|  |
| --- |
|  |
| Two Wheel Inverted Pendulum Balancing Robot |
| EE500 – Digital Controls |
|  |
| **Benjamin O'Brien** |
| **12/9/2012** |

|  |
| --- |
|  |

Contents

[Introduction 2](#_Toc342888941)

[Original Bill of Materials 2](#_Toc342888942)

[Revisions to Bill of Materials 7](#_Toc342888943)

[Control Analysis and Design 10](#_Toc342888944)

[Initial Control Scheme: Bang Bang Control 10](#_Toc342888945)

[Proportional Control 10](#_Toc342888946)

[Performing System Identification 10](#_Toc342888947)

[Increasing Sensing Ability 10](#_Toc342888948)

[Balance Bot Performance 11](#_Toc342888949)

[Lessons Learned 11](#_Toc342888950)

[Appendix A – Code 14](#_Toc342888951)

[Appendix B – Equations of Motion 15](#_Toc342888952)

[Appendix C – System Identification 19](#_Toc342888953)

# Introduction

This document details the design and implementation of a two wheeled inverted pendulum balancing robot. The goal of this project is to produce a robot capable of balancing on two wheels simliiar to the popular Segway personal mobility device. This paper details the original design and bill of materials for the robot as well as the update bill of materials after the robot was constructed. As well as the bill of materials, the methods used to tune the robot and the final performance is discussed along with the lessons learned by performing this project.

# Original Bill of Materials

The next sections detail the original parts selection for the balancing robot and the modifications made to the bill of materials during the construction and testing of the robot. The goal was to build a robot for under $200 dollars. The initial BOM meet this specification but due to changes and additions the budget was exceeded by $14.

|  |  |  |  |
| --- | --- | --- | --- |
| https://dlnmh9ip6v2uc.cloudfront.net/images/products/2/4/2/00242-1.jpg | **Infrared Proximity Sensor** | | |
| **Description:** Infrared proximity sensor made by Sharp. It has an analog output that varies from 3.1V at 10cm to 0.4V at 80cm. The sensor has a Japanese Solderless Terminal (JST) Connector. A pigtail with a JST connection is available so that one does not need to be manufactured and the board does not have to be soldered to. | | |
| **Price** | **Unit Price** | **Quantity** | **Total** |
| $11.95 | 2 | $23.90 |
| **Order Link** | <http://www.pololu.com/catalog/product/136> | | |
|  | | | |
| **Justification:** These sensors were selected for the available range and cost. They are to be mounted on top of the robot so that they face the ground. By placing them on opposite sides of the balance bot the two reported distances can be compared and the tilt angle of the balance bot can be calculated. When the system is balanced both sensors should read the same distance. Arranging the sensors in this way will also make it easier to detect noisy measurements because the two readings are complementary.  The sensors are non-linear, but due to the height of the robot and the desired angle of operation, the distances measured will not vary greatly and thus a linear approximation can be formed. The top tier of the robot is adjustable so it can be adjusted to find the optimal distance range.  https://dlnmh9ip6v2uc.cloudfront.net/images/products/2/4/2/IRSensor-3.jpg | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| https://dlnmh9ip6v2uc.cloudfront.net/images/products/8/7/3/3/08733-03-L.jpg | **Infrared Sensor Jumper Wire - 3-Pin JST** | | |
| **Description:** Three pin JST connector with red, black, and yellow colors. 30 cm wire outs. This cable comes fully assembled as shown and connects directly to many different Sharp sensors. | | |
| **Price** | **Unit Price** | **Quantity** | **Total** |
| $0.99 | 2 | $1.98 |
| **Order Link** | <http://www.pololu.com/catalog/product/117> | | |
|  | | | |
| **Justification:** These wires are for connecting the Sharp infrared sensors to the Ardunio and power source. Instead of soldering wires directly to the sensors, these jumper wires fit into the JST connectors already mounted on the sensors. These wires will facilitate easier and quicker construction and ensure a solid connection with the sensors. This should minimize noise that could be introduced from a faulty connection. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| http://b.pololu-files.com/picture/0J903.600.jpg?2f6bdb33ecdebb9d8f4d69952fa75527 | **TB6612FNG dual motor driver carrier.** | | |
| **Description:** This tiny board is an easy way to use Toshiba’s TB6612FNG dual motor driver, which can independently control two bidirectional DC motors or one bipolar stepper motor. A recommended motor voltage of 4.5 – 13.5 V and peak current output of 3 A per channel (1 A continuous) make this a great motor driver for low-power motors. | | |
| **Price** | **Unit Price** | **Quantity** | **Total** |
| $8.45 | 1 | $8.45 |
| **Order Link** | <http://www.pololu.com/catalog/product/713> | | |
|  | | | |
| Justification: This motor controller supports up to 3A draw on each channel which is in excess of the 1.6 Amp Peak draw from each motor, this provides a current buffer which will protect the controller. In addition this controller can be used with PWM inputs which will allow for simple control of the motors with the Arduino. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| http://b.pololu-files.com/picture/0J2814.600.jpg?8f42a31206af8665d564cf7ace918d83 | **50:1 Micro Metal motor HP w/ Extended Motor Shaft** | | |
| **Description:** These tiny brushed DC gearmotors are intended for use at 6 V, though in general, these kinds of motors can run at voltages above and below this nominal voltage, so they should comfortably operate in the 3 – 9 V range (rotation can start at voltages as low as 0.5 V).  Key specs at 6 V: 625 RPM and 100 mA free-run, 15 oz-in (1.1 kg-cm) and 1.6 A stall. | | |
| **Price** | **Unit Price** | **Quantity** | **Total** |
| $16.95 | 2 | $33.90 |
| **Order Link** | http://www.pololu.com/catalog/product/2213 | | |
|  | | | |
| **Justification:** The choice of the motor depends on the rpm and torque needed to run the robot and this motor supports more than what the robot needs. It has more than enough speed to recover the unbalanced robot and more than enough torque to actuate the robot. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| http://a.pololu-files.com/picture/0J1199.600.jpg?b6963fabbc341b265e8373ffe7edd746 | [**Pololu 42x19mm Wheel and Encoder Set**](http://www.pololu.com/catalog/product/1218) | | |
| **Description:** This set includes a pair of [42×19mm wheels](http://www.pololu.com/catalog/product/1090), a pair of [extended brackets](http://www.pololu.com/catalog/product/1089), and two matching [encoders](http://www.pololu.com/catalog/product/1217). Just pick a pair of [micro metal gearmotors](http://www.pololu.com/catalog/category/60) to complete your feedback-enabled drive system. | | |
| **Price** | **Unit Price** | **Quantity** | **Total** |
| $39.95 | 1 | $39.95 |
| **Order Link** | http://www.pololu.com/catalog/product/1218 | | |
|  | | | |
| **Justification:** This wheel encoder set was designed to be compatible with the selected motors and chassis. This will aid in construction of the robot and should allow for reliable reading of the wheel encoder data. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| http://b.pololu-files.com/picture/0J1437.600.jpg?e477ea4fb2965125ce6cb46260c361b9 | [**Pololu 5" Robot Chassis RRC04A Solid Black**](http://www.pololu.com/catalog/product/1501) | | |
| **Description:** This chassis base plate is a round piece of 1/8" (3 mm) acrylic with a 5" (127 mm) diameter and cutouts for building a differential-drive robot chassis based on our [micro metal gearmotors](http://www.pololu.com/catalog/category/60) and [42×19mm wheels](http://www.pololu.com/catalog/product/1090). The chassis has assorted general-purpose holes and slots that support many configurations of sensors and other robot components. | | |
| **Price** | **Unit Price** | **Quantity** | **Total** |
| $7.95 | 2 | $15.90 |
| **Order Link** | http://www.pololu.com/catalog/product/1501 | | |
|  | | | |
| **Justification:** This chassis is an adaptable prefabricated design which is compatible with the motors and encoders selected. In addition to its small size and low cost, its rigid design will support the necessary components for this balance bot. Two chassis are used to allow for a tier design enabling precise location of the Infrared sensors. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Lithium Polymer Battery - 7.4V 1000mAh | **Lithium Polymer Battery - 7.4V 1000mAh** | | |
| **Description:** This is a 7.4V 2 cell LiPo battery pack with 1000mAh of charge. It is rated at a maximum discharge of 25C. Dimensions are 70mm x 35mm x 18mm and weight is 3oz. | | |
| **Price** | **Unit Price** | **Quantity** | **Total** |
| $6.95 | 1 | $6.95 |
| **Order Link** | http://www.robotshop.com/productinfo.aspx?pc=RB-Spa-645&lang=en-US | | |
|  | | | |
| **Justification:** It supplies 7.4V which is within the range of the motor and controller. In addition, one already exists in the cardboard box of Ukulele project spare parts. | | | |

# Revisions to Bill of Materials

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Parallax 27911-RT 3-Axis Gyroscope Module** | | |
| **Description:** The Gyroscope Module is a low power 3-Axis angular rate sensor. The gyroscope shows the rate of change in rotation on its X,Y and Z axes. Temperature output data and raw measured angular rate is accessed from the selectable digital interface: I²C. The module is a small package design and has an easy to access SIP interface with a mounting hole for quick connectivity to the projects. | | |
| **Price** | **Unit Price** | **Quantity** | **Total** |
| $34.99 | 1 | $34.99 |
| **Order Link** | http://www.radioshack.com/product/index.jsp?productId=12310182 | | |
|  | | | |
| **Justification:** This gyroscope was selected due to its availability. It was purchased at a brick and mortar RadioShack and was the only available option. It has various update rates and sensitivity settings that allow for adaptation to the project requirements. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Arduino A000077 Proto Shield | **Arduino A000077 Proto Shield** | | |
| **Description:** This proto shield offers a variety connection points for soldering and attaching various components to pins on the Arduino and elsewhere. This allows for more solid connections than jumper wires and provides a more professional look. | | |
| **Price** | **Unit Price** | **Quantity** | **Total** |
| $12.97 | 1 | $12.97 |
| **Order Link** | http://www.radioshack.com/product/index.jsp?productId=12369524 | | |
|  | | | |
| **Justification:** The proto shield was selected as the best method for mounting the motor controller and gyroscope in a clean, organized fashion. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Arduino Motor Shield** | | |
| **Description:** The Arduino Motor Shield is based on the L298 which is a dual full-bridge driver designed to drive inductive loads such as relays, solenoids, DC and stepping motors. It lets you drive two DC motors with your Arduino board, controlling the speed and direction of each one independently. | | |
| **Price** | **Unit Price** | **Quantity** | **Total** |
| $34.99 | 1 | $34.99 |
| **Order Link** | Purchased from RadioShack B&M | | |
|  | | | |
| **Justification:** This shield was selected for its availability and ease of use. After a malfunction with the originally selected motor controller, a new one needed to be acquired. This shield was well within the required power specifications and allowed for easy integration with the Arduino based platform. | | | |

It should be noted that a different battery was used than the one present in the Bill of Materials. The selected battery was acquired from a personal collection and was no longer labeled so its exact details cannot be included. It differed from the listed battery in capacity (800mAh) and size.

The Completed BOM can be seen below. The items shown in Red are items listed in the original BOM that were either not used or destroyed in the process of creating the balance robot. The total slightly exceeds the $200 mark allowed for the construction of the robot. Ways to have met the budgetary goals will be discussed further in the Lessons Learned section of the report. It should be noted that construction materials were not included in the BOM. The cost of the related hardware and hookup wire was minimal and did not exceed more than $5. This hardware included an existing sheet of fluorescent lighting egg create, ¼” threaded rod, and some 4-40 nuts and bolts.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product Description** | **Unit $** | **Qnty** | **Total** | **Order Link** |
| [Pololu 5" Robot Chassis RRC04A Solid Black](http://www.pololu.com/catalog/product/1501) | $7.95 | 2 | $15.90 | <http://www.pololu.com/catalog/product/1501> |
| [Pololu 42x19mm Wheel and Encoder Set](http://www.pololu.com/catalog/product/1218) | $39.95 | 1 | $39.95 | <http://www.pololu.com/catalog/product/1218> |
| [50:1 Micro Metal Motor HP with Extended Motor Shaft](http://www.pololu.com/catalog/product/2213) | $16.95 | 2 | $33.90 | <http://www.pololu.com/catalog/product/2213> |
| [TB6612FNG Dual Motor Driver Carrier](http://www.pololu.com/catalog/product/713) | $8.45 | 1 | $8.45 | <http://www.pololu.com/catalog/product/713> |
| [Sharp GP2Y0A21YK0F Analog Distance Sensor 10-80cm](http://www.pololu.com/catalog/product/136) | $11.95 | 2 | $23.90 | <http://www.pololu.com/catalog/product/136> |
| [3-Pin Female JST PH-Style Cable for Sharp Sensors](http://www.pololu.com/catalog/product/117) | $0.99 | 2 | $1.98 | <http://www.pololu.com/catalog/product/117> |
|  |  |  |  |  |
| Lithium Polymer Battery – 7.4V 1000mAh | $6.95 | 1 | $6.95 | <http://www.robotshop.com/productinfo.aspx?pc=RB-Spa-645&lang=en-US> |
| **Total** |  |  | **$131.03** |  |
|  |  |  |  |  |
| Parallax 27911-RT 3-Axis Gyroscope Module | $34.99 | 1 | $34.99 | http://www.radioshack.com/product/index.jsp?productId=12310182 |
| Arduino A000077 Proto Shield | $12.97 | 1 | $12.97 | <http://www.radioshack.com/product/index.jsp?productId=12369524> |
| Arduino Motor Shield | $34.99 | 1 | $34.99 | Found in local RadioShack |
| **Revised Total** |  |  | **$213.98** |  |

# Control Analysis and Design

Despite derivation of the dynamic equations for the system, it was difficult to determine some of the parameters necessary to utilize the equations effectively. The motor constants were not available in the data sheet and various physical aspects, such as the moment of the inertia of the robot’s chassis, were very difficult to determine. As a result various control techniques were used and experimented with to acquire the best performance.

## Initial Control Scheme: Bang Bang Control

The initial control scheme relied on the two infrared sensors being used to determine how far the robot was from a set balanced position. Whichever side was lower, the robot would drive towards with a constant motor output. The motor output was not proportional to the tilt angle. Results showed promise, but did not result in a stable system.

It should be noted that at this time, the robot was just the chassis, wheels, motor, and sensors. The Arduino and motor controller were off board and a long tether was used to supply voltage to the motors. At this time the effects of the tether were not taken into consideration. Building resumed with the hope that a proportional control could provide the desired performance.

## Proportional Control

The next attempt was made with all the hardware onboard. Knowing from previous attempts that bang bang control was not sufficient, a simple proportional control was used. The Arduino remained tethered to a PC for data and performance logging. It was found that when tethered the proportional control seemed sufficient. However, once removing the robot from the tether the dynamics of the system changed completely. The proportional gain was tweaked in various ways but acceptable performance was never achieved with the tether.

## Performing System Identification

Using the logged data, system identification was attempted, at first using data collected by a tether and then by data collected using a wireless XBee system. The data collected with the tether was found to be unreliable when applied to an untethered robot, as expected, but at that time, no XBees were available. Once moving to the wireless data logging, the models and tuning generated became more accurate. Using these models different types of controllers were designed and tested with varying degrees of success.

## Increasing Sensing Ability

Initial results from the system identification showed promise and with minor adjustments it was possible to obtain much better performance than was previously achieved. However, the use of only IR sensors was proving to be inadequate for accurately detecting the state of the robot. As a result, a gyroscope was added that allowed for a less noisy, more refined way to detect angular displacement and angular rate. The best controller equations from the system identification were adapted to the new data coming from the gyro, which allowed for a well-tuned PID controller to be implemented very quickly. The IR sensors were then used only to correct the gyroscope drift. The resulting performance is discussed further in the Balance Bot Performance section.

# Balance Bot Performance

The final performance of the balance bot had excellent stability while the system was translating but had significant steady state error that prevented the robot from maintaining a stationary balance point. Due to the weight distribution of the robot, it was inherently unstable to such a degree that it was very difficult to maintain balance with the selected drive train. This was due to slack in the gearbox that prevented the smaller magnitude motor outputs from exerting the required force quickly enough to maintain balance. If larger forces were used, the gear lash was so significant that stability could not be achieved.

Once the robot had lost balance in one direction or another and the slack had been taken out of the gearbox, the robot would maintain its tilt angle for quite some time depending on the surface and the available room. During this time it would slowly accelerate but was rarely capable of accelerating fast enough to correct the balance of the robot.

The robot required a perfectly set initial position to work well. The procedure required for a good start required holding the robot perfectly vertical while the IR sensors and gyro were calibrated and zeroed out. The robot could then be released and allowed to fall down or could be held up until the motors began to react. If the initial position was not perfect the robot would prefer to drift to one side or another as it tried to target the now slightly off vertical target angle. Even with a perfect start, the robot would tend to favor one side due to the fact that the weight distribution of the Arduino and its shields was not perfectly uniform from side to side.

The robot’s response to external disturbances was quite robust. Many times during testing the robot would roll over cables and tools with little to no effect on the behavior. The robot could take larger disturbances such as hard flicks and moderate smacks on the boom without falling to the ground. It was even possible to quickly spin the robot about the z-axis and have it maintain and balance. The robot was also able to recover or pick itself up from just about any angle within 30 degrees of vertical though it would only sometimes find balance near the vertical when recovering from such large angles.

The balance bot was found to preform best on carpet and other high pile surfaces like AstroTurf. The thick material provided more resistance and helps alleviate some of the instability issues. On tables and slick floors performance was less stable and the robot would speed off and lose control easier. Other thoughts on improved balancing performance are discuss in the Lessons Learned section.

# Lessons Learned

There are many lessons to take from this project. Balancing performance could be improved in many ways involving the selection or motors, wheels, and sensors. As was discovered when building and testing the robot, the IR sensors proved to be too noisy and unresponsive at small angles of displacement to maintain stability and balance. Purchasing a full 6-DoF IMU from the beginning would have saved time and would not have cost any more than the two IR Sensors and the standalone gyroscope. The attempt to create a budget balance bot using cheap IR sensors resulted in budget performance despite various filtering techniques used on the IR Sensor output. The addition of an accelerometer would have also help resolve the issue of getting a perfect start. Instead of guessing where the best balance point is, the accelerometer could be used to determine where the zero angle is.

The stability and resistance to disturbances that were achieved with the selected motor and wheel combination showed promise that if more appropriate parts were selected balance could have been achieved. Firstly, avoiding gearboxes if at all possible would make balancing much easier. The purchased MicroMetal motors were thought to be of decent quality but the included gear box had a few degrees of play before it would engage fully. As discussed in the Performance section, the unresponsiveness of the gear train made it impossible to balance the robot at vertical. Spending more on ungeared or high quality geared motors would alleviate the issues associate with gear lash.

Continuing with the drivetrain, buying larger, rounder wheels would have provided a great benefit over the low quality, treaded, 42mm wheels that were used in this build. The treading on the selected wheels was not uniform and was spaced out too far. As a result, the wheels did not result in smooth application of force and added an additional disturbance to deal with when balancing the robot. Smoother, rounder wheels with little to no treading would not exhibit this issue. Furthermore, purchasing larger wheels would allow for more robust acceleration in comparison with the small wheels that were used. If the wheels were large enough it could possibly make up for the lash in the gearboxes.

The next lesson to take away from this project extends beyond the construction of a balance bot. While testing the balance bot a small short in the power circuitry resulted in another, undetected short. Once fixing the initial short the robot was powered back on, only to have the motor controller ignite. Much time was lost and the budget was adversely affected attempting to clean up the power circuitry and setting up a new motor controller. The new motor controller imparted a valuable lesson as well. It used a more expensive H-Bridge IC than the previous controller and as a result motor response was actually improved. Despite the extra cost, it is wise to buy existing shields for motor control instead of constructing one as it will save both time and eliminate possible short circuit conditions. Still, it is also important to look for problems everywhere in a circuit after a short has occurred in what appears to be an isolated area.

Another lesson involves time management when building and testing the robot. At each stage of construction the robot was played with in an attempt to see how well it would balance. Many of these test had the robot on some sort of tether which provided drag that made the robot perform much better than it would when untethered. As a result, much time was wasted tuning the tethered robot. It is recommended to complete the construction of the robot before attempting to improve balancing characteristics.

Even once construction was complete, it was found that data would need to be logged in order to perform a system identification. This was initially done with a USB tether and while the resulting equations produced in MATLAB were provided great performance while tethered the results were unsatisfactory when untethered. To get more accurate data, wireless XBee radios were used. It is recommended that if constructing a mobile balancing robot, the inclusion of a wireless communication module be used for communication. It allows for much more accurate data logging and removes the effects one could experience when using a tether.

While the exact equations generated by the system identification and tuning were not used it was found that the equations were a great help. The equations produced a more acceptable tune than random guess and check tuning and saved time as only minor tuning was necessary to work out the kinks. While the exact equations were not used, the time necessary to perform the analysis and get a good starting point was much less than the time it would have taken to manually compute the required gains. Additionally the use of SISO tool allowed for easy visualization of the tuned system’s response. In this way, solutions with slow rise times or oscillating overshoot could be thrown out without even having to test them on the robot.

# Appendix A – Code

The control loop for the balancing robot is show below. The full source is available in a github repository at: <https://github.com/benjamionob6/ArduBalance>

**void** loop(){

*//Our sensors don't update any faster than 10ms*

**if** (micros() **>** lastMicros **+** 10000){

*//Read from the IR Sensors*

  Sensor.takeReading();

*//Save off the previous gyro readings*

**double** lastGyro **=** gyroXrate;

**unsigned** **long** dt **=** micros() **-** lastGyroTime;

*//calculate the angle change degrees*

**double** angleChange **=** (lastGyro) **\*** (dt **/** 1000000.0);

*//Get new values*

  getGyroValues();

*//Take the difference in the last two gyro readings to get the angular acceleration...*

**double** gyroAccel **=** (lastGyro **-** gyroXrate);

*// Update the current angle based on calculations*

  currentAngle **+=** angleChange;

*//Check for significantly small error from the IRSensors and reset the angle*

*//to 0 no more than twice a second*

  fErr **=** Sensor.getBalanceError(0);

  bErr **=** Sensor.getBalanceError(1);

**if** ( abs(fErr) **<=** 2 **&&** abs(bErr) **<=** 2 **&&** (micros() **-** lastBiasFix) **>** 100000){

    currentAngle **=** 0;

    lastBiasFix **=** micros();

  }

**unsigned** **long** currentMicros **=** micros();

  lastMicros **=** currentMicros;

*//Allow 2 seconds for setting down the robot before we start balancing*

**if** ( currentMicros **<** 4000000 )

**return**;

  MotorOutput **=** 18.5 **\*** currentAngle **+** 1.24 **\*** gyroXrate **+** 0.6885 **\*** gyroAccel;

  Motor.SetMotors(MotorOutput,MotorOutput);

 }

}

# Appendix B – Equations of Motion

The balance bot equations were developed and then cross-checked against the following project report found online which assisted in linearization and equation verification of two-wheeled inverted pendulum:ftp://190.90.112.131/Facultades/FIET/DEIC/Materias/Control%20Inteligente/Parte%20III/clase%2023%20int/RBF/2003-Balance-Ooi.pdf – 



# Appendix C – System Identification

To perform the system identification on the balancing robot using one of the MATLAB toolboxes, collection of the current robot’s performance data must first be implemented.  For our robot, we logged in the, infrared sensors’ data, and the pulse width data output to the motors based on a simple proportional control as well as the time stamp at which each motor command was issued. A simple proportional control was used that would cause the robot to oscillate but maintain relative balance for a few seconds, which was long enough to develop a decent system model.

With that, the MATLAB System Identification Toolbox imported these data inputs and based on just these inputs, the transfer function defining the relationship between system inputs and outputs of the balancing robot was generated using least-squared approximation.  These transfer functions range from the third to the tenth order and usually the higher the order, the higher the accuracy percentage of the polynomial functions. However, when implementing the PID control for this robot, we usually only take at most the second or third order transfer functions, as tuning higher order functions is much more difficult, and the percentage accuracy are usually still fairly high.

Once a transfer function is chosen, further options to generate better gains for the PID control of the system can be done using SISO tool. SISO tool provides a root locus plot and bode plots along with graphs predicting the rise time, overshoot, settling time, steady state error, and stability the modeled response with the applied tune.

After we were satisfied with the simulated response, the toolbox produced both a tuned discrete time and continuous time transfer function. The discrete time transfer function was used, since the controller is implemented in the digital domain, and it is presented in the z-transform representation. We can then use this equation for robot’s PID controller and see whether or not the robot performs as predicted by the model.  Even if we did not use the whole equation into our robot’s system, the gains given are good estimation of what the PID gains need to be and only require some tweaking due to imperfections in the model.