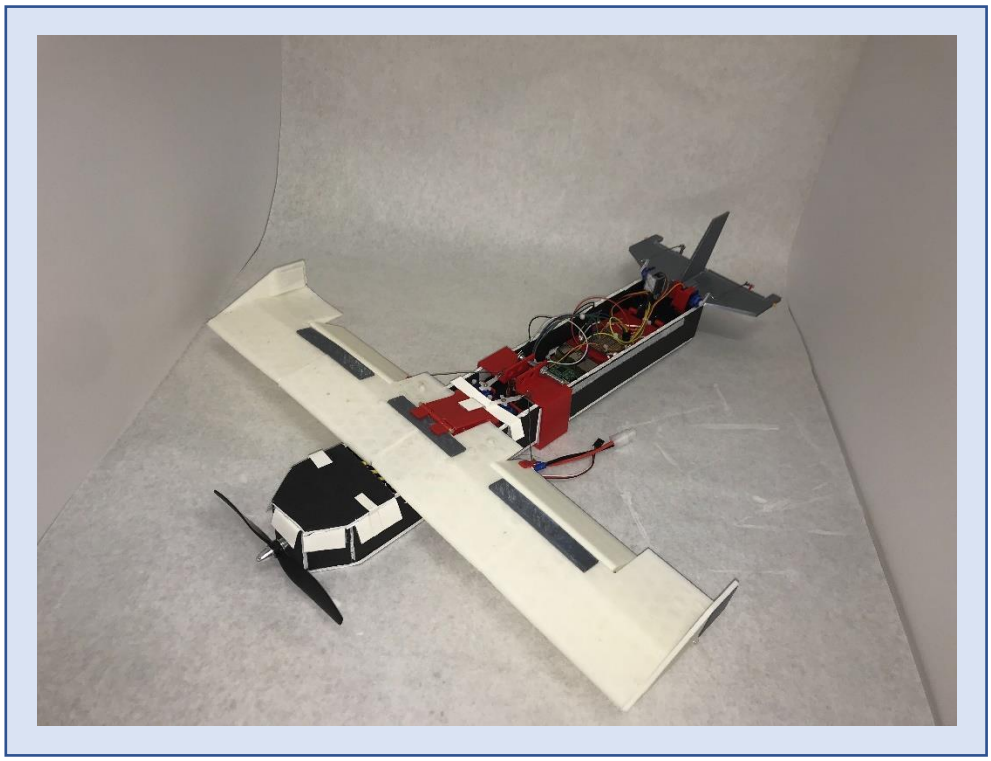
MODEL AIRPLANE



MAKER PROJECT TECHNICAL DOCUMENTATION

Written by Reyna Ayala

Beaverton High School

Class of 2021

OVERVIEW

MAIN OBJECTIVE

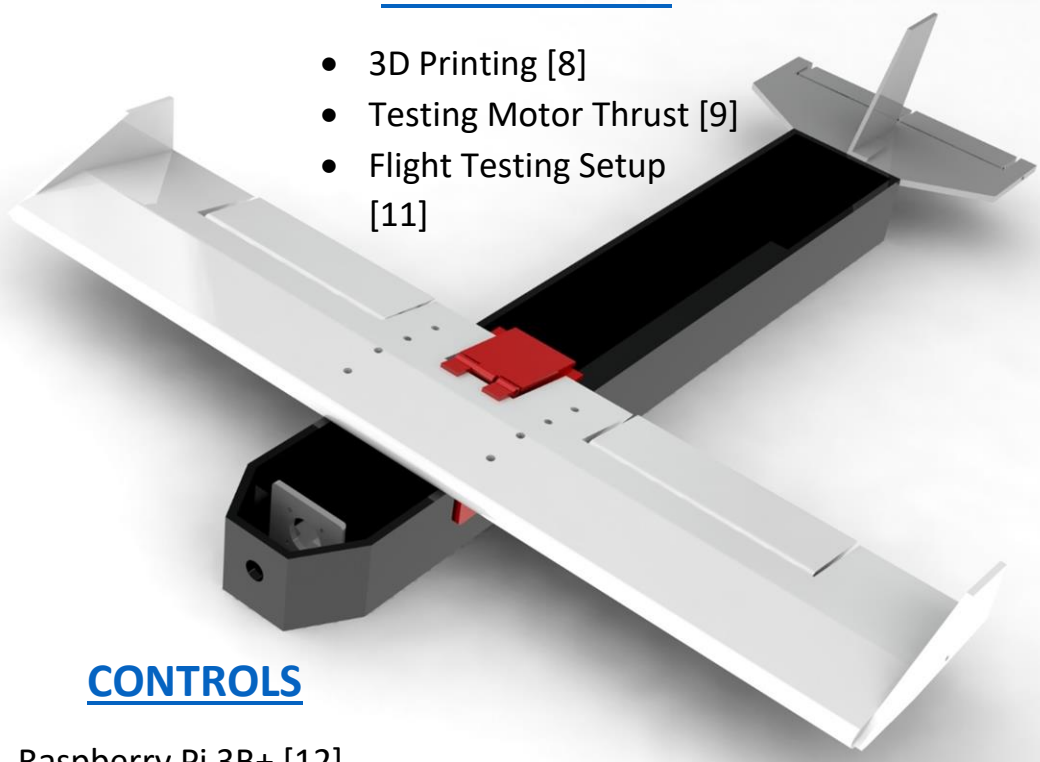
Design, build, test, and program an autonomous model airplane to learn how aeronautical design is influenced by fluid dynamics.

DESIGN PROCESS

- Aerodynamics Learning [3]
- CADD Modeling of Original Plane Design [5]
- Airflow Simulator [6]

PROTOTYPING

- 3D Printing [8]
- Testing Motor Thrust [9]
- Flight Testing Setup [11]



CONTROLS

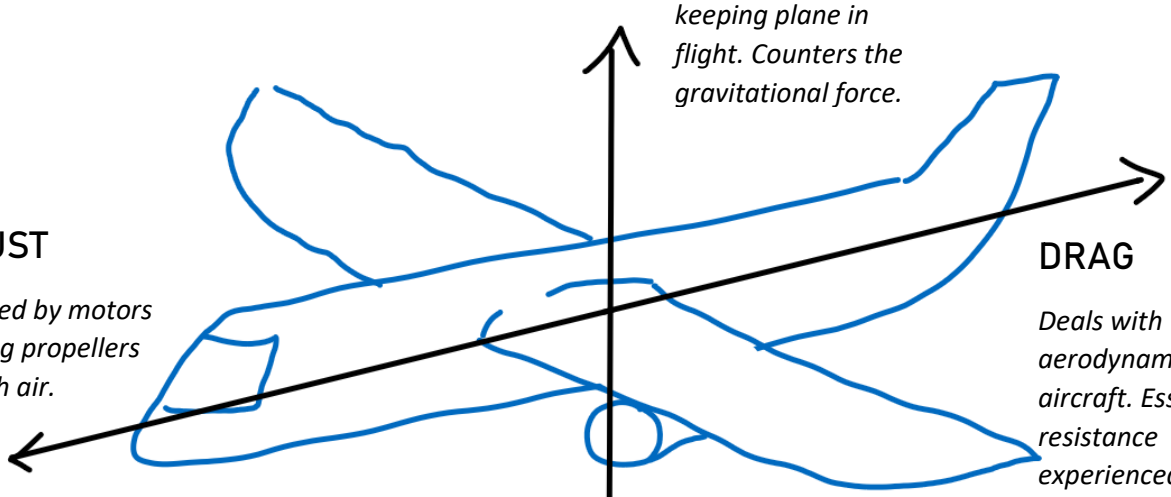
- Raspberry Pi 3B+ [12]
- Object-Oriented Python [12]
- Custom Circuit Board [13]
- Vision Code [16]
- Control Theory [17]

FINAL PRODUCT

- Reflection & Conclusion [19]
- Unexpected Obstacles [19]
- Improvements for the Future [20]
- Fun Fact [21]

DESIGN PROCESS

AERODYNAMICS LEARNING



LIFT

Force responsible for keeping plane in flight. Counters the gravitational force.

THRUST

Produced by motors spinning propellers through air.

DRAG

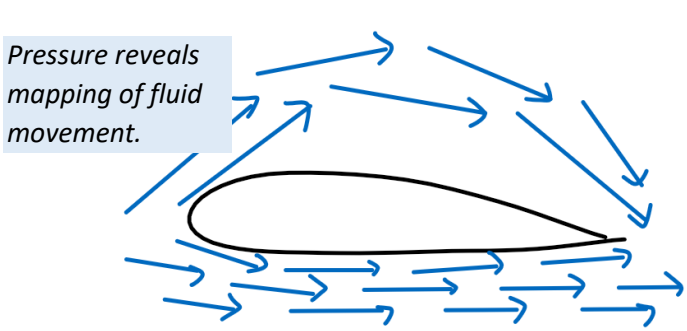
Deals with aerodynamics of aircraft. Essentially air resistance experienced by surface of vehicle.

WEIGHT

Also known as gravitational force experienced at the center of mass of the plane.

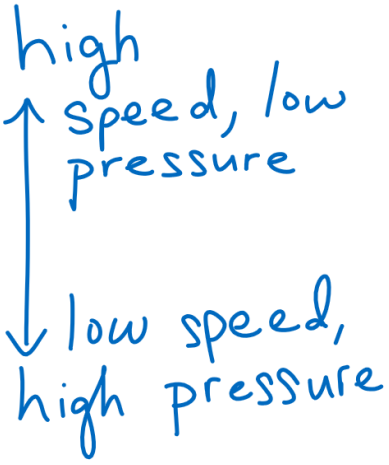
*Force vectors in diagram above are not to scale.

Airfoil Pressure Variation (Wing Side-Profile):

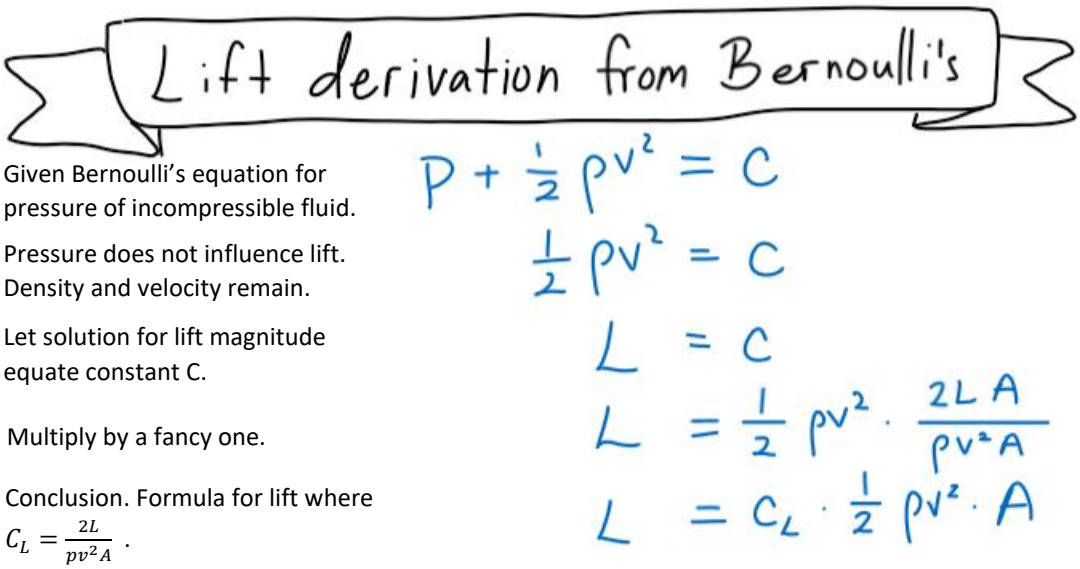


Pressure reveals mapping of fluid movement.

Lift is generated by air accelerating downward from top of airfoil – producing equal & opposite force!



By piecing together information from different sources, I created the proof below:



Given Bernoulli's equation for pressure of incompressible fluid.

Pressure does not influence lift. Density and velocity remain.

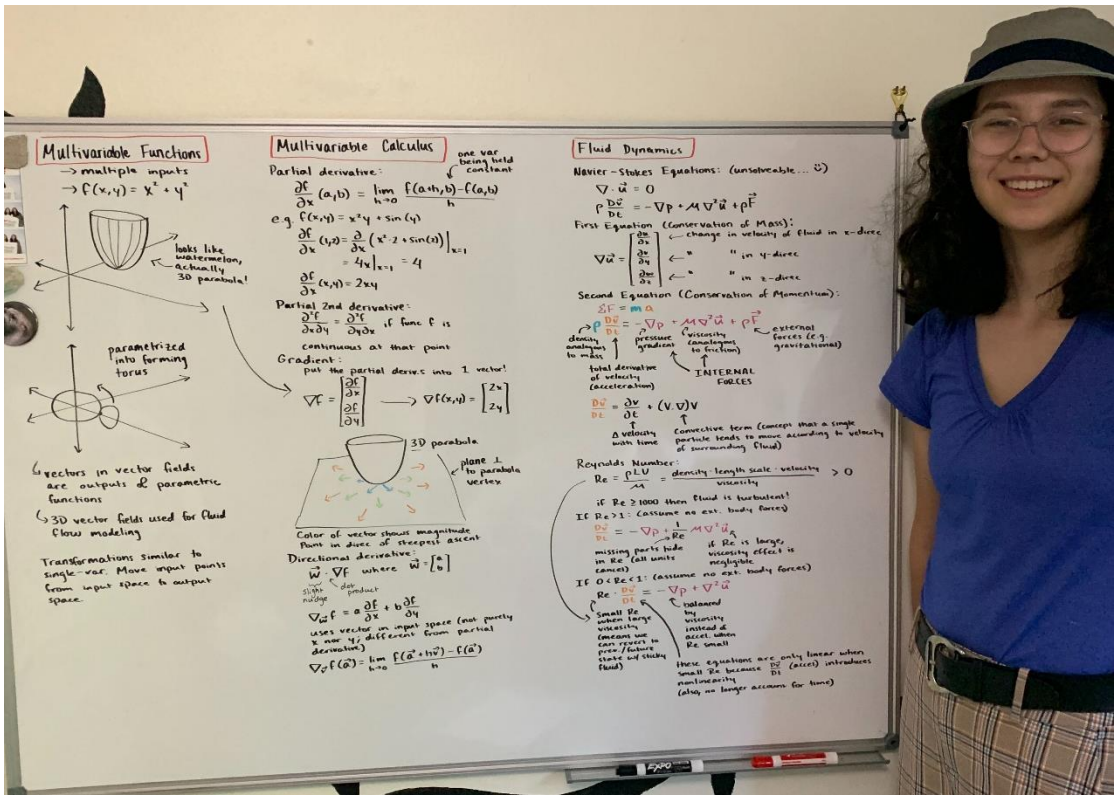
Let solution for lift magnitude equate constant C.

Multiply by a fancy one.

Conclusion. Formula for lift where $C_L = \frac{2L}{\rho v^2 A}$.

Main Source: https://web.mit.edu/16.00/www/aec/flight.html

Fluid dynamics involves a lot of advanced college-level math! In order to understand processes like calculating velocity distribution around an airfoil, I began to teach myself multivariable calculus:



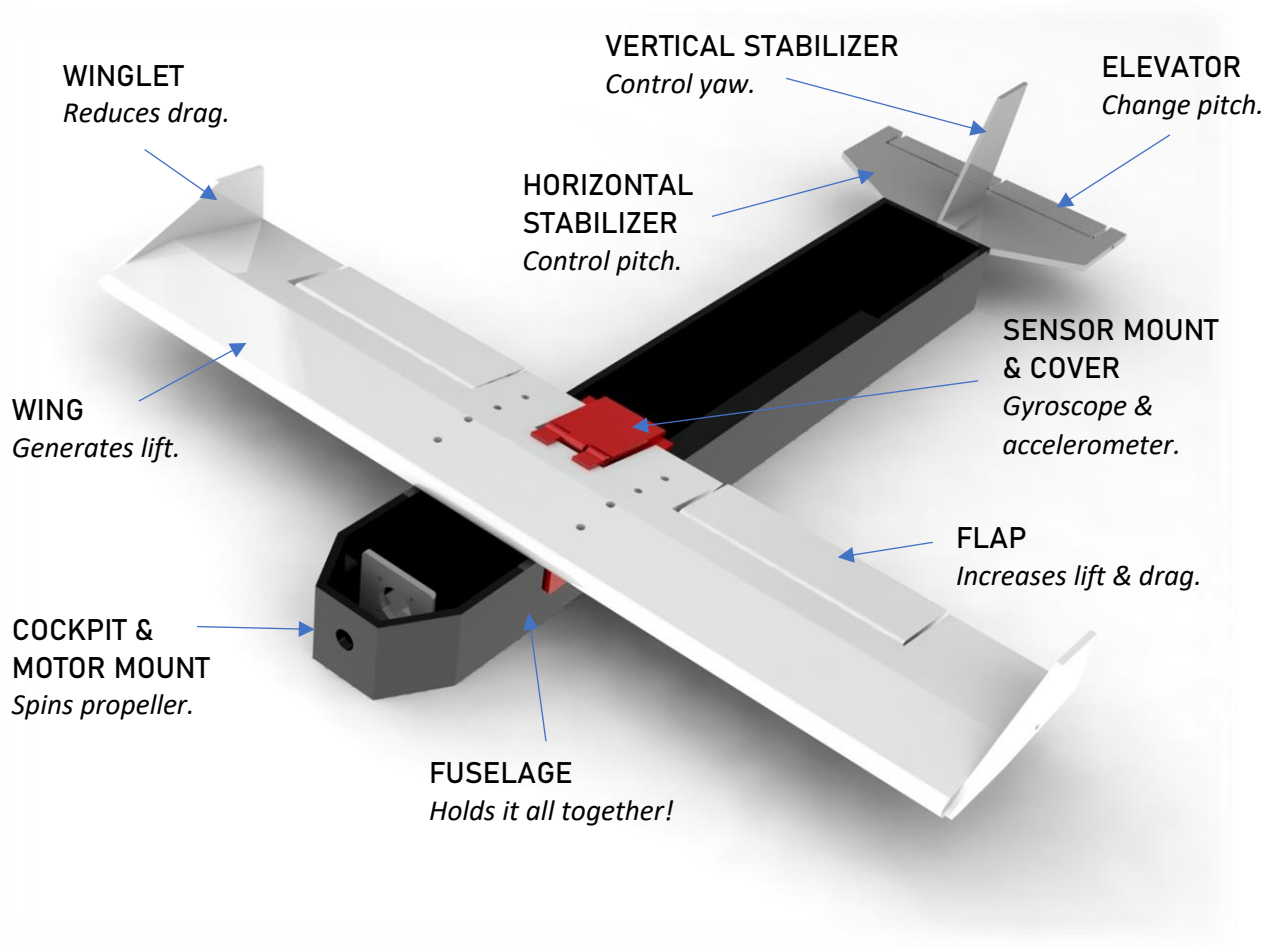
CADD MODEL

CADD: Computer-Aided Drafting & Design

- Used for creating 3D models & geometry of figures

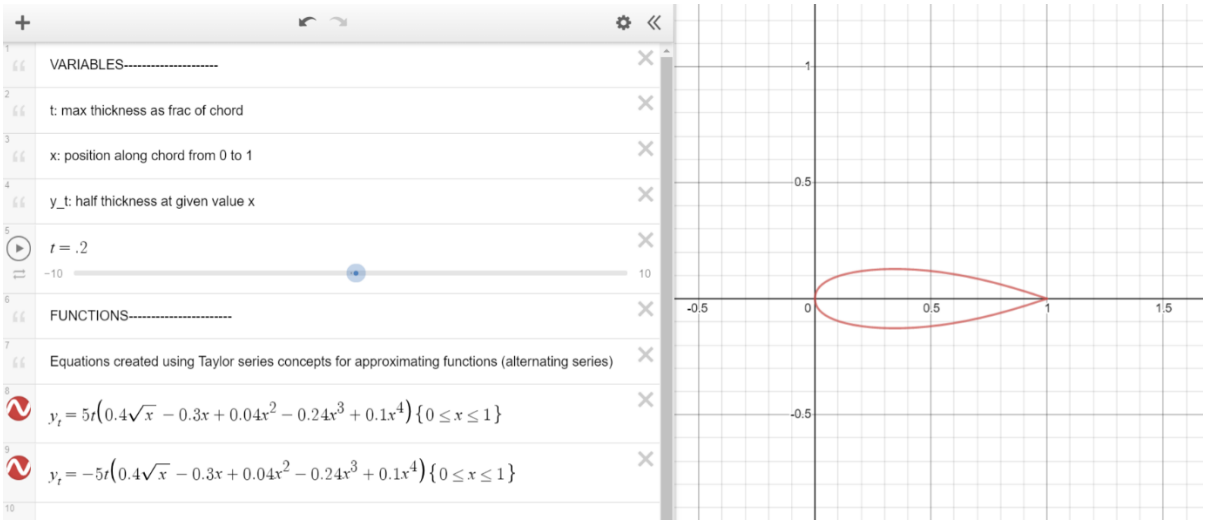
Software Used: SolidWorks 2019

- Learned through participation in the FIRST Robotics Competition
- Installed on personal laptop
- Software commonly used in industry



AIRFLOW SIMULATOR

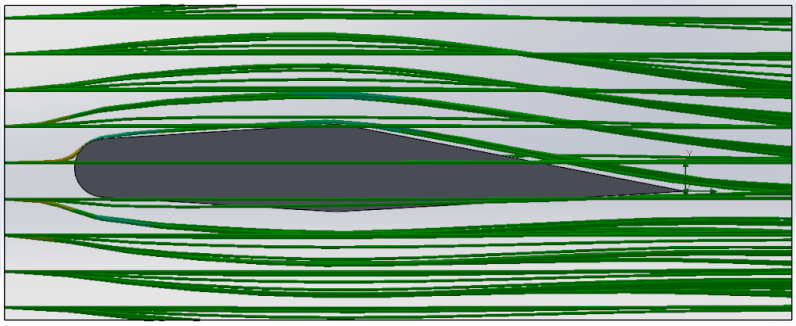
Some designs use symmetrical airfoils, like the one I modeled using parametric equations below:



Desmos link: https://www.desmos.com/calculator/zhmo2m6icy

In the CADD version of my plane, I created a semi-symmetrical airfoil. This shape provides decent stability, while also allowing for flight tricks (i.e. spins, flips).

GREEN LINES
Streams of airflow. Color denotes pressure value on gradient scale.

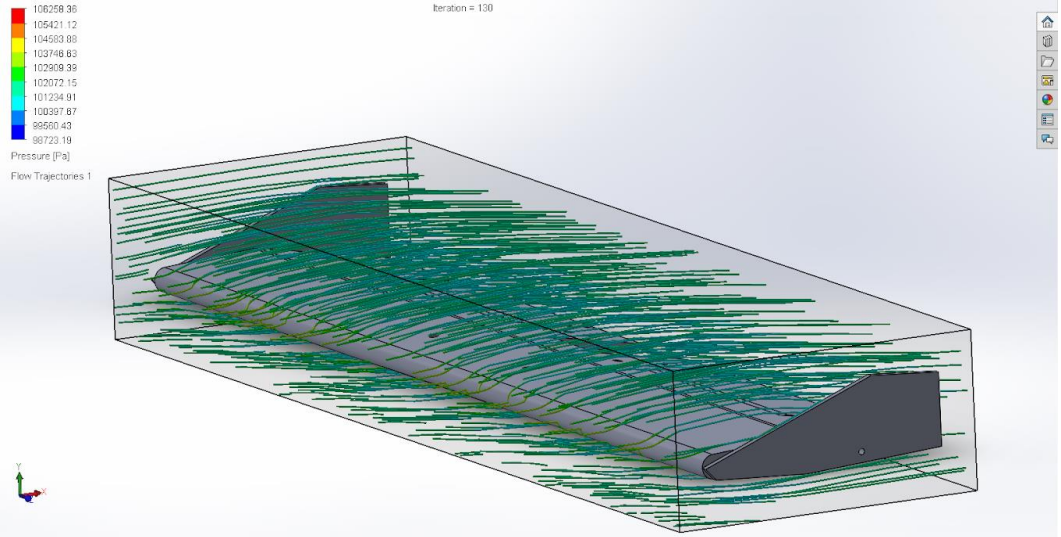


TWISTING
Airflow vortex! Vortices produce drag.

To produce this image, I used Computational Fluid Dynamics (CFD) software, which calculates airflow lines and air pressure values along streamlines.

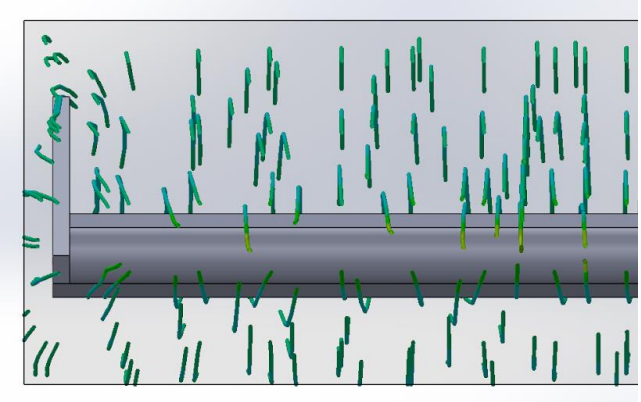
These pressure values hinted at the ability of the airfoil to produce a lift force! For my recreational purposes, the lift generated was sufficient.

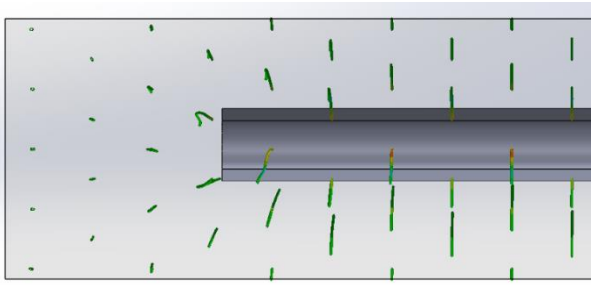
I performed CFD analysis on wing models with and without winglets:

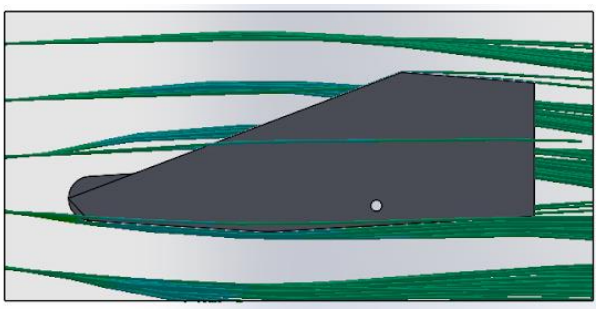


WITH WINGLET

Airflow is guided by the winglet. Parallel flow lines indicate more laminar than turbulent flow.







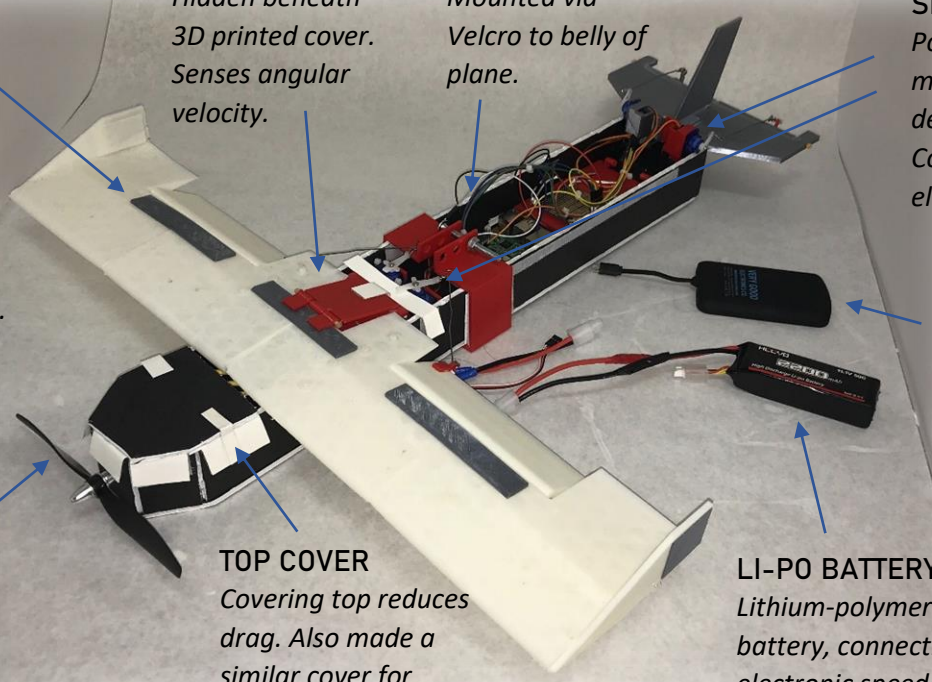
WITHOUT WINGLET

Uncontrolled vortices formed, increasing drag. Also notice a pressure increase for streams across the front face of the wing.

PROTOTYPING

3D PRINTING PROCESS

QIDI TECH
X-PRO 3D PRINTER
A temperature-controlled nozzle extrudes melted plastic (ABS) onto a build plate. The position of the nozzle is determined by reading a 3D modeling file type: "STL".



3D PRINTED BRACKET
Wing printed ¼ at a time, since too large for print bed. Fused ABS plastic with acetone & added brackets for support.

GYROSCOPE
Hidden beneath 3D printed cover. Senses angular velocity.

RASPBERRY PI
Mounted via Velcro to belly of plane.

SERVOS
Positional rotation motors with 180-degree range. Control flaps & elevators.

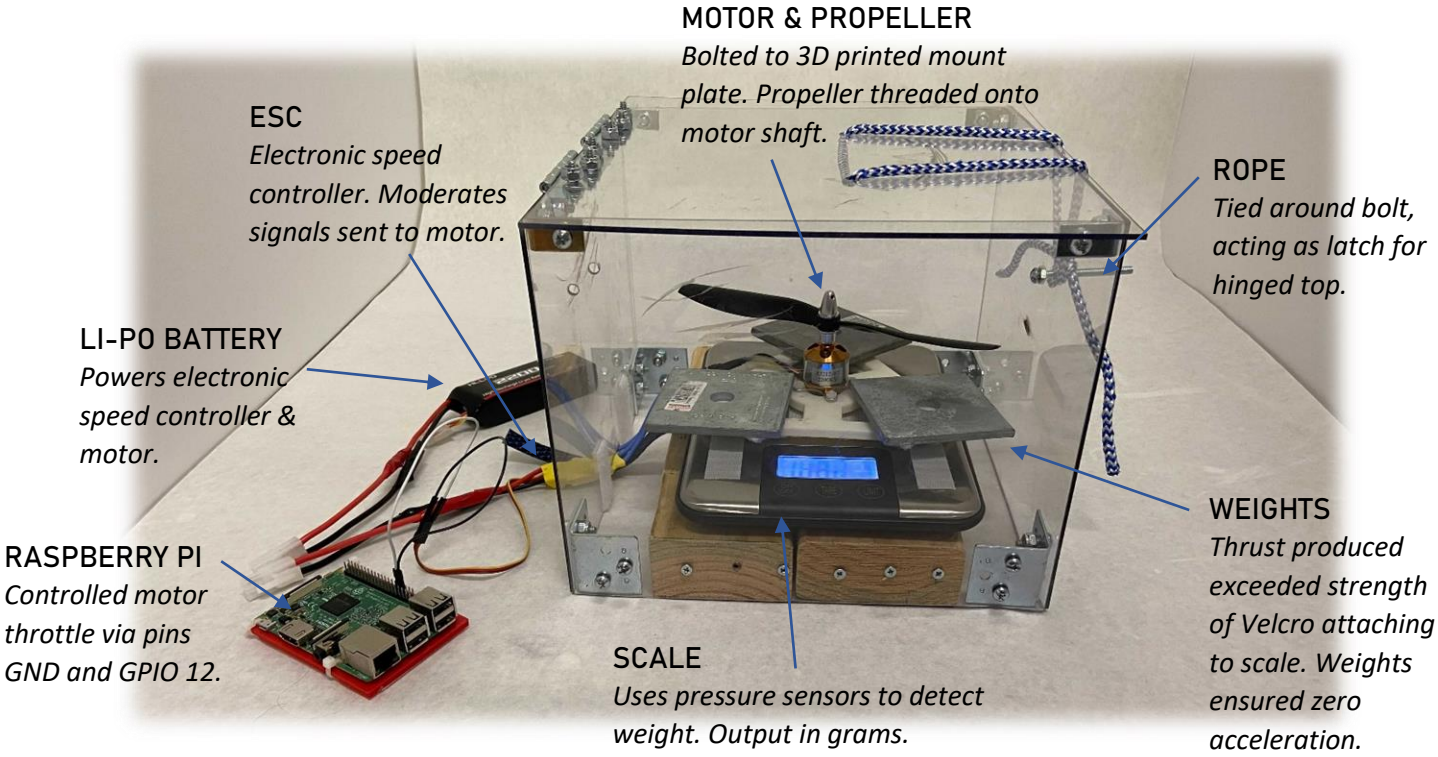
PHONE CHARGER
Mounted inside plane to power Raspberry Pi during flight.

PROPELLER
Generates thrust. Spun by DC motor.

TOP COVER
Covering top reduces drag. Also made a similar cover for back-end (not in picture).

LI-PO BATTERY
Lithium-polymer 11V battery, connecting to electronic speed controller (ESC) for DC motor. Rechargeable.

TESTING MOTOR THRUST

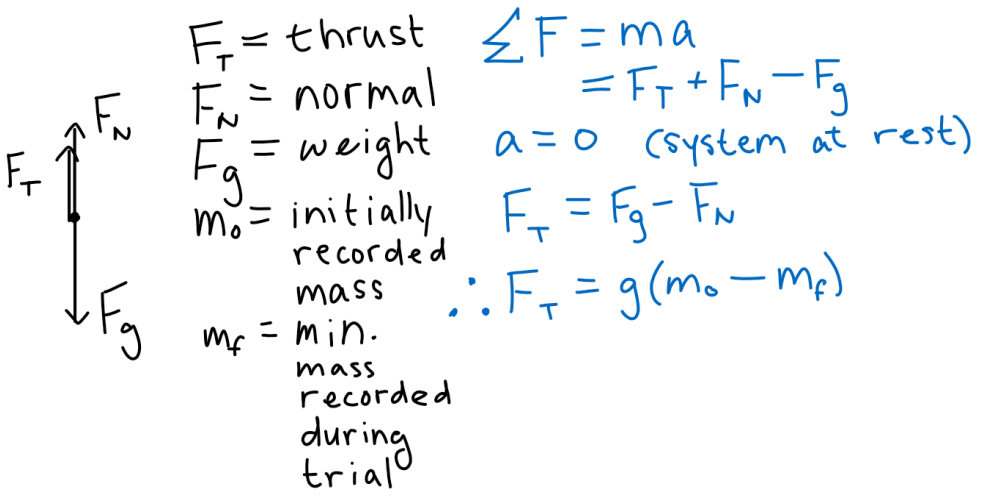


I independently devised a method for recording thrust force values produced by the motor, which would later be important for precise control of the plane's flight path. Around the testing equipment, I crafted a blast-shield made of plexiglass. As it turns out, this was critical to the safety of the experiment.

Before the weights were added, the thrust of the motor exceeded the strength of the Velcro that attached the 3D printed mount plate to the scale, resulting in the following fractured propeller:

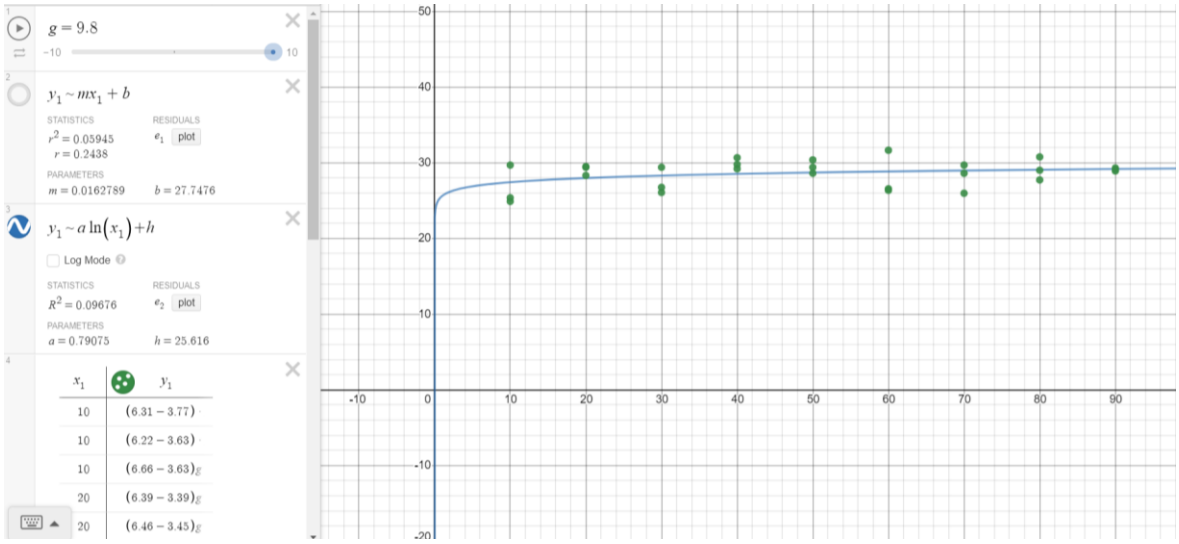
On the next page, I analyze the data I collected from using the testing box setup ("The Blast Box").

To calculate thrust produced by the motor and propeller, I performed:



I filmed the mass values as reported by the scale using my phone camera, selecting the minimum value to record (assumed to indicate maximum thrust).

This graph visualizes the trend for thrust produced (N) per percent throttle (%).



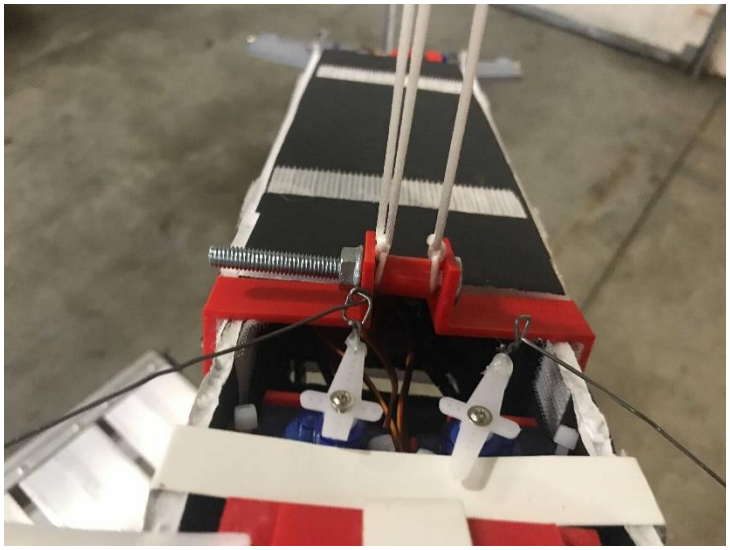
Desmos link: https://www.desmos.com/calculator/ifrugx4x8z

Given that the motor stalls when set at 0% throttle, I used a natural logarithmic regression to estimate the trend of throttle vs. thrust. I was disappointed to discover that the thrust produced had relatively little correlation with the desired throttle value. In effect, I will have to compensate for the poor precision of the motors when I implement a proportional-integral-derivative (PID) controller in creating autonomous code.

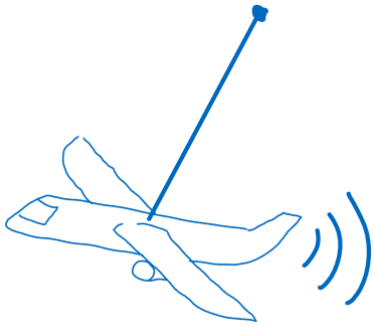
FLIGHT TESTING SETUP

Early in the prototyping phase, I realized that my initial testing would likely result in a catastrophic crash-landing.

During the assembly process, I printed the wing one quarter at a time to stay within the dimensions of the 3D printer's build plate; however, this process consumed excessive time and filament. To avoid repeating this tedious task, I designed a setup that minimized the damage risks of preliminary testing.



TESTING IN GARAGE
Nicknamed the "airplane pendulum" for obvious reasons…

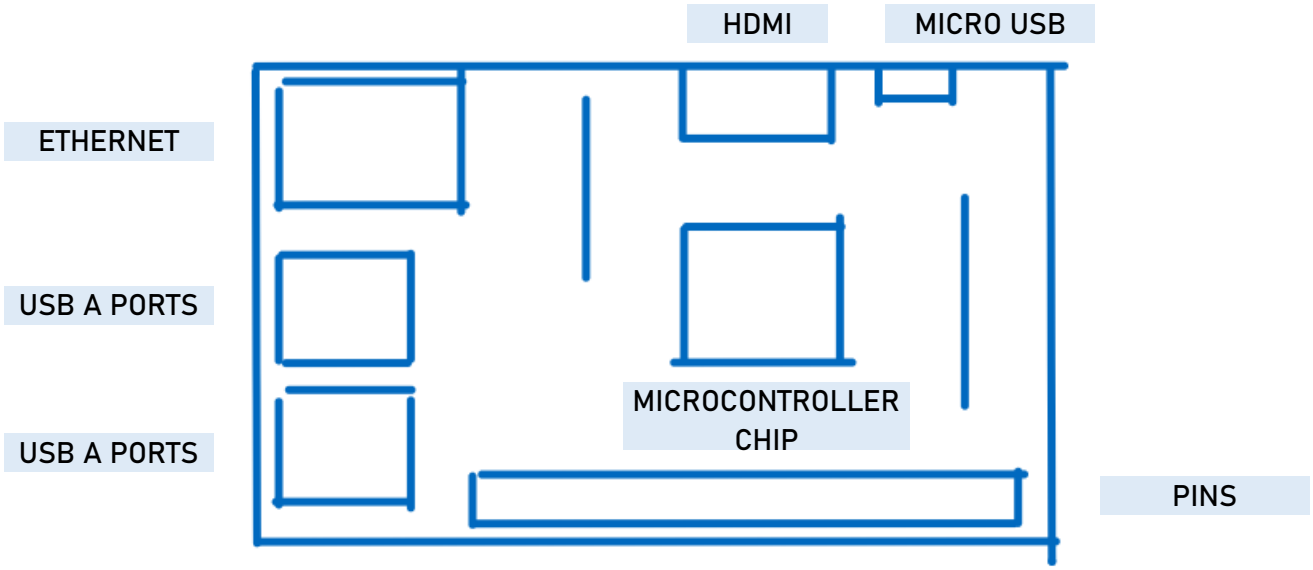


In the end, I decided to use the setup for gyro & accelerometer calibrations, but not for testing lift generation or manipulation of airflow.

advantage	disadvantage
Restricted flight range allows testing indoors.	Merely delays structural damage until more progress has been made.
Prevents structural damage from crashes.	Cockpit propeller causes angular instead of linear displacement. This changes how the wings interact with surrounding air.
Safer & controlled take-off.	Less accurate to flight conditions of professional designs.
Potential for great angular precision when designing procedure to test gyroscope & accelerometer (use natural radius of pendulum).	

CONTROLS

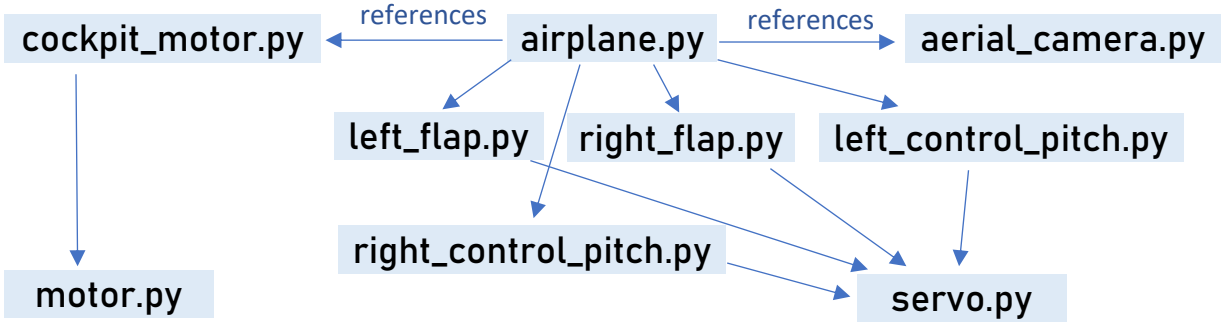
RASPBERRY PI 3B+



The Raspberry Pi is a powerful device that can process and store data. I learned how to establish a completely new operating system (Ubuntu MATE, which uses a Linux environment) and program the device using the Python 3 language.

OBJECT-ORIENTED PYTHON

Object-oriented coding takes advantage of the modular structure of the Python 3 language by referencing methods within classes – even across separate files.

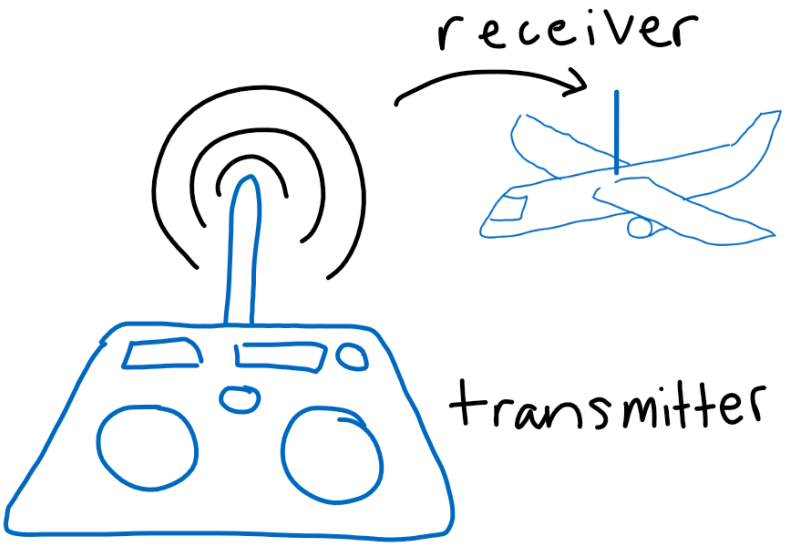


Although it adds an initial level of complexity, this style is common in industry due to its efficient debugging capabilities.

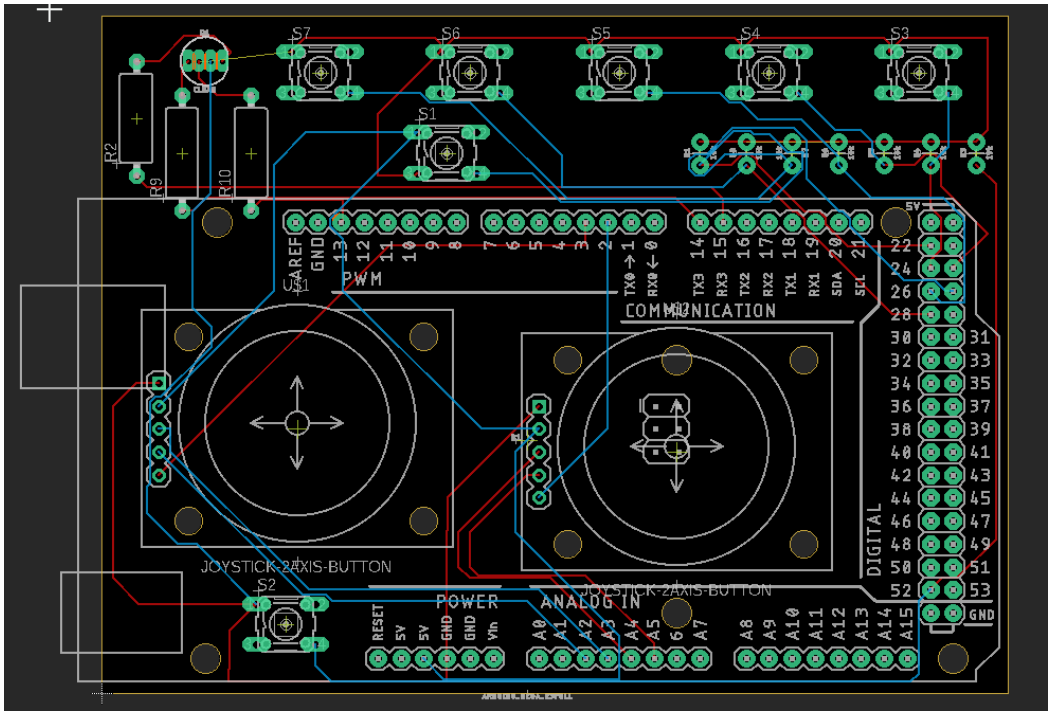
CLICK HERE TO VIEW MY CODE ON GITHUB

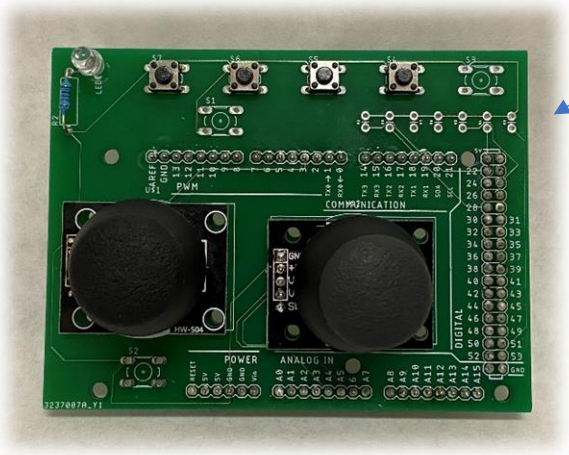
CUSTOM PRINTED CIRCUIT BOARD

As a steppingstone towards creating an autonomously controlled airplane, I decided to first establish remote control.



I designed a custom printed circuit board (PCB) using EagleCAD by Autodesk Inc. that plugs directly into an Arduino MEGA 2560 microcontroller. This board transmitted radio signals to control the receiving end on the airborne plane.

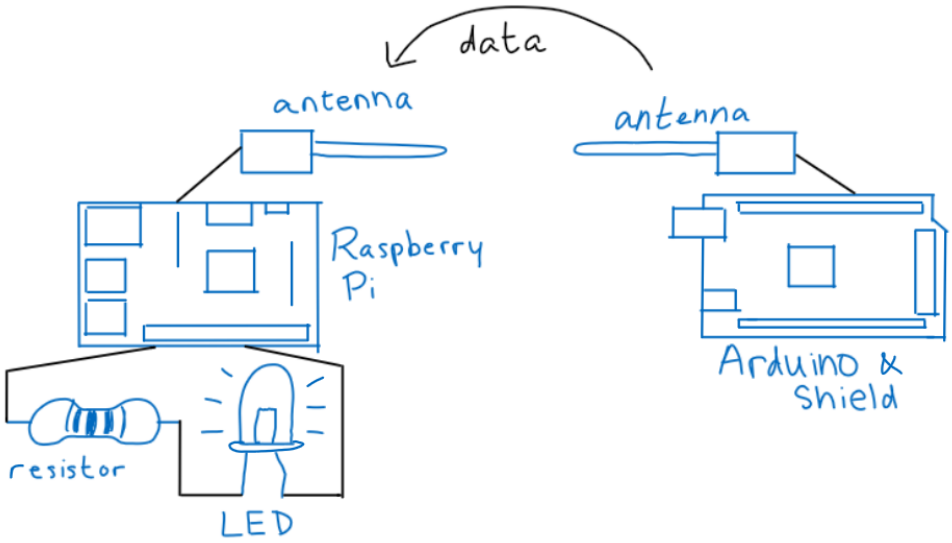




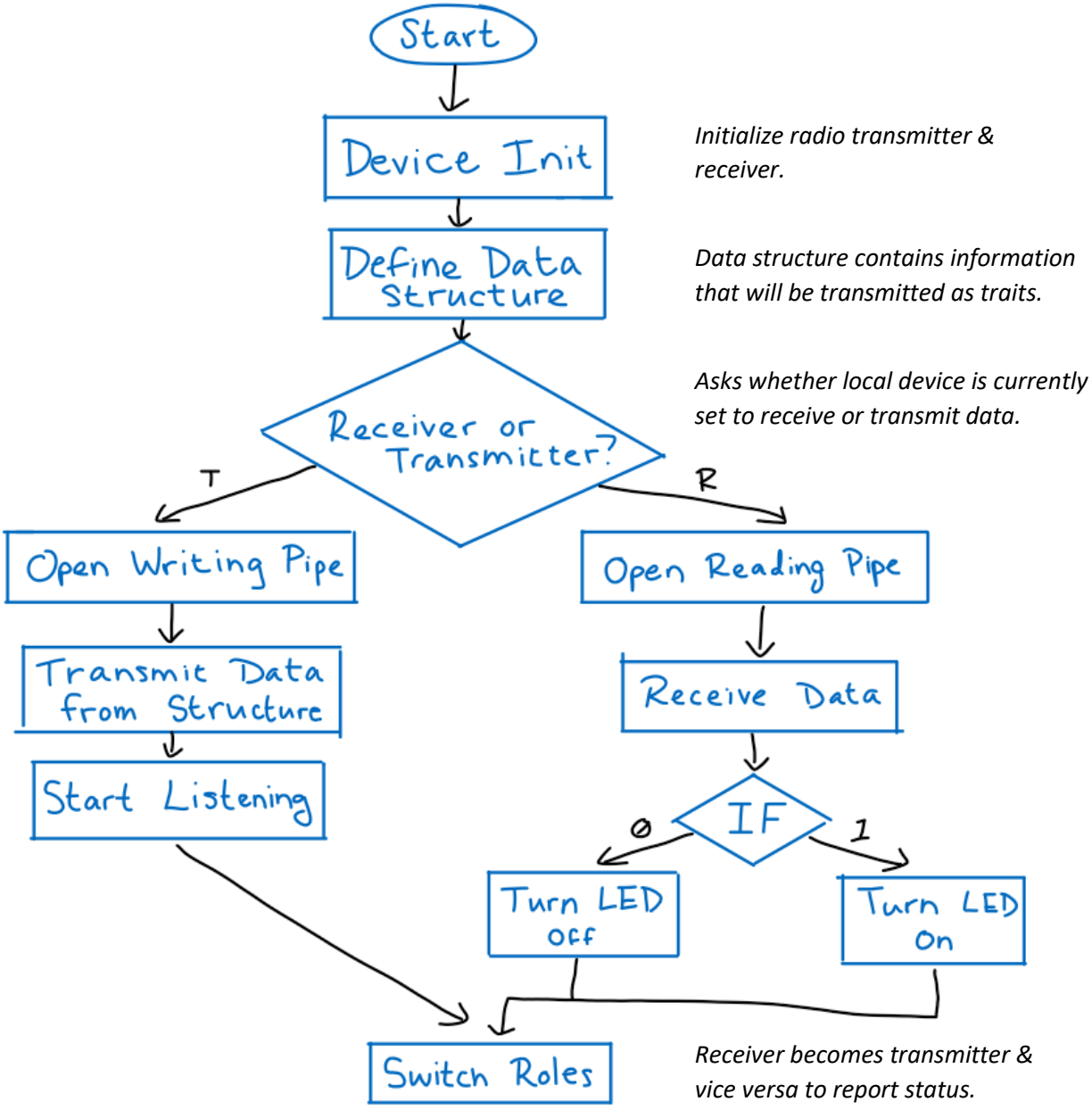
Not all components have been soldered onto the PCB because some are unnecessary at this testing stage.

To the left is the PCB I designed. On the right is the Arduino microcontroller. Since the PCB plugs directly into the Arduino pins, it is called an Arduino shield. Pictured below is the antenna.

After soldering necessary components onto the PCB, I started writing code to test the radio transmission capabilities of the device. While I am still refining the code, I plan to initially establish remote control by transmitting a Boolean value from the Arduino to the Raspberry Pi to ultimately toggle an LED on and off.



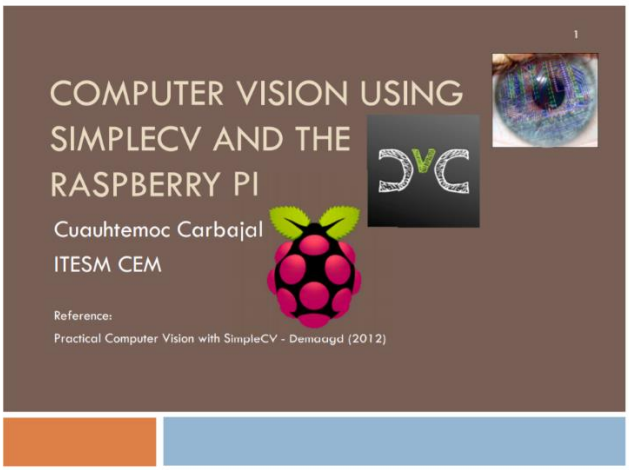
Once I am successful in this testing stage, I will advance to more complicated data structures, such as the analog values produced by joysticks.



The code diagram above is merely an abstraction of the code I developed based on an existing C/C++ radio control library (RF24). [Click here to view my radio control repository on GitHub.]

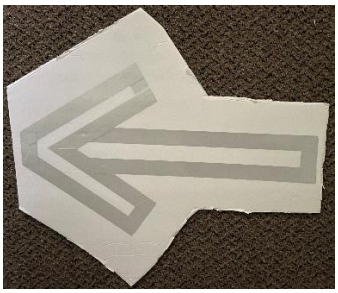
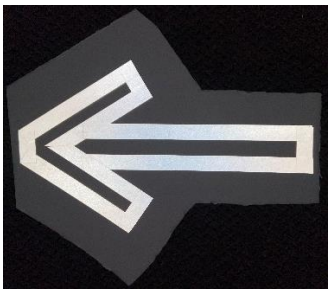
VISION CODE

While waiting for the circuit board and other parts to be shipped, I began looking ahead to map out my plan for computer vision, image processing, and path following. Currently, I plan to base my code on the Simple CV library (written in Python), starting with the reference guide below (image is hyperlinked):



Plan for implementation:

- Create autonomous routines in response to pressing certain buttons on remote controller*
- Create autonomous routines in response to detection of specific shapes
    - Using depth information extraction, avoid objects that obstruct flight path (e.g. trees)
- Integrate PID velocity control and gyroscope/accelerometer sensor values into a path following routine
    - User can walk below, guiding plane with vision tape arrow

*This does not involve vision or image-processing but is included in this list as a stepping-stone to the following bullet point.

CONTROL THEORY

I started developing this model airplane in March 2020. Starting June 2020, I became a Humanoid Robotic Legs Intern at the Portland State University Agile & Adaptive Robotics Lab.

Through this internship, I was exposed to the world of control theory, which I plan to apply to my model airplane project via PID implementation.

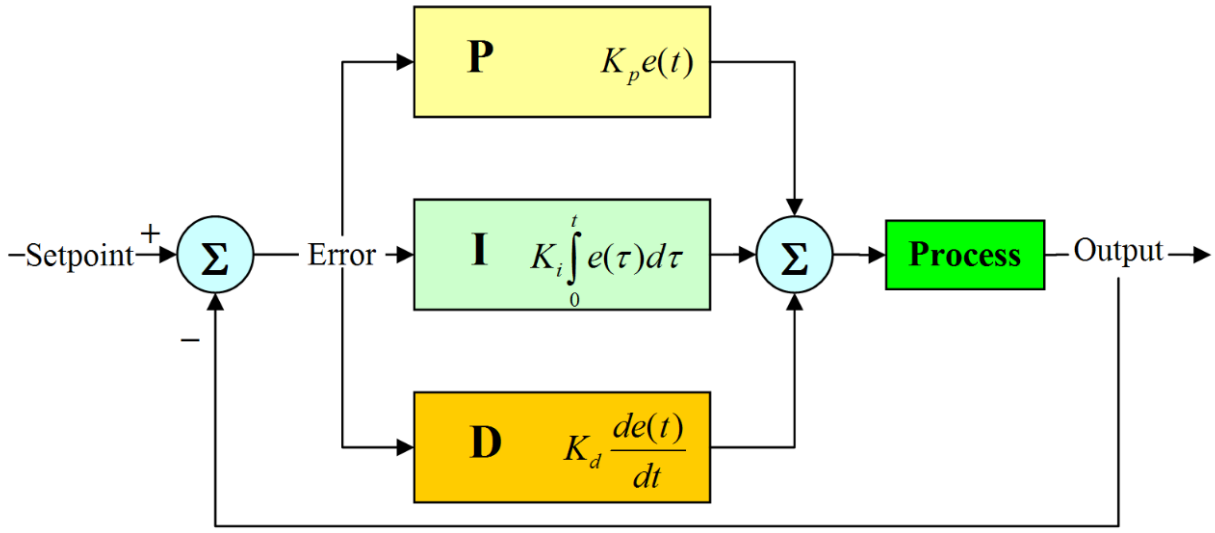


Image Citation: The Society of Aerial Cinematography. (2015, April 16). [PID loop visual representation.] Retrieved October 05, 2020, from https://thesoac.com/the-abcs-of-pids/

While I have worked with PID concepts in the past, I only ever dealt with simplified, abstract versions of the controller. In this airplane project, I plan to build PID loops from scratch to use in autonomous routines.

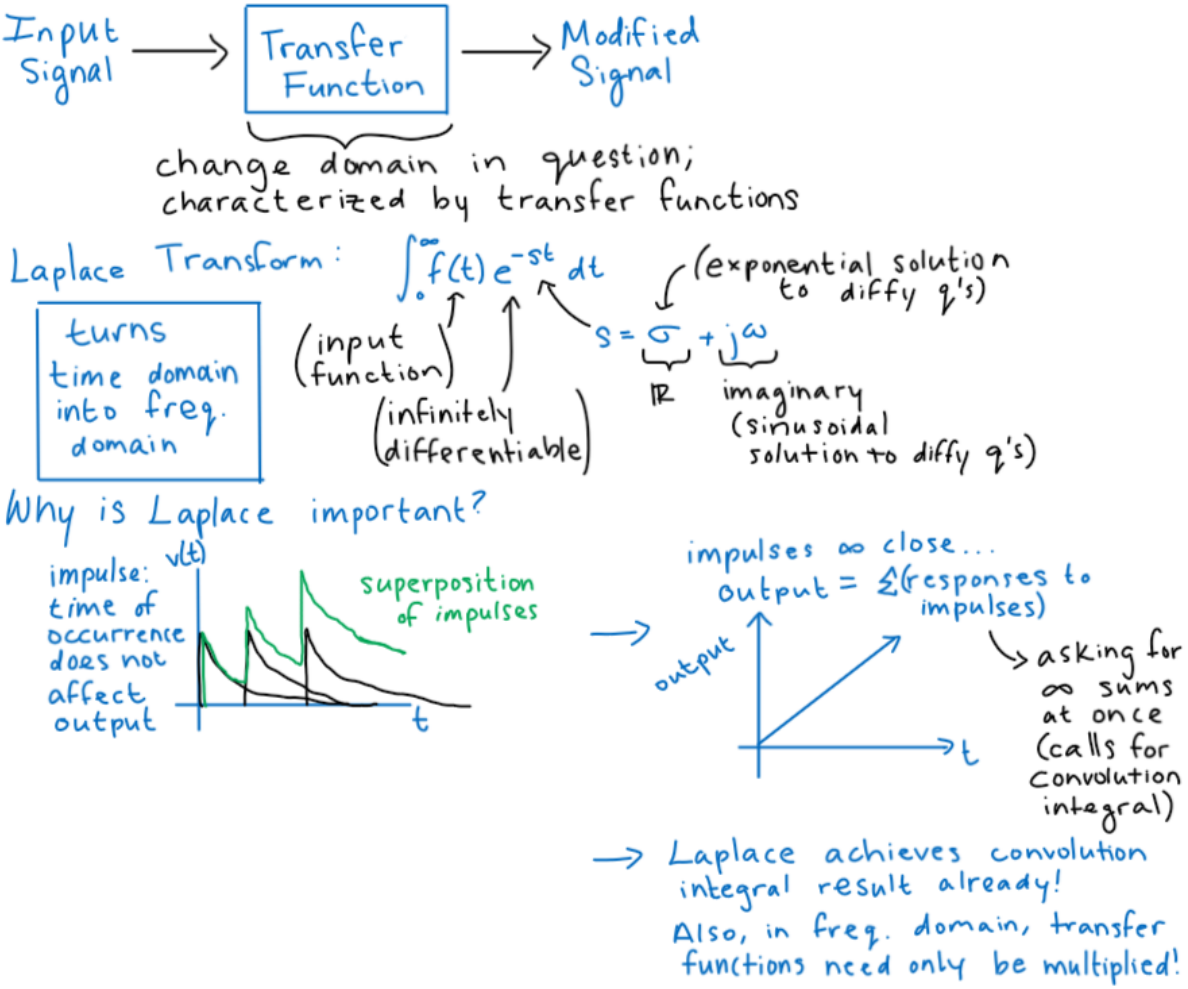
To best understand PID loops, I learned about the Laplace Transform, a foundational concept of control theory. The figure below compares the PID function in the time (top) vs. Laplace (bottom) domains.

$$u(t) = K_p e(t) + K_i \int_0^t e(t')\,dt' + K_d \cdot \frac{de(t)}{dt}$$

$$L(s) = K_p + \frac{1}{s} K_i + K_d s,$$

Main Source: https://medium.com/@dingyan7361/how-to-solve-a-simple-control-system-problem-with-laplace-transform-3d94ffa00009

Here are some notes I took about transfer functions, Linear and Time Invariant systems, and the Laplace Transform:



Main Source: https://www.youtube.com/channel/UCq0imsn84ShAe9PBOFnoIrg

FINAL PRODUCT

REFLECTION & CONCLUSION

Building a model airplane confirmed my interest in aerospace engineering and design. I dove deeper into the project than I ever expected. I found that I was not only modeling a literal airplane, but also the process that practiced engineers use when creating a safe and effective design.

In fact, I reached out to 27 different companies to learn about their engineering processes. If interested, click this link to read a sample response that I received.

I sincerely wish I could provide a video of the airplane in-flight, however, its current stage makes that possibility unrealistic. Not to mention, my goal throughout this process has been to let no question go unasked; naturally, this approach lends itself to fascinating but time-consuming investigations.

Of course, this notion is precisely why open-ended projects are such fun!

In fact, this project inspired an independent study in my senior year: Computational Physics Modeling Using Linear Algebra. Concurrently, I am learning linear algebra at Portland State University and self-studying AP Physics C: Mechanics (calculus-based) in order to combine both into a computational project modeling the flight path of the model airplane.

UNEXPECTED OBSTACLES

My model airplane project was constrained by the COVID-19 quarantine. Since I did not fee safe visiting hobby stores for supplies, I manually crafted the parts I could and ordered the rest online. My progress was consequently paused at various times due to shipping delays.

Furthermore, Portland received unprecedented low air quality due to wildfires towards the end of the summer. Thus, I was unable to test my airplane for weeks on end.

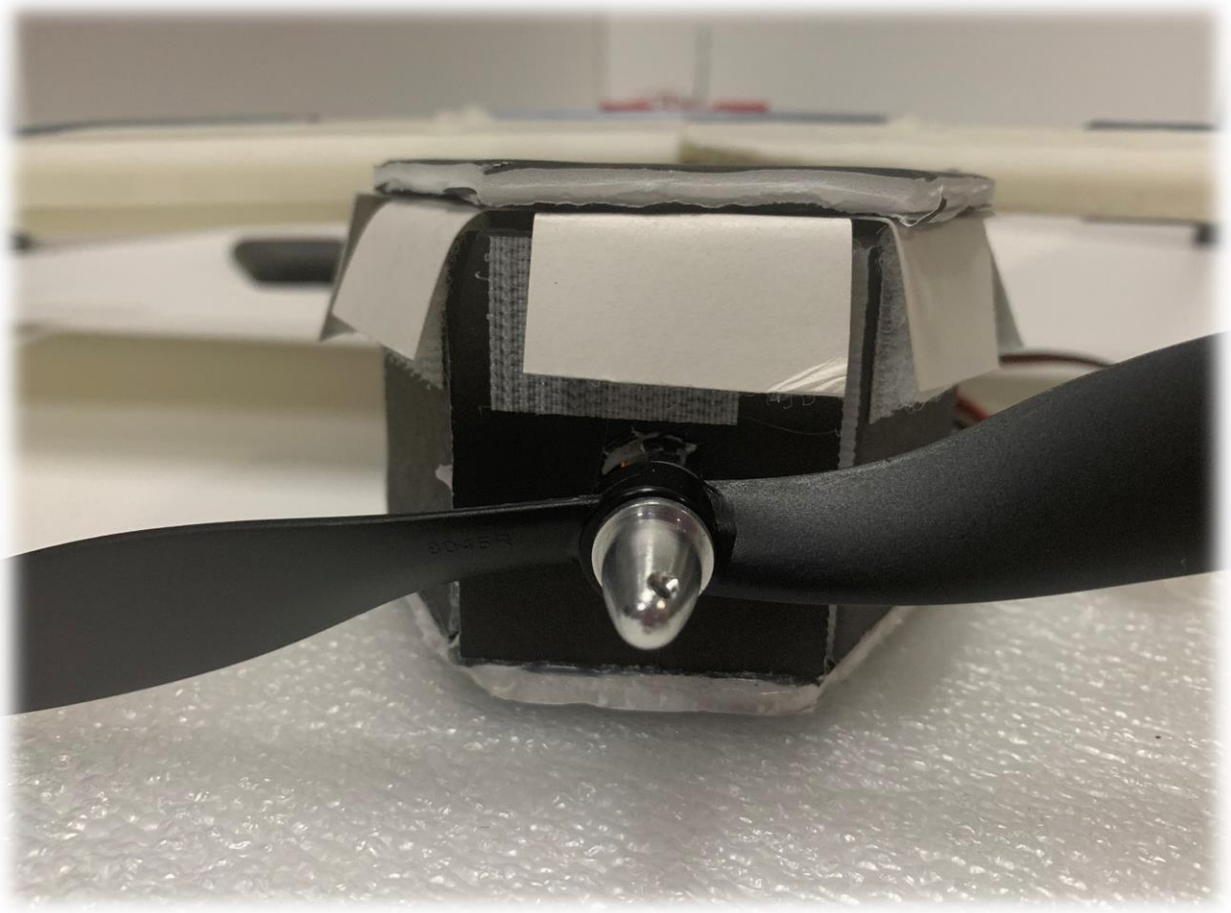
IMPROVEMENTS FOR THE FUTURE

My airplane project does not stop here!

Engineering is an inherently iterative process, so even if I had already completed a fully autonomous model, I would not be finished.

I plan to:

- Improve airplane control using gyroscope & accelerometer sensors
- Design a reliable launch & landing system
    - Deployable for minimal drag produced
    - Withstand significant force & prevent flipped landings
- Iterate & improve code for remote radio-signal control
    - Including language translations from C/C++ to Python
- Develop code for computer vision
    - Using image processing to respond to visual commands



FUN FACT

For my senior photos, I posed with my airplane! In this picture, I hoped to capture my personal values at the time the photo was taken, especially emphasizing my love for the creative engineering process.