Spring 2018

Ball Oscillating Bouncer

Eric Blok
ejb65@zips.uakron.edu

Daniel Altemese
dpa14@zips.uakron.edu

Ryan Nowacki
rmn24@zips.uakron.edu

Maram Qurban
mmq6@zips.uakron.edu

Recommended Citation

Blok, Eric; Altemese, Daniel; Nowacki, Ryan; and Qurban, Maram, "Ball Oscillating Bouncer" (2018). *Honors Research Projects*. 732.
http://ideaexchange.uakron.edu/honors_research_projects/732

Ball Oscillating Bouncer

Project Design Report

DT06

Daniel Altemese

Eric Blok

Ryan Nowacki

Maram Qurban

Faculty Advisor: Dr. Veillette

Date Submitted: 4/27/18

Table of Contents

Table of Figures

List of Tables

Abstract

The purpose of this report is to document the need, objectives, marketing and engineering requirements, as well as validate the design of an autonomous control device capable of continuously bouncing a table tennis ball on a paddle. This includes the design of a self-correcting system using lightweight materials, and as few sensors and components as possible to achieve a compact, portable design. To accomplish this, the system is designed to react to a ball falling from as short a distance as 10 centimeters above the paddle, meaning all sensor processing, control processing, and motor drives should be able to react within an appropriate timeframe. The overall system is broken down into four main parts: a sensor and sensor processing system, a controller and control processing system, an electromechanical motor system, and a DC power supply system. The sensor system shall be capable of detecting and analyzing ball trajectory and forward that information to the control processor. The control processor will generate a response to react to the ball trajectory and forward that information to the motor drives which will physically act to correct the ball position, thus completing the control sequence. (DA)

- System shall behave completely autonomously
- Four main subsystems: Sensor/Processor, Controller/Processor, Motors and Drives, and DC Power Supply.
- System will need to react quickly, sometimes in as short a time as 142 milliseconds.

1.0 Problem Statement

1.1 Need

The need for today's machinery to be able to work autonomously with precision and speed has increased in a society where quickness is everything. Such equipment would require sensors, various inputs and the ability to understand how to interpret the data obtained. As electrical and computer engineers, it will be up to them to be able to design the systems needed to further advance human society. However, everyone must start somewhere and the Ball Oscillator Bouncer will provide the basis for implementing the knowledge needed to design future systems. (EB)

1.2 Objective

The goal of this project is to design, construct, test, and verify a device that is capable of determining the trajectory of a table tennis ball by using an array of sensors. The device should be capable of determining the landing path of the ball and how to strike the ball in such a manner that the ball's final destination will be in the middle of the device. (RN)

1.3 Background

Various methods of sensors can be used in order to track any form of object. In order to better determine the most appropriate method for this project, many methods have been explored. The following information is to report the findings in the documents provided in the references. (RN)

1.3.1 Patent Search

Upon searching patents relevant to this project, patent US 5515853, which was issued in 1995, proved to be useful in this application. Patent US 5515853 is the use of a three-dimensional digital ultrasound tracking system. The patent proceeds to address the use of piezoelectric transducers to measure distances. In order to produce a three-dimensional tracking system, the system alternates duty cycles and attempts to triangulate by using three transducers which enables the system to accurately depict a three-dimensional image. [4] This is useful to the project as the use of piezoelectric sensors would be very practical in the manner of detecting an object moving through a medium. In this project the object moving would pertain to the table tennis ball, moving through the air. The piezoelectric sensors could be divided into sections on the board and then processed in such a manner that would allow us to, in a sense, render a three-dimensional image as mentioned in [4]. This may be one of many ways to detect position, and velocity, which could be used to determine potential trajectory paths, and be used for predictive analysis for future events of the system. (RN)

The patent, US 9557405, was issued January 31, 2017. The inventor details the ability to determine a projectile's trajectory with at least two sensors by having the first and second sensor estimate the trajectory with a first and second angular range, respectively [5]. A discrepancy is developed between the two sensors and the actual projectile trajectory is determined using the estimated discrepancy. The method described could provide a basis for gaining knowledge towards developing the group's own method for tracking a physical projectile. (EB)

The patent US 6059672 was published in May 9, 2000. It is about "Golf Training Device Particularly for the short game". One of the most important elements were used in order to build the device is a microprocessor to process the collected data, which are distance measuring by triangulation, grass depth and humidity measurement. [7] Those data are then displayed into miniaturized LCD [7]. The idea of using microprocessor and LCD is going to help in displaying the data needed to calculate the trajectory of the table tennis ball. (MQ)

1.3.2 IEEE Xplore Article Search

The article "Towards Table Tennis with a Quadrotor Autonomous Learning Robot and Onboard Vision" discusses the idea of designing a quadrotor drone capable of playing against a human opponent in a game of table tennis with at least a moderate degree of competency. The quadrotor would employ an onboard camera to estimate the position and velocity of the ball in 3D space, by comparing a series of images provided by the camera to its own reference frame. The report goes on to detail a series of equations that are used to determine the ball's position and velocity in order to generate a response. [1] While the idea proposed in [1] differs from the proposal presented in this document, the overall concepts of perception, estimation, and control are quite relevant here as well. In particular, the use of a similar camera sensor could be modeled and implemented into this project, as well as some elements of the controller used in [1]. (DA)

The article “A Control System of a Ping-Pong Robot Arm Based on Fuzzy Method” proposes an idea for low-cost control system for a ping-pong robot’s arm in order to ensure quick and accurate responses for the arm. This system uses an infrared touch screen in order to detect position of the ball when contact is made, which is more cost effective in comparison to a camera-based sensor system. The article then goes on to detail a fuzzy control scheme for the design, which was then implemented in order to balance the pong ball in practice. [2] Again, the idea implemented in [2] is similar to that of this project. As such, the implementation of fuzzy control design could prove useful in this project as well, as it would provide a more precise control to the system proposed here, although it may prove to be unnecessary. Additionally, the use of an infrared sensor system seems appropriate here, as it would be a cost-efficient way to monitor position and velocity of the ball. (DA)

The article “Ping-Pong Trajectory Perception and Prediction by a PC based High Speed Four-Camera Vision System” is a system that uses a high-speed vision system in order to track the velocity and trajectory of a ping-pong ball in three dimensions. The system is proposing the use of a ‘powerful industrial PC’ and four 1394 firewire cameras. [3] The information aforementioned in [3] would be useful to this project in that it would provide a means of calculating the trajectory and the final landing position of the ping-pong ball, as needed to control the balls bouncing motion. Once the final landing position is determined the system would then have to counteract to the position in such a manner to keep the ball from bouncing beyond the platforms reach. The final goal is to stabilize the ball in the middle of the platform for an indefinite period of time. (RN)

The article “Dual Axis Operation of a Micromachined Rate Gyroscope” detailed the procedure necessary to successfully manage dual axis operation with a micromachined rate gyroscope. The key, as stated in the article, to a successful simultaneous dual axis operation is the quad symmetry of the circular oscillating rotor design of the gyroscope [6]. The gyroscope mentioned in [6] would be useful as additional feedback to the controller to map the position of the “paddle”. The feedback from the gyroscope could then be used to adjust the paddle accordingly based on input of the projectile’s path of motion (obtained from other sensors). With the paddle set in the correct position, any contact with the projectile should work to correct the projectile’s angle of deviation from the origin. After contact, the process is set to repeat until the designated end sequence has been reached. (EB)

The article “Jointly learning trajectory generation and hitting point prediction in robot table tennis” explains the use of a robotic arm that is designed to hit a table tennis ball back towards the sender autonomously. The article goes on to explain that not only will the robotic arm find the desired hitting point, but the robotic arm will also define a robotic trajectory generation map which would allow the algorithms to define a racket trajectory in Cartesian coordinates. [8] The use of this technology could provide insight as a means to detect the trajectory path of the table tennis ball. Furthermore, the Ball Oscillating Bouncer will need to detect a hitting point for the table tennis ball. Thus, using the approach given in [8] will allow the Ball Oscillating Bouncer to react at the precise moment in order to hit the ball at the correct angle and speed, and with the correct amount of force. (RN)

1.4 Marketing Requirements

The following are descriptions of the terminology that will be used in the following marketing requirements. The marketing requirements are the requirements which must be met and satisfied by this proposed document. The terminology is as follows:

- The term “device” refers to the system that is going to be exclusively responsible for bouncing the ball. The device may also be referred to as Ball Oscillating Bouncer, or better known as B.O.B.
- The term “ball” refers to any object which will be directed towards the device. Specifically, in the terms of this document, a simple regulation-size table tennis ball will be used.
- The term “sensors” refer to any type of device that will be used to monitor speed, direction, or position, and then feed the information that is gathered back to the system in order for the system to respond in the correct manner.

The marketing requirements for the project are as follows:

1. The device shall be able to detect and react to an incoming ball dropped from above the paddle and bounce it continuously with reasonable accuracy.
2. The device shall be light and easy to transport.
3. B.O.B. shall serve to be a “desk prop”, similar to devices like “Newton’s Cradle” for simple entertainment purposes.
4. The device shall have a basic on/off switch which can be used for as an emergency override if necessary.
5. The device shall plug into the wall for power supply.
6. B.O.B. shall serve to showcase the power of human creativity and ingenuity.
7. The device shall revert to standby-mode automatically if the ball is removed from the system.
8. The device shall be able to work efficiently at room temperature.

1.5 Objective Tree

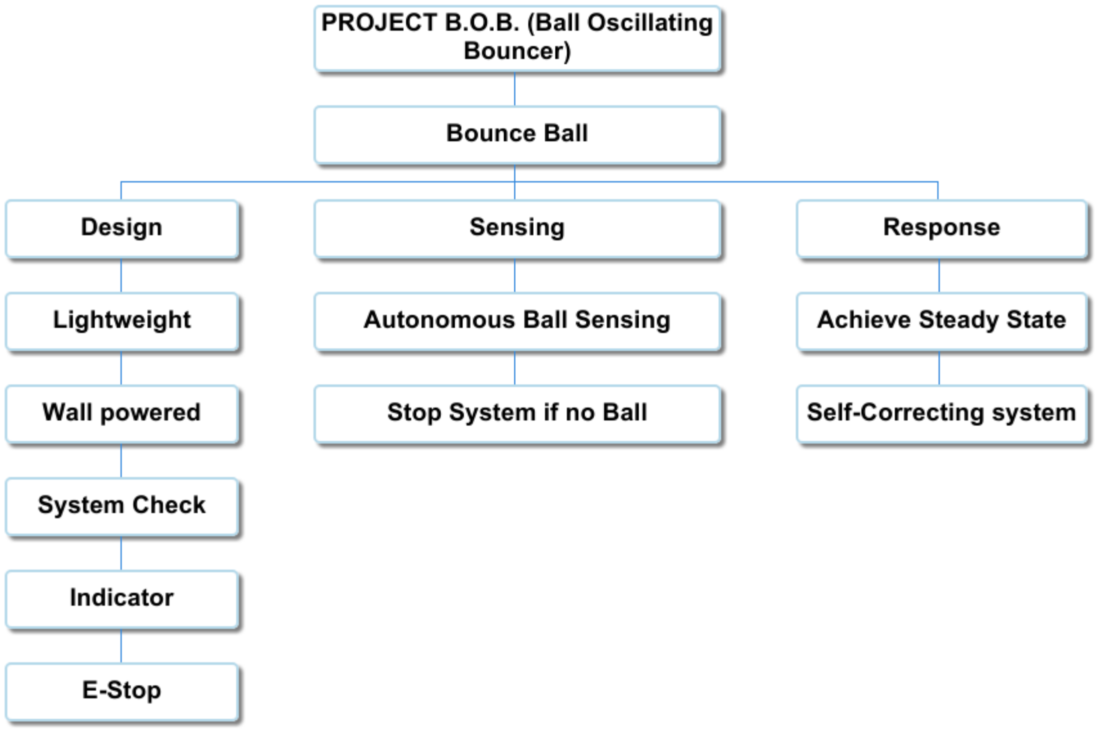


Figure 1: Objective Tree

2.0 Design Requirements Specification

Table 1: Design Requirements

Marketing Requirements	Design Requirement	Justification
1	The system shall keep the ball bouncing at a minimum height of 15 centimeters once it has been stabilized.	15 centimeters is a suitable height to demonstrate system, while minimizing the effect of outside disturbances.
2,3	The paddle of the system shall be no bigger than 50 cm wide by 50 cm square long.	The device must be designed to fit easily on a desk and must be reasonably easy to carry by hand.
1	The system shall be designed to bounce a regulation 2.7g table tennis ball.	All design and calculations are done with respect to a regulation table tennis ball.
1,3,6	The system shall be able to continue bouncing the ball for at least one minute.	One minute should be enough time to properly demonstrate an effective control system before disturbances can disrupt the system.
1	The system shall be capable of responding to a ball first dropped within a range of 10 cm to 50 cm.	The sensor system should be placed so that it can effectively detect the ball in this range. This also gives the overall system approximately 142 milliseconds to respond completely.
1	The paddle of the system shall be capable of tilting along two axes of rotation.	In order to keep the ball bouncing above the paddle, the paddle must be able to tilt in at least two dimensions for control.
1,7	The sensor system shall be capable of detecting all three dimensions of ball position.	The system will need to know all three dimensions in order to determine when to strike the ball, and at what angle.
1	The motor system must be capable of delivering at least 1 Ncm of holding torque.	Motors must be strong enough to hold the paddle in place and produce enough upward force to strike effectively.
1	The motor system shall be able to deliver a linear paddle velocity of at least 5 cm/s.	In order to counteract the coefficient of restitution of the bouncing ball, allowing it to continue bouncing at the desired height.
8	The motor system shall be designed to run continuously within a temperature range of 0^oC and 50^oC.	The motors and motor drivers must be properly cooled for room temperature conditions to avoid burning out the devices

3.0 Accepted Technical Design

3.1 Hardware Theory of Operation

The hardware is designed so that there is an effective line of communication between the sensors and the paddle actuators. This is done by utilizing two microcontrollers: one to read and analyze the sensor data, and one to generate the appropriate actuator response. The microcontroller designated for analyzing the sensor data calculates the predicted x and y coordinates of the table tennis ball's next bounce position, as well as the x and y velocity components of the ball. The maximum height is also calculated to determine the appropriate timing for paddle actuation. These five values will be sent to the actuator microcontroller, which calculates an appropriate response and sends it to the motor drivers, from which the final power control will be sent to the stepper motors. (EB)

3.2 Hardware Block Diagrams

Figures 2 through 7 show the system as it is broken down in increasing detail from the initial level 0 block diagram, to the final level 3 block diagram. The figures are accompanied by tables 2 through 14 to describe the various blocks shown. By viewing the figures and reading the tables, it should be possible to visualize how the hardware theory of operation will be implemented. (EB)

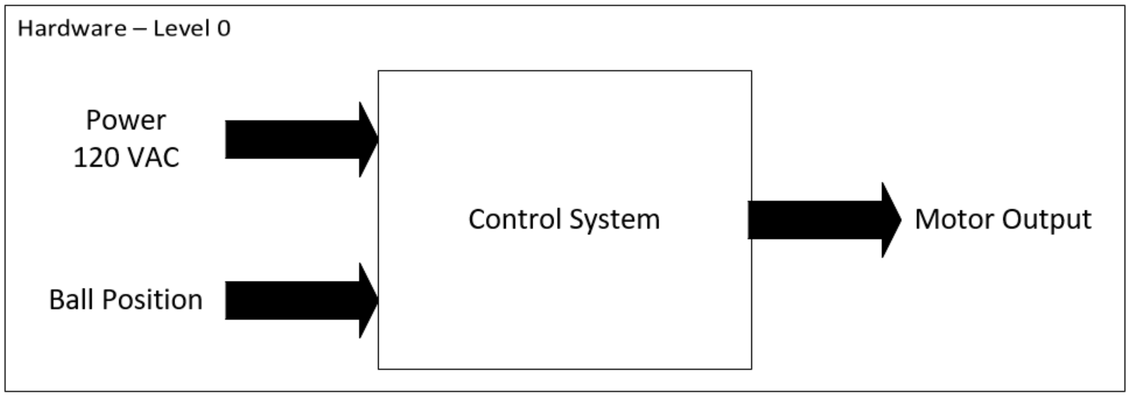


Figure 2: Level 0 Hardware Block Diagram

Table 2: Level 0 Hardware Functional Requirement Table

Module	Control System
Designer	Altemese, Blok
Inputs	Ball Position, 120 VAC Wall Power Supply
Outputs	Motor Control Signal (Motor Output)
Function	Obtain input from sensors relaying position of the ball to the processor; position is calculated and corrections are sent to the PID Controller to provide the proper response to the motors to return the ball to origin.

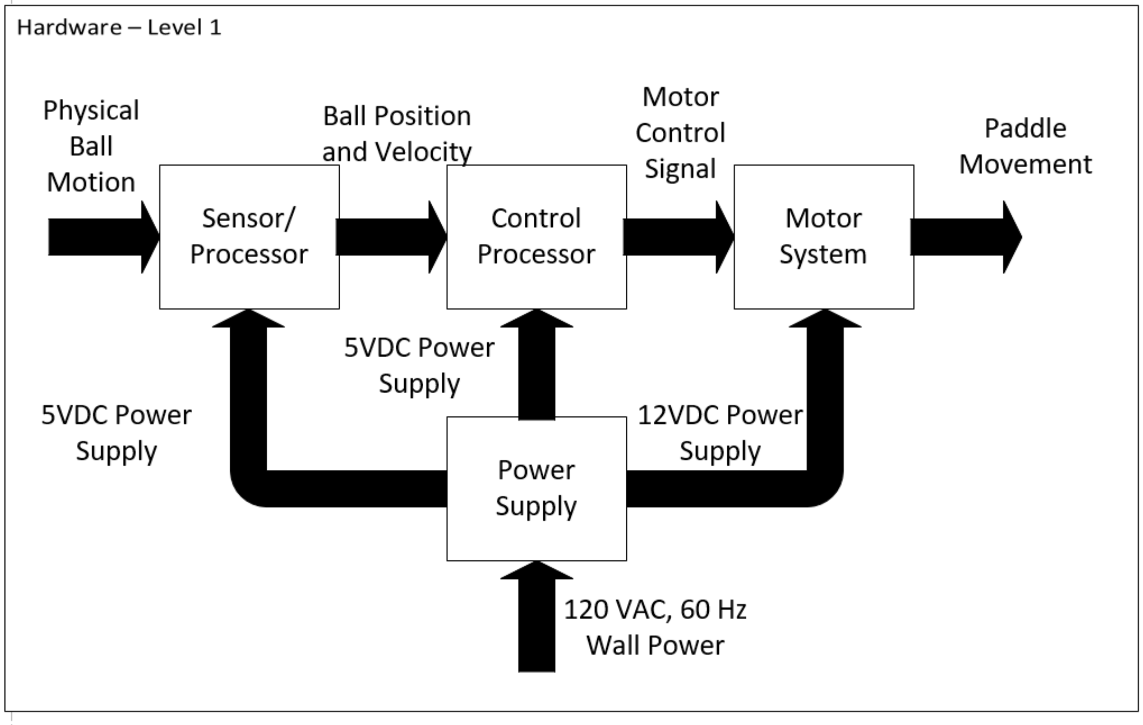


Figure 3: Level 1 Hardware Block Diagram

Table 3: Level 1 Sensor/Processor Functional Requirement Table

Module	Sensor/Processor
Designer	Altemese, Blok, Nowacki, Qurban
Inputs	Physical Ball Position & Velocity, 5 VDC Voltage Supply
Outputs	Ball Position and Velocity
Function	Process sensor information signal and calculate ball’s horizontal velocity and future position.

Table 4: Level 1 Controller/Processor Functional Requirement Table

Module	Control Processor
Designer	Altemese, Blok
Inputs	Ball Position & Velocity, 5V DC Power Supply
Outputs	Motor Control Signal
Function	Produce the proper motor control signal, based on “instructions” sent from the processor to strike the ball, and return the ball back to the center of the paddle.

Table 5: Level 1 Motor System Functional Requirement Table

Module	Motor System
Designer	Altemese, Blok
Inputs	Motor Control Signal, DC Voltage
Outputs	Paddle Movement
Function	Motor Drive receives motor control signal from controller and actuates the motors accordingly.

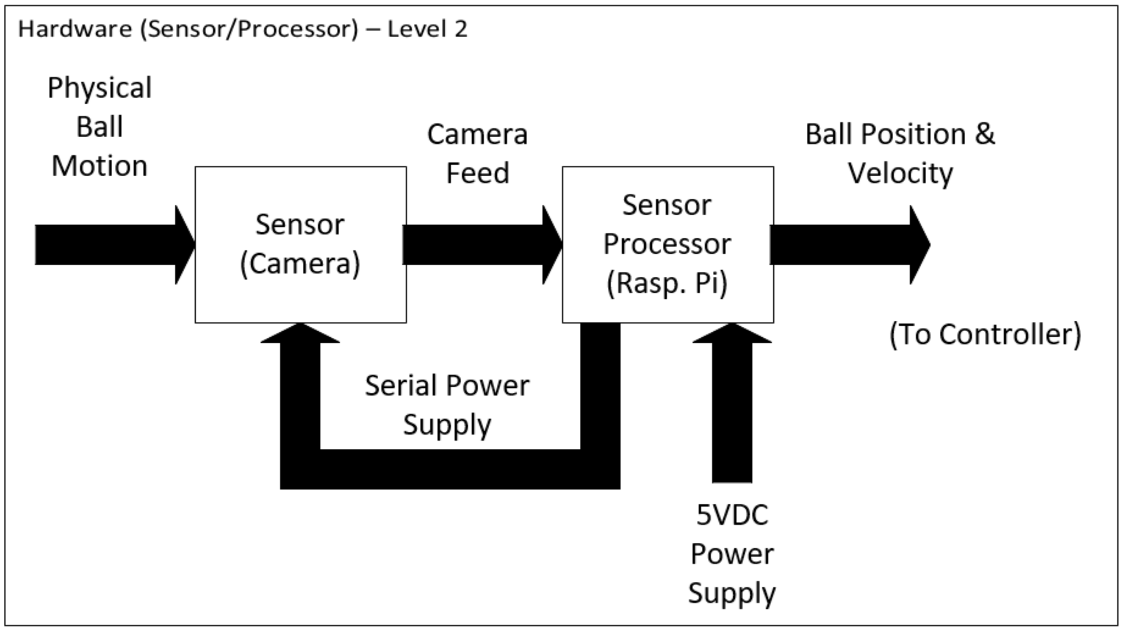


Figure 4: Level 2 Sensor/Processor Hardware Block Diagram

Table 6: Level 2 Sensor Functional Requirement Table

Module	Sensor
Designer	Manufacturer
Inputs	Physical Ball Motion, Serial Power Supply
Outputs	Camera Feed
Function	Obtain ball position data in real-time and forward data signal to processor.

Table 7: Level 2 Sensor Processor Functional Requirement Table

Module	Sensor Processor
Designer	Manufacturer
Inputs	Camera Feed, 5V DC Power Supply
Outputs	Ball Position & Velocity
Function	Receives real-time ball information and calculates ball trajectory; sends output signal to controller.

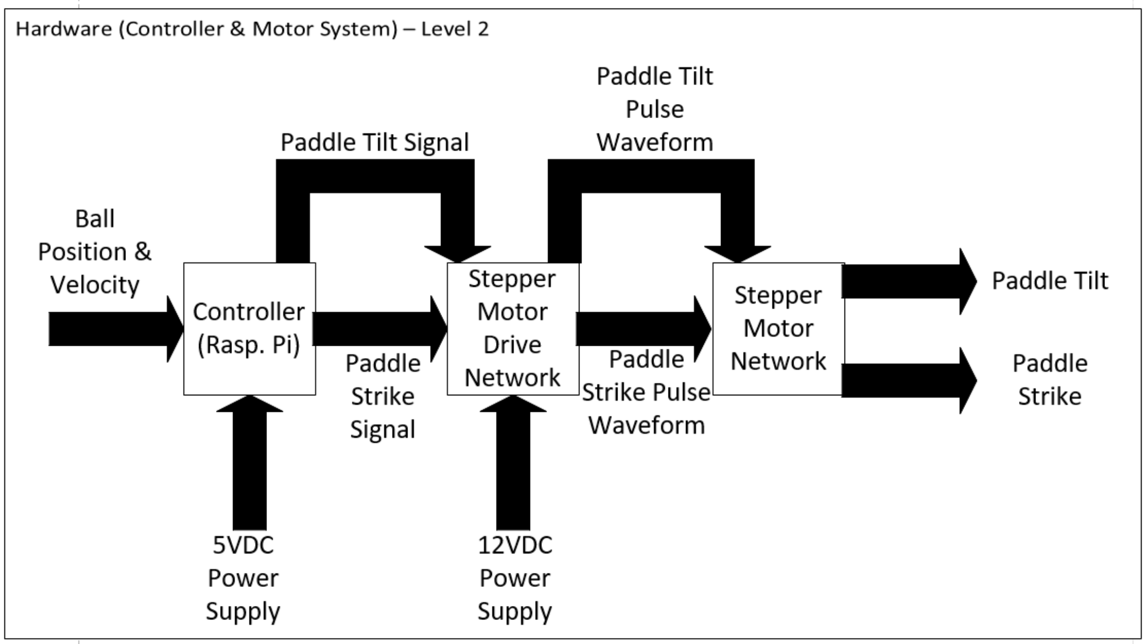


Figure 5: Level 2 Motor System Hardware Block Diagram

Table 8: Level 2 Controller Functional Requirement Table

Module	Controller (Raspberry Pi)
Designer	Manufacturer
Inputs	Ball Position & Velocity, 5V DC Power Supply
Outputs	X-axis tilt angle, y-axis tilt angle, z-axis strike time
Function	Using information from sensor/processor, determine appropriate paddle response. Send appropriate signals to motor drives to generate paddle response.

Table 9: Level 2 Stepper Motor Driver Network Functional Requirement Table

Module	Stepper Motor Driver Network
Designer	Altemese, Blok
Inputs	X-axis tilt angle, y-axis tilt angle, z-axis strike time, 12V DC Power Supply
Outputs	X-axis tilt waveform, y-axis tilt waveform, z-axis strike waveform
Function	Generate the necessary pulse signals to drive the motors based on controller output. These signals must tilt the paddle along two axes of rotation and drive the motors to strike at the appropriate time and velocity.

Table 10: Level 2 Stepper Motor Driver Network Functional Requirement Table

Module	Stepper Motor Network
Designer	Altemese, Blok
Inputs	X-axis tilt waveform, y-axis tilt waveform, z-axis strike waveform
Outputs	X-axis paddle tilt (cm), y-axis paddle tilt (cm), z-axis paddle strike (cm/s)
Function	Convert pulse signal waveforms into physical paddle tilt and motion through motor actuation.

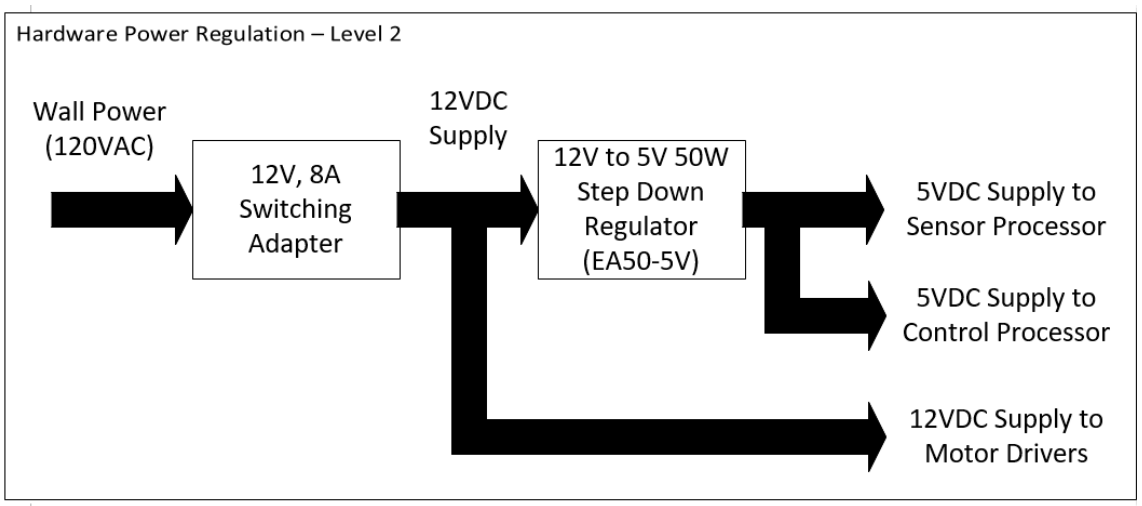

Hardware Power Regulation – Level 2
Wall Power (120VAC)
12V, 8A Switching Adapter
12VDC Supply
12V to 5V 50W Step Down Regulator (EA50-5V)
5VDC Supply to Sensor Processor
5VDC Supply to Control Processor
12VDC Supply to Motor Drivers

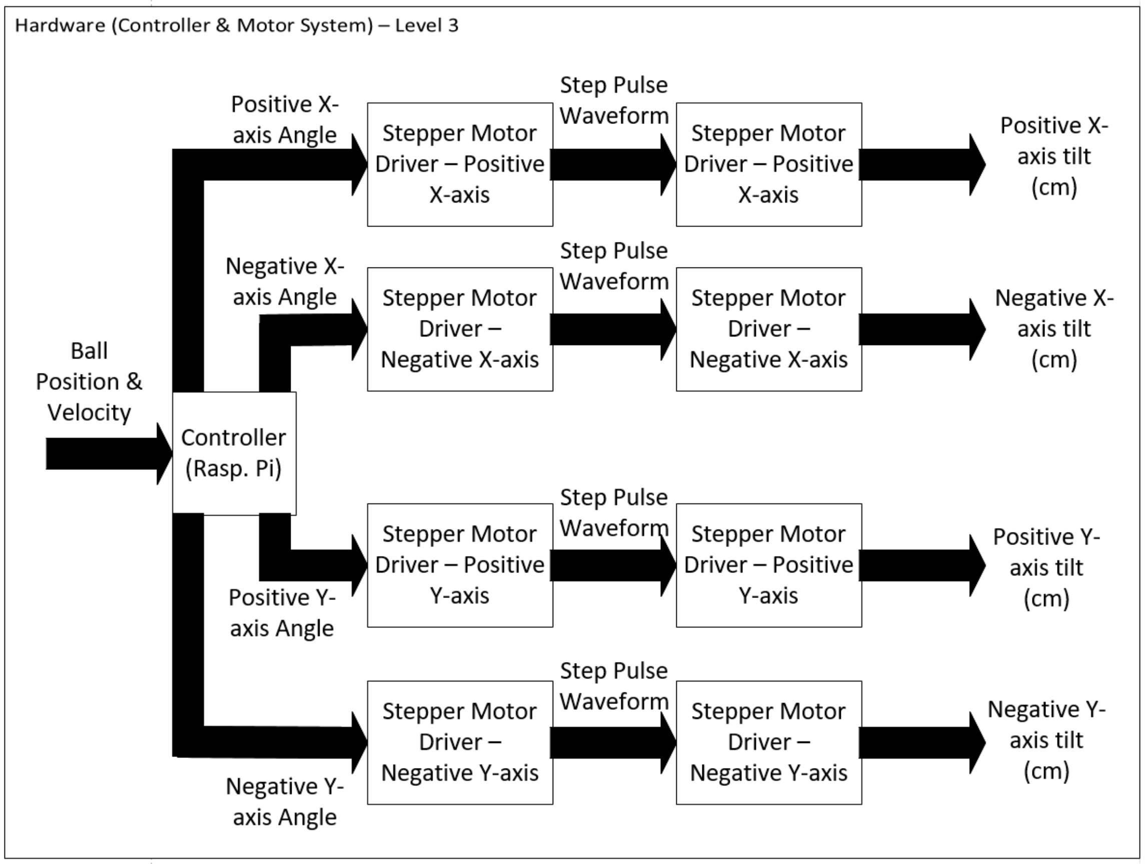
Figure 6: Level 2 Power Regulation Block Diagram

Table 11: Level 2 Switching Adapter Functional Requirement Table

Module	12V, 8A Switching Adapter
Designer	Altemese, Blok
Inputs	Wall Power (120 VAC, 60 Hz)
Outputs	12 VDC Power
Function	Convert 120 VAC power into 12 VDC power for stepper motor driver input power supply.

Table 12: Level 2 Step-Down Voltage Regulator Functional Requirement Table

Module	12V to 5V, 50W Step-Down Regulator (EA50-5V)
Designer	Altemese, Blok
Inputs	12 VDC Power
Outputs	5 VDC Power
Function	Transform 12 VDC power down to 5 VDC power for Raspberry Pi input power supply.



Hardware (Controller & Motor System) – Level 3
Ball Position & Velocity
Controller (Rasp. Pi)
Positive X-axis Angle
Negative X-axis Angle
Positive Y-axis Angle
Negative Y-axis Angle
Stepper Motor Driver – Positive X-axis
Stepper Motor Driver – Negative X-axis
Stepper Motor Driver – Positive Y-axis
Stepper Motor Driver – Negative Y-axis
Step Pulse Waveform
Stepper Motor Driver – Positive X-axis
Stepper Motor Driver – Negative X-axis
Stepper Motor Driver – Positive Y-axis
Stepper Motor Driver – Negative Y-axis
Positive X-axis tilt (cm)
Negative X-axis tilt (cm)
Positive Y-axis tilt (cm)
Negative Y-axis tilt (cm)

Figure 7: Level 3 Motor System Hardware Block Diagram

Table 13: Level 3 Stepper Motor Drivers Functional Requirement Table

Module	Stepper Motor Drivers
Designer	Altemese, Blok
Inputs	X-axis tilt angle, y-axis tilt angle, z-axis strike time
Outputs	X-axis tilt waveform, y-axis tilt waveform, z-axis strike waveform
Function	Generate the necessary pulse signals to drive the motors based on controller output. These signals must tilt the paddle along two axes of rotation, with two drivers representing each axis of rotation. A signal would be sent to all four drivers for a paddle strike on the ball.

Table 14: Level 3 Stepper Motors Functional Requirement Table

Module	Stepper Motors
Designer	Altemese, Blok
Inputs	Step Pulse Waveforms
Outputs	X-axis paddle tilt, y-axis paddle tilt, z-axis paddle strike
Function	Motors shall be placed to be capable of tilting the paddle along two axes of rotation, each motor representing one direction along its respective axis. All four motors shall be actuated together in order to strike the ball.

3.3 Software Theory of Operation

The theory of this software is to take an input from an attached camera and import the data into a microcontroller. The microcontroller will then analyze the image by applying a HSV filter to filter out the undesired colors. Once the image has been filtered, an erode and dilate mask will then be applied to the image, and a midpoint will be taken to track the ball. The ball midpoint position will be stored in an array that will be used to compute the final landing position of the ball on the paddle. Therefore, all trajectory motion will be taken on the upswing of the ball. Once the landing position has been calculated, these values will be sent to the paddle controller to determine the correct fashion in which the paddle must be moved in order to deflect the ball. This will be done by taking the position and converting these values to voltages to instruct the motors on how to reorient the paddle, and when to strike. This process will be recursive thus allowing the ball to be returned to the center of the paddle and attempt to maintain this position with minimal corrections needed. (RN)

3.4 Software Block Diagrams

The software components of this report will be using a camera signal that will be sent to a microcontroller via CSI. This data will contain the ball's' position information such as current X, Y and Z-position, as well as determine the future coordinates in the X, and Z plane. These future ball coordinates will then be sent to the paddle controller to calculate the amount the paddle needs to reorient in order to deflect the ball to the middle of the paddle. These values will be then sent to the correct actuators and the paddle adjusted as seen in Figures 8 through 10. (RN)

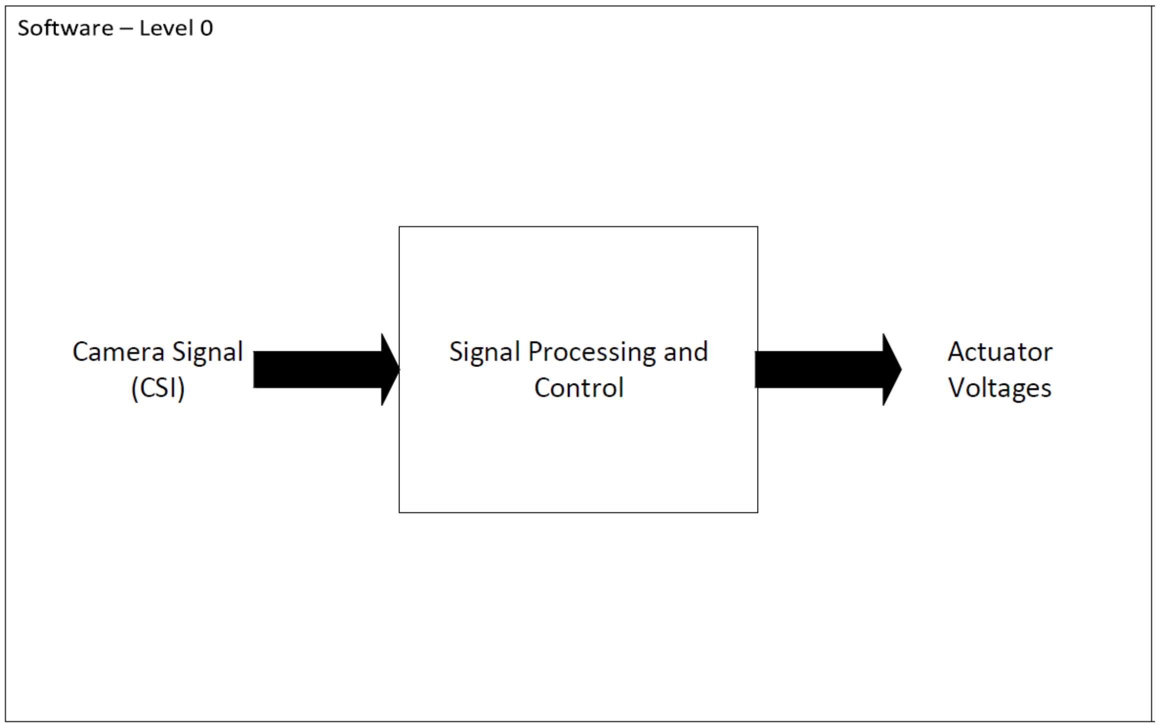


Figure 8: Level 0 Software Block Diagram

Table 15: Level 0 Signal Processing and Control Functional Requirement Table

Module	Signal Processing and Control
Designer	Nowacki, Qurban
Inputs	Camera Signal
Outputs	Actuator Pulse Signals
Function	The position values of the ball on a frame to frame bases will be analyzed and the resulting output will be the future ball coordinates that will be used to drive the actuators.

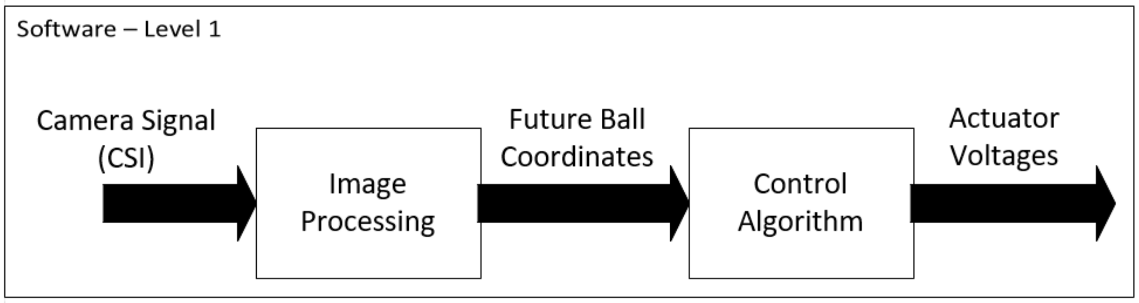


Figure 9: Level 1 Software Block Diagram

Table 16: Level 1 Image Processing Functional Requirement Table

Module	Image Processing
Designer	Nowacki, Qurban
Inputs	Camera Signal
Outputs	Future Ball Coordinates
Function	Analyze the frame by frame instances, to compute the midpoint of the ball. Once enough data has been taken, a future position coordinate will be pushed to the control algorithm to determine how to return the ball to the center of the paddle.

Table 17: Level 1 Control Algorithm Functional Requirement Table

Module	Control Algorithm
Designer	Nowacki, Qurban
Inputs	Future Ball Coordinates
Outputs	Actuator Voltages
Function	Using the future ball coordinates, determine the appropriate angle the paddle must be in order to deflect the ball in such a manner as to return it to the center of the paddle.

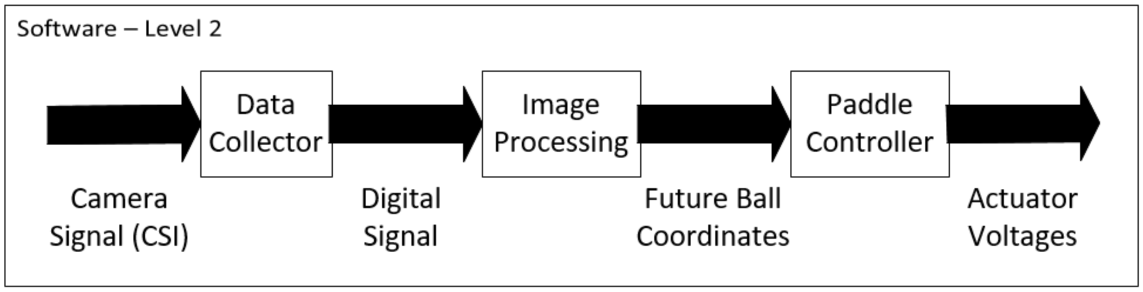


Figure 10: Level 2 Software Block Diagram

Table 18: Level 2 Data Collector Functional Requirement Table

Module	Data Collector
Designer	Nowacki, Qurban
Inputs	Camera Signal
Outputs	Digital Signal
Function	The data from the camera will be polled at a specific rate based on the frame rate of the camera. The information will then be converted to some usable format for the CPU to process and compute. The transmission will be a digital signal.

Table 19: Level 2 Image Processing Functional Requirement Table

Module	Image Processing
Designer	Nowacki, Qurban
Inputs	Digital Signal
Outputs	Future Ball Coordinates
Function	Using the data that was collected by the CPU, an image will be converted to HSV color scale, dilated, and erode to remove distorted pixels. The image will then be analyzed using moments to compute the midpoint of the ball and store in an array. Once the ball is on the downswing, a future ball coordinates will be generated.

Table 20: Level 2 Image Processing Functional Requirement Table

Module	Paddle Controller
Designer	Altemese, Blok, Nowacki, Qurban
Inputs	Future Ball Coordinates
Outputs	Actuator Voltages
Function	Using the future ball coordinates, a transfer function to determine the corrective action needed to be taken by the paddle will be computed and converted to voltages that will be sent to the actuators to handle accordingly.

3.5 Communication

The sensor processor raspberry pi and the motor controller raspberry pi shall communicate using the dedicated UART pins located on the raspberry pi 3. The baud rate shall be set to 512 kHz to allow for quick transfer of the ball's x, y, and z coordinates from the image processor to the controller. This will allow the controller to have adequate time to calculate the appropriate motor response in order for the paddle to correctly impact the ball at the right time. (EB)

3.6 Image Processing Flowchart

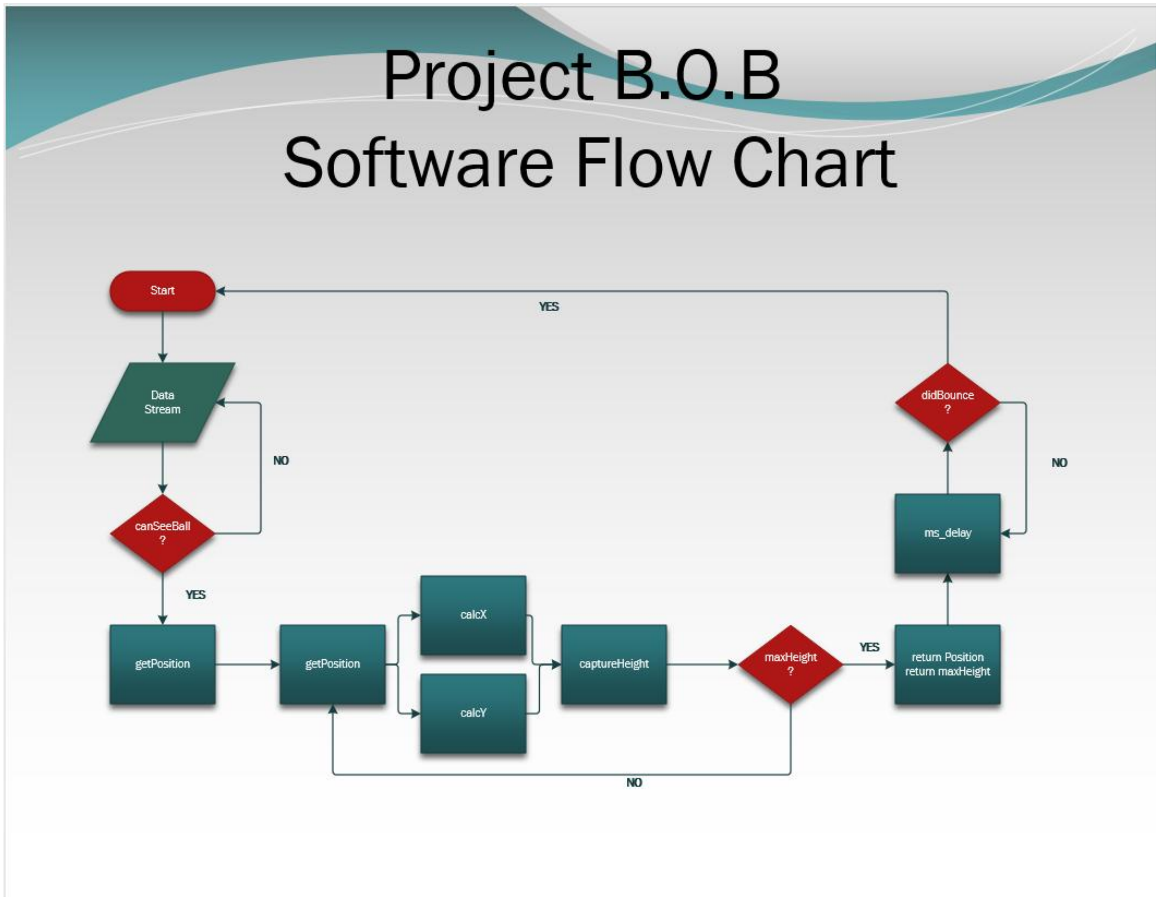


Figure 11: Software Pseudo Flowchart

The proposed flowchart for the software section is as displayed in Figure 11. When the program starts, data collection from the camera will begin to poll the information until the appropriate object is detected. Once the object has been recognized, the position will then be calculated by using a frame by frame comparison. (RN)

Using the positions that are gathered, an approximation of the X and Y coordinates in relation to the paddle will then be calculated, and the maximum height recorded. If the maximum height has not been achieved, then a recursion will occur, until the height is less than the previous height recorded. (RN)

Once the ball is in the downward swing from peak height, the pre-calculated position, and maximum height will be sent to the control algorithm to calculate the actuator motion needed to counteract the ball. (RN)

3.7 Pseudo Code

The following pseudo code shows the main loop and some different methods of the ball tracking, detection, center find in order to send the appropriate position to the controller. (MQ)

```
function main()

#setup camera interface

frame[640, 480] #set up a buffer to receive frames
posx [ ] #set up a variable to store x position information
posy [ ] #set up variable to store y position information
radius[ ] #set up a variable to store the radius information
Cnt[16] #set up a variable to store contour information
color_min[3] #define minimum color of object for detection
color_max[3] #define maximum color of object for detection
min_radius = 10 #define the minimum radius for detection

while(1)
    grab frame ()
    In_range() #mask frame to detect color, openCV function in_range()
    erode() #perform erosion in frame, #openCV function erode()
    dilate () #perform dilation, openCV function dilate()
    cnt[i] = find_contours ()
    If num_cnt > 1
        max_cnt = max_size(cnt)
        x, y, radius = find center(max_cnt)
        If radius > min_radius
            posx[i] = x
```

```
                    posy[i] = y
                    radius[i] = radius

                        #bounce detection algorithm, detect change in direction
                        If ( posy[i] >posy[i-1]) & (posy[i-1] < posy[i-2])
                            bounce ++ #increment the number of bounces

                        #send posx,posy, information to motor controller
                        send(posx)
                        send(posy)
                    end if

                end if

            end while

        end function
```

This pseudo code represents the main loop of the system and the methods are used in the project to detect and track the ball in order to find its center; then use the center to send the appropriate position to the controller. After defining and setting up all the necessary variables, a while loop started to grab frames from the camera. Then, in_range(), dilate(), and erode(), which are some built-in functions of openCV, are used to mask the ball and remove any small blobs left in the mask. Then, an if function is used and starts when at least one contour is found by looking at the radius of the ball to make sure it is close enough to the camera as desired. The contour is used to focus only on the largest colored object it looks for to avoid confusing two or more objects if found in the frame. After that, the center of the ball is determined and a bounce detection is performed to detect the change in direction to count the number of bounces and increase it. The final step then is to send the appropriate position of the ball the controller. (MQ)

3.8 MATLAB Simulation of Future Position Algorithm

Due to the physical complexities of a bouncing ball, a few assumptions must be made. The first assumption is that the air resistance on the ball will be negligible. The air resistance is negligible due to the natural weight of the ball. In the minimum amount of free fall, the balls weight will not be significant enough to allow a drag force to be exerted on the ball; Therefore, the ball will not be influenced by any outside air forces. This will allow for the algorithms to negate any air resistance that may be accrued by the ball. For reference, the force of air resistance on a ball weighing approximately 2.7 grams, traveling at 2 m/s is approximately ±2% of the weight of the ball. The second assumption is that the ball will bounce ideally, meaning elasticity will be preserved. This assumption is taken into consideration since the path of ball inclining, will replicate the ball's path on the decline. (RN)

Now that the assumptions have been taken into consideration the following equation can be used: (RN)

$$\mathrm{futureX}[\,\mathrm{len}(\mathrm{pastX}) - 1 - \mathrm{i}] = (\mathrm{pastX}[\,\mathrm{len}(\mathrm{pastX}) - 1] - \mathrm{pastX}[\mathrm{i}]) + \mathrm{pastX}[\,\mathrm{len}(\mathrm{pastX}) - 1] \quad (1)$$

$$\mathrm{futureY}[\,(\,\mathrm{len}(\,\mathrm{pastY}\,) - 1 - \mathrm{i}] = (\,\mathrm{pastY}[\,\mathrm{len}(\,\mathrm{pastY}\,) - 1] - \mathrm{pastY}[\,\mathrm{i}\,]) + \mathrm{pastY}[\,\mathrm{len}(\,\mathrm{pastY}\,) - 1] \quad (2)$$

Using equations (1) and (2), the future position of the ball will be calculated, and then accounted for in a reverse order. In other words, the final position of the ball will be calculated, then the previous points back to the peak of the ball will be calculated. A simulated scenario can be seen in the following, Figure 12. (RN)

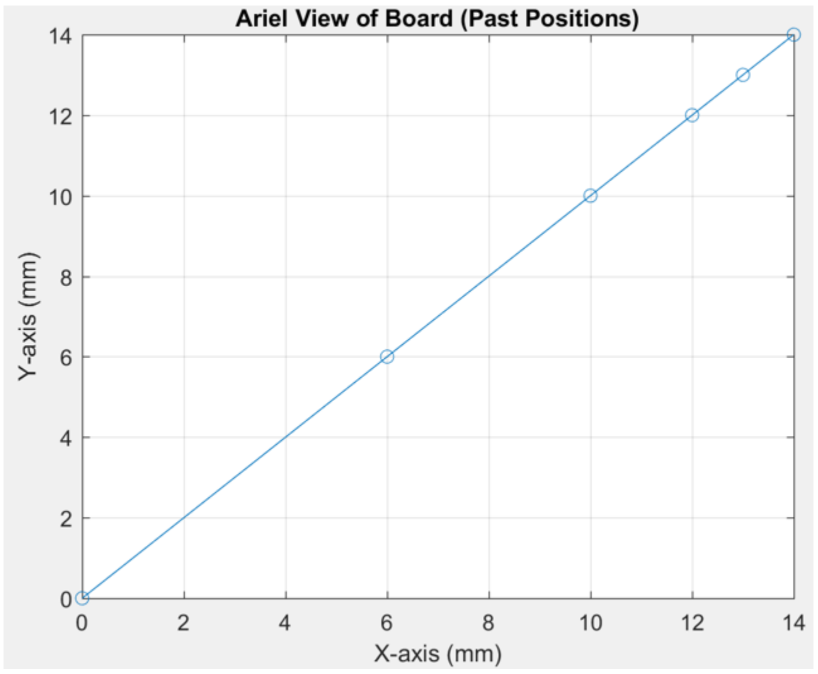


Figure 12: Simulated Past Positions

The 6 points were then passed to the equations mentioned above to calculate the path of the ball in the future as these points were taken on the rise to the peak of the ball. The future positions can be found in Figure 13. (RN)

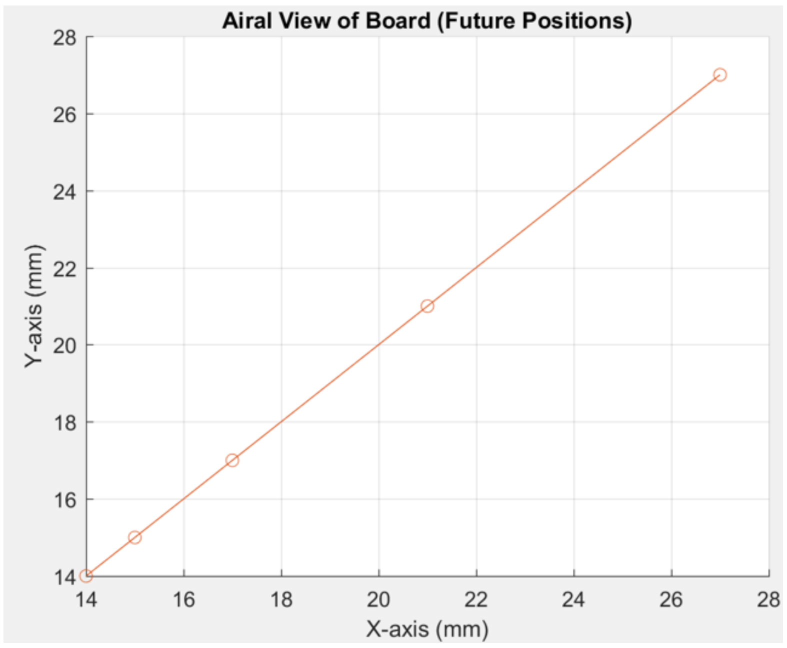


Figure 13: Simulated Future Positions

As seen in Figure 13 the points are in the future showing that the ball will land at approximately 27, 27. This shows that the equation is able to predict the future path of the ball and to a fairly accurate degree. It should be noted that due to the simulation method that was used, the arrays started at 1 rather than 0 as they do when real time programming; this change shifted the points and caused minute inaccuracy. (RN)

3.9 MATLAB Simulation Object Tracking

Due to the multitude of variables that affect image recognition, some simulations were completed in order to determine if tracking a table tennis ball on a single microcontroller was plausible. The first step was to prove the concept using a computer with MATLAB. Thus, this will be proof of tracking a ball using a 30 FPS camera built-in a laptop running MATLAB. (RN)

The first step was to import the camera components into MATLAB. Using one of the various image importing packages on MATLAB this can be complete. Once the software is open and a camera is completely installed, an image must be captured in order to start filtering the image. In Figure 14, this is an image of an orange ball that is being held mid-air. (RN)

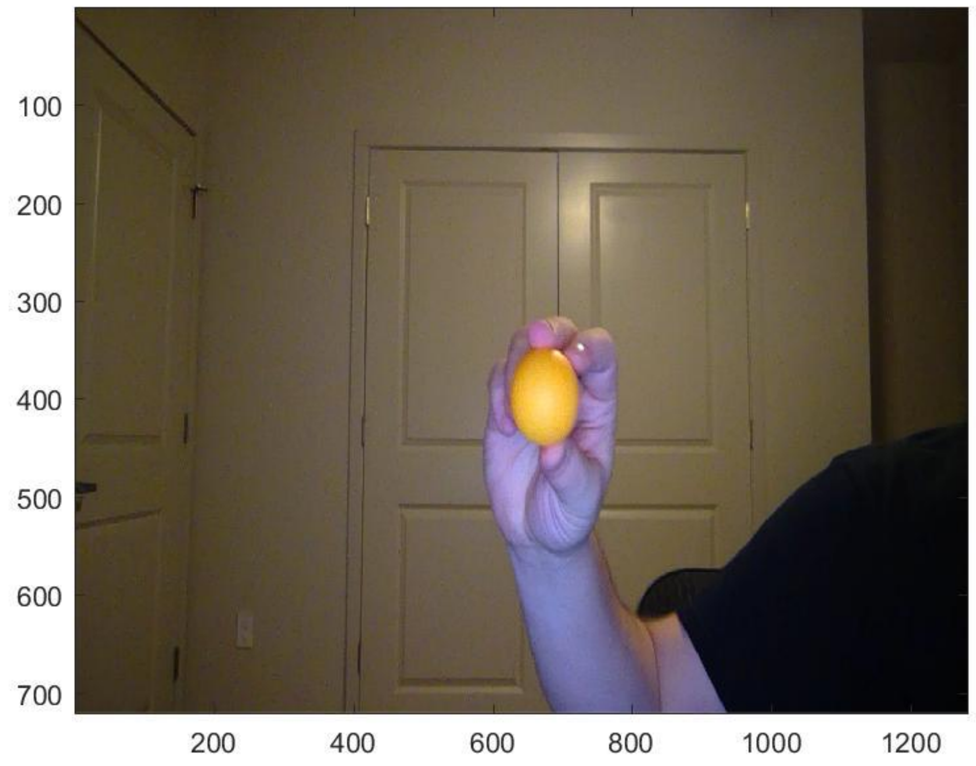


Figure 14: Simulation Base Image of Table Tennis Ball

Once the image has been captured the next step is to start filtering out the unwanted pixels of the image. This can be done in many ways; however, MATLAB is very limited on the possibilities to filter an image. This limitation is good for simulation, due to the fact that once implemented on actual image recognition software, many other, more advanced techniques can be used to filter out the desired pixels. Nonetheless, MATLAB uses RGB filtering and this can be found in Figure 15. (RN)

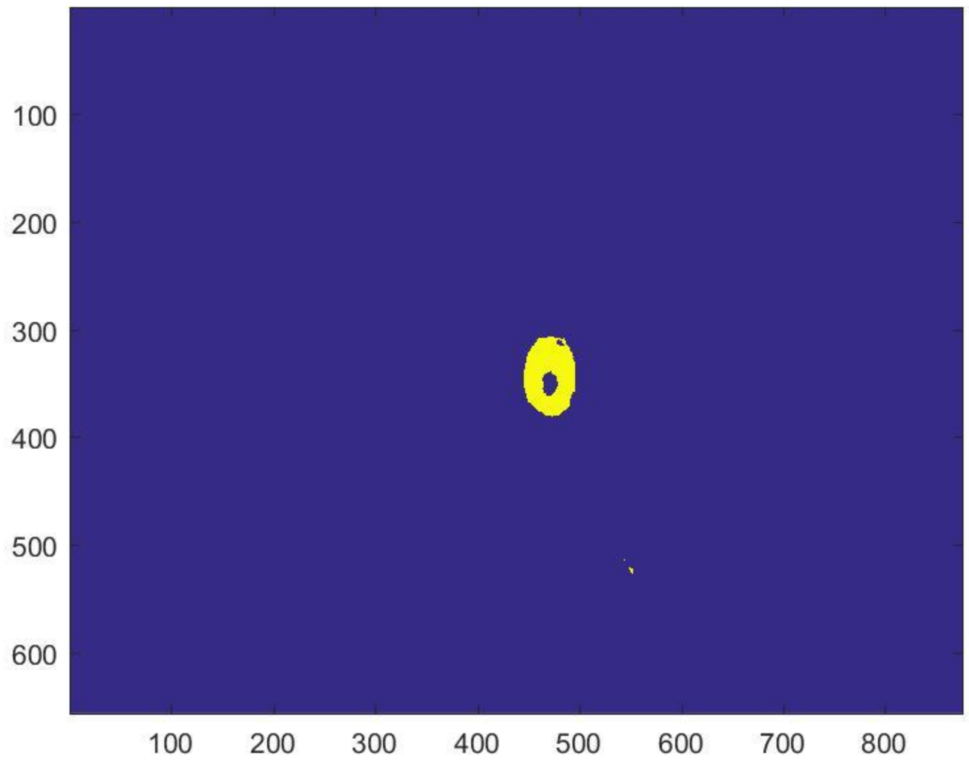


Figure 15: RGB Filter of Base Image for Simulations

As can be seen in Figure 15, there a few pixels that manage to squeeze through the filtering, and the filter did remove some parts of the ball. However, as it turns out, the sheer number of desired pixels is approximately 300 times greater than the few that managed to not get filtered, thus these pixels can be ignored. (RN)

Now that the image has been filtered to the degree that the user desires, a center point needs to be computed and placed on the image to ensure that the calculations are as expected. This method is completed by reading the image stream and then computing the first pixel and the last pixel, then simply finding the midpoint between the pixels. Doing this around the constant edge of the ball results in the center point. Again, this can be observed in the following Figure 16: (RN)

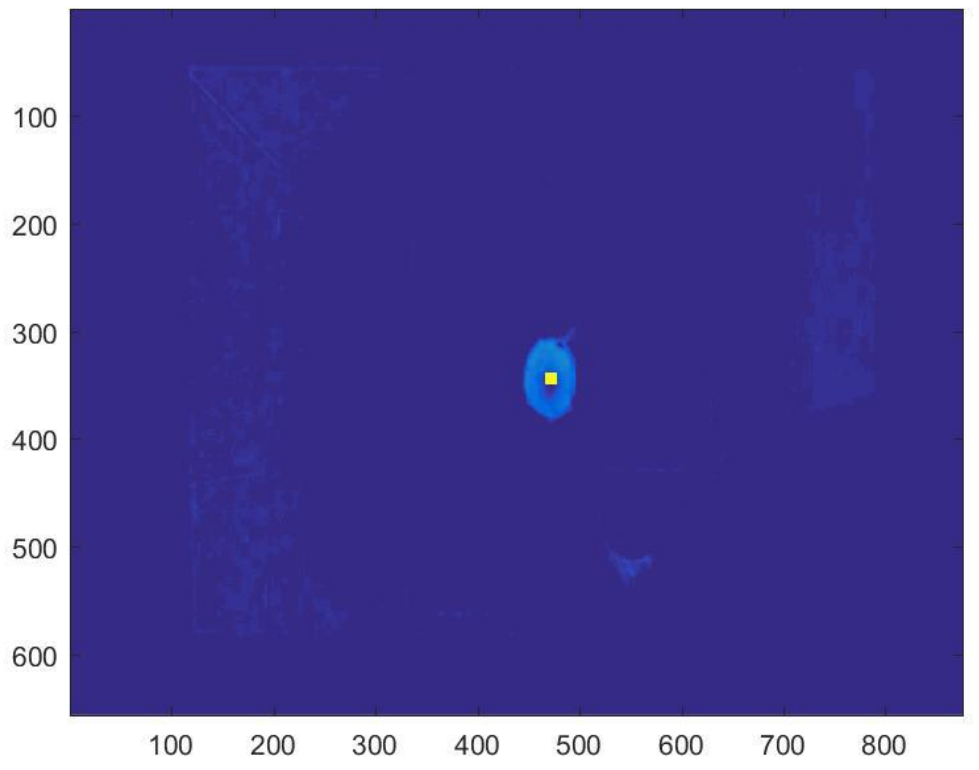


Figure 16: Midpoint of Base Image for Simulations

Now that the midpoint is calculated, the midpoint coordinates (X, Y) can be stored in an array on a frame by frame basis. Using that array, the aforementioned formula can be used to compute the landing position of the ball. In order to observe this in action, Y direction array is appended until the previous value is greater than the current position. This means that the ball is in free fall, or in other words, in its downward swing. Now the appending code, thus stops appending the midpoint and then computes the future position. (RN)

3.10 OpenCV Object Tracking

In order to test the accuracy, speed, and concept of tracking a colored ball using openCV and Python language in a personal computer using a built-in webcam, few steps were done. First, installing Python software version 2.7, openCV version 3.3, and Anaconda, which has all the necessary packages to run and implement the testing of the program using the installed Python software. This test was done with the help of an online tutorial, '*Ball tracking with openCV*' [9]. The goal of this research was to track the center of the ball by drawing a dot on the tracked center. The second step was to use an open source code and modify it to implement it in Python to fit this research properly. The results were shown in Figure 17: (MQ)

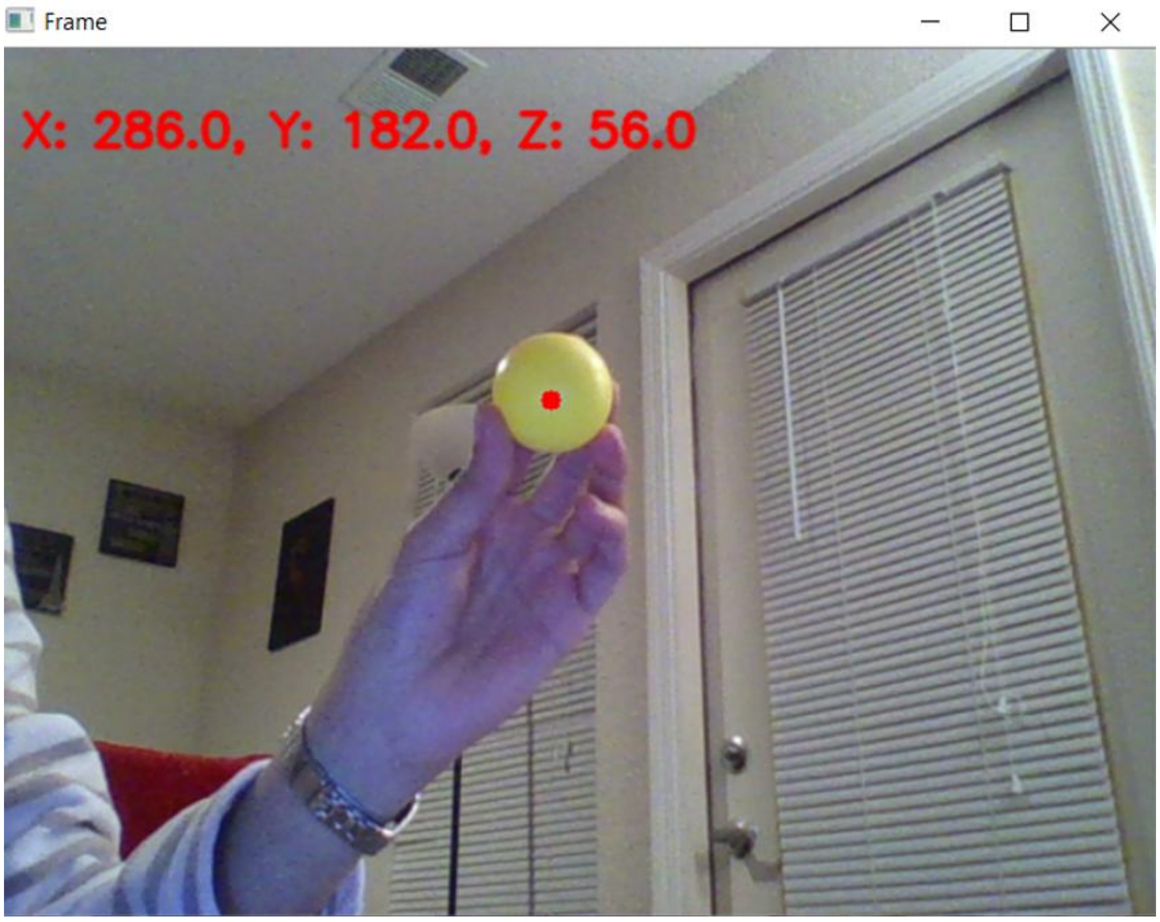


Figure 17: The resulting implementation of center finding using openCV

Figure 17 shows the center of the ball, (x, y, z) position in pixels, where x is the horizontal position, y is the vertical position, and z is the diameter of the ball, which can be interpreted as depth. In order to detect the yellow ball shown in the Figure, a loop started to grab frames from the camera and the OpenCV code went through the frames and resized the image a scale of 600 width. Then, the image was converted to a HSV color space to determine the yellow color of the ball. At that point, the lower and upper thresholds were set in order to look for the mask. The program also used two built in functions within openCV, which are erode and dilate. These two functions were used to compute the center of the ball by isolating the desired ball outline and superimpose the point on the image. Another important concept that was used in this model was the contour. It was used to find the actual ball location. Another loop was then used to find the center of the ball by looking for the largest contour in the image and focus only on the center. The contour then delivered the center (x,y,z) coordinates and displayed the coordinates on the screen. The modified part of the code was added to draw a dot in the center. Then, the (x, y, z) coordinates were shown in Figure 17. (MQ)

3.11 B.O.B. Webpage

In order to add some features to the project, a webpage was constructed to track the ping pong ball and the important static information of the project. For this part, a third raspberry pi was used for the server. This raspberry pi had its' own camera to act as a webcam server in order to display a streaming video, as well as the number of bounces of the ping pong ball. This was done using OpenCV software. (MQ)

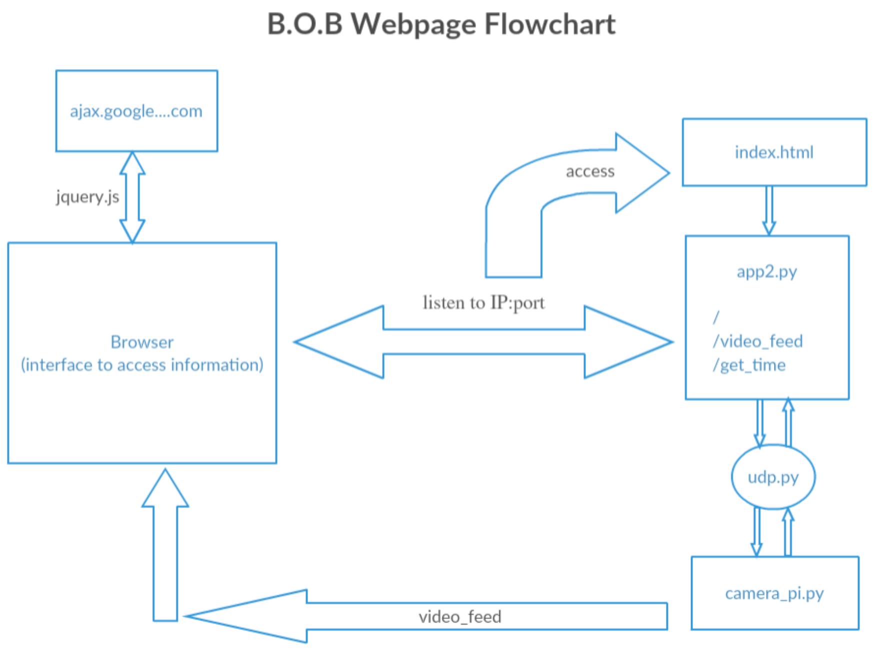


Figure 18: Flowchart of the Webpage

3.11.1 Theory of Operation (Browser Side)

1- When the initial request comes into the webserver, the app2.py file sends the index.html file to the browser.

2- The browser loads the jquery.js library file from the google site.

3- The browser then uses the jQuery library to execute the ajax code in the index.html file. The ajax code sends a request to the /get time URL, which is received and processed by the app2.py code. That request is fulfilled when app2.py returns json string containing the current time and the number of bounces. These strings are then parsed by the browser and used to change the text elements on the webpage. This process is called repeatedly by the browser using *setInterval* method in the JavaScript. (MQ)

4- The browser also parses the “img” tag in the index.html file. This tag sets up a separate communication with the app2.py file using the /video_feed URL. This is constantly refreshed by the app2.py file using the *gen* function, which repeatedly calls the camera_pi.py module. This action will process the video and return a stream of jpeg files that are routed to the browser. (MQ)

3.11.2 Theory of Operation (Server Side)

The main application “app2.py” starts up a webserver application based on the Flask module in python. The Flask module respond to requests from the browser based on the base URL and routes that are contained in the subdirectories after the base URL. For instance: if the base URL is the IP address http://172.20.10.3, the browser would make requests to a subdirectory such as video feed by using the URL http://172.20.10.3/video_feed/. These directories are known as routes within the Flask terminology. The application app2.py has three main routes for handling different request from the browser:

1- / route: it serves the main webpage, which is defined in the index.html template. This route returns the html and JavaScript that defines the overall functionality of the web app running on the browser. This includes the title of the webpage, the headings and text that appear on the page, as well as the JavaScript that makes the page active. (MQ)

2- video feed route: this route returns the streaming video. The app2.py file in this route repeatedly calls the camera_pi.py module, which process the video and returns a stream of jpeg files that are routed to the browser.

3- The get time route: this route returns via a JSON packet, the current server time and the number of bounces that have been counted by the camera_pi.py module. The communication between the camera_pi.py module and the app2.py module is accomplished using the udp protocol to communicate over the network interface to get the number of bounces. (MQ)

3.12 Mechanical Design

A few actuator designs were initially considered. The first design considered using a gyroscope to tilt the paddle; however, this idea was deemed too difficult and too expensive to implement successfully. The next idea was to use stepper motors to tilt the paddle, comparable to how an inkjet printer works. Stepper motors are a good choice as they are designed to provide accurate incremental steps and high torque at low RPM. The motor would be connected to a timing belt pulley to transfer radial movement into linear movement. The paddle, which would be a square surface with the same size diameter as a regulation size table tennis paddle, would be connected to a fixed position on the timing belt. This design is illustrated in Figure 19 below:

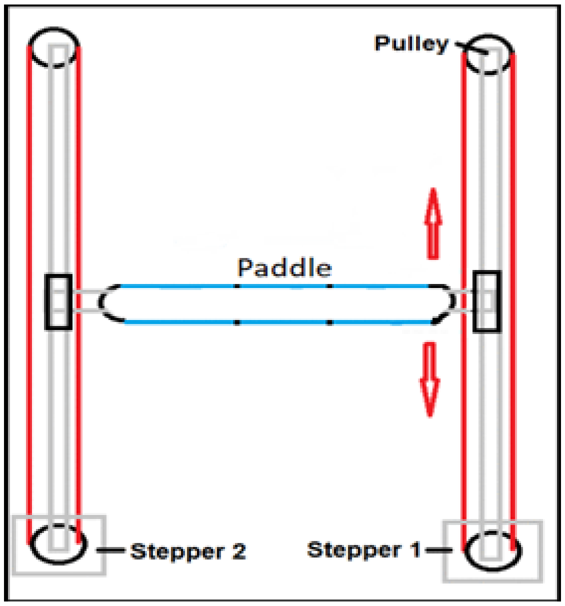


Figure 19: Mechanical Design, 1-Axis

This design would be replicated for the x-axis and the y-axis, resulting in four total belt drive motor configurations. The design would allow for moments along two axes, X and Y. Two motors, located opposite of each other, would be used to create the moment by varying the height of each support with respect to the starting position. The four motors, used in this fashion, will produce the full range of motion needed to control the table tennis ball. (EB)

3.13 Structural Design

Wood framing would be used to enclose the support system. The design would be such that the paddle is sunken down below the max height of the supports. There would also be a slit in the inner wall to allow the connection joints between the timing belt and paddle enough room to move freely. The sensor system would also be located within the slits to provide both an unobstructed view and protection from the elements. The outer wall would comprise of polycarbonate to allow an individual to view the timing belt system working in real time. Holes would be drilled in the side of the outer wall for air to freely move beneath the paddle and assist in cooling both the processor and motors. A preliminary design for this system, drawn in AutoCAD 3D, is illustrated below in Figure 20. (EB)

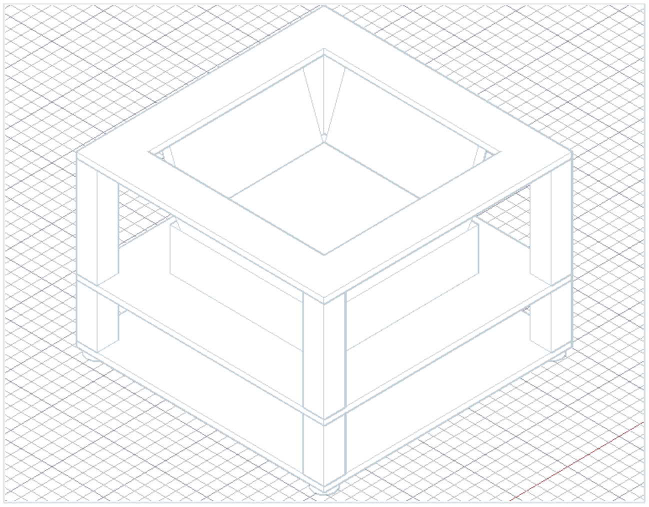


Figure 20: Preliminary Structural Design

3.14 Control System Modeling

In elementary physics, a projectile is understood to be an object upon which gravity is the only acting force. This principle is illustrated in equation (3). Given an initial height and velocity, the final height can be calculated for the given time, T. The continuous time system in equation (3) can be modified to equation (4) to demonstrate a discrete time system. For this system, T is the sampling period. The initial height and initial velocity are now the current iteration of system while the final height is the next iteration. (EB)

$$h_o = h_i + v_i T - 0.5gT^2 \quad (3)$$

$$h(k+1) = h(k) + v(k)T - 0.5gT^2 \quad (4)$$

The continuous time system in equation (5) will provide the final velocity for a given time period. The same process can be applied for velocity as well, resulting in a discrete time system in equation (6). T remains the sampling period while the initial velocity is now the current iteration of system while the final velocity is the next iteration. (EB)

$$v_o = v_i - gT \ \ (5)$$

$$v(k+1) = v(k) - gT \ \ (6)$$

For bouncing projectiles, it is important to consider the type of collision involved when the object strikes an obstacle. In an elastic collision, both momentum and kinetic energy are conserved. Almost no energy is lost to sound, heat, or deformation. In an inelastic collision, momentum is conserved, but kinetic energy is not. Some of the energy is dissipated in the form of heat, sound, and deformation of the objects. If a ball were to be dropped from an initial height, the ball would bounce off the ground. For an elastic collision, the ball would return to the initial height; however, perfectly elastic collisions do not exist in the real world. Most cases, such as a table tennis ball, take on the form of an inelastic collision. This is illustrated in equation (7), where the symbol *e* represents the coefficient of restitution. For a table tennis ball, this coefficient has been found to be between 80% and 85% [10]. When an object collides with an obstacle, the velocity reflected off the obstacle is equal to the velocity of the object before the bounce multiplied by the coefficient of restitution. The negative sign signifies the velocity is moving in the positive *h* direction after the bounce. (EB)

$$v(k+1) = -ev(k) \ (7)$$

The above discrete time system equations can be combined to form a model for a table tennis ball bouncing on an immovable obstacle (ground, paddle). With an initial height chosen to be 0.2032 [m], the plot in Figure 21 can be produced to illustrate a bouncing table tennis ball settling to a height of 0 [m]. (EB)

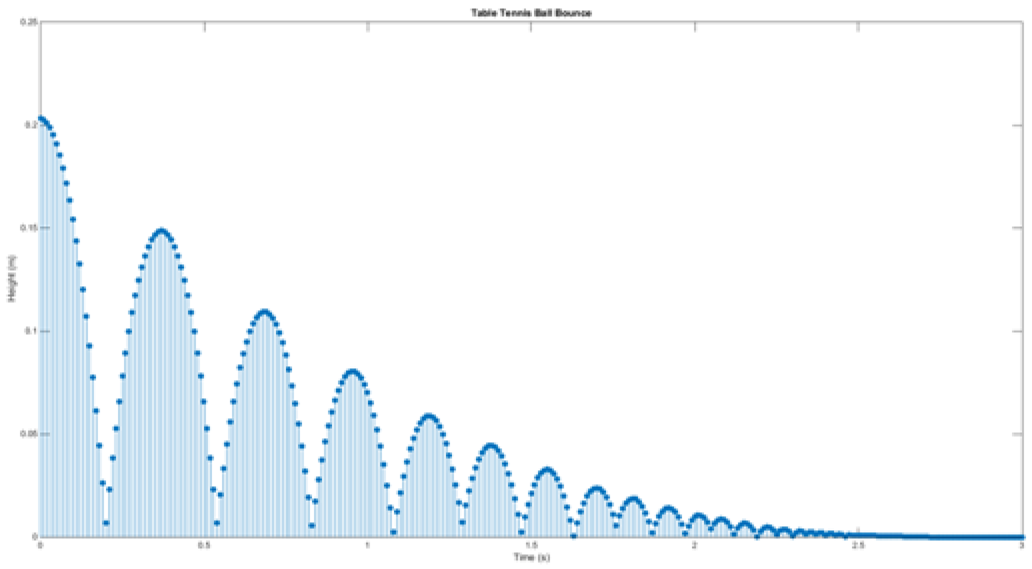


Figure 21: Table Tennis Ball Bouncing

In order to maintain the chosen height for table tennis ball, the paddle controller must be able to provide the energy lost when the ball bounces off the paddle. The difference in the energy lost is a direct result of the scaled velocity due to the coefficient of elasticity. By supplying enough force to add additional velocity to the ball, the table tennis ball should be able to reach the target height after every bounce. Equation (8) is needed in order to gain an understanding of the velocity required to return the ball to the target height. Once a target height is selected (h=0.2032 [m]), the target velocity can be calculated. The velocity of the paddle is calculated by taking the difference of the target velocity before the bounce and after the bounce, as illustrated in equation (9). Knowing the speed, the motors must produce is enough to generate an adequate controller for the system; however, a discussion is necessary as to why the force the paddle must produce to hit the ball is unnecessary. Suppose a paddle were to strike a standard table tennis ball with enough force to cause an acceleration of 100 [m/s]. Then the force here, using F=ma, would be calculated to be 0.27 [N]. This illustrates that finding the required force to move the table tennis ball is unnecessary as the value for the force is negligible. Another force that may be neglected is the force the ball exerts on the paddle. This is due to the large difference in mass between a standard ping pong paddle (70 - 100 grams) and a ping pong ball (2.7 grams). This means that even with the 70-gram paddle, the ball is still 25 times lighter than the paddle. The force produced by the ball on impact would be miniscule compared to the force the paddle could produce when it would strike the ball. (EB)

$$v_{target} = \sqrt{2gh_{target}} \quad (8)$$

$$v_{paddle} = v_{target} - ev_{target} \quad (9)$$

Two cases are illustrated below. In Figure 22, the ball starts at a higher initial height than the target height. After every bounce, the ball reaches a lower peak than the previous time. This holds true until the ball reaches the target height (the red line); the point where the velocity after the bounce plus the added velocity of the paddle are just enough to return the ball to the chosen height. (EB)

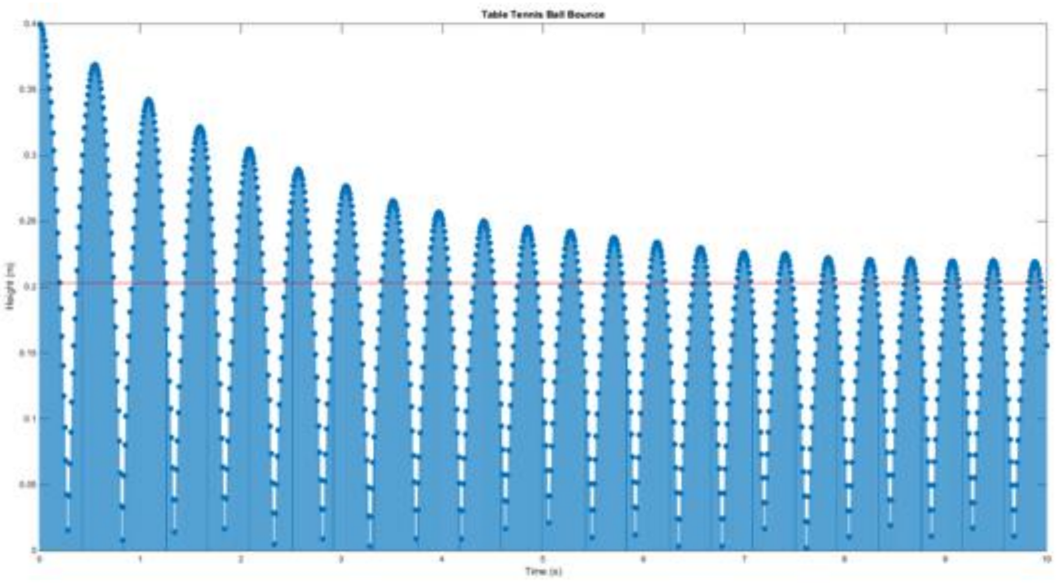


Figure 22: Table Tennis Ball Bouncing w/ Paddle (Above Target Height)

In Figure 23, the ball starts by resting on the paddle. The ball receives an impulse from the paddle. After every bounce, the ball reaches a new peak height as the velocity provided by the impulse force of the paddle is greater than the velocity before the bounce. This process continues until the ball reaches the target height (the red line); the point where the velocity after the bounce plus the added velocity of the paddle are just enough to return the ball to the chosen height. (EB)

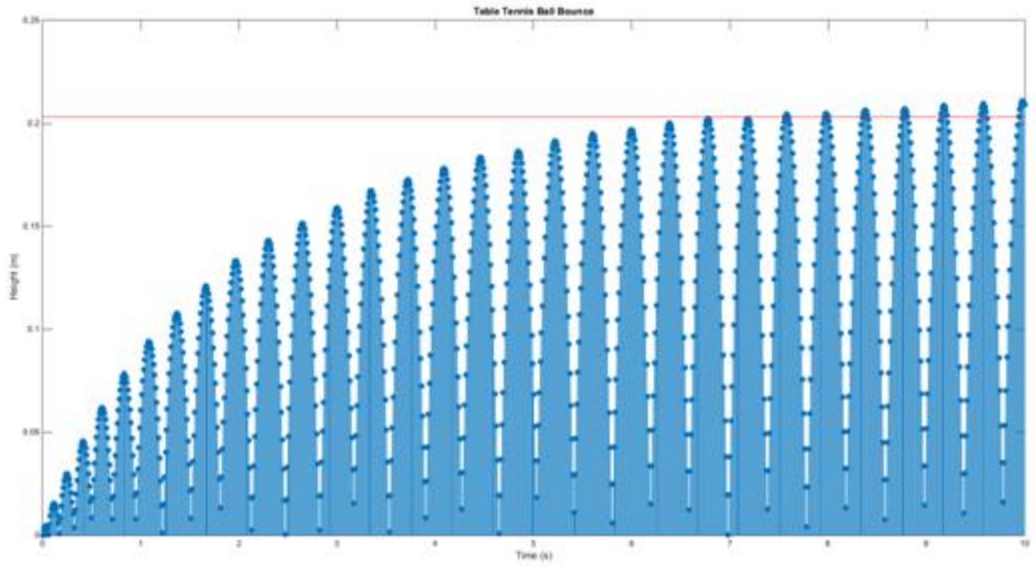


Figure 23: Table Tennis Ball Bouncing w/ Paddle (Below Target Height)

In the real world, the ball will not just have a vertical velocity; there will also be an associated horizontal velocity in both the x and y direction. To gain an understanding of how this can be corrected, the law of reflection, illustrated in Figure 24, must be considered. (EB)

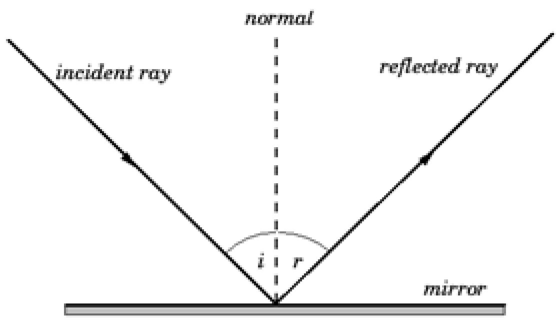


Figure 24: Law of Reflection

In the law of reflection, an incident ray strikes the surface (mirror) at an angle *i*, with respect to the normal. The reflected array will remain in the same plane as the incident ray and, therefore, will be reflected at an angle *r*, with respect to the normal. The angles i and r will be equivalent, and magnitude of the reflected ray will have the same value as the magnitude of the reflected ray. This principle can be applied to the table tennis ball bouncing off the paddle. The overall velocity, for the X and Y horizontal directions, can be found using equation (10). From this, the angle of the ball with respect to the normal can be calculated, similar to Figure 24. The derivation leads to equation (11), where the angle is the arcsine of the horizontal component over the total velocity. (EB)

$$v_o = \sqrt{v_{x,y}^2 + v_h^2} \quad (10)$$

$$\theta = \sin^{-1} \frac{v_{x,y}}{v_o} \quad (11)$$

The paddle could then be tilted in such a way that the incident angle is reduced to 0 degrees, thus resulting in the ball being reflected back from where it came from; however, simply adjusting the tilt of the paddle to the incident angle will result in a marginally stable system, where the angle is never corrected, and the ball just jumps between two points endlessly. Instead, tilting the paddle in such a way that a scaled version of the incident angle is reflected, resulting in a reduction of the horizontal velocity while forcing the ball back towards the center of the paddle. This scaled, reflected angle can be seen in Figure 25. There are three important parts illustrated in the plot. One, the incident angle is reflected in the opposite direction. Two, the reflection angle is scaled by a factor. Three, the angle is only adjusted whenever the ball strikes the paddle. (EB)

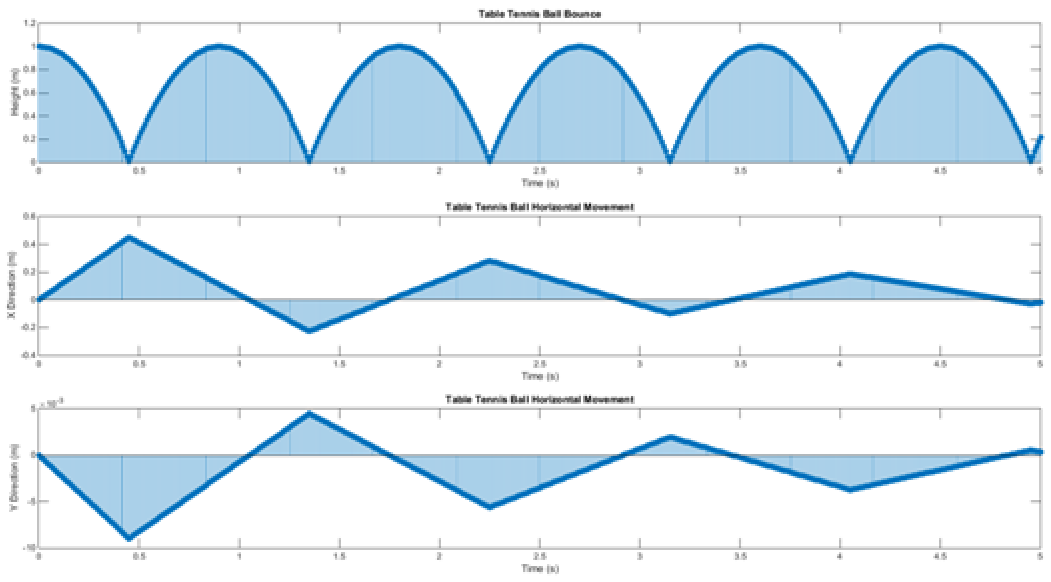


Figure 25: Table Tennis Ball Horizontal Velocity w/ Paddle

Figure 26 is scaled to illustrate that by using the technique described above, the horizontal components will be reduced to 0, and the ball will have reached a steady-state value centered at the origin of the paddle. (EB)

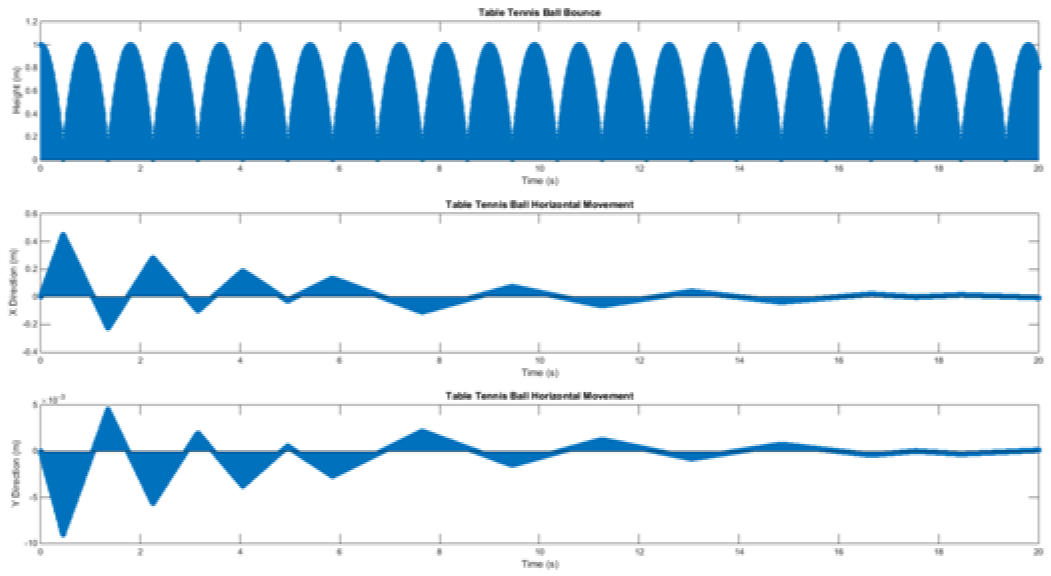


Figure 26: Tennis Ball Horizontal Velocity w/ Paddle

3.15 Control System Design

Using the standard position and velocity equations, a transfer function for the plant would be similar to equation (12) below:

$$G_P(z) = \frac{0.5T^2(z+1)}{(z-1)^2}U(z), T = 0.4 \quad (12)$$

The variable *T* represents the amount of time between bounces based on the target height. *U(z)* is the input for system. For the above transfer function, the input would be an acceleration added to the system. To modify this for B.O.B, a relationship had to be found between acceleration and the angle of the paddle. By using the law of reflection and equation (11) in the control system modeling section, it can be shown that for every one degree increase in angle, the horizontal velocity increases by 0.0175 [m/s]. In order to relate the angle to velocity, this value will need to be scaled by a factor of 4, resulting in 0.07 and the angle input, $\alpha(z)$, replacing the step input, *U(z)*. Equation (12) can be modified with the angle to velocity relation, resulting in equation (13). The system input would be the degree of the angle of the paddle while the system output would be the ball's position.

$$G_P(z) = \frac{0.035T^2(z+1)}{(z-1)^2}\alpha(z), T = 0.4 \quad (13)$$

The transfer function is a double integrator, and the plant will need a PD controller in order for the system to produce the desired system response. The pole of the PD controller can be placed at the origin of the z-plane while the zero should be placed close to the poles of the plant in order to have an effect on the system response. By setting the zero to 0.9, the controller equation can be found to be equation (14).

$$D(z) = \frac{k(z-0.9)}{z} \quad (14)$$

By placing the controller and the plant in a closed loop system, the root locus in Figure 27 is produced.

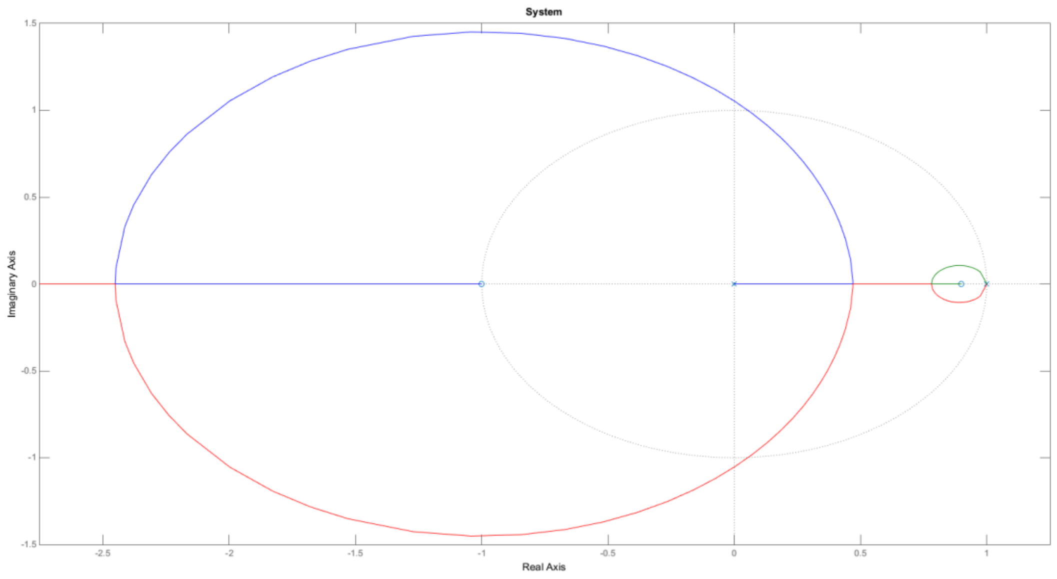


Figure 27: System Root Locus

This shows that the system is now stable for a range k={(0,17)}. By setting k=1, the step response output for the controller and the system are shown in Figure 28.

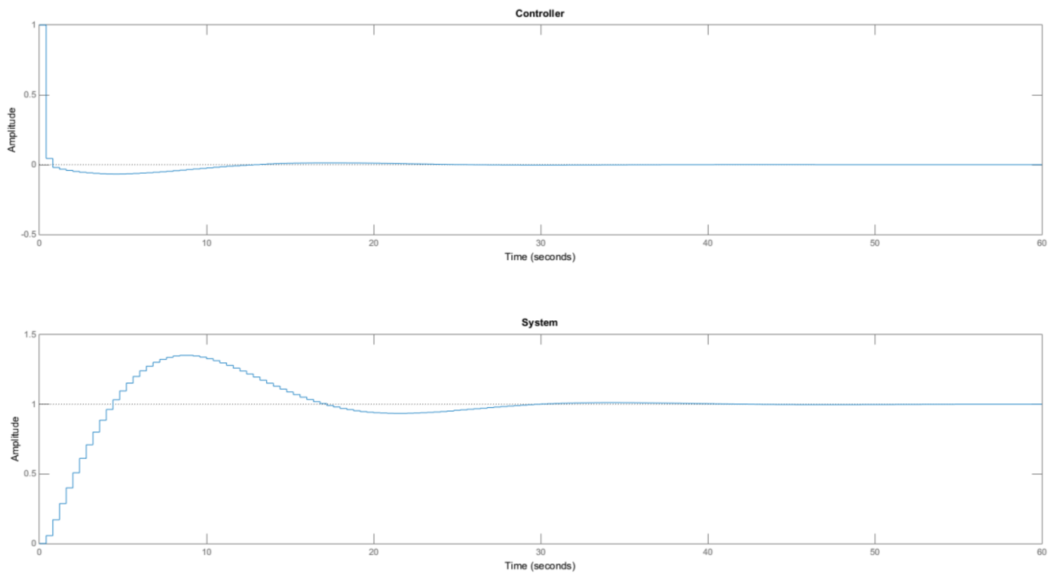


Figure 28: Controller/System Step Response

The figures illustrate that miniscule changes in the controller are enough to control the system. This is important as the controller output directly correlates to how much linear movement the stepper motors must move the paddle in order to control the table tennis ball. The initial impulse of the motor control signal is what produces the ball's horizontal velocity towards the center of the paddle. Immediately after this first strike, the paddle begins to reduce the ball's horizontal velocity by tilting in the opposite direction of the initial tilt. This helps to slow the ball as it approaches the center of the paddle. Disregarding the initial impulse, the maximum amplitude the controller achieves is 0.065. This value illustrates that the tilt by the paddle needed to correct the table tennis ball's trajectory is easily achievable with the designed PD controller. (EB)

3.15.1 Control Pseudo Code

The following pseudo code shows the code that will be used by the controller to calculate the motor response based the ball's position. (EB)

```
# Initialization
Import Libraries
Variable Setup
GPIO Pin Setup
Motor Setup
PID Setup
PID Variables
Serial Setup

#Main Loop
while True:
    while True:
        Read Serial Line
        if  There Are Coordinates:
            Assign Coordinates to Variables
            while True:
            if X Position is not Zero:
            Error = X Target - X Position
            Calculate Integral
            Calculate Derivative
            PID = Gain*(Error + Integral + Derivative)
            Save Variables
        while True:
            Error = Y Target - Y Position
            Calculate Integral
            Calculate Derivative
            PID = Gain*(Error + Integral + Derivative)
            Save Variables
    Set Direction of Motors
    Tilt Y-Axis
    Tilt X-Axis
    Motors Move to Strike
    Set Direction of Motors
    Return Motors to Starting Position
Clean Up GPIO Pins
```

3.16 Motor System Design and Constraints

In order to control the motion and angle of the paddle, a precise motor system capable of producing the necessary amount of force to counteract the ball will be needed. For this application, a system of stepper motors will be used as they are high-torque, high-precision motors that operate in steps, capable of moving in very small increments. These stepper motors will be used in conjunction with a belt drive system, which will act as a pulley to move the paddle. In order to specify the exact motor that will be needed for this system, some constraints must first be discussed. (DA)

First, the motor system must be capable of holding the paddle upright. For design constraint purposes, it is safe to assume that the mass of the paddle will be around 100 grams, and that the gear to be used for the belt drive will be about 1 centimeter in diameter. Applying this with Newton's Second Law of Motion:

$$T_{hold} = \frac{m_{pad*a_{grav}*d_{gear}}}{2} = \frac{(0.1kg)\left(9.8\frac{m}{s^2}\right)(0.01\,m)}{2} \quad (15)$$

The estimated necessary holding torque of the system is found to be roughly 0.49 N-cm. However, since stepper motor torque often declines at increasing speeds, the desired holding torque should much higher than this. Therefore, an arbitrary holding torque of at least 1 N-cm will be desired from the motor system, as that is double the value calculated in equation (15) above. (DA)

Along with the torque constraint, another major constraint for the motor system is reaction time. As B.O.B. is to be designed to react to a ball dropped from about 10 centimeters, that leaves about 142.8 milliseconds of time before the ball reaches the paddle, according to Newton's Second Law:

$$t_{min} = \sqrt{\frac{2*h_0}{a_{grav}}} = \sqrt{\frac{2*(0.1\,m)}{9.8\,m/s^2}} \quad (16)$$

Knowing this, the motor system should be able to react in at least one-third of that time, so as to leave time for the sensor and control processes to complete and send the necessary information, which leaves the motors with about 47.6 milliseconds to drive upward and strike the ball, and to adjust the paddle to the appropriate angle to keep the ball bouncing above the paddle. (DA)

For this application, four NEMA 17 17HM08-1204S stepper motors have been selected. According to the 17HM08-1204S data sheet, which can be reference in the appendix, each of these motors is capable of delivering 11 N-cm of holding torque, which is more than enough for this application. This motor also offers a step angle of 0.9°/step, which when combined with the estimated 1-centimeter diameter gear, will only move the paddle by 78.54 μm per step, allowing for great precision. However, since these motors are rated for 1.2 Amps of current, a driver will be needed for each in order to turn the 3.3V output pulses of the microcontroller into a rated current input to the motor coils. For this application, the Texas Instruments DR8825 stepper motor driver was selected. As this driver is rated for up to 1.5 Amps of current without heat protection, and is capable of micro step drive, it will be well capable of driving the 17HM08-1204S stepper motor. Looking back to the Control System Modeling section, a velocity of 0.2995 m/s will need to be added to the ball, according to equations (8) and (9). Factoring in the kinetic energy equation, the necessary paddle velocity v_p is found to be 0.0492 m/s using equation (17). Further, by factoring the estimated 1-centimeter diameter gear of the belt drive, the stepper motors will need to move at a speed of approximately 94 RPM, in order to keep the ball bouncing at a height of approximately 20 cm. In order to reach this angular velocity, a square wave with a frequency of about 627 Hz, assuming full-step drive, will need to be input to the DR8825 driver, according to equation (18) below

$$v_p = v_{ball} * \sqrt{\frac{m_{ball}}{m_{pad}}} = \left(0.2995\frac{m}{s}\right) * \sqrt{\frac{(2.7g)}{(100g)}} \quad (17)$$

$$f_{step} = \frac{(RPM)(360°/rot)\ (Step\ mode)}{(60\ sec/min)(step\ angle)} = \frac{(94\ RPM)(360°/rot)(1\ \mu step/step)}{(60\ sec/min)(0.9°/step)} \quad (18)$$

Along with generating the paddle velocity to strike the ball effectively, the motor system must also be capable of tilting the paddle along two axes of rotation. Four motors will be used, two for each axis of rotation. Because of this, the angle at which the paddle must be tilted can be considered in a single dimension by the control system. However, in order to implement the angle required by the control system, the motor control process must convert this linear angle into steps for each of the two dimensions. This means that the program must implement equation (19) below. Further, the program must implement equation (20) below using the 17HM08-1204S stepper motor, and still assuming a 1 cm gear diameter. (DA)

$$steps = \frac{360l}{\pi d_{gear}(step\ angle)} * tan\theta \quad (19)$$

$$steps = \frac{(360°)(50\ cm)}{\pi*(0.9°/step)(1\ cm)} * tan\theta = \frac{\mathbf{20{,}000}}{\boldsymbol{\pi}} * \boldsymbol{tan\theta} \quad (20)$$

To effectively drive the 17HM08-1204S stepper motors, a stepper motor drive is needed to convert the 3.3V pulse signals from the microcontroller, into the proper pulse current signals needed to drive the motors. Since the 17HM08-1204S stepper motors are rated for 1.2 amps of current, a driver capable of delivering that amount of current will be needed. For this application, four Texas Instruments DR8825s will be used to drive the four stepper motors. Since the DR8825 is rated to deliver 1.5 Amps without heat protection, it will be sufficient for this application. The DR8825 is also capable of micro stepping, which could be useful in producing a smoother motor response at constant velocity. A design schematic displaying the connections for DR8825 to the Raspberry Pi 3 is shown in Figure 29 below: (DA)

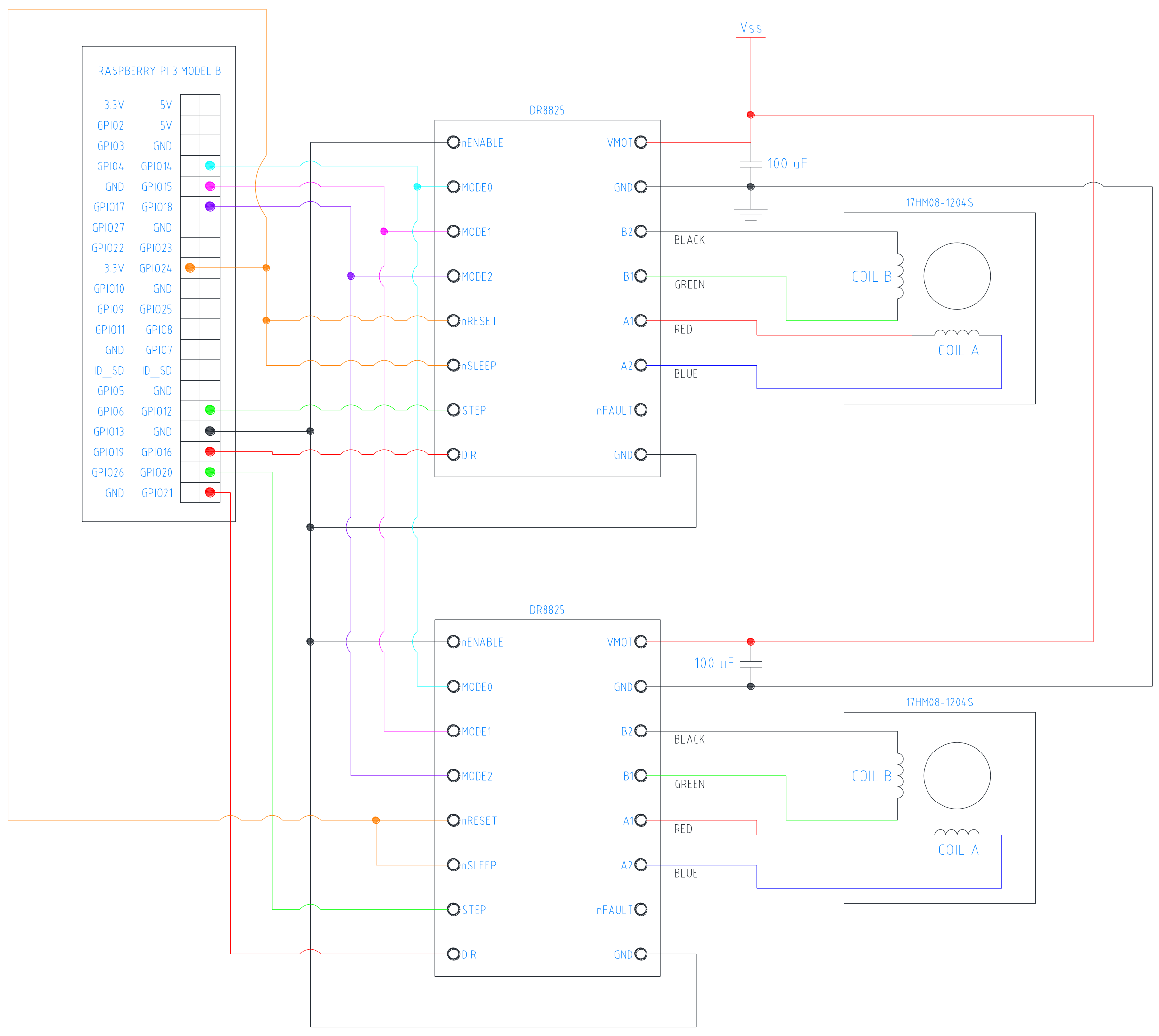


Figure 29: Motor System Schematic for X-Axis of Rotation

The schematic of Figure 29 above shows the necessary connections for the stepper motor drivers simplified into two motors acting on a single axis. The V_{MOT} port is connected to an external DC power source ranging from 8.2V to 45V V_{SS}. For the purposes of this document, V_{SS} is given a standard arbitrary value of 12V. The A and B pins are connected directly to the motor coils, as they will output the motor drive signals. The nENABLE pin is connected to logic ground, as it must be set to a logic low value for the driver to operate in active mode. The nSLEEP and nRESET pins are connected to the Raspberry Pi 3.3V output port as the sleep and reset function will not be needed for this project. The three MODE pins are used to set to the micro stepping mode based on a 3-bit input. These three pins are each tied together for all four drivers, as all four drives will always be operating in the same micro step mode. The STEP port requires an input square wave signal of frequency, f_{step} from equation (18) above, to control the response speed of the motor. Lastly, the DIR pin is used to control the direction the motor will spin, with a logic high and logic low driving the motor in opposite directions. For all of the drives, each of the MODE pins will be tied together with those of the other 3 drivers and connected to three GPIO pins on the Raspberry Pi. The respective STEP and DIR pins on the four drives will each need their own GPIO pin on the Raspberry Pi, as all four motors will need to be controlled individually at times. This means 11 of the 26 total GPIO ports on the Raspberry Pi 3 will be needed for motor drive operation. Furthermore, Figure 30 below further demonstrates the timing implementation that will be used in order to generate the pulses to drive the motors. (DA)

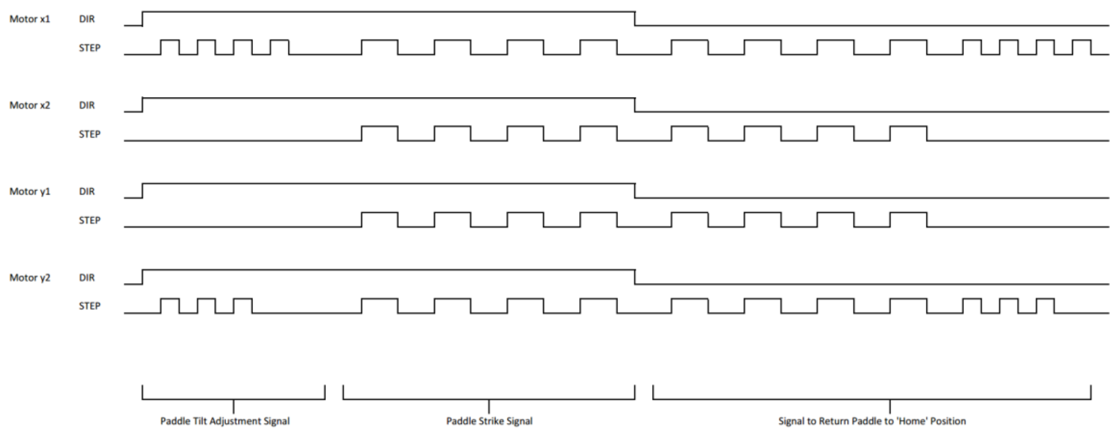


Figure 30: Example Pulse Timing Diagram for Paddle Motor Drivers

This timing diagram displays an example of an arbitrary drive sequence in which the paddle will tilt, then strike, then return to its initial position. For this, pulse signals will be sent to one or two motor drives in order to tilt the paddle in response to the control algorithm. Once tilted, all four motors will be driven simultaneously, and at the same rate to strike the ball. Afterwards, the direction pin signals will be changed to reverse the direction of all four motors, then the same signals will be driven through the step pin in order to return the paddle back to the same initial position. (DA).

3.17 Motor System Circuit Design

In order to simplify the circuit design of the motor drive circuit, a printed circuit board will be generated to simplify the interconnections of the four Pololu DRV8825 stepper motor driver boards and minimize the amount of connections needed to be made to the Raspberry Pi. Since all of the motor drivers will always need to be in the same step mode drive, all three of the Mode pins can be tied together respectively, requiring only three external connections to the Raspberry Pi instead of twelve. Likewise, all motors will need to be enabled or disabled at the same time, so all four of the nENABLE pins can be tied together as well. Both the 12 Volt and 3.3 Volt power supplies can also be tied together respectively to minimize the amount of external power connections that are needed, and further simplify external power wiring. However, the STEP and DIR pins must be pulled individually, as the motors will need to be able to act individually. The SLEEP and RESET pins will both be pulled logic high internally, as those pins will not be used in this project. A schematic design of this circuit board generated in Autodesk Eagle can be seen in Figure 31 below: (DA)

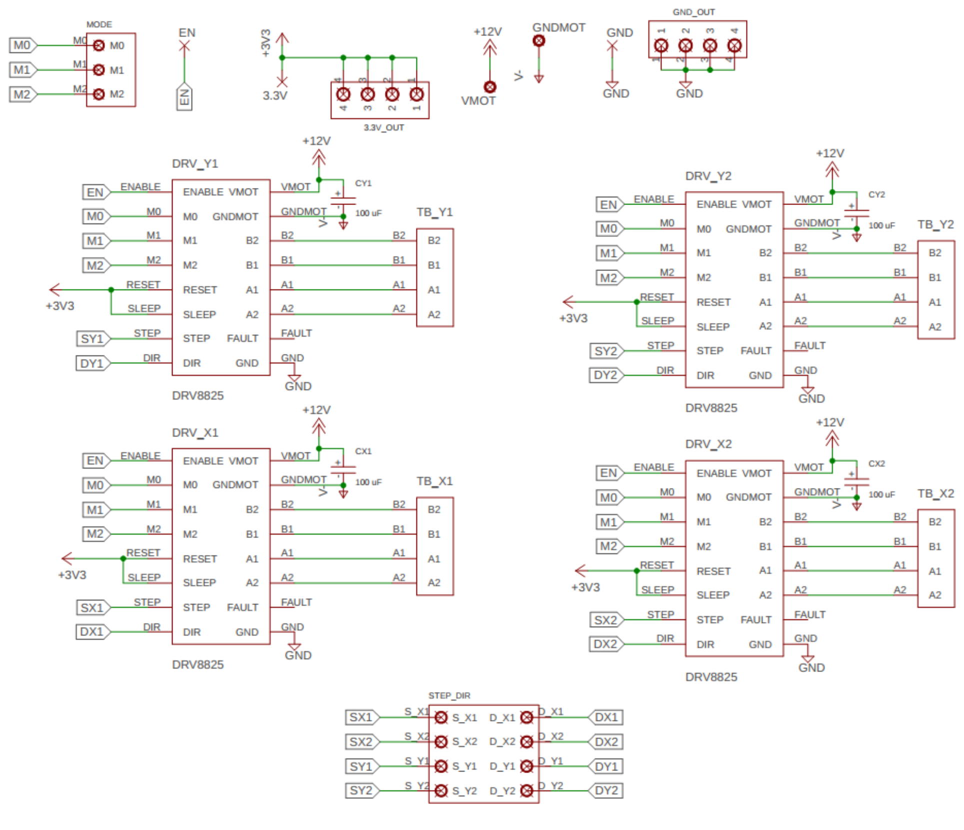


Figure 31: Motor Driver Printed Circuit Board Schematic

Once satisfied with this schematic design for the complete motor driver circuit, this design could then be implemented in a circuit board layout design. This layout design can be seen in Figure 32 below. (DA)

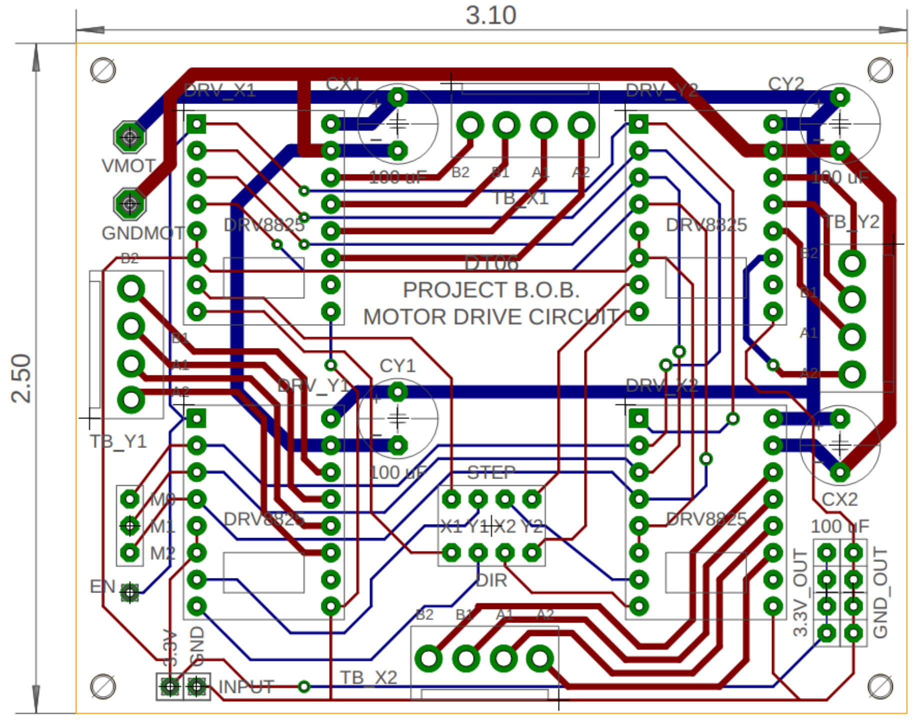


Figure 32: Motor Driver Printed Circuit Board Layout

The board of Figure 32 is a two-layer printed circuit board design to implement the schematic of Figure 31. It was designed using a 0.01-inch trace width for small signals, 0.024-inch trace width for each motor phase output, and a 0.05-inch trace width for all power rails. This was done to minimize heat dissipation from the potentially high-current carrying traces and prevent damage to the board. The red lines of Figure 32 represent the top-level traces of the circuit board, while the blue lines represent the bottom layer traces. The overall board is sized at 3.10 inches by 2.50 inches and has spare 3.3V and logic ground outputs for potential future expansion supplies that may be used in the final project. (DA)

3.18 Power Supply Needs & Design

Nearly all of the components of the project require a steady DC power supply. In order to account for this, some type of power supply system will need to be designed. While not all of the components have been selected yet, it is still possible to come up with some power supply estimates. As a basis for the estimate, the overall system shall likely include: 2-3 Raspberry Pi 3s, four NEMA 17 17HM08-1204S stepper motors, and four DR8825 motor drives. Planning for a worst-case scenario and using the basic power equation: P = VI, the power consumption of each respective component can be calculated. (DA)

At full load, the Raspberry Pi 3 draws 5.1 Volts and 2.5 Amps, meaning that three Raspberry Pis draw a total power of 38.25 Watts. Along with that, the 17HM08 stepper motors are rated for 3.6 Volts, and 1.2 Amps, or 8.64 Watts. Since the project will likely include 4 stepper motors, that means the power system could draw up to 34.6 Watts. (DA)

Adding all of these wattages together gives an overall estimated power consumption of 72.71 Watts for the overall system. Taking that number up a bit for a reasonable safety net, the overall power supply of the system should be rated to deliver around 100 Watts. Since all of the components in the system require a DC power input, the power supply will also need to be capable of supplying DC power. (DA)

Along with these power requirements, each component will need a regulated DC input voltage. As such, all of the Raspberry Pi 3s will need a 5 Volt regulated DC voltage input. Along with that, the DR8825 stepper motor driver is rated for a supply voltage range of 8.2 V to 45 V. As such, a voltage regulator within this range will be needed, and as such, a voltage regulator of 12 V should be sufficient and standard. As such, Figure 33 below shows a rough schematic of the power regulation (DA)

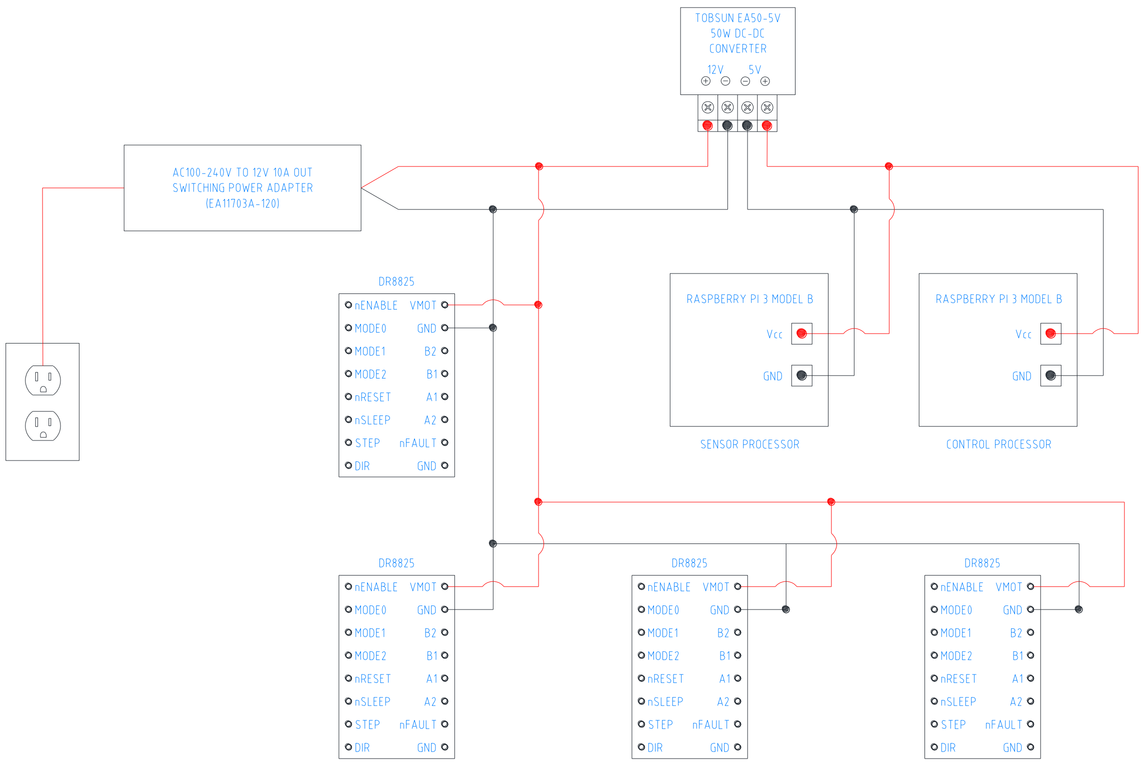


Figure 33: Power Supply System Schematic

As seen in the schematic of Figure 33 above, wall power is initially supplied to the system, and passed through a 12 VDC switching regulator. This regulator will provide the 12 VDC power supply needed for the four stepper motor driver inputs. This switching adapter is rated for 120 Watts, so it is more than sufficient to supply the entirety of B.O.B. However, as the Raspberry Pis require a 5 VDC input, the supply must be stepped down again. Since the goal of this power supply system is to only need one plug outlet, the 12 VDC output of the switching regulator will be passed through a 12/24VDC to 5 VDC regulator, which will feed the Raspberry Pis. This regulator is rated for 50 Watts, which is more than enough to supply the Raspberry Pi 3s, which will draw a maximum of 25 Watts between the two of them. (DA)

4.0 Project Parts

4.1 Parts List

Table 21 below compiles a list of all parts that are needed to construct the design of this report. Although additional parts may be desired in the future, this list covers all of the basics required by this design. (DA)

Table 21: Project Parts List

Qty.	Refdes	Part Num.	Description
1	DCPort	Ksmile	Female 2.1x5.5mm DC Power Cable Jack Adaptor
1	PWR50	5V-10A	12~24V/5V DC Regulator, 50W
1	PWR120	EA11703A-120	100~240/12V AC to DC Power Adapter, 120W
2	G2	BIQU	Pulley Wheel w/ Bearing Idler
1	G1	DRILLPROTiming	Timing Pulley Gear + Timing Belt
5	DX1,DX2,DY1,DY2	DRV8825	Stepper Motor Driver Carrier, High Current
4	X1,X2,Y1,Y2	17HM08-1204S	Stepper Motor, NEMA 17, 0.9deg, 3.6V 1.2A
4	MSup1	ST-M1	Stepper Motor Bracket, NEMA 17
2	CPU1,MCU1	3055	Raspberry Pi 3
1	CAM1	3100	Raspberry Pi NOIR Camera Board V2
4	C1	M101	100uF Capacitor, 24V

4.2 Budget

Expanding upon the Parts list above, the Material Budget Information of Table 22 below shows the estimated cost for a prototype including all of these parts.

Table 22: Material Budget Information

Qty.	Part Num.	Description	Cost	Cost
1	Ksmile	Female 2.1x5.5mm DC Power Cable Jack Adaptor	$4.99	$4.99
1	5V-10A	12~24V/5V DC Regulator, 50W	10.95	10.95
1	EA11703A-120	100~240/12V AC to DC Power Adapter, 120W	38.00	38.00
2	BIQU	Pulley Wheel w/ Bearing Idler	10.98	21.96
1	DRILLPROTiming	Timing Pulley Gear + Timing Belt	20.99	20.99
5	DRV8825	Stepper Motor Driver Carrier, High Current	8.74	43.70
4	17HM08-1204S	Stepper Motor, NEMA 17, 0.9deg, 3.6V 1.2A	12.08	48.32
4	ST-M1	Stepper Motor Bracket, NEMA 17	2.63	10.52
2	3055	Raspberry Pi 3	35.00	70.00
1	3100	Raspberry Pi NOIR Camera Board V2	29.95	29.95
4	M101	100uF Capacitor, 24V		
			Total	$299.38

5.0 Project Schedules

5.1 Final Design Gantt Chart

The Final Design Gantt chart of Table 23 below shows the division of work between the group's four members, and their respective areas of work. It also shows a rough timetable of the design process, from a systemic background, and the amount of time spent on the design of each Project B.O.B. subsystem. (DA)

Table 23: Final Design Gantt Chart

Task Name	Duration	Start	Finish	Resource Names
SDP1 fall2017				
Project Design				
Preliminary Design Report				
Problem Statement	21 days	Tue 8/29/17	Mon 9/18/17	
Need	14 days	Tue 8/29/17	Mon 9/11/17	Eric Blok
Objective	14 days	Tue 8/29/17	Mon 9/11/17	Ryan Nowacki
Background	14 days	Tue 8/29/17	Mon 9/11/17	All Members
Marketing Requirements	14 days	Tue 8/29/17	Mon 9/11/17	All Members
Objective Tree	14 days	Tue 8/29/17	Mon 9/11/17	All Members
Preliminary Design Gantt Chart	14 days	Tue 9/5/17	Mon	All Members

			9/18/17	
Block Diagrams Level 0, 1, ... w/ FR tables	**21 days**	**Tue 9/12/17**	**Mon 10/2/17**	
Hardware Modules & FR Tables	14 days	Tue 9/12/17	Mon 9/25/17	Dan Altemese,Eric Blok
Software Modules & FR Tables	14 days	Tue 9/12/17	Mon 9/25/17	Ryan Nowacki,Maram Qurban
Theory of Operation	**22 days**	**Tue 11/7/17**	**Tue 11/28/17**	
Hardware Theory of Operation	5 days	Tue 11/7/17	Sat 11/11/17	Eric Blok
Software Theory of Operation	5 days	Tue 11/7/17	Sat 11/11/17	Ryan Nowacki
Block Diagrams Level 2 w/ FR tables & ToO	**7 days**	**Tue 9/19/17**	**Mon 9/25/17**	
Hardware modules & FR Tables	7 days	Tue 9/19/17	Mon 9/25/17	Dan Altemese,Eric Blok
Software modules & FR Tables	7 days	Tue 9/19/17	Mon 9/25/17	Maram Qurban,Ryan Nowacki
Preliminary Design Presentation 9:55am	1 day	Tue 9/19/17	Tue 9/19/17	
Preliminary Psuedo Code	29 days	Tue 9/19/17	Tue 10/17/17	Ryan Nowacki,Maram Qurban
Midterm Report	**28 days**	**Tue 9/19/17**	**Mon 10/16/17**	
Design Requirements Specification	7 days	Tue 9/19/17	Mon 9/25/17	All Members
Midterm Design Gantt Chart	28 days	Tue 9/19/17	Mon 10/16/17	All Members
Midterm Design Presentation 9:55-11:35am DT06	1 day	Tue 10/24/17	Tue 10/24/17	
Design	**70 days**	**Tue 9/19/17**	**Mon 11/27/17**	
Design Calculations	**70 days**	**Tue 9/19/17**	**Mon 11/27/17**	
Electrical Calculations	**65 days**	**Tue 9/19/17**	**Wed 11/22/17**	
Motor System	59 days	Tue 9/19/17	Thu 11/16/17	Dan Altemese,Eric Blok
Computing	65 days	Tue 9/19/17	Wed 11/22/17	Ryan Nowacki,Maram Qurban
Control Systems	56 days	Tue 9/19/17	Mon 11/13/17	Eric Blok,Dan Altemese
Monitoring System	56 days	Tue	Mon	Maram Qurban,Ryan

		9/19/17	11/13/17	Nowacki
Power, Voltage, Current	49 days	Tue 9/19/17	Mon 11/6/17	Dan Altemese,Eric Blok
Mechanical Calculations	**17 days**	**Mon 11/6/17**	**Wed 11/22/17**	
Structural Considerations	17 days	Mon 11/6/17	Wed 11/22/17	Eric Blok
System Dynamics	65 days	Tue 9/19/17	Wed 11/22/17	Dan Altemese,Eric Blok
Block Diagrams Level 3 w/ FR tables & ToO	**15 days**	**Mon 11/6/17**	**Mon 11/20/17**	
Level 3 Motor & Controller Block Diagram	15 days	Mon 11/6/17	Mon 11/20/17	Dan Altemese,Eric Blok
Software Design	**50 days**	**Mon 10/2/17**	**Mon 11/20/17**	
Modules	**35 days**	**Tue 10/17/17**	**Mon 11/20/17**	
Simulations	43 days	Mon 10/2/17	Mon 11/13/17	Ryan Nowacki
Proof of Concept	29 days	Mon 10/23/17	Mon 11/20/17	Maram Qurban,Ryan Nowacki
Psuedo Code	50 days	Mon 10/2/17	Mon 11/20/17	Maram Qurban,Ryan Nowacki
Hardware Design	**42 days**	**Tue 10/17/17**	**Mon 11/27/17**	
Simulations	42 days	Tue 10/17/17	Mon 11/27/17	Eric Blok
Schematics	**42 days**	**Tue 10/17/17**	**Mon 11/27/17**	
Motor Drive Schematic	15 days	Mon 11/6/17	Mon 11/20/17	Dan Altemese,Eric Blok
Project Poster	7 days	Tue 11/28/17	Mon 12/4/17	
Final Design Report	**42 days**	**Tue 10/17/17**	**Mon 11/27/17**	
Abstract	42 days	Tue 10/17/17	Mon 11/27/17	Dan Altemese
Parts Request Form	42 days	Tue 10/17/17	Mon 11/27/17	All Members
Budget (Estimated)	42 days	Tue 10/17/17	Mon 11/27/17	All Members
Implementation Gantt Chart	42 days	Tue 10/17/17	Mon 11/27/17	All Members
Conclusions and Recommendations	42 days	Tue	Mon	Dan Altemese

		10/17/17	11/27/17	
Final Design Presentations 9:55-11:35am Part 1	1 day	Tue 11/28/17	Tue 11/28/17	
Final Design Presentations 9:55-11:35am Part 2	1 day	Tue 12/5/17	Tue 12/5/17	

5.2 Proposed Implementation Gantt Chart

The Gantt chart of Table 24 below shows a proposed timeframe for the testing, troubleshooting, programming, and construction of Project B.O.B. through the Spring of 2018. It serves to propose a rough schedule for the implementation of the design outlined in this report. (DA)

Table 24: Proposed Implementation Gantt Chart

Task Name	Duration	Start	Finish	Predecessors	Resource Names
SDPII Implementation 2017	**105 days**	**Tue 1/16/18**	**Mon 4/30/18**		
Revise Gantt Chart	14 days	Tue 1/16/18	Mon 1/29/18		All Members
Implement Project Design	**97 days**	**Mon 1/15/18**	**Sat 4/21/18**		
Hardware Implementation	**56 days**	**Mon 1/15/18**	**Sun 3/11/18**		
Breadboard Motor Driver	**13 days**	**Mon 1/15/18**	**Sat 1/27/18**		
Test Controller/Motor Driver	13 days	Mon 1/15/18	Sat 1/27/18		Dan Altemese,Eric Blok
Layout and Generate PCB	14 days	Sun 1/28/18	Sat 2/10/18	6	Dan Altemese,Eric Blok
Assemble Hardware	7 days	Sun 2/11/18	Sat 2/17/18	7	Dan Altemese,Eric Blok
Test Hardware	14 days	Sun 2/18/18	Sat 3/3/18	8	Dan Altemese,Eric Blok
Revise Hardware	14 days	Sun 2/18/18	Sat 3/3/18	8	Dan Altemese,Eric Blok
MIDTERM: Demonstrate Hardware	5 days	Sun 3/4/18	Thu 3/8/18	9	Dan Altemese,Eric Blok
SDC & FA Hardware Approval	0 days	Mon 3/12/18	Mon 3/12/18		
Software Implementation	**56 days**	**Tue 1/16/18**	**Mon 3/12/18**		
Develop Software	**27 days**	**Tue 1/16/18**	**Sun 2/11/18**		
Sensor Software	27 days	Tue 1/16/18	Sun 2/11/18		Maram Qurban,Ryan Nowacki
Control Software	27 days	Tue	Sun		Dan Altemese,Eric

		1/16/18	2/11/18		Blok
Test Software	21 days	Mon 2/12/18	Sun 3/4/18	14	All Members
Revise Software	21 days	Mon 2/12/18	Sun 3/4/18	14	All Members
MIDTERM: Demonstrate Software	5 days	Mon 3/5/18	Fri 3/9/18	18	Maram Qurban,Ryan Nowacki
SDC & FA Software Approval	0 days	Mon 3/12/18	Mon 3/12/18		
Structural Implementation	**15 days**	**Mon 1/15/18**	**Mon 1/29/18**		
Construct Paddle Support System	15 days	Mon 1/15/18	Mon 1/29/18		Dan Altemese,Eric Blok
Construct Motor Timing Belt System	15 days	Mon 1/15/18	Mon 1/29/18		Dan Altemese,Eric Blok
System Integration	**42 days**	**Mon 3/12/18**	**Sun 4/22/18**		
Assemble Complete System	14 days	Mon 3/12/18	Sun 3/25/18		All Members
Test Complete System	21 days	Mon 3/26/18	Sun 4/15/18		All Members
Revise Complete System	21 days	Mon 3/26/18	Sun 4/15/18		All Members
Demonstration of Complete System	7 days	Mon 4/16/18	Sun 4/22/18	27	All Members
Develop Final Report	**105 days**	**Tue 1/16/18**	**Mon 4/30/18**		
Write Final Report	98 days	Tue 1/16/18	Mon 4/23/18		All Members
Submit Final Report	0 days	Mon 4/23/18	Mon 4/23/18	30	All Members
Spring Recess	7 days	Mon 3/26/18	Sun 4/1/18		All Members
Project Demonstration and Presentation	0 days	Mon 4/23/18	Mon 4/23/18		All Members

6.0 Design Team Information

Daniel Altemese	-	Electrical Engineering
Eric Blok	-	Electrical Engineering
Ryan Nowacki	-	Electrical Engineering
Maram Qurban	-	Computer Engineering

7.0 Conclusion

This report has documented all of the steps required to design, and build an autonomous table tennis ball bouncing robot, known as the Ball Oscillating Bouncer, or B.O.B. Using mathematical calculations, simulations, and experimental programming, this design is well validated throughout this report, and should be rather simply implementable. By following the pseudo-code and constructing the schematics and design recommendations of this report, B.O.B. should be able to meet all of the marketing, and design requirements specified in this report. Although future implementation will require lab testing, programming, and overall troubleshooting, all evidence compiled in this report would suggest that this endeavor should be successful. (DA)

8.0 References

[1] R. Silva, F. S. Melo, and M. Veloso, “Towards table tennis with a quadrotor autonomous learning robot and onboard vision,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2015–December, pp. 649–655, 2015.

[2] X. Yu, J. Xu, S. Liu, and K. Chen, “A control system of a ping-pong robot arm based on fuzzy method,” *2011 IEEE Int. Conf. Mechatronics Autom. ICMA 2011*, pp. 398–403, 2011.

[3] X. Chen, Q. Huang, W. Zhang, Z. Yu, R. Li, and P. Lv, “Ping-pong trajectory perception and prediction by a PC based High speed four-camera vision system,” *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*. IEEE, Haidian District, pp. 1087–1092, 2011.

[4] A. W. Smith, Wayne L.; Vesely, Ivan; Gubbels, “THREE-DIMENSIONAL DIGITAL ULTRASOUND TRACKING SYSTEM,” 5515853, 1996.

[5] B. A. Takahashi, Jonathan R.; Harris, “Tracking projectile trajectory with multiple sensors,” 9557405, 2017.

[6] T. Juneau, A. P. Pisam, J. H. Smith, C. Hall, M. Engineering, and U. C. Berkeley, “Dual Axis Operation of a Micromachined Rate Gyroscope,” *1997 Int. Conf. Solid-state Sensors Actuators Chicago, June 16-19, 1997*, pp. 883–886, 1997.

[7] D. H. Zeiner-Gundersen, “Golf training device particularly for the short game,” 6059672A, 2000.

[8] Y. Huang, D. Buchler, O. Koc, B. Scholkopf, and J. Peters, “Jointly learning trajectory generation and hitting point prediction in robot table tennis,” *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids).* IEEE, Cancun, pp. 650-655, 2016.

[9] Rosebrock, Adrian. “Ball Tracking with OpenCV.” *PyImageSearch*, 14 Sept. 2015, www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/.

[10] Adli Haron and K A Ismail, “Coefficient of restitution of sports balls: A normal drop test,” *2012 IOP Conf. Ser.: Mater. Sci. Eng.* 36-012038, 2012.

9.0 Appendix

Specification Sheets & Datasheets

- https://www.omc-stepperonline.com/download/17HM08-1204S.pdf - NEMA 17 Stepper Motor Specification Sheet
- https://www.pololu.com/file/download/drv8825.pdf?file_id=0J590 - DR8825 Stepper Motor Driver Specification Sheet
- http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf - Raspberry Pi 3 Datasheet
- https://github.com/rellimmot/Sony-IMX219-Raspberry-Pi-V2-CMOS/blob/master/RASPBERRY%20PI%20CAMERA%20V2%20DATASHEET%20IMX219PQH5_7.0.0_Datasheet_XXX.PDF - Sony IMX219 Camera Datasheet
- http://resources.mini-box.com/online/PWR-ACDC-12V-10A-120W/PWR-ACDC-12V-10A-120W-specs.pdf - 12V DC Power Adapter Specification Sheet