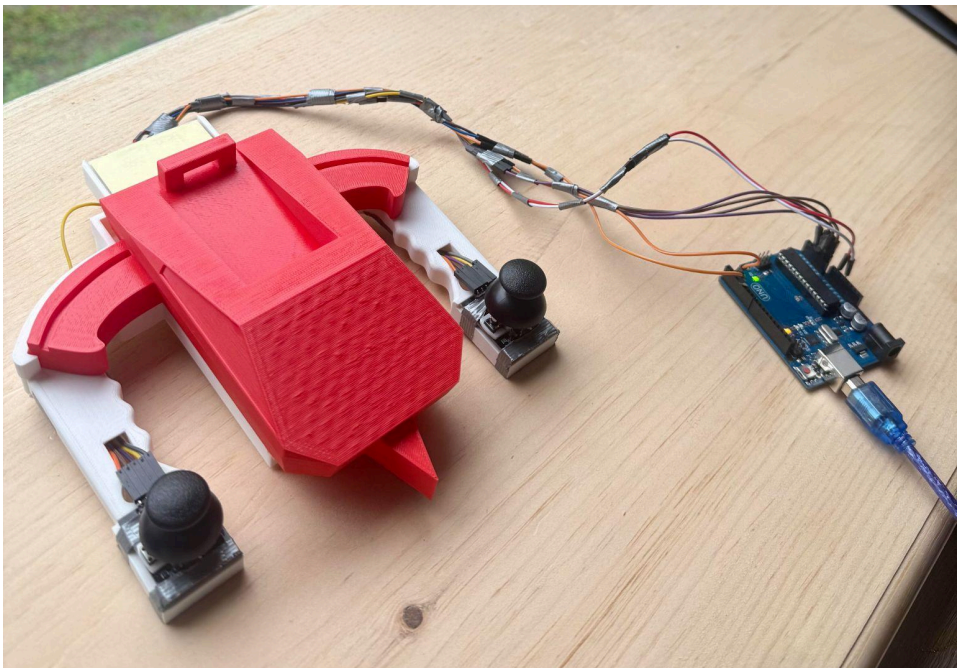
Thomas Watson

Personal Project 1: AIO Arduino-Matlab Video Game System

Built Using Custom Analog/Digital input, ODE's, Trigonometry and Design Logic



Runge–Kutta method

(1) $y' = F(x,y), \qquad y_0 = f(x_0) \rightarrow \ y = f(x)$

(2) $y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(h^5)$

$$k_1 = hF(x_n,\ y_n)$$
$$k_2 = hF(x_n + \frac{h}{2},\ y_n + \frac{k_1}{2})$$
$$k_3 = hF(x_n + \frac{h}{2},\ y_n + \frac{k_2}{2})$$
$$k_4 = hF(x_n + h,\ y_n + k_3)$$

Goal:
Design and Manufacture (3-D Print with Prusa 4s) a video game controller with optimized ergonomics. Use this controller to control self-constructed video games that are governed by first principles thinking (math, physics, and logic).

Purpose:
Display rounded skill set (ergonomic mechanical design, interface between hardware and software, data manipulation and creativity).

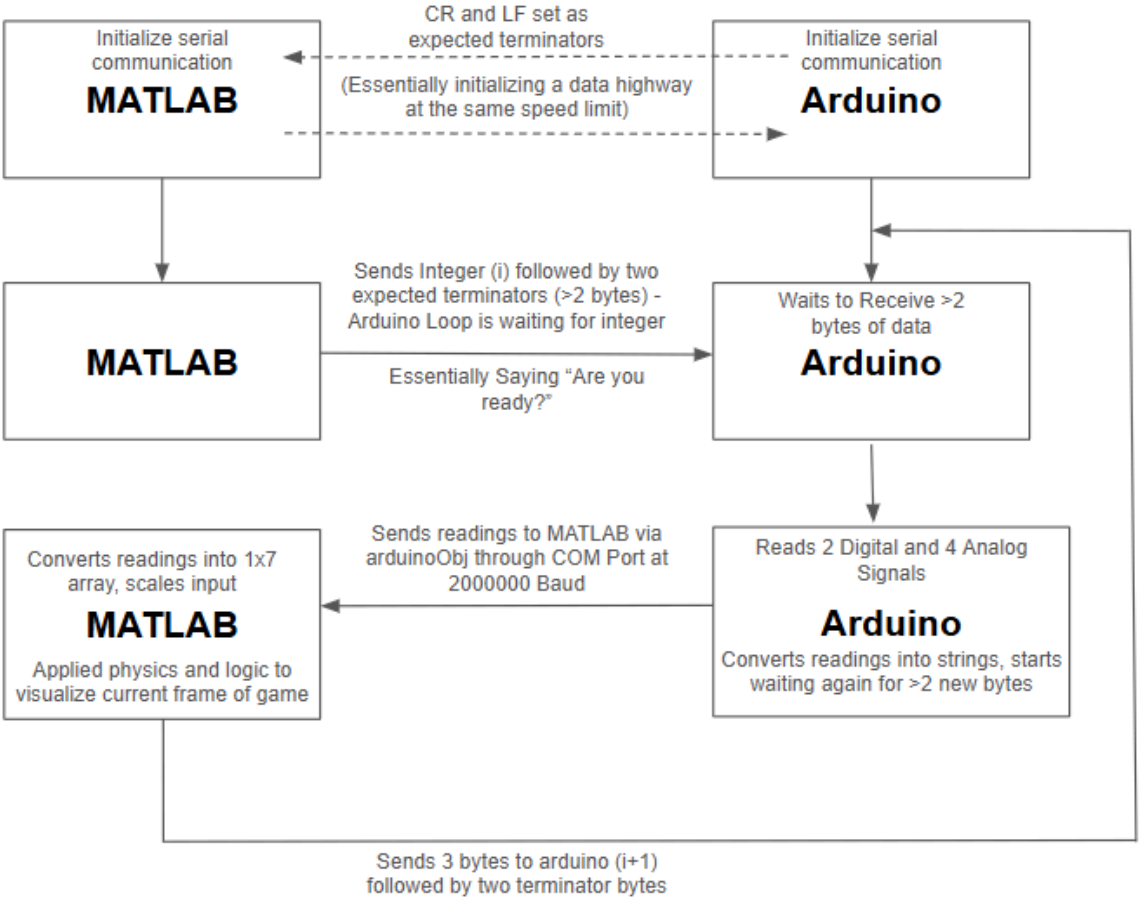
The system uses the following interesting talking points:

- Arduino/MATLAB “handshaking” (physical analog/digital input converted to control math-governed objects).
- RK4 (Runge-Kutta) differential equation modeler (Newtonian physics governed movement).
- Ergonomic improvements over the Xbox One controller.

How:
- Read in Instantaneous Arduino analog and digital input.
- Send this input data to MATLAB at (nearly) instantaneous intervals.
- MATLAB reads input data, scales and converts data to an output array.
- Use indices of the output array to control mathematically governed shapes and photos, giving the illusion of a video game.
- Run this loop again as smoothly and as fast as possible until a mathematically decided “game over” or “winner” condition is met.

Intro Concepts:

Arduino - MATLAB handshaking - how data is properly bridged and used:



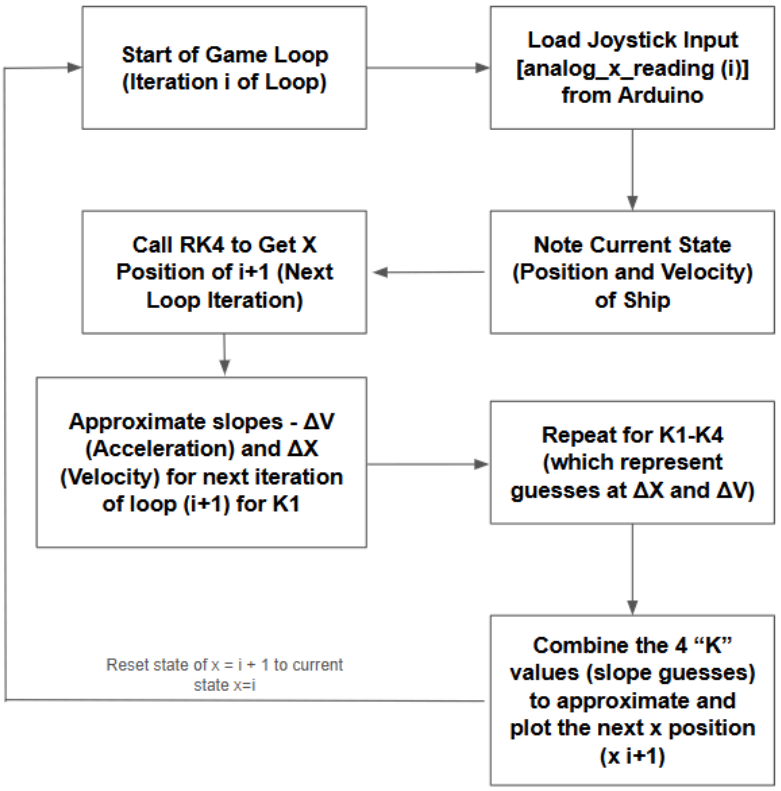
Arduino - MATLAB handshaking - Telephone/texting analogy:

(Let Arduino = Arnie and MATLAB = Matt)

- Arnie and Matt hop on the same text message website. (Initializes serial communication at the same Baud rate).
- Matt texts "Are you ready to send info?" (Sends integer + CR/LF)
- Arnie says "Yes !" and reads input information from the joystick. (Converts input signals to a 1x7 array while Matt is awaiting reply).
- Arnie then replies to Matt by texting him the 1x7 array he read. (i.e sends analog/digital voltage readings, ex = [12. 1023, -233…..]).
- Matt reads this 1x7 array and adjusts his video game to display these new readings using physics and math. (Feeds info into RK4 function, button clicking etc..).
- Matt says "are you ready to talk again?" (Next iteration of the loop starts when the Arduino is ready).

RK4 ODE Approximator:

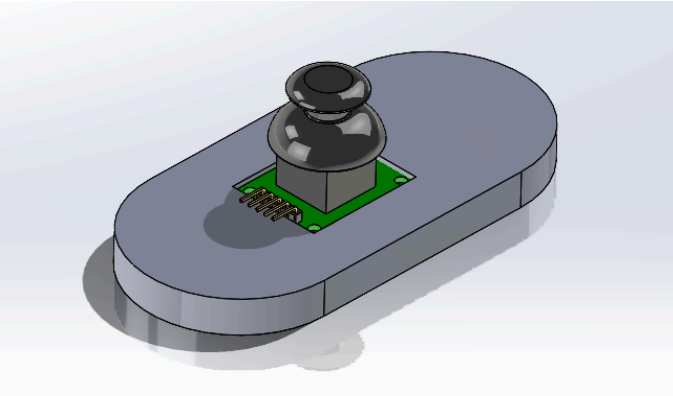
Goal: Approximate ship movement to mimic a mass-damper system (mimic a force that will slow ship down once input is removed instead of an immediate stop).



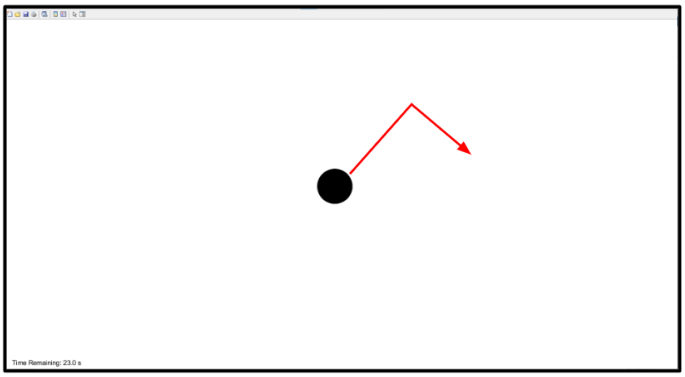
Within each K calculation:

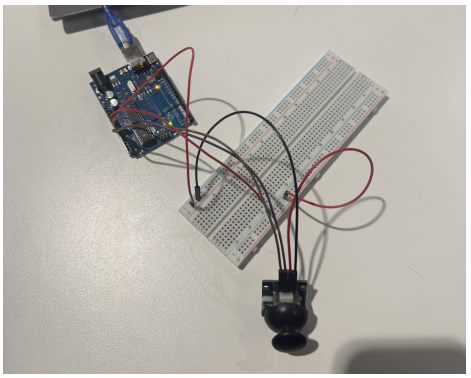
```
% Runge-Kutta (RK) fourth order method for alien's y-coordinate
function y_new = yRK4(ji, ti, yi, h, w1, w2, w3, w4, c2, c3, c4, a21, a31, a32, a41, a42, a43) %
    global screen_height; % call global variables
    k1 = h * l(ji, ti, yi); % Runge-Kutta calculations
    k2 = h * l(ji, ti + c2 * h, yi + a21 * k1); % Runge-Kutta calculations
    k3 = h * l(ji, ti + c3 * h, yi + a31 * k1 + a32 * k2); % Runge-Kutta calculations
    k4 = h * l(ji, ti + c4 * h, yi + a41 * k1 + a42 * k2 + a43 * k3); % Runge-Kutta calculations
    y_new = yi + w1 * k1 + w2 * k2 + w3 * k3 + w4 * k4; % Runge-Kutta calculations
    % check for collision with screen edges
    if y_new(1) < 0 || y_new(1) > screen_height % check if the y position of the alien
                                                % is past the screen height
        y_new(2) = -y_new(2); % reverse velocity if alien is at the edge of the screen
    end
end


% function containing the differential equations dx/dt=f(t,x) to solve for alien's y-coordinate
function dydt=l(j,~,y) % function input variables
    global m c; % bring global variables that were defined at beginning of code
    dydt(1,1)=y(2); % mass drag system for y
    dydt(2,1)=-c/m*y(2) + j/m; % mass-drag system in zero-g with applied force u
end
```

Game Version/ Design 1:



- Original design
- 1 Thumbstick (2 Analog inputs only)
- No digital input (no clicking)
- No ergonomic consideration
- Adjusted for stick drift (hardware adjustment)

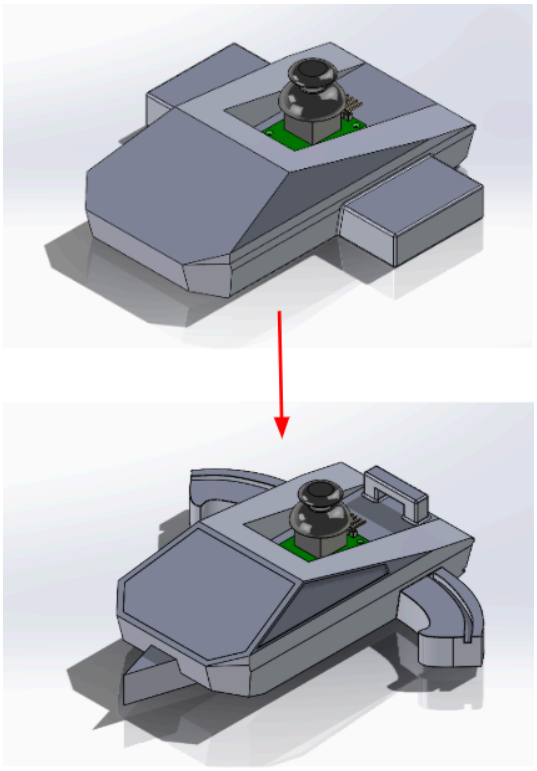




Capabilities:

- Built working and error-proof connection between Arduino and MATLAB using “Handshaking” method (sending bits/arrays at extremely fast intervals with a common Baud rate, data is only sent when either software states it is “ready”).
- 2 Analog Inputs
- Circle is controlled by governing RK4 differential equation modeling, mimicking Newtonian physics (the circle has a mass and a damping constant).
- Adjusted for hardware discrepancies within the software (stick drift, physical 1080/1920 figure window display).

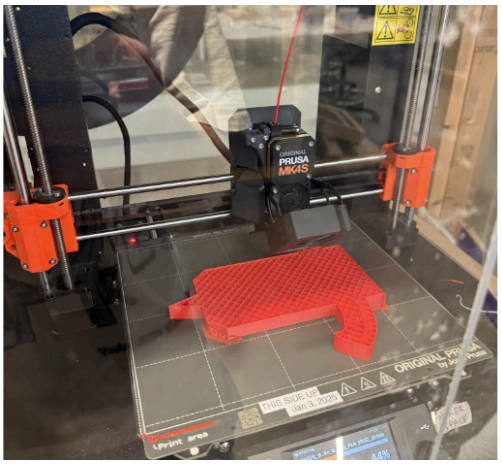
Game Version/Design 2:

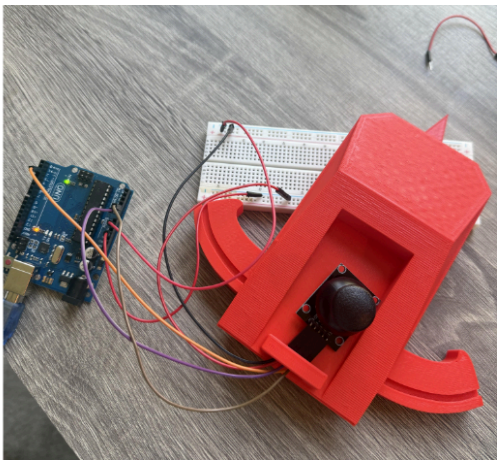


- Developed Cyber Truck inspired controller geometry. Logic was to build something at the cross section of creative and ergonomic.

- Added Tesla logo as controller handles.

- Thought this design would be somewhat ergonomically viable - I was wrong.

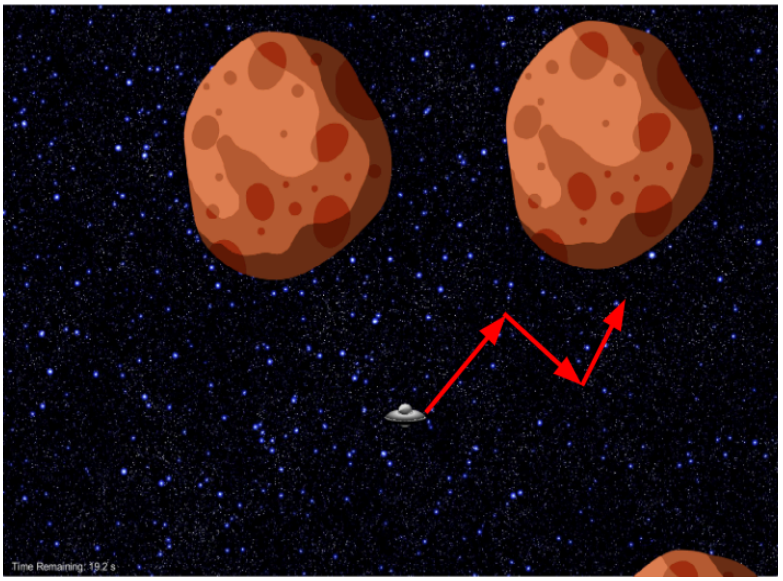




V2 Hardware Improvements:

- Improved hands-on design.
- Cyber Truck inspired 3D printed controller (with Tesla Logo to hold).
- Something felt off - holding a 2-handed controller with one joystick was awkward.
- Breadboard left floating - leaving cable management as a mess.
- Limited to only one degree of freedom moving virtually.

V2 Software Improvements:



- Photos attached to moving objects, creating space evaders.

- Obstacles added and “Game Over” logic created.

- Digital button press (sends “0” or “1” input to MATLAB) to end the game.

- If the current (invisible) radius of the alien ship overlaps the radius of asteroid = game over.

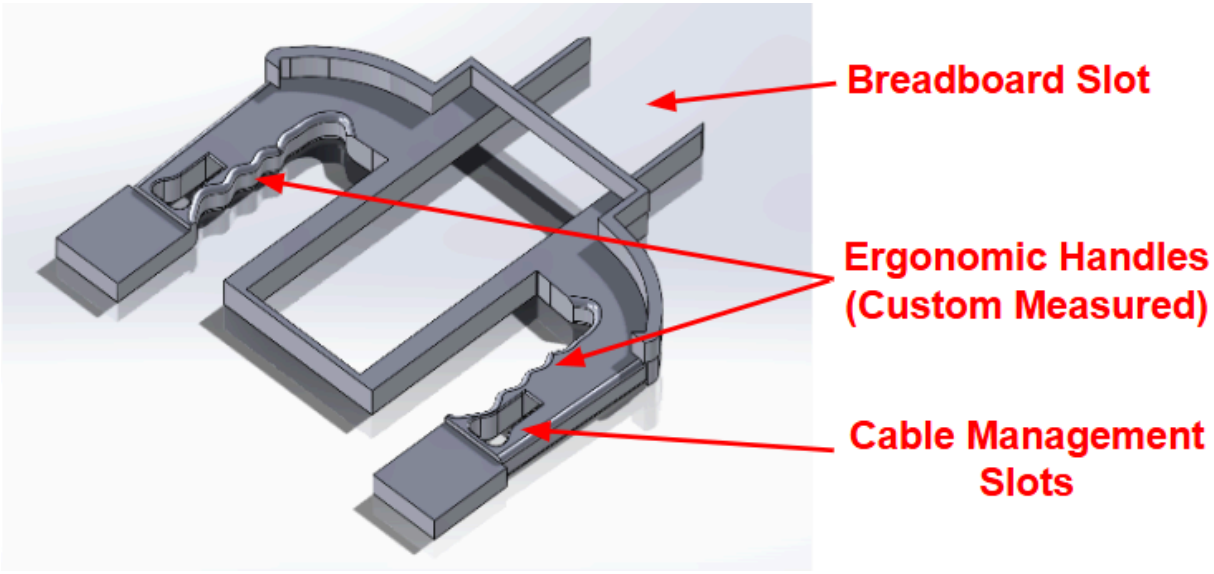
- Asteroids spawn at random intervals between a set time, also spawn at random x coordinates at y = 1080.

Required Improvements for Hardware/Software V3:

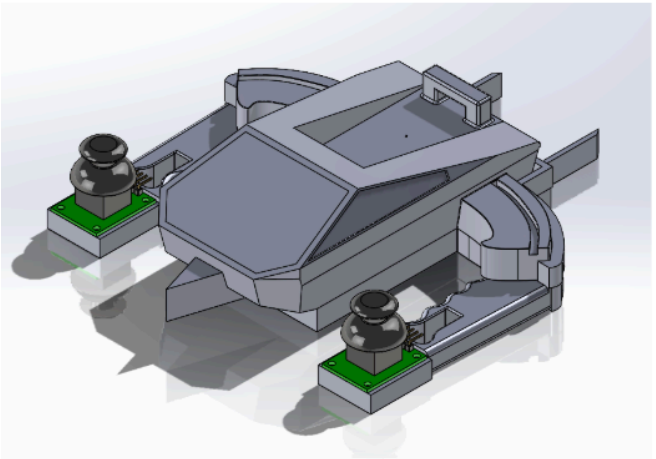
- Add another analog/digital input (joystick) to allow increased interactivity.
- Improve ergonomics by designing and manufacturing two ergonomic joystick handles.
- Design slot for circuit breadboard integrated into this new section of controller.
- New game that incorporates these two joysticks interactively.
- Decrease lag.

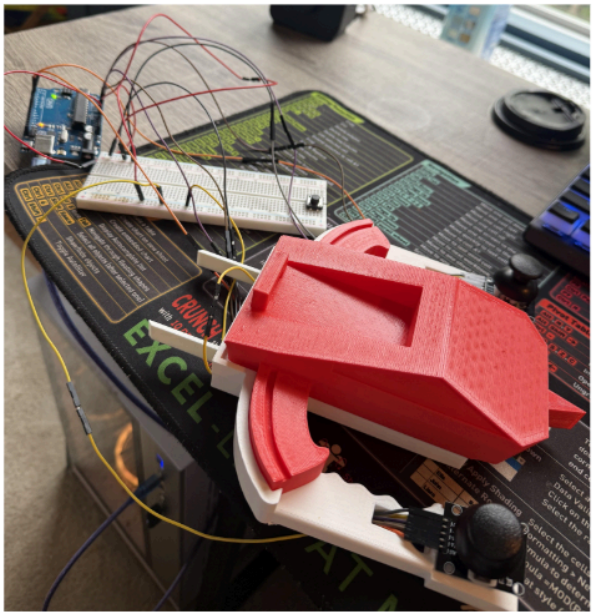
Game Version/Design 3:

Hardware V3:

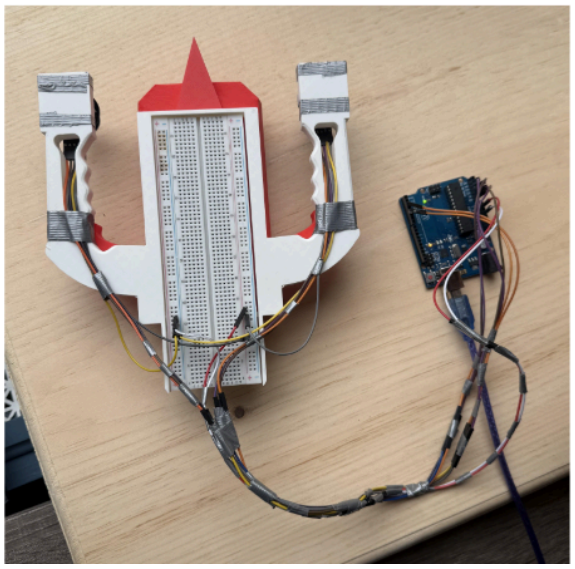


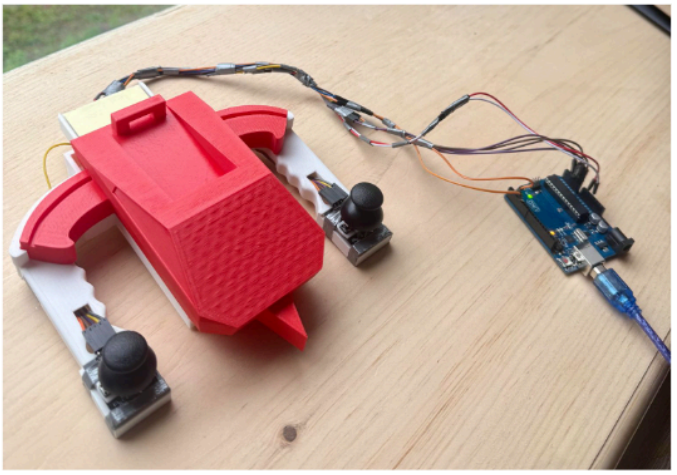
- Designed and manufactured lower controller section to mesh with the existing design.
- Optimized cable management by adding cable slots near thumbsticks.
- Added on-component breadboard slot to streamline visual appeal (make it physically feel/look more like a real video game controller).





Hardware V3 Continued:





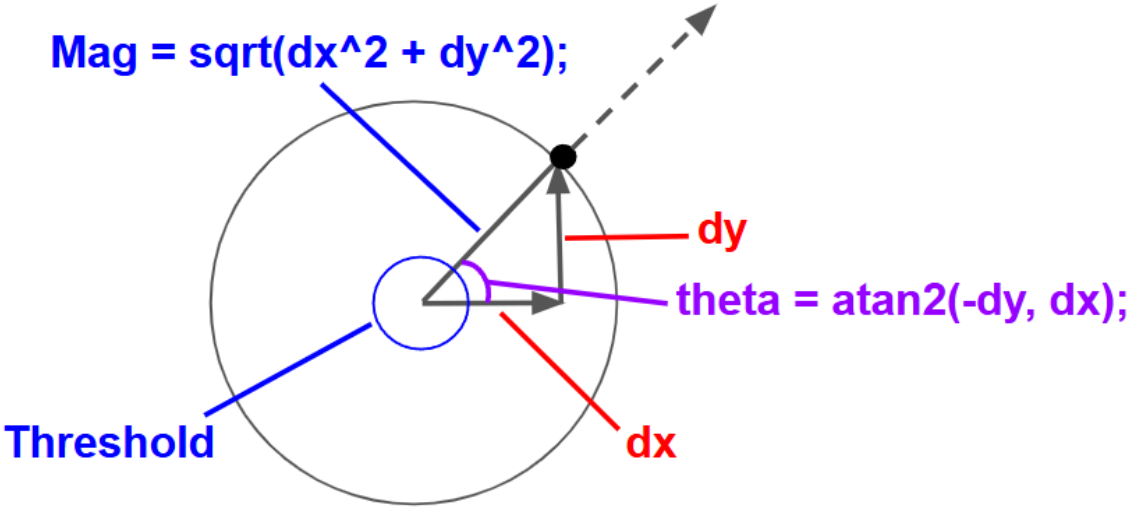
Software V3:

- Needed to develop a game that now takes 4 analog inputs and two digital inputs - and each input has a respective (productive) output.

- Obstacle circles added as plotted shapes to decrease lag.
- Main menu and game over menus created to call game functions.
- Right stick controls Newtonian-governed movement of spaceship, left stick controls trigonometrically-governed aim and obstacle shooter.

Software V3 continued:

- Added aim circle based on trigonometry.
- Aim circle is plotted for each instance of the loop, only if the Magnitude of the analog input is greater than the defined threshold (required or the aim circle will attempt to aim when your thumb is off the joystick).




```
Raw Arduino Data: 140,519,490,0,497,0,0
dy = -6, dx = -6036, magnitude = 6036.003
Threshold Met! Current angle is: 3.1406 radians
```

- Balls shoot normally from the magnitude line when the left digital stick is pressed (when the value of input_array(7) = 1, and its last value in the loop was zero).
- Player points increase when the shot ball hits the obstacle ball.

Areas for Improvement (always looking to Improve):

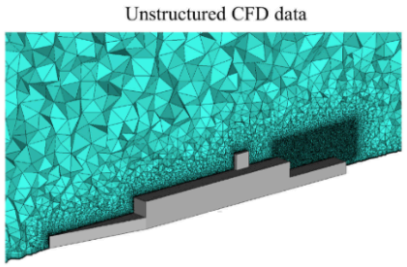
- Add a 2-person air hockey game (essentially placing photos on top of pong) by controlling two objects. Each side is an object controlled by a single player.
- Add a 3-D printed cable management sleeve for the bottom of the controller.
- Manufacture custom joysticks for ergonomic improvement.

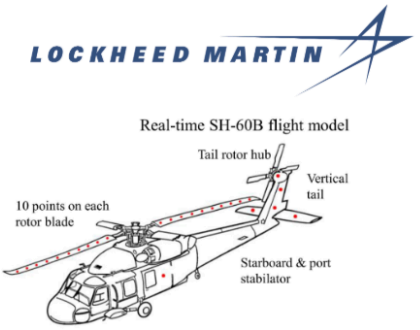
Other Cool Projects:

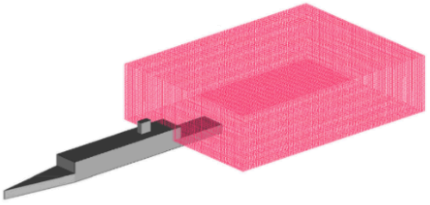
1. Canada's CSC project: Heliops landing data logic board:

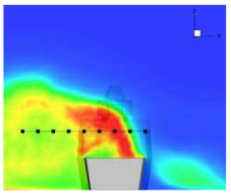
Problem: Did not know if/if not we could land a helicopter within the following variables:

- Different headwind angles.
- Different Turbine Engine throttles.
- Different Arrangements of active Turbine Engine Outtakes.









Constraints: Knew only the following Variables:

- Turbine Engine Uptake Mass Flow at 100% throttle.
- Turbine exhaust temperatures at 100% throttle.
- Turbine Engines that must be turned on at 7 different headwinds.
- Uptake mass flow rate when each of 3 Turbines are on at once.

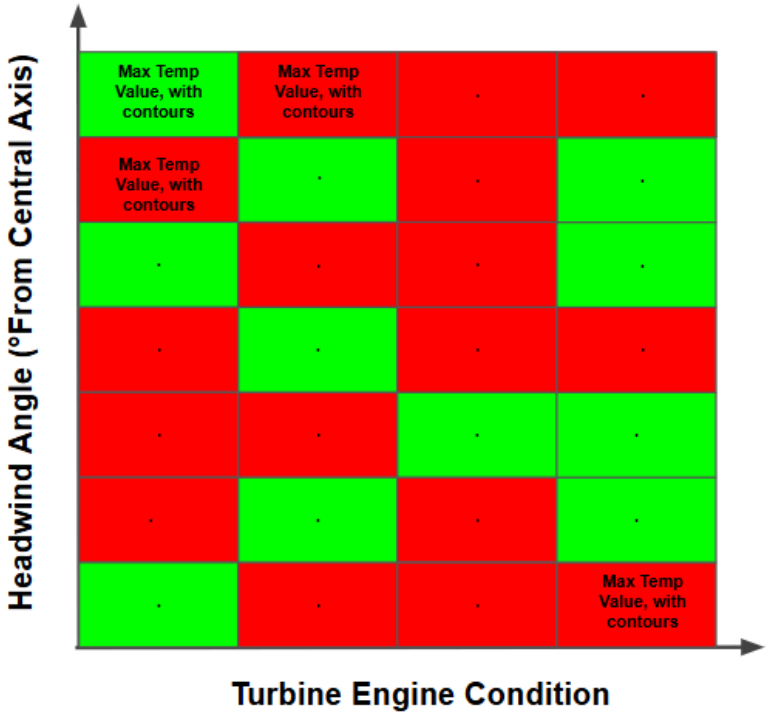
What I did:

- Applied thermodynamic fundamentals to calculate exhaust mass flow rate(s) for different scenarios of Turbine engine on/off conditions.

$$\dot{m}_{\text{exhaust}} = \rho_{\text{exhaust}} \; A_{\text{exhaust}} \; v_{\text{exhaust}}$$

$$\dot{m}_{\text{exhaust}} = \underbrace{\frac{p_e}{R_{\text{gas}}\, T_{\text{exhaust}}}}_{\rho_e} \times A_{\text{exhaust}} \times \underbrace{\sqrt{2\, c_p \left(T_{\text{exhaust}} - T_{\text{ambient}}\right)}}_{v_e}$$

- 7 headwind angle simulations for 4 different uptake conditions different RANS simulations on full-throttle to deliver 28 contours - covering all possible conditions.
- Used these contours to present max temperature over the flight deck for different scenarios. X (4 values) was Engine throttle condition and Y (7 values) was headwind angle.



- Provided Vendors with concise logic board of different wind and engine conditions

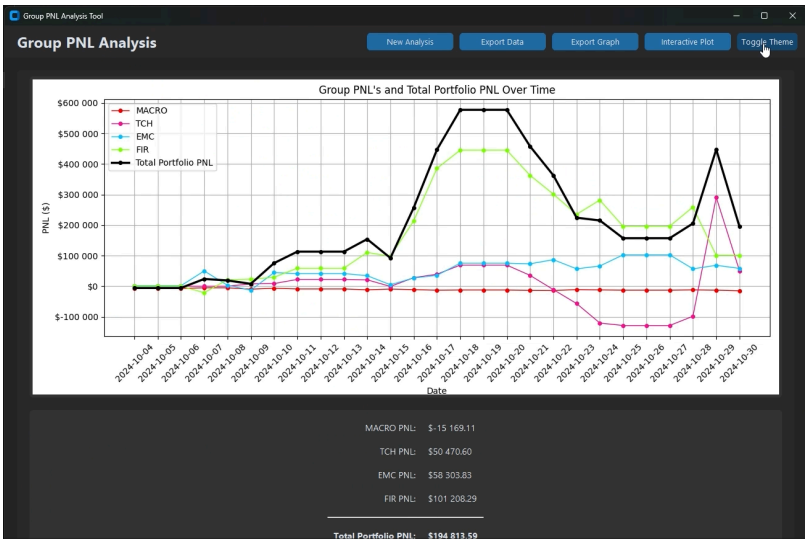
- Vendors could use this data table to validate (with their simulations and flight conditions) if or if not the helicopter could land.

2. Quantitative Analyst at Dalhousie Investment Society: Club PNL tracker app:

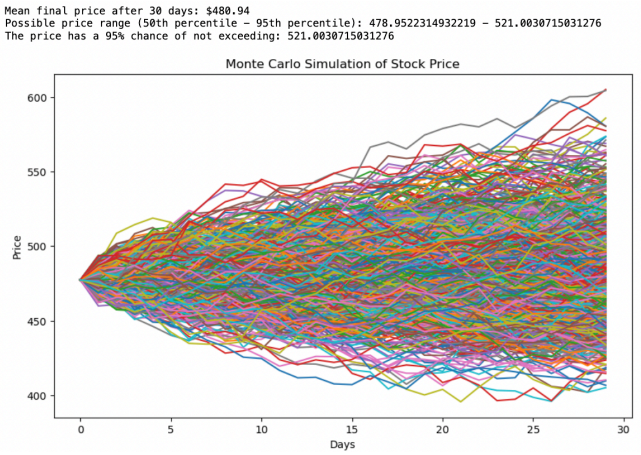
Fundamental issue with Investment Society: **They did not know their PNL**. Why? Was not viable/easily accessible with Bloomberg Terminal.

Steps:

- Team-developed Python script to pull Bloomberg Trade tickets into excel.
- Pre-allocated equations determined each trades' PNL for each of the four teams (Commodities, Global Macro, Real-Estate, Forex).
- Using the PNL of each trade, PNL calculator pulls in Raw data and calculates daily PNL's.
- Daily PNL's are calculated and graphed linearly for each team, and entire portfolio.

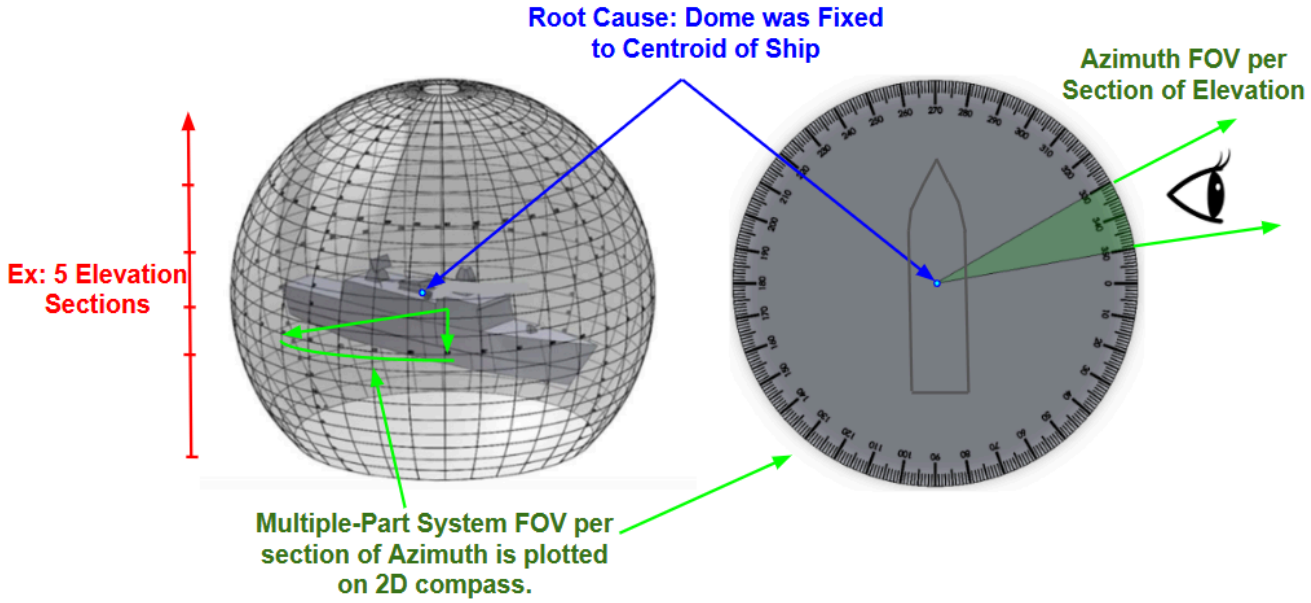


Weekly quantitative tool presentations:

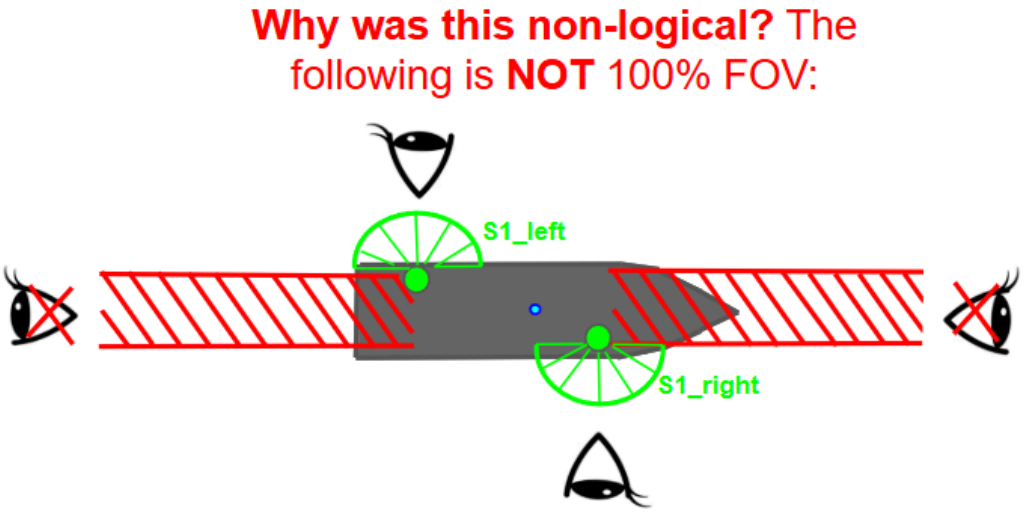


3. Lead of Redevelopment of Quick Look Analysis at Lockheed Martin:

Logic Problem that was overlooked for years:



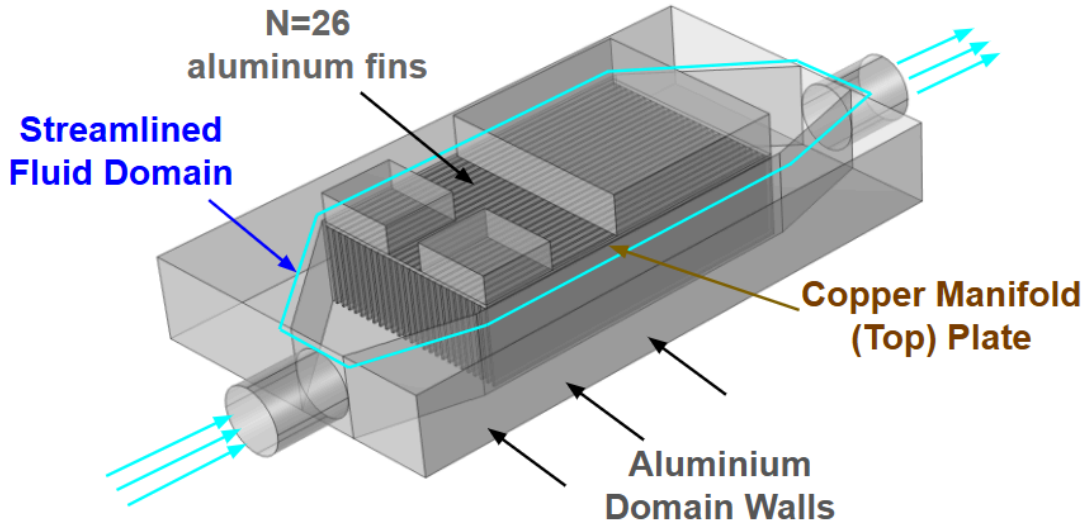
Fixing the Dome (Left) measuring tool on the center of the ship led to multiple discrepancies.



Developed a proposal, explained new logic to Senior Engineers. New proposed method was introduced and is still in practice today.

4. Highest Competition Efficiency - Liquid Chip Cooling Design at Dalhousie University:

Increased chip efficiency (amount of heat dissipated) by ~350%



	Heat Generated Through Chip	% Increase of Original Design	Maximum Temperature
Chip 1	426.8 W	533.5% (of 40W)	84.963 °C
Chip 2	142.72 W	178.4% (of 80W)	84.973 °C
Chip 3	138.2 W	345.5% (of 80W)	84.975 °C
Total	707.72 W	353.9% (of 200W)	-

<u>In My Free Time (Outside of Engineering):</u>

- **Head of Governance and Trading at Dalhousie Blockchain Society**
- **Engineering vs Commerce Charity Hockey Game**
- **Competitive Jiu-Jitsu**
- **Bogey golfer (at best)**