## horizontal line



Final Technical Report

Robotic Air Hockey System  
2018.08.13

David Eelman - 6365316  
Stanislav Rashevskyi - 7028178  
Thomas Abdallah - 7141518

Conestoga College

Electronic Systems Engineering  
Capstone Project II - EECE74135

Semester 8, Class of 2018

Table of Contents

[horizontal line 0](#_Toc521934232)

[Abstract 3](#_Toc521934233)

[1 Introduction 3](#_Toc521934234)

[1.1 Background 3](#_Toc521934235)

[1.2 System Description 3](#_Toc521934236)

[1.3 Literature Review 4](#_Toc521934237)

[1.4 Proposed Solution and Rationale 5](#_Toc521934238)

[2 Design and Test Methods 5](#_Toc521934239)

[2.1 Paddle Controller 5](#_Toc521934240)

[2.2 Master Controller 9](#_Toc521934241)

[2.3 Puck Tracker 11](#_Toc521934242)

[2.4 User Interface 12](#_Toc521934243)

[2.5 Test Methods and Plan 15](#_Toc521934244)

[System Test Plan 15](#_Toc521934245)

[Paddle Controller 15](#_Toc521934246)

[Master Controller 18](#_Toc521934247)

[Puck Tracker 19](#_Toc521934248)

[User Interface 19](#_Toc521934249)

[2.6 Implementation 20](#_Toc521934250)

[2.7 Safety and Impact on Environment and Society 22](#_Toc521934251)

[3 Results and Analysis 23](#_Toc521934252)

[3.1 Qualitative Analysis 23](#_Toc521934253)

[3.2 Measurement Results and Quantitative Analysis 23](#_Toc521934254)

[4 Conclusion and Recommendations for Further Work 24](#_Toc521934255)

[5 Appendices 25](#_Toc521934256)

[Appendix A – Coordinate System 25](#_Toc521934257)

[Appendix B – UI Images 26](#_Toc521934258)

[Appendix C – State Machines 28](#_Toc521934259)

[Appendix D – Microcontroller Interface Circuit Schematics & PCB Layout 29](#_Toc521934260)

[Appendix E – Goal Light Circuit Schematic & PCB Layout 36](#_Toc521934261)

[Appendix F – Data Sheets 38](#_Toc521934262)

[Appendix G – CAD Models 44](#_Toc521934263)

[Appendix H - Risk Analysis 47](#_Toc521934264)

[6 References and Bibliography 48](#_Toc521934265)

# Abstract

This document will detail the technical details of the engineering design, integration, and validation of the robotic air hockey system developed for the Capstone design project course during semester 7 and semester 8 of the Electronic Systems Engineering program at Conestoga College.

# 1 Introduction

## 1.1 Background

The Robotic Air Hockey System is capable of autonomously playing air hockey against a human player. This project attempts to address the problem of a lack of public knowledge about the ESE program. This problem needed to be solved as there is limited marketing material available that showcases the technical knowledge and capabilities taught in the ESE program that can be understood by both technical and non-technical audiences. Our project has allowed group members to develop industry relevant technologies while applying them to a fun and interactive game that adds value to the ESE program through public demonstrations.

## 1.2 System Description

The Robotic Air Hockey System is comprised of two major devices, a robotic system, and the supervisory controller system (Figure 1). The robotic system (referred to as the Paddle Controller) is an embedded system responsible for controlling the movements of the air hockey robot based on commands received over CAN. The supervisory controller system is comprised of three software components, each running as a separate process on a single computer, communicating with each other using inter-process communication.

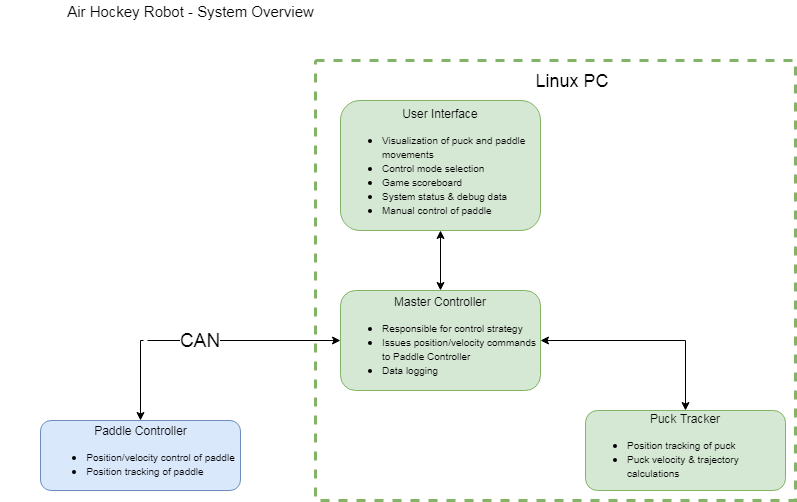


Figure 1 - System Block Diagram

## 1.3 Literature Review

The Robotic Air Hockey System relies on successfully integrating several unique technologies into one holistic system. This section will review literature covering examples of air hockey playing robots, methods for tracking objects in motion, and mechanisms for high speed position control.

Existing Air Hockey Robots

This project was inspired by an open-source project built by jjrobots [1] [2]. The jjrobots design uses an Arduino to drive three stepper motors from a 3D printer to control the movements of the air hockey paddle. In the version 1 of the jjrobots design, a PC running OpenCV is used to track the positions of the puck and paddle which are sent to the Arduino with a serial connection. In version 2 of the design, an Android smartphone running a custom application was used to track the puck and paddle positions while also providing a user interface for the system. The jjrobots system implemented three distinct control strategies: defense, offense, and offense + defense. Our project implemented a simple defense strategy at first, and more advanced strategies once the core platform was stabilized. Our proposed project was inspired by the jjrobots design, but our implementation is our own work with the detailed system design being driven by system level performance requirements.

Methods for Tracking Objects in Motion

This project requires accurate real-time position tracking of the air hockey puck in order to calculate its trajectory and velocity. There are a multitude of methods that can be used to track the position of an object. Table 1 summarizes several technologies that were considered for this project. Computer vision was used in the final project to achieve position tracking for the puck.

Table 1 - Technologies to Track Objects in Motion

|  |  |  |  |
| --- | --- | --- | --- |
| Technology | Description | Pros | Cons |
| Magnets [3] | Array of sensors directed toward moving magnetic object. | * Low cost * Low power | * Inaccurate measurements * High complexity to track small object on large surface |
| Computer Vision [4] [5] [6] | Camera images processed by a computer to locate specific shapes or colours. | * Open-source solutions available * Accurate tracking of objects in motion | * Requires large amount of processing power * Susceptible to changes in lighting conditions |
| Ultrasonic Sensors [7] | Array of sensors directed toward moving object. | * Low cost * Readily available | * High complexity to track small object on large surface |

Mechanisms for High Speed Position Control

The air hockey paddle needs to move very quickly in order to compete with the best human players. Research into high speed linear position control mechanisms was conducted in order to prepare for the detailed design phase of the project (Table 2). System level performance targets drove the design of the final system which used a series of belt-driven linear actuators.

Table 2 - Technologies for High Speed Position Control

|  |  |  |  |
| --- | --- | --- | --- |
| Technology | Application Examples | Pros | Cons |
| Belt Drive [1] [2] | * Air hockey robot (jjrobots) * 3D printers | * Low cost * Scales to larger/smaller travel distances * Off the shelf components available | * High speed rotating components * Moderate mechanical complexity * Low mass |
| Linear Electric Motors [8] | * Maglev trains * High speed pick and place machines | * High speed * Few moving parts | * Custom designed * High cost * High mass |
| Linear Actuators [9] [10]  (Electromechanical, pneumatic, hydraulic) | * Industrial automation * Heavy equipment * Active suspension | * Few moving parts * High force output | * High cost * Fixed stroke lengths * High mass |

## 1.4 Proposed Solution and Rationale

This project improves upon an existing open-source design that was implemented using hardware and software from a 3D printer [1]. The proposed system utilizes the open-source computer vision library OpenCV to track the position of the puck on the playing surface and calculate its speed and trajectory. A mechanical system capable of moving an air hockey paddle in two-dimensions was implemented that can accelerate the paddle to speeds high enough to defend against pucks travelling at speeds up to 10m/s and attack the puck by colliding with it while in motion. The electrical system to control the movement of the air hockey paddle was implemented using a combination of off-the-shelf hardware and custom designed electronics. The software to control the system is our own. A user interface was implemented to allow the system to be demonstrated to both technical and non-technical audiences. Thorough documentation and a debugging interface was developed to allow for the project to continue functioning in the future.

# 2 Design and Test Methods

## 2.1 Paddle Controller

The Paddle Controller is comprised of three major components, an electro-mechanical system, embedded hardware that interfaces with the electro-mechanical system and the microcontroller, and embedded software running on the microcontroller.

**Electro-Mechanical System**

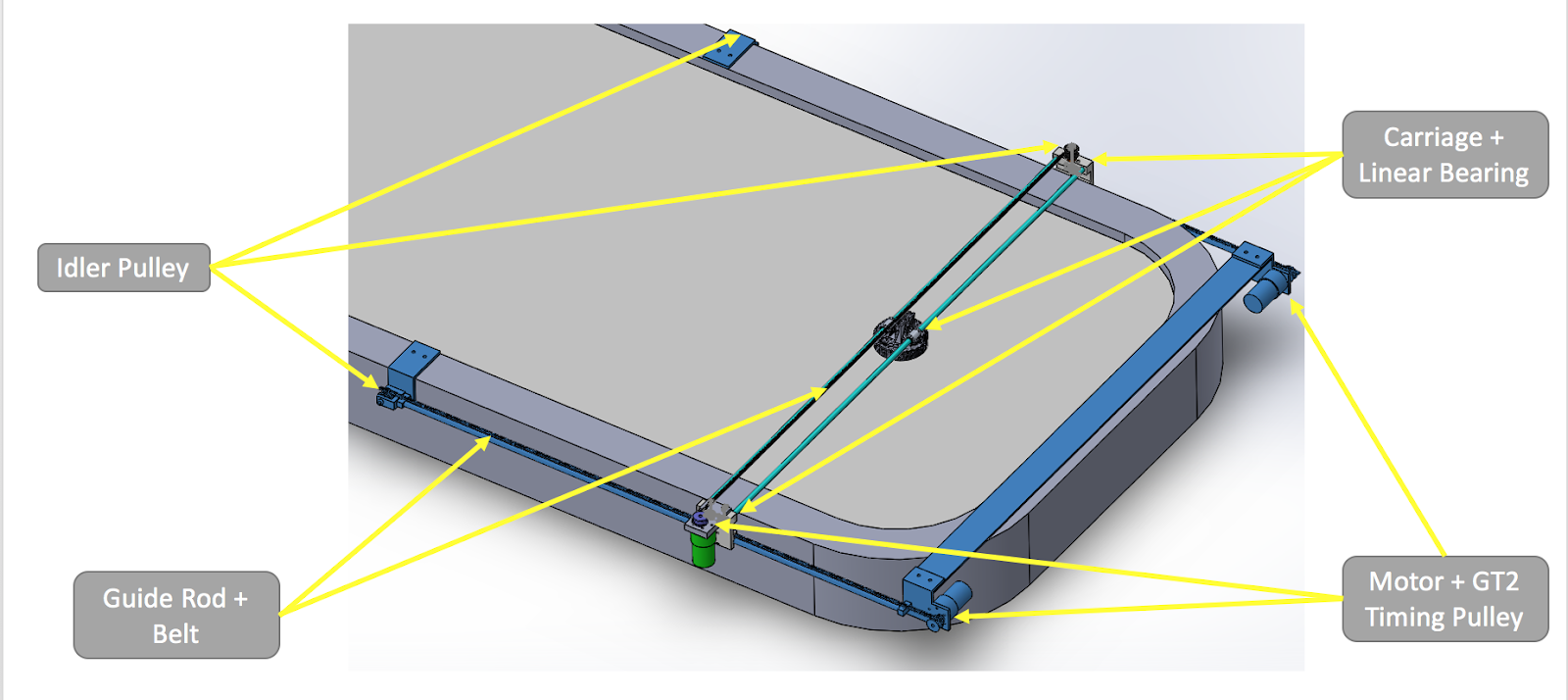


Figure 2 - Electro-Mechanical System Major Components

The electro-mechanical system (Figure 2) is capable of moving the air hockey paddle in two-axes (Figure 14 - Coordinate System). A single brushed DC motor drives a timing belt to move the air hockey paddle left and right in the X-Axis. Two brushed DC motors driving timing belts are used to move the entire X-Axis assembly forward and backward in the Y-Axis.

A quadrature encoder is mounted to the output shaft of each motor enabling accurate control of speed and position for each motor.

Limit switches are mounted at the end of travel on both ends of each of the three linear rails. If the robot travels outside of the playable area and strikes any limit switch, power to all three motors is shut off to stop the robot from moving any further.

A home position switch (simple micro-switch) is mounted on each of the three linear rails allowing the microcontroller to locate a known “home” position after a reset or during a calibration routine.

Infrared break-beam sensors are installed around the perimeter of the electro-mechanical system to detect intrusions into the robots area of movement. If any of the four break-beam sensors is tripped power to all three motors is shut off to stop the robot from moving any further.

An emergency-stop is installed on the human side of the air hockey table. When pressed, power to all three motors is shut off to stop the robot form moving any further.

The CAD models for the mechanical system are included as attachments with the submission of this report.

**Embedded Hardware**

A custom circuit board serves as the interface between the electro-mechanical system and an off-the-shelf microcontroller (Figure 3). Schematics for the circuit board and data sheets for the major components can be found at the end of this document.

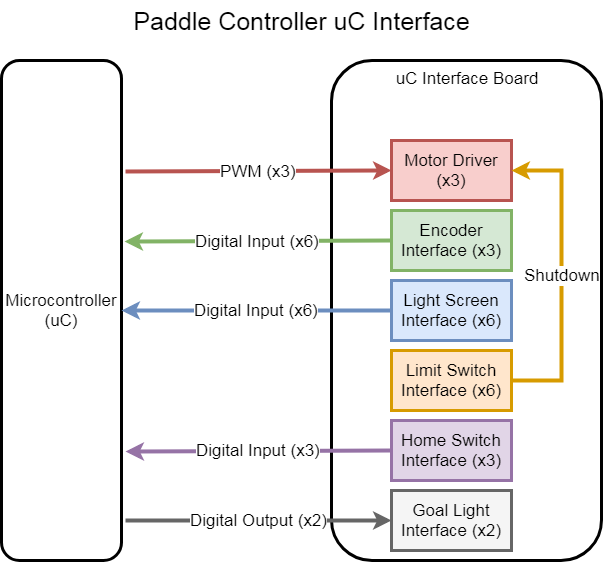


Figure 3 - Microcontroller Interface Board

The circuit board includes three motor driver circuits allowing each brushed DC motor to be controlled using a single PWM signal from the microcontroller.

Both phases of each of the three quadrature encoders are filtered using a Schmidt trigger. The “A-phase” of each encoder triggers an interrupt on the microcontroller where the “B-phase” of the corresponding encoder is read to determine the direction of rotation for the motor. The “B-phase” of each quadrature encoder is latched on the rising edge of its corresponding “A-Phase” using a D-Flip-Flop to ensure that its value does not change prior to being read during the interrupt-service-routine on the microcontroller.

The infrared (IR) “light screens” (or break-beam sensors) are comprised of an IR LED and an IR sensor that is sensitive to 56 kHz pulses of IR light. The IR LED’s are driven by 555 timers operating in a stable mode outputting a 56 kHz waveform. The IR output of the sensors are filtered using a Schmidt trigger and passed to the microcontroller for processing.

The six limit switch outputs are combined using logical AND-gate, with the output driving the “shutdown” pin on each of the motor driver integrated circuits.

The home switch outputs are filtered using a Schmidt trigger and passed to the microcontroller for processing.

Additional circuitry is included to drive two LED “goal lights” which may be implemented in the future.

**Embedded Software**

An off-the-shelf microcontroller running software written in C is used to interface with the motors and sensors via the embedded hardware. The embedded software communicates with the Master Controller using CAN, transmitting information about the air hockey paddle position, velocity, state, and any error information. The Master Controller sends commands to the embedded software to control the position and velocity of the air hockey paddle, as well as issuing state machine commands (Figure 4).

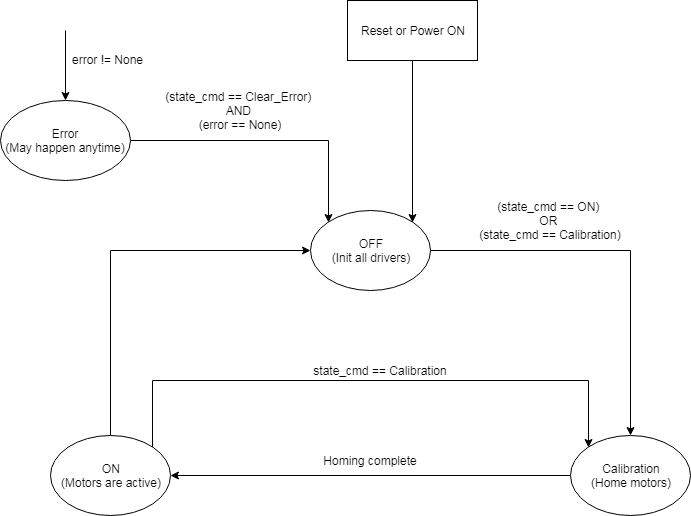


Figure 4 - Paddle Controller State Machine

## 2.2 Master Controller

../5_Master_Controller/MC_flowchart.png

Figure 5 - Master Controller Supervisory flowchart

The Master Controller (MC) is responsible for game strategy control, trajectory predictions, data logging and supervisory control. MC process is written in Python 2.7 and ran on off-the-shelf Linux Computer (i7, 16Gb RAM, Ubuntu 14.04).

Figure 5 demonstrates the constant MC program flow and is described below:

**Rx IPC and Tx IPC data**

MC establishes inter-process communication with the Puck Tracker (PT) and the User Interface (UI) processes through Python process-based “threading” interface called *multiprocessing*.

Using multiprocessing, MC creates two arrays (multiprocessing.Array together with enumerations specified in a JSON file) and a queue (multiprocessing.Queue) for communication with the PT.

PT transmit array is mainly used to identify the velocity and position of the puck on the playing surface, as well as to show current states and errors of the process. Receive array is used to configure the web-camera and change PC states. Visualization receive queue is used to pass camera video stream to the MC.

Using multiprocessing, MC creates two arrays (multiprocessing.Array together with enumerations specified in a JSON file) and a queue (multiprocessing.Queue) for communication with the UI. UI transmit array is mainly used to identify current menu page, and send data relevant to that page (e.g. diagnostics interface, manual game paddle positions, settings values, etc), as well as to show current states and errors of the process. Receive array is used to change UI states, list current states and errors of other modules, send scored goals, etc. Visualization transmit queue is used to pass camera video stream with drawn game strategy decisions (puck trajectory predictions, paddle movement, etc.) to the UI.

**Rx CAN and Tx CAN data**

Master Controller communicates with the Paddle Controller board using CAN bus, receiving information about the air hockey paddle position, velocity, goals, state, and any error information. The Master Controller transmits commands to the embedded software to control the position and velocity of the air hockey paddle, as well as issuing state machine commands (Figure 4).

**Log Data**

All received and transmitted messages are stored in HDF5 file and analyzed in Matlab. Debug information is stored in a text file using Python *logging* module.

**Analyze Rx data and Make decision**

In data analysis, MC tracks states, errors, quit requests, game modes, and settings. Then MC makes decisions: calculates paddle position for the automated game; handles manual game (paddle position is filtered and sent from the UI touch display); updates settings; calibrates camera; resolves errors and quit requests, etc.

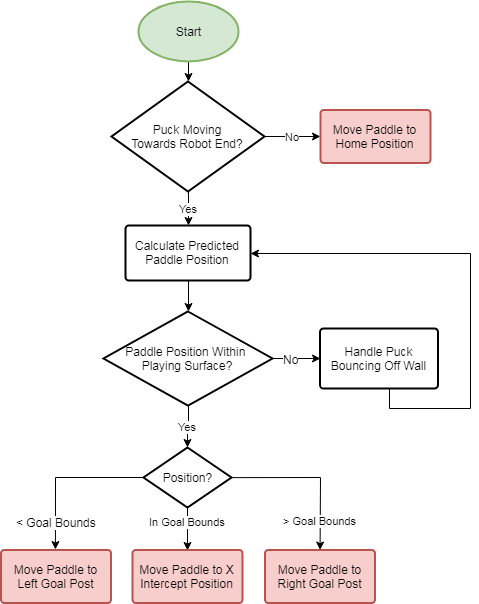


Figure 6 - Complex Defense Algorithm for Automated game

Complex Defense Algorithm for Automated game is shown in Figure 6. Its main purpose is to make sure that paddle defends the goal boundaries at all cost, it has no offensive strategy. An equation of a line (y = mx + b) is a fundamental math concept that lies underneath Calculate Predicted Paddle Position step of the Defense Algorithm. PT from two consecutive camera frames determines two puck position values on the playing surface, this gives us X and Y vectors. Afterward using the vectors and y = mx + b, algorithm predicts the trajectory and final position of the moving puck. Next, each camera frame allows us to average out trajectory and position calculations to improve accuracy. Then robot blocks puck at Y = 0, and returns puck to human if necessary. Defense Control Strategy State Machine is shown in Appendix C – Figure 19.

Offense Algorithm for Automated game’s objectives are to score goals on the human and do not let the human score goals. The algorithm is implemented using the same principle as Complex Defense Algorithm. However, robot attempts to hit the puck near the middle of the table (Y closer to the center line) with max speed. If attack is not possible (due to travel time) then robot is reverted to defensive movements. Offense Control Strategy State Machine is shown in Appendix C – Figure 20.

## 2.3 Puck Tracker

The Puck Tracker (PT) is implemented using python and OpenCV - an open source computer vision library. The puck tracker runs as an independent process on a Linux computer and communicates with the Master Controller through inter process communication. The puck tracker utilizes a PlayStation Eye camera mounted above the table and connected to the PC via USB to provide the tracking capabilities. The puck trackers functionality can be split up into two key parts: fiducial locating and tracking of the air hockey puck. See Figure 7 for a detailed block diagram of the puck tracker system.

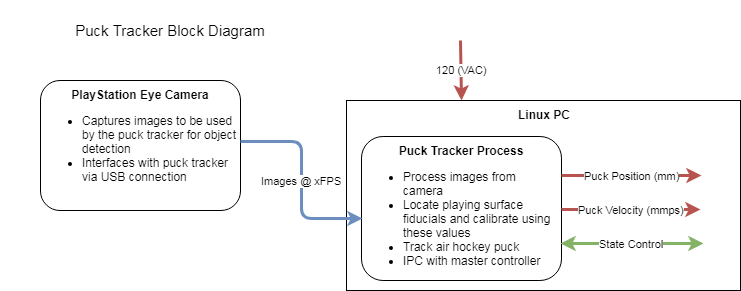


Figure 7 - Puck Tracker Block Diagram

**Fiducial Locating**

The playing surface of the air hockey table has 4 pink two inch circles at the corners. These fiducials (markers) are used by the puck tracker to dynamically frame the playing surface as seen from the camera mounted above the air hockey table. The puck tracker uses OpenCV to detect the fiducials based on a color threshold as well as size, and then saves these fiducial locations to a JSON file. The fiducial locations are then used to compute a transformation matrix which is applied to every incoming frame of the camera to correct the image perspective for the camera. This correction is necessary to make the image appear “top down” to the system even though the camera is mounted at one end of the table. The corrected images are used during puck tracking as explained in the section below.

**Puck Tracking**

When the puck tracker is in normal operation, it is continuously using OpenCV to detect the air hockey pucks location and velocity on the playing surface frame by frame from the incoming video stream of the camera. In order to provide position in mm and velocity in mm per second to the master controller, the puck tracker must have a perspective corrected image of the playing surface (as explained in the fiducial finding section). Once this corrected image is acquired, OpenCV is used to detect the puck within the image. The method to detect the puck involves applying a color threshold to the image in a range that includes just the color of the puck and removes all other colors. The puck tracker then draws contours around all the objects located in the image and checks the radius and area of the located contours to ensure that it is the pucks shape. Once detected, a circle is drawn around the puck and the pixel coordinates are translated to a position in mm relative to the xy coordinate system. Consecutive puck position readings allow us to report velocity of the puck using distance divided by time. The puck position and velocity are reported to the master controller for further use.

## 2.4 User Interface

The User Interface (UI) is implemented using python and Kivy – an open source Python library for rapid development of multi-touch applications – on a 10.1” capacitive touch screen. The UI runs as an independent process on a Linux computer and communicates with the Master Controller through inter process communication. The UI consists of 6 different screen options which can be navigated to using the main menu (Figure 8) and each screen has an option to return to the main menu. The UI screens are described in detail in the sections below.

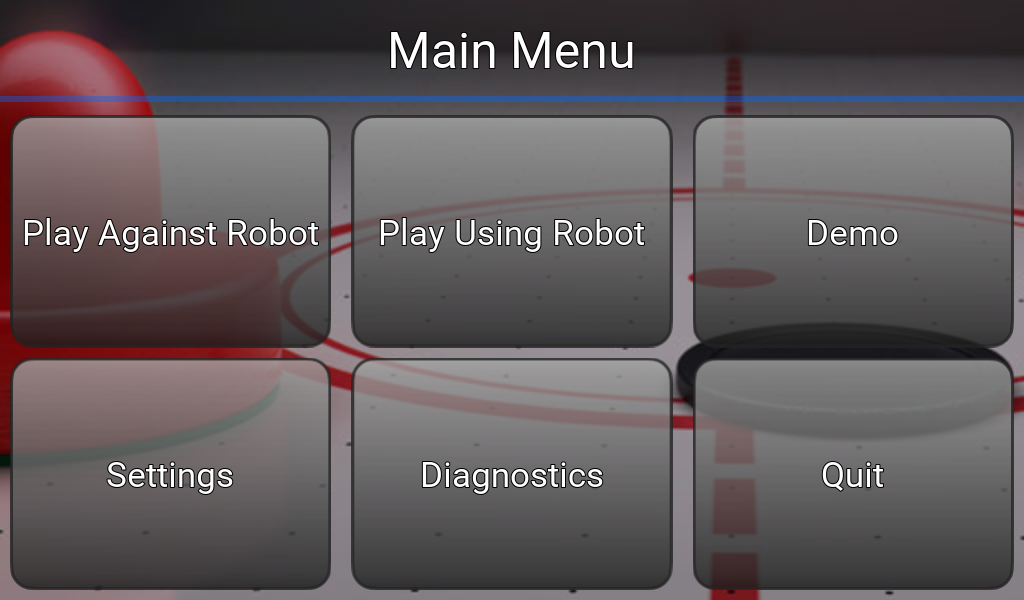


Figure 8 - Main Menu

**Play Against Robot (Appendix B - Figure 15)**

This screen includes a visualization of the puck tracking system and game control for playing against the robot. The visualization portion of this screen shows the tracking of the air hockey pucks movement as well as the predicted path of the puck as it moves towards the robots net and the required paddle controller position to intercept the puck in motion. The game control portion of the screen allows the user to play a timed game against the robot and keeps track of the score using the break-beam sensors of the paddle controller. The user can pause, resume, or reset the game using on screen buttons at any time.

**Play Using Robot (Appendix B - Figure 16)**

The multi-touch screen allows us to implement a method to manually control the paddle controller using a graphical air hockey paddle located on a playing surface on the screen. The screen also includes game control identical to the system described in the section above.

**Demo**

While this screen has yet to be implemented, the motive is to have a predefined routine that the paddle controller can complete to demonstrate its capabilities.

**Settings (Appendix B - Figure 17)**

The settings screen includes options to control the game interface. These options currently include game length (1, 2, or 5 minutes), game mode (defense or offense), and independent control of the x & y axis speeds (slow, medium, or fast).

**Diagnostics (Appendix B - Figure 18)**

The diagnostics screen provides real-time data regarding the state and error reporting of the four core modules (user interface, paddle controller, master controller, and puck tracker). There are buttons to calibrate the paddle controller and puck tracker, as well as a button to clear all errors manually. The paddle controller calibration will be performed entirely by that module; however, the puck tracker calibration includes two additional screens with sliders to adjust the color thresholds of the puck tracking system. These color thresholds occasionally need to be adjusted due to lighting conditions. This calibration routine also includes finding the location of the puck tracker fiducials for when the camera position has been adjusted. All of these settings are saved to a JSON file for long term storage when requested by the user.

**Quit**

The quit option allows the user to quit the User Interface application.

## 2.5 Test Methods and Plan

All modules of the system are capable of being tested separately by providing simulated inputs and verifying the correctness of the modules outputs (black-box testing). Detailed component level testing was performed where applicable (ex: testing power rails on the Paddle Controller circuit board). The fully integrated system was tested under normal operating conditions to verify its function and performance. The fully integrated system was also be tested while in “error states” to verify correct operation when one or more modules cannot function as designed. All testing was non-destructive to maintain the integrity of the prototype. Detailed test plans are found below.

## System Test Plan

|  |  |  |
| --- | --- | --- |
| **Test** | **Purpose** | **Expected Result** |
| Defense Test   1. Power on system. 2. Select “defense” from game mode settings in UI. 3. Start game from UI. | Verify operation of defense operating mode. | Air hockey paddle moves in X-Axis only to stop human from scoring goals. |
| Offense Test   1. Power on system. 2. Select “offense” from game mode settings in UI. 3. Start game from UI. | Verify operation of offense operating mode. | Air hockey paddle moves in X and Y-Axis to stop human from scoring goals and to attempt to score goals on the human. |
| Manual Control Test   1. Power on system. 2. Select “manual control” from game mode settings in UI. 3. Start game from UI. | Verify operation of manual control mode. | Air hockey paddle moves in X and Y-Axis following manual control movement on touch-screen. |

## Paddle Controller

**Electro-Mechanical System Test Plan**

|  |  |  |
| --- | --- | --- |
| **Test** | **Purpose** | **Expected Result** |
| X-Axis motor test   1. Apply 24V to positive motor terminal, GND to negative motor terminal 2. Apply GND to positive motor terminal, 24V to positive motor terminal. | Verify operation of the X-Axis motor. | 1. Paddle moves to the right. 2. Paddle moves to the left. |
| Y-Axis motor test   1. Apply 24V to positive motor terminals, GND to negative motor terminals 2. Apply GND to positive motor terminals, 24V to positive motor terminals. | Verify operation of the Y-Axis motors. | 1. Paddle moves forward. 2. Paddle moves backwards. |
| Quadrature encoder test (x3)   1. Rotate motor clockwise 2. Rotate motor counter-clockwise   Repeat for each of the three quadrature encoders. | Verify operation of the quadrature encoders. | 1. “B-phase” output is low (0V) on each “A-phase” output rising edge (0-5V transition). 2. “B-phase” output is high (5V) on each “A-phase” output rising edge (0-5V transition). |
| Limit switch test (x6)   1. Press the limit switch 2. Release the limit switch   Repeat for each of the six limit switches. | Verify operation of the limit switches. | 1. Limit switch terminals short-circuited. 2. Limit switch terminals open-circuited. |
| Home switch test (x3)   1. Press the home switch 2. Release the home switch   Repeat for each of the three home position switch. | Verify operation of the home position switches. | 1. Home switch terminals short-circuited. 2. Home switch terminals open-circuited. |
| Infrared break-beam sensor test (x4)   1. Obstruct path of break beam sensor with solid object 2. Do not obstruct path of break beam sensor   Repeat for each of the four break-beam sensors. | Verify operation of the break-beam sensors. | 1. IR sensor output is low (0V). 2. IR sensor output is high (5V). |
| Emergency stop test   1. Press the emergency stop 2. Release the emergency stop | Verify the operation of the emergency stop. | 1. 0V applied to both terminals of all three motors. 2. 0-24V applied to both terminals of all three motors. |

**Embedded Hardare Test Plan**

|  |  |  |
| --- | --- | --- |
| **Test** | **Purpose** | **Expected Result** |
| Power supply test   1. Apply 24V and Ground to power input connector. | Verify operation of the 15V and 5V regulators. | V24V power rail measures 24VDC +/- 0.25VDC.  V15V power rail measures 15VDC +/- 0.1VDC.  V5V power rail measures 5VDC +/- 0.1VDC. |
| Motor test (x3)   1. Apply 50% duty cycle PWM waveform to input of motor driver circuit. 2. Apply 95% duty cycle PWM waveform to input of motor driver circuit. 3. Apply 5% duty cycle PWM waveform to input of motor driver circuit.   Repeat of each of the three motor driver circuits. | Verify operation of the motor driver circuits. | 1. Average DC voltage measured between motor driver positive and negative outputs = 0VDC +/- 0.25VDC. 2. Average DC voltage measured between motor driver positive and negative outputs = 21.6VDC +/- 0.25VDC. 3. Average DC voltage measured between motor driver positive and negative outputs = -21.6VDC +/- 0.25VDC. |
| Quadrature encoder test (x6)   1. Apply 0-5VDC square wave to A-phase and B-phase inputs.   Repeat for each of the three quadrature encoder interface circuits. | Verify operation of the quadrature encoder interface circuits. | A-phase output is inverted copy of A-phase input. B-phase output is “latched” value of B-phase input for each A-phase rising edge. |
| Limit switch test (x6)   1. Short-circuit the limit switch output pins. 2. Open-circuit the limit switch output pins.   Repeat for each of the six limit switch interface circuits. | Verify operation of the limit switch interface circuits. | 1. Motor driver “shutdown” pin measures 0VDC. 2. Motor driver “shutdown” pin measures 5VDC. |
| Home switch test (x3)   1. Short-circuit the home switch output pins. 2. Open-circuit the home switch output pins.   Repeat for each of the three home switch interface circuits. | Verify operation of the home switch interface circuits. | 1. Home output pin measures 5VDC. 2. Home output pin measures 0VDC. |
| IR LED driver test (x6)   1. Measure output to IR LED.   Repeat for each of the six IR LED driver circuits. | Verify operation of the IR LED driver circuits. | 1. IR LED positive output is 56 kHz +/- 2 kHz square wave at 0-5VDC. IR LED negative output is grounded. |
| IR sensor test (x6)   1. Apply 5V to IR sensor input pin. 2. Apply 0V to IR sensor input pin.   Repeat for each of the six IR sensor interface circuits. | Verify operation of the IR sensor interface circuits. | 1. IR sensor output pin measures 0VDC. 2. IR sensor output pin measures 5VDC. |

**Embedded Software Test Plan**

|  |  |  |
| --- | --- | --- |
| **Test** | **Purpose** | **Expected Result** |
| Motor test (x3)   1. Command positive X-Axis movement. 2. Command negative X-Axis movement. 3. Command positive Y-Axis movement. 4. Command negative Y-Axis movement. | Verify operation of the motor control software. | 1. X-Axis motor PWM output is >50% duty cycle. 2. X-Axis motor PWM output is <50% duty cycle. 3. Y-Axis motor PWM outputs are >50% duty cycle. 4. Y-Axis motor PWM outputs are <50% duty cycle. |
| Quadrature Encoder test (x3)   1. Rotate motor clockwise. 2. Rotate motor counter-clockwise.   Repeat for each of the three motors. | Verify operation of the quadrature encoder software. | 1. Motor position variable increases. 2. Motor position variable decreases. |
| Home switch test (x3)   1. Press home position switch   Repeat for each of the three home position switches. | Verify operation of the home position switch software. | 1. Motor position variable is reset to its default value (varies depending on motor). |
| Infrared break-beam sensor test (x4)   1. Obstruct path of break-beam sensor with solid object   Repeat for each of the four break-beam sensors. | Verify operation of the break-beam sensor software. | 1. PWM output to all three motors are 50% duty cycle (i.e. off). |
| State machine test   1. Reset microcontroller 2. Obstruct path of any break-beam sensor with a solid object. 3. Stop obstructing all break-beam sensors. Send state command “CLEAR\_ERROR” over CAN. 4. Send state command “ON” over CAN. 5. Send state command “CALIBRATION” over CAN. 6. Send state command “OFF” over CAN. | Verify operation of the Paddle Controller state machine software. | 1. State = “OFF” 2. State = “ERROR” 3. State = “OFF” 4. State = “CALIBRATION”, homing routine begins. State = “ON” after homing routine is complete. 5. State = “CALIBRATION”, homing routine begins. State = “ON” after homing routine is complete. 6. State = “OFF” |

## Master Controller

|  |  |  |
| --- | --- | --- |
| CAN bus test   1. Transmit MC\_Cmd\_PC (value 0x04030201) message over CAN bus 2. Receive PC\_Status (value 0x01020304) message over CAN bus | Verify communication protocol with Paddle Controller over CAN bus | MC receives PC\_Status = “0x01020304”  PC receives MC\_Cmd\_PC = “0x04030201” |
| IPC with PT test   1. Change each value in pt\_tx array 2. Change each value in pt\_rx array 3. Move web-camera | Verify communication protocol with Puck Tracker using Python multiprocessing array and queue and logging | 1. See changes in pt\_tx signals under HDF5 file 2. See changes in pt\_rx signals under HDF5 file 3. Video stream on UI display has changed within 0.3 s |
| IPC with UI test   1. Change each value in ui\_tx array 2. Change each value in ui\_rx array 3. Move web-camera | Verify communication protocol with User Interface using Python multiprocessing array and queue and logging | 1. See changes in ui\_tx signals under HDF5 file 2. See changes in ui\_tx signals under HDF5 file 3. Video stream on UI display has changed within 0.3 s |
| Defense Algorithm test   1. Hit puck to move to the left of goal area 2. Hit puck to move to the right of goal post 3. Hit puck to move into the goal boundaries 4. Repeat above steps with multiple bounces | Verify defense algorithm | 1. Paddle travels to the left goal post 2. Paddle travels to the right goal post 3. Paddle blocks the oncoming puck within the goal boundaries 4. Paddle should react the same as above |
| Offense Algorithm test   1. Place puck on the robot half of the table 2. Hit puck towards to robot 3. Hit puck towards to robot with multiple bounces | Verify offense algorithm | * 1. Paddle hits the puck to go within the opponent’s goal boundaries |
| Error and quit handling test   1. Disconnect Webcam 2. Turn off Paddle Controller 3. Reconnect Webcam 4. Turn on Paddle Controller 5. Press quit under UI 6. Start the MC | Verify error and quit handling | 1. UI diagnostics shows error under PT state 2. UI diagnostics shows that PC is off 3. UI diagnostics shows that PT is active 4. UI diagnostics shows that PC is ON 5. All processes terminated 6. UI diagnostics shows that PT, PT, MC are active |

## Puck Tracker

|  |  |  |
| --- | --- | --- |
| **Test** | **Purpose** | **Expected Result** |
| Find fiducials through UI | Verify that the fiducials can be located through software and saved to JSON file for future use | Fiducials found and locations saved to JSON |
| Measure puck tracker position output and compare to real world measurement (both x and y axis) | Verify that the position reporting is accurate | Puck tracker position output equal to physical results +/- 10 mm |
| Hit puck at high velocity and check tracking | Verify that at high speeds we don’t lose puck tracking | Puck can be tracked at high speeds |
| Unplug camera | Verify that an error is thrown when the camera is unplugged | Puck tracker reports error |
| Place object similar in color to the puck on playing surface | Verify that the puck is still tracked and not the new object | The puck tracker will continue to track the puck due to radius and area constraints |

## User Interface

|  |  |  |
| --- | --- | --- |
| **Test** | **Purpose** | **Expected Result** |
| Start application from command line | Verify that the Kivy application starts | Kivy application opens to intro screen |
| Main Menu tests – navigate to each screen and back to main menu | Verify that each menu button brings you to the correct screen and back | Each menu button navigates to correct screen |
| Play Against Robot Screen test   1. Start game 2. Play against robot 3. Pause game 4. Resume game 5. Reset game | Verify that the game control buttons work and visualization active | Observe visualization of gameplay   1. Game clock starts 2. Robot moves 3. Game clock stops (as does robot) 4. Game clock resumes 5. Game clock and score reset |
| Play Using Robot Screen test   1. Start game 2. Drag paddle around screen 3. Pause game 4. Resume game 5. Reset game | Verify that the game control buttons work as well as manual control | 1. Game clock starts 2. Robot moves 3. Game clock stops (as does robot) 4. Game clock resumes 5. Game clock and score reset |
| Demo Screen test   1. Hit demo button | Verify paddle controller movement | Paddle controller completes predefined motion routine |
| Settings Screen test   1. Change every listed setting 2. Go to Play Against Robot Screen 3. Confirm settings changed | Verify that the settings screen adjustments match gameplay settings | Each changed setting modifies the game play of the Play Against Robot screen |
| Diagnostics Screen test   1. Calibrate puck tracker 2. Calibrate paddle controller 3. Clear errors 4. Observe state and error reporting | Verify that the diagnostic features of the user interface are working | 1. Puck tracker calibrates successfully 2. Paddle controller calibrates successfully 3. Errors are cleared successfully 4. State and error are reported |
| Quit Button test | Verify that hitting the quit button closes the Kivy application | Kivy application closes |

## 2.6 Implementation

The schematics for the custom designed PCB can be found in the Appendix. The source code for all modules of the air hockey robot will be attached separately with the submission of this report. The figures below show images of the constructed air hockey robot.



Figure 9 - Assembled Air Hockey Robot



Figure 10 - Assembled Paddle MicroController Expansion Board

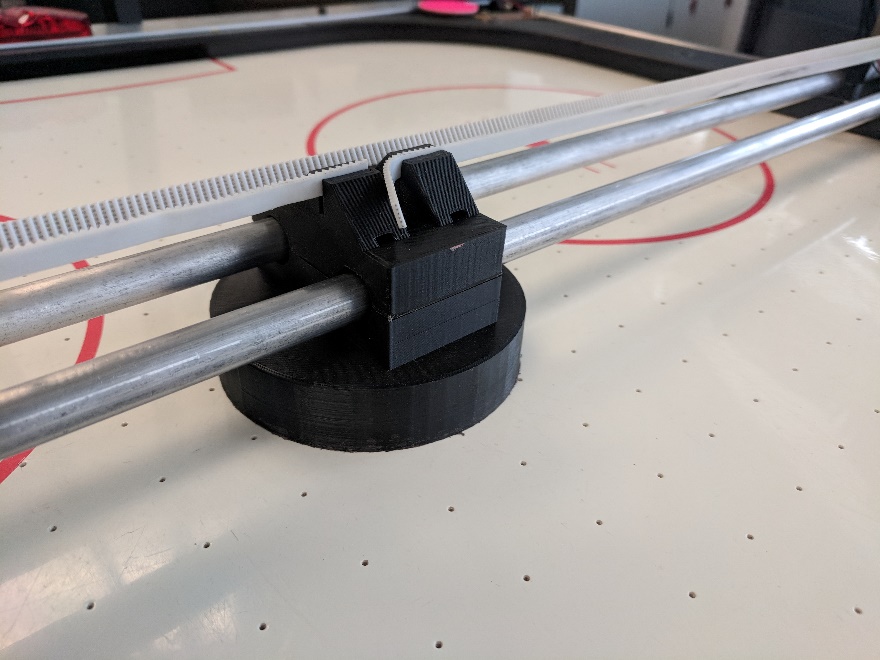


Figure 11 - Air Hockey Robot Paddle

## 2.7 Safety and Impact on Environment and Society

This project should have a positive impact on the ESE program by increasing the programs exposure within the academic community, and by providing outward visibility for potential students, employers, and industry partners. This project can also serve as a platform to allow disabled people to play air hockey, or for professional players to train against. This project has no notable impact on the environment as only one prototype has been constructed and mass-production is not planned.

This solution features a high-speed robot operating in close proximity to humans which may pose several safety hazards. If a human were to reach into the operating area of the robot they would be at risk of pinching, crushing, or cutting their fingers and hands. If the robot were to suffer a catastrophic failure while operating at high speeds components of the robot may become projectiles that could strike bystanders.

To mitigate these safety risks the robot has been designed and constructed with several safety features. First, a virtual barrier has been constructed around the robots operating area using IR LEDs and IR sensors to create a “light screen”. If any object crosses any of these virtual barriers and enters the robots operating area power to the motors will be shut off and the robot will stop moving. Second, the motor control software includes overload protection logic that will shut off power to the motors if excessive resistance is detected while applying torque to the motors. Finally, an emergency stop is mounted prominently on the humans’ side of the air hockey table. When pressed, all power to the robot is shut off and the robot will stop moving.

# 3 Results and Analysis

## 3.1 Qualitative Analysis

The fully-integrated Air Hockey Robot is capable of competing against an average human player in a game of air hockey and represents a dramatic performance improvement over the open-source design that was constructed using 3D printer parts [1]. Our solution also features many relevant technologies used in industry such as motor driver design and construction, tracking high-speed objects in real-time and intercepting objects in motion in real-time

## 3.2 Measurement Results and Quantitative Analysis

The step response for the Air Hockey Robot was measured and is shown in Figure 12. The robot is able to achieve a peak velocity of 4m/s in the Y-Axis and over 5m/s in the X-Axis. Compared to the modelled step response of the system (**Error! Reference source not found.**) the constructed version is ~20% slower. The speeds of the integrated system are lower due to the software control of the robot which artificially slows down the robot as it approaches the end of its travel limits. This is to prevent mechanical damage from the robot colliding with the hard limits of the air hockey table. With the addition of a mechanical device to slow down the robot at the ends of travel (such as a spring and damper) the robot should be able to reach higher speeds without causing mechanical damage to the system.

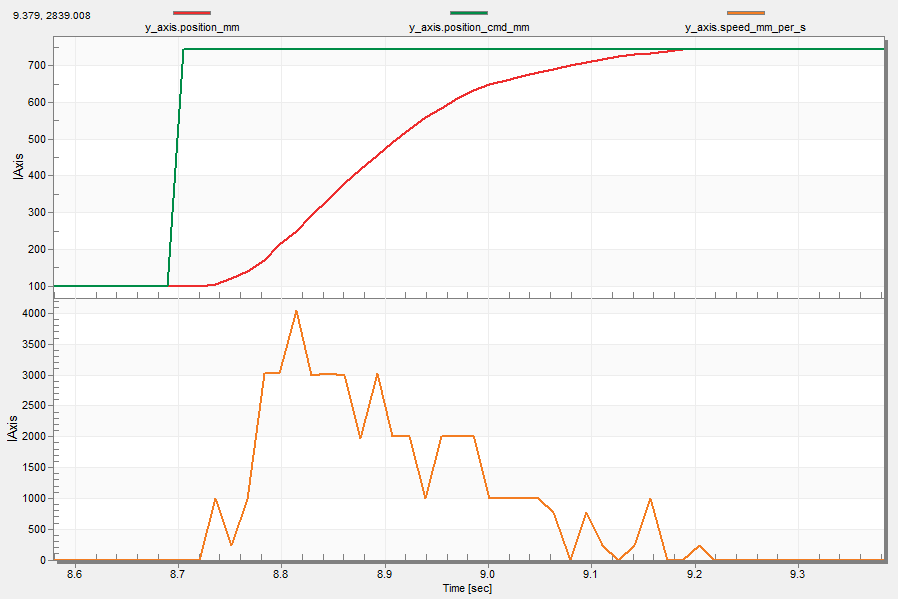
At the current operating speeds the Air Hockey Robot can compete against a human player which was the goal of this project. 

Figure 12 - Y-Axis Step Response

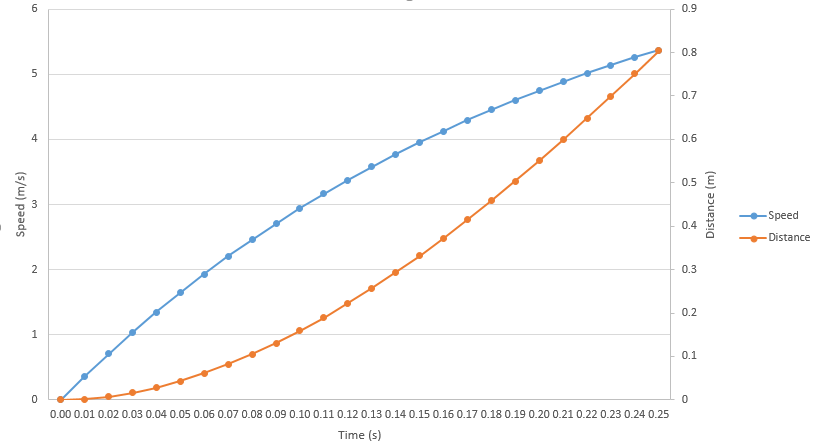


Figure 13 - Y-Axis Modelled Step-Response

# 4 Conclusion and Recommendations for Further Work

The design and integration of the air hockey robot has been highly successful in the scope of this project. The prototype that has been constructed has met all performance targets and can reliably play air hockey against a human player. The User Interface provides easy access for humans to control the air hockey robot and debug the system at a high level. Robustness of the prototype mechanical and electrical systems has been satisfactory.

Further development of the prototype could be conducted to improve performance and robustness, specifically reducing friction in the mechanical system of the Paddle Controller and improving the strength of the 3D printed components. For mass production the system should be redesigned to have all functions run off of a single embedded controller and eliminate the need for an expensive desktop computer. All 3D printed parts would be replaced with injected molded parts, and the mechanical design could be improved to reduce the number of fasteners used to reduce cost.

# 5 Appendices

## Appendix A – Coordinate System

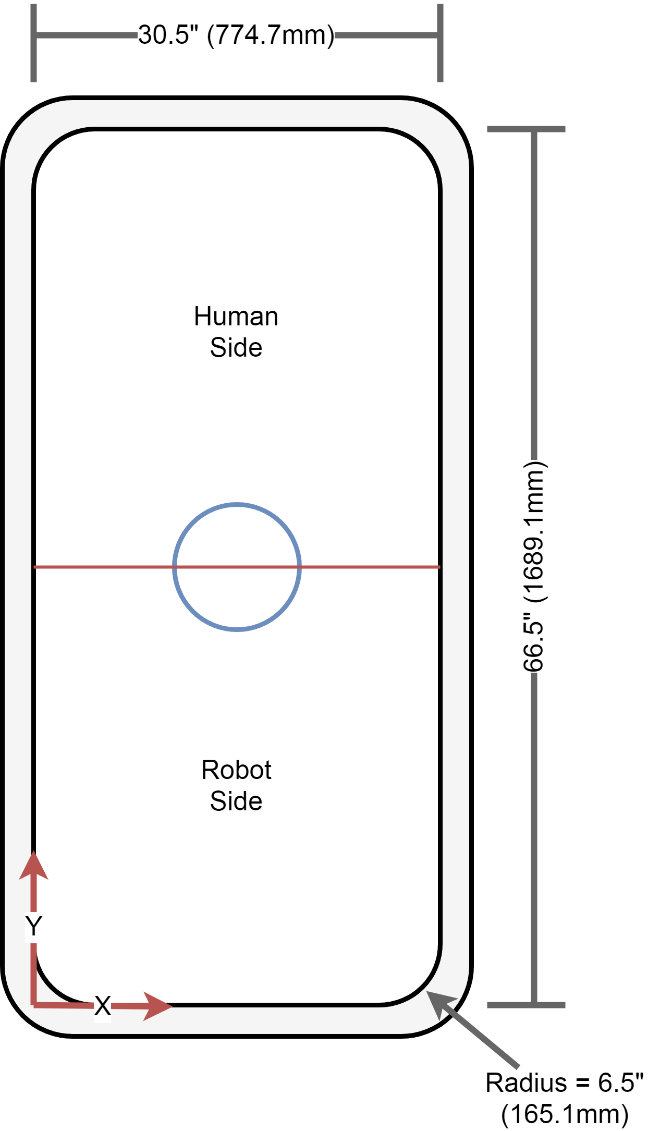


Figure 14 - Coordinate System

## Appendix B – UI Images

****

Figure 15 - Play Against Robot

****

Figure 16 - Play Using Robot

****

Figure 17 - Settings

****

Figure 18 – Diagnostics

## Appendix C – State Machines



Figure 19 – Defense Control Strategy State Machine

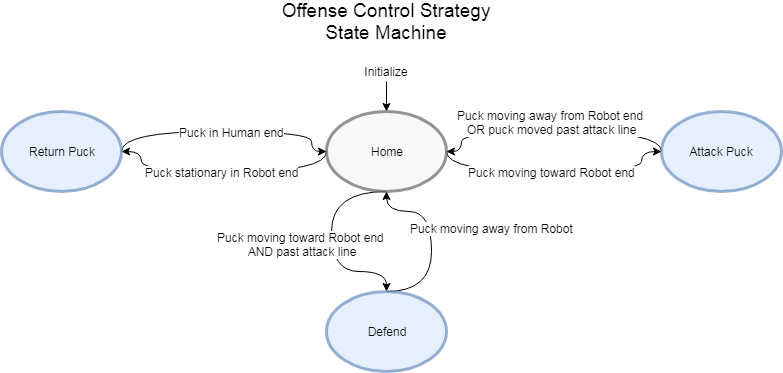


Figure 20 – Offense Control Strategy State Machine

## Appendix D – Microcontroller Interface Circuit Schematics & PCB Layout

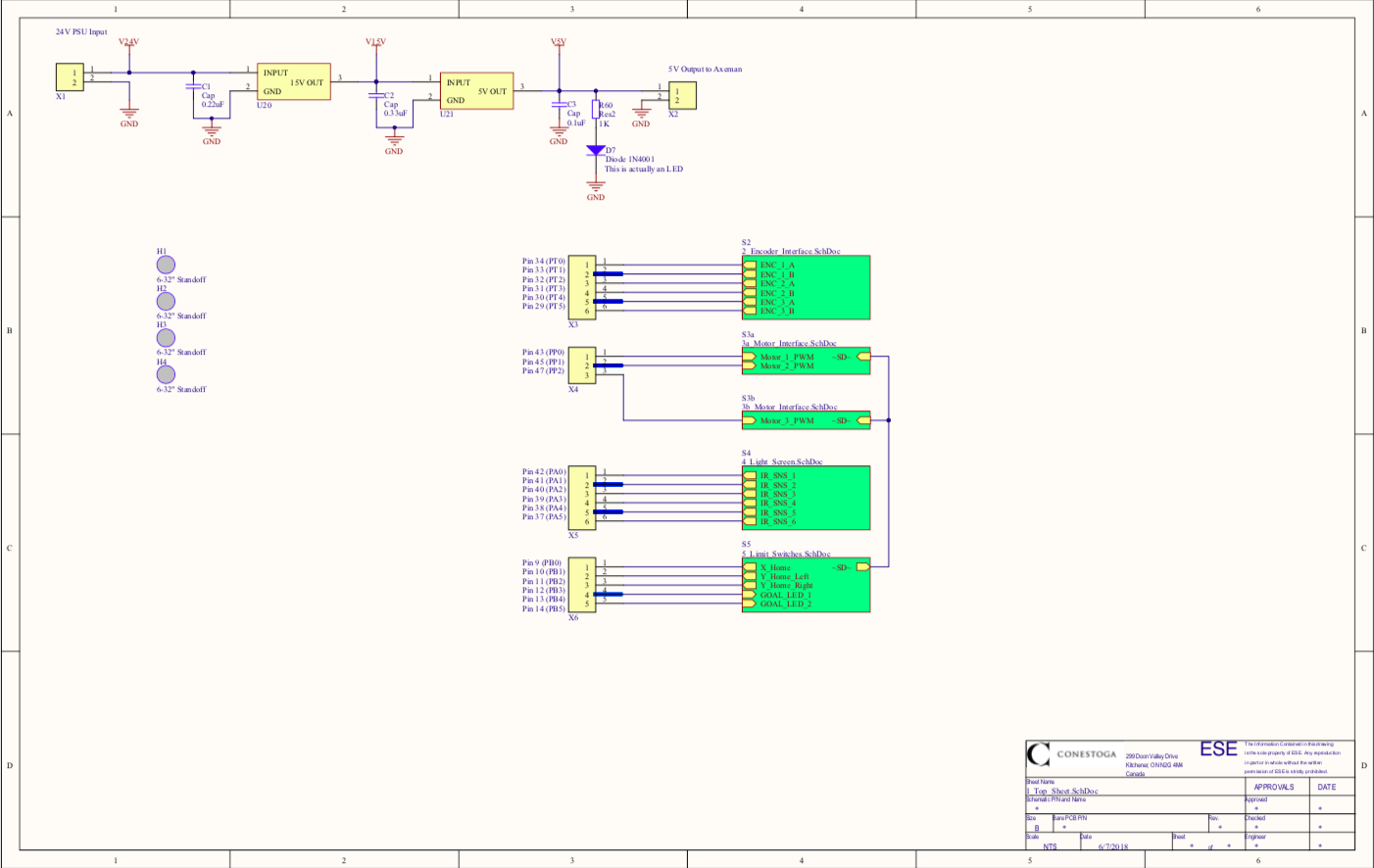
****

Figure 21 - Top Level Schematic & Power System

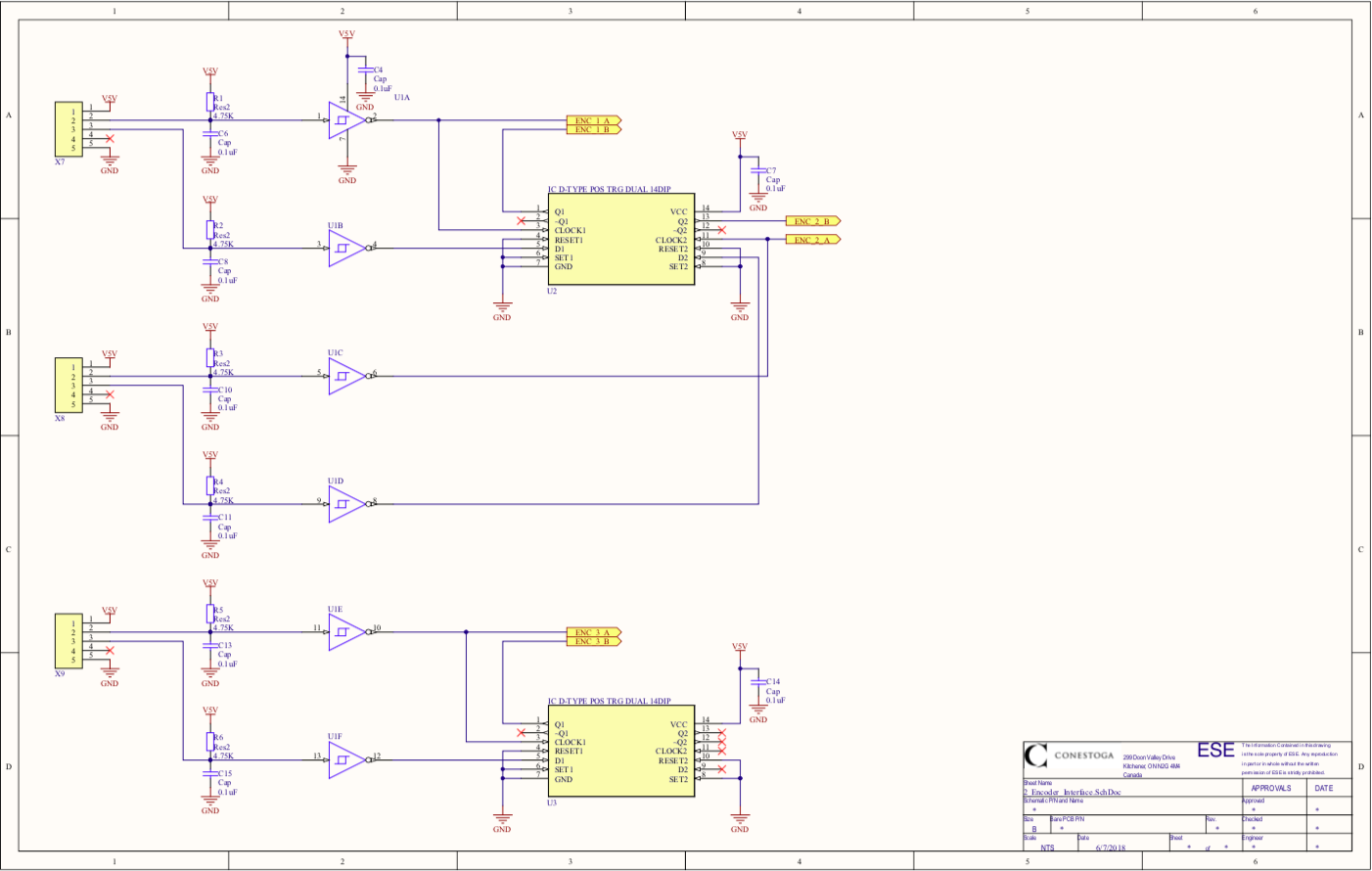


Figure 22 - Encoder Interface

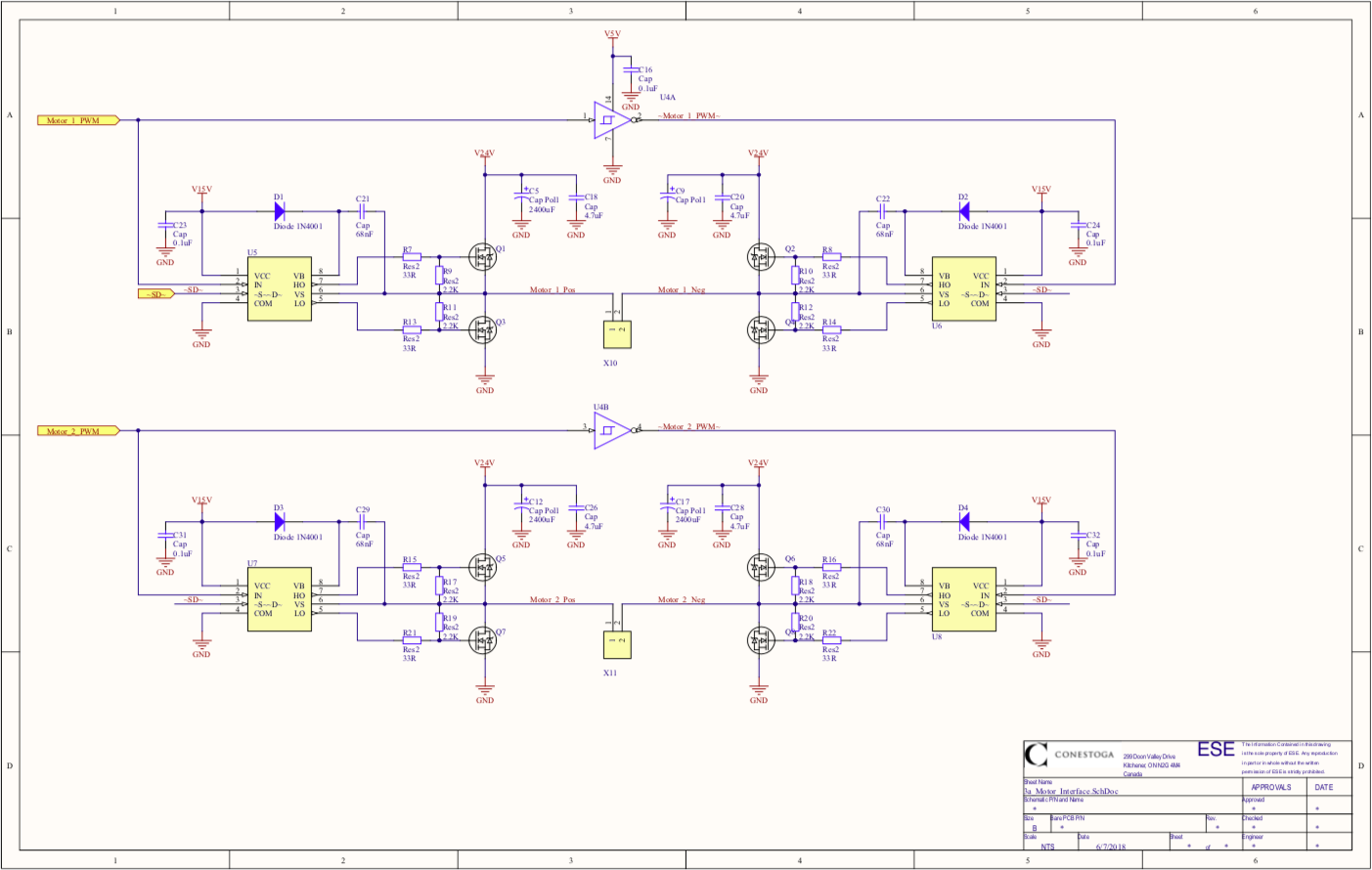
****

Figure 23 - Motor Interface (Part 1)

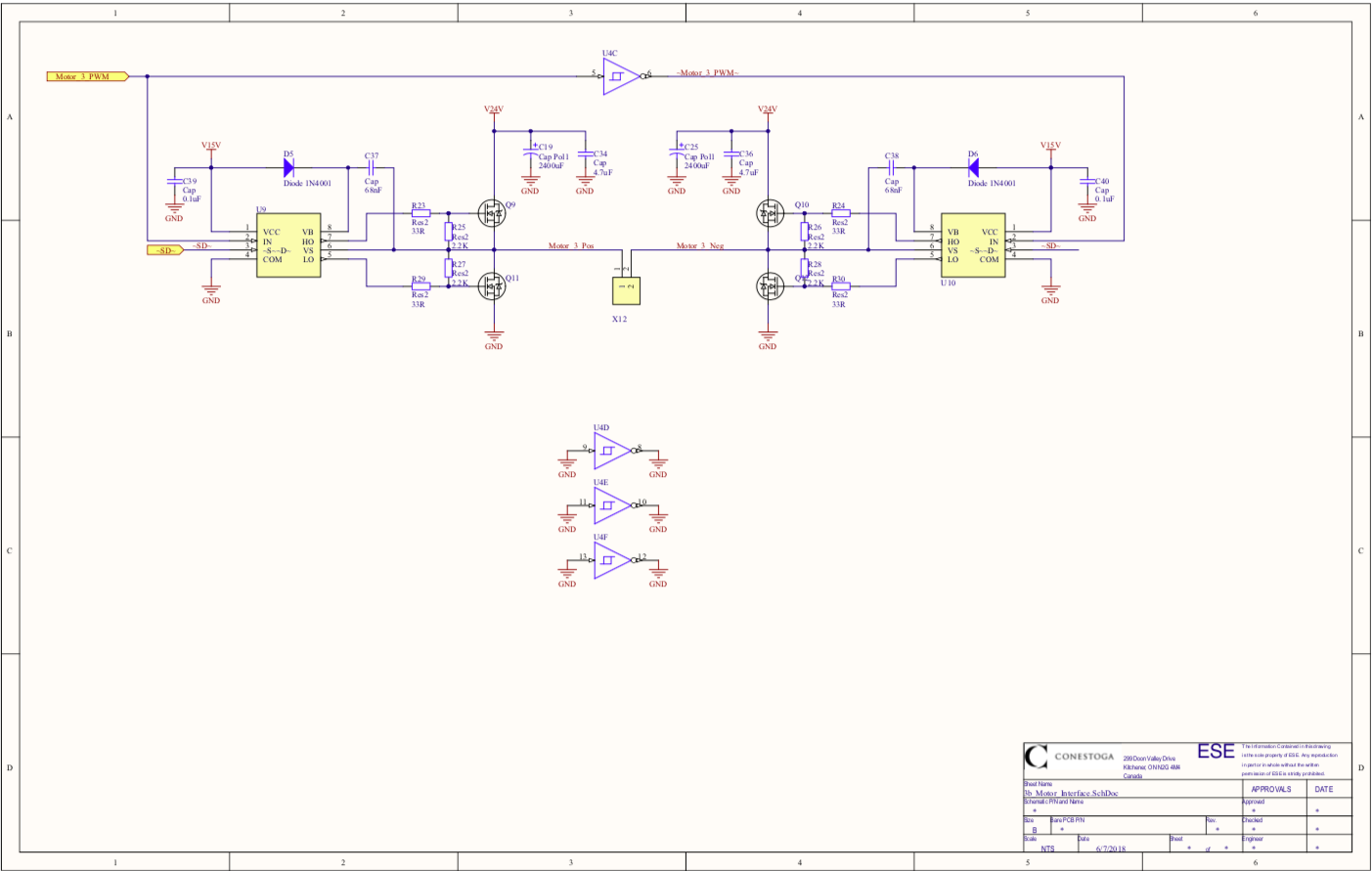
****

Figure 24 - Motor Interface (Part 2)

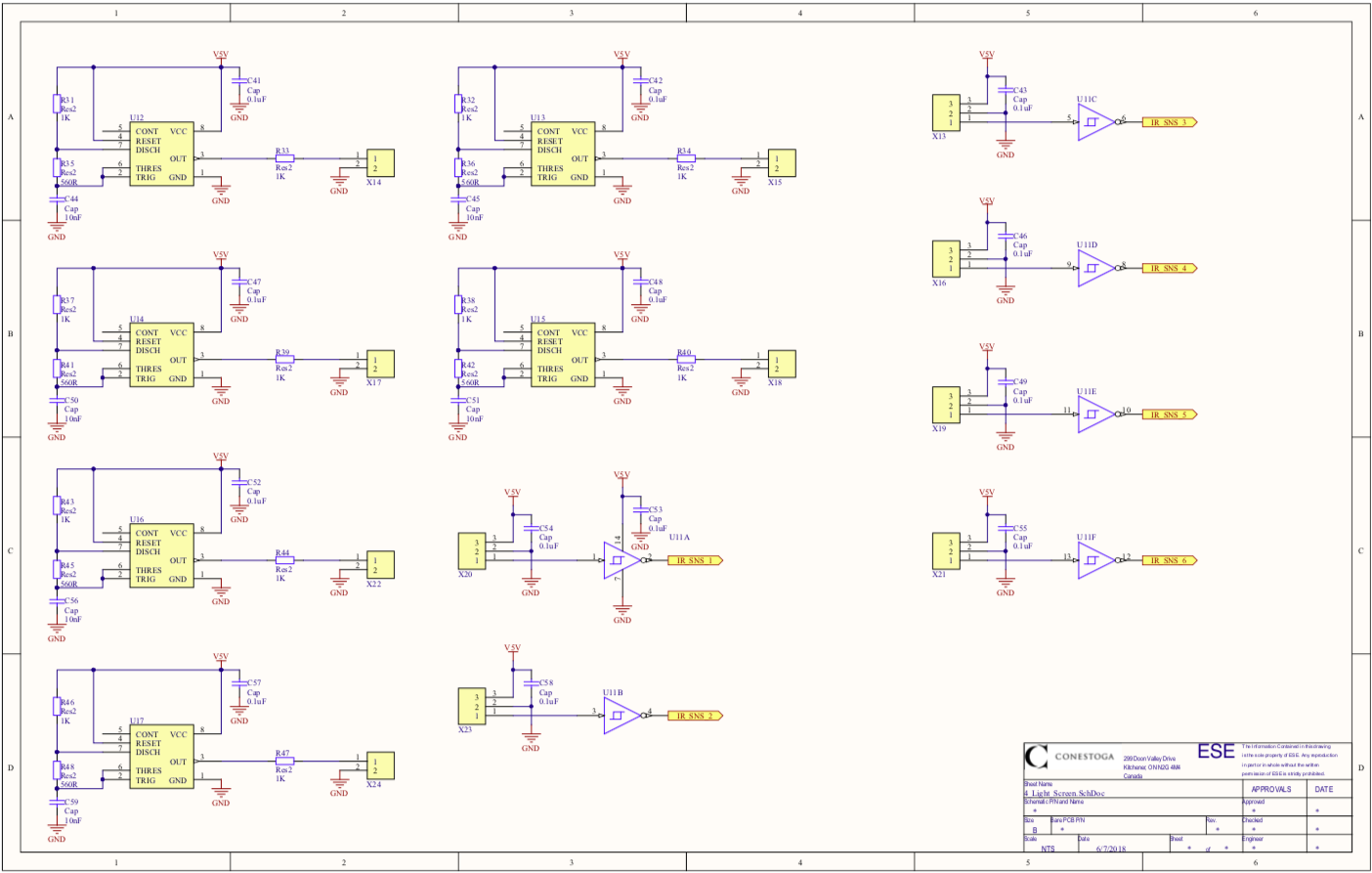


Figure 25 - Light Screens

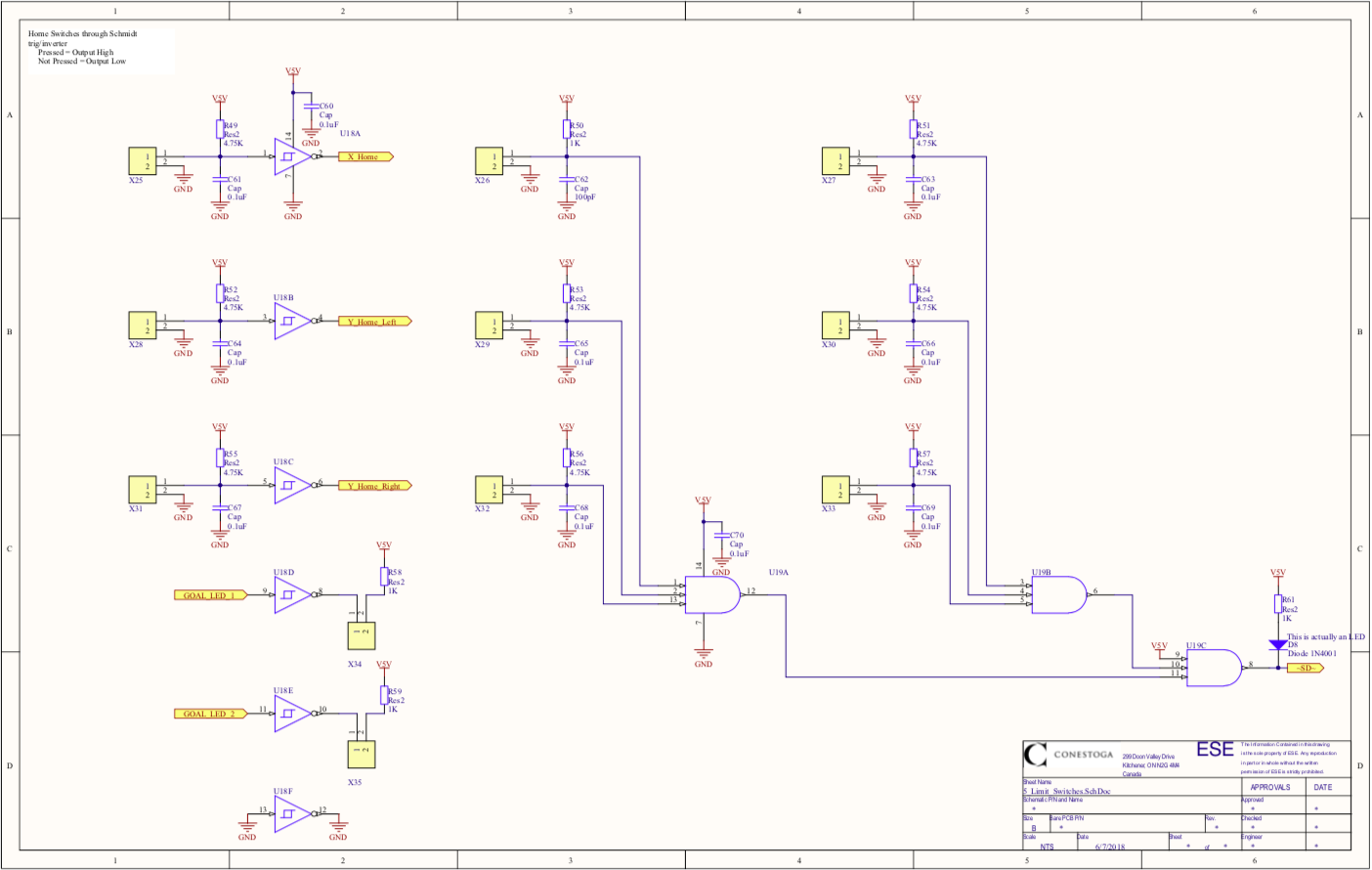


Figure 26 - Limit Switches

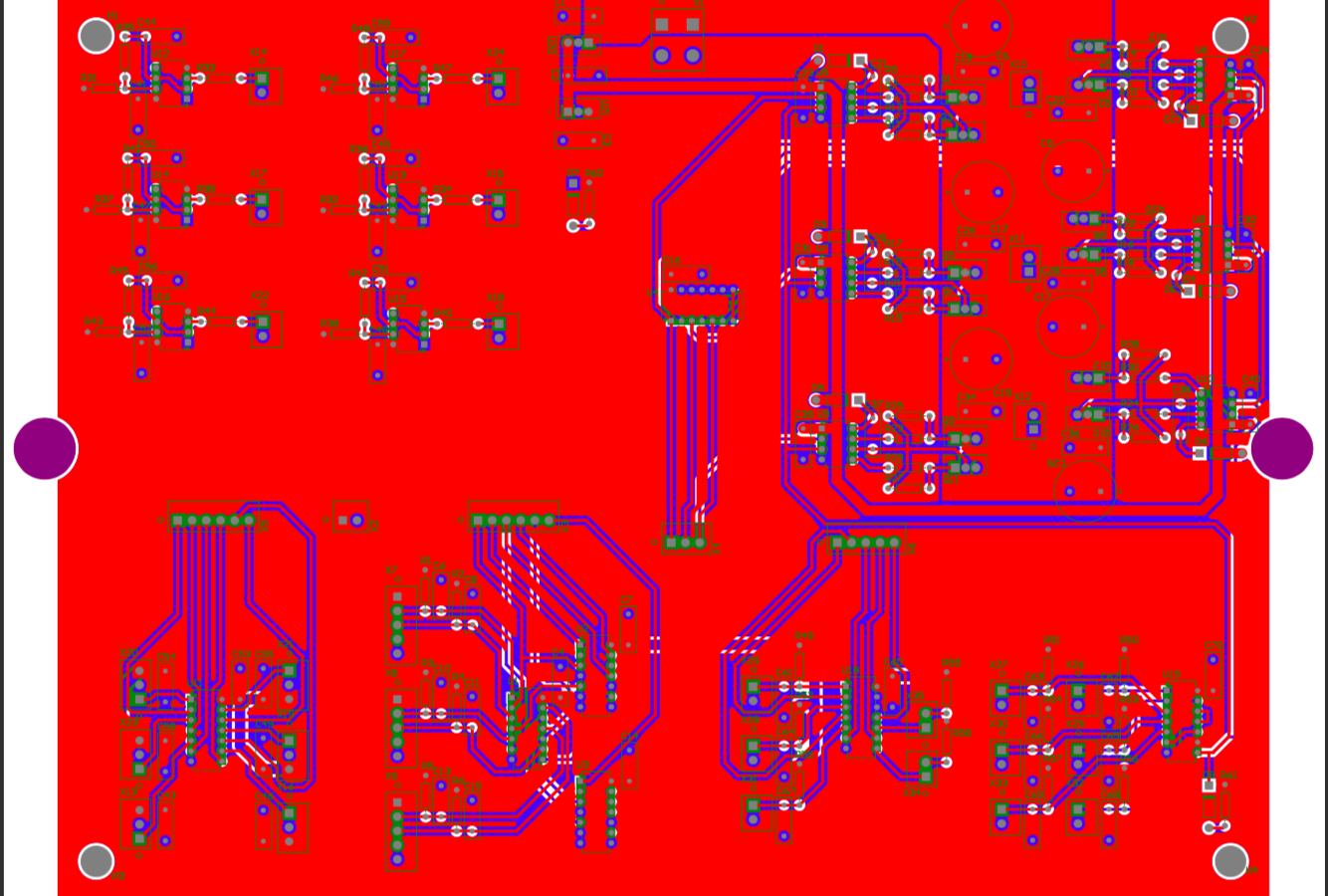


Figure 27 - PCB Layout

## Appendix E – Goal Light Circuit Schematic & PCB Layout

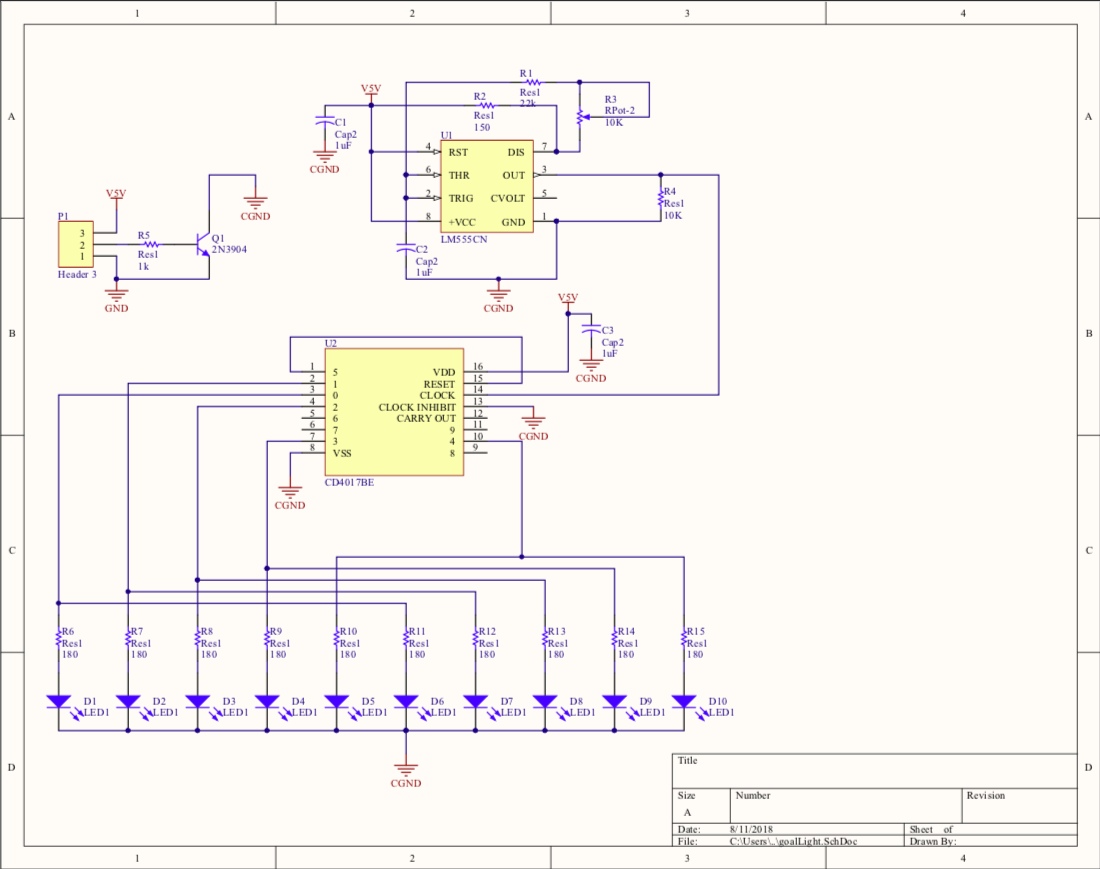


Figure 28 - Goal Light Schematic

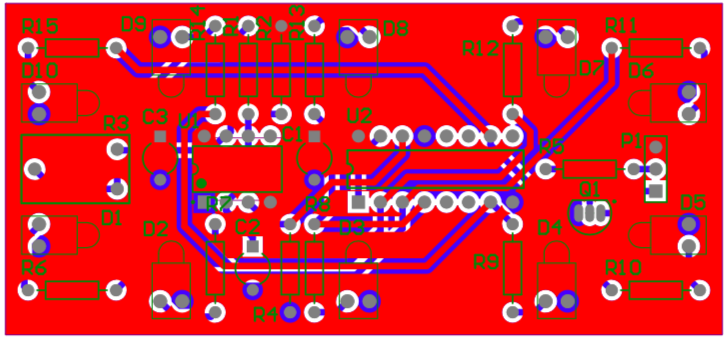
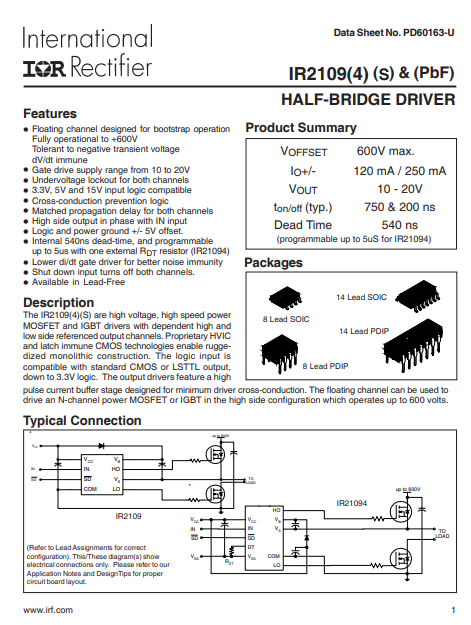


Figure 29 - Goal Light PCB Layout

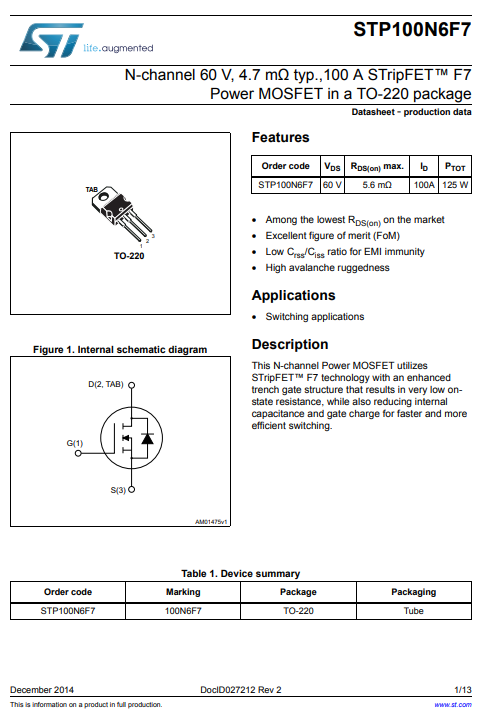
## Appendix F – Data Sheets

This appendix includes the first page from the data sheets of each major component in the electrical system. PDF versions of the full data sheets will be submitted with this report.

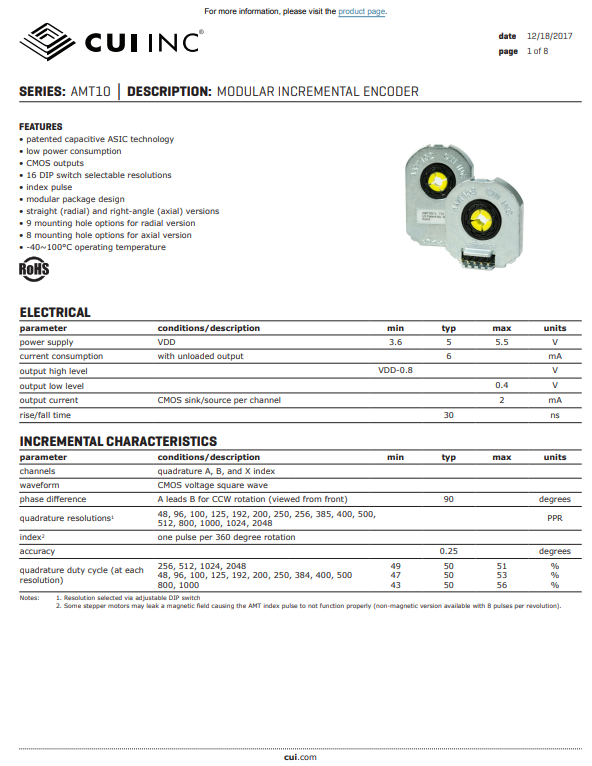
**Half-Bridge Driver (IR2109)**



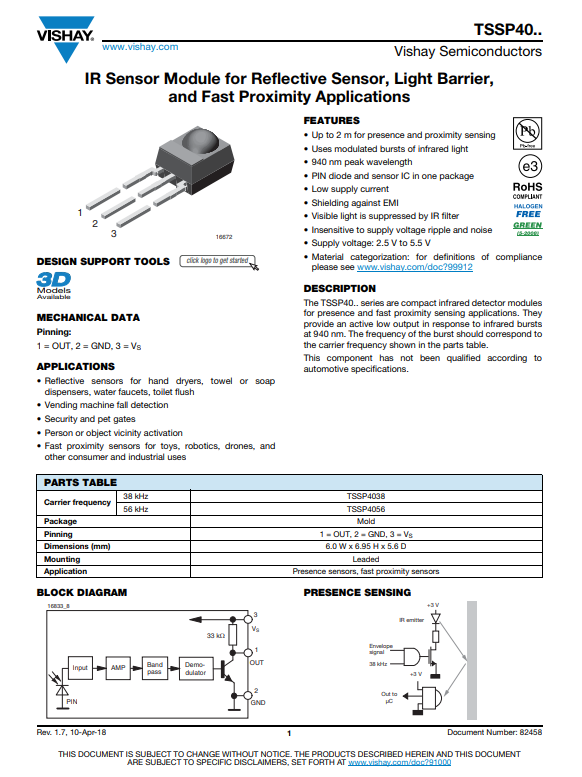
**STP100N6F7 MOSFET**



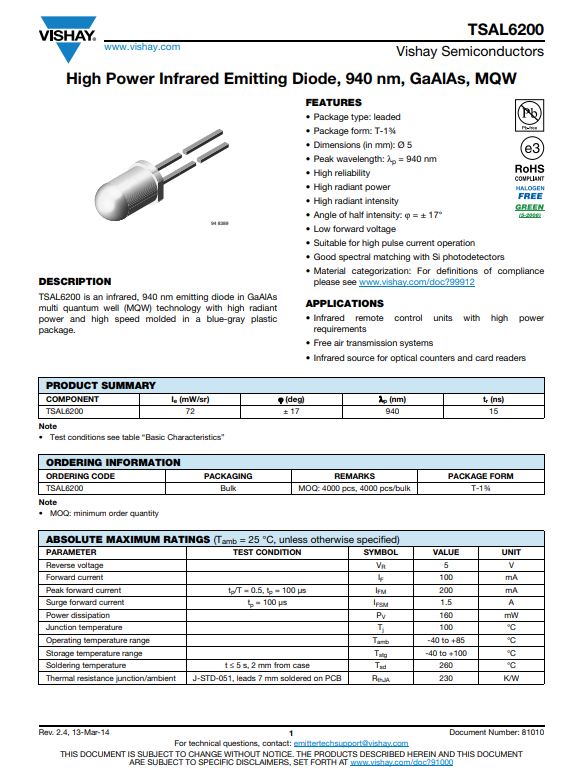
**AMT102-V Quadrature Encoder**



**TSSP40 IR Sensor Module**



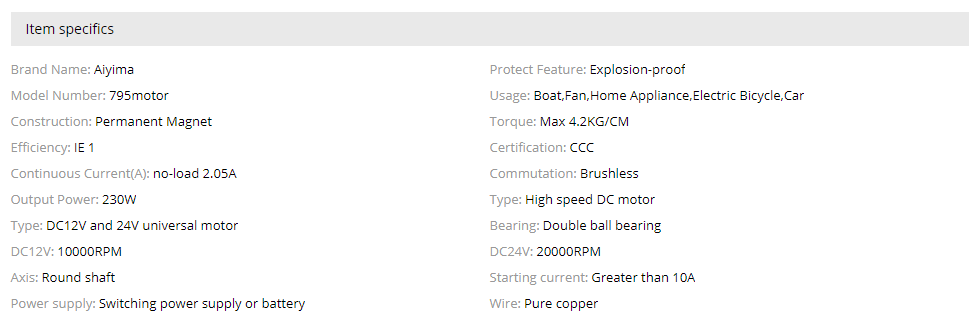
**High Power Infrared LED**



**AIYIMA 795 Brushed DC Motor (No Full Datasheet Available)**

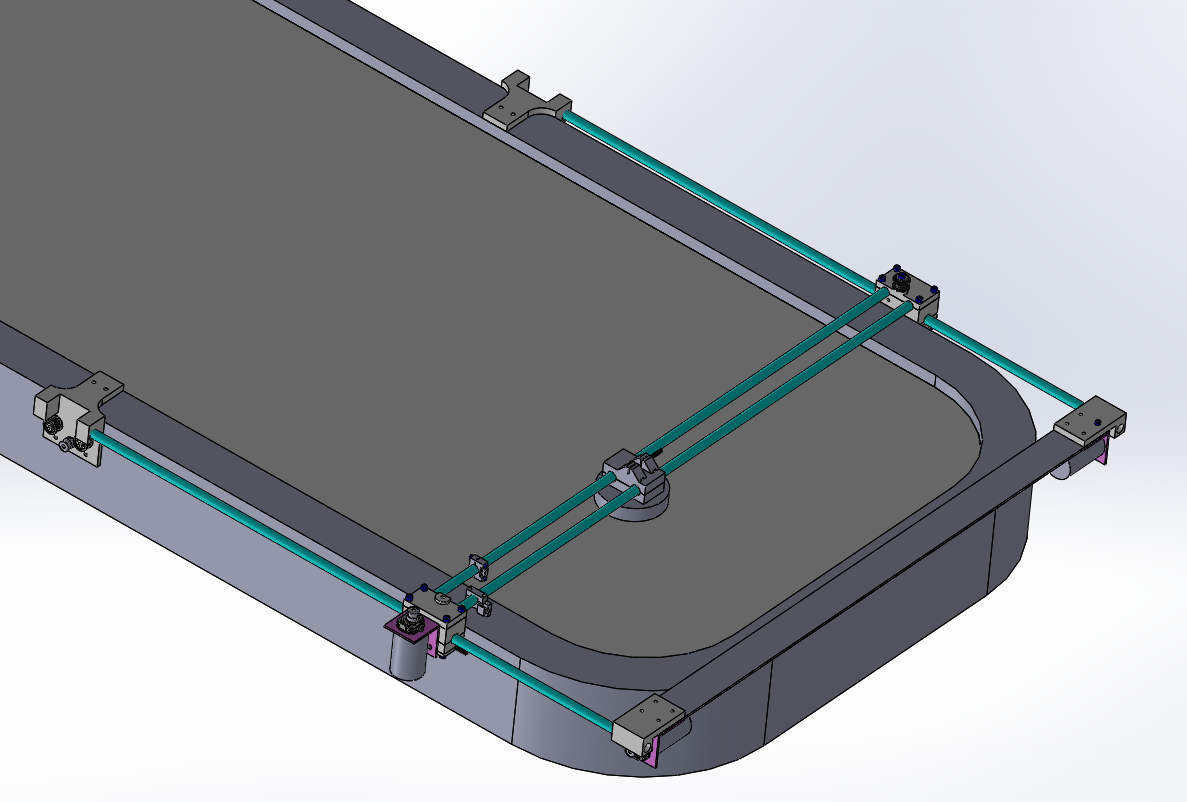
<https://www.aliexpress.com/item/Aiyima-795-DC-Motor-Large-Torque-High-Power-DC12V-24V-Universal-Motor-Double-Ball-Bearing-Mute/32837636904.html?spm=a2g0s.9042311.0.0.7b6a4c4d2JasWI>



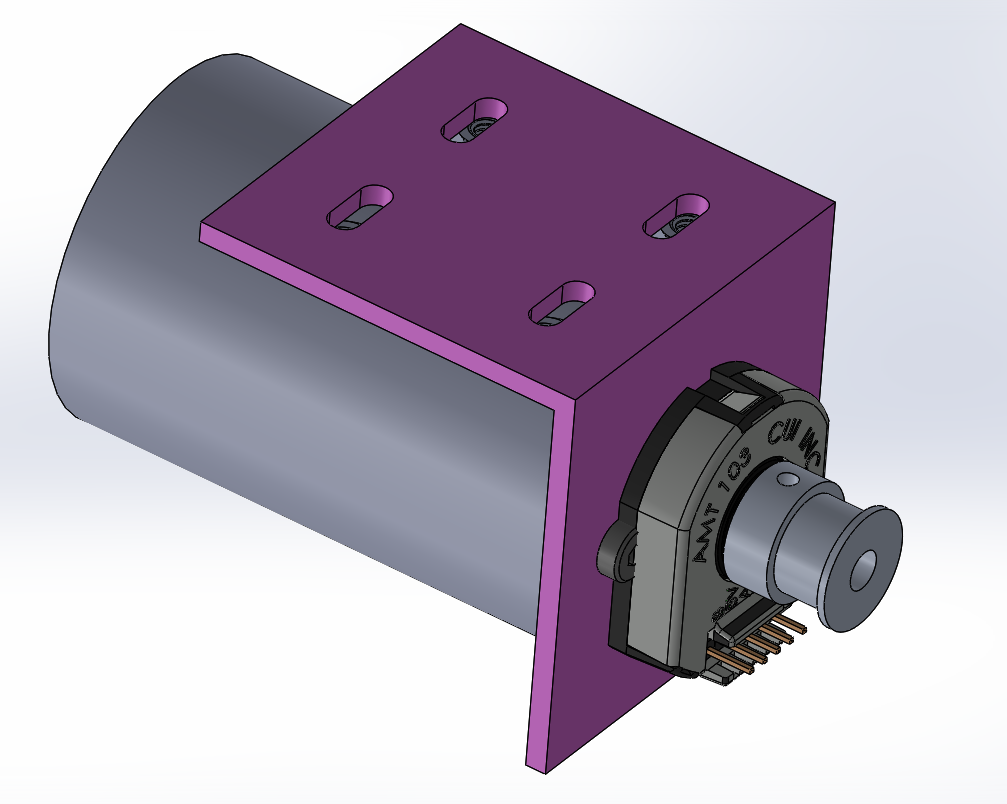


## Appendix G – CAD Models

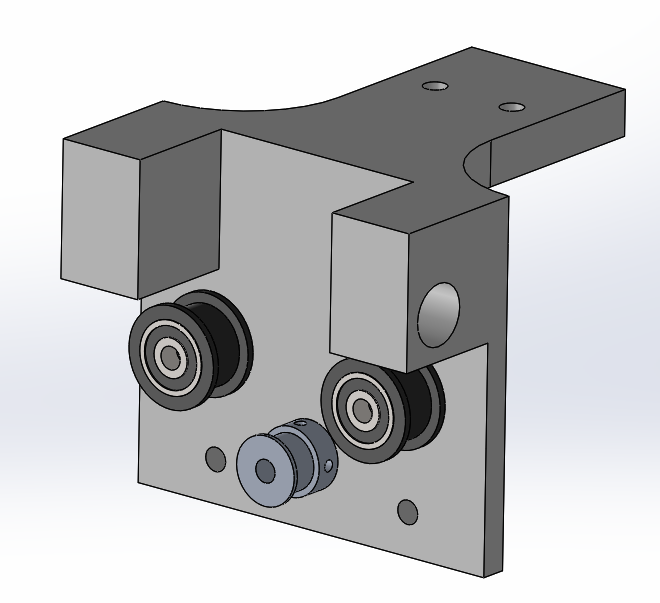
**Air Hockey Robot Assembly**



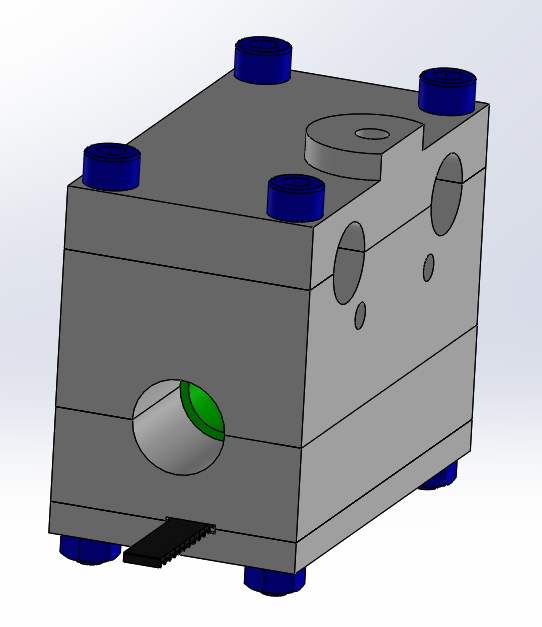
**Motor + Encoder Assembly**



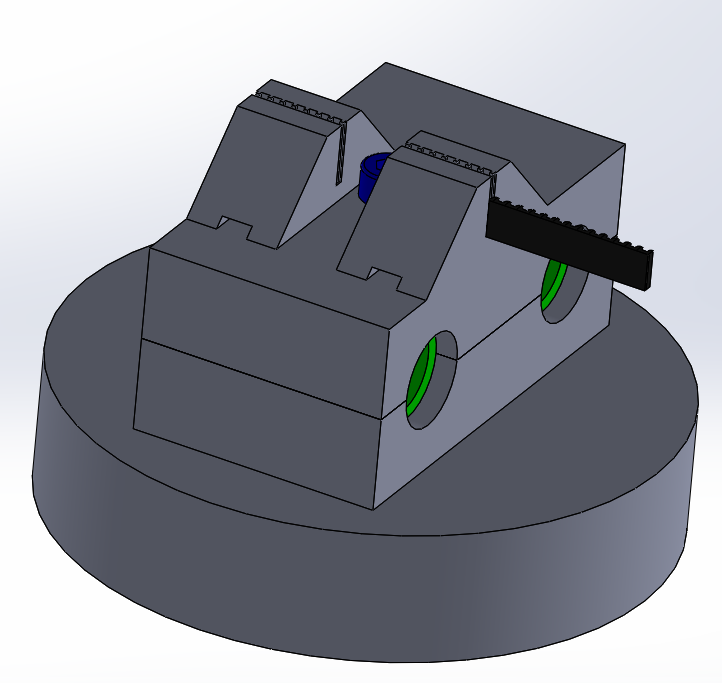
**Y-Axis Connecting Rod Mount Assembly**



**X-Axis Carriage Assembly**



**Paddle Assembly**



## Appendix H - Risk Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Risk (Priority Highest to Lowest) | Category | Impact on Project Objectives | Potential Risk Reduction |
| 1. Mechanical design/integration problems | Technical | Unable to control robot motion | -Start mechanical prototyping early  -Leverage group members Mechanical Engineering experience |
| 2. Real-time object tracking problems | Technical | Unable to automate robot motion | -Start object tracking prototyping early  -Leverage proven open-source object tracking solutions |
| 3. Security of project in shared classroom | External | Lack of lab workspace availability.  Potential damage to project hardware. | -Utilize dedicated ESE lab space  -Advocate for continued support of ESE dedicated lab space |
| 4. Catastrophic loss of data | Organizational, external, technical | Schedule delays. | -Utilize source control for all project materials  -Manually back up all data once per week |
| 5. Managing scope creep | Organizational, project management | Schedule delays.  Lack of focus on core features. | -Strictly define scope of project during planning phase  -Additional features shall only be implemented after 100% completion of core project features |
| 6. System sizing incorrect | Technical, performance | Lower than desired system performance. | -Use system level performance requirements to drive component design  -Define system level performance requirements based on real-world data |
| 7. Inexperience with HMI design & implementation | Technical | Less relatable demonstration.  Worse user experience.  Difficult to debug. | -Define user interface features early (see: Managing scope creep)  -Start HMI prototyping early  -Utilize popular GUI implementation solutions |

# 6 References and Bibliography

[1] "Air Hockey Robot (a 3D printer hack)", *jjrobots*, 2018. [Online]. Available: https://www.jjrobots.com/air-hockey-robot-a-3d-printer-hack/. [Accessed: 08- Feb- 2018].

[2] "AIR HOCKEY ROBOT EVO - jjrobots", *jjrobots*, 2018. [Online]. Available: https://www.jjrobots.com/the-open-source-air-hockey-robot/. [Accessed: 08- Feb- 2018].

[3] "Multiple Objects Positioning and Identification Method Based on Magnetic Localization System - IEEE Journals & Magazine", *Ieeexplore.ieee.org*, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/7496898/. [Accessed: 08- Feb- 2018].

[4] H. Parekh, D. Thakore and U. Jaliya, "A Survey on Object Detection and Tracking Methods", *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 2, 2018.

[5] S. A ̊gren, *Object tracking methods and their areas of application: A meta-analysis*. 2018.

[6] "Create an Object Tracking System: Integrating Servo Control with Object Detect", *Allaboutcircuits.com*, 2018. [Online]. Available: https://www.allaboutcircuits.com/projects/create-an-object-tracking-system-integrating-servo-control-with-object-dete/. [Accessed: 08- Feb- 2018].

[7] W. Hamlet and W. Kusewich, "Using The Ultrasonic Sensor to Determine Location", *Engineering.nyu.edu*, 2018. [Online]. Available: http://engineering.nyu.edu/mechatronics/summit/SUMMIT2007/group6-Will\_Bill/Summit2007Project.pdf. [Accessed: 08- Feb- 2018].

[8] "Motors and Selecting the Right One", *Learn.sparkfun.com*, 2018. [Online]. Available: https://learn.sparkfun.com/tutorials/motors-and-selecting-the-right-one. [Accessed: 08- Feb- 2018].

[9] "Linear Positioning - Motion Control Application", *Oriental Motor U.S.A. Corp.*, 2018. [Online]. Available: http://www.orientalmotor.com/applications/linear-positioning.html. [Accessed: 08- Feb- 2018].

[10] "Know Your Pneumatics: Linear Actuators", *Blog.parker.com*, 2018. [Online]. Available:   
 http://blog.parker.com/know-your-pneumatics-hints-tips-for-specifying-linear-actuators.   
 [Accessed: 08- Feb- 2018].