



ANDREW ZHU

Design Portfolio

2020-2022

TABLE OF CONTENTS

<u>PROJECT</u>	<u>START DATE</u>	<u>PAGE</u>
Multi-Stage Rocket	Fall 2021	2
Structural Roll Cage	Spring 2022	
Smart Trashcan	Summer 2021	
Rider's Guard	Fall 2021	
Grabber Cane	Summer 2021	
3D S.T.A.R. Logo	Spring 2022	

INTRODUCTION

Howdy! My name is Andrew, and I am a current junior studying in Mechanical Engineering at the University of California, Berkeley. You are currently viewing my engineering portfolio. Here, I have tried to not only detail my technical knowledge and understanding, but also my engineering mindset and thought process.

I have been fortunate to have been able to work on many projects both inside and outside the classroom, and I wish to share them with you. It is with my hope that they will help in the assessment of how I may contribute to your company!

Thank you for your time!

ABOUT ME

I have always been the curious type and have always been the kinetic learning type. My understanding of things comes visually and physically so I often like to draw things out and construct tangible models. Naturally, projects have always been the best learning medium for me to absorb, apply, and express what I have learned. I work the best when there is a clear goal and feasible timeline, and I enjoy working on teams that share a similar work ethic but offer different perspectives.

At my essence, I am a hard worker and a critical thinker as well as a team player.

MULTI-STAGE ROCKET

Overview:

One of the projects supported by S.T.A.R. was creating a multistage rocket that would be able to separate mid-air. In almost all cases, multistage rockets are exponentially more complex to design and manufacture than a single stage rocket.

For starters, more than one rocket motor must be taken into consideration. While they are inherently what make a rocket go up, they are considerably dense, meaning that they contribute significantly to weight. The electronics also becomes significantly more complicated. Multiple altimeters are needed to stage the rocket, ignite the second stage motor, and successfully deploy the parachute recovery system.

So, why did we still decide to pursue it? From a technical perspective, the ability to jettison dead weight makes the rocket cheaper to launch and reach new heights, literally. But deep down near the launch pad, we wanted to do something that no other rocketry team here at Berkeley had ever attempted. Here are my contributions to the success of Stage Separation v1:

Changes in Avionics Bay Design:

An avionics bay is designed to hold the electronics of the rocket and thus is crucial for the successful recording of flight data and recovery deployment. To that end they should be designed to be easily retrievable and structurally sound. Previously designs have been made by past members but after discussions with the project manager and recovery lead, we decided they can still be optimized for space even with an extra and larger battery.

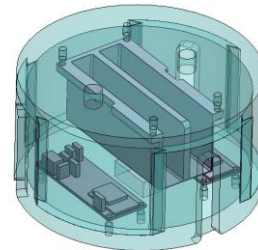
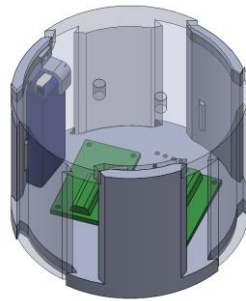


Figure 1: Previous design vs. current design. The new design, 3d printed with carbon fiber infused Onyx filament, can hold two Eggtimer Quantum altimeters and their respective, larger LiPo batteries. It also has cutout slots to account for airframe screws.

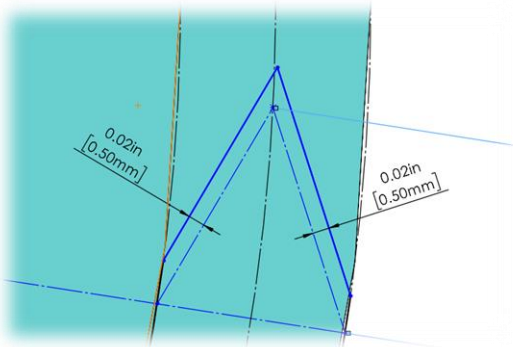


Figure 2: Sketch showcasing tolerance on the top shell.

In addition to keeping a low profile, the Avionics bay also was designed to be streamlined and durable. I introduced the idea of using heat-set inserts since the previous design used thin diameter hex socket screws. Thin screws in PLA material makes it very prone to over tightening and thus stripped screw holes. Pre-tapped inserts and a soldering iron were used to place them in, allowing all components to be securely attached. This is crucial when vibration during flight is prominent.

This avionics bay is considered axial, which means it is taken out of the rocket axially and opens up axially. It is comprised of two shells, a top and bottom. In an effort to reduce post-processing time, I was able to tolerance a sliding fit which accounted for the Markforged X7 industrial printer's ± 150 micrometers accuracy and 0.4mm nozzle diameter by offsetting the female interface by 0.5 mm.

In Figure 2, the blue dashed lines projected onto the top shell outline the corresponding male interface from the bottom shell. They are used as reference for determining where the actual cut is for the top shell. Thus there is a gap of 0.5 between the interfaces.

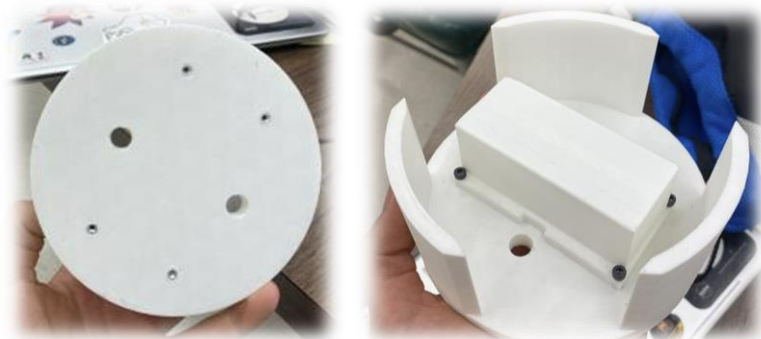


Figure 3: 4 aluminum heat set inserts fused in the shell. Designed for M3 screws.

Changes to the Stage Separation Mechanism:

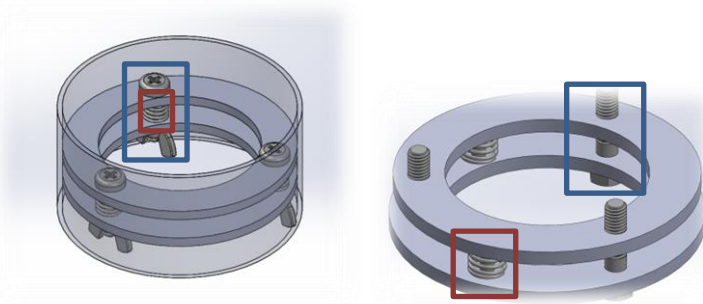


Figure 4: The left has nylon pyrobolts (highlighted in blue box) and springs (highlighted in red box) assembled in concert while the right has pyrobolts and springs assembled separately. This helps reduce the chance of destroying the springs which will be epoxied on one end to the steel rings. The steel rings are attached to the airframe tube using epoxy filleted on both sides.

This mechanism is at the heart of stage separation. The idea behind this is when the rocket is in its coasting phase, a sufficient amount of force from our custom made pyro bolts (nylon bolts drilled and filled with 2 grams of black powder and an e-match and finally capped with epoxy) and compressed springs can adequately break the shear pins holding the two stages together.

Both images on the right showcase this idea. However, the one on the left, suggested by me, is designed for repeatability since the springs are isolated from the pyrobolts. Thus, we can reduce the cost of materials and ensure that the spring properties remain unchanged.

U-bolt selection and FEA Validation:

Recovery happens smoothly if the parachutes deploy correctly. If not, the rocket will come back down like a lawn dart at very fast speeds. This is not only extremely dangerous but also detrimental to the rocket and the onboard electronics. There are many different failure modes to consider during the recovery phase, such as the ejection charges not igniting, the lack of pressure build up, the shock cord breaking, and the parachute tangling. I was tasked with selecting an adequate bulkhead that would not yield when shock forces are exhibited on it from the parachute deploying.

Our stage separation rocket has three parachutes: an upper stage drogue and main parachute and a lower stage main parachute. I selected the upper stage main parachute to test on as the worst case scenario since it is the only one that deploys when the rocket is moving (the other two parachutes deploy at apogee where the velocity is zero so no shock forces occur due to inertia).

I used SolidWorks Static Simulation to test the maximum amount of force calculated from [OpenRocket](#) exhibited on the U-bolt as shown in Figure 9. A drag force model provided by [NASA](#) was used to corroborate this. Using the drag coefficient of a parachute 2.2, the average air density $1.229 \frac{kg}{m^3}$, the velocity before deployment $17 \frac{m}{s}$, and area $0.76^2 \pi$, the shock force was 708.95N.

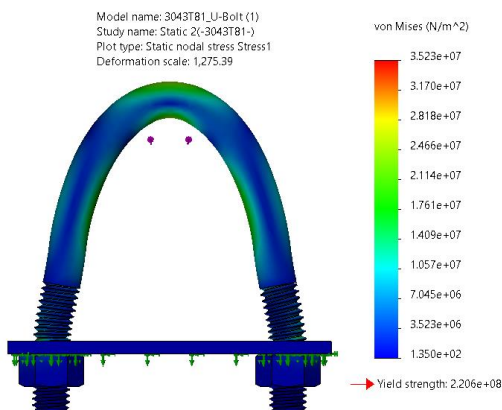


Figure 6: Stress showcasing highest stress at $3.523E7$ Pa while the yield strength of this material is $2.206E8$ Pa.

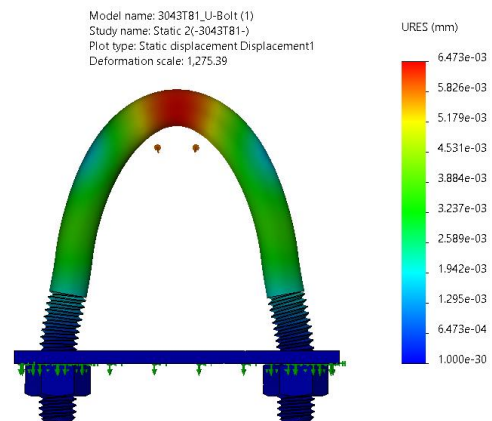


Figure 5: Displacement results with a 1275.39 (exaggerated) deformation scale. Max deformation of $6.473E-3$ mm at the top.

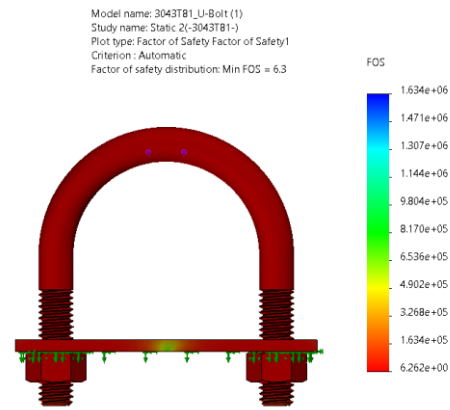


Figure 7: Factor of Safety of 6.26 calculated based on highest stress and yield strength. The U-bolt is clearly strong enough.

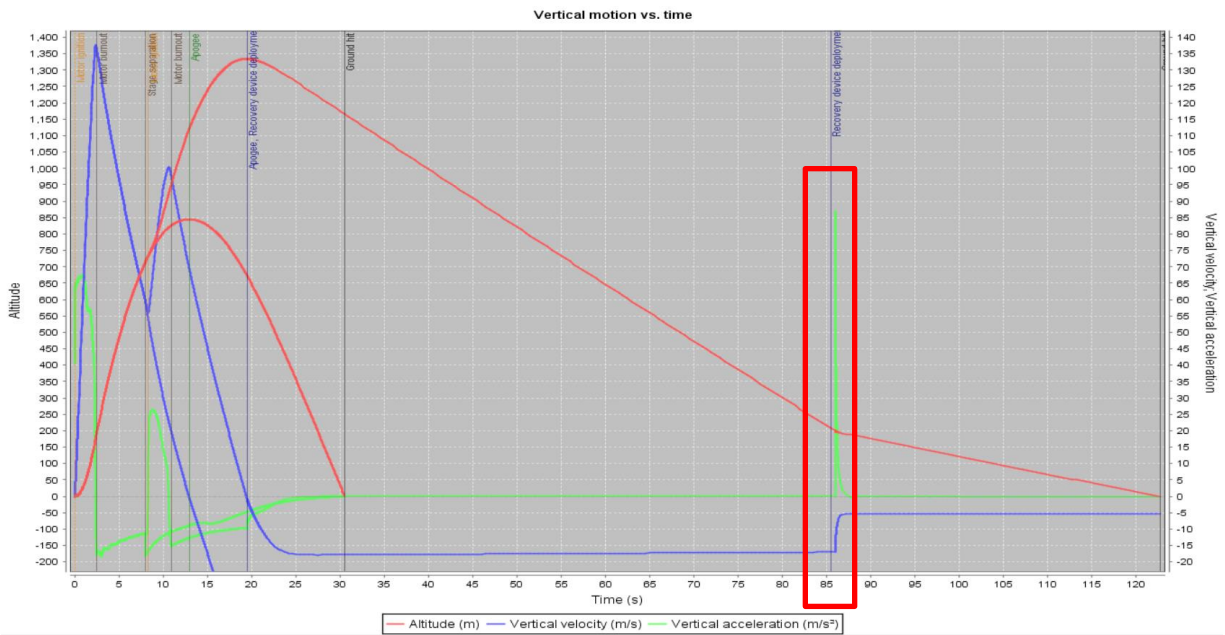


Figure 8: This is OpenRocket flight data recording the displacement, velocity and acceleration in the vertical direction of both stages. The red box highlights the shock force exhibited on the top stage at 250 m at a terminal velocity (from the drogue) of around -17 m/s.

Rocket Overview:

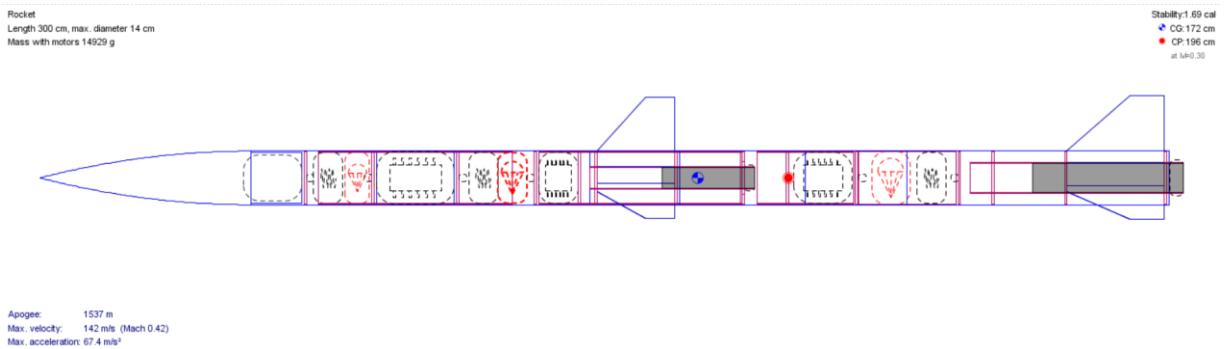


Figure 9: This 2D diagram from OpenRocket highlights the stage separation rocket's shape and internals such as the parachutes, avionics bays, rocket motors, and ballasts. It also includes center of gravity and center of pressure.

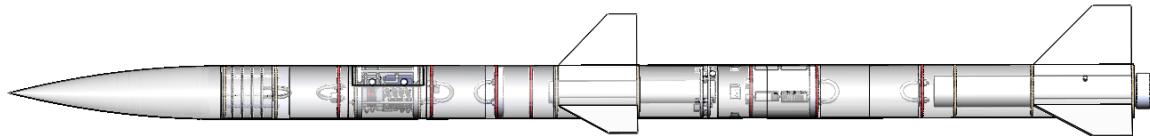


Figure 10: This is the 3D model representation from SolidWorks and is an exact 1:1 scale to OpenRocket. It is worth noting that all designs were first completed and tested in OpenRocket before anything was modeled in SolidWorks.



Figure 11: The stage separation rocket in its final form at [Friends of Amateur Rocketry \(FAR\)](#) in the Mojave Desert.

STRUCTURAL ROLL CAGE

Overview:

One of the most important things to consider when designing a car is safety. To that end, everything on CalSol's newest car, Excalibur, which will be competing in the [American Solar Challenge](#), must be thoroughly considered and tested. As the main member on the roll cage project, I have made many design iterations. To make it structurally sound, the roll cage requires research into many areas to have a cohesive understanding.

From the design perspective, I had four main things to consider. For starters, the roll cage must fit under the already manufactured canopy, which is the aerodynamic shell that fits over the driver region. Secondly, it must satisfy design specifications requesting 50 mm of clearance between the driver's helmet and roll cage. Thirdly, it must survive the specified impact cases given by the race officials. Lastly, the overall design should be as lightweight as possible.

From a manufacturing perspective, I had four main processes to consider: how to tube bend thin walled tubing without buckling occurring, how to miter and notch the tube ends for perfect assembly fitment, how to properly weld and select filler to preserve material properties, and how to post welding heat treat to relieve heat affected zones.

As of now, the roll cage design phase has been finished. Here I plan to showcase the design process for the final iteration:

Material Selection:

One of the first things to discuss is what material to build the roll cage project out of. Two materials that were suggested by former members have been 6061-T6 aluminum and 4130 steel since they have been used in past car iterations and are very popular choices in the industry.

Grade 6061-T6 aluminum alloy

Made with magnesium and silicon, it is useful in almost any application. Our most well designed and most raced car, Zephyr, has a chassis frame and integrated roll cage made from this material. It has stood the test of time for more than seven years and races now.

It has the benefits of:

- relatively great UTS of 310 MPa and YTS of 276 MPa
- great heat treatability
- relatively easy machining
- easy weldability
- supporting anodizing for protective surface

Type 4130 alloy steel

Made with molybdenum and chromium, it is heavily used in high stress applications and is found in the motorsport industry.

While it hasn't been as prolifically used as 6061-T6 in our past cars and projects, it has the promise of being a great material for reducing weight while maintain a very high strength.

It has the benefits of:

- great UTS of 670 MPa and YTS of 435MPa
- good workability when annealed
- heat treatability
- great weldability

While on paper both choices are excellent, many other factors influence the material selection besides just the properties themselves. For example, what kind of cross sectional tubing will be used? What will the wall thickness be? What kind of basic bend geometries will we be seeking to conform to the space between the driver's helmet and the car's canopy? A lot of these questions can ultimately be answered with CAD and simulation iterations using the FOS results as feedback. Based on the results, I ended up choosing 4130 steel since it could maintain the minimum FOS of 1.1 for all test cases.

Design Choice:

The final design is quite different from the typical roll cage seen on motorsport cars. It doesn't follow the contour of a car's monocoque but instead the driver's occupant cell. In addition to being structurally sound, the roll cage is designed with manufacturability in mind.

Overall Geometry

The two hoops are angled inwards by a couple degrees to help triangulate the geometry. In other words, if a load is applied in the backwards on the top of the front hoop, its trapezoidal geometry (from a side view) is more structurally rigid than a rectangular geometry. Similar to triangles, trapezoids have corner angles that cannot deform without an accompanying deformation in the edges first. Choosing a rigid geometry like the trapezoid helps take the stress out of the weld connection points.

The two hoops also each exhibit two 5 inch bends with a straight section in between. This is for manufacturing purposes since our immediate manufacturing source use a rotary draw bender with a 5 inch radius die which requires a straight bar clearance section before a second bend can be made. Additionally, it makes coping the top straight beam extremely straightforward.

The sidebars are quite unique since they use bent tubing. There were two main reasons for this. The main reason was to provide the required space to satisfy the 50mm clearance all around the driver's helmet while also making manufacturing relatively simple with a roll bender. The other reason is in the case of plastic deformation, the bars will bend safely outwards and away from the occupant cell.

Cross Section Shape and Size

Taking inspiration from regular roll cage designs, this roll cage uses a hollow circular cross section for the beams.

The circular cross section, unlike a rectangular or wide flange cross section, exhibits strong symmetry and thus an isotropic behavior in its radial directions. In the unpredictable nature of a crash, loads will be applied in an unpredictable manner as well so symmetry is ideal.

The reason for choosing a hollow circular cross section as opposed to a solid rod was because of the better strength to weight ratio. From a cross sectional point of view, a beam is stiff when material is far away from the neutral axis where exhibits no compression or tension. Thus, hollow is more material efficient than solid cross sections. This is also one reason why a 1.5-inch outer diameter was ultimately chosen.

The wall thickness of the tubing is thin walled based on the rule of thumb $\frac{D}{t} > 20$ where D represents the diameter of the tube and t is the wall thickness. The reason for this is to reduce weight but not to a point where local buckling becomes an unsolvable issue during tube bending. Thus 0.049-in and 0.035-in where the thicknesses that were settled on.



Figure 12: Isometric view of the finalized roll cage design with manufacturing feasibility. Made of 1.5-in OD 0.049-in and 0.035-in wall thickness 4130 steel tubing. This design has a weight of 10.68 pounds which is reduction of 13.3% from the previous design

FEA Simulation Cases:

Setup

Extensive load cases were considered in SolidWorks Static Simulation. Per the request of the regulations, the meshing of the roll cage cannot be done with surface or beam meshing since they oversimplify the analyses. Thus, solid body meshing was used, decomposing the roll cage into finite tetrahedrons. The mesh quality was improved by removing sharp corners in places where stress concentrations would occur.

The regulations also state that a circular loading patch of no more than 150mm may be designated on a test location, which are chosen as the top of each hoop. The reason for this rule was to limit the amount of contact area the forces could spread over, effectively lowering the pressure.

There were five different test cases to consider for each hoop. Each test load differed in either angle of attack or magnitude. Magnitudes are specified as a multiple of the total weight of the car since a static simulation is being used.

Results

The results were very interesting. All case results had comfortable FOS (calculated with material YTS) results above 1.1 and very small max deformation values below 25mm as required by the regulations.

In almost every case test between the front and rear hoops, the rear hoops obtain higher FOS values. Some speculations as to why this is could be because of the multiple load paths that can be taken from the rear hoop to the front hoop.

It is also worth noting that the maximum stresses always occur in locations where the tube meets. This helps confirm that these locations are places of higher concentration and should be taken in careful consideration during the manufacturing phase. In other words, proper coping of the tubes and welding procedures are a necessity in order to match the welds simulated. We will be testing different weld samples based on welding procedures for weld strength using an Instron machine in the future

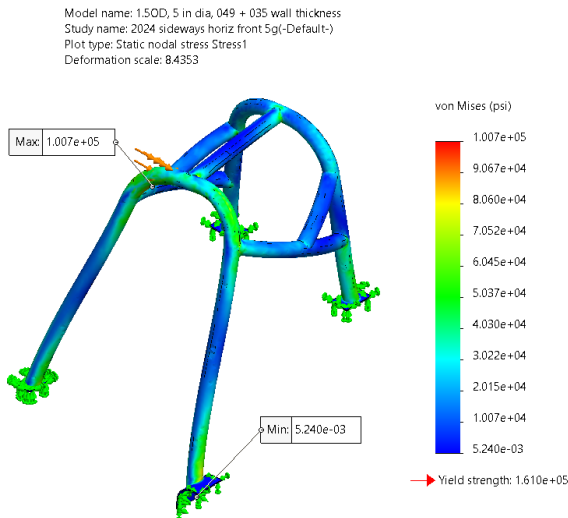


Figure 13: 5G Sideways load on front hoop with 1.6 FOS

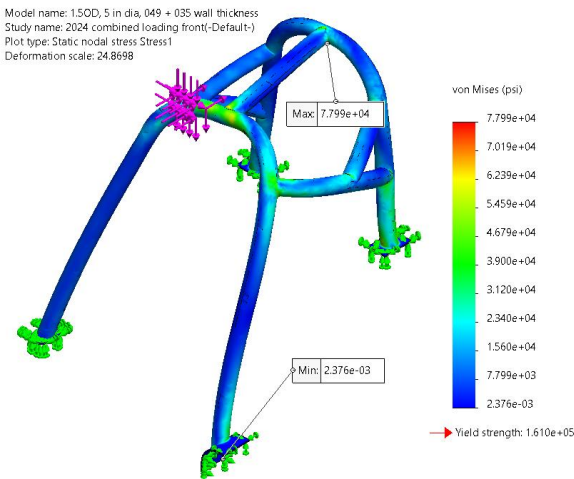


Figure 15: Combined (5G backward, 1.5 sideways, 5G down) loading on front hoop with 2.1 FOS.

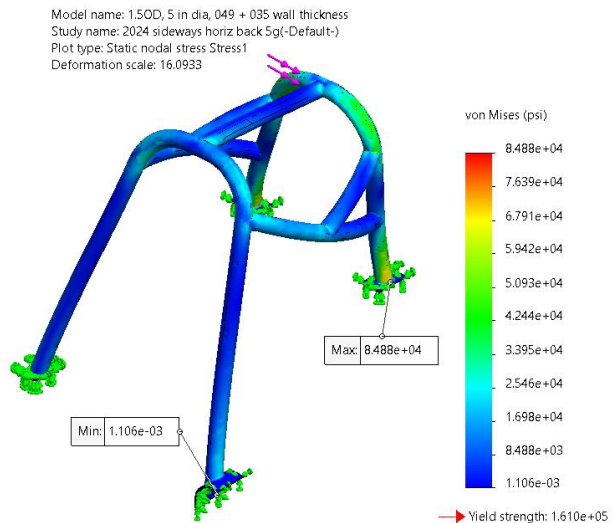


Figure 14: 5G Sideways load on rear hoop with 1.9 FOS

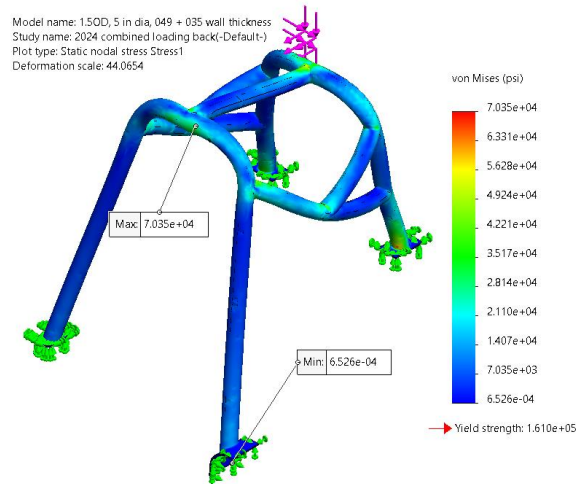


Figure 16: Combined (5G backward, 1.5 sideways, 5G down) loading on rear hoop with 2.3 FOS.

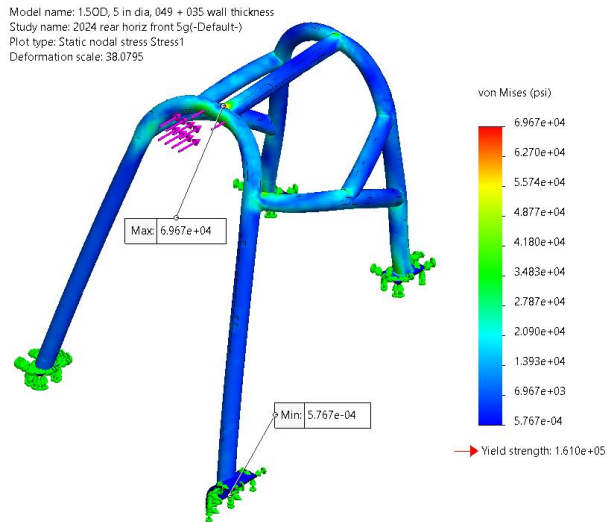


Figure 17: 5G Front load on front hoop with 2.3 FOS.

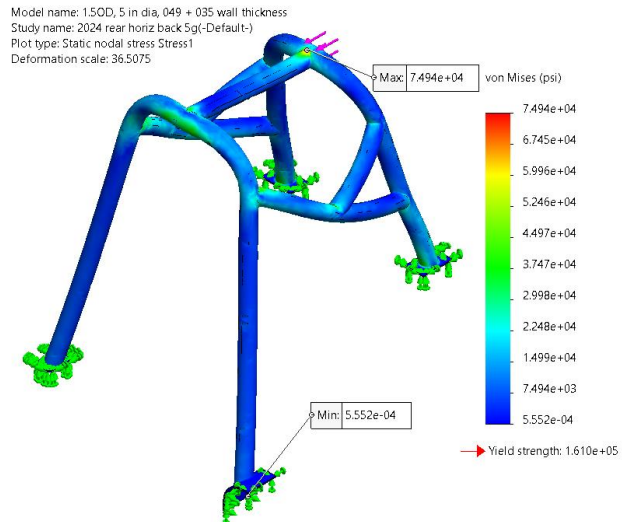


Figure 18: 5G Rear load on rear hoop with 2.1 FOS.

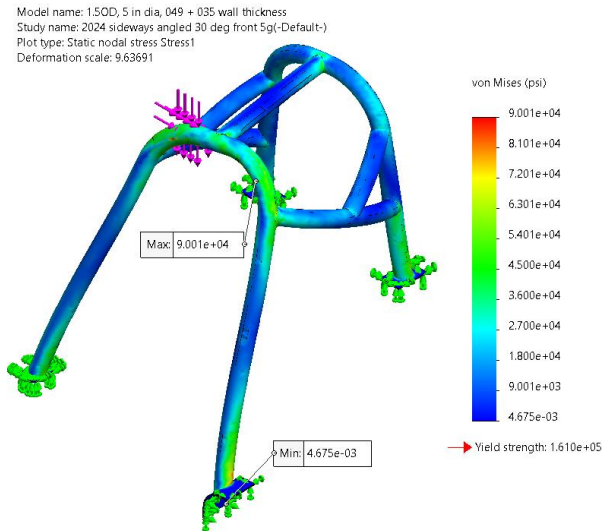


Figure 19: 5G 30 degree sideways load on front hoop with 1.8 FOS.

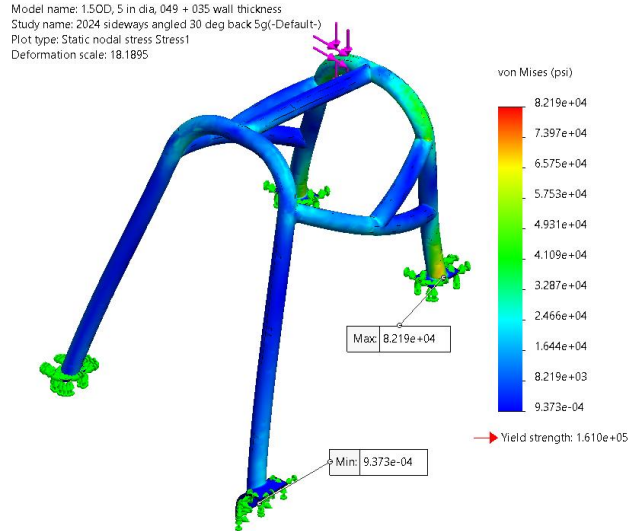


Figure 20: 5G 30 degree sideways load on rear hoop with 2 FOS.

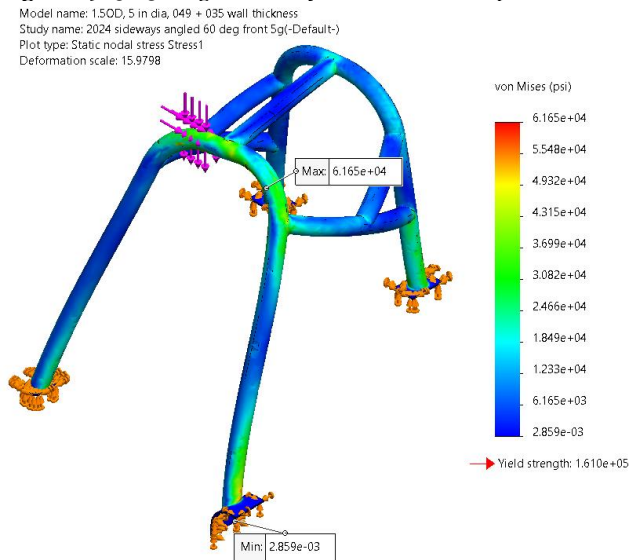


Figure 21: 5G 60 degree sideways load on front hoop with 2.6 FOS.

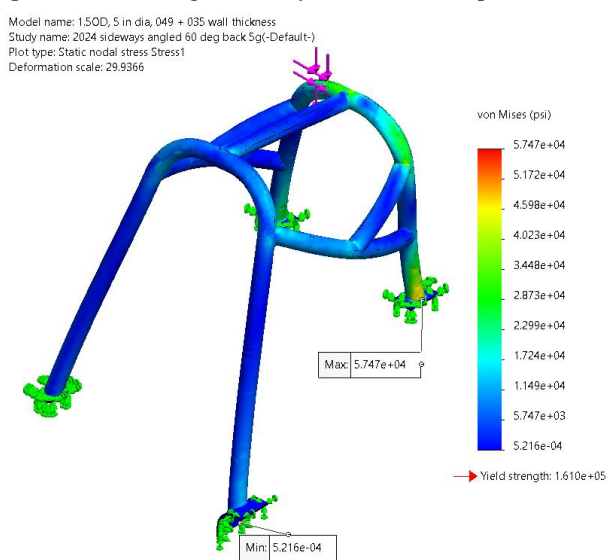


Figure 22: 5G 60 degree sideways load on rear hoop with 2.8 FOS.

SMART TRASHCAN

Under Construction

RIDER'S GUARD

Overview

As technology improves and the world becomes more aware of its carbon footprint, there has been decent momentum in the so called micromobility movement. With conventional transportation clogging the streets of busy metropolitan areas during critical hours, electric micro transportation systems like eskateboards, escooters, portable ebikes, and the popular OneWheel have begun to dominate the streets.

As the demand grows higher, new and improved versions come out frequently with better top speeds, longer battery life, increased wheel traction, lighter weight, and smoother braking. However, in none of these iterations was there a notion to prioritize safety features.

To help solve this issue, the Rider's Guard was designed for electric skateboards to provide awareness of the rider's presence to others (especially cars) sharing the road with a motion sensing brake light and also an emergency alert system capable of pinging messages to first responders and close contacts in the event that the rider has been "confirmed" to be incapacitated.

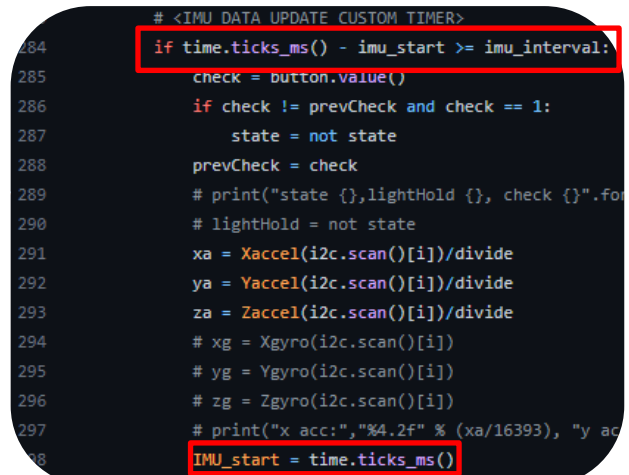
This was a course based project in which students had a short amount of time towards the end of the semester to create an IoT device of their choosing. Here are my contributions to the two-person project:

Performing Multiple Tasks Simultaneously

During this project, the idea of an interrupt method was unknown to us and wasn't mentioned in our class lectures. This initially made it difficult to design the ESP32 microcontroller to perform multiple tasks at the same time in the main while() loop. To remedy the issue, custom interrupts were made and widely used in our code.

While very simple to implement, it took some time and ingenuity. First, a time interval was specified to tell the program how frequently to check the code block. `time_ticks_ms()` imported from the time library keeps track of the system time in milliseconds since the start of the program.

Since `time.tick_ms()` records the absolute time passed, it is necessary to have a dynamic reference time for the system time to compare to. When the difference between the absolute time and reference time is greater than the specified interval, the code block runs and at the very end, the reference time inherits the time at which the code block finishes.



```
# <IMU DATA UPDATE CUSTOM TIMER>
284 if time_ticks_ms() - imu_start >= imu_interval:
285     check = button.value()
286     if check != prevCheck and check == 1:
287         state = not state
288     prevCheck = check
289     # print("state {},lightHold {}, check {}".format(state, lightHold, check))
290     # lightHold = not state
291     xa = Xaccel(i2c.scan()[i])/divide
292     ya = Yaccel(i2c.scan()[i])/divide
293     za = Zaccel(i2c.scan()[i])/divide
294     # xg = Xgyro(i2c.scan()[i])
295     # yg = Ygyro(i2c.scan()[i])
296     # zg = Zgyro(i2c.scan()[i])
297     # print("x acc:", "%4.2f" % (xa/16393), "y ac
298     imu_start = time_ticks_ms()
```

Figure #: This is an example of the custom timer used to check values from the IMU. `imu_interval` was set to 10. This meant that every 10 milliseconds, the microcontroller would check and update new values from the IMU to the variables `xa`, `ya`, and `za`. After this is done, `imu_start` takes on whatever the current system time is.

Determining an Emergency

One of the more interesting issues was determining what constituted an emergency given that the IMU data would be the main input into the ESP32. Was it when the rider hit something? The impulse readings would be easily distinguishable but what if the rider simply wasn't on the skateboard at the time of supposed accident? Should the device then be attached to the rider? But what if random movements from the rider triggered the brake light to turn on at inappropriate times? The list of possibilities for false accident cases was long.

To resolve the myriad of potential cases, I decided to mount the device to our skateboard. The device would trigger an audible alarm 3 seconds after no motion from the board was detected. The reasoning for this is that it is very hard for electric skateboards to be perfectly still even when a rider is waiting at a stop light. The

board will tilt side to side and the rider will naturally fidget back and forth so acceleration data will never be constant. The only time it would be is if the board is stationary with no rider input. There are two scenarios in which this could be the case: an accident occurred in which the rider is incapacitated or the rider is not standing on the stationary skateboard. Thus, the audible alarm was implemented to not only alert passerbys of the situation but also help distinguish between the two cases. If no accident actually occurred, the rider can stomp on the board to trigger a high enough impulse for the IMU to read or a large button could be to be pressed within 10 seconds to cancel the alarm. If 10 seconds passes by with neither of these inputs detected, the GPS, which was worked on by my partner, would ping an emergency to first responders and close contacts.

```

361 # Speaker Activation Count tracker. Will reset to zero if y acceleration
362 if abs(prev_xa-xa) < .3 and abs(prev_ya-ya) < .3 and abs(prev_za-za) < .3:
363     current_fall = 1
364 else:
365     current_fall = 0
366     alarm_start_check = 0
367     alarm_sent_check = 0
368     L2.duty(0)
369 prev_xa = xa
370 prev_ya = ya
371 prev_za = za
372 if prev_fall == current_fall:
373     fall_count += current_fall
374 if prev_fall != current_fall:
375     fall_count = current_fall
376 prev_fall = current_fall

```

Figure #: Speaker activation code designed to determine if there is 3 continuous seconds of no movement

```

29 # <NOTES SETUP WITH CORRESPONDING FREQUENCIES>
30 C3 = 131
31 D54 = 311
32 B4 = 494
33 F5 = 698
34 F56 = 1480
35
36 #<ALARM SONG SETUP>
37 song = [B4,F5,B4,F5]
38
39 L2.duty(85)
40 L2.freq(song[note_pointer])
41 if note_pointer < len(song)-1:
42     note_pointer += 1
43 else:
44     note_pointer = 0

```

Figure #: Song using the standard B and F notes used by police sirens. The two notes alternate every loop.

Interpreting Acceleration Data

This was one of the more difficult tasks. The first step is using the I2C protocol to transform raw data into raw acceleration values.

```

204 # <XYZ ACCELERATION PULLED RAW DATA>
205 def Xaccel(i2caddr):
206     xacc = int.from_bytes(i2c.readfrom_mem(i2caddr,0x28,2),"little")
207     if xacc > 32767:
208         xacc = xacc - 65536
209     return xacc
210 def Yaccel(i2caddr):
211     yacc = int.from_bytes(i2c.readfrom_mem(i2caddr,0x2A,2),"little")
212     if yacc > 32767:
213         yacc = yacc - 65536
214     return yacc
215 def Zaccel(i2caddr):
216     zacc = int.from_bytes(i2c.readfrom_mem(i2caddr,0x2C,2),"little")
217     if zacc > 32767:
218         zacc = zacc - 65536
219     return zacc

```

Figure #: Conversion of data stream into x, y, z acceleration values. 65536 and 32767 are calibration values to equate the acceleration due to gravity to the value 1.

Seeking an alternative, I made empirical observations of the IMU's behavior. After examining the graphed values of acceleration vs. time, I noticed very tall positive spikes and dips associated with the natural slight jerk from the motors whenever they kicked in reverse for braking. I decided to detect braking by measuring the jerk—after all, a collection of consecutive accelerations data points measured at specified time intervals easily gave it. Jerk could be determined with a simple comparison of acceleration values between two times. By incorporating a custom interrupt that updated very quickly, a set of criteria that looked for acceptable values that reflected deceleration and a closely calibrated brake threshold value of 0.4, I had our reliable brake detection.

The IMU is a MEMS device that works analogous to a weighted ball in a square box with pressure pads on all six walls. As a result, the IMU always measures the acceleration due to gravity by default in whatever orientation it is in. Thus, a measured value of 1 represents 1G. While acceleration data is good, it is insufficient without velocity. A classic example: zero acceleration could mean zero or constant nonzero velocity. To get the correct timing and situations for our brake lights to activate, we would have to integrate for velocity. However, an advanced algorithm, like the Kalman filter, would be needed to correct drift error due to measurement inaccuracies. With our tight project timeline, I realized that developing a corrective algorithm for this application was not feasible.

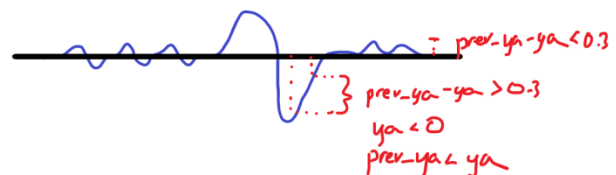


Figure #: A rough sketch of acceleration data vs time illustrating what happens when the motors begin braking. Jerk can be observed. The threshold of 0.3 was tuned by me after multiple empirical tests.

```

40 #IMU Data Update Custom Timer Inits
41 imu_interval = 10
42 imu_start = 0
380 if abs(prev_ya-ya) > .3 and prev_ya < ya and ya < 0 and abs(xa) < .4:

```

Figure #: Just like the collection of IMU data, the ESP32 compared the previous and current acceleration data every 10 milliseconds.

The Setup

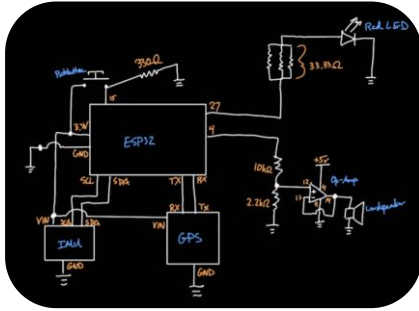


Figure #: Circuit Diagram. While not professionally drawn using KiCad, it conveys the essence of how the electronics work.

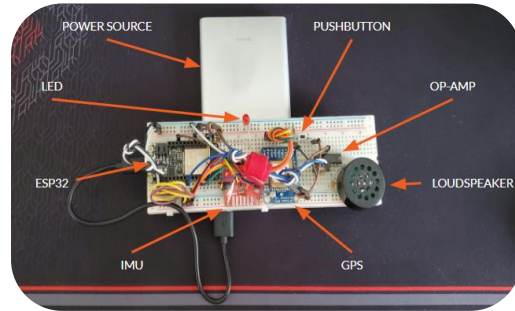


Figure #: Hardware Diagram. While crude, it can be mounted on the electric skateboards that we both own using temporary adhesives.

At the center lies the ESP32 microcontroller which is our minicomputer. The entire device was powered by a portable charger rated with 5 volts since the ESP32 can officially hand it and will be required by the loudspeaker component. The GSP module uses the UART protocol and has 4 wires with 2 connecting to the TX and RX pins and 2 connecting to ground and power. The IMU module uses the I2C protocol and also has 4 wires with 2 connecting to the SCL and SDA pins and 2 connecting to ground and power. The SCL pin is designed to sync the microcontroller clock with the IMU module while the SDA sends the data package. A pushbutton, designed to cancel the alarm, connected in series with a 330 Ohm resistor to ground and power and GPIO pin 15. An LED connected in series with parallel resistors to ground and GPIO pin 27. Finally, the most complicated portion, the loudspeaker, used a TLV2734IN operational amplifier connected to 5V using the onboard USB pin to amplify the sound. The conventional 3.3V output is insufficient to drive the loudspeaker rated with a resistance of 8 ohms and power of 1W; more current is needed or else it would be too quiet, rendering it useless in our situation.

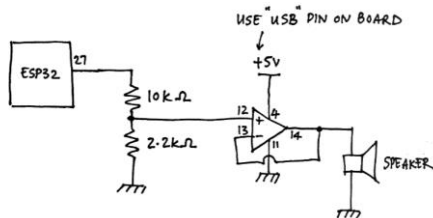


Figure #: Non-inverting op-amp in voltage follower mode. This creates a buffer between the high impedance input and low impedance output (high current output)

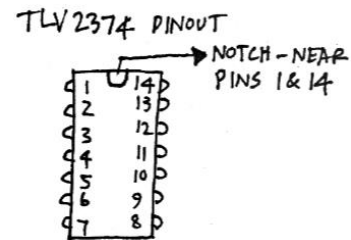


Figure #: Op amp diagram. Only half the pins are used since this can support 2 separate op amps.

Improvements

We mainly used MicroPython for this project because of our familiarity with Python. This allowed for the rapid prototyping and testing of code changes, which was especially useful since we had an ambitious project but a tight timeline. However, MicroPython is known to suffer from performance issues because requires a Python based interpreter to translate higher level code line by line into machine language. This can take 3-4 times longer while also requiring more memory. In later stages, we hope to switch to Arduino since it is similar to C, which requires only a compiler. Thus, the entire source code can be translated into machine code prior to execution.

Additionally, components are wired using jumper cables and facilitated with a breadboard. In future iterations, we plan to have a custom PCB board with smaller electronic components and delve into detail the Kalman Filter so that drift error from the IMU can be mitigated. Additionally, for louder sounds, we would switch to the BS170 NMOS mosfet because of its ability to pass a higher current. We would also like to consolidate the device into a smaller form factor with a protective casing such that it can be attached to most popular micromobility transportation systems using a universal interface instead of just electric skateboards.

GRABBER CANE

3D S.T.A.R. Logo

Every year, S.T.A.R. has a crowd funding event designed to raise money so that the different projects supported by the club can continue its research and development. As an act of gratitude to our generous donors, our club's business team prepares countless thank you gifts. In the academic 2021-2022 year, I offered to create a 3D version of our club's logo that can be placed proudly on any desk.

After a discussion with the business team about the design specifications and delivery date, I went to work using SolidWorks to create the logo.

This project consisted of 5 components in total, in which 3 were custom designed. When creating each part, I took great advantage of the 'loft' and 'revolve' features to create many of the complex contours. The 'mirror' and 'rotate geometry' features were used strategically as well to reduce the design time and ultimately get the desired 3d look. Multiple planes were also used to facilitate projection of sketches onto curved surfaces and creating sketches not on the given coordinate planes.

Uncolored Final Iteration of the Custom Parts



Figure #: Isometric view of Main Frame showcasing revolve, mirror, and projection sketches.

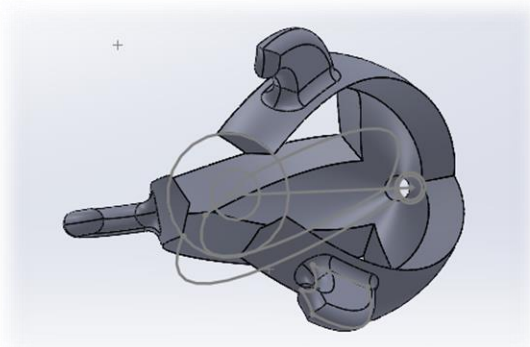


Figure #: Isometric view of the underside of Rocket Model showcasing projection and revolve sketches.

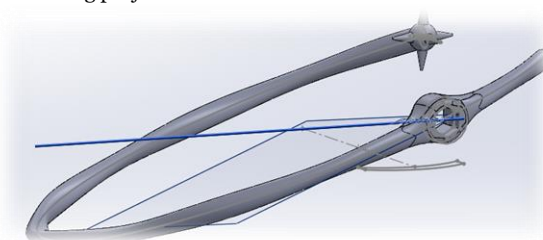


Figure #: Isometric view of Shooting Star Attachment showcasing two sketch planes offset rolled 11.5 degrees and pitched 22 degrees.

Since these 5 components were designed to be assembled together, I had to consider the type of fit needed as well. The business team also requested a design that could fit in a relatively small box. This meant that the Shooting Star Attachment could not be permanently fixed onto the Main Frame.

Thus I had to design hexagonal cutouts in the Main Frame and Shooting Star Attachment components so that fit a 0.5-inch hexagonal aluminum standoff could be slip fitted on. While no standard fit tolerance charts have been made for hexagonal cross sections, I was still able to tolerance using prior experience. From flat to flat, the hexagonal cross section is a nominal 0.25-inches but measurements from 10 different samples hovered between 0.249 and 0.255. Knowing that the Markforged X7 industrial printer has an accuracy of ± 150 micrometers, I sized the hexagonal cross sections to 0.26-inches from flat to flat. This created a nice slip fit.

Snapshots of the Design Progress



Figure #: Original 2D Logo



Figure #: Rendered 3D Logo



Figure #: Pre-processed 3D Logo



Figure #: Post-processed 3D Logo