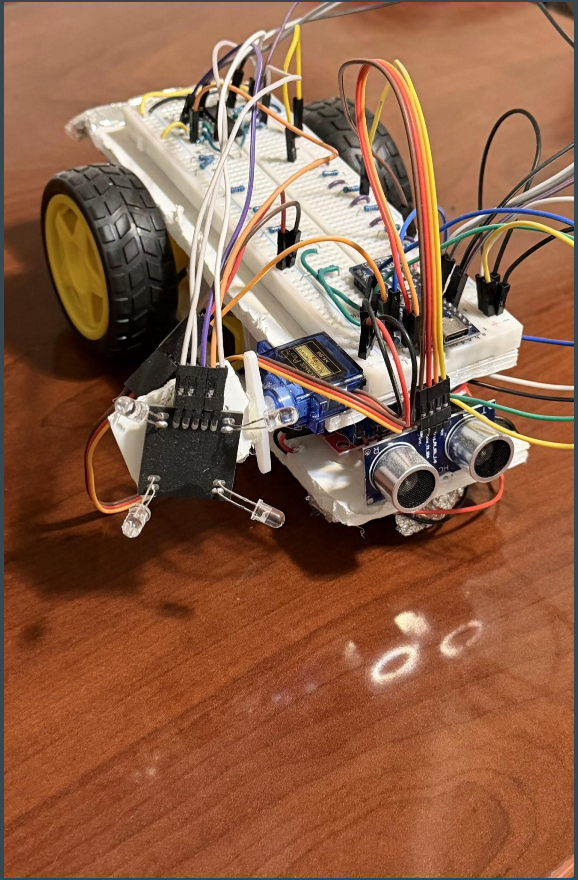
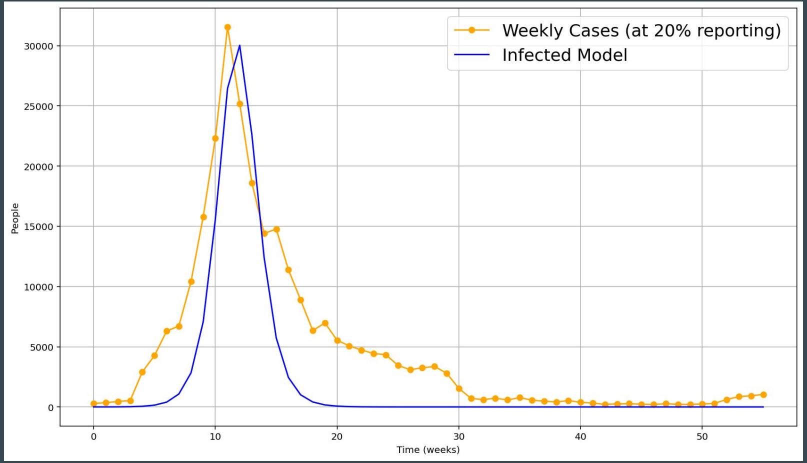
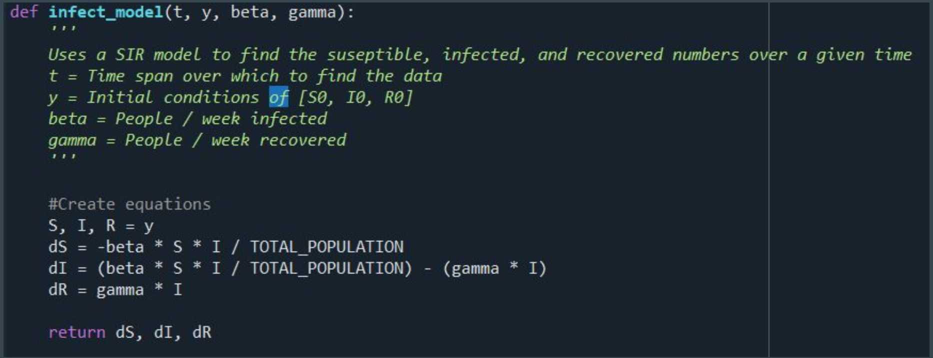


Engineering Portfolio

Zander Gonzales
zandergonzales02@gmail.com

Influenza Prediction Model





```
def infect_model(t, y, beta, gamma):
    '''
    Uses a SIR model to find the suseptible, infected, and recovered numbers over a given time
    t = Time span over which to find the data
    y = Initial conditions of [S0, I0, R0]
    beta = People / week infected
    gamma = People / week recovered
    '''

    #Create equations
    S, I, R = y
    dS = -beta * S * I / TOTAL_POPULATION
    dI = (beta * S * I / TOTAL_POPULATION) - (gamma * I)
    dR = gamma * I

    return dS, dI, dR
```

Created a Python script that used a dynamic system in order to take in influenza data and find physical constants to predict general epidemiological behavior

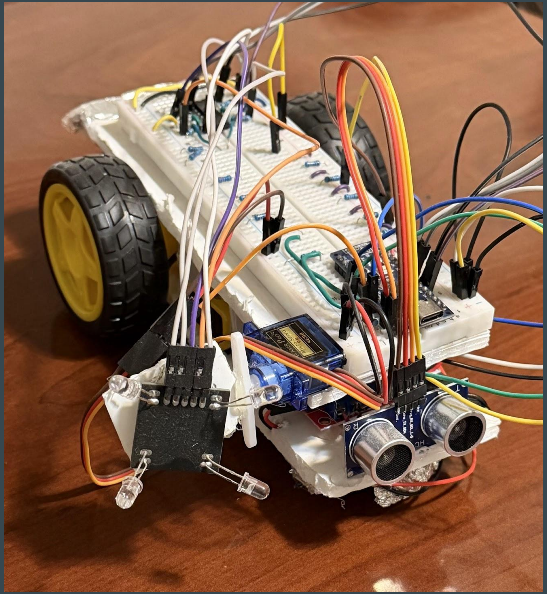
- Used a double for loop to sort through a grid of possible constant values
- Used mean-squared error to calculate how close the model was following
- Utilized real values from publicly available csv files

Link to the project code

Light Tracking Robot

Created a robot, as well as code to control it, that would follow light. It also had built in collision avoidance and speed control changed through a capacitive touch sensor.

- Physical Circuit Design: Put together and tested the circuits using electrical components
- Digital Code Design: Created a finite state machine that used logic and sensor data to autonomously follow light



Link to functional robot showcase

Ideal Bowling Ball Material Selection

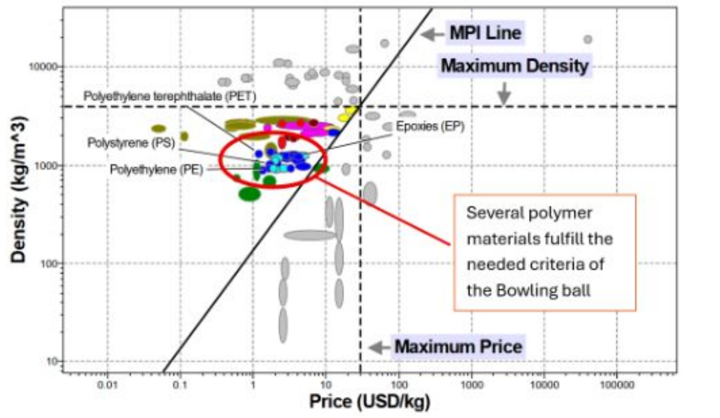
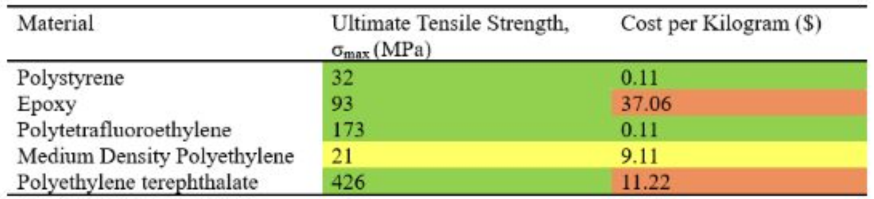


Figure 5: Ashby Chart relationship between Density and Price. Chart generated using Granta EduPack Software.

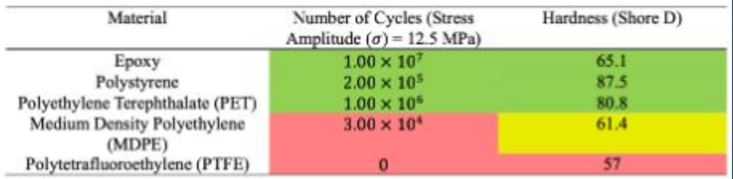
Using material property considerations, determined reasonable constraints for bowling ball requirements and compared different materials to find the optimal material, and displayed these properties in created figures and tables.

Table 1: Material performance indexes. Cost data from Materials Science and Engineering textbook.



Material	Ultimate Tensile Strength, σ_{max} (MPa)	Cost per Kilogram ($)
Polystyrene	32	0.11
Epoxy	93	37.06
Polytetrafluoroethylene	173	0.11
Medium Density Polyethylene	21	9.11
Polyethylene terephthalate	426	11.22

Table 2: Number of cycles and hardness. Data from matweb and specialchem.com

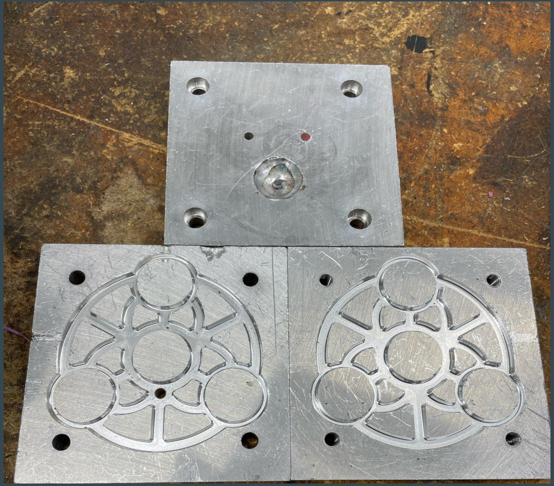


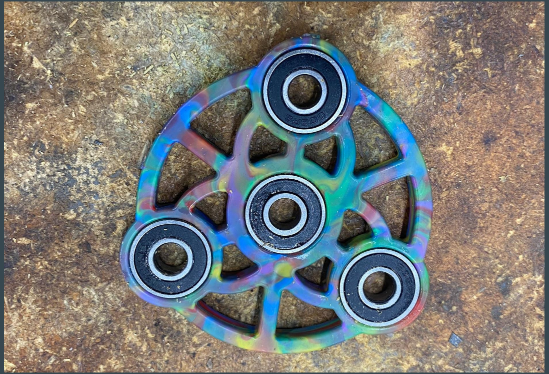
Material	Number of Cycles (Stress Amplitude (σ) = 12.5 MPa)	Hardness (Shore D)
Epoxy	1.00×10^7	65.1
Polystyrene	2.00×10^5	87.5
Polyethylene Terephthalate (PET)	1.00×10^6	80.8
Medium Density Polyethylene (MDPE)	3.00×10^4	61.4
Polytetrafluoroethylene (PTFE)	0	57

Custom Injection Molded Fidget Spinner

Used 3D modeling to design a fidget spinner model that would fit in bearing, and then used that to machine a mold and injection mold the final product.

[Link to documentation]





Skills used:

- CAD Modeling
- FEA Analysis
- Machining
- Injection Molding

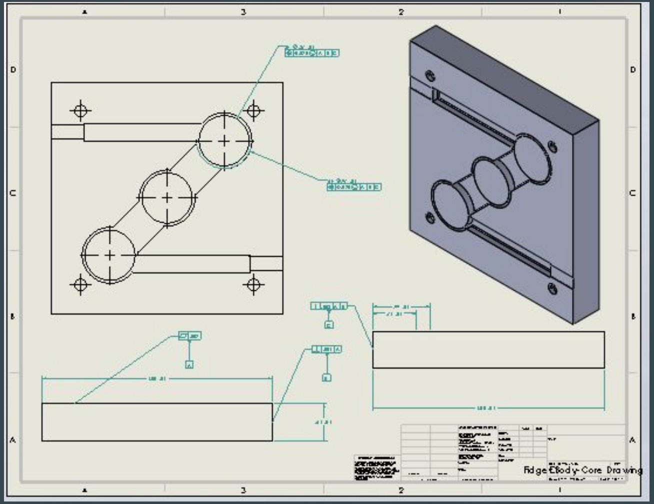
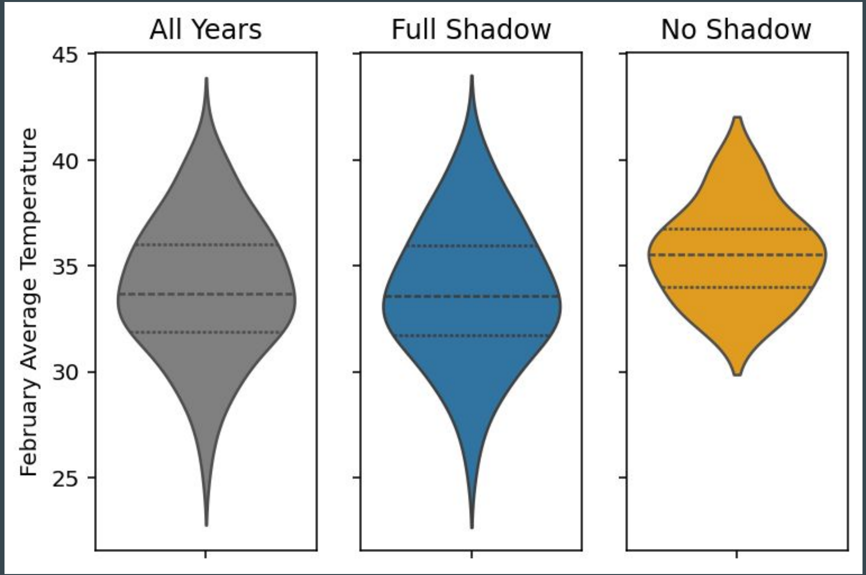


Photo of earlier original design

Statistical Analysis of Punxsutawney Phil's Predictions

Did a one sided mean comparison hypothesis test in order to look at the accuracy of Punxsutawney Phil's predictions. By looking at 118 years of data regarding both the prediction, as well as weather data from NOAA's National Climatic Data Center, I was able to see that him not seeing a shadow had a statistical significance in having warmer weather when no shadow was seen.



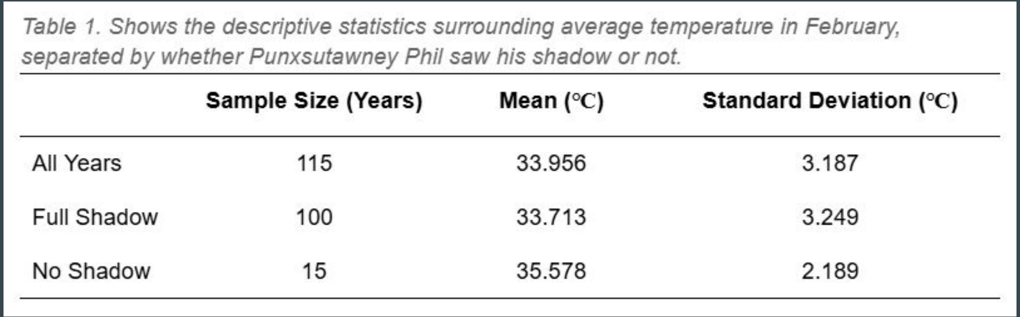


Table 1. Shows the descriptive statistics surrounding average temperature in February, separated by whether Punxsutawney Phil saw his shadow or not.

	Sample Size (Years)	Mean (°C)	Standard Deviation (°C)
All Years	115	33.956	3.187
Full Shadow	100	33.713	3.249
No Shadow	15	35.578	2.189

Calculating Coefficient of Drag and Static Friction

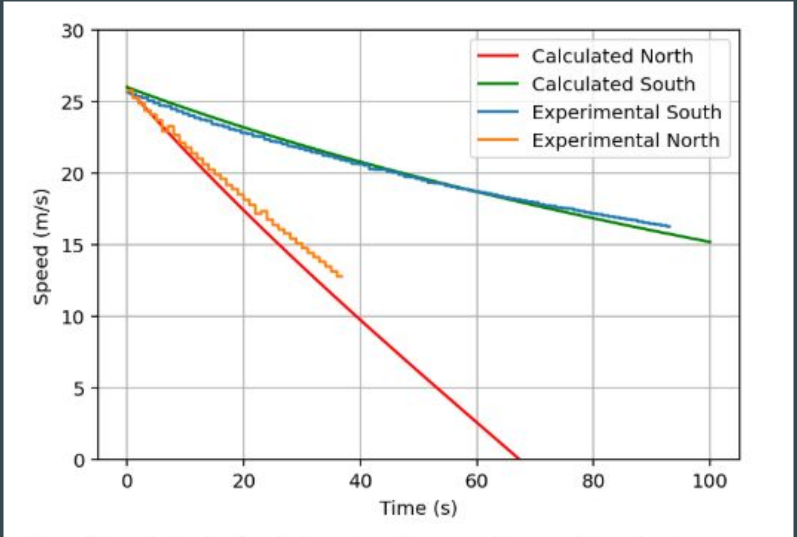
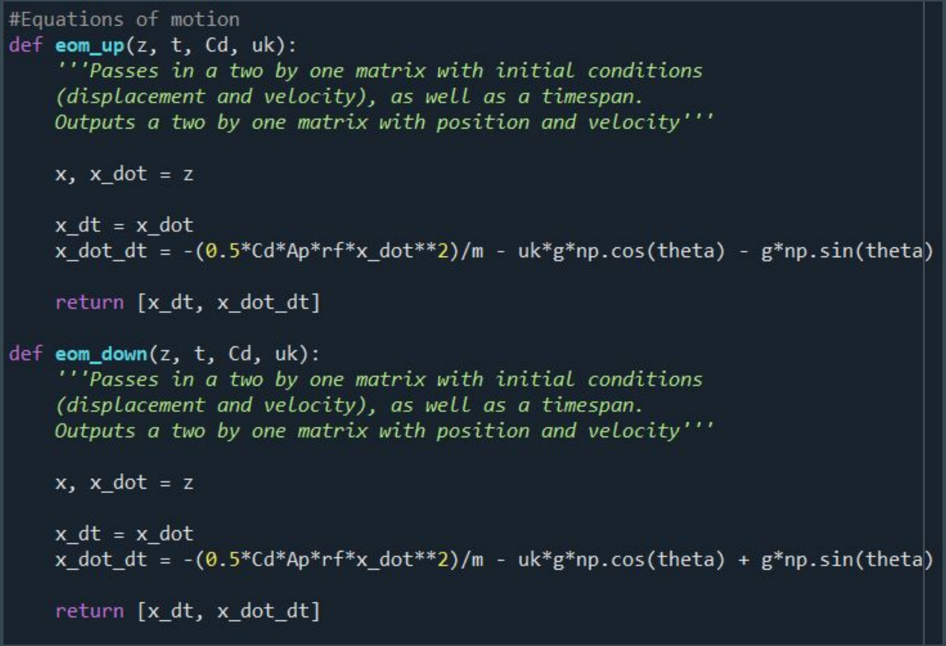


Figure 1. A graph showing the velocity vs time of the car coasting up and down the slope, including both experimental car data and calculated car data.



```
#Equations of motion
def eom_up(z, t, Cd, uk):
    '''Passes in a two by one matrix with initial conditions
    (displacement and velocity), as well as a timespan.
    Outputs a two by one matrix with position and velocity'''

    x, x_dot = z

    x_dt = x_dot
    x_dot_dt = -(0.5*Cd*Ap*rf*x_dot**2)/m - uk*g*np.cos(theta) - g*np.sin(theta)

    return [x_dt, x_dot_dt]

def eom_down(z, t, Cd, uk):
    '''Passes in a two by one matrix with initial conditions
    (displacement and velocity), as well as a timespan.
    Outputs a two by one matrix with position and velocity'''

    x, x_dot = z

    x_dt = x_dot
    x_dot_dt = -(0.5*Cd*Ap*rf*x_dot**2)/m - uk*g*np.cos(theta) + g*np.sin(theta)

    return [x_dt, x_dot_dt]
```

Took real data from a car coasting both up and down hill, and used a dynamic system to model and solve for two separate variables, as well as calculating error.