Ballbot Design Review Report

Rashmi Anil, Ishaan Gupta, Ryan Stentz
Electrical and Computer Engineering, Carnegie Mellon University
{ranil, ishaang, rastentz}@andrew.cmu.edu

Abstract

After a long and tiring session of serving practice, tennis players currently use a ball hoper – a heavy handheld machine - to collect balls. With our new highly improved autonomous tennis assistant, practice is now more efficient and less painful. Ballbot is a robot that uses computer vision to quickly and autonomously collect tennis balls and bring it back to the tennis player. Since Ballbot collects tennis balls as they are being hit, players no longer need to allocate time to clean the court after practice and can instead focusing on improving their techniques.

Index

Index Terms— Autonomous, Ball-hopper, Computer Vision, iRobot, Jetson Nano, OpenCV, Tennis

I. INTRODUCTION

In tennis equipment market, there exist very few advanced electronic devices assisting in the feeding and picking of tennis balls. In fact, most tennis players spend more time picking up balls after playing than the amount of time they spend hitting. Currently the most popular option to pick up balls is the ball hopper. While this method is functional, it takes up time and carrying all of the balls around the court is heavy and cumbersome. With our project, we introduce Ballbot – an autonomous ball boy that collects tennis balls while players are practicing or hitting with a ball machine. Ballbot eliminates the time wasted chasing balls and helps maximize the players' time on court. This tennis assistant allows a tennis player to focus on their technique and skills while completely eliminating the tedious task of gathering the tennis balls.

With Ballbot, tennis practice will be more efficient and less painful. Ballot is an essential tennis assistant that helps the tennis player collect balls during practice. We aim to build a robot that can quickly and autonomously collect tennis balls at an average rate of 6 balls per minute and have a battery life of at least 30 minutes. Our projects scope focuses on serving practice. This is because many balls will be clustered together and the majority of the balls will be on one side of the net. The robot will detect and collect 30 lime-green tennis balls on an debris-free, outdoor hard court with reasonable amounts of daylight.

II. DESIGN REQUIREMENTS

To ensure that the Ballbot is efficient, good at serving its purpose and easy to use, we have several design specifications that are critical to the success of our project. Our high-level user requirements are as follows:

- Quickly and autonomously collect about 30 tennis balls: The average amount of balls that is served before the player rests.
- Collect Balls at an average speed of 6 balls per minute: The average tennis player can serve about 6 balls a minute so we want to be able to pick up at balls at this speed.
- Has a battery life of at least 30 minutes when being used: The average tennis player practices their serve for about 30 minutes so the robot needs to be able to last for at least one practice session.
- Weighs less than 20 pounds: needs to be easy to carry onto and off the court along with all the other tennis equipment.

All of the requirements above ensure that the robot is functional and at least meets the requirements of existing solutions. To test the speed of ball collection and battery, we plan to scatter balls throughout the court and time the robot to see how long it takes for the robot to clear the court and leave the robot running until the battery dies.
Our next set of requirements involve the technical the technical aspects of this project. In terms of software, we want a robot that is:

- Completely Autonomous: This requirement is the most important idea behind the project.
- Can process video frames at at least 10 frames per second: In order to easily track balls
- Has a 0% false positive rate for detecting a ball: robot should not start moving towards a nonexistent ball as this wastes time and energy.
- Have a less than 5% false negative rate for detecting a ball: It is rare that a robot does not go after a ball.

While we understand that achieving a 0% false positive rate for detecting tennis balls is very low, we think that this is achieve achievable due to the way we are constraining this problem. Since we are focusing on serving practice on a debris-free court, there shouldn't be anything aside from tennis balls on the other side of the court and the balls should be

clustered together. And since balls should be the only objects on the court, achieving a 0% false positive rate should not be too bad. In contrast, we are allowing ourselves to have a 5% false negative rate because we understand that our computer vision algorithm is not going to be perfect and we might not be able to identify some objects as balls in some cases (e.g, ball is in a dark corner of the court). We plan on testing these requirements in two different settings: one where we place the robot in a court scattered with many tennis balls and one where we place the robot on a court with no tennis balls. Calculating the number of false positives and false negatives in these settings should give us a good estimate as to what the rates are.
Finally, our hardware requirements are as follows:

- Robot can capture tennis balls within 1 ft from the center of the robot: 1 ft is an ideal number that doesn't make the robot too long and unwieldy but also gives us a big enough radius to collect many balls at once.
- Robot should hold-on to un-launched tennis balls while turning: This is to make sure that we don't loose balls we have in our grasp.

These hardware requirements ensure that the design of our robot is not flawed. That is, we ensure that once a ball is within our "1 ft from the center of the robot" grasp, we do not loose it. We can test that this is the case by manually rotating the robot once it has balls in its grasp. If while turning the balls slip out, then we will know that the design did not meet the requirements.

III. ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

The architecture of the Ballbot consists of 2 major parts, the software doing the compute vision and motor controls, and the hardware responsible for actually collecting tennis balls. These sections can be seen clearly in our system diagram which can be found in the Appendix as figure 1.

Overall, all the software is contained in the Nvidia Jetson Nano. This includes the code for tennis ball tracking, controlling the iRobot Create 2, and controlling the motor speeds of the tennis ball launch system. The iRobot Create 2 is used as the base of the Ballbot and carries the Tennis Ball Transport system, Nvidia Jetson Nano, and other hardware to power all the parts. The tennis ball transport system is the mechanism designed to funnel tennis balls towards two spinning wheels that will launch the tennis ball up a ramp and into a basket on the back of the Ballbot.

The main part of the software is the ball tracking algorithm. The input to this algorithm is image frames received from the Intel RealSense Camera over UART. These frames have RGB values for each pixel as well as depth information. The ball tracking algorithm first reads each input frame and stores it as an array. It then converts each from from the RGB color space to the HSV color space which helps increase accuracy in computer vision applications. The next step is to do color thresholding on each of the channels of the HS color space to detect only tennis balls. This is the part of the algorithm sensitive to fluctuations in lighting. The Ballbot automatically adjusts the parameters for thresholding each time it is turned on so that is can account for new lighting conditions. The thresholding decides which pixels are part of tennis balls and which ones are not.

The output of the thresholding is sent to the noise filtering algorithm. The noise filtering algorithm removes the noise of random pixels that were accidentally classified as being part of a tennis ball. Once the noise is removed, the algorithm is left with groups of pixels it thinks is a tennis ball. Each of the groups in thought of as a tennis ball and the algoithm finds the locations of these detected tennis balls in the frame. Then the algorithm picks which tennis ball to go after and tracks it between frames. This allows the ball tracking algorithm to keep track of the position of the tennis ball the Ballbot needs to collect. This position information is sent to the software controlling the iRobot Create 2 as well as the software controlling the Tennis ball launch control motors.

The iRobot controlling software will use the position of the tennis ball and determine if the iRobot needs to turn or adjust its speed. These commands will then be sent to the iRobot Create 2 over UART so the iRobot will keep adjusting its trajectory until the tennis ball is captured, after which it will go after a new ball. The code controlling the motors to collect tennis balls will slow down the motors when the Ballbot is not near a ball and speed them up when the Ballbot is close to a ball to conserve power.

In the tennis ball transport system, the L298N motor controller is responsible for interfacing with the motors collecting the tennis balls. It receives 12V from the LiPo battery to power the motor and a PWM signal form the Nvidia Jetson Nano determining how fast it should spin the motors. The motor controller uses the power and PWM signal to control two 755 DC brush motors at the front of the Ballbot.

Each motor has a rubber wheel attached to it that can deform to the shape of the tennis ball. The motors are placed in the front of the Ballbot at a distance apart where the two rubber wheels have slightly less than enough space for a tennis ball to pass in between them. The motors spin in opposite directions from each other so that any tennis ball that enters the middle of the 2 rubber wheels will be pulled in by both wheels.

The tennis ball will be sped up to 2.2 m/s from the friction of the rubber wheels. The tennis ball will then travel up the acrylic ramp right behind the wheels and onto the wooden runway on top of the iRobot Create 2. Behind the iRobot will be a basket that the iRobot is dragging along. The basket will have a capacity of 30 tennis balls and will travel on swivel wheels. Once the tennis ball reaches the end of the runway, it will drop into the bsket where all the tennis balls are being collected.

To power the Nvidia Jetson Nano and the motors in the tennis ball transport system, we have a 12V rechargable LiPo Battery. While the battery can be connected directly to the motor controller board for the batteries, the Nvidia Jetson Nano needs 5V for which we have a buck converter. The iRobot Create 2 has its own internal rechargeable battery that is charged using the iRobot home base. The iRobot can automatically navigiate to its charging base.

IV. DESIGN TRADE STUDIES

During the design phase, we explored multiple approaches for both the hardware and software solutions for Ballbot. These different approaches came with their benefits and tradeoffs. Included below is an analysis of these different approaches.

Hardware

A. *Robot Base*

In our final design, we elected to use the iRobot Create 2 base, an off-the-shelf robot base. However, we originally were considering the idea of designing and building our own custom robot base. This approach has the benefit of allowing more control over the mechanical parameters of our robot. This additional control would allow us to meet more ambitious requirements. However, designing and building an entire robot base requires significant experience in mechanical design, a skill which our group lacked. Thus, we opted for a premade robot base.

B. *Tennis Ball Extraction Mechanism*

The tennis ball extraction mechanism is responsible for picking up tennis balls. The current design is detailed in the figure below:

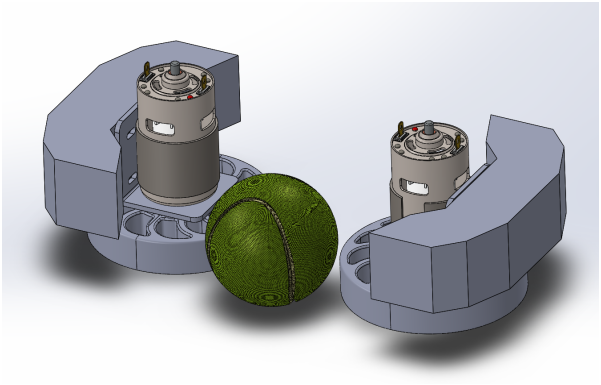


Figure 1: CAD Model of Tennis Ball Extraction System

The above design utilizes two T81 compressible rubber wheels attached to two 755 DC brushed motors. We chose brushed motors over brushless motors for their lower cost. While brushless motors offer better speeds and slightly better power efficiency, they cost around 5-6 times more than their brushed counterparts. Additionally, we determined that with 4 inch wheels, our motors would need to spin at a relatively low RPM 600 to accelerate the tennis balls to 3.11 meters a second. As such, we do not benefit much from a brushless motor's higher RPM and thus chose brushed motors.

For the wheels, we went with T81 compressible rubberized wheels with a 4 inch diameter. We chose rubberized wheels to increase traction with the tennis ball. Also, we wanted the wheels to compress as this would help alleviate strain on the robot frame.

C. *Frame*

The robot frame was designed with a focus on maximizing structural strength and ease of construction while minimizing weight and cost. We decided to use a combination of 1.5"x2.5" wooden planks and 1/8" acrylic for the frame's construction. We decided on 1.5"x2.5" wooden planks for their high structural rigidity and low cost. We chose 1/8" acrylic for the ramp base, ball runway, and robot top plate since it is easy to cut in circular shapes via a laser cutter. We were originally planning on constructing the entire robot frame from 1/8" acrylic, but we deemed that acrylic lacked the required structural strength. Additionally, acrylic is too thin, meaning we would have to build hallow boxes from the acrylic, introducing lots of fragile, weak points in our frame. Thus, we chose to use wood instead of acrylic for the majority of the frame's construction.

Our current frame design is very front-heavy. To mitigate this, we decided to add swivel caster wheels to the end of the frame. We were originally planning on adding counter weights to the back side of the robot, but this approach would add more weight overall. As a consequence, the robot would likely slower and more sluggishly, so we decided that adding wheels for support to the front side was a better option.

D. *Robot Arms*

The robot's arms exist at the very front of the robot with the goal of funneling tennis balls towards the ball launching mechanism. These arms were designed to maximize the number of tennis balls that can be captured without changing the robot's direction. Without robot arms, we would need higher accuracy in our tennis ball tracker since there would be very little room for error. The current design sports 1.5"x2.5" wooden planks protruding from the robot at a 45 degree angle. We choose a 45 degree angle to simplify the construction of the robot, as this angle is fairly easy to cut from wood.

E. *Camera*

The camera is the most important sensor for Ballbot. We use the camera to detect and track balls for collection. If a tennis ball is out of view of our camera or appears too noisy to be correctly identified, then Ballbot will miss it. As such, it is important that our camera have a wide field of view and sufficiently high resolution. With these requirements in mind, we choose the Intel RealSense Depth Camera D435. This camera records at 1080p and has an over 90 degree field of view. Additionally, this camera has depth-sensing capabilities which can be used as an additional input to our ball tracking algorithm. Before deciding on the D435, our group was considering using the Raspberry Pi Camera V2. This camera also supports recording at 1080p, but has a significantly smaller field of view. Additionally, the Raspberry Pi Camera V2 does not have hardware support for automatic exposure control, while the D435 supports this feature.

However, the D435 is an expensive camera priced at $180, so we also considered borrowing an iPhone 11 Pro camera. This camera is capable of recording at 1080p with a 120 degree field of view. However, this solution requires desiging a custom camera mount that's easily detachable. Additionally, we would need custom software to extract video from the iPhone for processing on the Jetson Nano. When finalizing our parts list, our TA offered to lend us their D435 camera, so its cost was no longer a concern. Thus, we chose the D435.

F. *Computer*
We are currently developing our computer vision algorithms on a NVIDIA Jetson Nano. The Jetson Nano contains a quad core 1.5 GHz processor and a dedicated 128 CUDA core GPU. We considered using the Raspberry Pi 4 as a cheaper solution, but we feared it would not meet our compute requirements. The Raspberry Pi contains a similar CPU, but lacks a GPU. Since we plan on running computer vision algorithms in real time, we would benefit greatly from having a dedicated GPU. Thus, we selected the Jetson Nano as our onboard computer.

Software

A. *Tennis Ball Tracking*
When designing the ball tracking algorithm, we emphasized speed and accuracy to meet our overall design requirements. Our current tennis ball tracking algorithm utilizes computer vision color thresholding to reduce an image to only the pixels belonging to a tennis ball. This algorithm is relatively simple to implement, and as such, runs relatively quickly. The major drawback to this algorithm is its sensitivity to lighting changes.

We also explored more sophisticated algorithms for ball tracking, like a using a deep convolutional neural network. These types of networks, while very accurate and robust, perform considerable more calculations on the input. As a consequence, they may overwhelm the computing capabilities of our Jetson Nano. Additionally, these networks require large amounts of data to achieve high accuracy on general inputs. This amount of data is hard to collect given the current circumstances of the world.

B. *Robot and Motor Control*
All of Ballbot's movements are categorized into two types controls: the iRobot Create 2 controls and the front motor controls. The iRobot Create 2 accepts controls via commands sent over UART. The NVIDIA Jetson Nano can connect to the UART interface via a special USB serial cable. With this setup, we can use the NVIDIA Jetson Nano to issue all iRobot Create 2 controls. The two 755 DC motors on the front of the robot are controlled by L298N motor controllers. These controllers accept PWM signals to adjust the motor speed. We considered adding another microcontroller to handle PWM generation, but after a bit of research, we discovered that the NVIDIA Jetson Nano is capable of outputting PWM signals on its GPIO pins.

V. SYSTEM DESCRIPTION

As shown in the system block diagram which is figure 2 in the appendix, our system consists of two major subsystems: the software used for tennis ball tracking, controlling the iRobot's movement, and controlling the tennis ball collection motor speeds as well the hardware responsible for collecting the balls into the basket behind the iRobot.

A. *Intel RealSense Depth Camera*
During our design process, we considered several cameras before deciding to use the Intel RealSense Depth Camera D435. It was important that our camera have a wide field of view and sufficiently high resolution. This is because without a wide field of view, the Ballbot would miss balls that were close to it but slightly out of the field of view of the camera. Also, without a high resolution, there would be much more noise in the computer vision algorithm, causing our false positive and false negative rates to increase. This camera records at 1080p and has an over 90-degree field of view so it would meet our needs for having high quality data as input to the computer vision algorithm. Also, we chose the Intel RealSense Depth Camera because we needed our ball detection to run quickly at 10 frames per second so the camera needed to be able to record at least that many frames per second. The RealSense camera can record at 90 FPS which not only meets our needs, but goes beyond them, giving us more data to improve the accuracy of tracking the tennis balls between frames. What really sets the RealSense camera aprat from other camers is its depth sensing capabilities. The depth map of each pixel as an additional input to the path planning algorithm. We are planning on using this depth information to find the closest ball to the Ballbot and go after it to optimize the path. Without the depth information, we would have to rely on how large the ball is in out field of view which would fluctuate a lot depending on noise and slight changes in lighting.

B. *Ball Tracking Algorithm*
The Ball tracking algorithm reads images from the Intel Realsense Depth Camera and extracts the location of the nearest tennis ball. To accomplish this the read images go through the following pipeline:

1. *HSV Color Thresholding*
The image read from the Intel RealSense Camera is an RBG image which isn't suitable for image processing. We initially tried to perform color thresholdingg on the RGB image but realized that it was difficult to narrow down the green channel of the image to only allow pixels that have the specific green color of tennis balls. To fix this, we converted the image from the camera to the HSV color space using OpenCV. This is because the HSV color space abstracts hue(the color) into one channel so it is easy to locate the range of hues that fall into the green that tennis balls usually have. We predetermined ranges for each of the 3 channels, hue(H), saturation(S) and Brightness(V)

that allowed only pixels with the color of a tennis ball to pass through the filter. To do this, we manually adjusted the ranges for each of the channels until we saw that the filter blocked out all the pixels not part of a tennis ball.

However, we found that at different times of the day, these ranges need slight adjustments based on the current lighting. We noticed that usually adjusting the range for the Brightness channel fixed any issues due to lightning conditions. To solve this, we added a calibration step to the Ballbot where on startup, we would place a tennis ball in front of it and it would auto adjust the range of the brightness channel until the tennis ball was being detected. We determined that a tennis ball was being detected if an approximately circular region of pixels was making it through the filter. Once the HSV color thresholding was applied, we essentially had an array where each pixel was either blocked or allowed through by the filter.

2. *Noise Filtering*
   After our color thresholding algorithm was working reasonably well, we saw that pixels that were part of the tennis ball were almost always being let through by the filter, but so were other random pixels in different parts of the image. To fix this, we decided to add noise filtering that would essentially blur the signal. To accomplish this, we first eroded the image using OpenCV which shrinks each group of pixels slightly. By doing this, random pixels that got past the thresholding disappear since they shrink down to 0 pixels while groups of pixels that represented tennis balls remain since there were many pixels together. Then we dilate the image which brings any cluster of pixels remaining back to its original size. By doing this process, we lose some of the granularity of the image, but since we only care about the positions of each ball, this is not a big issue. The benefit or removing the noisy pixels is well worth cost of blurring the image.
3. *Tracking Between Frames*
   Once the noise is removed, we have large groups of pixels for each tennis ball. The center of each of these groups is found using OpevCv's moments function. Then we use the RealSense depth camera to determine which of the groups of pixels is closest to the camera. The closest one represents the closest tennis ball and is the one the Ballbot will go after. We find the center corresponding to this group of pixels and assume that is the location of the center of the tennis ball the Ballot needs to collect. The depth of the tennis ball is sent to the motor controlling code and the location of the center of the tennis ball is sent to the iRobot control system. If at this point no balls are seen in the frame, then both pieces of code receive "None" as the position of the tennis ball.

C. *iRobot Create 2 Control System*
The code controlling the iRobot Create 2 receives as input the position of the center in the image read of the tennis ball the Ballbot needs to go after. The code employs a simple algorithm of determining how far from the center of the picture the tennis ball is located. If the tennis ball is to the right of the center line, the algorithm sends the command to turn right to the iRobot. The amount to turn is proportional to the difference between the center line and the tennis ball position. The speed of the iRobot is set to be inversely proportional to the difference so that the Ballbot moves faster when the tennis ball is closely aligned with the center of the Ballbot. If the control system receives as input that no balls are found, it will tell the Ballbot to rotate in place until a ball can be seen and the the input will no longer be None.

D. *Tennis Ball Collection Motor Control*
The tennis ball collection and motor control system is core to the design of this project. Our design utilizes two T81 compressible rubber wheels attached to two 755 DC brushed motors. We chose brushed motors over brush-less motors for their lower cost. While brush less motors offer better speeds and slightly better power efficiency, they cost around 5-6 times more than their brushed counterparts. For the motors to successfully launch the tennis ball into the basket, they needed to propel the tennis balls at a speed of 2.2m/s. This is because the motors launch the tennis ball up a 45 degree ramp. Once the ball leaves the ramp, it will follow projectile motion until it lands in the basket. While the basket is between 2 and 3 feet away from the motors in our design, we did the calculations for launching the ball 1 meter away. This way, we know if we can hit our target speed, the ball can land in the basket since we can always slow down the motors, but there is a limit to how fast we can speed them up. To calculate what the initial x and y velocities in terms of time, we did:

$$y = y_0 + v_y t - \frac{1}{2} g t^2 \qquad x = x_0 + v_x t$$
$$0 = v_y t - \frac{1}{2} g t^2 \qquad 1 = v_x t$$
$$0 = t(v_y - \frac{1}{2} g t) \qquad v_x = \frac{1}{t}$$
$$v_y = \frac{1}{2} g t$$

Since the ball was going up a 45 degree ramp, the initial x and y velocities would be the same. Then to calculate the time it would take for the ball to land in the basket, we did:

$$\frac{1}{2} g t = \frac{1}{t}$$
$$t^2 = \frac{2}{g}$$
$$t = 0.4517 \text{ seconds}$$

Finally, to solve for the initial x and y velocities, we substituted the value for t and got $v_x = 2.2$m/s and

$v_y = 2.2$m/s. Combining these, we get that the initial velocity has to be $\sqrt{2.2^2 + 2.2^2} = 3.11$m/s to launch the ball successfully into the basket. While we did not account for air resistance, we saw that it did not have a significant effect on the calculations and we compensated for any slowdown by aiming to launch the ball further away than we needed to.

Once we found the required launch speed of the tennis ball, we converted that to how fast the wheels on our motors need to spin. Since we had rubber wheels with 4 inch ≈ 0.1016 meter diameters, the calculations for RPM were as follows:

$$\frac{3.11}{2\pi r} \cdot 60 = \frac{3.11}{0.3192} \cdot 60 = 584.586 \text{ RPM}$$

The 755 DC motors provided us with a low enough RPM to make the wheels spin at around 600 RPM which should launch the tennis ball into the basket.

For the wheels, we chose compressible runner wheels with a 4 inch diameter. It was important that the wheels were compressible so that we could alleviate strain on the robot frame. Additionally, the rubber helped with the grip by increasing traction while collecting the tennis balls. See Figure 1 for a visualization of how the motors and the wheels interact to collect tennis balls.

E. *iRobot Create 2 Base*

The iRobot Create 2 base is responsible for the movement of the Ballbot. We decided to use the iRobot Create 2 instead of creating or own base because it extrapolated the movement of the robot to simply writing the correct values to a serial port. The Create 2 Open Interface provides a guide on how to interface with the Create 2. We can send a unique opcode for each command, followed by parameters for the command. For example, to drive backwards at a speed of 200 mm/s with a turn radius of 500 mm, we could send 137 (the opcode for the drive command), 255 (the upper byte of the velocity), 56 (the lower byte of the velocity), 1 (the upper byte of the turn radius, and 244 (the lower byte of the turn radius). Through this interface, we can easily control the Create 2 base by sending commands over UART.

F. *Power*

We have two major components that need power on Ballbot: the 755 DC motors and the Jetson Nano. The Jetson Nano requires 5 volts for its input, and the 755 DC motors require around 12 volts. We wanted to use a single power source, so we picked a higher voltage battery, and lowered its voltage using a buck converter for the Jetson Nano.

Combined, the two 755 DC motors and Jetson Nano consume 35 watts of power under full load. To meet our battery life requirement, we choose a 11.1 volt 2200 mAh LiPo battery which provides power to a 35 watt load for around 40 minutes.

The iRobot Create 2 has its own power source and does not require additional power. The battery in the iRobot create can last up to 2 hours.

G. *Tennis Ball Transport System*

The tennis ball transport system is the mechanical system designed to capture tennis balls from the tennis court and transport them to the basket at the back of the Ballbot. A CAD modle of the system using the Create 2 as a base can be seen below:

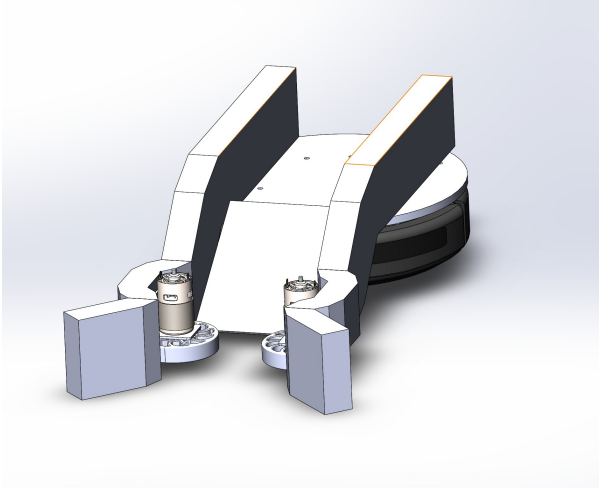


Figure 2: CAD Model of Tennis Ball Launch System

The tennis ball transport system is broken down into four major components:

1. *Robot Arms*
   The robot arms protrude outward at a 45 degree angle. These arms were designed to funnel balls towards the propulsion mechanism while the robot moves in a straight line. There are two swivel caster wheels (not shown) attached to the bottom of each arm for structural support.
2. *Tennis Ball Propulsion*
   The propulsion mechanism is responsible for accelerating tennis balls. It accomplishes this through the use of two rotating rubber wheels. These wheels are driven by two 755 DC brushed motors spinning around 600 rotations per minute. The motors are mounted on a custom cut wooden frame made from 1.5”x2.5” wooden planks.
3. *Tennis Ball Ramp & Runway*
   The tennis ball ramp and runway are responsible for guiding the tennis balls up the robot base and into the basket behind the robot. The ramp is set at a 45 degree angle, which is steep enough to quickly deflect the ball and shallow enough that the ball stays on the ramp. The ramp and runway railings are made from 1.5”x2.5” wood, and their surfaces from 1/8” acrylic.
4. *Tennis Ball Basket (not shown)*
   The tennis ball basket receives and stores tennis balls from the runway. It rests on a free-moving platform pulled by the robot.

VI. PROJECT MANAGEMENT

Over the course of the semester, we have 3 major milestones – establish proof of concept, basic integration and full implementation. The first three weeks of the semester were spent finalizing a design and establishing a proof of concept. During this phase, the group spent some time learning about the iRobot create and Jetson Nano and how to interface with it. A basic ball propulsion mechanism was also built to ensure that tennis balls could be picked up and launched into a basket with two motors and wheels. The second milestone is the basic integration. In this phase, the robot will be assembled and some basic movement will be programmed. In the final implementation phase, computer vision will be used to detect tennis balls and the robot will be programmed to move towards and collect tennis balls.

Team Member Responsibilities
While all the members of the team are responsible for driving the project forward and making sure that the team does not fall behind schedule, we each have our own independent responsibilities to ensure that everyone have enough work to do. We split up the project into three major parts: constructing the robot, programming the iRobot and using OpenCV to do the computer vision aspects of this project. The table below details the primary and secondary responsibilities of each team member.

Team Member	Primary Responsibility	Secondary Responsibility
Rashmi	Program the iRobot create to respond to the outputs from the computer vision algorithm and plan the path.	Help measure and cut parts needed for mechanism that will collect tennis balls.
Ishaan	Designing a computer vision algorithm to detect and track tennis balls.	Integrate iRobot control system, computer vision, and motor control into one software component.
Ryan	Designing and constructing a mechanism to collect tennis balls on the ground and store it in a basket.	Help develop computer vision and aid in testing all parts of the Ballbot

Budget
The budget table is Figure 3 in the Appendix Section. It is located on Page 6. The hardware tools that were used were: the iRobot Create Base, 2 RS555 Motors to launch the tennis balls, L298N motor driver, Intel RealSense Depth Camera D435i, buck converter, 12V Lipo rechargable battery, Nvidia Jetson Nano. For software we will be using the OpenCV and pycreate2.

Risk Management
Even though we are just a few weeks into building the robot, we have already encountered many risks with regards to the design of the project. One of the major difficulties that we encountered was using acrylic to build the exterior of the robot. While this would have given the exterior a sleek finish, cutting necessary shapes proved to be much more difficult than anticipated. Additionally, we realized that if we used acrylic, the robot might not be as sturdy and robust as we want. To mitigate against this risk, we decided to use wood instead of acrylic. We had to redesign and simplify several parts of the robot to make the construction easier. However, since wood is much heavier than acrylic, we now needed to make sure that the pieces to build the frame and the ramp of the robot was not too heavy and that robot could still satisfy our speed metrics. This meant we had conservative while redesigning the exterior. Although the wood was heavier, it made our robot more robust and more likely to survive a hit from a tennis ball.
Secondly, with regards to the computer vision aspects of this project, we realized that detecting the tennis balls using OpenCV, we had to re-threshold the colors every time we decided to test. This was because the lighting and time of day the testing was being done was different each time. To mitigate against this risk, we decided to add an algorithm that auto-thresholds the color based on the given lighting. Currently, as part of the initial set-up process, we need to hold a tennis ball to the camera so the thresholding can be done and the robot can be calibrated. The future goal is to be able to store some color data for a tennis ball so that the user does not need to hold the ball to the camera for calibration.
In terms of the schedule, we had about 8 weeks from when all of our parts arrived to finish building the robot. This was because the majority of our team was going home for Thanksgiving and not returning. This meant that our schedule was packed during this time. We had to ensure that our MVP was working by the end of these 8 weeks. We didn't want to leave one team member in charge of trying to get everything to work after Thanksgiving. To mitigate this risk, we had internal weekly team goals and met up at least 3 times a week to ensure that we were on track. Everyone kept a log of what they were working on and asked for help when it was required. While each of us had a specific area that we focused our energy towards, we all worked together helping each other when it was necessary.

APPENDIX

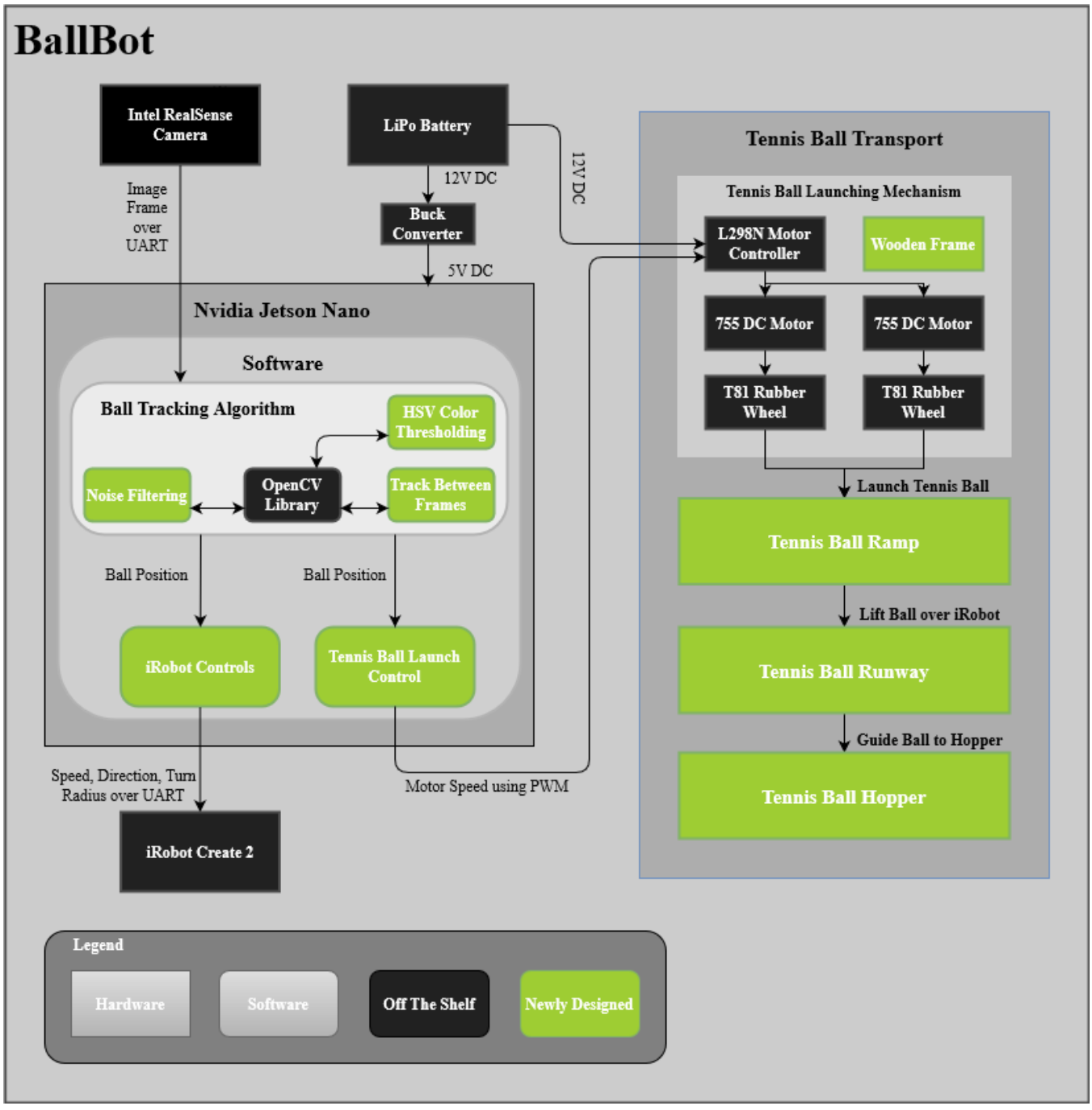
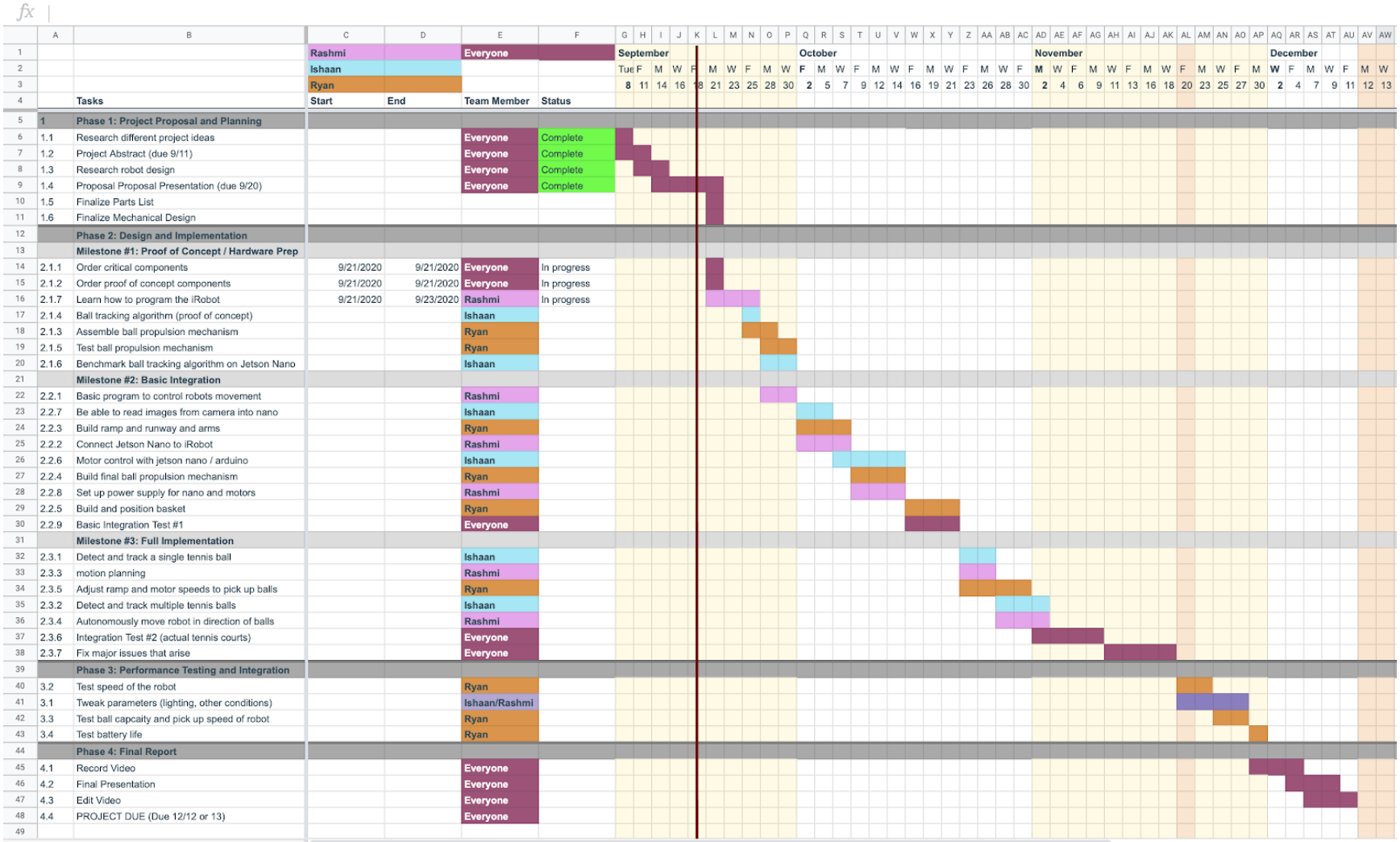


Figure 3: System Block Diagram



	Tasks	Start	End	Team Member	Status
1	**Phase 1: Project Proposal and Planning**				
1.1	Research different project ideas			Everyone	Complete
1.2	Project Abstract (due 9/11)			Everyone	Complete
1.3	Research robot design			Everyone	Complete
1.4	Proposal Proposal Presentation (due 9/20)			Everyone	Complete
1.5	Finalize Parts List				
1.6	Finalize Mechanical Design				
	Phase 2: Design and Implementation				
	Milestone #1: Proof of Concept / Hardware Prep				
2.1.1	Order critical components	9/21/2020	9/21/2020	Everyone	In progress
2.1.2	Order proof of concept components	9/21/2020	9/21/2020	Everyone	In progress
2.1.7	Learn how to program the iRobot	9/21/2020	9/23/2020	Rashmi	In progress
2.1.4	Ball tracking algorithm (proof of concept)			Ishaan	
2.1.3	Assemble ball propulsion mechanism			Ryan	
2.1.5	Test ball propulsion mechanism			Ryan	
2.1.6	Benchmark ball tracking algorithm on Jetson Nano			Ishaan	
	Milestone #2: Basic Integration				
2.2.1	Basic program to control robots movement			Rashmi	
2.2.7	Be able to read images from camera into nano			Ishaan	
2.2.3	Build ramp and runway and arms			Ryan	
2.2.2	Connect Jetson Nano to iRobot			Rashmi	
2.2.6	Motor control with jetson nano / arduino			Ishaan	
2.2.4	Build final ball propulsion mechanism			Ryan	
2.2.8	Set up power supply for nano and motors			Rashmi	
2.2.5	Build and position basket			Ryan	
2.2.9	Basic Integration Test #1			Everyone	
	Milestone #3: Full Implementation				
2.3.1	Detect and track a single tennis ball			Ishaan	
2.3.3	motion planning			Rashmi	
2.3.5	Adjust ramp and motor speeds to pick up balls			Ryan	
2.3.2	Detect and track multiple tennis balls			Ishaan	
2.3.4	Autonomously move robot in direction of balls			Rashmi	
2.3.6	Integration Test #2 (actual tennis courts)			Everyone	
2.3.7	Fix major issues that arise			Everyone	
	Phase 3: Performance Testing and Integration				
3.2	Test speed of the robot			Ryan	
3.1	Tweak parameters (lighting, other conditions)			Ishaan/Rashmi	
3.3	Test ball capcaity and pick up speed of robot			Ryan	
3.4	Test battery life			Ryan	
	Phase 4: Final Report				
4.1	Record Video			Everyone	
4.2	Final Presentation			Everyone	
4.3	Edit Video			Everyone	
4.4	PROJECT DUE (Due 12/12 or 13)			Everyone	

Figure 4: Milestone and Schedule Chart

Name	Source	Unit Cost	Quanity	Total		Budget
iRobot Base	iRobot website	$200.00	1	$200.00		$600.00
12V, 12000 RPM DC Brush Motor	Amazon	$20.00	3	$60.00		$400.00
L298N Motor Driver Controller Board	Amazon	$10.00	1	$10.00		$340.00
Intel RealSense Depth Camera D435i	Borrowed from TA	BORROWED	1	$0.00		$330.00
Nvidia Jetson Nano	Nvidia online store	$100.00	1	$100.00		$330.00
Acrylic	Amazon	$50.00	1	$50.00		$230.00
Swivel Wheels	Amazon	$7	1	$7.00		$180.00
11.1V, 2200mAh Lipo Battery	Amazon	$20	1	$20		$173.00
12v to 5v Buck Converter	Amazon	$10	1	$10		$153.00
Mounting Bracket for motor	Amazon	$9	2	$18		$143.00
Wheels to pick up tennis ball	Banebots	$3.50	2	$7		$125.00
Hubs for wheels	Banebots	$3.15	2	$6.30		$118.00
Extra wheels to pick up tennis ball	Banebots	$3.50	2	$7.00		$111.70
Wood Planks	Already Owned	Already Owned	4	$0.00		$104.70
					Remaining Budget	**$104.70**

Figure 5: Budget and Parts List