

Southern Polytechnic State University

Four Degree of Freedom Control System Using a Ball on a Plate

PREPARED BY
Jonathan Bruce
Chris Keeling
Ronald Rodriguez

2 May 2011

Submitted in Partial Fulfillment of the
Requirements for the Bachelors of Science
Degree in Mechatronics Engineering

ABSTRACT

For the proposed project, the well-known ball and plate control problem is explored. The basic idea is that a system using sensors, actuators, and a control law will keep a free rolling ball in a desired position on a flat plate accounting for external disturbances which may occur. This report clarifies functional requirements, evaluates alternative concepts, presents the design requirements, shows minimum performance success criteria, details the physical design, outlines the control systems, overviews the prototyping and testing phase, as well as mentions some post prototype optimizations which may be considered for future development. Over all, the design and prototype was successful and met all minimum performance success criteria.

Table of Contents

ABSTRACT	i
Table of Contents	ii
1. Introduction	1
1.1. Overview.....	1
1.2. Background.....	1
1.3. Objectives	2
2. System Requirements	3
2.1. Technical Requirements and Specifications	3
2.2. Activity Analysis	4
2.3. Minimum Performance Success Criteria	4
3. System Configuration	5
4. Component Selection and Major Subsystems	8
4.1. Actuation Selection	8
4.2. Material Selection	9
4.3. Shaft Material and Design.....	10
4.4. Drive Selection and Design.....	11
4.5. Ball Position Detection	12
4.6. Software.....	13
4.7. Microcontroller	13
4.8. Graphical User Interface	14
4.9. Controller Analysis and Design Verification	14
5. Implementation.....	19
5.1. Tilt Mechanism.....	19
5.2. Actuation.....	22
5.3. Power and Wiring.....	24
5.4. Vision Processing	26

5.5. <i>Graphical User Interface</i>	28
5.6. <i>State-Space Controller</i>	29
6. Test Data	31
7. Optimization	32
8. Conclusion	34
Reference	35
Appendix A: Technical Drawings	36
Appendix B: Initial Wiring Schematic	48
Appendix C: Bill of Materials	49
Appendix D: Modified S-FUNCTION	50
Appendix E: Modified Guide Generated Code	57
Appendix F: Ball and Plate Gain Controller Code	59
Appendix G: Final Simulink Model	59
Appendix H: Savox SC1258TG Specifications	61
Appendix I: Roboteq AX500 Specifications	62

1. Introduction

1.1. Overview

The ball and beam experiment is one of the most popular and widespread control systems projects in undergraduate studies. It is so widespread because of the fact that many control systems techniques and practices can be studied and applied. The purpose of the system is to balance a small ball on a pivoting beam at a certain point over any period of time and disturbance. This particular project will take the ball and beam experiment and extend it by essentially adding a second ball and beam system in parallel creating what is known as a ball and plate experiment.

1.2. Background

To understand the concept of the ball on plate system, it is first imperative to understand the simpler ball on beam system. The theoretical model of the ball on beam system is shown in Figure 1. This model of the ball and beam system is a two degree of freedom system. The first degree is the angle of the beam which is actuated by the electric motor. The second degree of freedom is the ball's position along the beam, which is non-actuated and is a function of the angle of the beam.

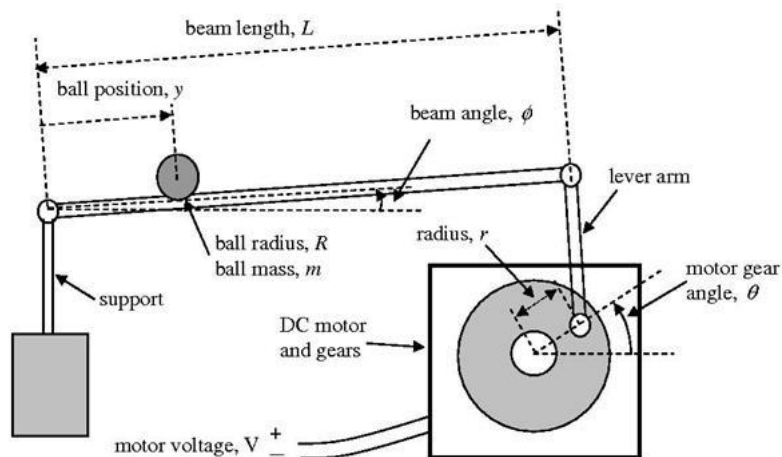


Figure 1: Ball and Beam Model

This system is referred to as under-actuated system because not all degrees of freedom are forced by the controller. Also, the system is known as a marginally stable system. That is in an open loop control system, a step input to the system's beam angle results in the ball's position continuously changing without bound until the ball falls off the end of the beam. Another negative inherent in the system, is the system's nonlinear nature. The acceleration enacted on the ball due to changes in beam angle has a non-linear relationship to the ball's position.

1.3. Objectives

One of the main problems with past implementations of the ball and plate system is the nonlinearity that is inherent in the mechanical pivot design of the system. Secondary to this problem, past implementations have used complex and expensive actuation, control, and programming equipment.

If the system could be designed so that the origin of the plate is fixed in relation to the origin of the two rotation frames, the system would be simplified and improved. In addition to that improvement, the implementation of the system using affordable and readily available equipment would open the use of the system up to many more groups.

This project will focus on creating a ball and plate control system that will come as close as possible to simulating the ideal theoretical model of the system. Also within the project scope is the goal to reduce cost and complexity of the system by streamlining the control of the system with the use of a graphical user interface.

2. System Requirements

2.1. Technical Requirements and Specifications

Upon completion of the project, the following design requirements were set as goals that should be met:

Functional Performance

- Maximum settling time of 15 seconds
- System will operate on 110-120 volt, 60 Hz power supply
- Camera will operate at 15 frames per second or better
- Balance an object between $\frac{3}{4}$ " and 2" in diameter
- Support objects of up to $\frac{1}{4}$ lb.
- Working plate size of 2'x2'

Operating Environment

- Educational lab temperatures of 50°-125° F and humidity of 10-90%
- Pressure range of 1 atm \pm 50%

Safety

- Will not burn or electrocute operator or bystanders
- Safety shut off if interference is detected

Economic

- Useful life of greater than 5 years
- Electricity as only operating expense

Maintenance and Repair

- No regular maintenance should be required
- Materials should be corrosion resistant
- Repairs should require no special tools or equipment

Ease of Use

- Operator need only undergraduate basic course work to operate
- Reading and understanding data sheet should take less than 15 minutes
- All software needed should be free or available on the Southern Polytechnic State University campus

Manufacturing

- Manufacture cost should be under \$500
- A single prototype will be made
- Prototype should be complete by April 28, 2011

2.2. Activity Analysis

The following is an activity analysis that shows how the product will be used and ultimately be retired:

Set Up

- 1) Read Data Sheet
- 2) Run Matlab Program
- 3) Adjust Camera
- 4) Place Ball

Use

- 5) Set desired position
- 6) Subject ball to force
- 7) Repeat steps 5 and 6
- 8) Replace ball when necessary

Retire

- 9) Remove servos and controller for future use
- 10) Scrap the frame

2.3. Minimum Performance Success Criteria

While the team considers all design requirements reachable, the following conditions should be met to consider the final product minimally sufficient:

- Build a working prototype
- Balance a ball in desired position on the plate though any reasonable disturbance.
- Communicate knowledge learned through formal report

3. System Configuration

With the functions and requirements of the system determined, a configuration will now be specified. The most essential element of the system is the method in which the plate is pivoted. After some quick research and brainstorming ideas, three suitable solutions were identified. These three solutions and their description is shown below:

1. **Roller Ball:** In this system a cage with driven multidirectional wheels are placed on a ball that is solidly mounted to ground. When the wheels are turned the cage and in turn the plate's angle is altered.
2. **Labyrinth:** This system is based around the age old labyrinth game's pivoting layout. One frame rotates inside of another to give the plate two degrees of rotation.
3. **U-Joint:** In this system the plate is joined to one end of a u-joint at its center while the other end of the u-joint is mounted to the ground. The plate is then actuated by levers placed on the actuators near the edge of the plate.

Table 1 shows an Alternatives Matrix that was used to determine the best design for the pivoting sub-system out of the three designs that were considered. The ranking is based on six areas of importance; cost and simplicity are regarded as the highest factors. The team assigned a number between 1 and 5 to each design in each category and a weighted sum was calculated for the three systems. The labyrinth game's pivoting design was determined as the best pivoting sub-system. A sample system is shown in Figure 2.

Table 1: Pivot Alternatives Matrix

	Importance	Roller Ball	Labyrinth	U-Joint
Cost	30%	3	3	4
Simplicity	30%	4	5	3
Manufacture ability	15%	3	3	4
Existing Information	5%	0	3	5
Physical Advantage	5%	5	5	3
Durability	15%	3	4	2
	Weighted Sum	3.25	3.85	3.40



Figure 2: Labyrinth Game

Figure 3 shows the general form of the system as developed and agreed upon by the design team. It closely parallels the labyrinth game with an inner plate rotating inside another outer frame. The rotating axis of the inner frame and outer plate are perpendicular to each other and in the same plane. The inner plate working area was set as two feet by two feet. Complete drawings of this system can be found in Appendix A.

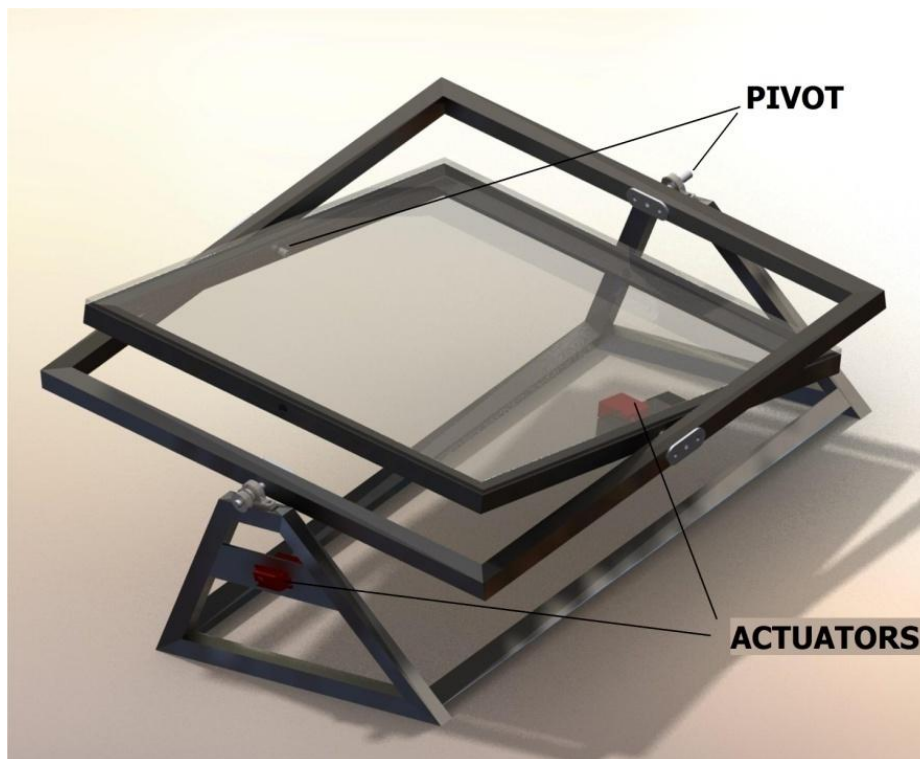


Figure 3: General Model

Once the plate configuration was developed, a basic system block diagram was made showing the relationships of each major sub-system. Using this data the system's sub-systems were then developed. The system block diagram is shown in Figure 4.

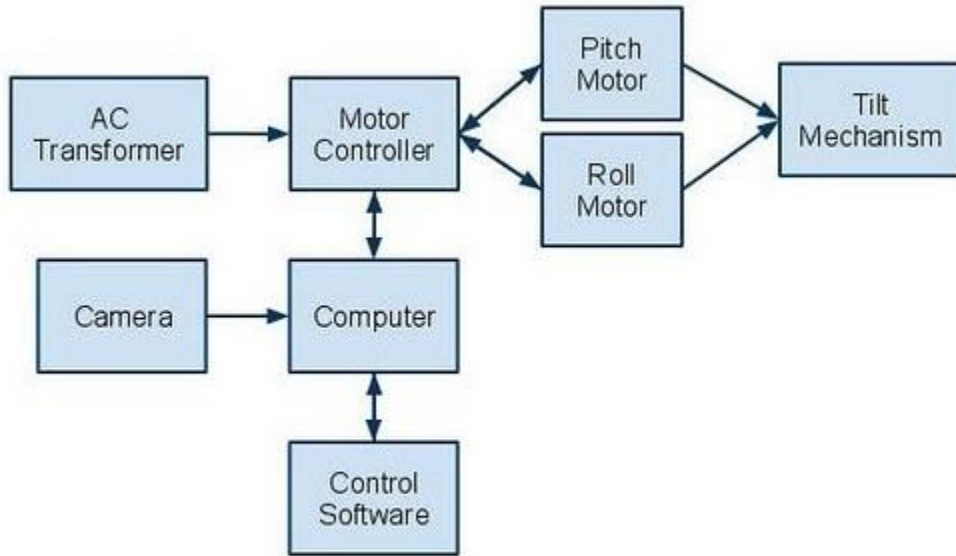


Figure 4: System Block Diagram

4. Component Selection and Major Subsystems

4.1. Actuation Selection

After the plate configuration was determined, an actuation mechanism was developed. It is important to determine the actuation mechanism, because other calculations will be based upon it. The torque of the desired actuator will also need to be a reference for how robust the structure should be designed. There are typically three actuator types that can accomplish the motion needed, a DC gear motor, a DC stepper motor, and a DC servo motor. With a budget of \$100 per actuator, a search was conducted for suitable models of each actuator type. The data in Table 2 shows the selected models along with their cost, torque, and speed. As can be seen, the torque is much higher for the servo motor and the speed is relatively higher on the gear motor and stepper motor.

Table 2: Actuation Types

	DC Gear motor	Stepper Motor	DC Servo
Manufacturer/Part #	Globe Motors/415A157-2	SDP-SI/S9117M-D15HT	Turnigy/HV300
Price	\$120.15	\$92.22	\$74.00
Torque(oz-in)	33.5	36.1	533
Speed(rev/s)	2.8	3.3	1.4

A simple time calculation for an object to fall the two foot length of the plate vertical is shown as follows:

$$t = \frac{\sqrt{2*d}}{g} = \frac{\sqrt{2 \cdot 2\text{ft}}}{32.2 \frac{\text{ft}}{\text{s}^2}} = 0.352 \text{ s}$$

Using this time value, the equation below is used to calculate the needed speed to rotate the plate from vertical to horizontal within that time period:

$$\omega(0.352\text{s}) = \frac{\phi}{t} = \frac{90^\circ}{0.352\text{s}} = 255.7 \frac{\text{degrees}}{\text{sec}} = 0.71 \frac{\text{rev}}{\text{s}}$$

Since every actuator falls within the needed speed criteria calculated, the deciding factor is torque and price. In both of these areas the servo motor excels so it is the obvious choice. The specifications for the Turnigy HV300 is shown below:

- Dead band: 0.0008 ms (Default)
- Control System: +Pulse Width Control
- Working frequency: 1520 μ s / 333hz
- (RX) Required Pulse: 3.5 ~7.4 Volt Peak to Peak Square Wave
- Operating Voltage: 8.4 ~12.0 V DC Volts
- Operating Temperature Range: -10 to + 60 Degree C
- Operating Speed (8.4V): 0.180 sec/60° degrees at no load
- Operating Speed (12V): 0.130 sec/60° degrees at no load
- Stall Torque (8.4V): 26.90 kg.cm (373.6 oz/in)
- Stall Torque (12V): 38.40 kg.cm (533.3 oz/in)
- 360° Modifiable: NO
- Motor Type: Coreless Motor
- Potentiometer Drive: Indirect Drive
- Driver Type: FET
- Bearing Type: Dual Ball Bearings
- Gear Type: Titanium Gears
- Programmable: NO
- Connector Wire Length: 15.0 cm (5.9 in)
- Dimensions: 40X20X45 mm (1.57x0.97x1.77 in)
- Weight: 78.5g (2.77 oz)

4.2. Material Selection

After communicating with a local metal supplier it was suggested that the frame be constructed of square tubing that is one inch square and 1/8 inch thick. Using the cross sectional area of this tubing and setting the size of the plate at two square feet, Table 3 shows the results of calculations to determine the density of the material needed. Using 6063T6 aluminum as a reference, it was calculated that the torque that would be produced from the plate is 480 oz-in which lies below 533 oz-in, the maximum of the chosen actuator. 6063T6 was chosen as a reference because the only other locally available light weight alloy at the time was 6061T5. While 6061T5 has better welding characteristics it was only available in full twenty-four feet lengths and is twice as expensive as 6063T6.

T6 temper 6063 has an ultimate tensile strength 30,000 psi (207 MPa) and yield strength of 25,000 psi (172 MPa). In thicknesses of 0.124-inch (3.1 mm) or less, it has elongation of 8% or more; in thicker sections, it has elongation of 10%.

Table 3: Material Selection Calculations

A 28"x28" frame will need to hold the weight of a 1/2lb force (ball) on its outer edge as well as the weight of a 2'x2' inner frame all made out of 1"x1"x0.125 tube (square). Using 6063T6 will allow the use of an available servo with a torque rating of 533 oz-in.

Problem Definition Parameters			
Parameter	Symbol	Units	Value
Cross Sectional Area	A	in ²	0.234
Inner Frame Volume	V _i	in ³	21.53
Outer Frame Volume	V _o	in ³	23.4

Design Variables				
Parameter	Symbol	Units	Value	
Material Density	l	lb/in ³	0.097	6063T6 Tube

Performance Characteristics						
Engineering Characteristics	Symbol	Units	Value	Type	Value	Condition
Torque	t	oz-in	533	>	480	Satisfied
Weight	W	lb	5	>	4.54	Satisfied

4.3. Shaft Material and Design

The structure will need a shaft to allow the actuator connection to the central axis of the plate, support the plate, and allow rotation. This shaft will be subjected to a torque equal to the maximum torque of the actuator, 533 oz-in. The most available shaft diameter size is 3/8", so the calculations below use this value as a reference. The resulting shear stress of the shaft, using the weight of the inner and outer frames from the material calculation section, equates to 400 psi. Using a commonly available shaft material, 303 stainless steel, the allowed shear stress is 11.2 million psi. This material is not only economical but also gives a safety factor of 28,000 in the shaft.

Table 4: Shaft Selection Calculations

The shafts will need to hold the weight of the inner and outer frame at 4.54lb as well as withstand the torque of the actuator at 533oz-in.

Problem Definition Parameters			
Parameter	Symbol	Units	Value
Weight of Frames	W_f	lb	4.54
Maximum Torque	t_m	oz-in	533

Design Variables			
Parameter	Symbol	Units	Value
Shaft Diameter	d	in	0.375

Performance Characteristics						
Engineering Characteristics	Symbol	Units	Value	Type	Value	Condition
Shear Stress	σ	ksi	11200	>	0.4	Satisfied

4.4. Drive Selection and Design

There were three options considered for connecting the actuator to the shaft pivots of the plate:

- directly with a coupler
- with gears
- with a chain or belt

Since the inner plate will be placed inside of the outer frame, the connector must be very compact, less than one inch tall. The chosen option for this connector is limited only to a direct couple to the actuator. Calculations for the strength of this connection are omitted as the connection has a very high safety factor.

Table 5: Chain Drive Calculations

A chain system is calculated to work with the maximum torque from the actuator, 533 oz-in, using standard 0.25" pitch length chain.

Problem Definition Parameters			
Parameter	Symbol	Units	Value
Maximum Torque	t_m	oz-in	533

Design Variables			
Parameter	Symbol	Units	Value
Pitch Diameter	PD	in	1.955

Performance Characteristics						
Engineering Characteristics	Symbol	Units	Value	Type	Value	Condition
Tensile Load	T	lb	787	>	34.1	Satisfied

4.5. Ball Position Detection

There are two types of position detection devices that were considered for this system. One type of sensor that was considered is a pressure sensitive resistive device, much like the touch screen on newer cell phones. The second option considered is a simple camera mounted above the plate that would be able to relay position data to the system. Though the resistive panel offers a more stable and easier to implement solution, the price was out of the budget for this project. Of easily available touch panels, the best product to fit the application is the 3M 17-9311-25. This touch screen is smaller than the goal working area of two feet at only 22.37" in diagonal and costs \$264 from Mouser Electronics. Using a \$25 web camera with a resolution of only 480x480 pixels gives a resolution of one pixel every 1/20 inch. For the purposes of this project the resolution is considered satisfactory.

4.6. Software

As stated in the design requirements, the system must be able to operate on software that is either free or readily available to students at Southern Polytechnic State University. A review of the software in multiple labs throughout the campus revealed that the two solutions that were most widely available are LabView and Matlab/Simulink. LabView licenses installed on computers at the school however do not have the image acquisition and manipulation toolboxes that would be needed installed. While Matlab/Simulink installations on available computers also lacked the control systems toolbox they do have the image acquisition toolbox. Because of these factors Matlab/Simulink was chosen as the processing software.

4.7. Microcontroller

With the knowledge that the system would use DC Servos which are pulse width modulation controlled devices and Matlab/Simulink as the software to be used, a search for suitable hardware to control the system was conducted. Because of its availability and cost effectiveness, an Arduino Nano microcontroller was chosen. This decision is based mainly on the fact that the Arduino has readily available free software for interfacing with Matlab/Simulink. To use this interface, the Arduino Nano microcontroller is loaded with a 'server' program. The microcontroller then will be able to control the DC servos by sending a signal over USB using simple Matlab functions. This eased implementation and made the system very user friendly. The specifications for the Arduino Nano are shown below:

- Microcontroller Atmel ATmega168
- Operating Voltage 5 V
- Input Voltage 7-12 V
- Input Voltage (limits) 6-20 V
- Digital I/O Pins 14 (of which 6 provide PWM output)
- Analog Input Pins 8
- DC Current per I/O Pin 40 mA
- Flash Memory 16 KB of which 2 KB used by bootloader
- SRAM 1 KB
- EEPROM 512 bytes
- Clock Speed 16 MHz
- Dimensions 0.73" x 1.70"

4.8 Graphical User Interface

The following specifications were design requirements for the Graphical User Interface:

- Display x,y coordinates and velocity components of ball
- Graph position of ball
- Communicate with the Arduino and Simulink
- Allow controls options for communication with Arduino, for different display and image processing options, for different implementations of control algorithms and for control of plate
- A display box showing messages and next steps to take
- Display current angles of plate

GUIDE was used to simplify the development of the GUI. GUIDE is MATLAB's **Graphical User Interface Development Environment** and provides a set of tools for creating a GUI. Using the GUIDE Layout Editor, you can populate a GUI by clicking and dragging GUI components.

4.9 Controller Analysis and Design Verification

In order to design a controller for the ball and plate system, a mathematical model of the physical dynamics of the system is developed. Consider one dimensional ball and beam system. The variables used are defined below:

m	Ball mass
R	Ball radius
J	Ball moment of inertia
x	Ball position
g	Gravity
θ	Plate angle
r	Motor output gear radius
L	Length of plate
α	Motor angle

To help simplify the equations for ease of modeling, friction will be neglected and the assumption that no slippage will occur between the ball and the plate. Using these constraints the force equation centered around the motion of the ball is

$$F_{Gravity} = mg\sin(\theta)$$

$$F_{Normal} = mx \theta^2$$

$$F_{Ball} = \frac{J}{R^2} + m x$$

$$F_{Ball} + F_{Gravity} = F_{Normal}$$

$$\frac{J}{R^2} + m x + mg\sin \theta = mx \theta^2$$

$$0 = \frac{J}{R^2} + m x + mg\sin \theta - mx \theta^2$$

Now if the assumption of small angle changes are considered

$$\theta \approx 0$$

$$\sin \theta \approx \theta$$

$$\cos \theta \approx 1$$

$$0 = \frac{J}{R^2} + m x + mg\theta$$

Also, it may be desirable to have motor angel as a variable instead of plate angle. A relation between the two are given by

$$\theta = \frac{r}{L} \alpha$$

Substituting this relationship into the previous equation and solving for x

$$-mg\theta = \frac{J}{R^2} + m \ x$$

$$-mg \frac{r}{L} \ \alpha = \frac{J}{R^2} + m \ x$$

$$x = \frac{-mg \frac{r}{L} \ \alpha}{\frac{J}{R^2} + m}$$

This is a mathematical representation of system in one direction. Since we are assuming small angle motions, the two directions, x and y , are decoupled. This means the model above will apply to both direction independently. Therefore the equations in the x and y direction are

$$x = \frac{-mg \frac{r}{L} \ \alpha_y}{\frac{J}{R^2} + m} \qquad y = \frac{-mg \frac{r}{L} \ \alpha_x}{\frac{J}{R^2} + m}$$

Where α_y is the motor angle around the x axis and α_x is the motor angle around the y axis. For the system developed, the motor angle and the plate angle are in direct relation because of a 1:1 ratio used in the gearing.

Note that the inertia of a ball is given by two equations for either a solid ball or a hollow ball.

$$J_{Solid} = \frac{2}{5}mR^2 \qquad J_{Hollow} = \frac{2}{3}mR^2$$

Substituting this relationship into the previous equations and simplifying

$$x_{Solid} = \frac{-mg\theta_y}{\frac{2}{5}mR^2 + m} \quad y_{Solid} = \frac{-mg\theta_x}{\frac{2}{5}mR^2 + m}$$

$$x_{Solid} = -\frac{5}{7}g\theta_y \quad y_{Solid} = -\frac{5}{7}g\theta_x$$

$$x_{Hollow} = \frac{-mg\theta_y}{\frac{2}{3}mR^2 + m} \quad y_{Hollow} = \frac{-mg\theta_x}{\frac{2}{3}mR^2 + m}$$

$$x_{Hollow} = -\frac{3}{5}g\theta_y \quad y_{Hollow} = -\frac{3}{5}g\theta_x$$

Once the physical system has been modeled, a controller can be designed. By the nature of the system, many small miscalculations or calibration errors can dramatically affect the system response. Therefore, a state-space controller was chosen to help deal with these issues. Taking the equations for the x and y acceleration, a state-space model is obtained. Only the hollow ball case is shown.

$$\begin{matrix} x \\ y \\ x \\ y \end{matrix} = \begin{matrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \begin{matrix} x \\ y \\ x \\ y \end{matrix} + \begin{matrix} 0 \\ 0 \\ -\frac{3}{5}g \\ 0 \end{matrix} \begin{matrix} 0 \\ 0 \\ \theta_y \\ -\frac{3}{5}g \end{matrix} \begin{matrix} 0 \\ 0 \\ \theta_x \\ \theta_x \end{matrix}$$

$$\text{output} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix} \begin{matrix} x \\ y \\ x \\ y \end{matrix}$$

The initial controller design was setup to implement a settling time of 4 seconds. Using the “place” function in Matlab, a K gain matrix was obtained. Then the Simulink model shown in Figure 5 was used to simulate the controller. The output can be seen in Figure 6.

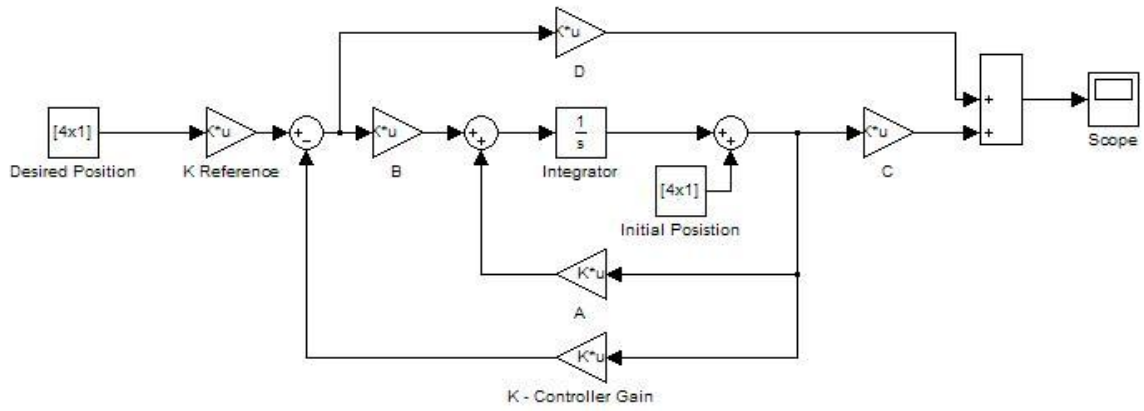


Figure 5: Controller Simulink Model

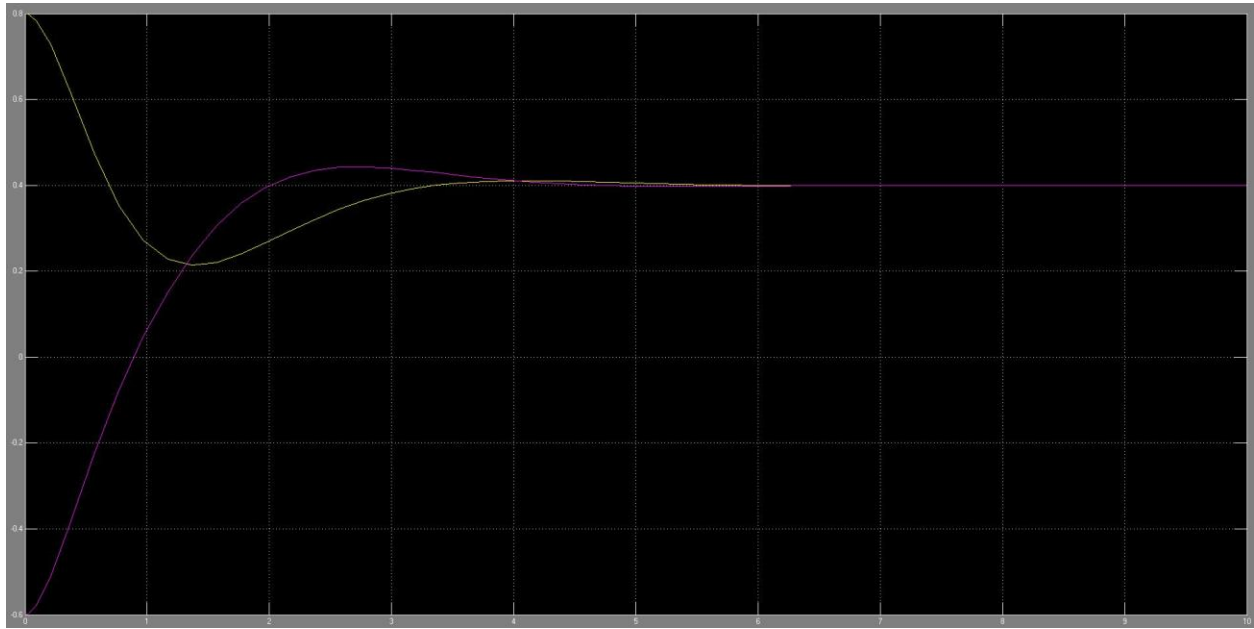


Figure 6: Initial Controller Simulation Response

5. Implementation

5.1. Tilt Mechanism

All effort was made to keep the prototype as close to the theoretical design as possible. With this in mind, construction of the tilt mechanism was built from 6063 Aluminum tubing. The structure was cut from raw materials, prepared, and welded with the help of Randle Johnson on the Southern Polytechnic State University campus. Figure 7 shows a picture of the structure during construction.



Figure 7: Tilt Mechanism During Construction

Figure 8 shows the method used to attach the shafts to the frame of the structure. A $\frac{1}{2}$ " by $1 \frac{1}{4}$ " steel plate was cut to length and drilled and honed through the center for the $\frac{3}{8}$ " shaft. Two additional holes were drilled for $\frac{1}{4}$ " bolts. Each shaft was then placed in its respective mounting bracket and the whole assembly was drilled through and pinned with a hardened pin. This pin holds the shaft from both rotational and translational motion relative to the mount.



Figure 8: Shaft Mount

All four shafts are solidly bolted using the $\frac{1}{4}$ " bolts to the outer rotating frame. Two shafts stick inward and through the inner rotating plate. The inner rotating frame is equipped with brass bushings, as shown in Figure 9, to limit friction and support load. Brass bushings were used in this area because of the limited thickness of the mounting surface. Using a roller bearing would require the frame tube size to be increased from 1" to $1\frac{1}{4}$ ", raising costs and the weight of the system.

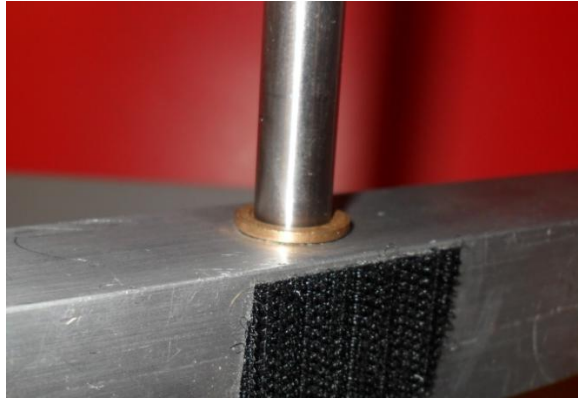


Figure 9: Brass Bushings

The remaining two shafts face outward from the outer rotating frame. These shafts are supported by a pillow block mounted bearing on each side as shown in Figure 10.

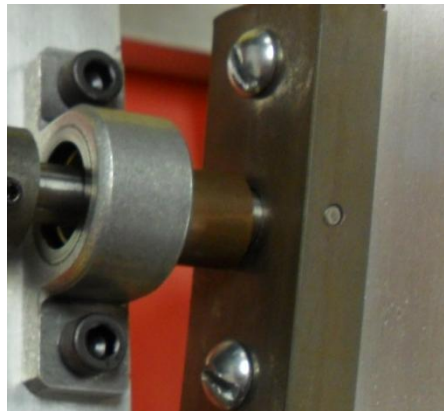


Figure 10: Pillow Block

To mount the actuators, simple mounts were made from $\frac{1}{4}$ " thick sheet aluminum. These were made to bolt to the main frame and not be welded. This design was implemented because it allows the motor mounts to be switched out, and therefore the actuation type changed for future study. Figure 11 shows the outside actuator mount and chain drive configuration for DC servo.

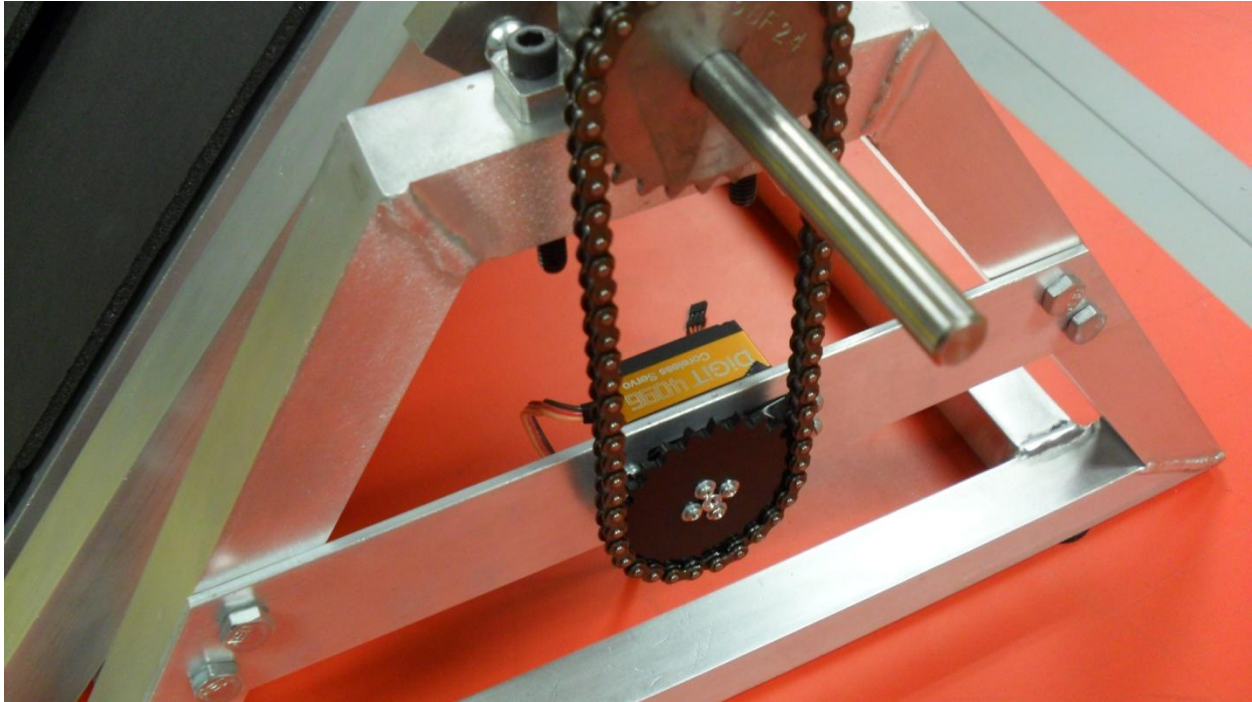


Figure 11: Actuation Mount and Chain Drive

5.2. Actuation

Construction began with the intent to use the Turnigy HV300, that was specified in the design specifications. After a short time, testing problems occurred and both actuators that were purchased sheared the shafts near the case as shown in Figure 12.



Figure 12: Broken Turnigy Servo

It was later determined that the cause of the failure in these two actuators was not using the actuator beyond its rated torque, but rather a combination of using too short of a mount screw for the sprocket and an over tightening of the chain. While replacing these with identical actuators would have been the best option, shipping times would have been two weeks so other options were investigated. Two Savox SC1258TG servos were found locally and purchased. The specifications for this servo can be found in Appendix H: Savox SC1258TG Specifications. Unfortunately before the Savox servos were able to be installed one was stolen.

Since the hobby servos also seemed to be struggling to move the outside frame, research was conducted to see what other options were available. The that was chosen was using a DC globe motor and a motor controller. A Globe Motor was readily available for free, part number 415A157-2. Since the implemented control signal was PWM, a controller that would accept PWM signal and translate that information to a DC voltage signal was needed. A search revealed the Roboteq AX500 motor controller. The AX500 is capable of driving two brushed DC motors of up to 15 amps each in a closed loop system to control either speed or position of the motor. Another reason for using this particular controller is that it allows an PWM input control signal for the motor position. This makes it plug and play for the Arduino controller and developed code. Specifications for the Roboteq AX500 can be found in Appendix I: Roboteq AX500 Specifications. Figure 13 shows the Globe motor mounted on its interchangeable mount.

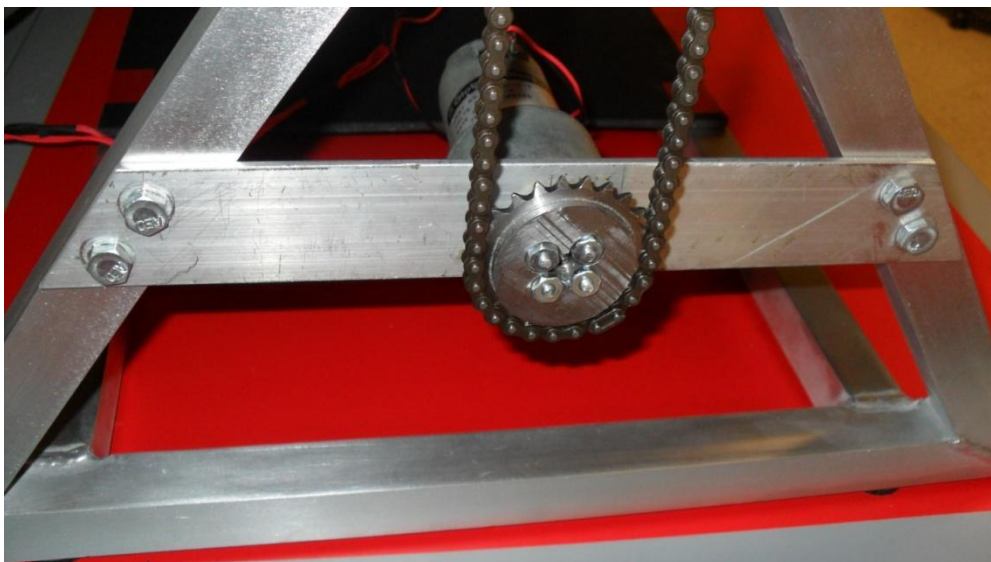


Figure 13: Mounted DC Motor

Figure 14 shows the method used to feedback the position data to the AX500 control module. A simple voltage divider is created using a linear rotary potentiometer. This potentiometer is attached directly to the plate opposite the motor to feedback actual plate angle. One drawback to the AX500 is that it only allows 8-bit resolution on the sensor in PWM control mode.



Figure 14: DC Motor Position Feedback

5.3. Power and Wiring

Figure 15 shows the arrangement for powering the system and enclosing electronics. With a rated maximum current of 5 amps for the Savox servo and 7 amps for the Globe DC motor, a high performance power supply was needed. Lab power supplies that were available on the Southern Polytechnic State University campus fell well short of the needed amperage. The most economical solution was a personal computer power supply. One was obtained locally for under \$25 that had 3.3, 5, and two 12 volt rails and was capable of supplying 27 amps at the 12 volt level.

The enclosure for the Arduino microcontroller and the Roboteq motor controller is also shown in Figure 15. On the upper right corner is the serial connector for the Roboteq controller. Connecting this to a computer allows you to change specifications such as PID constants through Roboteq's Roborun utility. The two wires entering the enclosure on the top are the control wires for the Savox servo and the DC motor respectively. Starting from the lower left of the enclosure the switch turns off/on the Roboteq controller, the first connector leads to the potentiometer feedback, the second connector leads to a safety shut off switch, the third connector leads to the DC motor, and the last connector brings power to the enclosure.

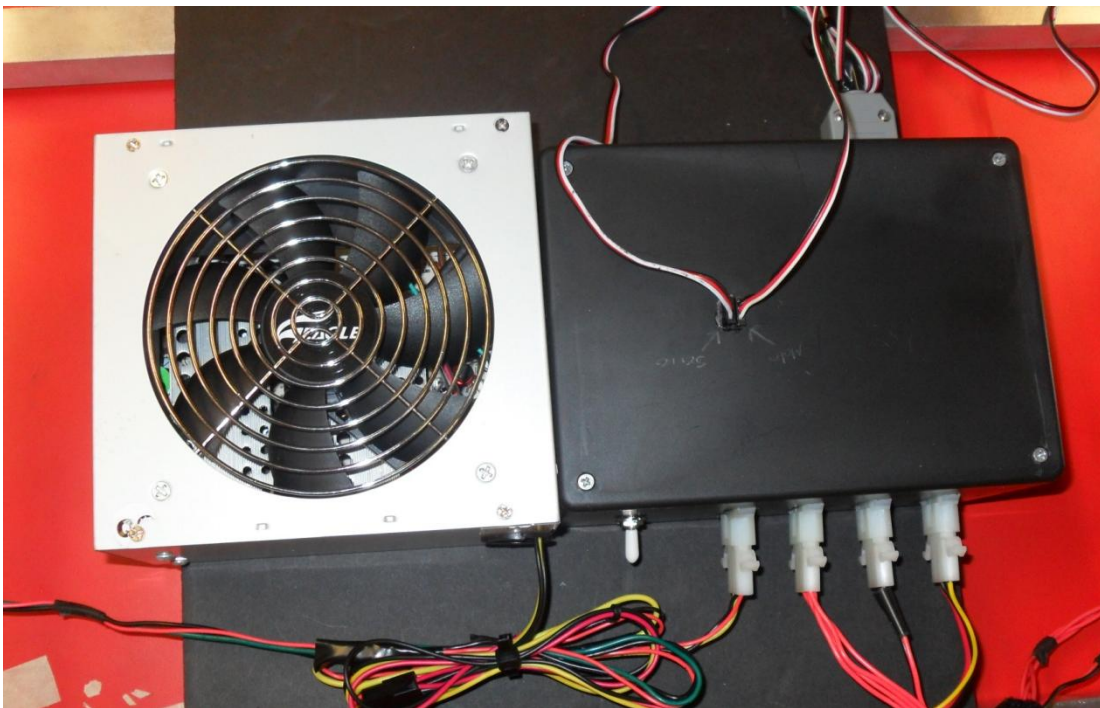


Figure 15: Power and Wiring

5.4. Vision Processing

A video camera is placed above the plate system which is used to get x and y coordinates of the ball, along with the translational velocity. Using the Video and Image Processing blockset in Simulink, the centroid of the ball can be found. To find the centroid, an auto threshold algorithm followed by a blob analysis algorithm can be applied. The Autothreshold block converts an intensity image to a binary image using a threshold value computed using Otsu's method. This block computes this threshold value by splitting the histogram of the input image such that the variance of each pixel group is minimized. Figure 16 shows the results of the image processing,

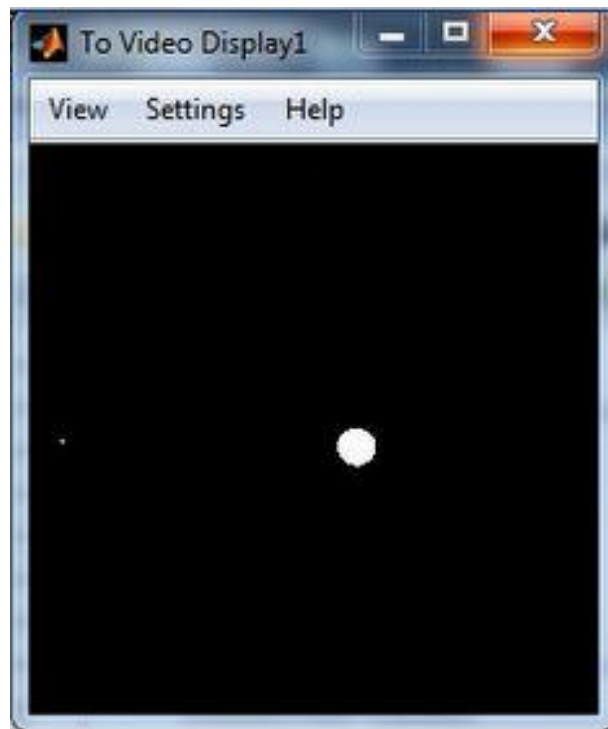


Figure 16: Autothresholding of the Image

Following the autothresholding, the Blob Analysis block receives the binary image and does analysis on the prominent blobs, outputting the minor axis and centroid of each blob. Using an Embedded MATLAB Function block, the minor axis is used to find which blob is the actual ball. Figure 17 shows the live feed of the ball being tracked.

Once the centroid of the ball was found, the coordinates were changed from the camera coordinates to new system coordinates. This change was applied to make the origin the center of the plate and to account for the rotation of the camera on top of the plate.

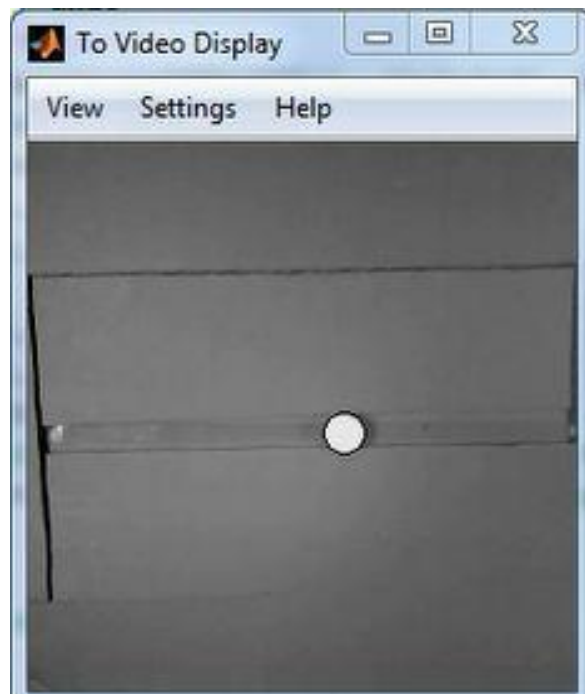


Figure 17: Ball Tracking

The centroid of the ball is sent to the S-function block and the velocity components of the ball are calculated by storing the x and y coordinates into global variables and using the *tíc* and *toc* functions to find the elapsed time between different displacement positions.

The option of turning on and off the feeds of the camera was used to allow better performance of the system. By having the feeds on, the processing power is cut and significantly affects the frame rate of the image processing.

Two different options for the feed were given, Live Feed and Threshold Feed, both of which were made possible with the use of the Simulink blocks in Figure 18. The final setup of the Simulink model can be found in Appendix D.

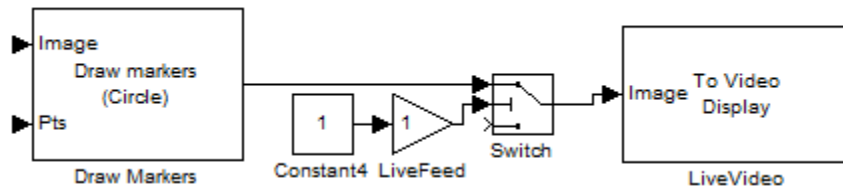


Figure 18: Simulink Model for Viewing the Ball and Plate

5.5. Graphical User Interface

A graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices with images rather than text commands. The GUI will serve two purposes. First, the GUI will be used to facilitate the development stages of the Ball and Plate system. By having a GUI, the developers will be able to troubleshoot the equipment and debug the control algorithm without having to change the text code. Second once the ball and plate system is developed and fully built, the GUI will allow any user to operate the plate. It will also allow a user to try and balance the ball where they deem appropriate. Figure 19 shows the final design of the GUI. All the inputs and outputs were test for quality assurance.

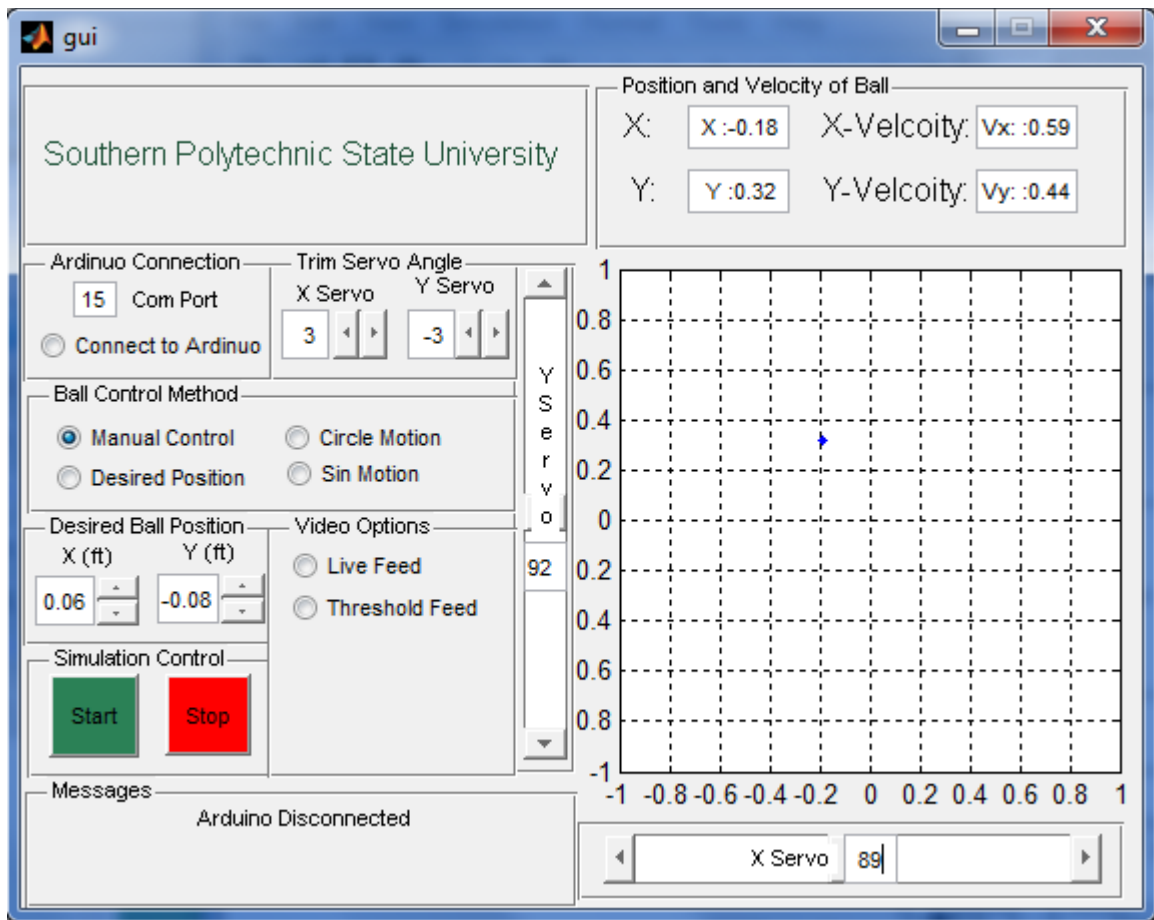


Figure 19: Finalized Graphical User Interface (GUI)

5.6. State-Space Controller

The controller developed in section 5 was implemented on the physical system with some mixed outcomes. First, the latency of the system, which was not accounted for in our mathematical model, was much higher than expected. Second, the state gain was not large enough to move our servos. Therefore, the poles of the controlled system were pushed farther to the left to account for the oversight in the system modeling and latency. Upon simple trial and error, the finalized controller poles were implemented. The new system output can be seen in Figure 20Figure 19.

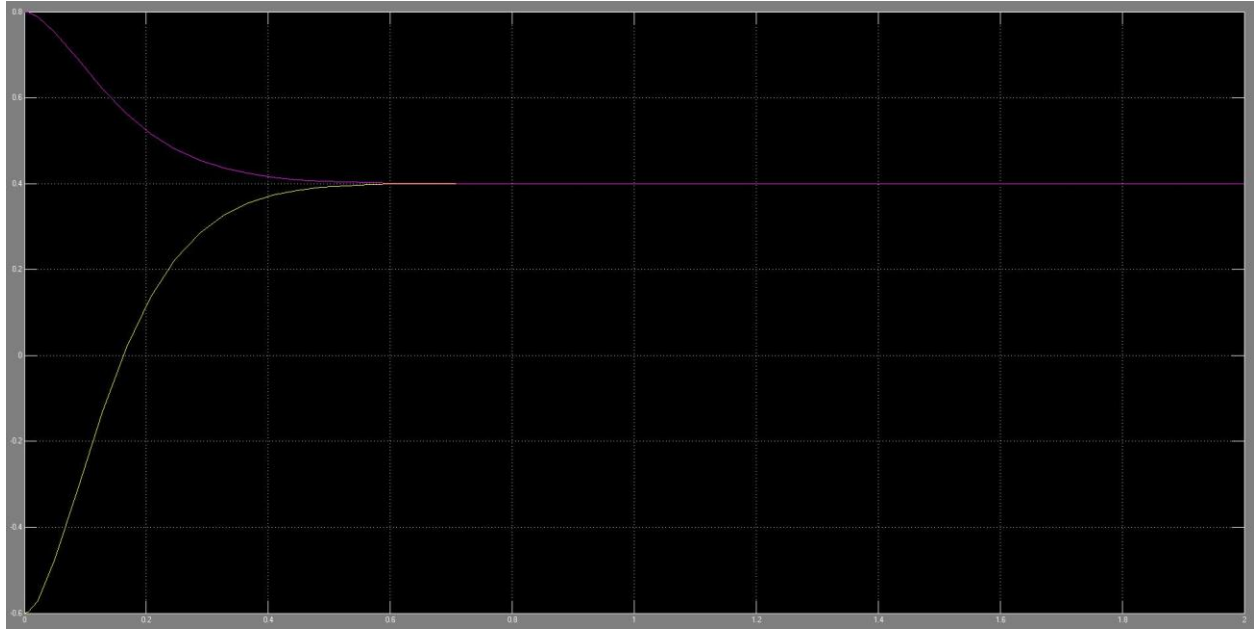


Figure 20: Finalized Controller Simulation

6. Test Data

Many test were run using the ball and plate system. For many of the test, different poles and different desired positions on the plate were used. Table 6 below shows the best results for when a ping pong ball was thrown at different velocities and angles with the poles set at $-11\pm 4i$ and $-11.5\pm 4i$ and the desired position set at $(0,0)$.

Table 6: Test Results

Run	Settling Time (sec)	Error Along x(ft)	Error Along y(ft)
1	13	0.04	0.1
2	15	0.03	0.08
3	16	0.05	0.12
4	10	0.03	0.9
5	9	0.01	0.1
Average	12.6	0.032	0.26

The error along the y axis is large due to the fact that the DC motor controller. As mentioned in section 5.2, the low sensitivity of the sensor made the controller have a large steady state error that could not be overcome by our controller. These results were consistent when the desired position was set somewhere other than at the origin $(0,0)$.

7. Optimization

Due to early failure of the chosen actuators several compromises were made in the actuation of the system. The Savox servo is capable but not as strong as what is really needed and the Roboteq controlled DC motor is very strong but rather slow and has low resolution on the position sensor. A possible solution to these problems is a rather expensive stepper motor. The stepper motor would need to be geared to overcome its weak area, a 1.2-2.5 degree step resolution. The stepper motor would then be able to provide positive and accurate position control.

In order to control the motors accurately and effectively a more robust microcontroller or computer interface would also need to be developed. While the Arduino itself provides very low lag in its control the characteristics of the PWM signal and the cascading of several different controllers introduces latencies. A serial control signal would be the best option to replace the PWM signal.

To take advantage of the faster communication between controllers the processing would need to be improved as well. With every feature of the current program working frame rates in the camera lagged to as low as 2-5 frames per second. Matlab was also using a full 1/2 gigabyte of memory to run with the frame resolution set to 240x240 pixels. The solution to this is using the multithreading capabilities of Matlab by way of a mex file. Figure 21 shows the effect of using a mex file compared to the regular Matlab file in an image processing function. As can be seen the processing time using a regular Matlab function increases exponentially with image size, limiting image resolution and therefore restricting the robustness of the program. Using a mex file on the other hand is fairly flat with an increase in picture size, which would allow the full use of the 720x720 pixel camera.

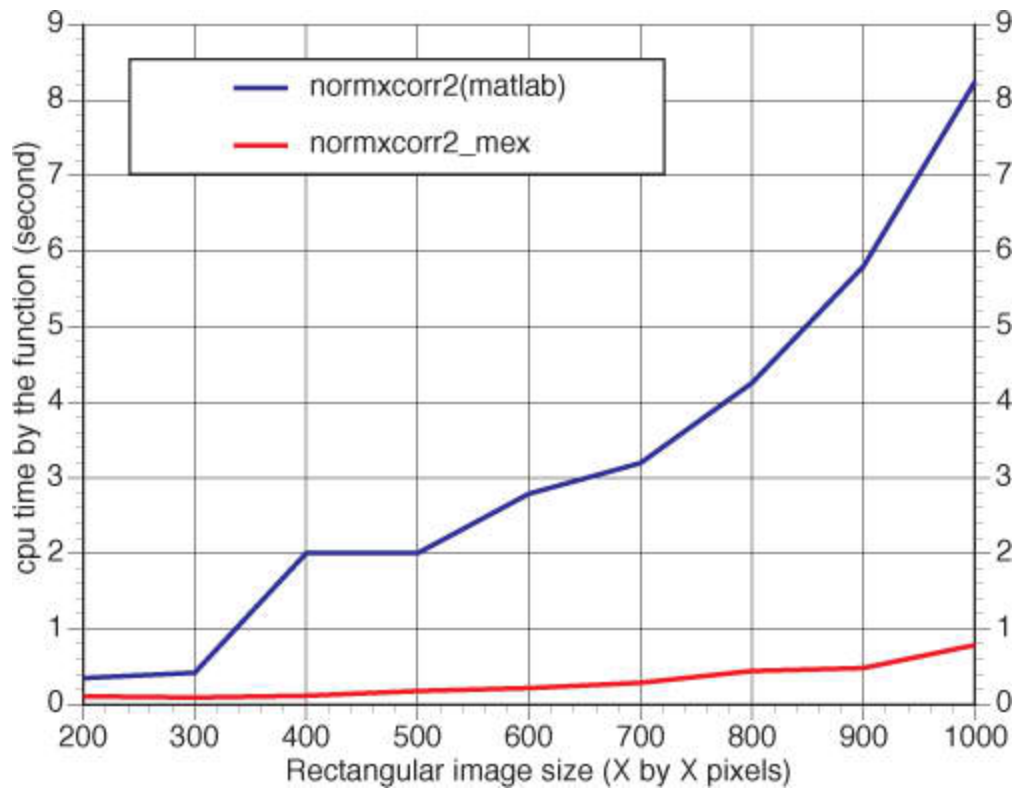


Figure 21: Matlab vs Mex Comparison

It would also be of value to increase the robustness of the image processing. This could be done by including more disturbance rejecting features as well as more object recognition filters.

8. Conclusion

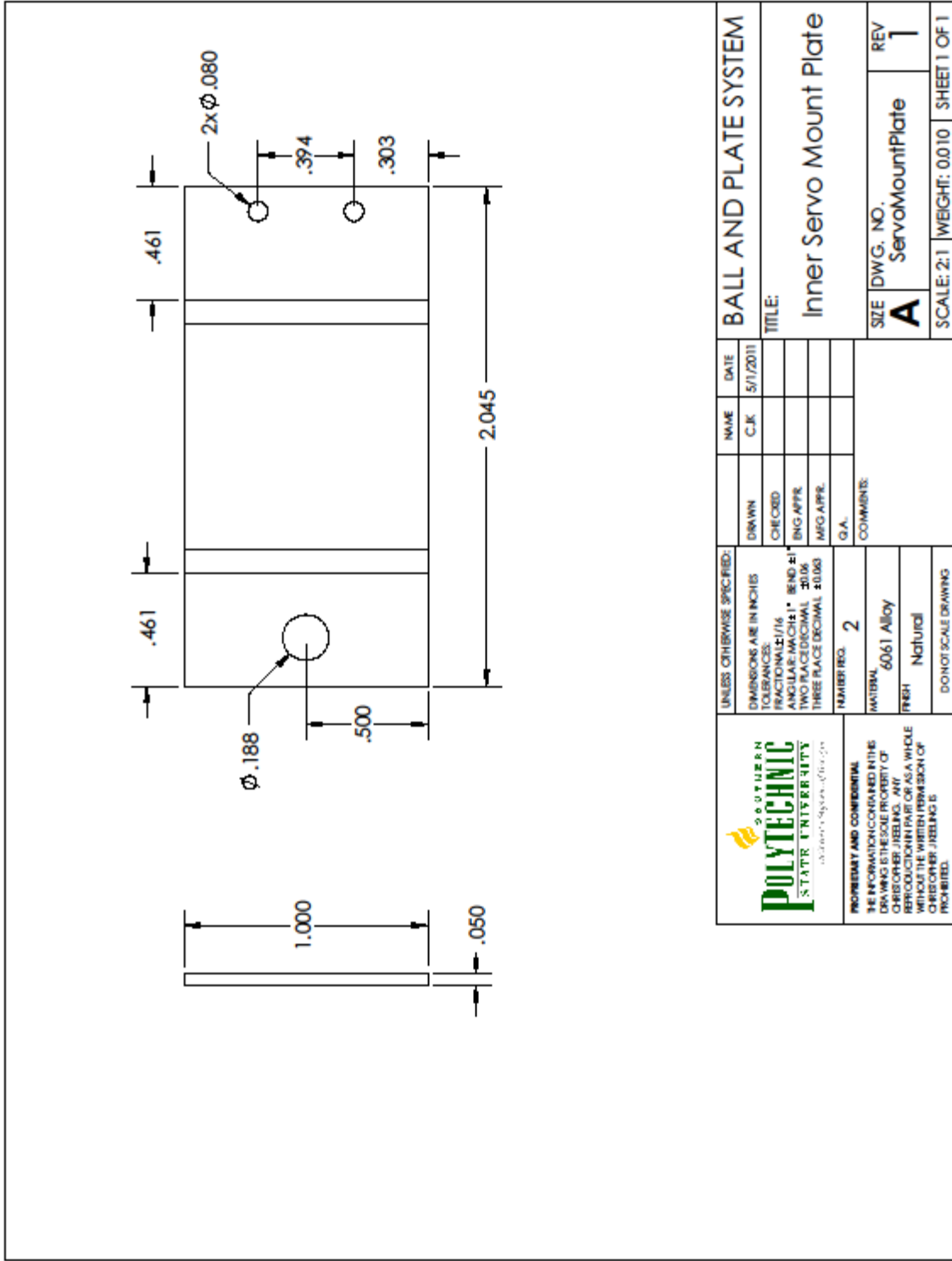
A ball and plate system was developed which met and exceeded the minimum performance success criteria. Though some of the initial specifications had to be changed during prototyping, the main idea of the ball and plate system was completed. The system was able to push a ball from a random position on the plate to a user desired position within an average of 15 seconds and with a maximum error of 20%. This error was accounted for in the fact that our secondary DC motor actuation system did not have the sensitivity to settle the ball close to our desired location.


Throughout the semester, major engineering design steps were implemented on a practical basis. Though there was a compressed timeline, the basic structure of report writing, engineering economy, theoretical design, prototyping, and optimization were learned. Also, the functional requirements, alternative concepts, design requirements, minimum performance success criteria, physical design, control system, prototyping and testing, as well as some optimizations were met. Each team member now has a practical understanding of how an engineering design project is implemented.

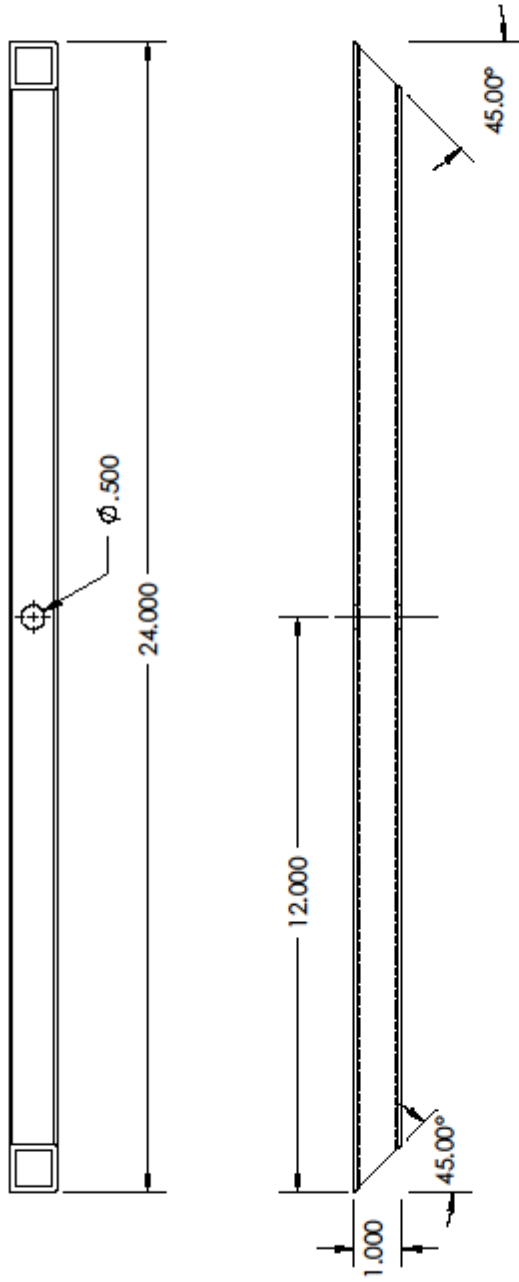
Reference


- [1] “Graphical User Interface.” [Online]. Available:
http://en.wikipedia.org/wiki/Graphical_user_interface [Accessed: April 1, 2011]
- [2] “MATLAB.” [Online]. Available:
<http://en.wikipedia.org/wiki/MATLAB> [Accessed: April 1, 2011]
- [3] “Simulink.” [Online]. Available:
<http://en.wikipedia.org/wiki/Simulink> [Accessed: April 1, 2011]
- [4] MathWorks. “MATLAB Support for Arduino.’ [Online]. Available:
<http://www.mathworks.com/academia/arduino-software/arduino-matlab.html>
[Accessed: March 13, 2011]
- [5] MathWorks. *R2011a Documentation*. [Online]. Available:
<http://www.mathworks.com/help/index.html> [Accessed: March 13, 2011]
- [6] Keeling, Christopher J. *Modeling a Ball and Beam System Driven by an Electric Motor*. 23 Nov. 2010.

Appendix A: Technical Drawings

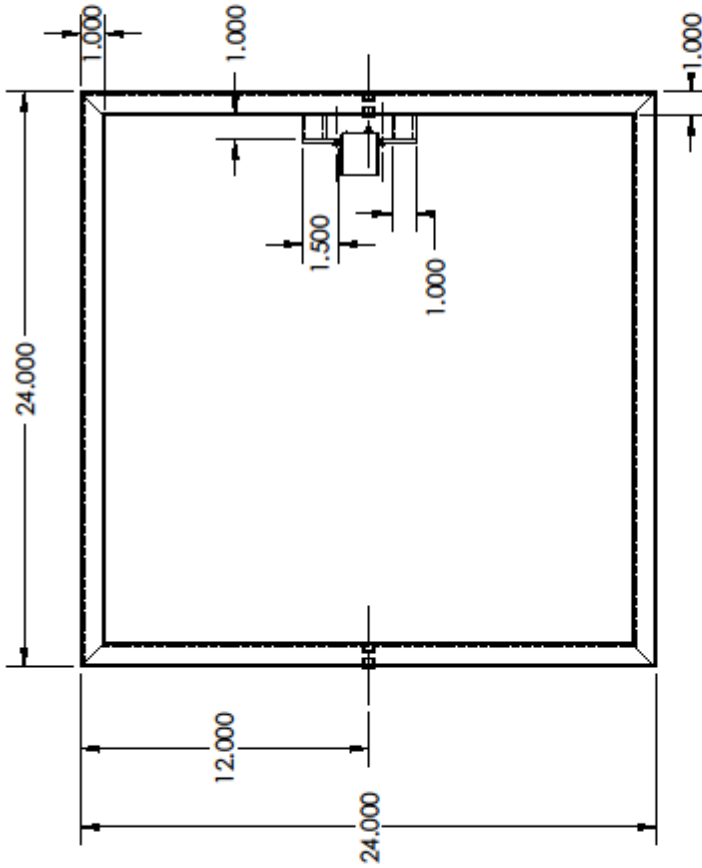


 <p>SOUTHERN POLYTECHNIC STATE UNIVERSITY <i>Southern Polytechnic Institute</i></p>		UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±.015 ANGULAR ±.031° BEND ±1° TWO PLACE DECIMAL ±.008 THREE PLACE DECIMAL ±.003		DRAWN CHECKED ENG APPR MFG APPR G.A. COMMENTS:	NAME C.J.K.	DATE 5/1/2011	BALL AND PLATE SYSTEM TITLE: Inner Servo Mount Plate	SIZE DWG. NO. ServoMountPlate REV 1
MATERIAL 6061 Alloy FINISH Natural		NUMBER OF 2	SCALE: 2:1 WEIGHT: 0.010 SHEET 1 OF 1		DO NOT SCALE DRAWING		1	
PROPRIETARY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF CHRYSLER JEEP LINC. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF CHRYSLER JEEP LINC. IS PROHIBITED.		2		3		4		5

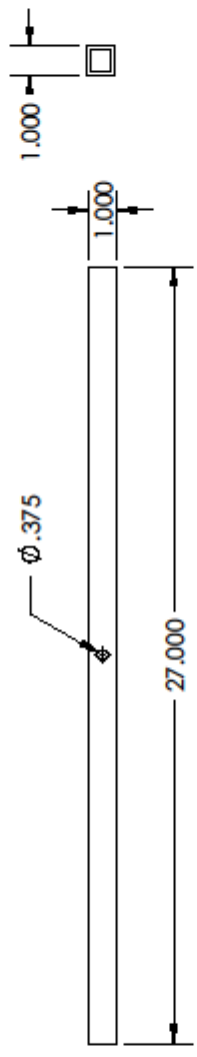


 <p>SOUTHERN POLYTECHNIC STATE UNIVERSITY www.southern.edu</p>	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL: 1/16 DECIMAL: 0.005 ANGULAR: MACH 1° BEND ±1 TWO PLACE DECIMAL ±0.004 THREE PLACE DECIMAL ±0.003		NAME	DATE	BALL AND PLATE SYSTEM TITLE: INNER BAR
	DRAWN	CJK	2/12/2011		
	CHECKED				
	ENG APPR				
			MFG APPR		
			G.A.		
			COMMENTS: ONLY 2 OF THE REQUIRED 4 PIECES NEED HOLE DRILLED		
			NUMBER REQ.	4	
			MATERIAL	6061 Alloy	
			FINISH	NATURAL	
			DO NOT SCALE DRAWING		
PROPRIETARY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF CHRISTOPHER J. REELING. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF CHRISTOPHER J. REELING IS PROHIBITED.					
			SCALE:	1:3	WEIGHT: 0.966
			SIZE	DWG. NO.	REV
			A	innerbar	1
					SHEET 1 OF 1

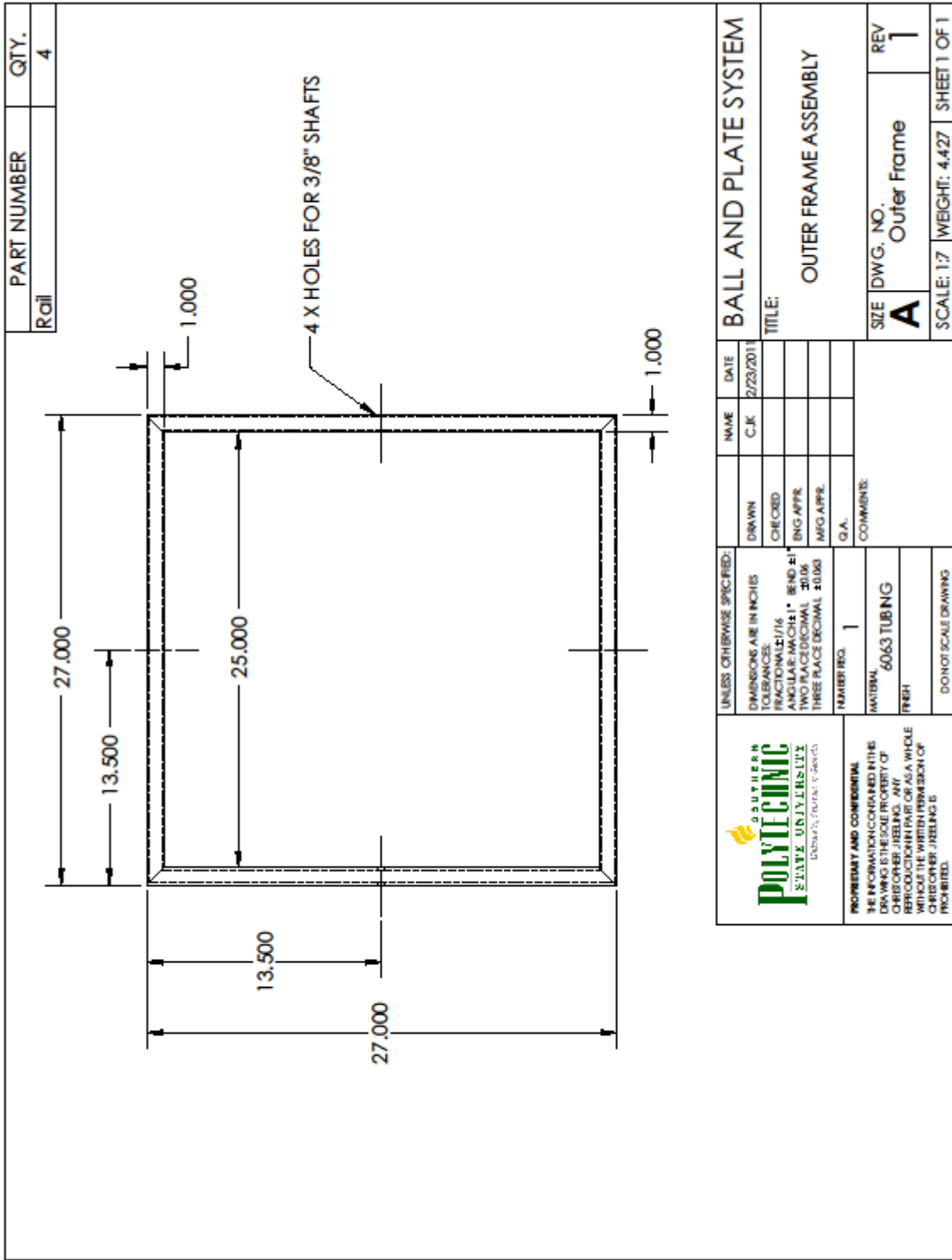
PART NUMBER	QTY.
innerbar	4
Bronze Bushing	4
MKS HV300 servo	1
ServoMount	2



UNLESS OTHERWISE SPECIFIED:		NAME	DATE	BALL AND PLATE SYSTEM	
DIMENSIONS ARE IN INCHES		CJK	2/23/2011	TITLE:	
TOLERANCES:		DRAWN		INNER FRAME ASSEMBLY	
FRACTIONAL 1/16		CHECKED		SIZE	DWG. NO.
ANGULAR MACH 1° BEND ±1		ENG APPR		A	innerframe
TWO PLACE DECIMAL ±0.06		MFG APPR		SCALE: 1:6	WEIGHT: 4.649
THREE PLACE DECIMAL ±0.003		G.A.		SHEET 1 OF 1	REV
HATCH REQ. 1		COMMENT:			1
MATERIAL		A ASSEMBLY IS MADE OF MIXED MATERIAL INCLUDING ALUMINUM TUBING			
FINISH		DO NOT SCALE DRAWING			
<p>PROPRIETARY AND CONFIDENTIAL</p> <p>THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF CHRIS CORP. JEREBING. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF CHRIS CORP. JEREBING IS PROHIBITED.</p>					



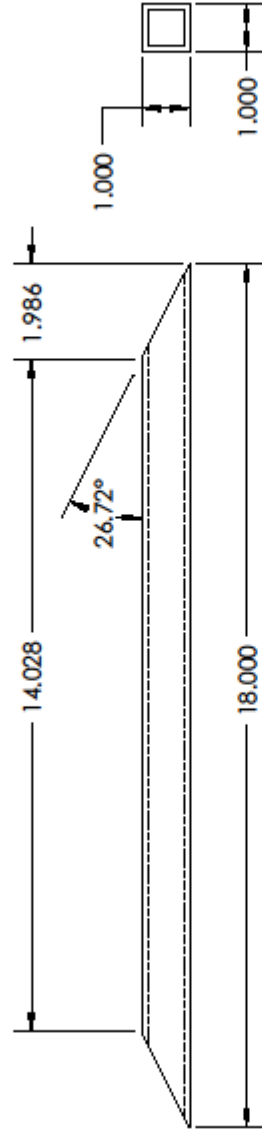
	UNLESS OTHERWISE SPECIFIED:		NAME	DATE	BALL AND PLATE SYSTEM OUTER FRAME RAIL
	DIMENSIONS ARE IN INCHES		C./K.	3/5/2011	
	TOLERANCES:		DRAWN		
	FRACTIONAL 1/16		CHECKED		
ANGULAR MATCH 1°		ENG APPR.			TITLE:
TWO PLACE DECIMAL		MFG APPR.			SIZE DWG. NO. Rail
THREE PLACE DECIMAL		G.A.			A
NUMBER REQ. 4		COMMENTS:			SCALE: 1:5 WEIGHT: 1.107 SHEET 1 OF 1
MATERIAL 6061 Alloy					
FINISH FRESH					
DO NOT SCALE DRAWING					
PROPRIETARY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF CHRISTOPHER J. REELING. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF CHRISTOPHER J. REELING IS PROHIBITED.					




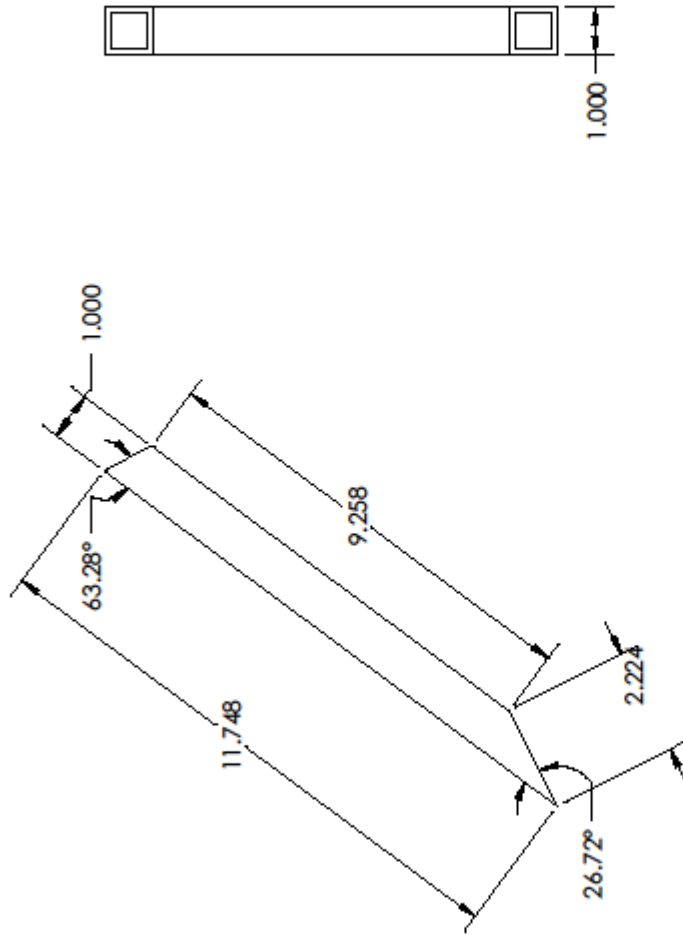
BALL AND PLATE SYSTEM		TITLE:	OUTER FRAME ASSEMBLY	
DRAWN	NAME	DATE	DWG. NO.	REV
CHECKED	C.J.K.	2/23/2011	A	1
ENG APPR.			SCALE: 1:7	WEIGHT: 4.427
MFG APPR.			SHEET 1	OF 1
G.A.				
COMMENTS:				
6063 TUBING				
FINISH				
DO NOT SCALE DRAWING				


UNLESS OTHERWISE SPECIFIED:
 DIMENSIONS ARE IN INCHES
 TOLERANCES:
 FRACTIONAL ±.010
 DECIMAL ±.005
 ANGULAR ±.01°
 HOLE ±.005
 HOLE POSITION ±.010

PROPERTY AND CONFIDENTIAL
 THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF CHRISTOPHER J. JEBEL, ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF CHRISTOPHER J. JEBEL IS PROHIBITED.

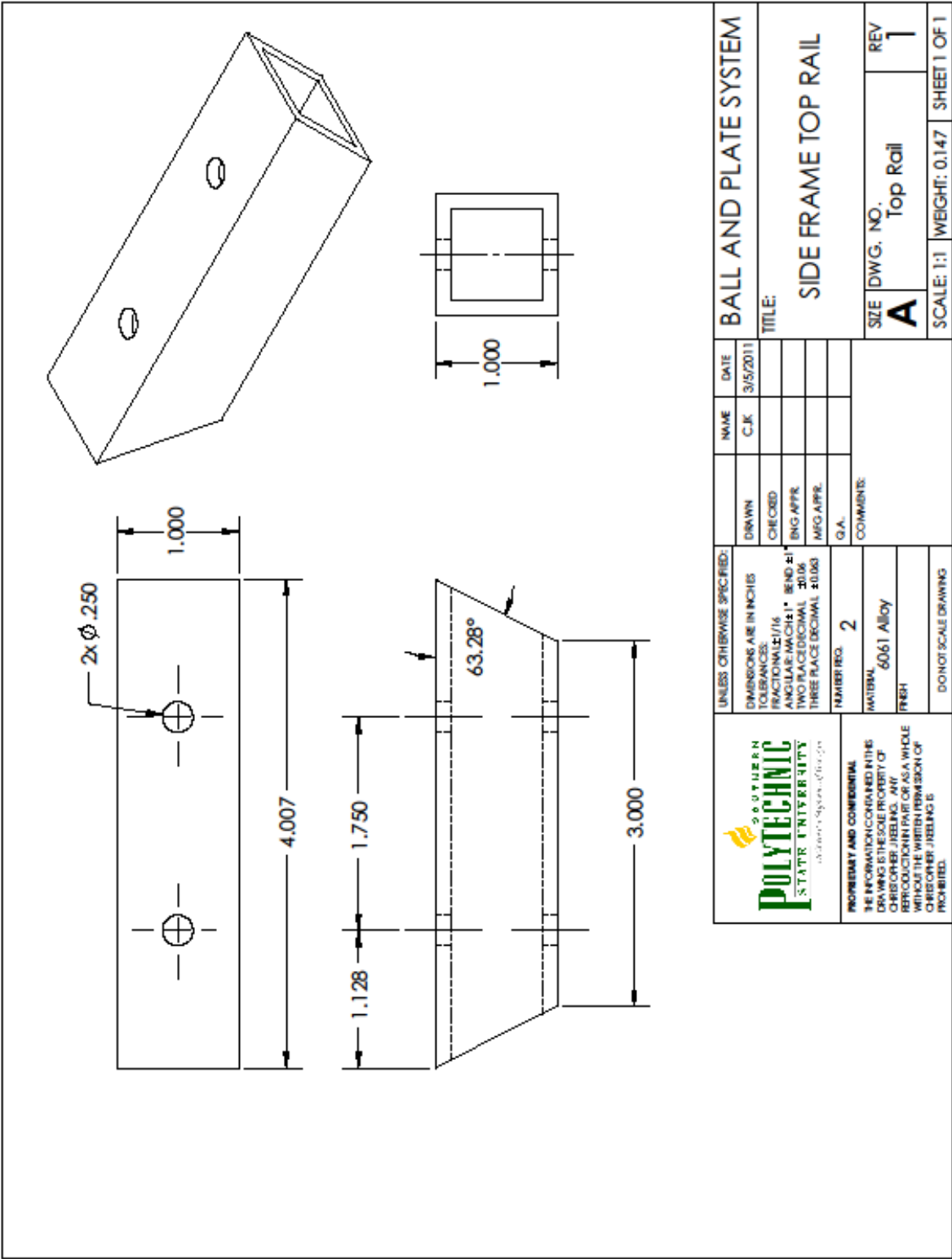



 <small>PROPERTY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF CHRYSLER JEEP INC. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF CHRYSLER JEEP INC IS PROHIBITED.</small>	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± 1/16 ANGULAR ± 0.031° BEND ± 1 TWO PLACE DECIMAL ± 0.004 THREE PLACE DECIMAL ± 0.0003		NAME CJK	DATE 3/5/2011	BALL AND PLATE SYSTEM TITLE: SIDE FRAME BOTTOM RAIL
	DRAWN CHECKED ENG APPR MFG APPR Q.A. COMMENTS:	MFG REQ. 2 MATERIAL 6061 Alloy FINISH FRESH	DO NOT SCALE DRAWING	SCALE: 1:3 WEIGHT: 0.683 SHEET 1 OF 1	

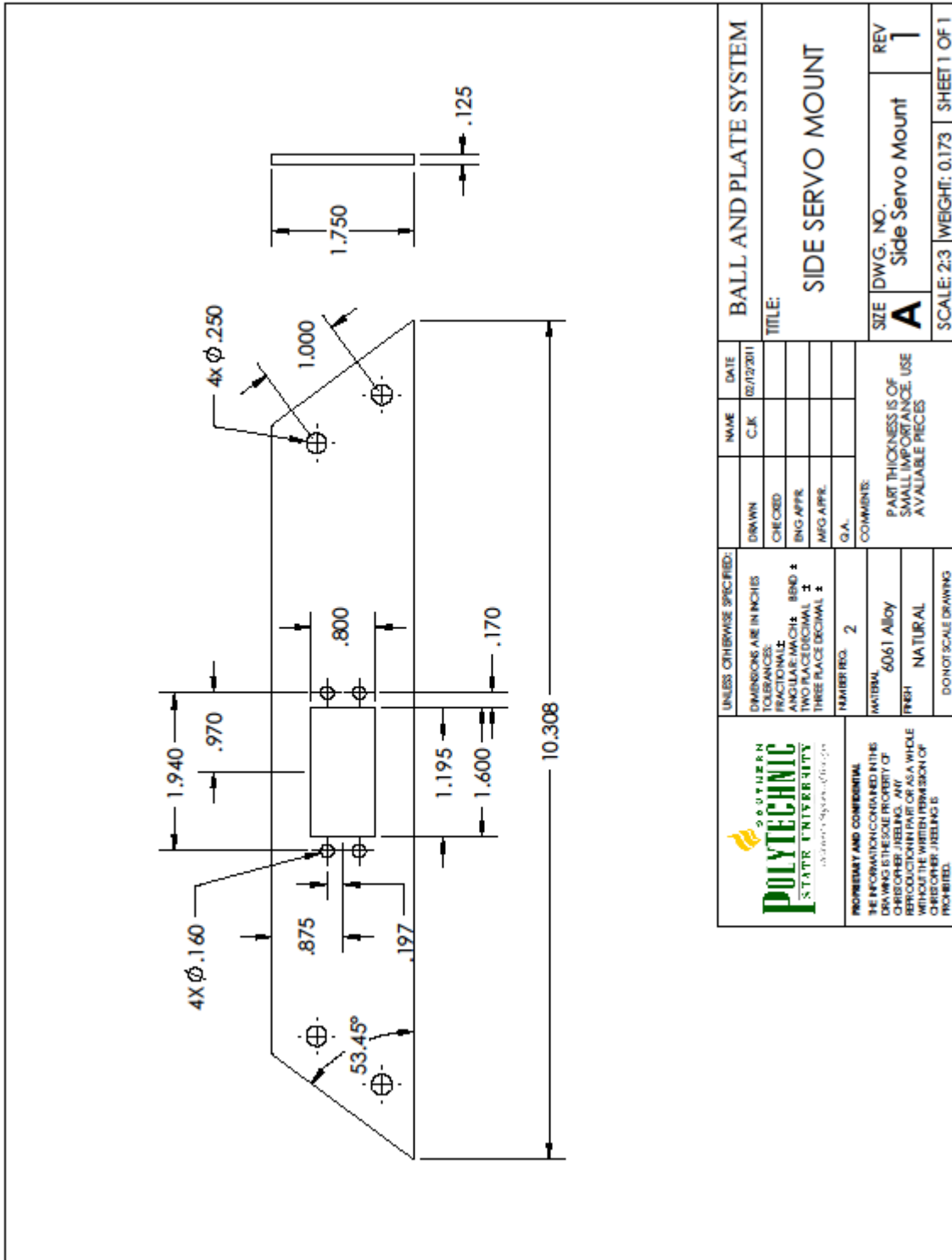



 <p>SOUTHERN POLYTECHNIC STATE UNIVERSITY www.spsu.edu</p>	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±1/16 ANGULAR ±0.041° BEND ±1° TWO PLACE DECIMAL ±0.005 THREE PLACE DECIMAL ±0.003		DRAWN CJK	DATE 3/5/2011	BALL AND PLATE SYSTEM SIDE FRAME SIDE RAIL
	MATERIAL 6061 ALLOY FINISH NATURAL	NAME/REV. COMMENTS:	CHECKED ENG. APPR. MFG. APPR. Q.A.	SIZE DWG. NO. A Side Rail	
PROPRIETARY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF CRIBCORP/JRELENG. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF CRIBCORP/JRELENG IS PROHIBITED.		DO NOT SCALE DRAWING	SCALE: 1:3 WEIGHT: 0.448	SHEET 1 OF 1	

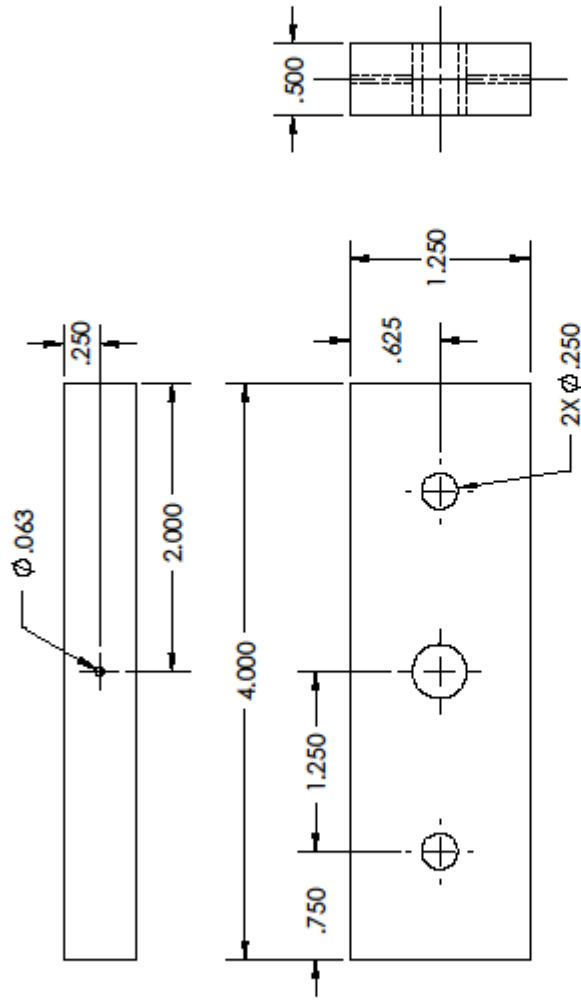
5 4 3 2 1




	UNLESS OTHERWISE SPECIFIED:		NAME	DATE	BALL AND PLATE SYSTEM TITLE: SIDE FRAME TOP RAIL
	DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± 1/16 ANGULAR MATCH ± 1° BEND ± 1° TWO PLACE DECIMAL ± 0.06 THREE PLACE DECIMAL ± 0.005	DRAWN CHECKED ENG APPR. MFG APPR. G.A.	CJK 3/5/2011		
PROPERTY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF SOUTHERN POLYTECHNIC INSTITUTE. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF SOUTHERN POLYTECHNIC INSTITUTE IS PROHIBITED.	NUMBER NO. 2 MATERIAL 6061 Alloy FINISH	COMMENTS:	SIZE DWG. NO. A Top Rail	REV 1	SCALE: 1:1 WEIGHT: 0.147 SHEET 1 OF 1

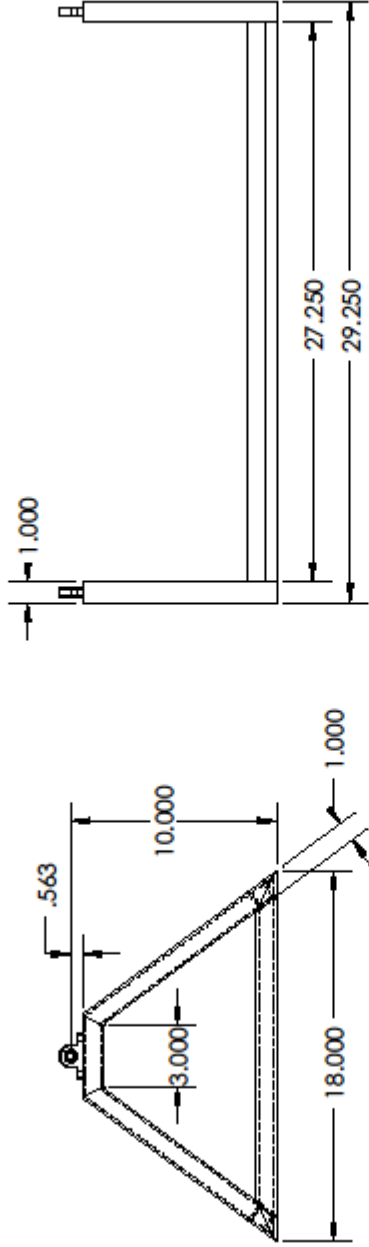


 <p>SOUTHERN POLYTECHNIC STATE UNIVERSITY <i>Southern Polytechnic Institute</i></p>	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL: ANGULAR: MACH ± .0001 TWO PLACE DECIMAL ± .005 THREE PLACE DECIMAL ± .001		NAME C.J.K.	DATE 02/12/2011	BALL AND PLATE SYSTEM	
	MATERIAL: 6061 ALLOY FINISH: NATURAL PART THICKNESS IS OF SMALL IMPORTANCE. USE AVAILABLE PIECES DO NOT SCALE DRAWING		COMMENTS: DRAWN CHECKED ENG. APPR. MFG. APPR. Q.A.		TITLE: SIDE SERVO MOUNT	
PROPRIETARY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF CHRISTOPHER J. REILING. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF CHRISTOPHER J. REILING IS PROHIBITED.					SIZE A	DWG. NO. Side Servo Mount
					SCALE: 2:3	WEIGHT: 0.173
					SHEET 1 OF 1	



 <p>SOUTHERN POLYTECHNIC STATE UNIVERSITY <small>University System of Georgia</small></p>	UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±0.015 ANGULAR ±0.031° BEND ±1° TWO PLACE DECIMAL ±0.004 THREE PLACE DECIMAL ±0.003		NAME C.J.K.	DATE 5/1/2011	BALL AND PLATE SYSTEM TITLE: SHAFT MOUNT	
	DRAWN CHECKED ENG. APPR. MFG. APPR. Q.A.	NUMBER 4	COMMENT: DRILL PIN HOLE WITH SHAFT IN PLACE	SIZE A		DWG. NO. shaftmount
MATERIAL: Plain Carbon Steel FINISH:			DO NOT SCALE DRAWING		SCALE: 1:1 WEIGHT: 0.674	SHEET 1 OF 1

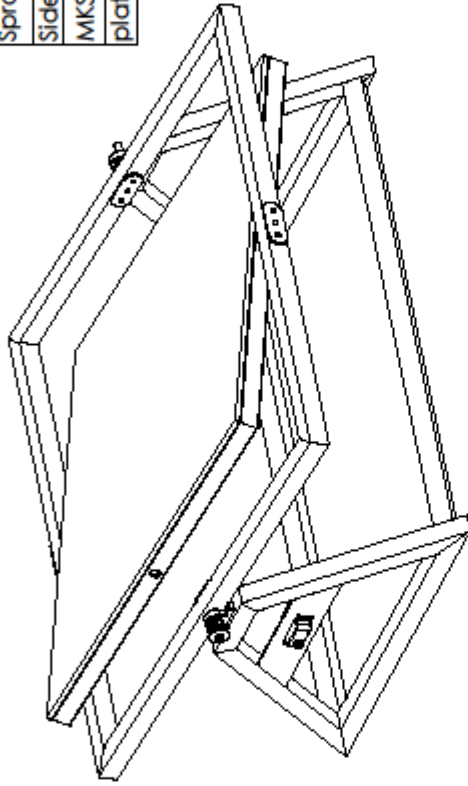
PART NUMBER	QTY.
Bottom Rail	2
Side Rail	4
Top Rail	2
Runner	2
Pillowblock	2




UNLESS OTHERWISE SPECIFIED:		NAME	DATE	BALL AND PLATE SYSTEM	
DIMENSIONS ARE IN INCHES		CJK	2/23/2011	TITLE:	
TOLERANCES		DRAWN		BOTTOM FRAME ASSEMBLY	
FRACTIONAL 1/16		CHECKED		SIZE	DWG. NO.
ANGULAR MACH 1° BEND ±1		ENG APPR		A	Bottom Frame A ssembly
TWO PLACE DECIMAL ±0.005		MFG APPR		SCALE: 1:1	WEIGHT: 6.078
THREE PLACE DECIMAL ±0.003		G.A.		SHEET 1 OF 1	
MATERIAL		COMMENTS:			
FINISH		DO NOT SCALE DRAWING			
PROPRIETARY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF CRIS CORP. ANY REPRODUCTION OR TRANSMISSION IN ANY FORM OR BY ANY MEANS WITHOUT THE WRITTEN PERMISSION OF CRIS CORP. IS PROHIBITED.					

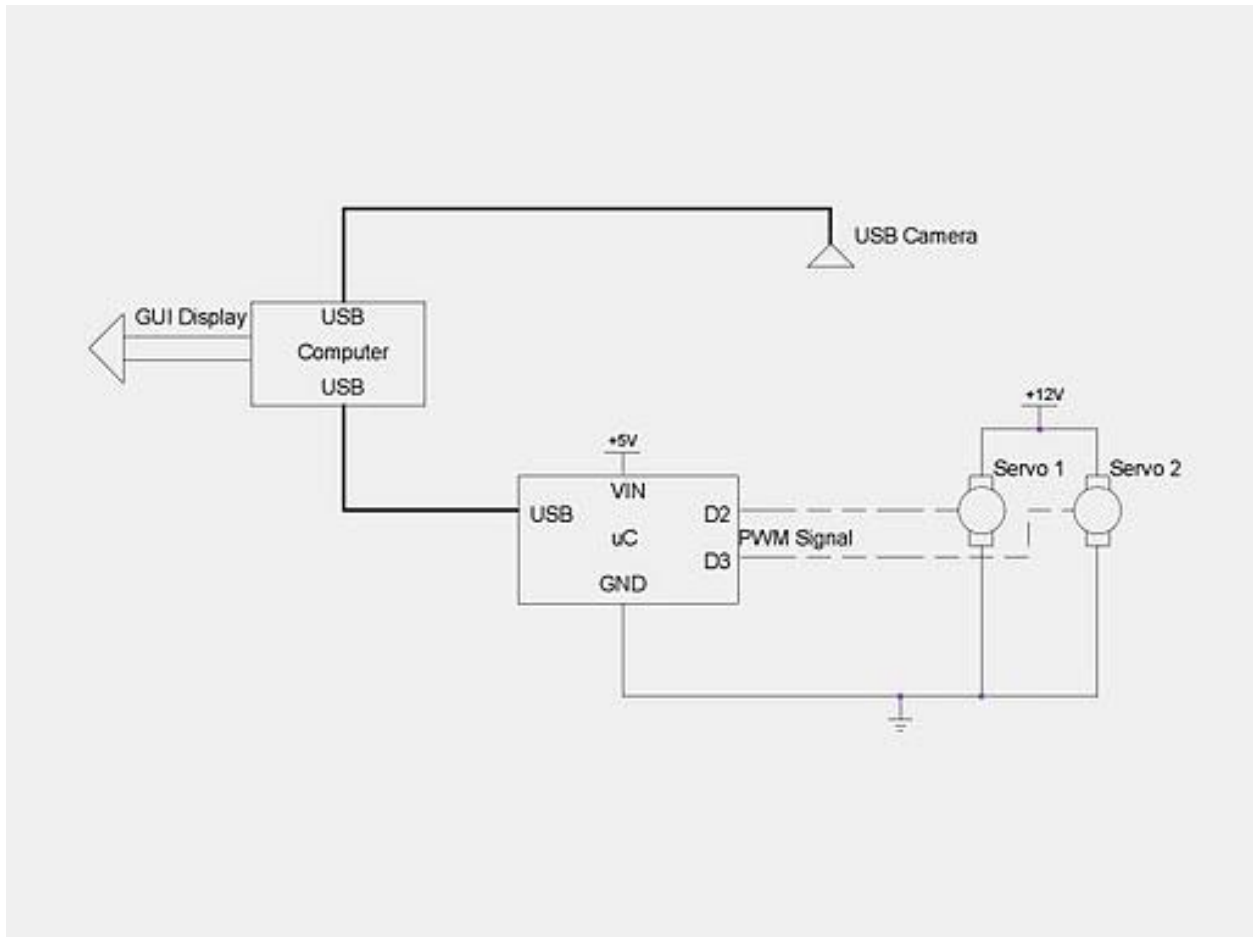
5 4 3 2 1

PART NUMBER	QTY.
Bottom Frame Assembly	1
Outer Frame innerframe	1
Shaft	4
Sprocket	1
Side Servo Mount	1
MKS HV300 servo plate	2
	1



 <p>PROPERTY AND CONFIDENTIAL THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF CHRIS OBER. REPRODUCTION, ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF CHRIS OBER, IS PROHIBITED.</p>		UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ±.015 DECIMAL ±.005 ANGULAR ±.0041° BEND ±1° TWO PLACE DECIMAL ±.005 THREE PLACE DECIMAL ±.003	NAME C.J.K.	DATE 3/22/2011	BALL AND PLATE SYSTEM SYSTEM ASSEMBLY	
		DRAWN CHECKED ENG APPR. MFG APPR. Q.A. COMMENTS:	MATERIAL MIXED FINISH FRESH	NUMBER REQ. 1 DO NOT SCALE DRAWING	SIZE DWG. NO. A Assembly	REV 1
SCALE: 1:1 WEIGHT:			SHEET 1 OF 1			

Appendix B: Initial Wiring Schematic



Appendix C: Bill of Materials

Item #	Part Name	Description	Part #	Num. Req
1	Inner Servo Mount Plate	6063 plate	AL1	2
2	Outer Servo Mount Plate	6063 plate	AL2	1
3	Inner Bar	6063 square tube stock	AL3	4
4	Outer Bar	6063 square tube stock	AL4	4
5	Side Frame Bottom Rail	6063 square tube stock	AL5	2
6	Side Frame Side Rail	6063 square tube stock	AL6	4
7	Side Frame Top Rail	6063 square tube stock	AL7	2
8	Bottom Frame Runner	6063 square tube stock	AL8	2
9	Shaft Mounts	Mild Steel	ST1	4
10	Servo	Turnigy 12V PWM servo	HV-300	2
11	Pillow Block	Mounted needle roller bearing, 0.375" inside diameter	A7Z33-PB375N	2
12	Bronze Bearing	0.375" inside diameter bronze bearing	A7B4-F010	4
13	Sprocket	24 tooth sprocket, #25 chain size, 0.375" bore with set screw	A6C725B24	2
14	Chain	#25 chain size, mild steel, sold by the foot	A6Q725	3
15	Chain Clip	#25 mild steel spring chain clip	A6Q725SCCL	1
16	Shaft	303 Stainless steel shaft, 0.3747" diameter, 3.5" long	A7X112100	4
17	Servo to Shaft Adapter	6061 aluminum, Futaba splines, 0.375" bore	FSA-375	1
18	Servo Hub	6061 aluminum, Futaba splines, tapped	SH503F	1
19	Plate	UHMW polyethylene 2'x2'x0.25"	85705K38	1
20	Camera	iHome 30fps Webcam	IH-W351DW	1
21	Controller	Arduino Nano V2.3	Nano	1

Appendix D: Modified S-FUNCTION

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)
%SFUNTMPL General M-file S-function template
% With M-file S-functions, you can define you own ordinary
differential
% equations (ODEs), discrete system equations, and/or just about
% any type of algorithm to be used within a Simulink block diagram.
%
% The general form of an M-File S-function syntax is:
% [SYS,X0,STR,TS] = SFUNC(T,X,U,FLAG,P1,...,Pn)
%
% What is returned by SFUNC at a given point in time, T, depends on
the
% value of the FLAG, the current state vector, X, and the current
% input vector, U.
%
% FLAG RESULT DESCRIPTION
% -----
% ---
% 0 [SIZES,X0,STR,TS] Initialization, return system sizes in
SYS,
% initial state in X0, state ordering
strings
% in STR, and sample times in TS.
% 1 DX Return continuous state derivatives in
SYS.
% 2 DS Update discrete states SYS = X(n+1)
% 3 Y Return outputs in SYS.
% 4 TNEXT Return next time hit for variable step
sample
% time in SYS.
% 5 Reserved for future (root finding).
% 9 [] Termination, perform any cleanup SYS=[].
%
%
% The state vectors, X and X0 consists of continuous states followed
% by discrete states.
%
% Optional parameters, P1,...,Pn can be provided to the S-function
and
% used during any FLAG operation.
%
% When SFUNC is called with FLAG = 0, the following information
% should be returned:
%
% SYS(1) = Number of continuous states.
% SYS(2) = Number of discrete states.
% SYS(3) = Number of outputs.
% SYS(4) = Number of inputs.
% Any of the first four elements in SYS can be specified
% as -1 indicating that they are dynamically sized. The
% actual length for all other flags will be equal to the
% length of the input, U.
% SYS(5) = Reserved for root finding. Must be zero.
% SYS(6) = Direct feed through flag (1=yes, 0=no). The s-function
% has direct feed through if U is used during the FLAG=3
```

```

% call editing this to 0 is akin to making a promise
that
% U will not be used during FLAG=3. If you break the
promise
% then unpredictable results will occur.
% SYS(7) = Number of sample times. This is the number of rows in
TS.
%
%
% X0 = Initial state conditions or [] if no states.
%
% STR = State ordering strings which is generally specified as
[].
%
% TS = An m-by-2 matrix containing the sample time
% (period, offset) information. Where m = number of
sample
% times. The ordering of the sample times must be:
%
% TS = [0 0, : Continuous sample time.
% 0 1, : Continuous, but fixed in minor
step
% sample time.
% PERIOD OFFSET, : Discrete sample time where
% PERIOD > 0 & OFFSET < PERIOD.
% -2 0]; : Variable step discrete sample
time
% where FLAG=4 is used to get time
of
% next hit.
%
% There can be more than one sample time providing
% they are ordered such that they are monotonically
% increasing. Only the needed sample times should be
% specified in TS. When specifying than one
% sample time, you must check for sample hits explicitly
by
% seeing if
% abs(round((T-OFFSET)/PERIOD) - (T-OFFSET)/PERIOD)
% is within a specified tolerance, generally 1e-8. This
% tolerance is dependent upon your model's sampling times
% and simulation time.
%
% You can also specify that the sample time of the Sfunction
% is inherited from the driving block. For functions
which
% change during minor steps, this is done by
% specifying SYS(7) = 1 and TS = [-1 0]. For functions
which
% are held during minor steps, this is done by specifying
% SYS(7) = 1 and TS = [-1 1].
% Copyright 1990-2002 The MathWorks, Inc.
% $Revision: 1.18 $
% The following outlines the general structure of an S-function.
%
switch flag,
%%%%%%%%%%

```

```

% Initialization %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
case 0,
[sys,x0,str,ts]=mdlInitializeSizes;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Derivatives %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
case 1,
sys=mdlDerivatives(t,x,u);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Update %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
case 2,
sys=mdlUpdate(t,x,u);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Outputs %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
case 3,
tic;
sys=mdlOutputs(t,x,u);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GetTimeOfNextVarHit %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
case 4,
sys=mdlGetTimeOfNextVarHit(t,x,u);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Terminate %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
case 9,
sys=mdlTerminate(t,x,u);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Unexpected flags %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
otherwise
error(['Unhandled flag = ',num2str(flag)]);
end
% end sfuntmpl
%
%=====
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the Sfunction.
%=====
=====
%
function [sys,x0,str,ts]=mdlInitializeSizes
%
% call simsizes for a sizes structure, fill it in and convert it to a
% sizes array.
%
% Note that in this example, the values are hard coded. This is not a
% recommended practice as the characteristics of the block are
% typically
% defined by the S-function parameters.
%
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 0;

```



```

sizes.NumOutputs = 0;
sizes.NumInputs = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1; % at least one sample time is needed
sys = simsizes(sizes);
%
% initialize the initial conditions
%
x0 = [];
%
% str is always an empty matrix
%
str = [];
%global a;
%a=arduino('COM15');
%a.servoAttach(1);
%a.servoAttach(2);
global connected;
connected=false;
fig=gui;
global h;
h = guihandles(fig);
set(h.manual, 'Value', 1);
global runtime;
global oldx;
global oldy;
global k;
k = BMatrixCalc(0.00595, 0.131233596/2 , 0 );
%k = BMatrixCalc(0.1,((1.68/12)/2),1);
%
% initialize the array of sample times
%
ts = [0 0];
% end mdlInitializeSizes
%
%=====
=====
% mdlDerivatives
% Return the derivatives for the continuous states.
%=====
=====
%
function sys=mdlDerivatives(t,x,u)
sys = [];
% end mdlDerivatives
%
%=====
=====
% mdlUpdate
% Handle discrete state updates, sample time hits, and major time step
% requirements.
%=====
=====
%
function sys=mdlUpdate(t,x,u)
sys = [];
% end mdlUpdate

```

```

%
%=====
=====
% mdlOutputs
% Return the block outputs.
%=====
=====
%
function sys=mdlOutputs(t,x,u)
global a;
global h;
global runtime;
global oldx;
global oldy;
global k;
global connected;
global olds1;
global olds2;
%position of ball centroid
i=u(2)%x
j=u(1)%y
set(h.x, 'String', sprintf('%3s:%.2f%', 'X ',i/120) );
set(h.y, 'String', sprintf('%3s:%.2f%', 'Y ',j/120) );
%velocity of ball
Vx=((i-oldx)/runtime)
Vy=((j-oldy)/runtime)
set(h.Vy, 'String', sprintf('%3s:%.2f%', 'Vy ',Vy/120) );
set(h.Vx, 'String', sprintf('%3s:%.2f%', 'Vx ',Vx/120) );
%servo trim
trimy=get(h.trimy,'Value');
trimx=get(h.trimx,'Value');
set(h.showytrim, 'String', sprintf('%3.0f%',trimy));
set(h.showxtrim, 'String', sprintf('%3.0f%',trimx));
%desired position
desiredx=get(h.desiredx,'Value')
desiredy=get(h.desiredy,'Value')
set(h.showdesiredx, 'String', sprintf('%0.2f%',desiredx));
set(h.showdesiredy, 'String', sprintf('%0.2f%',desiredy));
%Plot ball centroid
xmin = -1;
xmax =1 ;
ymin = -1;
ymax = 1;
plot(i/120,j/120,'LineWidth',2,'Marker','.');
axis([xmin xmax ymin ymax]);
set(gca,'XTick',-1:.2:1, 'YTick',-1:.2:1);
grid on;
%Establish connection with Arduino
if get(h.connect, 'Value')
if (connected==false)
a=arduino('COM15');
a.servoAttach(1);
a.servoAttach(2);
connected=true;
end
else
delete(instrfind({'Port'},{'COM15'}));

```

```

connected=false;
set(h.message, 'String','Arduino Disconnected' );
end
%servo control
%s1=pivot about x axis
%s2=pivot about y axis
if get(h.desired, 'Value')
s=BPController([i;j;Vx;Vy], [desiredx;desiredy;0;0], k)
%s1=round((90)-0.32*s(2));
s1=round((90)+s(2));
s2=round((90)+s(1));
set(h.servo1, 'Enable', 'off');
set(h.servo2, 'Enable', 'off');
set(h.manual, 'Value', 0)
else
set(h.servo1, 'Enable', 'on');
set(h.servo2, 'Enable', 'on');
s1=round(180*(get(h.servo1,'Value'))); %other servo
%s1=round(-180*(get(h.servo1,'Value')-1));
%s2=round(-180*(get(h.servo2,'Value')-1)); %HV300 Servo
s2=round(180*(get(h.servo2,'Value'))); %other servo
set(h.manual, 'Value', 1)
end
%video feed control
if get(h.livefeed, 'Value')
set_param('test/LiveFeed', 'Gain', '1');
else
set_param('test/LiveFeed', 'Gain', '0');
end
if get(h.thresholdfeed, 'Value')
set_param('test/ThresholdFeed', 'Gain', '1');
else
set_param('test/ThresholdFeed', 'Gain', '0');
end
%adjust desired angle by trim amount
s1=s1+trimy
s2=s2+trimx
%send angle for servos to Arduino
if (connected==true)
if olds1~=s1
a.servoWrite(1,s1);
end
if olds2~=s2
a.servoWrite(2,s2);
end
set(h.pos1, 'String', sprintf('%3.0f%',s1-trimy));
%set(h.pos2, 'String', sprintf('%4.0f%',-(a.servoRead(2)-
180)+trimx));%HV300
set(h.pos2, 'String', sprintf('%3.0f%',s2-trimx));%other servo
end
%reset run time and store coordinates to memory for velocity
calculation
runtime=toc;
oldx=i;
oldy=j;
%save new values to send to arduino
olds1=s1;

```

```

olds2=s2;
sys = [];
% end mdlOutputs
%
%=====
=====
% mdlGetTimeOfNextVarHit
% Return the time of the next hit for this block. Note that the result
is
% absolute time. Note that this function is only used when you specify
a
% variable discrete-time sample time [-2 0] in the sample time array in
% mdlInitializeSizes.
%=====
=====
%
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1; % Example, set the next hit to be one second later.
sys = t + sampleTime;
% end mdlGetTimeOfNextVarHit
%
%=====
=====
% mdlTerminate
% Perform any end of simulation tasks.
%=====
=====
%
function sys=mdlTerminate(t,x,u)
sys = [];
% end mdlTerminate

```

Appendix E: Modified Guide Generated Code

```
function varargout = gui(varargin)
% GUI M-file for gui.fig
% GUI, by itself, creates a new GUI or raises the existing
% singleton*.
%
% H = GUI returns the handle to a new GUI or the handle to
% the existing singleton*.
%
% GUI('CALLBACK',hObject,eventData,handles,...) calls the local
% function named CALLBACK in GUI.M with the given input arguments.
%
% GUI('Property','Value',...) creates a new GUI or raises the
% existing singleton*. Starting from the left, property value
% pairs are
% applied to the GUI before gui_OpeningFcn gets called. An
% unrecognized property name or invalid value makes property
% application
% stop. All inputs are passed to gui_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
% one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help gui
% Last Modified by GUIDE v2.5 20-Apr-2011 00:34:29
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
'gui_Singleton', gui_Singleton, ...
'gui_OpeningFcn', @gui_OpeningFcn, ...
'gui_OutputFcn', @gui_OutputFcn, ...
'gui_LayoutFcn', [], ...
'gui_Callback', []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end
if nargin
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to gui (see VARARGIN)
% Choose default command line output for gui
set(handles.servo1, 'Value', .5);
set(handles.servo2, 'Value', .5);
handles.output = hObject;
% Update handles structure
```

```

guidata(hObject, handles);
% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in Start.
function Start_Callback(hObject, eventdata, handles)
% hObject handle to Start (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
open('test.mdl');
set_param('test/videosource','ROIWidth','240');
set_param('test/videosource','ROIHeight','240');
set_param('test/Gain','Gain','0');
sim('test.mdl');
% --- Executes on button press in Stop.
function Stop_Callback(hObject, eventdata, handles)
% hObject handle to Stop (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
set_param('test/Gain','Gain','1');
delete(instrfind({'Port'},{'COM15'}));
set(handles.message, 'String', 'Arduino Disconnected');

```

Appendix F: Ball and Plate Gain Controller Code

```

function [ U ] = BPController( X, D, K )
% This function will take in a 2x1 vector of position coordinates for
the
% ball. Then output a motor angle.
% DEVELOPED BY JONATHON BRUCE.
X = X;
K = K;
%if X(1)<=-120 && X(2)>=120
% U = [0,0,0,0];
% return
%end
Ud = K*D;
Up = K*(X/120);
P = Up - Ud;
%P(2) = P(2)*0.75;
U = P;
end

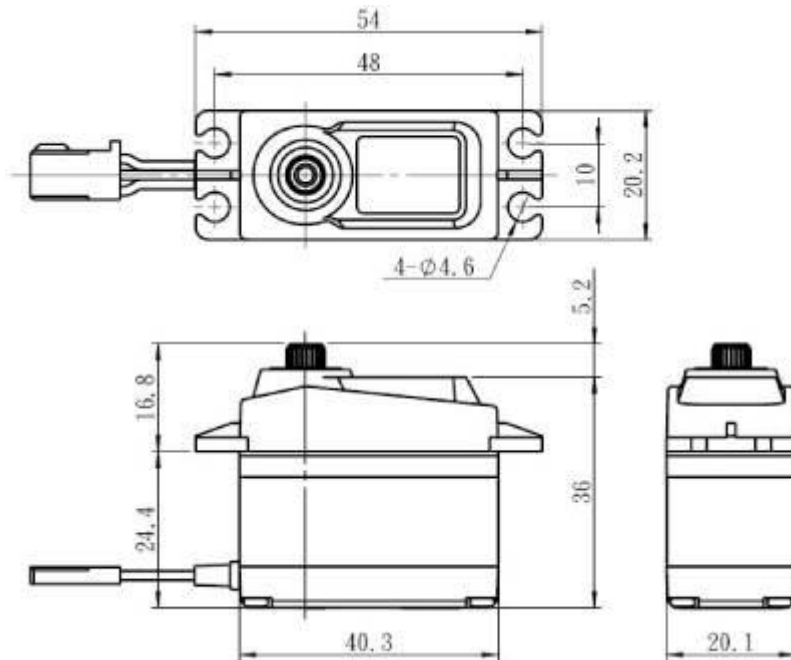
function [ K1 ] = BMatrixCalc( mass, radius, solid )
%UNTITLED3 Summary of this function goes here
% DEVELOPED BY JONATHON BRUCE
M = mass;
S = solid;
R = radius;
g = 32.2;
if S == 1
J = (2/5). *M.*(R^2);
%bx = -(M.*g.*(rx./L))./((J/(R^2))+M);
%by = -(M.*g.*(ry./L))./((J/(R^2))+M);
%return;
%end
elseif S == 0
J = (2/3). *M.*(R^2);
else
fprintf('Please input correct ball type: 1 - Solid, 0 -
Hollow.\n');
return;
end
bx = -(M.*g)./((J/(R^2))+M);
by = -(M.*g)./((J/(R^2))+M);
A = [0,0,1,0;0,0,0,1;0,0,0,0;0,0,0,0];
B = [0,0;0,0;bx,0;0,by];
C = [1,0,0,0;0,1,0,0];
ssTest = ss(A,B,C,0);
P1 = [-9,-9.1,-10,-10.1]*1.1;
K1 = place(ssTest.a,ssTest.b,P1);
%ssTest2 = ss((ssTest.a - (ssTest.b*K1)),ssTest.b,ssTest.c,ssTest.d);
%P = P1*4;
%L = place(ssTest.a',ssTest.c',P);
%Ae = ssTest2.a - (L*ssTest2.c);
%ssTest3 = ss(Ae,ssTest.b,ssTest.c,ssTest.d);
end

```

Appendix G: Final Simulink Model

Appendix H: Savox SC1258TG Specifications

Dimensions(mm):	40.3x20.2x37.2
Weight(g):	52.4
Speed(@4.8V sec/60):	.10
Torque(@4.8V oz-in):	133.3
Speed(@6.0V sec/60):	.08
Torque(@6.0V oz-in):	166.6
Gear:	Titanium & Aluminum
Bearing:	2BB
Case:	Aluminum
Running current (at no load):	100 mA @ 4.8V
Running current (at no load):	120 mA @ 6.0V
Stall current (at locked):	4000 mA @ 4.8V
Stall current (at locked):	5000 mA @ 6.0V
Idle current (at stopped):	5 mA @ 4.8V
Idle current (at stopped):	5 mA @ 6.0V
Limit angle:	200°±10°
Connector wire gauge:	# 22 AWG
Connector wire length:	250 ±5 mm
Horn gear spline:	25T
Control system:	Pulse width modification
Amplifier type:	Digital Controller
Operating Travel:	100° (when1000→2000 μsec)
Neutral position:	1500 μsec
Dead band width:	3 μsec
Rotating direction:	Counterclockwise (when1500→2000 μsec)
Pulse width range:	800→2200 μsec
Maximum travel:	Approx 130°(when900→2100 μsec)



Appendix I: Roboteq AX500 Specifications

Operating Voltage	12V to 24V DC
Number of Channels	2
Max Current Per Channel	
30s	15A
1min	10A
3min	8A
1hr	7.5A
Surge Current	>30A
ON Resistance	100 mOhm
Current Limiting	By automatic power output reduction according to user preset limit and temperature
Temperature protection	Automated current limit reduction starting at 80o C (175o F) heat sink temperature
Voltage protection	Output shut off below 8V
Power Wiring	Terminal strip. AWG 14 max cable
R/C Inputs	2 + 1 accessories (1.0ms - 1.5ms center - 2ms, Adjustable)
Serial Interface	RS232. 9600 bauds
Analog Interface	2 inputs (0V - 2.5V center - 5V)
Input Corrections	Ch1 & Ch2 mixing for tank steering. Programmable deadband. 4 Exponent & Logarithmic command curves.
Optical Encoder Inputs	No
Analog Inputs	4 inputs, 8-bit resolution
Digital Outputs	1 output, 24V 100mA max.
Digital Inputs	3 general purpose inputs
Open Loop Speed	Forward & Reverse Speed Control. Separate or Mixed
Closed Loop Speed	Use Tachometer on analog inputs & PID
Position Mode	Use Potentiometer on analog inputs & PID
Controller Configuration	Jumper-less using PC utility
Operating Temperature	-40 to +85oC heat sink temperature
Enclosure	Unenclosed, board level
Controller size	4.2" (106mm) wide x 2.9" (50mm) long x 1.5" tall (38mm) tall including heat sink
Supplied Cables	4' (1m) RS232 cable for PC connection. 10" (25cm) RC cable to Radio.
Weight	3 oz (85g)