9.81 N/kg

-

(6)

Equation 7 is non-linear and can be made linear by assuming that the angle of the pendulum is small which means that the sine of θ_p is θ_p .

(7)

It is still needed to eliminate F_r so the relationship described by equation 8 is used.

(8)

In equation 8, X_p is the position of the center of mass of the pendulum. X_p is given by the following equation.

(9)

In equation 9, a small angle approximation was made so that $\sin(\theta) = \theta$. Notice that the distance that the center of mass of the pendulum is the distance which the wheels travel added to the distance which the pendulum rotates.

The variables which determine the state of the system are the motor torque, the angular positions of the pendulum and the wheels. Since the velocity of the wheels is the command input, the angular position of the robot is written in terms of linear velocity. It is clear that the angular velocity of the wheel, the angle of the pendulum and the angular velocity of the pendulum can be measured. This system could be compensated via state variable feedback and velocity can be the command input. The angular velocity of the wheel can be easily converted to linear velocity.

Also Torque is related to current by a factor of K_a so the input to the system could be generated. Current rather than torque is the input to the system.

The state variable equations are of the following form.

$$\dot{x} = Ax + Bu \quad (10)$$

$$y = Cx \quad (11)$$

Before equations 5 through 8 are manipulated so that they are of the form of equations 10 and 11, it is useful to look at the equations without the damping terms so that they can be simplified before they are converted to their state variable representations. The two governing equations given by equation 12 and 13.

$$(2J_w + (2M_w + M_p)R_w)\theta_w s^2 + M_p R_w l \theta_p s^2 = 2T_m \quad (12)$$

$$(J_c s^2 - M_p g l)\theta_p = -2T_m \quad (13)$$

Equations 12 and 13 can be simplified into the following form. Equations 14 and 15 correspond to equations 12 and 13.

$$J_a \theta_w s^2 + J_b \theta_p s^2 = 2T_m \quad (14)$$

$$J_c \theta_p s^2 - T_c \theta_p = -2T_m \quad (15)$$

Using J_a , J_b , J_c and T_m , the equations 12, 13, 14, and 15 may be written in the following form.

$$\begin{bmatrix} \dot{\theta}_p \\ \ddot{\theta}_p \\ \dot{V}_w \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{T_c}{J_c} & -\frac{2D_f}{J_c} & \frac{2D_f}{J_c R_w} \\ -\frac{J_b T_c R_w}{J_a J_c} & 2\frac{D_f R_w}{J_a} \left(\frac{J_b}{J_c} + 1\right) & -2\frac{D_f}{J_a} \left(1 + \frac{J_b}{J_c}\right) \end{bmatrix} \begin{bmatrix} \theta_p \\ \dot{\theta}_p \\ V_w \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{2K_a}{J_c} \\ \frac{2K_a R_w}{J_a} \left(1 + \frac{J_b}{J_c}\right) \end{bmatrix} I \quad (16)$$

Notice that in equation 16 the input vector is multiplied by K_a so that the input is switched from torque to current. Equation 17 is the system matrix.

Equation 17 is of the MatLab calculated system matrix with its numerical value.

$$\begin{bmatrix} \dot{\theta}_p \\ \ddot{\theta}_p \\ \dot{V}_w \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 14.71 & -0.0052 & 0.0748 \\ -13.3477 & 0.0274 & -0.3910 \end{bmatrix} \begin{bmatrix} \theta_p \\ \dot{\theta}_p \\ V_w \end{bmatrix} + \begin{bmatrix} 0 \\ -0.348 \\ 1.8194 \end{bmatrix} I \quad (17)$$

If state variable feedback alone is use, then the system will not have zero error, so a method called integral control was used. In integral control, the feedback gains which alter the A matrix compose the inner loop and the outer loop is composed of an integrator and a gain after the error section of the loop. The output of the system is the velocity.

Because the outer loop was modeled with an integrator in the loop transfer function, the poles were chosen to be at -3.6 , $-15-j*6.5$, and $-15+j*6.5$. The gains which were chosen for the system are $K1 = -1.1206$, $K2 = -323.9$, and $K3 = -43.7$. Figure D4 shows the system root locus after the poles were placed. This root locus is for continuous time.

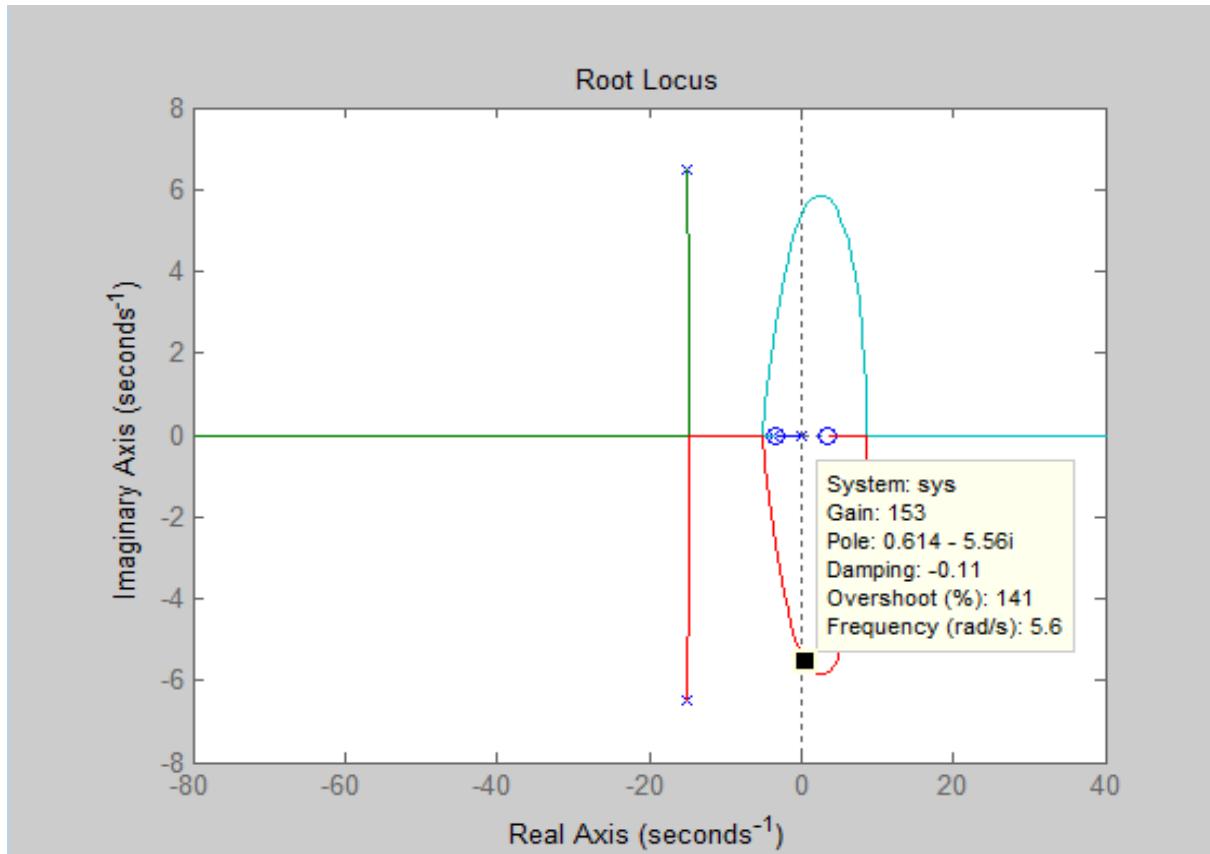


Figure D4: Root Locus of Longitudinal Controls

Notice that in figure D4, the integrator pole goes and meets the zero, and the two complex poles come in and then branch out towards the zero and positive infinity. It was decided that the gain should be chosen so that the poles are at the breakout point which is closest to the left zero. The gain which was chosen was 25. Figure 18 shows the stability regions at the given gain.

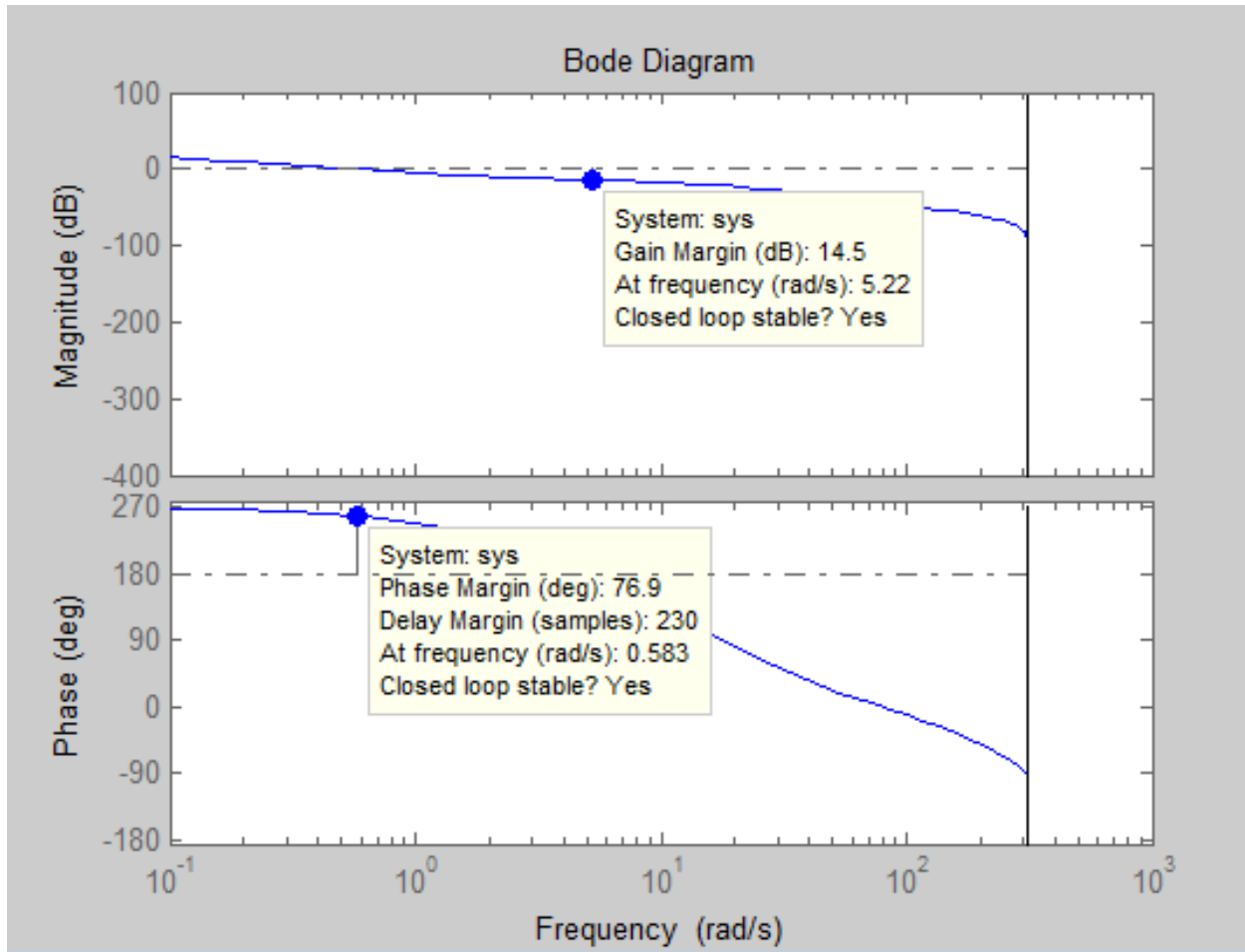


Figure D5: Bode Plot of Longitudinal Controls

Notice that the gain margin is about 5.3 which is excellent considering the instability of the system. Notice that the phase margin is 76.9 degrees which is also excellent. In the Simulink simulation, a gain of 100 was tried and it was found that the system was unstable. The actual gain and phase margins cannot be determined from the MATLAB bode plot. The actual gain margin is expected to be about 2.6 based on experimenting with the Simulink simulation. The reason might be that when the Simulink simulation was ran, everything but the integrator and the zeroth order hold was continuous and in reality the continuous time simulations will be less stable than when everything is discrete as was the case in MATLAB. Figure 18 shows the Simulink block diagram of the system.

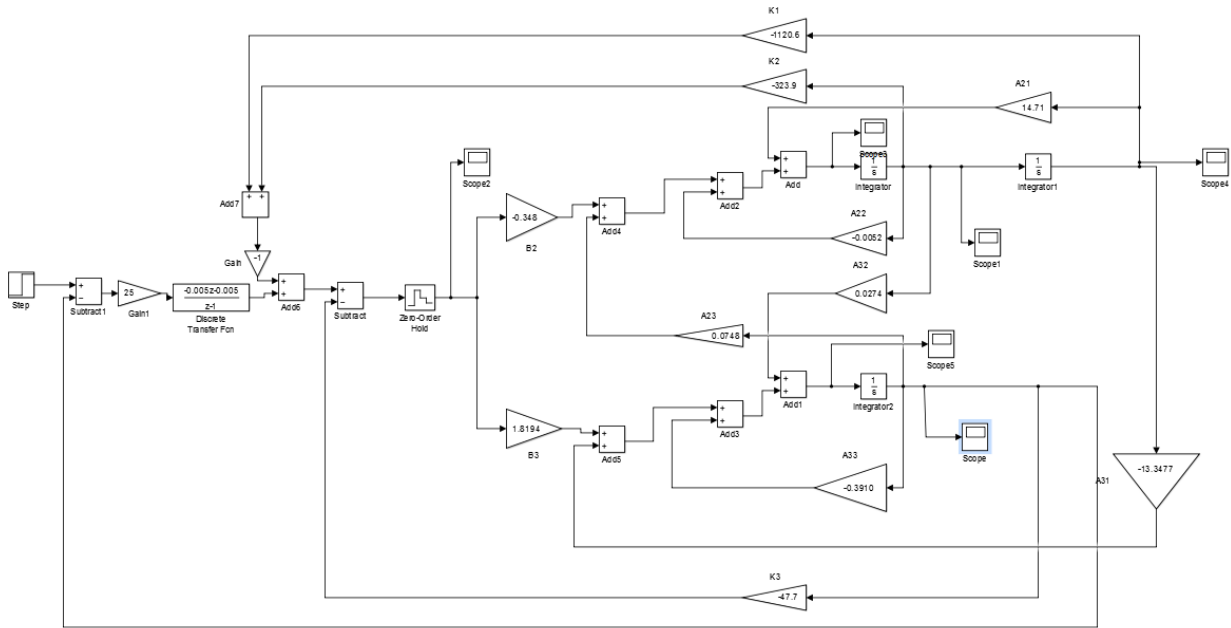


Figure D6: Longitudinal Controls Loop Simulink Diagram

Notice the discrete integrator and the gain at the outer loop. Also notice the zeroth order hold which is located at the motor input. Aside from those two discrete blocks the rest of the system is continuous. Figure 19 is of the step response of the system with velocity at the output and 1 m/s as the command input.

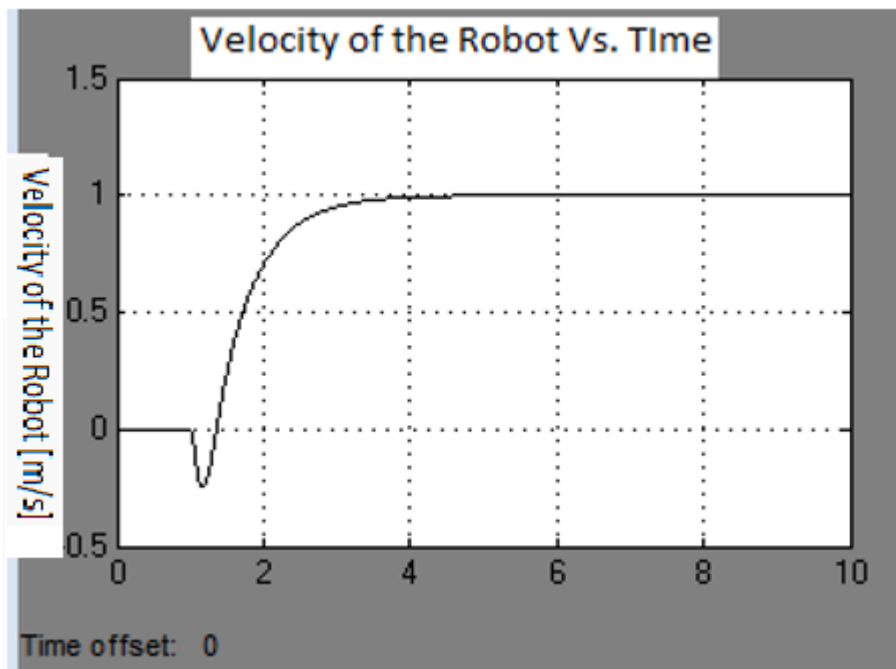


Figure D7: Step Response of Longitudinal Controls with Velocity as the Output

Notice that the 2% settling time is at about 3.1 seconds, and there is zero steady state error. It is undesirable to have a fast settling time because as shown from the root locus plot, the faster the settling time the smaller the gain margin. Figure 20 is of the current output of one of the motors.

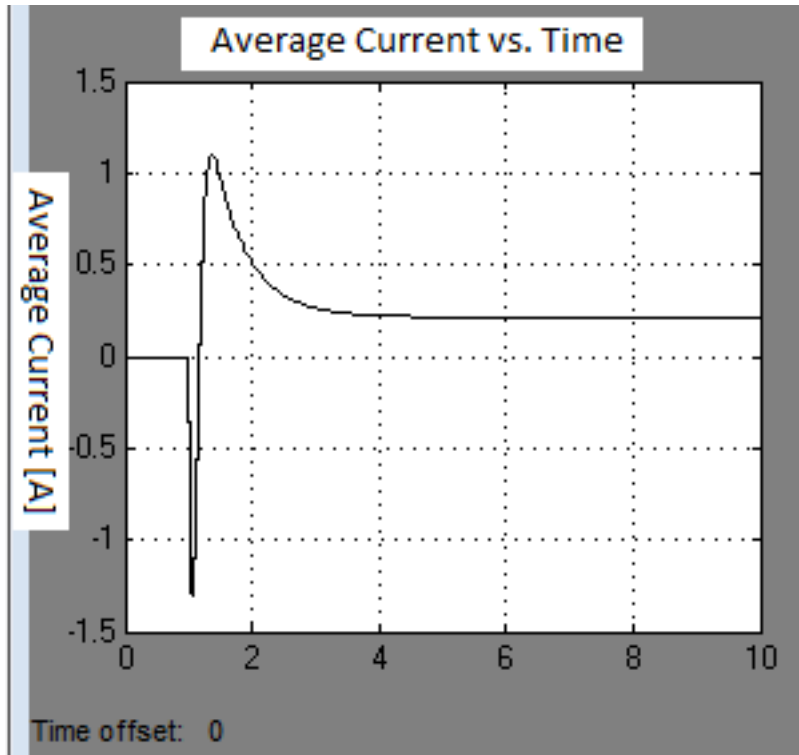


Figure D8: Step Response with Current as the Output

Notice that with the sampling frequency of 100 Hz, the curve for the current of the motor appears continuous even at the down spike. Because the torque is the current times K_a , the torque which is applied to the system can be deduced. In order to get the robot to move backwards a negative torque is applied and then the torque goes positive once the robot moves forwards. The torque peaks as the robot is accelerating the most and then dies down when the only force acting against the robot is friction. Notice that friction is not negligible and would not be negligible if the actual friction was half of the theoretical value. Before calculating the theoretical friction it was necessary to calculate the torque constant which is shown in equation 18.

$$\text{---} \tag{18}$$

In equation 18, T_{stall} is the stall torque and I_{stall} is the stall current. The value of k_a was found to be 0.29. The following equation gives the damping term.

$$\text{---} \tag{19}$$

Equation 19 was derived from the fact that the damping term times the angular velocity is equal to the no-load torque. I_{nl} is the no-load current and ω_{nl} is the no-load velocity. Figure 21 is of the angular position of the pendulum.

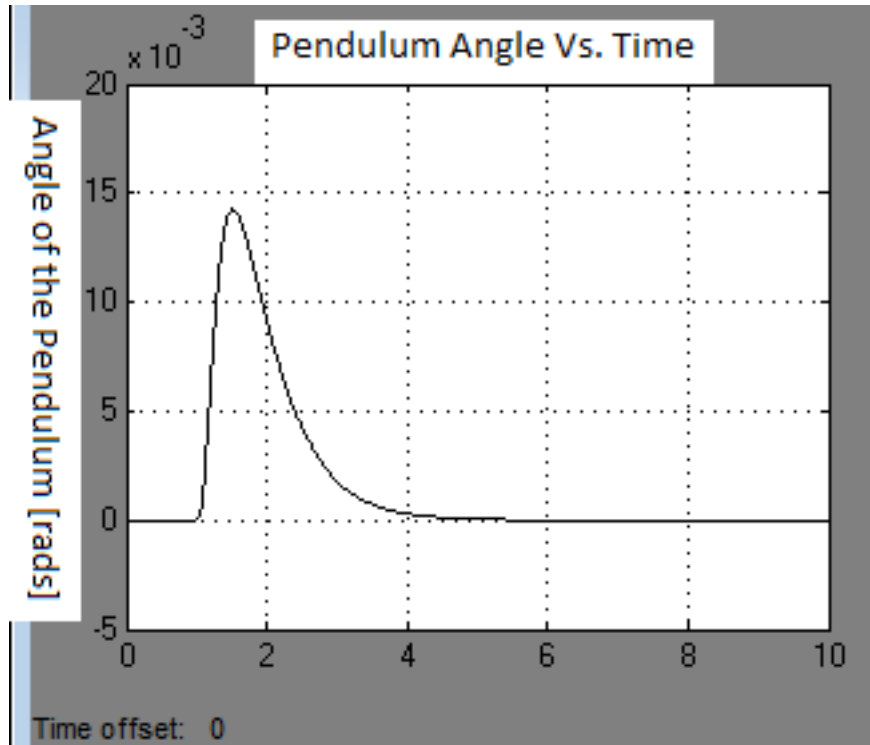


Figure D9: Step Response with Pendulum Angle as the Output

Notice that in the above figure, the pendulum will shift to about 0.83 degrees and then it will shift back to about zero. If the velocity is reduced to $\frac{1}{3}$ of its simulated value, then the maximum degrees deviation will be reduce to $\frac{1}{3}$ of the initial degree deviation value. It is foreseeable that the pendulum will deviate more than figure 21 is showing in actuality due to initial conditions and inaccurate sensor readings but according to Dr. Veillette of The University of Akron, the control loop will still perform its function if it is designed correctly. In figure 22, the zoomed in version of the angular position plot was observed.

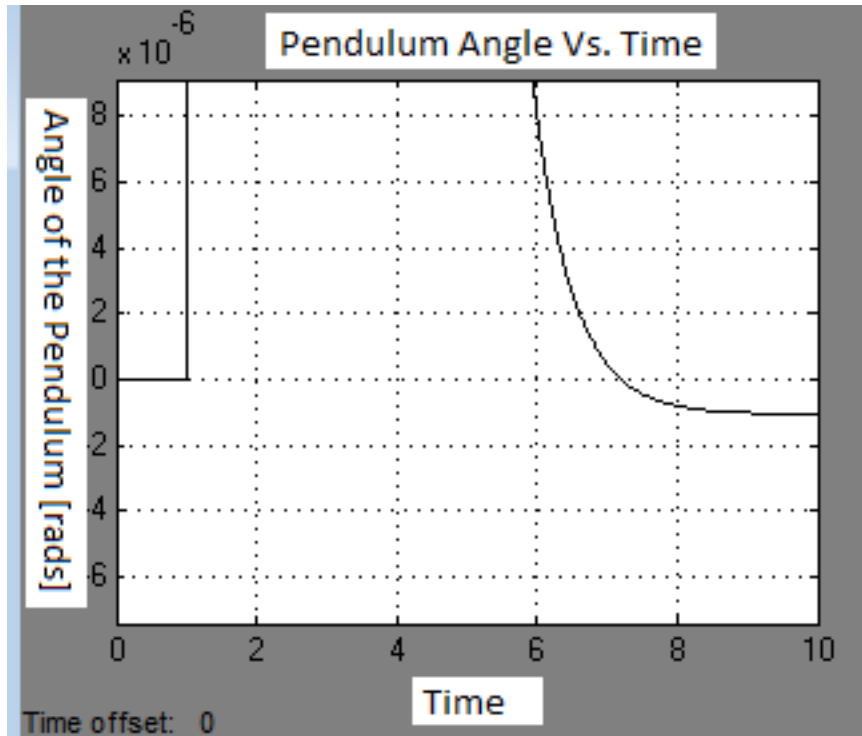


Figure D10: Zoomed in Step Response with Pendulum Angle at the Output

Notice that the pendulum angle is negative when the robot has reached its maximum velocity but it is negative by about 1.14×10^{-4} degrees. This is the case because there is a torque which the angular velocity damping term of the robot is placing on the pendulum. Gravity must be used to counteract this force so that the pendulum is not in motion and the velocity of the robot is constant. In reality, the pendulum will not remain fixed at that small deviation in angle but it is anticipated pendulum angle will oscillate in one direction when the damping force becomes excessive and in the other direction when the counter torque becomes excessive. Figure 23 is of the angular velocity of the robot.

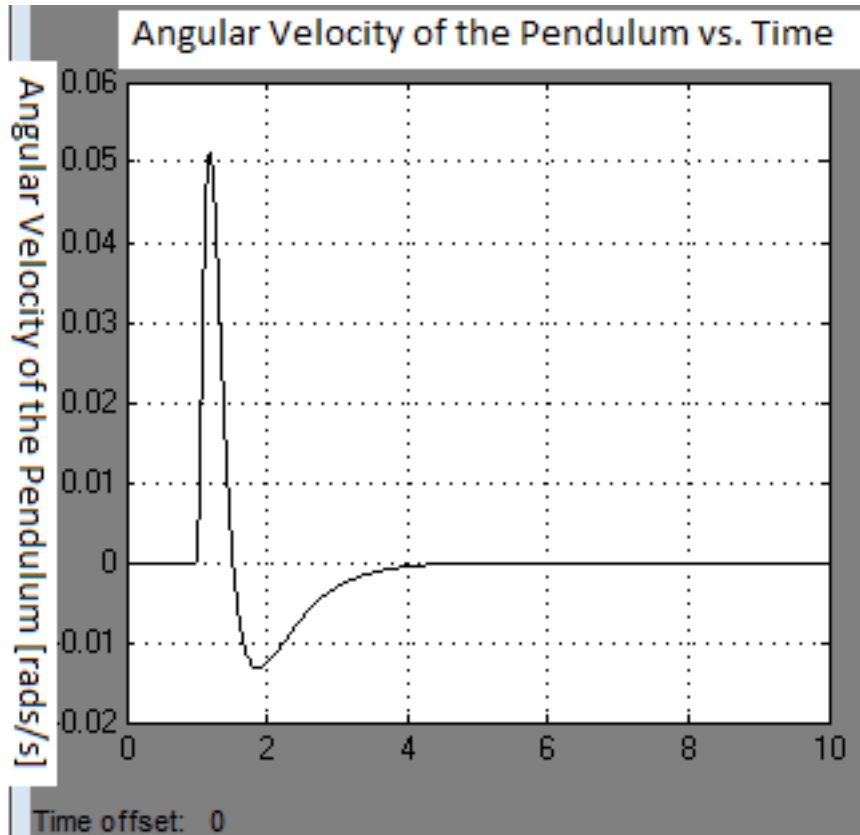


Figure D11: Step Response with Pendulum Angular Velocity as the Output

Notice that the angular velocity of the robot is positive as the robot tilts forward and negative as the pendulum approaches zero and zero as the robot reaches its steady state. In actuality it is anticipated that the pendulum velocity will oscillate around zero due to the damping of the velocity. The angular velocity of the robot deviates to about 2.86 degrees per second at its peak deviation, so the sensors will not be stained by too much since they can detect about 114 bits per degrees/second.

Before developing the longitudinal controls for the system, it is necessary to create a system model. Figure 24 shows an image of the model of the system.

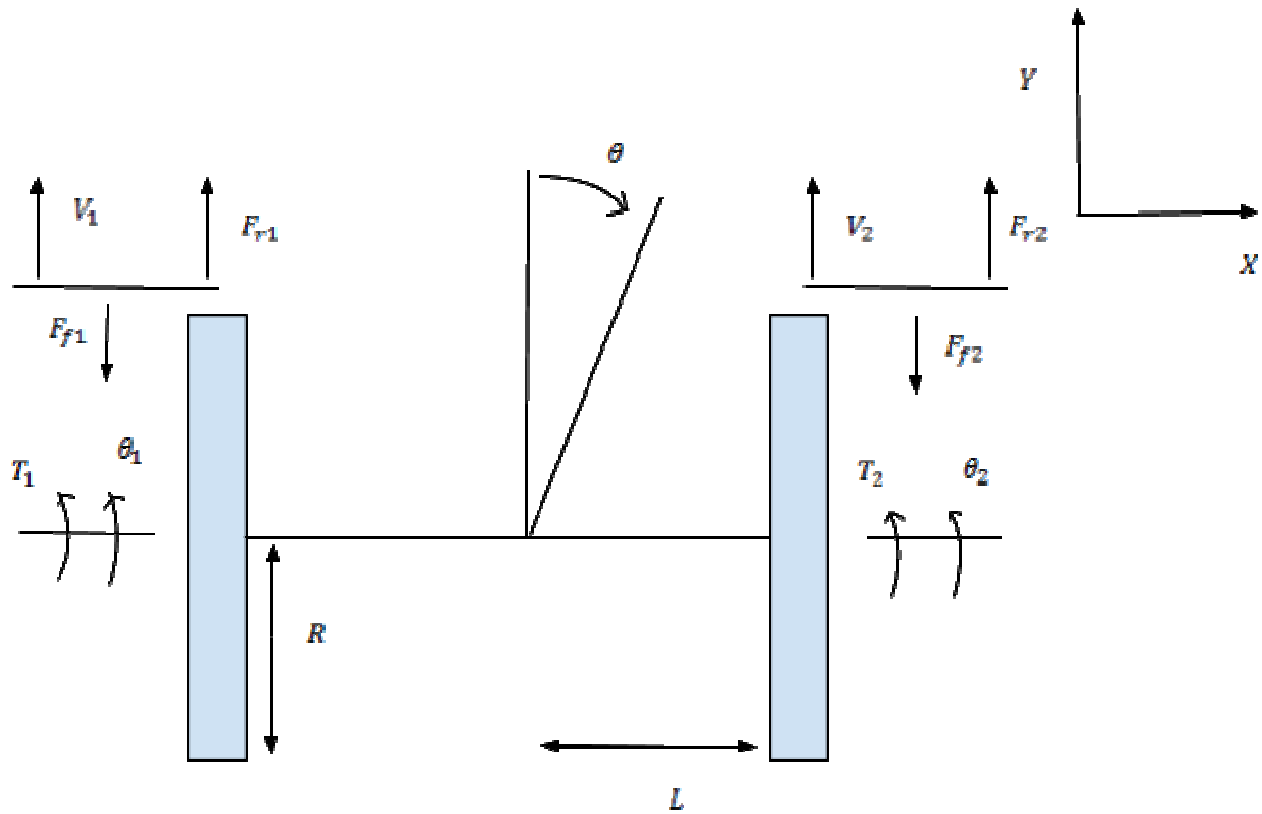


Figure D12: Free Body Diagram for the Lateral Controls

In the above system, the two wheels are the grey boxes and F_{f1} and F_{f2} are the frictional forces which are acting on wheels 1 and 2 respectively. T_1 and T_2 are the torques which are acting on wheels one and two respectively and V_1 and V_2 are the velocities of wheels one and two respectively. θ is the angle of the robot forward direction with respect to the y axis. R is the radius of each wheel. F_{r1} and F_{r2} are the reaction forces due to the twisting of the body of the robot which are acting on the wheels. Equation 20 shows the relationship between the difference in the angular position of the wheels and the θ .

$$- \quad - \quad - \quad (20)$$

The small angle approximation that the sine of an angle is equivalent to the angle is used. Equation 21 shows the relationship between the velocity in the x direction and the difference in angle in between the two wheels.

$$- \quad (21)$$

The parameter v is the average velocity of the robot which is assumed to be constant at 0.333 m/s so that the calculations are simplified. Moreover, it is anticipated that the robot will not leave the line until the robot has reached its actual speed due to the line initially being straight. The \dot{x} term denotes the derivative of x . Equation 22 gives the moment of inertia for the body of the robot as the robot is twisting at an angle θ . The robot is modeled as a rod, but the actual moment of inertia will be experimentally determined.

$$-$$
(22)

The equation which relates the reaction force to delta theta is equation 23.

$$-$$
(23)

Delta Fr is the differential force which is acting on the wheels. Equation 24 shows the rotational forces acting on the wheels.

$$(24)$$

Notice that every term in the equation is a torque. Equation 25 shows the linear forces acting on the wheels.

$$(25)$$

Fr and Ff are eliminated in equation 26 as follows.

$$-$$
(26)

Notice that in equation 26, the torque term was replaced with the current times the torque constant.

All equations which were relevant to the longitudinal controls system were put in state variable form as is shown in equation 26

$$-$$

$$-$$
(27)

Notice that differential current is the input of the matrix and there is a row of zeroes of row three of the A matrix which implies that the system has an integrator. The output of the system is the distance in the x direction from which line sensor is triggered to the center of the robot. The x variable is also generated the input to the system. Integral control state variable feedback was used for the system and the control loop is shown as in figure 25.

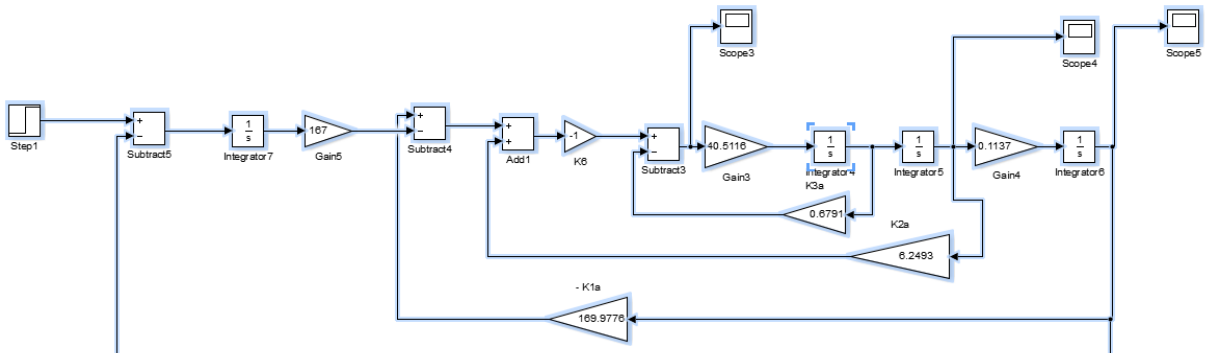


Figure D13: Lateral Controls Loop Simulink Diagram

The gains K1a through K3a are 169.9776, 6.2493 and 0.6791. The gain which is to the right of the outer loop integrator is 167. The loop is also in continuous time, which can be done because

the system is naturally stable. There is no harm in closely approximating what the differential current will be because the line sensors are giving a digital approximation of x anyways. Figure 27 shows the interaction of the two control systems and all of the feedback loops.

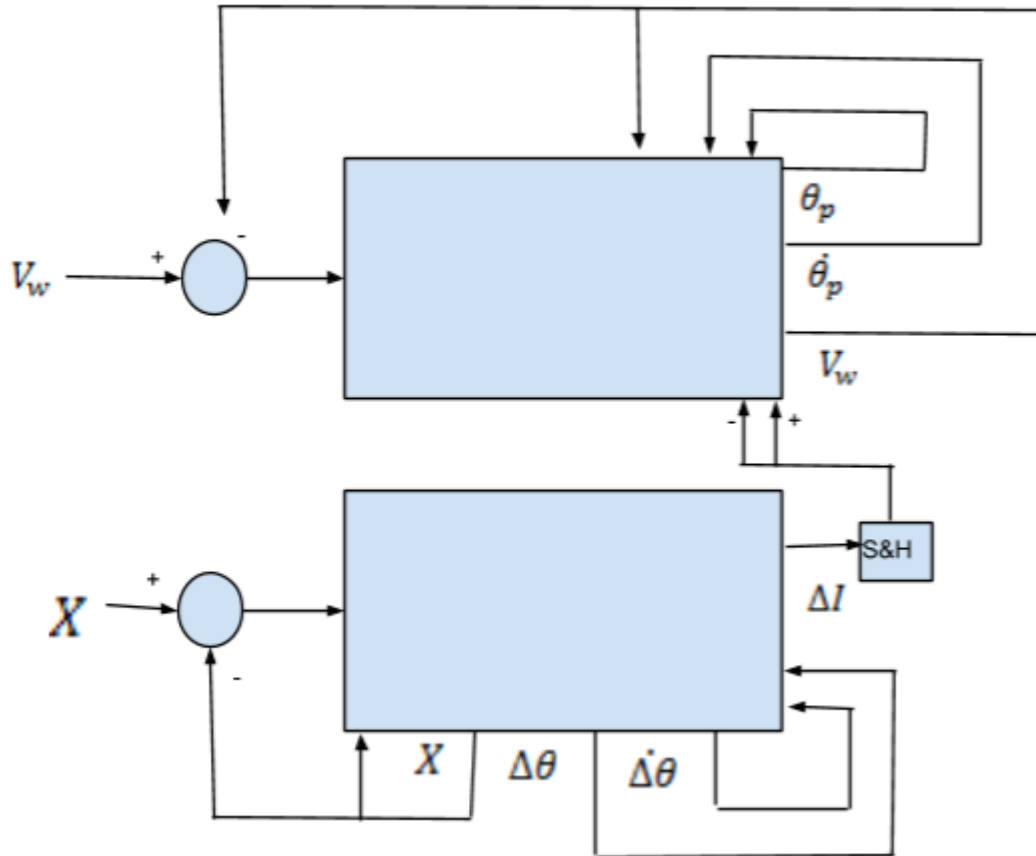


Figure D14: Longitudinal and Lateral Controls Interface

The S&H block is a sample and hold block. The sample and hold block will be synchronized with the digital longitudinal controls system and at an identical sampling frequency. Notice that the differential current is both added and subtracted from the longitudinal controls or more specifically the left and right motors respectively. To see what is meant by the left and right motors of the balance bot refer to figure 25.

Figure 28 shows the step response of the lateral controls with x position as the output.

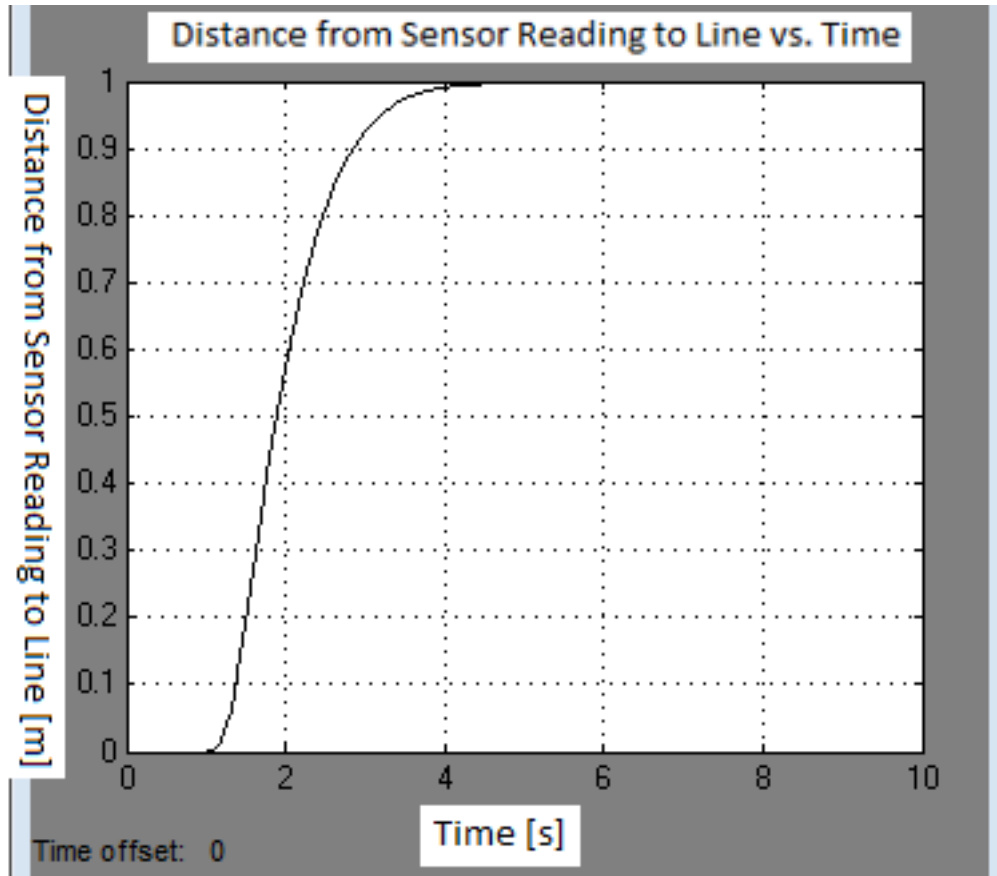


Figure D15: Lateral Controls Step Response

Notice that the 2% settling time of the step response is about 3 seconds. It was decided that it is more important that the robot would not exert a torque on the on the pendulum which would give the inverted pendulum instability than there be a fast settling time. Notice that the steady state error is also zero which is needed. In the actual robot design, the command input to the lateral controls will be 0 m so that when the sensor detects that the robot center is 3 cm from the line, the robot will move back towards the line until the two middle sensors are triggered. Figure 29 shows the step response of the longitudinal controls with differential current at the output.

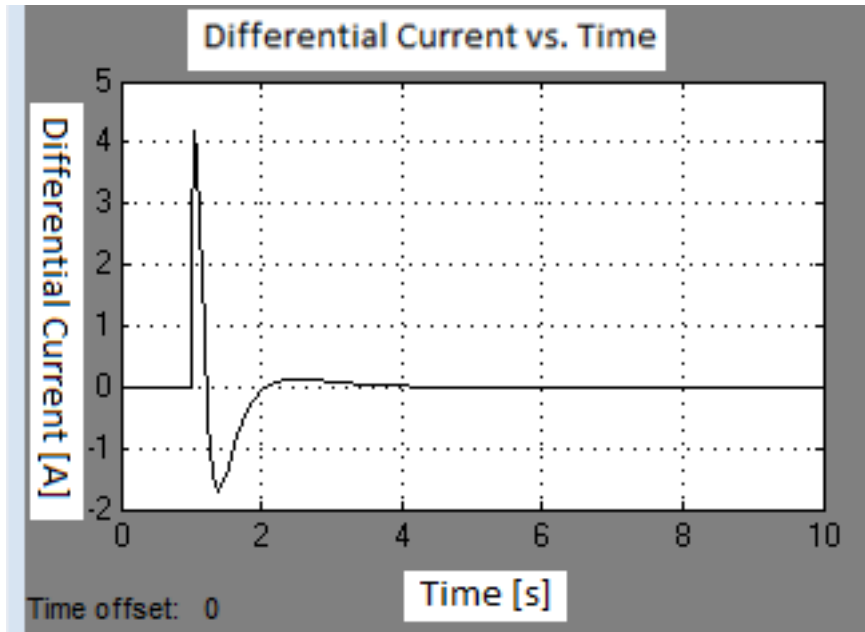


Figure D16: Step Response of Lateral Controls with Current as the Output

Notice that the current peaks at 4.2 amps and then oscillates back to zero, but this plot is for a step response of 1 m so the step the actual current will be $4A/10=0.4A$ which is reasonable. The factor of 10 was found by assuming that the maximum sensor distance would be about 10 cm which is the maximum distance that the robot would detect. This sensor distance will be significantly is the worst case scenario and the expected maximum sensor reading distance will be 4 cm. Figure 30 shows the step response of the lateral controls with the difference in angle between the two motors at the output.

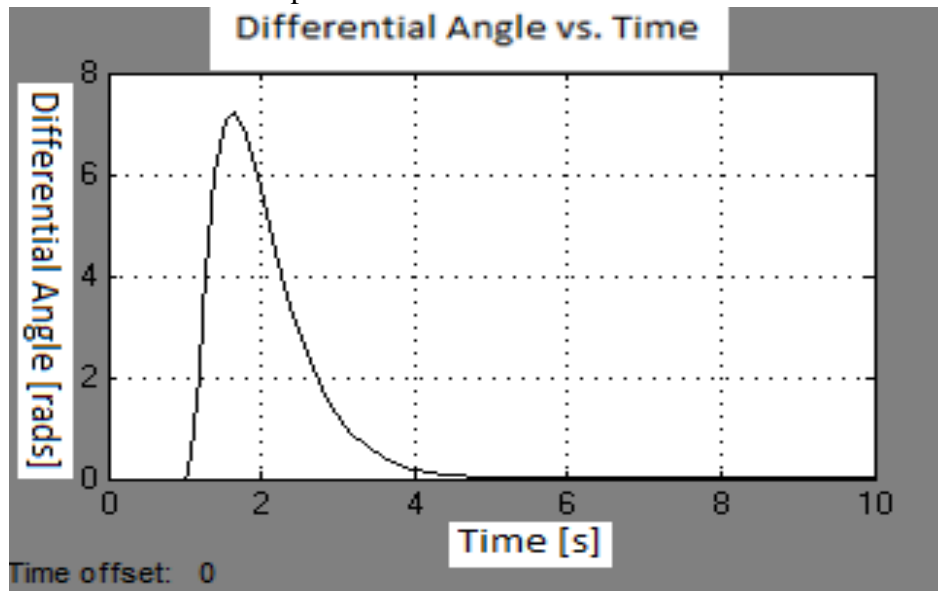


Figure D17: Step Response of Lateral Controls with Differential Angle at the Output

Notice that the differential angle between the two wheels peaks at 7 and then goes back down to zero when the robot is parallel to the axis. When the wheels are in phase with each other, the robot is going straight.

In order for the robot to stop at its target, it will have to receive a “slow down” command which will be issued when the robot is nearing its destination so that the robot may slow down in time. In order to give the robot this command, there will be a two by 4 inch piece of white line that is perpendicular to the robot path and when the robot crosses over it so that multiple sensors go off, and these sensors give a signal which tells the robot to slow down. The robot will receive a step response which is the negative of its maximum velocity, and the perpendicular white line will be placed so the robot will be able to be at about zero speed when it is close to its target point.

Motor Driver (NP)

The schematic in Figure E1 is of the original current control motor driver circuitry. This motor driver was not used due to changes made in the controls and high cost of the circuit components. Instead the team had chosen to use the dual VNH2SP30 Motor Driver Carrier MD03A shown in Figure E2. This board drives both motors using PWM with resolution 0-800 corresponding to 0-12V.

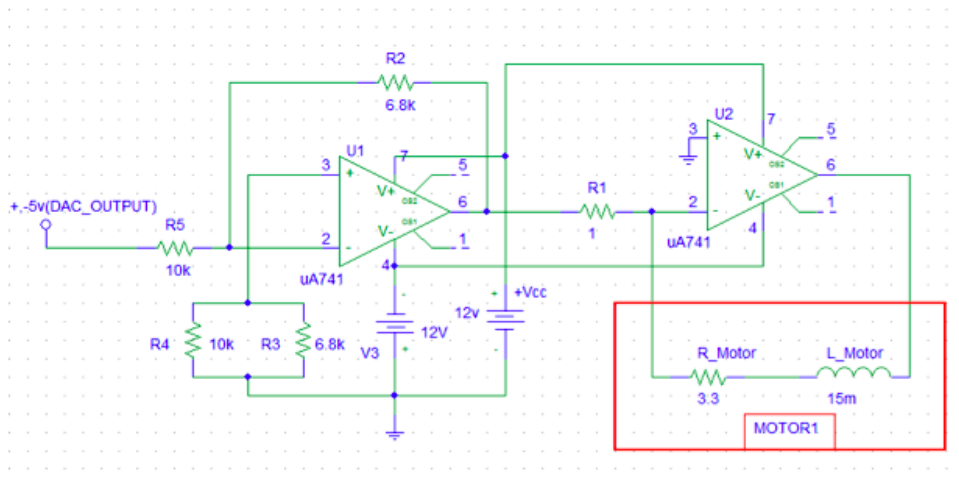


Figure E1: Schematic of Motor Driver Circuit

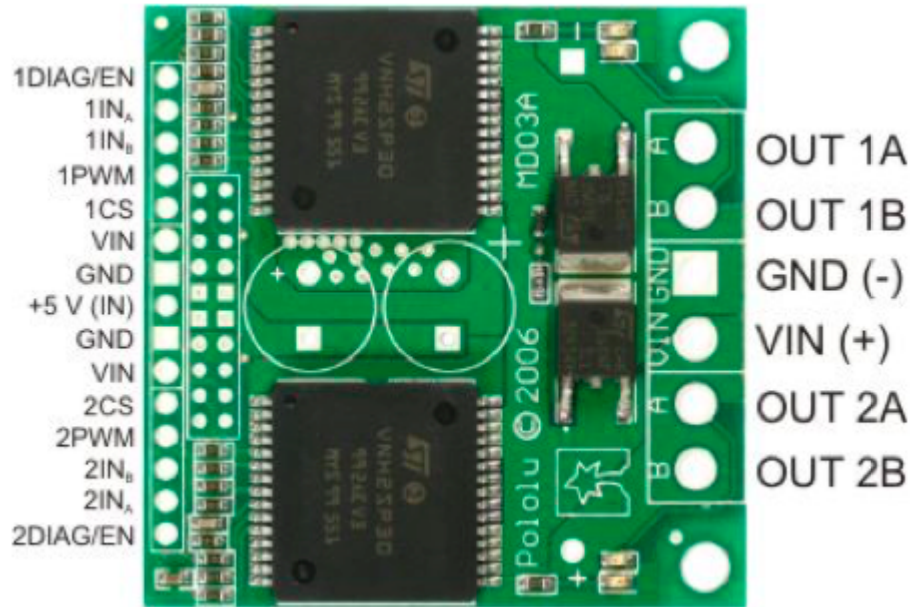


Figure E2: Motor Driver

Hardware Calculations (NP)

To determine the required performance parameters for a DC motor many calculations must be performed.

First, the required acceleration is calculated using an initial velocity of 0 m/s, a designed velocity of 0.33 m/s, and a desired time of 1 second to reach the designed velocity is

$$Acceleration \left[\frac{m}{s^2} \right] = \frac{Designed \ Velocity \left[\frac{m}{s} \right] - Initial \ Velocity \left[\frac{m}{s} \right]}{Time[sec]} \quad (H1)$$

$$Acceleration \left[\frac{m}{s^2} \right] = \frac{0.33 \left[\frac{m}{s} \right] - 0 \left[\frac{m}{s} \right]}{1[sec]} = 0.33 \left[\frac{m}{s^2} \right] \quad (H2)$$

Assuming the weight of the robot is 5.0 Kg and the wheel radius is 0.136m. The required torque to reach the desired velocity is

$$Torque = Mass[kg] * Acceleration \left[\frac{m}{s^2} \right] * Wheel \ Radius \ [m] \quad (H3)$$

$$Torque = 5 [kg] * 0.33 \left[\frac{m}{s^2} \right] * 0.136 [m] \quad (H4)$$

$$Torque = 0.2244 [Nm] \quad (H5)$$

The maximum required torque is 0.377 [Nm] seen from the simulation in Figure 21

Based on the parameters shown above the wheel revolutions per minute necessary to attain the design velocity is

$$RPM = \frac{Designed\ Velocity}{Wheel\ Radius} * \frac{60}{2\pi} \quad (H6)$$

$$RPM = \frac{0.33 \left[\frac{m}{s}\right]}{0.136[m]} * \frac{60}{2\pi} \quad (H7)$$

$$RPM = 23.171 \quad (H8)$$

The angular velocity is calculated as

$$Angular\ Velocity = RPM * \frac{2\pi}{60} \quad (H9)$$

$$Angular\ Velocity = 23.171 * \frac{2\pi}{60} = 2.426 \left[\frac{rad}{sec}\right] \quad (H10)$$

Resulting in the power consumption of two motor being

$$Power = Torque * Angular\ Velocity \quad (H11)$$

$$Power = 0.58 * 2.426 = 1.41 [Watts] \quad (H12)$$

Battery (DL)

The battery used was changed from a 12V (14.5V peak), 5Ah NiMH battery pack with a standard discharging rate of 5 Amps. Instead, a 12V (14.5V peak), 2.5 Ah NiCad battery pack with a much higher discharging rate of 50 amps was used. The original battery would not have supplied enough current, and the new one supplied more than enough. For the PIC (and associated sensors), a standard 9V battery was used, since this is within the 9-15V range that is specified, and a capacity of about 400mAh should be more than sufficient.

Mechanical Design (NP)

The robot butler was designed to be approximately 70 cm in height to match the average height of a restaurant table and the final product was 69.7 cm. The final chassis design was changed from a stack of three 40.64 cm diameter round acrylic plates to 17.78 cm by 30.48 cm acrylic plates with components placed in between the spacing of each plate. The reason for the change in dimensions was due to lighten the robots weight and increase the tilt angle the robot will have before having its lowest tier touch the ground. The acrylic plates are held together using 4 stainless steel threaded rods between each tier and secured together with nuts. This design enables quick installation of components with the ability to increase or decrease the height if desired. The ability to changes the height of the tiers proved to be a valuable asset while troubleshooting to balance the bot. The designed height was reached with the use of 13.6 cm wheels are used and the spacing between tiers 1-2 are about 20.57 cm apart and tiers 2-3 are about 35.56, resulting in a total height of approximately 69.7 cm.

The robot butler should not exceed a maximum weight of 7 Kg and the finals weight of the bot was approximately 4 Kg. With the majority of mechanical and electrical components chosen, table 4 shows the total estimated weight of the robot butler. The miscellaneous components were overestimated and include: adapters, circuitry components/chips, fasteners, wheels, wires.

Component [Quantity]	Estimated Weight [KG]
Acrylic Circle [3]	1.56
Battery [1]	2
IMU [1]	0.0028
Line Sensor [1]	0.00309
Motor [2]	0.207
Support Beams [8]	0.073
Miscellaneous	1.0
Total Estimated Weight	4.77

Table 5: Estimated Weight of Robot Butler

The mechanical design consist of two dc gear motor with encoder, one microcontroller, IMU (inertial mass unit) sensor, a battery, and motor driver circuitry. Figure E4 shows the original placements of components and Figure E5 shows the actual final location of the components. On the bottom of the first tier, the dc motors are mounted using custom l-brackets as shown in Figure E6. The IMU, logic level shifters, motor driver battery and motor drive circuitry were mounted on the top of the first tier. Next, the microcontroller was secured on the top of the second tier leaving the third tier open for the load (food or beverage). Everything was mounted using metal standoffs and screws except the microcontroller and 12v battery, which were fastened using Velcro.

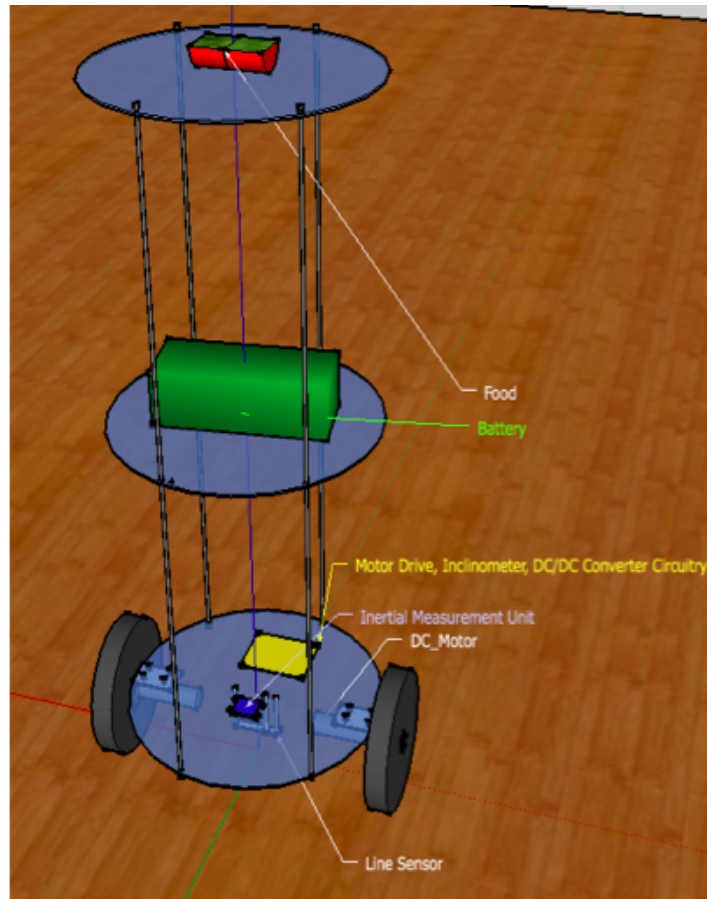


Figure E4: Physical Model of Butler Bot

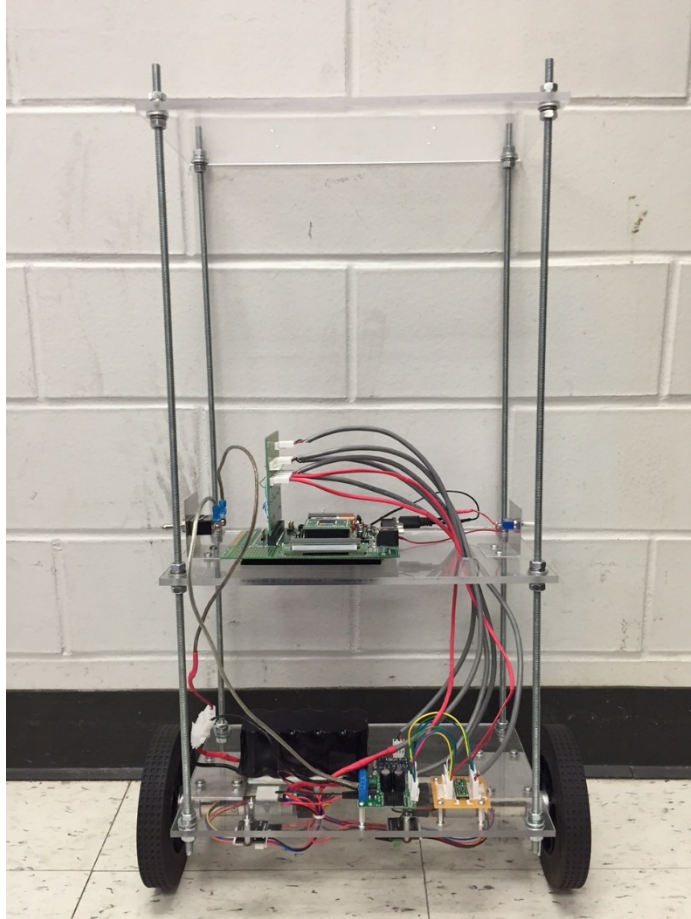


Figure E5: Final Physical Model of Butler Bot

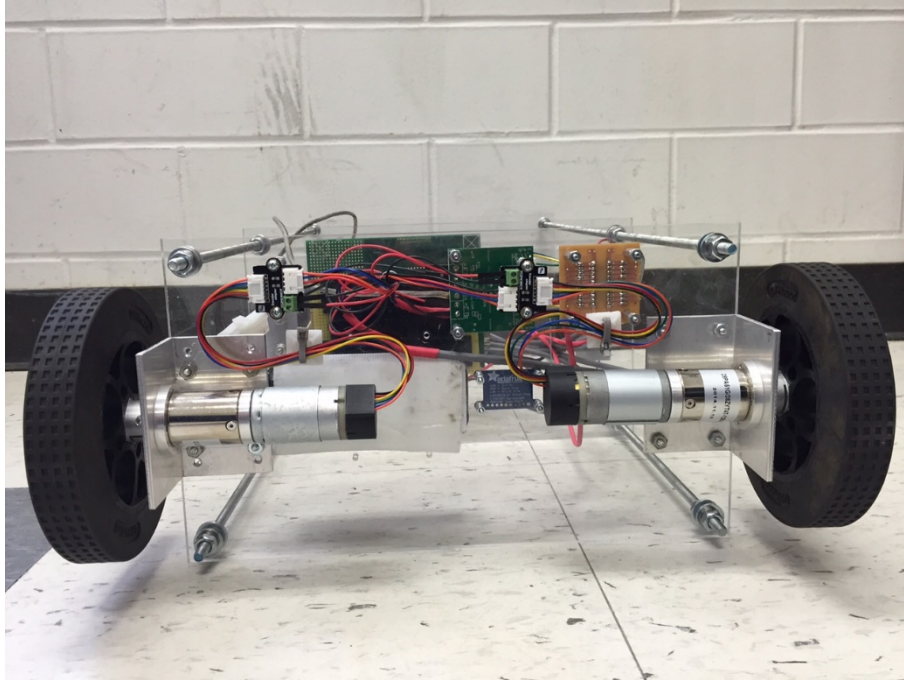


Figure E6: Bottom View of the First Tier

Operation, Maintenance and Repair Instructions (TG)

AlfreD, the two-wheeled-balancing robot, is a simple machine to operate. There are two switches, one on each of the sides. One controls power to the motors, and the other controls power to the explorer 16 board. First place the robot on the ground upright and make sure the wheels are touching the ground. The motors need to be powered first so the error does not accumulate in the controller. Flip the switch applying power to the motors, then flip the switch on the opposite side applying power to the microcontroller. The robot should immediately start correcting itself and start balancing. As AlfreD is balancing objects can be placed on the top acrylic plate, but for optimum balancing care must be taken to keep the mass as close to the center line as possible. If AlfreD bot ever begins to lose balance and fall over, either from a disturbance too strong to overcome or natural causes, turn off both switches. Because the motors receive drive instruction from the explorer 16, when it loses power the motors will not drive so it does not matter which goes off first, both will stop the motors from spinning. Reorient the robot and you are ready to again apply power and begin balancing.

Testing Procedures

Motor Testing: (DL)

To ensure that the calculated model of the robot's system was as accurate as possible, several motor tests were done. A number of parameters were measured at a given drive voltage, with different loads as well. Thus, using a known wheel radius, gravitational constant, torque constant, load mass and moment, as well as measured current and applied voltage, critical calculations could be made (although the plant model was not used in the finalized controls system, this was still a useful learning exercise). Firstly, the torque constant was easily confirmed using this method, using the equation $K_t = T/I$, and these values were found to closely

correlate with what was expected. Much more importantly, the viscous damping coefficient could be calculated (this was a significant factor in the plant model transfer function in the state-space and phase lag control loop implementations). This was done with a simple rearrangement of the damping equation, giving $K_d = -(T - K_t \cdot I) / \omega$, with the following results:

Motor Test				
	Test 1	Test 2	Test 3	Test 4
Wheel radius (m)	0.0619	0.0619	0.0619	0.0619
Voltage applied	4	6	7	10
Current(generated)	2.57	2.91	3.52	4.04
Speed	5.09	8	5.57	8.51
Gravity	9.81	9.81	9.81	9.81
Torque constant	0.3148	0.3148	0.3148	0.3148
Motor Constant	0.3148	0.3148	0.3148	0.3148
Mass (Kg)	0.144	0.144	0.55	0.55
Torque	0.087442	0.087442	0.333981	0.333981
Damping	0.141767	0.103578	0.138979	0.110201

Table 6: Motor test results

Moment of Inertia Measurement and Testing: (DL)

Two different methodologies were employed for finding the moment of inertia of the robot, which was initially a critical parameter in the controls plant model until PID control was implemented. First, the masses and distances of all the components of the pendulum were measured, and assumed to be point moments of inertia using the following MatLab code:

```
%DT10
%Self-Balancing Two-Wheeled Robot
%Moment of Inertia Calculations
%April 2, 2015
%David Laubli
%conditions: all masses are in kilograms, all distances are in meters

%masses:
m_tier=0.3703;           %mass of one acrylic plate
m_fasteners=0.00594;    %mass of fasteners for one acrylic plate
m_pic=0.1701;           %mass of pic board with all associated circuitry
m_9v=0.0468;           %mass of 9v battery
m_cables=0.125;         %mass of all cables
m_rods=0.29;            %mass of four rods
m_battery=0.599;        %mass of battery
m_batterycase=0.0907;   %mass of battery case
m_sensors=0.0907;      %mass of all sensors
m_switches = 0.1;       %mass of switches

%radii:
r_tier1=0.0301;         %radius of first tier
r_tier2=0.0301+0.2;     %radius of second tier
r_tier3=0.0301+0.2+0.3405; %radius of third tier
r_fasteners1=r_tier1;
r_fasteners2=r_tier2;
r_fasteners3=r_tier3;
r_pic=0.0301+0.2+0.025;
r_9v=0.0301+0.2+0.016;
r_cables=0.15;
```

```
r_rods=0.0301+0.279+0.015;
r_battery=0.0301+0.025;
r_batterycase=r_battery;
r_sensors=0.03;
r_switches=0.0301+0.2+0.015;

%moments of inertia
i_tier1=m_tier*r_tier1^2;
i_tier2=m_tier*r_tier2^2;
i_tier3=m_tier*r_tier3^2;
i_fasteners1=m_fasteners*r_fasteners1^2;
i_fasteners2=m_fasteners*r_fasteners2^2;
i_fasteners3=m_fasteners*r_fasteners3^2;
i_pic=m_pic*r_pic^2;
i_9v=m_9v*r_9v^2;
i_cables=m_cables^r_cables^2;
i_rods=m_rods^r_rods^2;
i_battery=m_battery*r_battery^2;
i_batterycase=m_batterycase*r_batterycase^2;
i_sensors=m_sensors*r_sensors^2;
i_switches=m_switches*r_switches^2;

%summation of moments of inertia
i_alfred=...
    i_tier1+...
    i_tier2+...
    i_tier3+...
    i_fasteners1+...
    i_fasteners2+...
```

```

i_tier3=m_tier*r_tier3^2;
i_fasteners1=m_fasteners*r_fasteners1^2;
i_fasteners2=m_fasteners*r_fasteners2^2;
i_fasteners3=m_fasteners*r_fasteners3^2;
i_pic=m_pic*r_pic^2;
i_9v=m_9v*r_9v^2;
i_cables=m_cables^r_cables^2;
i_rods=m_rods^r_rods^2;
i_battery=m_battery*r_battery^2;
i_batterycase=m_batterycase*r_batterycase^2;
i_sensors=m_sensors*r_sensors^2;
i_switches=m_switches*r_switches^2;

%summation of moments of inertia
i_alfred=...
    i_tier1+...
    i_tier2+...
    i_tier3+...
    i_fasteners1+...
    i_fasteners2+...
    i_fasteners3+...
    i_pic+...
    i_9v+...
    i_cables+...
    i_rods+...
    i_battery+...
    i_batterycase+...
    i_sensors+...
    i_switches;|

```

The resultant of this calculation was 0.01669 kg*m². Following this, the center of gravity of the pendulum was measured and found to be 0.1936 cm from the axis of rotation. The weight of the pendulum was measured at 4.453 kg, and calculating the moment of inertia from this using $I = m*r^2$ gave a moment of inertia of 0.00136 kg*m². It was assumed that the more global calculation was the more accurate one, and the simulations with that assumption were much more reasonable.

Controls testing: (DL)

Once the robot was physically constructed, the phase-lag controller with a pole located at 2.5 and a zero located at 2 was implemented in C code using an iterative algorithm. The results of this were as expected from the step response--it appeared that the bot could not keep up with the extremely fast response that was required by this controller, and the response was wildly and frantically oscillatory and of course unstable.

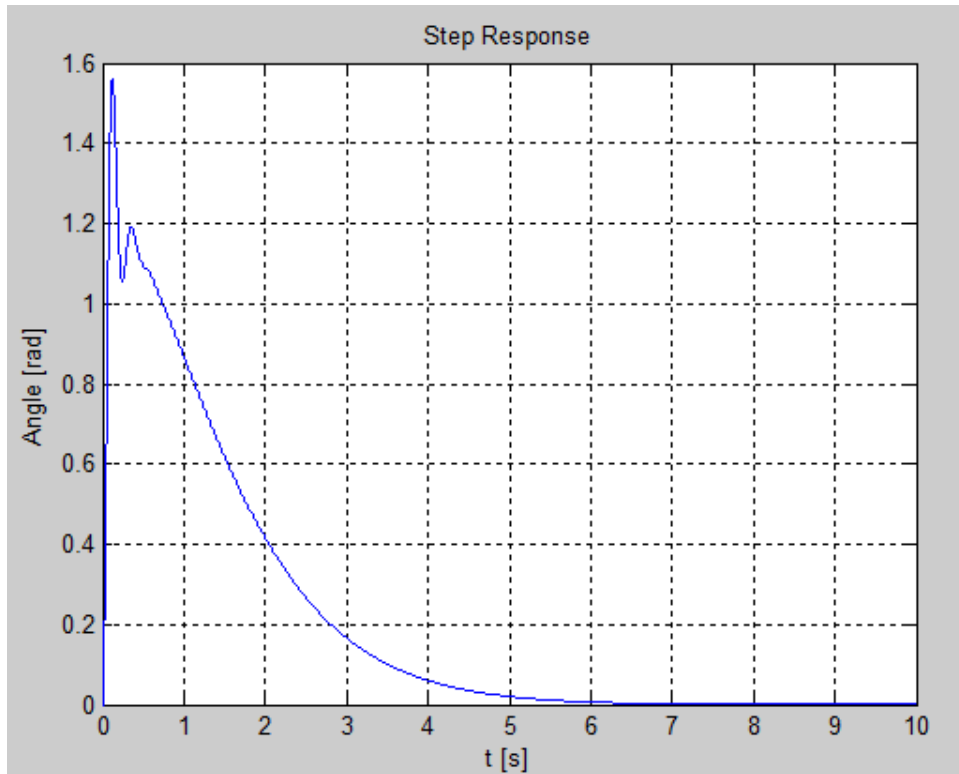


Figure F1: Step Response of Original Phase-Lag Controller

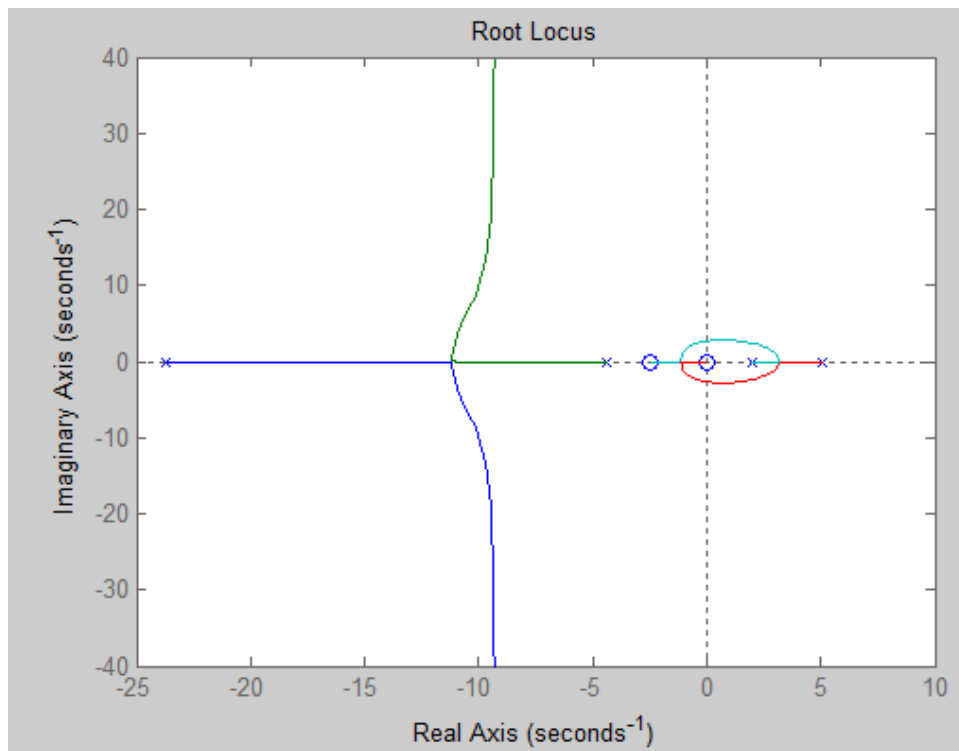


Figure F2: Root Locus Plot of Original Phase-Lag Controller

Following this, the controller was revised to give a more reasonable and realistic step response simulation. As can be seen, this type of behavior should be well within the limits of the actuators of the robot, but when programmed and tested, this controller exhibited similar wild and jerky oscillations to the original, flawed controller. All the signals coming from sensors and from the complementary filter, as well as the PWM output, were observed to be as expected and free of noise. Adjusting the threshold, either in terms of degrees or of voltage, at which the actuators move, did not improve the response. Evidently, something in the plant was most likely incorrectly modeled, and it remains undetermined what exactly this is. It could be the viscous damping coefficient, as the value obtained for this parameter was considerably greater than in most DC motors, which is certainly not a good sign. Also, the moment of inertia of the pendulum was calculated using individual component moments of inertia to be $0.01669 \text{ kg}\cdot\text{m}^2$, which is approximately a tenth of the value that was found using the center of gravity and mass of the pendulum itself ($0.00136 \text{ kg}\cdot\text{m}^2$). These calculations were double- and triple-checked, and are also likely a source of error.

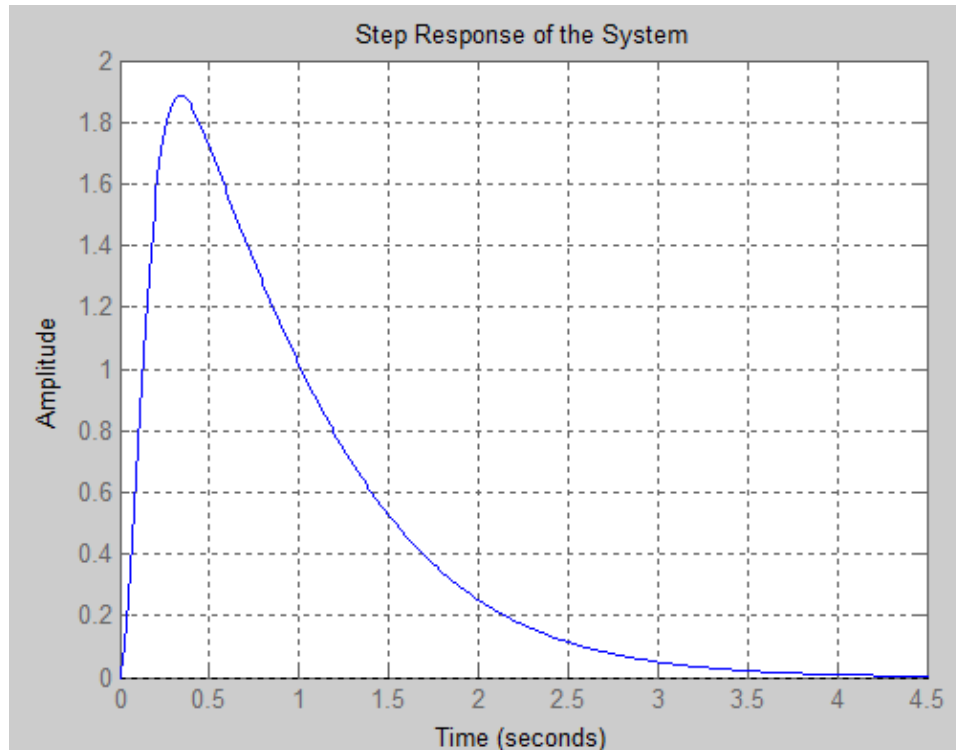


Figure F3: Step Response of Modified Phase-Lag Controller

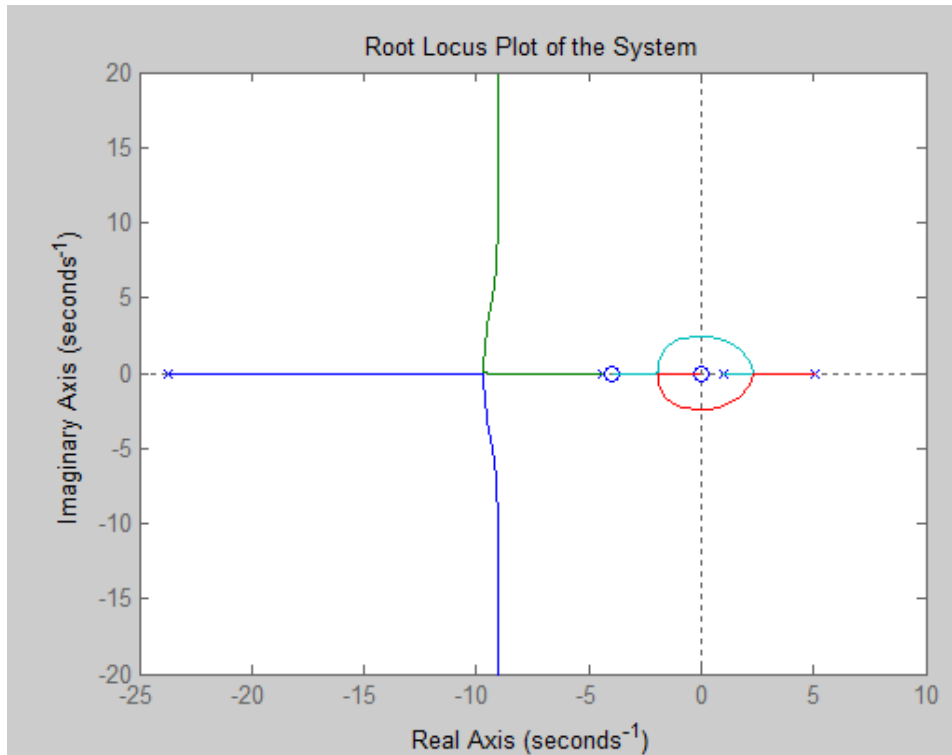


Figure F4: Root Locus Plot of Modified Phase-Lag Controller

It was then decided that the simplest route to take would be to implement a PID controller, using the Nichols-Ziegler tuning method, meaning that the proportional gain K_p was found first. This was done by finding a point at which the initial gain produced approximately steady oscillations, and this was found to be around a gain of 300, which is divided by two for a gain K_p of 150. The period of oscillation (discounting the smaller, less steady oscillations) was about $T_u = 4$ seconds, and so K_i was found to be $2 * K_p / T_u = 75$, and K_d to be $K_p * T_u / 8 = 75$ as well. These gain values were tested on the robot, and the robot balanced somewhat well, but with high-frequency oscillations. It drifted and lost its balance quite quickly. Given the general rule of thumb that K_d improves stability if its value is *small*, K_d was approximately halved to 35, resulting in better balancing with fewer high-frequency oscillations. However, the robot still drifted and fell soon. The gain K_d was again reduced, this time to 15, which resulted in further improvement, in the form of fewer oscillations, and a longer balancing time. After reducing K_d to 5, oscillations were virtually eliminated, but the robot still drifted and eventually fell. Finally, K_d was set to 2, and stability was further improved. This seemed to be better than leaving K_d at 0, as the response then seemed to be slightly slower, even though the difference was close to negligible.

Test	P	I	D	Results
1	100	0	0	No Oscillations
2	125	0	0	No Oscillations
3	150	0	0	No Oscillations
4	200	0	0	No Oscillations
5	500	0	0	High Sustained Oscillations
6	300	0	0	Lower Sustained Oscillations
7	150	75	75	Balancing Some HF Oscillations Drifts and falls
8	150	75	35	Balancing fewer HF Oscillations Drifts and falls
9	150	75	15	Balancing fewer HF Oscillations Drifts and falls
10	150	75	5	Balancing fewer HF Oscillations Drifts and falls
11	150	75	2	Balancing fewer HF Oscillations Drifts and corrects

Table 7: Balance test results

Project Schedule: (DL)

The details of the original schedule are shown in the Gantt chart below. Many of these were revised throughout the course of the project, as was expected to happen. First of all, the motor driver circuitry was decided against, so the DAC and Op-Amp were researched and worked on for a number of weeks, but not all the soldering and interfacing was completed. The parts for line following were not ordered, as the decision was made to focus solely on balancing the robot first. Physical construction was begun over a month later than expected, which was mostly due to difficulties encountered in finding equipment to cut and drill the acrylic plates with, since the material is easily damaged and great care must be taken in machining it. The physical layout of parts was delayed because of this too, and also because a battery for the project was not decided on until the final weeks of the project. It was difficult to find a battery that offered high energy capacity and allowed high current pull, while remaining within a reasonable weight range. Eventually, a battery pack was found on a site that customizes components for battle robots, which turned out to be an ideal solution. Controls code was not finished until the last week of the project, instead of being complete in February, because the controls scheme switched to PID control later on, and was tuned up until the day before demonstrations. Also, sensor integration was delayed somewhat due to difficulties with coding interrupts and obtaining all the information necessary to communicate with and receive usable data from the sensors. Once the DAC motor circuitry was dropped from the project, motor driver boards and logic level shifters were immediately looked into and purchased, and integration of these components was smooth and rather uneventful. The team originally decided to drop the DFRobot motors, as detailed in

the motors section, in favor of some by Pololu, but these were dropped and the DFRobot motors resorted to since they had considerably more torque, and it was discovered that the Pololu motors would be all but impossible to mount to the acrylic plate due to the shape and position of the mounting bracket. It was also decided that the inclinometer was not nearly as useful as the IMU's accelerometer, due to its slower response, and so other options for determining pendulum angle were looked into, and a complementary filter thoroughly researched and settled on, since it was a simple and robust solution. It was implemented in the last two weeks of the project, at the same time as the controls. State-space controls continued to be worked on and tuned throughout the first several weeks of the semester, but it was decided that classical controls would be far simpler and more efficient in terms of processing power to implement in code. Thus, a basic phase-lag controller was developed, which (as described in the controls testing section) did not function as expected. This was when PID control was attempted, and found to work quite well, during the last week of the project.

Category	Subcategory	Function	9-Feb	16-Feb	23-Feb	2-Mar	9-Mar	16-Mar	23-Mar	30-Mar
Motor	Circuitry			Parts In		Finish Motor Layout				
		DAC			Finish Circuitry theory begin solder	Finish Soldering				
	Op-Amp									
	Interfacing					Interface and Controls finished				
Encoders			Interfacing Begin							
Sensors	Inclinometer		Filter Theory							
		Filter								
	Integration									
	Accelerometer									
	Filter									
	Integration									
	Gyro									
	Filter									
	Integration									
Line following		Parts Ordered Pending								
Construction					Parts In/ begin constructin		Construction Finished			
	Model									
		Physical Construction								
	Parts Layout	Finish Layout								
Code	Controls Code									
		Theory				Transfer Function Implented				
	Sensor Integ	Begin integration	Filters							

Table 8: Original Gantt Chart

Category	Subcategory	19-Jan	26-Jan	2-Feb	9-Feb	16-Feb	23-Feb	2-Mar	9-Mar	16-Mar	23-Mar	30-Mar	6-Apr	13-Apr	17-Apr	
Motor	DAC	Coded, initialized and tested					Dropped									
	Op-Amp	Circuitry constructed and tested					Dropped									
	DFRobot Motors	Encoders tested and coded		Motors tested		Dropped						Replaced Pololu motors				
	Pololu 30:1							Motors and encoders tested		Dropped						
	Pololu 50:1						Motors and encoders tested		Dropped							
Electronic Components	Inclinometer	Initialization and testing														
	Accelerometer			Initialization and testing												
	Gyro			Initialization and testing												
	PIC	Individual and integrated component initialized, coded and tested														
	Logic Level Shifters						Initialized and tested with multiple motors									
	Motor Driver Board						Initialized and tested with multiple motors									
	Complimentary filter			Researched						Code: implemented and tested						
Construction	Tier									Holes drilled						
	Motor Bracket									Design and fabrication (DFRobot)						
	Component Mounting											Final Positions determined and mounted				
	Placement Design						Component placement drawn up, agreed upon and finalized					Switches, battery added				
Controls	State space controller designed and simulated						Controls algorithm redesigned (phase lag)			Moment of inertia		PID controller designed and tuned				

Table 9: Revised/Actual Gantt Chart

Parts List (JP)(DL)(NP)

Battery

Part Number: BPK-CP2500-12

Vendor: Robot MarketPlace

Website Ordering Link: <http://www.robotmarketplace.com/products/BPK-CP2500-12.html>

Description: Team Nightmare 12V 2.5ah CP NiCad Battle Pack

Price: \$62.99

Quantity: 1

Battery Charger

Part Number: TSD ERUSC01

Vendor: Amazon

Website Ordering Link: <http://www.amazon.com/TSD-Universal-Charger-7-2-12V-batteries/dp/B001DHC2LO>

Description: The TSD ERUSC01 can charge large and small plug NiMH and NiCd airsoft batteries.

Price: \$ 22.17

Quantity: 1

Cast Acrylic Tier Layers

Part Number: N/A

Vendor: TAP Plastics

Website Ordering Link:

[http://www.tapplastics.com/product/plastics/cut to size plastic/acrylic sheets cast clear/510](http://www.tapplastics.com/product/plastics/cut%20to%20size%20plastic/acrylic%20sheets%20cast%20clear/510)

Description: Cast Acrylic Clear

Price: \$16.00

Quantity: 3

Inclinometer

Part Number: 551-1053-1-ND

Vendor: Digi-Key

Website Ordering Link: <http://www.digikey.com/product-detail/en/SCA830-D07-1/551-1053-1-ND/1888929>

Description: Inclinometer Y-Axis +/-1G SPI

Price: \$44.37

Quantity: 1

Inertial Measurement Unit

Part Number: 1604

Vendor: Adafruit

Website Ordering Link: <http://www.adafruit.com/product/1604>

Description: 10-DOF IMU Breakout

Price: \$29.95

Quantity: 1

Logic Level Shifters

Part Number: 2595

Vendor: Pololu

Website Ordering Link: <https://www.pololu.com/product/2595>

Description: Logic Level Shifter, 4-Channel, Bidirectional

Price: \$2.50

Quantity: 2

Microcontroller

Part Number: DM240001

Vendor: Microchip Direct

Website Ordering Link:

<http://www.microchipdirect.com/ProductSearch.aspx?Keywords=DM240001>

Description: Explorer 16 Development Board (100-pin)

Price: \$129.99

Quantity: 1

Motors

Part Number: FIT0277

Vendor: Robot Shop

Website Ordering Link: <http://www.robotshop.com/en/12v-silent-dc-motor-146rpm-encoder.html>

Description: 12V Silent DC Motor 146 with Encoder

Price: \$46.27

Quantity: 2

Motor Drive Board

Part Number: 708

Vendor: Pololu

Website Ordering Link: <https://www.pololu.com/product/708>

Description: Dual VNH2SP30 Motor Driver Carrier MD03A

Price: \$59.95

Quantity: 1

Wheels (Pair)

Part Number: FIT0252

Vendor: DFRobot

Website Ordering Link:

http://www.dfrobot.com/index.php?route=product/product&path=66_46_101&product_id=882

Description:

Price: \$26.05

Quantity: 1

Screw (L-Bracket to Tier1)

Model Number: 605393

Vendor: Lowes

Web Ordering link: http://www.lowes.com/pd_62054-37672-605393_1z0vdg3+1z0vrdj_?productId=4746581&Ns=p_product_qty_sales_dollar|1&pl=1¤tURL=%3FNs%3Dp_product_qty_sales_dollar%7C1%26page%3D1&facetInfo=1

Description: 10 count #4- 40 x 1-in

Price: \$1.24

Quantity: 1

Screw (L-Bracket to Motor)

Model Number: 28626

Vendor: Fastenal

Web Ordering link: <https://www.fastenal.com/web/products/details/28626>

Description: #4-40 x 3/16-in Round-Head Zinc

Price: \$0.0259

Quantity: 8

Support Beams

Model Number: TROD-01

Vendor: Lynxmotion

Web Ordering link: <http://www.lynxmotion.com/p-338-threaded-rod-12-x-2-56-6.aspx>

Description: 12" x 2-56

Price: \$7.88

Quantity: 8

Support Beams Bolts

Part Number: 0170884

Vendor: Fastenal

Web Ordering link: <https://www.fastenal.com/web/products/details/0170884>

Description: 4-40 Stainless Steel Small Pattern Hex Nut

Price: \$0.0665

Quantity: 16

Design Team Information (JP)

Thomas Garabedian, Electrical Engineering

David Laubli, Electrical Engineering

Jordan Paul, Computer Engineering

Nikheel Patel, Electrical Engineering

Michael Redle, Electrical Engineering

Conclusions and Recommendations (TG)

The balancing Butler Bot will help the serving industry. Adding an automated element to the restaurant business causes labor costs to be reduced and customers to have a more consistent and rewarding service experience. In the early stages, the bot will be able to deliver between the table and the delivery room. From there, capability should expand to traveling between different paths and tables. The current controls system is being reviewed for improvements. The final control system is a state based variable system that incorporates each of the sensors into a feedback system to establish a better understanding of the current balance state of the Butler Bot. Because the Butler Bot is constantly rebalancing, each sensor will be interfaced with the micro controller for a low cost and consistent operation. Using the obtained information, the controller will output a signal to the DC motors, instructing how much torque to apply to each wheel. Each wheel will have its own DC motor because turning and balancing will require independent actions. The array on the bottom of the Bot will tell the robot if it is on the path, and will give instructions for adjustment when it strays. The system will be a benefit to the future of robotics and automation.

References (DL)

- [1] Ammar Wasif, Danish Raza, Waqas Rasheed, Zubair Farooq, and Syed Qaseem Ali. (2013). **Design and implementation of a two wheel self balancing robot with a two level adaptive control.** *IEEE*, 187-193.
- [2] Blackwell, T., Casner, D., Wiley, S., & Nelson, B. (2008). In Anybots 2.0 I. (Ed.), **Remotely controlled self-balancing robot including a stabilized laser pointer** (700/54 ; 700/245; 700/62; 700/65 ed.). California, USA: G05B 13/02 (20060101); G05B 19/18 (20060101).
- [3] Groothuis, S. S. (2008). Self-balancing robot 'dirk'. *University of Twente*,
- [4] Ji, L., Xizhe, Z., & Yanlong, L. (2011). **Balance control of two-wheeled self-balancing mobile robot based on TS fuzzy model.** *Strategic Technology (IFOST), 2011 6th International Forum on*, 1, 406-409.
- [5] Jianwei Zhao Xiaogang Ruan. (19-23 Dec. 2009). **The flexible two-wheeled self-balancing robot intelligence controlling based on boltzmann** *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, , 2090-2095.
- [6] Nor Maniha Abdul Ghani, Faradila Naim, Tan Piow Yon. (2011). **Two wheels balancing robot with line following capability.** *World Academy of Science, Engineering and Technology*,
- [7] Takenaka, Toru (Saitama, J., Akimoto, Kazushi (Saitama, J., Kobashi, Shinichiro (Saitama, J., Murakami, & Hideo (Saitama, J. (November 5, 2013). In Honda Motor Co., Ltd. (Tokyo, JP) (Ed.), **Control device of inverted pendulum type vehicle** *US 8478490 B2* (701/1 ; 701/69; 73/514.36 ed.). Japan: G01P 15/00 (20060101).
- [8] Zhao, J. (2008). **The control and design of dual - wheel upright self-balance robot.** *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, , 4172-4177.
- [9] **Accelerometer & Gyro Tutorial** (2010). Retrieved October 14, 2014 from <http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/?ALLSTEPS>
- [10] **How To Select A DC Motor** (2008). Retrieved October 14, 2014 from <http://www.faulhaber-group.com/n378442/n.html>
- [11] **Kamal, Ibrahim (2008). Infra-Red Proximity Sensor** Retrieved October 14, 2014 from <http://ikalogic.cluster006.ovh.net/infra-red-proximity-sensor-part-1/>

Appendices (JP)

RJ11 Breakout Adapter Mechanical Drawings

<http://site.gravitech.us/MicroResearch/Breakout/RJ11-TERM/rj11-term.PDF>

DC-to-DC Converter

<http://www.cui.com/product/resource/v78xx-2000.pdf>

DAC Datasheet

http://www.analog.com/static/imported-files/data_sheets/AD5662.pdf

IMU Datasheets:

<http://www.adafruit.com/datasheets/L3GD20H.pdf>

<http://www.adafruit.com/datasheets/LSM303DLHC.PDF>

<http://www.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf>

IR Line Sensor Array User Guide:

<http://www.pololu.com/docs/0J12>

Motor Spec Sheet:

<http://www.robotshop.com/media/files/images/12v-silent-dc-motor-146rpm-encoder-4-large.jpg>

PIC24EP256GP206 Datasheet:

<http://ww1.microchip.com/downloads/en/DeviceDoc/70000657H.pdf>

Wheel Mechanical Drawing:

<http://www.robotshop.com/media/files/pdf/mechanical-drawing-fit0252.pdf>