Final Report:

Autonomous Air Hockey System

Team AH2

Edward Bear

Elexander Fryer

Dylan Hastings

Christian Steinmetz

Avery White

1 Problem Statement

The primary objective of this project was to build an autonomous air hockey system that has the ability to play a game of air hockey against itself, a robot, or a human player. The system is comprised of multiple subsystems that work together as a whole to achieve the final goal.

This problem can be decomposed into two broad engineering challenges. The first revolves around determining current and future puck locations, and generating the optimal location to move the paddle to in order to effectively block shots taken by the opponent. This is achieved by a computer vision system that provides the current puck location, a predictive algorithm that determines future puck locations, and gameplay logic that produces the path of the paddle paddle based on the predicted puck trajectory. The second engineering challenge revolves around building a mechanical system that translates the intent of the paddle control algorithm into movement of the paddle in the physical world. This required the use of motors, motor controllers, and the associated inverse kinematics that translate desired paddle XY coordinates into the proper motor angles.

Without both of these systems operating at an appropriate level of performance, the entire system will not effectively perform its task of playing and winning a game of air hockey against its opponent.

2 Design Objectives

2.1 Performance Goals

Our goal was to design a rotary-rotary arm system with the ability to cover the majority of our side of the playing field. We focused on constructing an arm that would be able to move fast enough to block a moderately quick incoming puck. The ability to block an incoming puck is dependant on its speed as well as the top speed of the motors and the latency in the vision and gameplay logic subsystems. For this reason, motors were selected so as to produce enough torque to move the motors rapidly, and care was taken in the design of the vision and gameplay logic subsystems to minimize latency with multithreading. Due to the powerful nature of the mechanical system, we were also concerned with implementing protections to limit erratic and unnecessarily powerful behavior in the arm system. Therefore one of the greatest challenges stemming from our performance goals was to build a fast and powerful system that is still safe to operate and does not damage itself in operation.

2.2 Design Principles

A rotary-rotary design was employed for the arm since this design closely resembles a human arm and provides easily interpretable kinematics with basic DC motors. In order to meet our goal of building a fast moving system to block incoming pucks, we reduced the overall mass of the arm assembly as

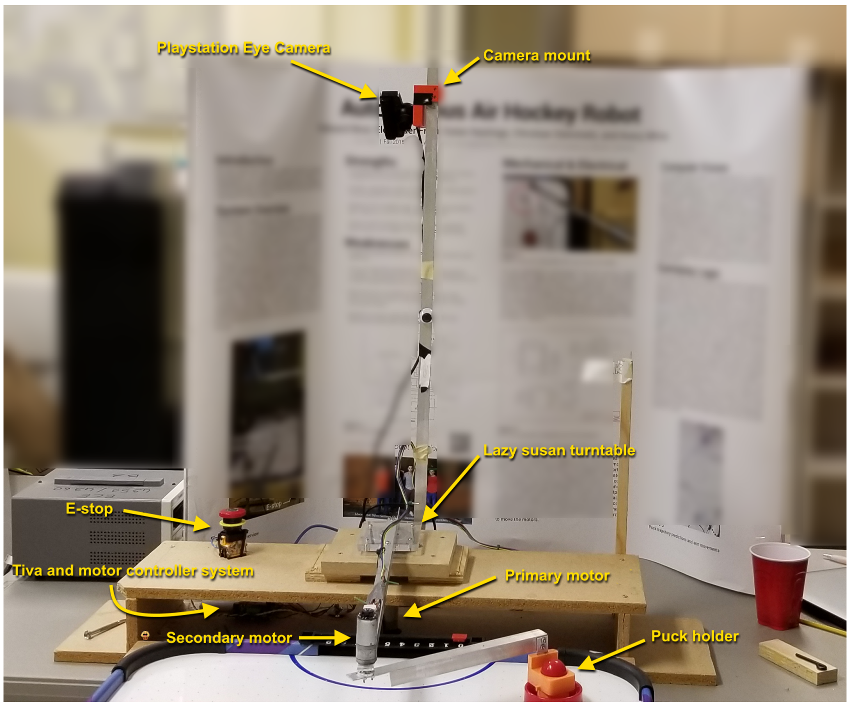

Figure 1: Entire system showing the main parts that make up the system as a whole.

much as possible in order to maximize the achievable speed. This was achieved by using lightweight aluminum for the construction of the arm along with laser-cut acrylic. Wooden materials were used to construct other portions of the system, such as the arm assembly enclosure and base, to reduce cost in areas where weight was less of a concern. We focused on using off-the-shelf metal hardware for better rigidity and durability in comparison to 3D printed parts, which tend to be more fragile. 3D printed parts were utilized for specialized connections such as a paddle holder to connect with the arm assembly and the PlayStation Eye camera mount since they would not be subject to an undue amount of mechanical stress. We chose a camera placement that maximized our view of the table so as to begin tracking the puck as soon as possible. This design choice kept the camera out of the way of other systems and made its position permanent even during table changes. An overview of the entire system with various parts labelled is shown in Figure 1.

3 Final Design

3.1 Mechanical Subsystem

The mechanical subsystem consists of the arm assembly, DC motors with rotary encoders, flex-coupler, mounting hubs, and turntable. The arm assembly consists of two separate sections of aluminum, which provide a mechanically sound, yet lightweight system for the rotary-rotary arm assembly.

The main arm section is 46.75 cm in length and is mounted to the turntable bearing system shown in Figure 2. The turntable bearing system serves to mechanically couple the main DC motor to the main arm section, while reducing off-axis forces on the motor shaft. The turntable is mounted to a wooden base structure, which houses the main DC motor and the accompanying motor control electronics. The aluminum arm is connected to a laser-cut piece of acrylic that connects to another piece of laser-cut piece of acrylic. This lower piece of acrylic has a 6 mm mounting hub attached, which facilitates the connection to the flex-coupler and finally the main DC motor. This design mitigates potential off-axis forces on the main DC motor and allows for smooth movement of the main arm.

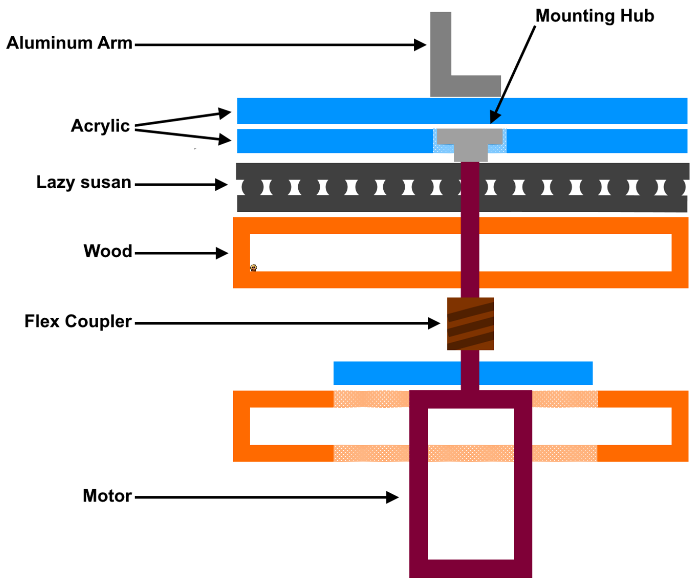

Figure 2: Diagram of the main motor mounting scheme.

The secondary arm section is 24.25 cm in length, shorter than the main arm section. The secondary motor is mounted to the main arm and the secondary arm is attached directly to the motor with a 4 mm mounting hub. At the end of the secondary arm is a custom-designed, 3D printed paddle holder that allows the paddle to float, allowing vertical movement within the holder to facilitate small differences in the height of table throughout the playing field. Both the primary and secondary arm sections are shown in Figure 3, along with the 3D printed paddle holder.

The two DC motors used in the design were selected based on calculations from an experiment which determined the required RPM to block incoming pucks of various speeds, presented in more detail in the Preliminary Report. Based on these calculations, motor speeds of at least 82 RPM are required to block a fast moving puck. The main motor is the most powerful with a stall torque of 40 kg-cm, no-load RPM of 146, and a 51:1 gearing ratio. The secondary motor is smaller (to achieve a low arm assembly weight) and has a stall torque of 12 kg-cm, no-load RPM of 350, and a 34:1 gearing ratio.

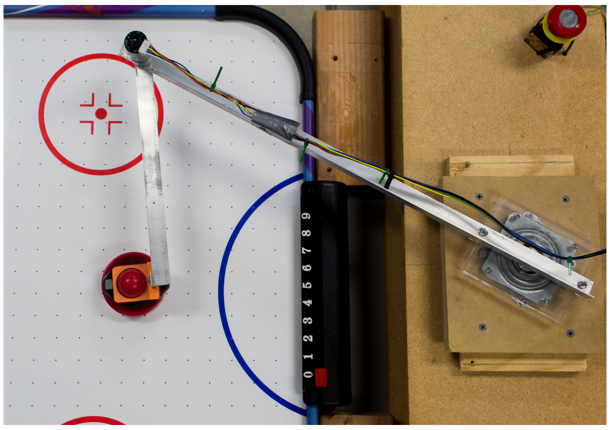

Figure 3: Custom fabricated 3D printed paddle holder provides precise control of the paddle position.

3.2 Control Subsystem

The control subsystem translates XY paddle positions on the air hockey table to actuation of the motors so that the the paddle will move to the specified position on the table. The control subsystem includes the Tiva, the dual channel H-bridge motor driver, the UDP communication link between the Tiva and the C++ vision and gameplay program, and the motor rotary encoders. An overview of the motor control subsystem is shown in Figure 4.

The UDP communication link between the Tiva and the computer provides a means to transmit the results from the computer vision subsystem to the Tiva. The Tiva then sends the appropriate control signals to the motor controller in order to apply the correct voltage and polarity to the two DC motors.

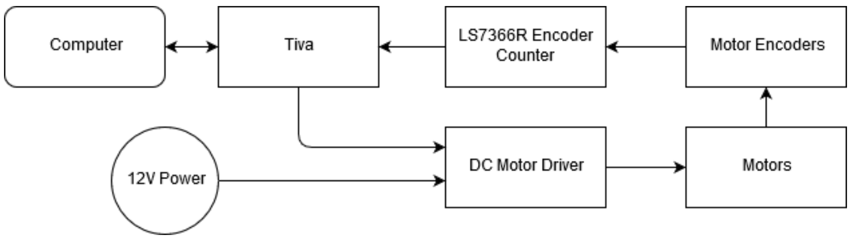

Figure 4: Control subsystem block diagram.

In order to control the motors, a closed-loop system is implemented using two proportional controllers in the Tiva. The rotary encoders attached to each motor produce feedback signals that are read by the LS3766R counters. The Tiva then reads the value of these counter over the SPI bus, and the counts per step factor for each motor is used to determine the current angle of the motors. These angles are used to calculate an error signal used in the proportional controllers. This control signal is then analyzed to produce a PWM output from the Tiva for each motor, along with a direction signal.

The dual channel motor controller receives this PWM and direction signal for each motor from the Tiva. These signals are used to generate the voltages sent to the motors, sourced from the power supply.

3.3 Computer Vision Subsystem

The vision subsystem consists of the PSEye camera attached to an aluminum pole adjusted to a height in which the whole hockey rink can be seen. Additionally, included in the vision subsystem are the portions of the code regarding the homography, tracking of the puck's position in the frame, and the conversion of this position from pixel coordinates to centimeter coordinates.

For the homography, four hard-coded, tested points in the camera's frame were used to perform the homography. Included in the homography portion of the vision subsystem are a few other hard-coded points that were place on the frame in locations that were designed to match the red dots on the hockey table. The purpose of painting these other points is that, for the tuned homography, we required a way to align the position of the physical system after each run of the program, so that the coordinate system was consistent between runs and between air hockey tables.

In order to track a puck, a few features were introduced. The ability to threshold the resulting homography frame was required and this was achieved by including a slider bar GUI that allowed us to change the minimum and maximum threshold values for Hue, Saturation, and Value. Tuning these values allowed us to optimize the system for the green color of the puck and any change in lighting conditions. After applying the HSV thresholds, the vision subsystem performs two erodes and dilates to remove noise present in the image. After creating the thresholded image we utilized the functionality of moments to determine the XY position of the puck's centroid and the area of puck. It should be noted that we kept track of the area of the puck so that we would not track it if the puck was in frame but vertically above the table by roughly 1 ft, as it would have a significantly larger area at this point.

The last piece of functionality that the vision subsystem is responsible for is the conversion of the XY pixel coordinates into a centimeter based coordinate system. This was done by dividing the puck's location by a pixel per centimeter conversion factor and assigning that result to a global variable that the other threads in the system could use.

3.4 Gameplay Logic Subsystem

The gameplay logic subsystem translates information about the current and past locations of the puck into movement of the arm that produces effective air hockey gameplay, such as defending oncoming shots, scoring goals, and freeing the puck from the corner for example. One of the key parts of this system is the trajectory prediction algorithm.

A C++ class implements this functionality in a class called Puck. This class encapsulates all of the information regarding the puck in real-time such as its radius, location, velocity, and future trajectory. It also contains methods that compute the velocity of the puck given past points and compute future locations of the puck based on this velocity. An instance of this object is instantiated as a global variable in the main program and a multithreaded architecture is utilized to separate the operation of the vision system, the trajectory predictions, and the gameplay logic.

Every four frames the vision thread will pass on the past four locations of the puck for the trajectory thread to calculate a new velocity and trajectory. The velocity is calculated by performing a least square regression fit on these puck position points. This information is then stored in the Puck object, along with the latest puck position, in order to calculate a new trajectory. Trajectories are calculated by stepping the puck from its current location based on the calculated velocity, assuming a constant acceleration. The gameplay thread is then constantly looking at the latest trajectory stored in the Puck object in order to determine how it should respond. This was somewhat challenging to implement since there were three separate threads attempting to read and write to the object during run-time. We were able to solve this problem using mutex within the getter and setter methods of the Puck class. The puck trajectory and arm movement paths are shown in real-time on the GUI as shown in Figure 5.

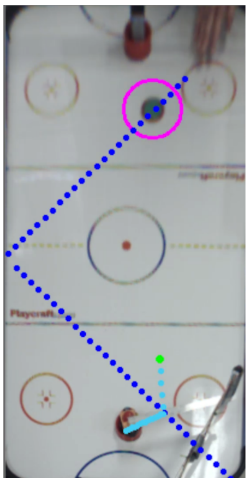

Figure 5: Puck trajectory (dark blue) and arm movements (light blue) updated in real-time display.

There are five main states of gameplay that can be executed at any given time based on the location and velocity of the puck. The first is the block and hit mode which occurs when a quick moving, oncoming puck is detected. The latest trajectory will be used to find a point on our side of the playing field to intercept the puck and then move the paddle towards the goal in an effort to score.

The next is follow and hit which occurs when there is a low velocity puck on our side of the playing field. In this case the arm is moved to a location directly behind the puck and the paddle will move through the puck in an attempt to clear it from the zone. Special checks are implemented before every step of the arm movement path is sent to the Tiva over UDP to ensure that the set coordinate is not too close to any of the walls or within the goal zone. There is also additional checking that stops the movement of the arm if the puck is positioned in front of the goal and behind the paddle in an effort to reduce self scoring.

If the puck is outside of our zone, not detected on the table, or moving out of our side of the playing field, the paddle with return to its home position, which is directly in front of the goal zone. This helps to reduce potentially erratic movements if the arm has to move long distances from one side of the playing field to the other in a short period of time and also ensure the goal is covered. The final case is utilized when a puck becomes stuck in either corner of the board. In this case a specialized circular arm path will be generated which will smoothly free the puck from the corner.

In order to ensure that the system is adaptive while in operation, so it does not act on a piece of information and continue with improper behavior if the state of the puck changes, during each step of the execution of an arm movement path, the velocity of the puck is checked against the value used for the calculation of the current arm. If these values differ by the set threshold the arm will abandon its current arm movement and recalculate a new arm movement.

4 Cost Accounting

Development Cost	$302.40
Out-of-Pocket Development Cost	$274.40
Cost of Artifact	$273.40

Table 1: Cost analysis overview

The Development cost of this project is based on everything that was purchased including the estimated cost of donated items, such as a replacement secondary motor, and wood. The Out of Pocket cost is based on everything we thought we would use, even those that we ended up taking off the final project, but excluding the things we borrowed or were given to us. The Final Artifact cost is based off of every item that we used on the final project, excluding the items we took off, and only considering the items once, if they broke. All these estimations are based off Table 4 in Appendix A.

5 Performance Characterization

5.1 Methods and Results

The final homography points that were needed for an accurate coordinate system, given the camera's orientation is fixed, can be found in Table 2. The way that these points were found was by using the homography clicking feature that was provided in the demonstration code to give points and a homography image that we could qualitatively determine its validity by looking at the frame's encapsulation of the rink. We first looked at whether the frame captured more of the table or not enough of the table. It was noticed that there was skewing in the image due to the homography points, which led to inconsistency in the X or Y values when there should not be. For example, the value of the X coordinate should be constant when moved along the walls towards and away from the camera, and the Y should be constant when moving parallel to the camera's field of view. The homography points were adjusted in an iterative process until these the change present in values read from the vision subsystem were below 2 pixels of variation.

Point Number	Pixel Row Location	Pixel Column Location
1	159	342
2	152	146
3	535	3
4	554	475

Table 2: Homography coordinates

To translate the Row and Column location of the puck's location into a XY coordinate system that is measured in centimeters, a 3.04762 pixels/cm conversion factor was used for the X values and a 2.985 pixels/cm conversion factor for the y values. To find these conversion values we had to take into account the actual dimensions of the board (66 cm x 134 cm) and the radius of the puck (3.15 cm). In addition, we had to have the XY pixel locations of the puck against each wall. From these calculated conversion factors they were incrementally adjusted until the puck's location on each wall was +/- 3.15 cm (puck radius) as appropriate within 1 cm of error.

Another area of the project that required significant tuning and measure was determining the optimal gain values for the two PID controllers implemented in the Tiva. After initial trial and error testing, proportional and derivative values were determined. These were tuned based on a tracking of trajectories, wherein the Tiva would receive a series of set points along the desired movement path of the arm. With our initial proportional and derivative gains, the arm often experienced undesirable oscillations. Therefore we significantly lowered these gains for the first two milestones. Afterwards we tested the accuracy of the arm set points using a custom cm spaced grid we created and found that there was roughly 3-4 cm of error in the arm movement. Using this grid paper for further tuning of our gain values, we managed to obtain a 1 cm radius of error in the system after increasing the proportional gain value to 10 and eliminating the derivative gain altogether. This resulted in a trade-off between arm oscillations and arm accuracy.

Since our ability to block and hit a block relied upon moving to a given location in a certain, limited amount of time, it was necessary to understand the timing of the UDP communication link between the C++ program and the Tiva. When moving the arm to a given location we needed to determine how many steps to take in order to reach the location in a set amount of time. To determine this we performed an experiment to observe the time required to send and receive a single packet to the Tiva. The results of this experiment are shown in Table 3. We found the time per step by using the time.h library, clock(), and CLOCKS_PER_SEC values and functions to find the difference between the start time and end time of a single transmission. This information was then used in the code to determine, dynamically how many steps were required to reach the given location in a given amount of time.

Steps	Seconds	Seconds/step	ms/step
125	0.253	0.002024	2.024
250	0.502	0.002008	2.008
500	1.002	0.002007	2.007
1000	2.007	0.002004	2.004
2500	5.010	0.002004	2.004
5000	10.033	0.002007	2.007

Table 3: UDP transmission loop timing

Even though the arm was tracking a trajectory, fed as separate set points to the Tiva over time, there is still potential for erratic behavior if too few steps are used in an arm movement of a given distance. When this occurs the arm will overshoot, and if near a wall, will likely collide with the wall with great force. To help counteract this phenomenon, we included a safety in the code that checks that the distance per step of a particular path is beneath a set threshold. If the distance per step is too great the arm will not move. The distance per step threshold that was set 0.2 after a number of trails with the am. The way that we tested this value was by observing the aggressiveness of the arm when moving it back and forth between points on the table, and lowering the threshold if it collided with the walls.

5.2 Discussion

As discussed in the preliminary report, one of our specifications that had to be tested for was the paddle hitting the wall. This was apparent since the very beginning of the project, wherein the arm would slam into the walls fairly frequently. This was due to overshoot present in the system whenever not enough steps or points were used to travel to the set point. However, utilizing the distance per step threshold as discussed above, we were able to effectively make the system less erratic in these cases. A dynamic path mode was also implemented which based the number of steps in the path on the distance the arm was to travel. It should be stated as well, the data collected regarding the time per step was what enabled us to create this feature. Although we were able to almost eliminate the system slamming into the walls of the hockey rink, we reduced the systems aggressiveness so much that it could not hit the puck fast, which was one of our systems weakest points in regards to the competition.

Another key component that we thought needed to be tested in the preliminary report was the accuracy of the vision and control subsystems. Although we did not test the system in the same way as specified in the preliminary report, the system as a whole fulfilled our requirement to have an accurate system since we attempted to mitigate errors by reducing them to +/- 1 cm in both the vision and control subsystems. Although this is a relatively small of amount of error given the dimensions of the table are 66 cm x 134 cm, this did turn out to be rather significant given how precise the system needs to be to accurately hit the puck to a target location. Ultimately, the errors in the system after extensive tuning were low enough so as to block incoming pucks of moderate speed but not so low as to allow the system to accurately hit a puck to various target locations. For that reason there was a lot of stochasticity in regards to the gameplay performance, in that sometimes the arm movement was effective in hitting the puck as desired and at other times it was not with little consistency.

6 Project Postmortem

6.1 Technical

Late into the development of the project we discovered a flaw in our mechanical design that led to the failure of our secondary motor. The secondary motor was mounted directly to the secondary arm section with no mechanical relief (i.e. a flex coupler). This was done to save weight since we believed there would be no major issues with this configuration. This worked well until we decided to add additional weight to the paddle in order to keep it from "jumping" over the puck. This created additional off-axis forces on the secondary motor and eventually led to it's failure right before the first competition. We disassembled the motor gearbox and discovered that a tooth broke off inside.

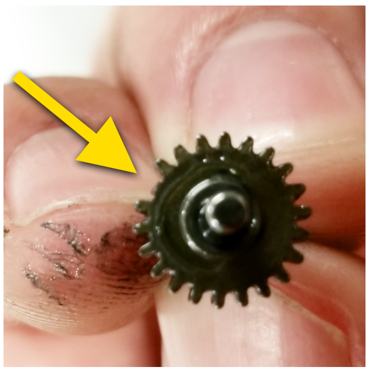

Figure 6: Broken gear tooth from the secondary motor gearbox.

Another aspect of the project that we would revisit with additional time was the camera pole mount. Each arm movement the entire system would vibrate causing the camera to move. This vibration would in turn affect the vision subsystem distorting the velocity and trajectory calculations momentarily. While we did not observe many issues with this, we believe it could improve performance. Simply using a more rigid material to support the camera, like wood, would most likely solve the issue.

Later in the testing of our system we also discovered that some errors in blocking a puck resulted from the reality that the device to hold our paddle did not place the center of the paddle at the center of the secondary arm. This subtle offset meant that as the paddle was moved around the table the actual location of where the puck would meet that paddle changed, creating a complex system, resulting in less than desirable performance in some cases. This could be remedied in software but it would be much easier to redesign the 3D printed paddle holder in order to place the center of the paddle at the same center as the arm itself.

Most likely the largest challenge we faced was in designing the high-level gameplay logic and state machine. This involved wiring together all of our subsystems to create a single cohesive system that was good at playing air hockey. At first we experienced fairly good results when implementing parts like combining the vision system with our block and hit algorithm. Things became more challenging as we added in more states, and therefore more complex overall behavior. The addition of cases for freeing pucks from the corner as well as trying to hit slow moving pucks in our zone had the effect of ruining the entire performance of the system since small changes in the puck locations or inconsistencies in the playing field field itself left the system constantly switching through states and ultimately doing nothing.

We attempted to remedy this by constantly rechecking the status and breaking out of operations if they were no longer valid, but we often ran into false negatives, wherein the system would stop its motion even though it should have continued. We spent a lot of time tuning these thresholds but ultimately our final performance was less than desired. The complexity of our gameplay logic ultimately led to this low performance but we simply needed additional time to rework our logic so as to utilize the tools we had built more effectively. Our strengths lie in the vision system and the arm movement methods we have developed that allow us to move the arm in linear, and curved paths, and build compound paths that include both. Our electrical and mechanical system are solid and provide a good foundation for our gameplay logic to operate effectively given the aforementioned improvements.

6.2 Nontechnical

Christian and Dylan focused their efforts on code development. They worked continuously on further developments in an effort to improve the effectiveness of the robot's actions. Christian focused on trajectory predictions, arm movement, and gameplay logic. Dylan focused on vision system and homography, GUI, and gameplay logic. Dylan also was responsible for submitting all the reports as the point of contact. Eddie, Avery, and Elex participated in brainstorming of code, and helping to development aspects of the robot's character, along with Dylan and Christian. Eddie was also the designer of all 3D printing parts needed for the robot and also handle PID modifications that needed to be made. Avery designed the robot's physical build with input from Dylan and Eddie. Improvements were made by Avery throughout the semester, including spacers, and flags. Elex built the air hockey robot's arm with assistance from Avery. Christian, with assistance from Avery and Eddie, wired the machine's electrical system. Elex and Avery provided assistance with testing the system, which led to improvements in the code.

Since we all had varying levels of availability, we did not have a designated time to show up to work on the robot. Whenever one of us was free, we were expected to go spend some time working on the robot, or helping the others that were there. Eddie had the most free time, so he was there every day after noon when he got off work. Dylan and Christian were there about the same, and usually met together to brainstorm or discuss improvements to the code. Christian quickly became the leader of the group, and led discussions and developed plans to guide the team.

7 Highlights

Adaptive trajectory calculations and arm logic using multithreaded architecture

This is explained in 3.4 Gameplay Logic Subsystem, and was likely the most complex part of the system, and also very powerful. We found early on that due to inconsistencies and non-idealities in the air hockey tables our initial trajectory predictions were often incorrect. With this multithreaded approach we were able to continually update and make improvements to our predictions in real-time and translate this to continuous updates in the movement of the arm.

Dynamic arm path movements and safety protections

Explained in 5.1 Methods and Results, this functionality was developed as a way to ensure more consistent and safe operation of the arm in our dynamic system. We implemented functionality in order to not execute arm movements that would result in erratic behavior with the creation of a "distance per step" metric and after significant tuning we chose a value to ensure safe operation.

System adaptability and alignment

Since we knew the system needed to move to different locations we focused on building a system that would have this ability. This includes the adaptability of the vision system described in 3.3 Computer Vision Subsystem, the parameterization of the inverse kinematics, and the physical spacer system and tuned homography mentioned in 5.1 Methods and Results.

A Parts List

Item description	Part Number	Vendor	Qty	Cost	Type	Final Design
DC Motor Driver	DRI0018	DFRobot	1	$44.50	Purchased	Yes
Large 12V DC motor	FIT0277	DFRobot	1	$47.90	Purchased	Yes
Small 12V DC motor	FIT0493	DFRobot	2	$58.00	Purchased	Yes
Flex-coupler	RB-SCT-1156	DFRobot	2	$9.98	Purchased	Yes
Aluminum angles	55970	Lowe’s	1	$14.48	Purchased	Yes
4’x8’ MDF wood	37461	Lowe’s	1	$31.95	Donated	Yes
3x3” ball bearing turntable	RB-SCT-1084	RobotShop	1	$4.95	Purchased	Yes
6 mm mounting hubs	RB-POL-136	RobotShop	2	$15.90	Purchased	Yes
4 mm mounting hubs	RB-Nex-89	RobotShop	2	$13.80	Purchased	Yes
Screws	#4-40UNC	Amazon	220	$19.99	Purchased	Yes
D-shafts	40225	Amazon	6	$12.95	Purchased	Yes
Zip ties			60	$6.00	Purchased	Yes
Wiring			25	$5.00	Donated	Yes
LS7366R		Dr. Groff	2	$2.00	Donated	Yes
Acrylic			1	$10.00	Donated	Yes
E-stop			1	$5.00	Donated	Yes

Table 4: Complete project bill of materials