V3 Centrifuge Design Cycle: Assembly, Testing and Documentation

By Benjamin Zaidel, Andrew Churukian, Montanna Riggs

BIOE123: Systems Prototyping Lab

Block Diagrams

For our V3 centrifuge we created multiple block diagrams to understand our individual subsystems. We created a [whole system block diagram] to show how all of our systems will work together– how their outputs act as inputs. We then proceeded to break each system down individually. Our block diagrams are broken down into the actuator subsystem, sensor subsystem, and control subsystem.

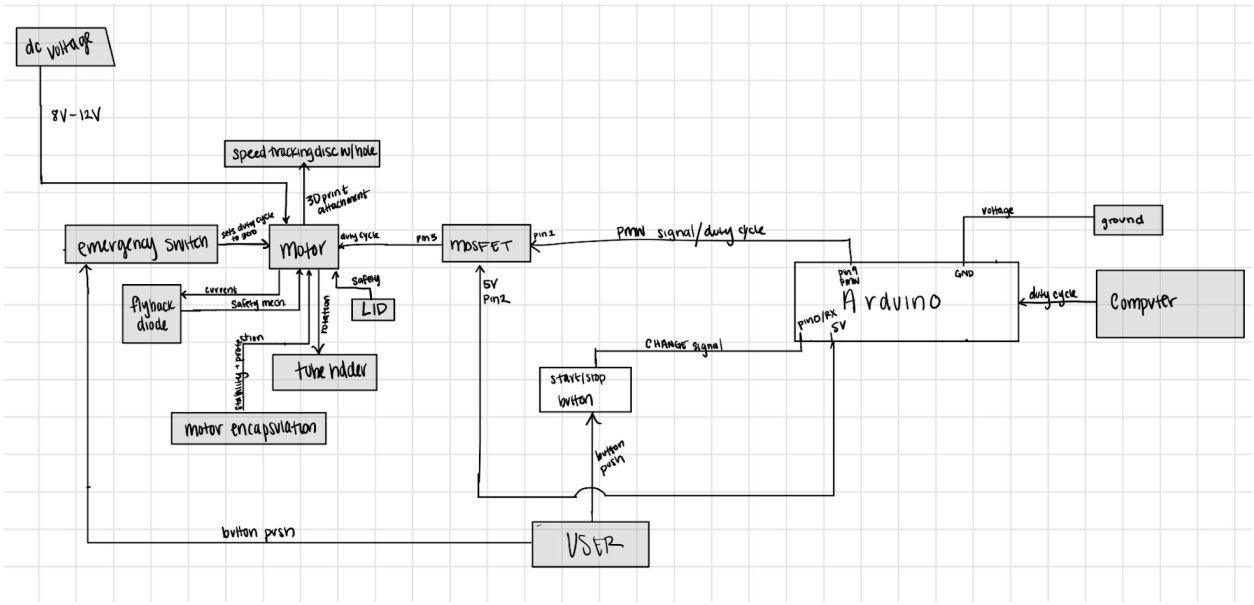

Figure 1. This is our updated V3 actuator subsystem block diagram with our lid incorporated. The purpose of this block diagram is to show the parts that are responsible for controlling the movement of our centrifuge. This includes the on/off switch, the emergency switch, and the mosfet.

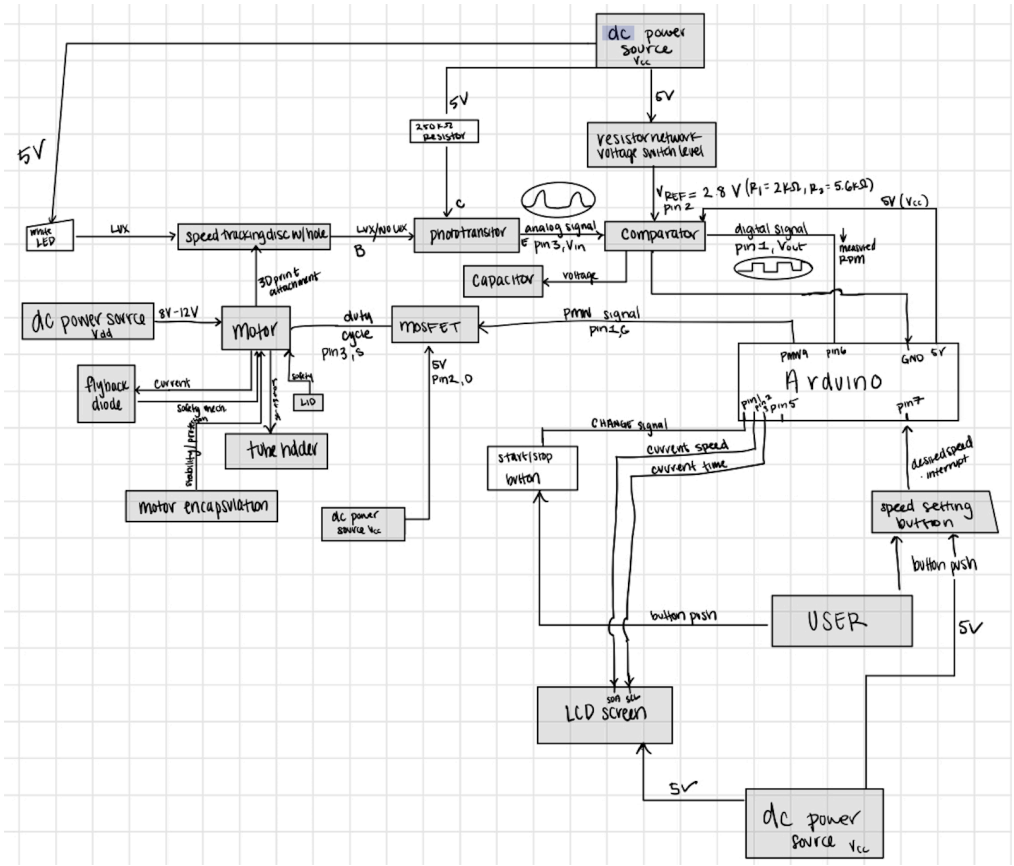

Figure 2. This is our udpated V3 sensor subsystem block diagram with updated Vref, updated arrows, and more. This diagram shows the different components involved in detecting changes. This includes the phototransistor, start/stop button, speed and time setting button, the mosfet, and the comparator.

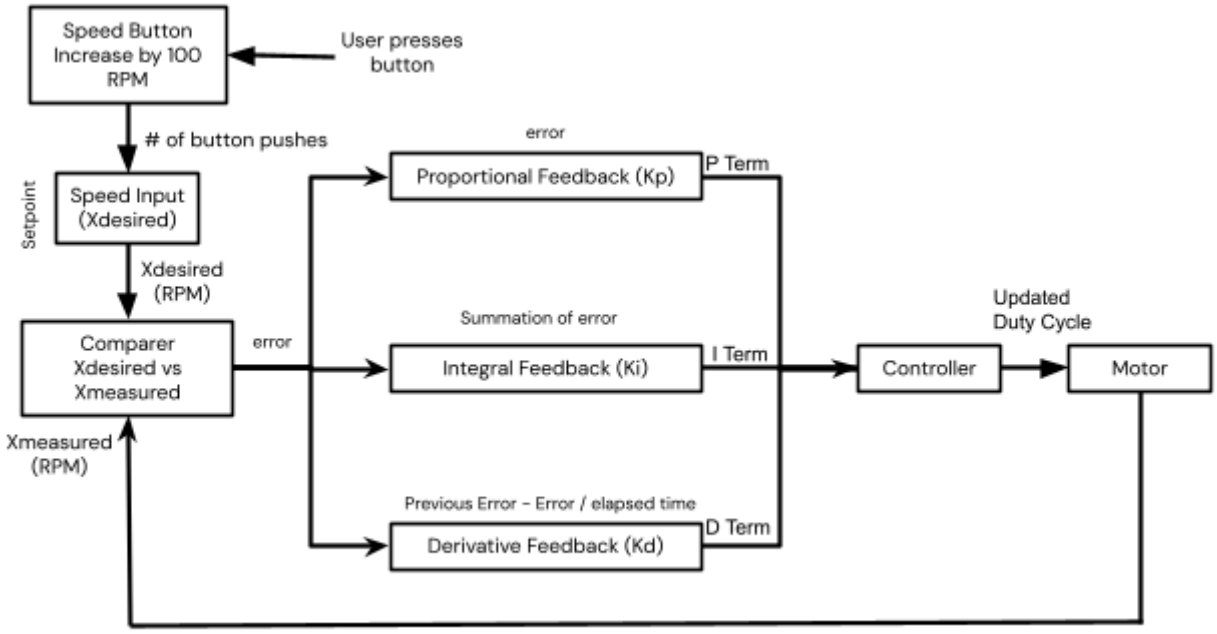

Figure 3. This is our update V3 control subsystem block diagram. This diagram shows how the arduino will work to control the larger system. For our centrifuge we have two control systems, an open-loop control and a proportional control. The open-loop control has no feedback system to regulate the output, while the proportional controller does.

Circuit Building - Andrew Churukian

Our circuit design can be broken down into 5 main parts: the motor, LED, comparator, LCD screen, and buttons. The circuit is powered by an arduino and 5 Volts from a DC power supply. See our annotated circuit diagram below.

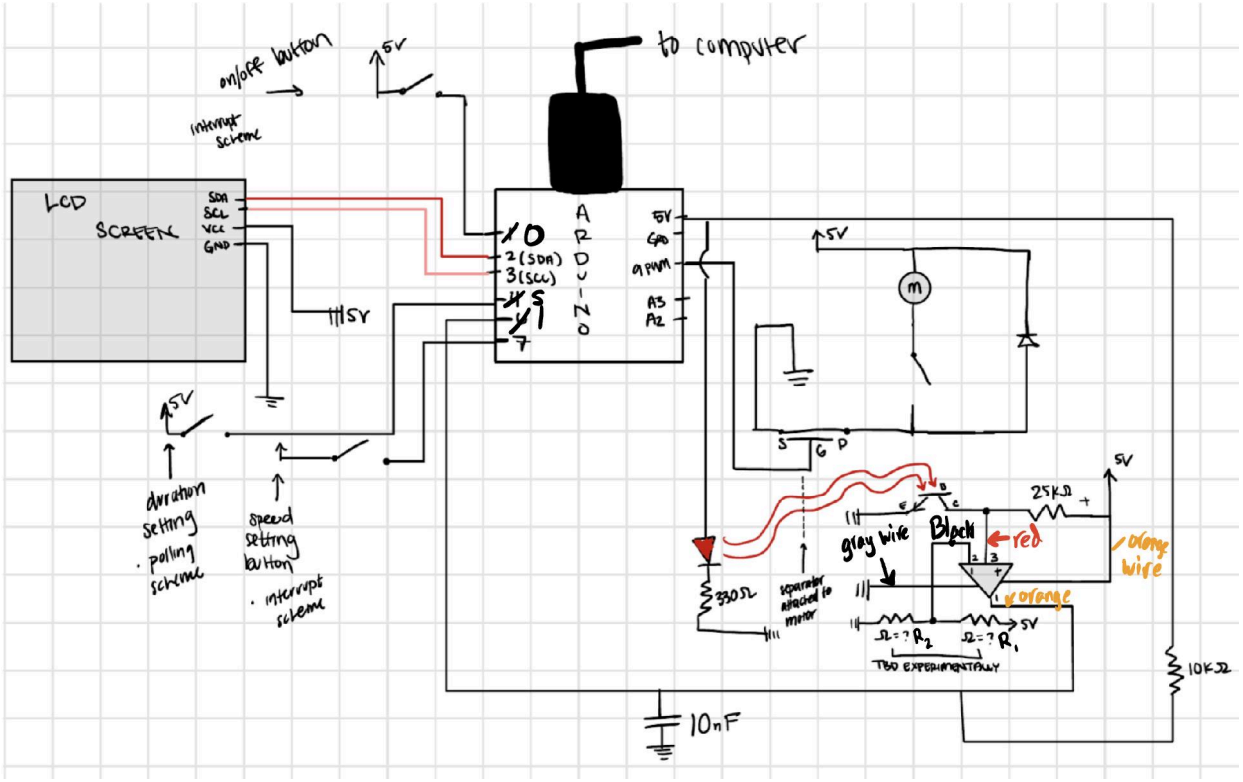

Figure 4. This is our annotated circuit diagram. An issue we ran into was using incorrect pins on the arduino, so a lot of our troubleshooting for the circuit was determining the correct arduino pins. We decided to label the color of the wires used for the comparator to make it easier when looking at the breadboard circuit. Note: R1 = 1.5 KOhms and R2 = 5.6 KOhms. The reasoning behind these values is explained in the Comparator Circuit section.

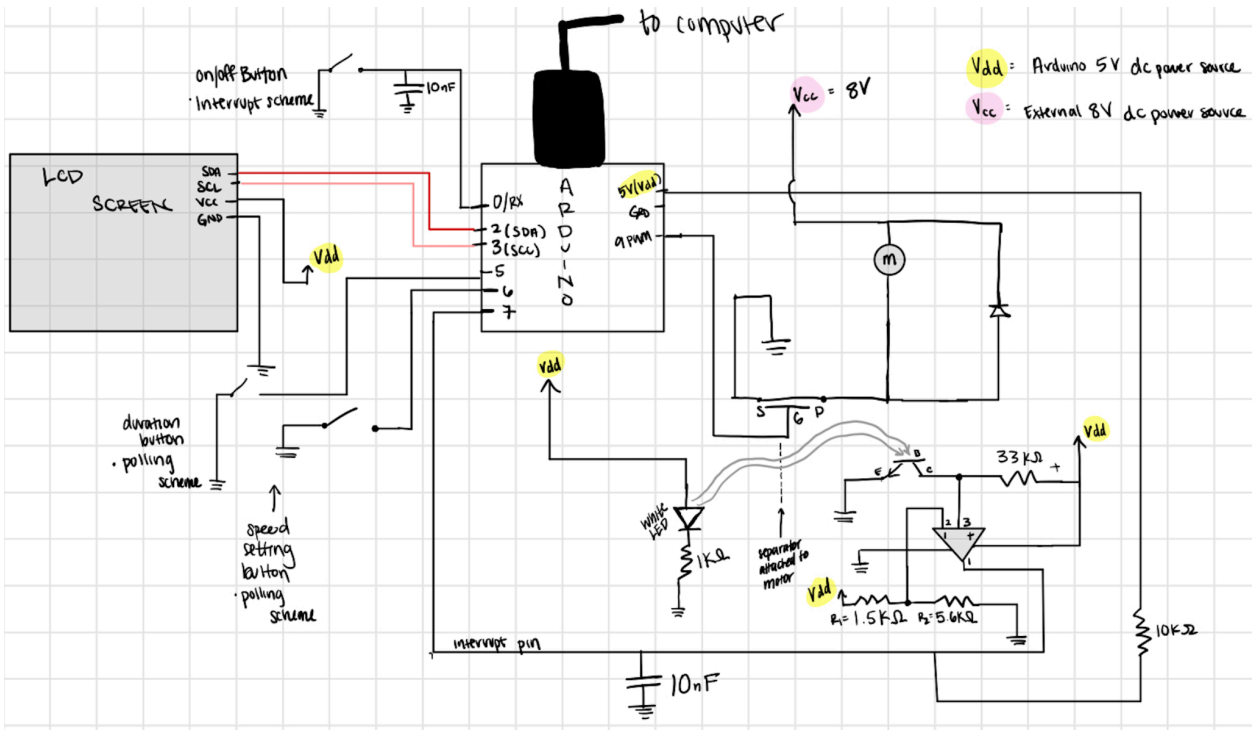

Figure 5. Shows our complete updated V3 circuit diagram with updated pins, resistors, LED types, and button circuits. Most notably, we clarified our power sources by isolating the motor circuit power from the other arduino backed power inputs so that we could raise the motor voltage to 8V and added a capacitor to our on/off button scheme to reduce oscillation at the voltage switch.

Motor Circuit

For the motor part of the circuit we used a mosfet connected to the PWM pin 9 to control the duty cycle of the motor. The motor is connected to both a 5 Volt power source and ground. In addition, a flyback diode is connected in parallel with the motor to prevent voltage spikes and to help dissipate current. When the motor is turned off, due to the variance in signal from the duty cycle in the mosfet, a build up of voltage can occur, which can damage the circuit. The diode prevents this damage from occurring in the motor. The wires connecting to the motor are threaded through our circuit encapsulation.

For V3, we were having difficulty with trying to get our motors speed to increase when we increased the voltage. We wanted to increase the voltage, to allow for our centrifuge to reach our desired max speed of 2500 RPM. Originally, we had

the DC power supply connected to other aspects of the circuit, which made it difficult to increase the speed. We ended up reconnecting the circuit so that only the motor was powered by the DC power supply, and everything else was powered by the 5V source of the arduino. This immediately fixed our problems and allowed us to increase the speed of our centrifuge and allowed us to hit our max speed.

LED Circuit

The LED part of the circuit uses a white LED light. We chose the white LED light over the red LED because it appeared to shine brighter, which we thought would be useful to use with our phototransistor. We calculated the voltage drop of a white LEDand used a 100 Ohm resistor. We threaded the LED through our circuit encapsulation, and positioned it just outside of our motor encapsulation. Our phototransistor was put inside the motor encapsulation and we made a small hole in the motor encapsulation to allow for the LED to shine on the phototransistor each time it rotates. We used this to track the RPM of our centrifuge.

Comparator Circuit

The comparator we used had 8 pins. Pin 3 was used for the input voltage and was connected to a phototransistor, which allowed current to flow through when the LED lined up with the phototransistor. This produced varying voltages at pin 3, to control the signal outputted by the comparator. Pin 2 was connected to the reference voltage, which was compared with the input voltage. We chose two resistors to set a constant voltage. See the figure below to see how we calculated the two resistor values. Pin 1 was used for the digital output signal, which either produced a high or low voltage depending on if Vin is greater or less than Vref. We connected a pull-up resistor and a capacitor to the digital output as well to help with the signal. In addition, we connected pin 4 to ground and pin 8 to a voltage source. We used the comparator to track the revolutions to produce an RPM of our centrifuge.

When going from V2 to V3 we had to adjust our V reference point. We began to address the issues of our internal tachometer system and we measured the

voltages of Vin and Vref. We found that the Vref was not low enough, which was preventing the ability of our system to count each time the centrifuge completed a rotation. To fix this we recalculated R1 and R2, so that we had a reference point of about 3.7V. When we adjusted R1 to 2 KOhms and kept R2 at 5.6 KOhms, this helped to fix our issues and we began to see a reading on the LCD screen.

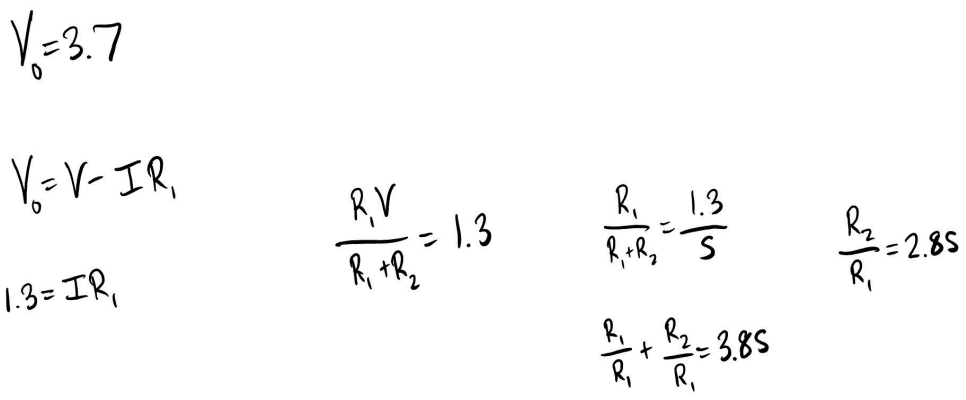

Figure 6. To figure out the resistors we needed to produce the optimal reference voltage for our comparator, we used the above known values and equations. We determined that an R1 of 2 KOhms and an R2 of 5.6 KOhms would give the best Vref.

LCD Screen Circuit

Our LCD screen was used to produce visuals for the user, so they would know the current speed of the centrifuge and the amount of time remaining for the centrifugation. We connected the SDA and SCL pins to pins 2 and 3, respectively. Also, we connected the LCD to a 5 Volt power supply and ground.

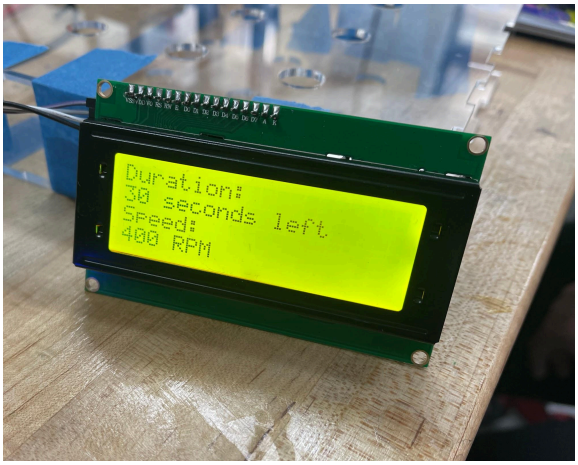

Figure 7. LCD screen showing the time remaining and speed of the centrifuge. Shown again in Figure 18 below.

Button Circuits

Our circuit contained three buttons connected to our arduino with two buttons on an interrupt scheme and one button on a polling scheme. The On/Off button was connected to an interrupt scheme. We did this to ensure the user would be able to start and stop the centrifuge at any point in the cycle. It was important for our design to be able to stop at any point if there is a problem with the centrifuge. Our time setting button and speed setting button were connected to a polling scheme. We chose to do this as we felt it was unimportant to be able to change time or speed at any point, and our team felt that possibly needing to press the button multiple times was not a big issue. In addition, we needed to use many of the interrupt pins for other parts of the circuit, so we did not have the pins available to make these buttons on an interrupt scheme. We wanted to produce an emergency shut off button for our circuit. This button would be used when the circuit needs to be stopped immediately for safety reasons. Unfortunately, we were unable to make this button because we didn't have time, due to troubleshooting more substantial errors.

For our V3 circuit, we noticed that when we clicked the on/off button we experienced some errors where the motor would immediately reset when you initially clicked the button. To fix this, we added a capacitor, which prevented the

button from oscillating when clicked. In addition, we wanted to be able to click the buttons without the need for a separate breadboard and we wanted to be able to attach the buttons to our circuit encapsulation box. To do this we soldered wires to the buttons and connected them to their respective arduino pins. This allowed users to click the button and observe the LCD screen in the same place.

Aesthetics

For our V3 design we wanted to organize the circuitry and wires on the breadboard to make it more appealing to the user, since our circuit encapsulation box was see-through. To do this we used wire cutters to trim down wires that were too long. This resulted in most of the wires and resistors laying flat on the breadboard. In addition, there were a few wires that were being held to their parts by tape, like our phototransistor. We decided to solder these wires to prevent the wires from accidentally being pulled off while testing.

Software Construction - Montanna Riggs

Our software construction can be divided into 7 subsystems: duration setting software, speed setting software, tachometer software, control software, on/off button software, LCD software, and motor software . All seven integrations are designed to enable the successful completion of the centrifuge requirements. V3 software approach focused on troubleshooting subsystems with a bottom up approach.

Duration Setting Software Subsystem

Our duration setting subsystem is designed to enable requirements 2.2.1, 2.2.3, and 2.2.3. To enable user time setting in increments of 30 seconds, we created a button counting polling scheme with a range of 30 seconds to 10 minutes. To test accuracy, we built in a countdown timer tracked via LCD screen into our software.

Speed Setting Software Subsystem

Our speed setting subsystem relies on user input as well to meet requirements 2.1.1, 2.1.2, and 2.1.3. We collected a desired speed via a button counting polling scheme in increments of 100 RPM with the range of 0 to 2500 RPM. To test accuracy, we incorporated an internal tachometer and PID control system described separately below due to its complexity.

Tachometer Software Subsystem

Our tachometer software relied on a LED-phototransistor circuit to track RPM. This integrated system relies on the integration of software, circuit, and mechanical subsystems to communicate the number of rotations. The circuitry uses a comparator to convert analog phototransistor output signals to digital output sent to the arduino. Software then calculates the current speed using the Arduino's internal timer and number of rotations in 2 second intervals.

Control Software Subsystem

Our control software subsystem utilizes PID feedback to minimize error. Using experimentally determined PID constants, we designed the system to address the speed error within the 30 second ramp up requirement by fine tuning the duty cycle. We were able to use the serial monitor to improve the system with documentation under integration.

On/Off Button Software Subsystem

Our on/ off software is coupled to an on/off button with an ISR scheme to track button state (either 1 or 0). This allows the user to control the initiation and cancellation of the centrifuge as per requirement 3.1 and 3.2 before and during centrifugation. It also acts as a safety mechanism to prevent the centrifuge from starting before the user chooses a duration and a speed as well as inserts all their samples.

Motor Software Subsystem

Our motor software subsystem writes duty cycle to the MOSFET while using the updated duty cycle from the control software subsystem to continually minimize error.

LCD Software Subsystem

Our LCD system is meant to provide the user feedback we had previously understood from the serial monitor in the Arduino IDE. Our software allows us to communicate the desired speed and time the user inputted as well as the current speed and time left once the device starts running. It also comments “All Done!” when duration runs out and comments “Reset!” if the user stops centrifugation mid duration.

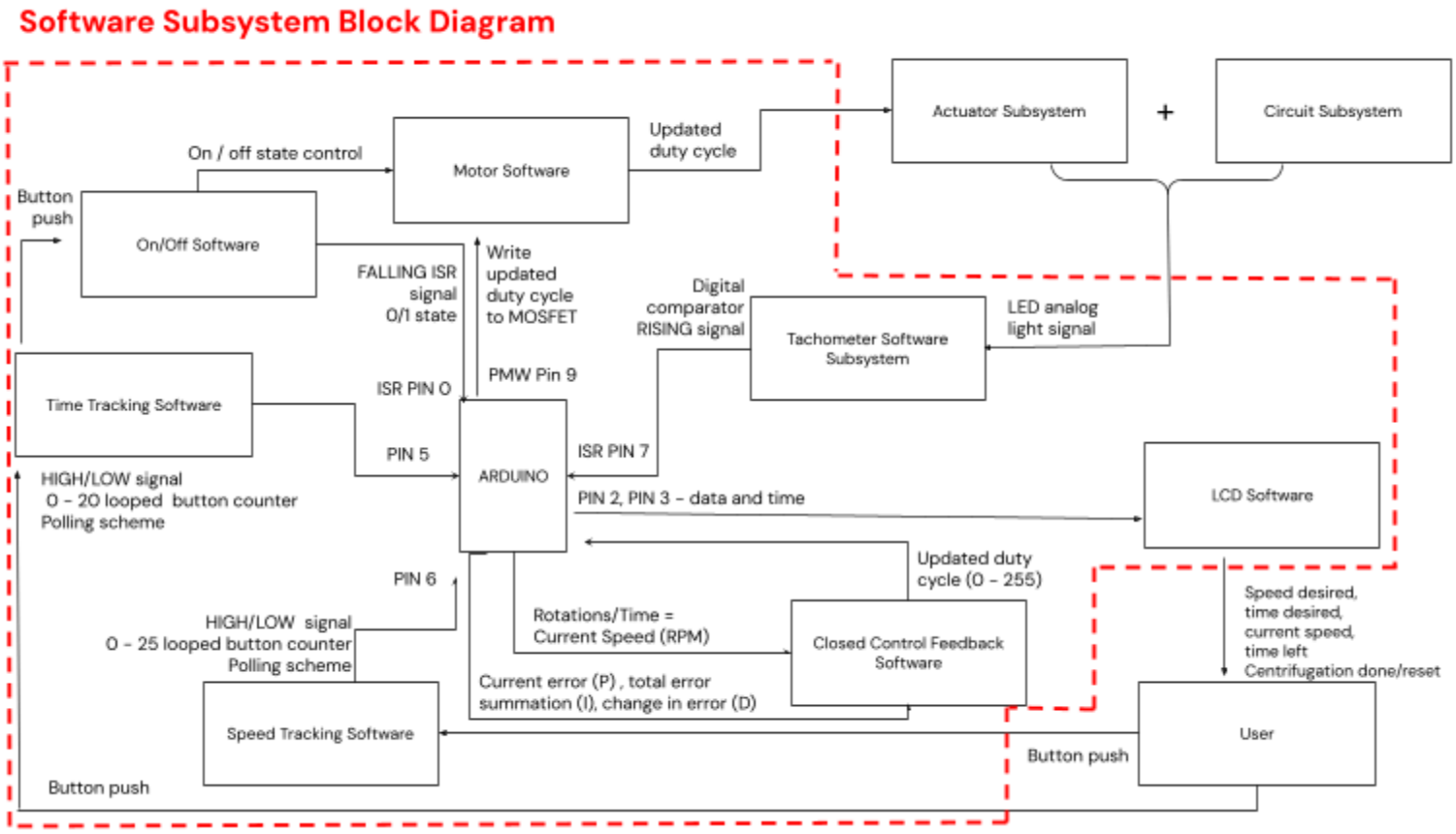

Figure 8. shows the block diagram for our software subsystem design.

The design of the software subsystem began in pseudocode during V2. By delineating the functionality of the code in blocks, we were able to imagine the interactions between software subsystems as well as circuit subsystems and actuator subsystems. We can also use pseudocode to sanity check our general understanding. We also relied heavily on our block diagram to keep track of the inputs and outputs traveling through our software as well as the circuit diagram to keep track of pins and to sanity check our signals.

V2 Centrifuge Pseudo Here.

Using our V2 pseudocode as a baseline, we assembled out our full code in software blocks. We quickly learned in V2 that attempting to write all the code at once and then to test all the code at once makes for complex and frustrating

troubleshooting. Frustrated and exhausted, we ended our V2 design in a stalemate with our tachometer. However, entering the V3 with stronger design principles, we were able to integrate the software, circuit, and mechanical hardware needed to support the tachometer subsystem (more under integration).

TinkerCAD proved useful to an extent with assembly and testing. This included simple debugging to redesigning and simplifying systems. TinkerCAD was specifically helpful to analyze the interactions between circuit and software subsystems e.g. we tested all our button schematics in TinkerCAD before building them out shown in figure 8. This was extremely helpful for troubleshooting initial code bugs and allowed for quick circuit changes as well.

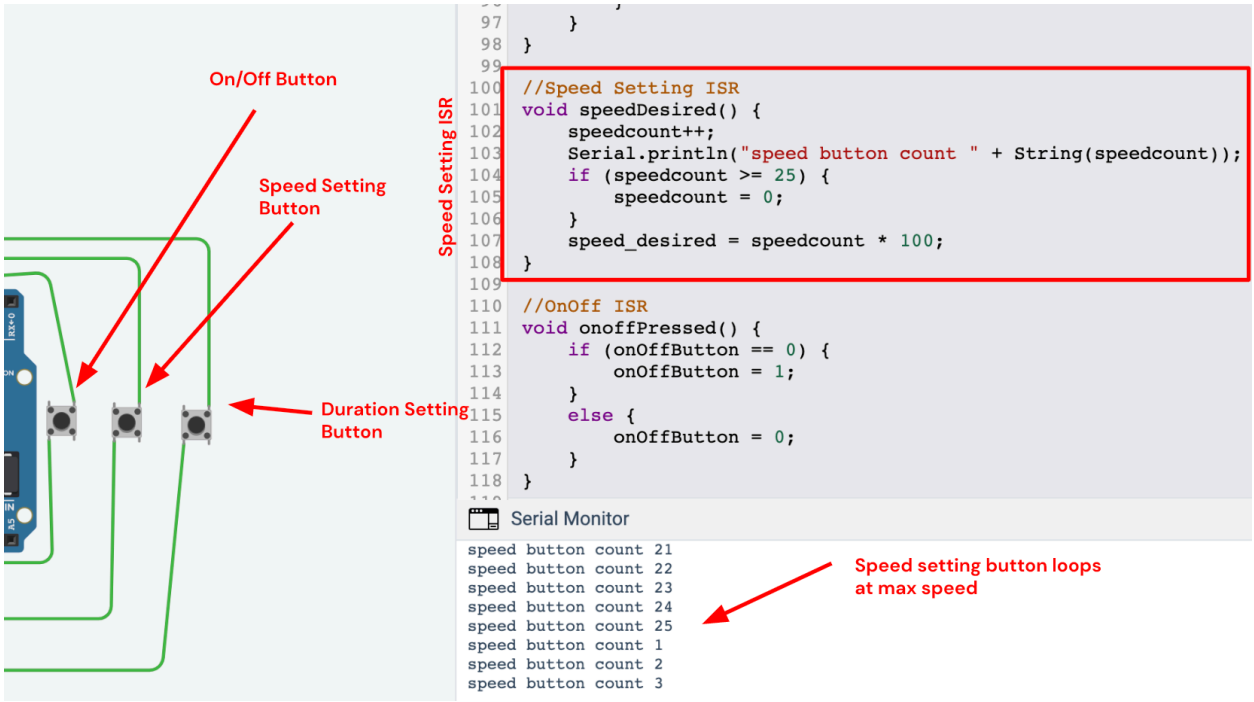

Figure 9. shows an example TinkerCAD simulation of our speed setting software subsystem and circuitry. The speed setting button is tracked in an ISR which calculates the intended speed in increments of 100 RPM.

A notable software difference enabled by the functional tachometer subsystem was our PID control feedback loop. We decided to implement this code by scratch tracking, error, previous error, and a summation of error. We experimentally fine tuned the constants to meet the 30 second ramp up requirement with our final values at Kp = .01 , Kd = .001, and Ki = .0001. The serial monitor proved to be an excellent resource during these tests.

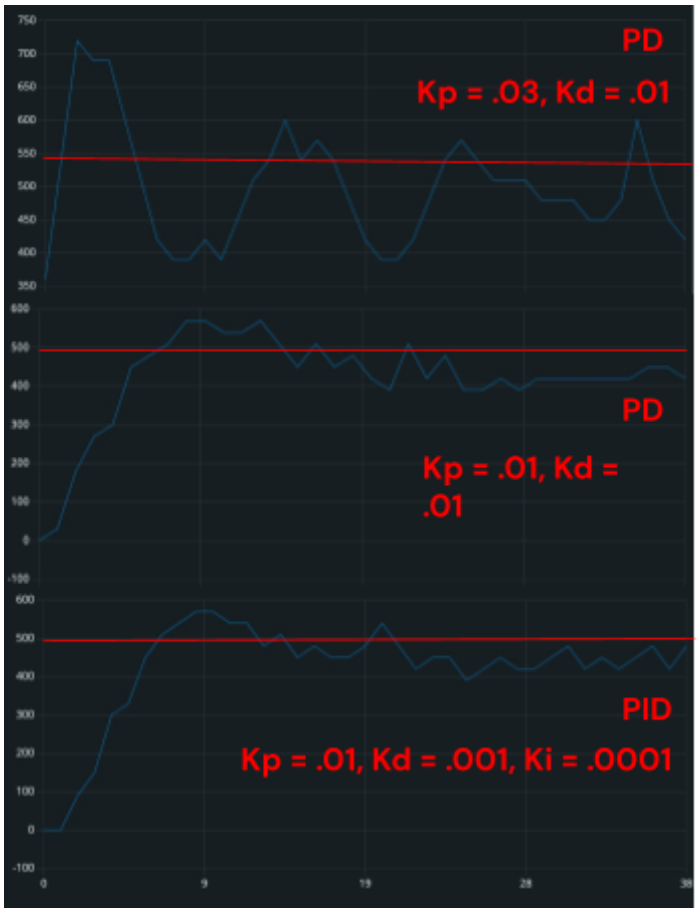

Figure 10. shows iterations of our control system software through three different time (seconds) vs RPM graphs with a goal RPM of 500. From top to bottom, the control system becomes more accurate both in time to stabilization and stability. The first two iterations were PD control systems and the last has integration incorporated. We left V3 using the

```
//Control Feedback Loop with PID Control
  if (elapsedTime % 2 == 0) { //udpate speed control every 2 seconds
    if (!errorstate) {
      error = speed_desired - s_measured; // general error
      error_integral += error; // integral error is the summation of all error over the course
      if ((s_measured < lower) || (s_measured > upper)) {
        errorDeriv = (previous_error - error) / 2; // derivative error uses the prior error
        duty_cycle = duty_cycle + .01 * error + .001 * errorDeriv + .00001 * error_integral; //

        //Edge Case Protection
        if (duty_cycle >= 255) {
          duty_cycle = 255;
        }
        if (duty_cycle <= 0) {
          duty_cycle = 0;
        }
        previous_error = error;
      }
    errorstate = true;
    }
  }
  else {
    errorstate = false;
  }
  analogWrite(MOTORGATE, duty_cycle); // write updated duty cycle to the MOSFET
```

Figure 11. shows the PID software subsystem in its final V3 form. Like our tachometer, we measure error every 2 seconds. Duty cycle edges cases are accounted for before the system sends the updated duty cycle to the MOSFET.

With our control system and tachometer in place and our code debugged, we vastly improved the V3 code from V2. Our complete V3 code can be found here. While elements are similar to V2, it's much more robust. Key changes are listed below.

Main Software Updates	**V2**	**V3**
Tachometer	Our tachometer was not working and we traced it back to a code issue as the comparator was sending signal to the Arduino.	Through the troubleshooting process explained in integration, we sourced the bug and validated our tachometer subsystem as an accurate measure of speed +/- 5% error
Resetting	Inability to reset the centrifuge (run more then once) appeared as an interesting bug we ran into during V2. Instead, we had to reupload the code every time we ran it.	We sourced the issues to a matter of "states" in our code that were not tracked. State 1: On/Off button in off state, no duration on the centrifuge

		State 2: On/Off button in on state, duration is on the centrifuge State 3: User stops the centrifuge mid spin, On/Off button in off state, duration is on the centrifuge State 4: User does not stop the centrifuge, duration runs to zero, On/Off button in on state (since no button was pushed) The difference between the later two states made resetting tricky, but once we knew the issue, a quick state tracking system solved the bug!
PID Control	None existent due to lack of internal tachometer functionality. No way to achieve feedback.	Once the tachometer was running smoothly, we incorporated PID to fine tune current speed
Buttons	Speed setting button in an ISR scheme on pin 1 meant we were unable to use the serial print function to troubleshooting efficiently and effectively	Shifted the speed setting button to polling scheme and decided that ease of troubleshooting trumped the design idea to allow users to change the speed mid centrifuge

Following the completion of all software subsystems, we integrated with circuit and mechanics.

Mechanical Build - Benjamin Zaidel

I was responsible for assembling the mechanical subpart of the V3 build (see draft of V2 build here). Our original vision was to have a motor + rotor base + tube holder complex that was completely encapsulated by an independent 3D printed structure. In making the encapsulation stand-alone, we were hoping it satisfied Requirement 7 by protecting the motor from any physical harm as well as

preventing the user from coming into contact with any spillage. Below is a list of changes that were made to the mechanical subpart from V2 to V3, all of which will be discussed at some point in this section or in the integration section. A major ideological change from V2 to V3 was the idea that the mechanical subsystem does not exist in a vacuum; many of the new changes made were done during integration such that the mechanical system seamlessly worked with both the circuit and the code.

Mechanical Subpart Change	**Relevant Requirement**	**Discussed in section**
External encapsulation flipped upside down	1.1	Mechanical
Radius of internal tube holders reduced	1.1, 2.1.1	Mechanical
Supporting box had etched labels	2.1.2, 3.1, 2.2.2, 3.2	Mechanical
Buttons soldered, made larger	2.1.2, 3.1, 2.2.2, 3.2	Integration
Phototransistor soldered to breadboard	2.1.3	Integration
LCD glued into box	4.1	Integration
Acrylic dome + lid added on hinges	7.1	Mechanical
Supporting box made larger	7.3.1	Mechanical
Tube holders attached to external encapsulation	9.1 (new)	Integration
Attached to 9V wall plug	n/a (extra)	Integration

Before generating any computer-aided designs, we started with sketches made from estimations based on the size and height of the motor.

Sketches

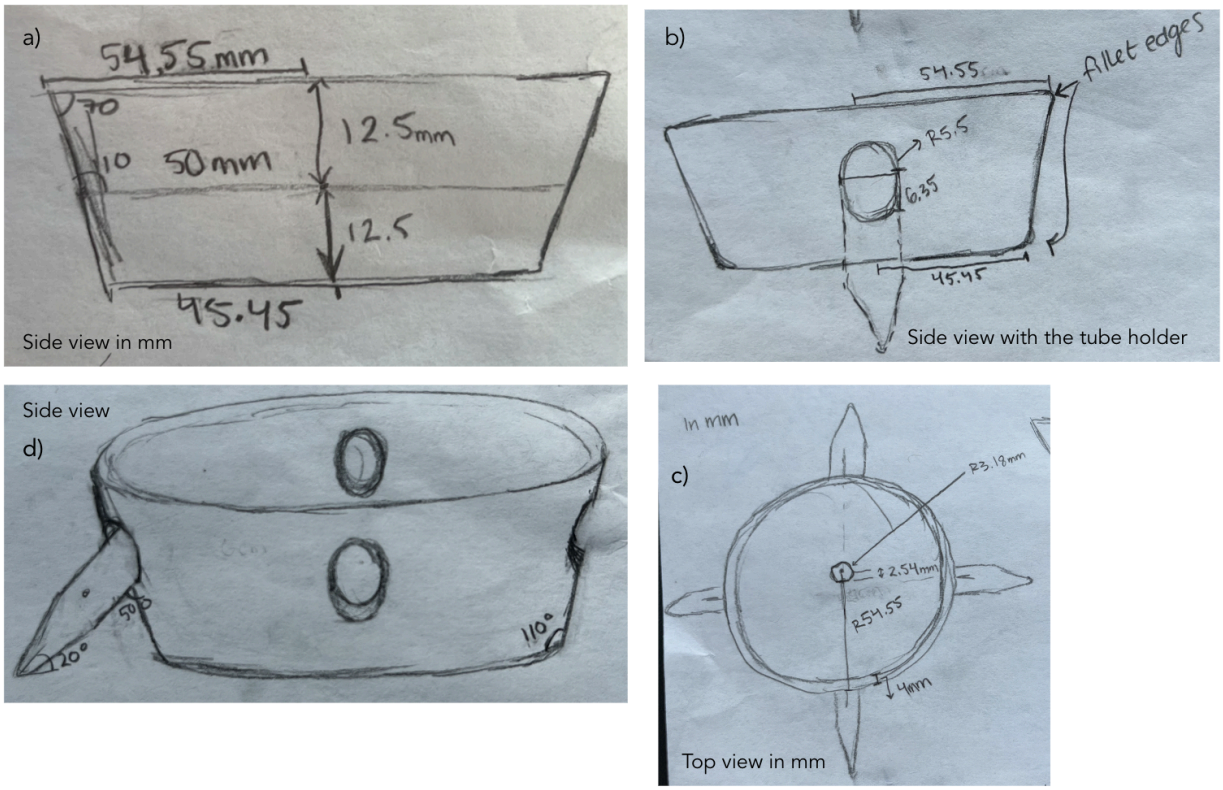

Figure 12. All measurements in mm. A) shows a side view of the tube holder shown geometrically. We chose a trapezoidal shape so tube holders lay flat to increase stability. B) shows a side view of the tube holder with the tube hold included. C) shows the top view of the tube holder D) shows a realistic rendition of the side view with angles.

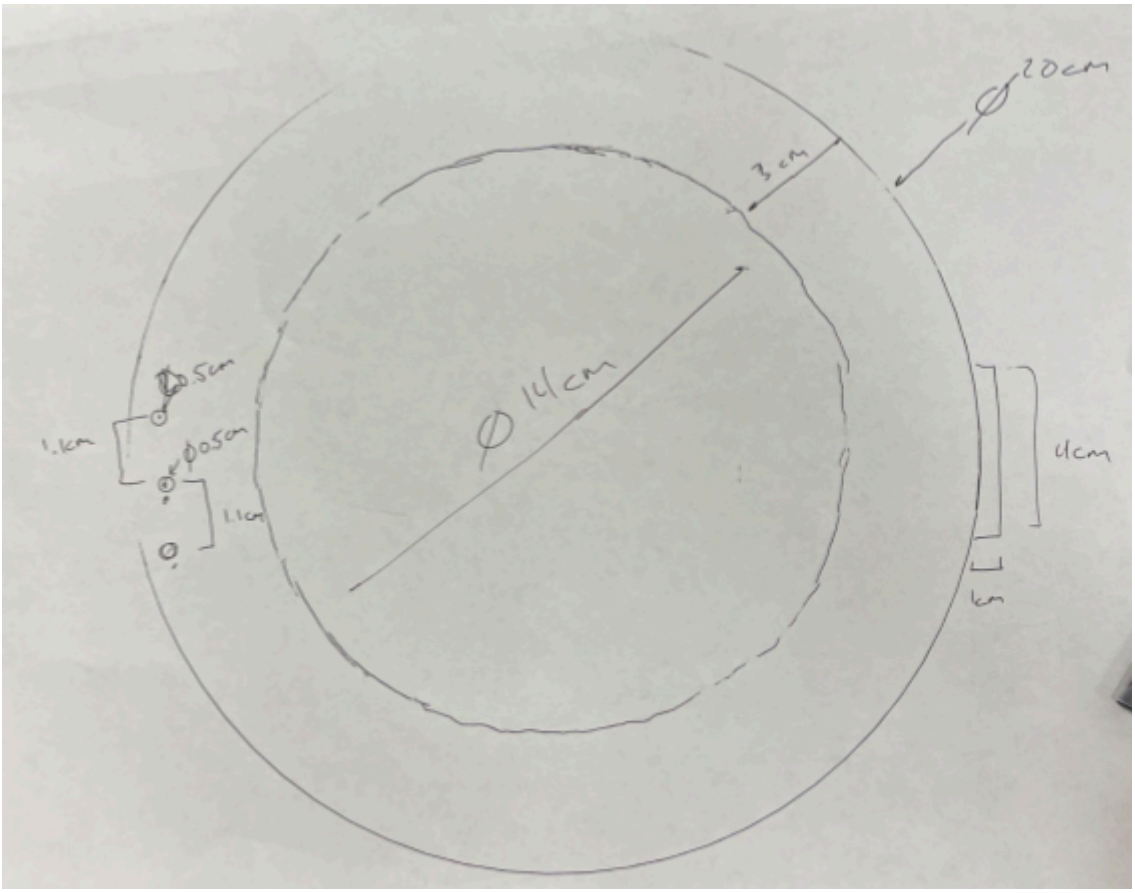

Figure 13. All measurements in cm. Shows a sketch of the upper lid that was used to finish the external encapsulation of the centrifuge. This part would go on to be laser cut with a hole in the middle for an acrylic dome to fit in.

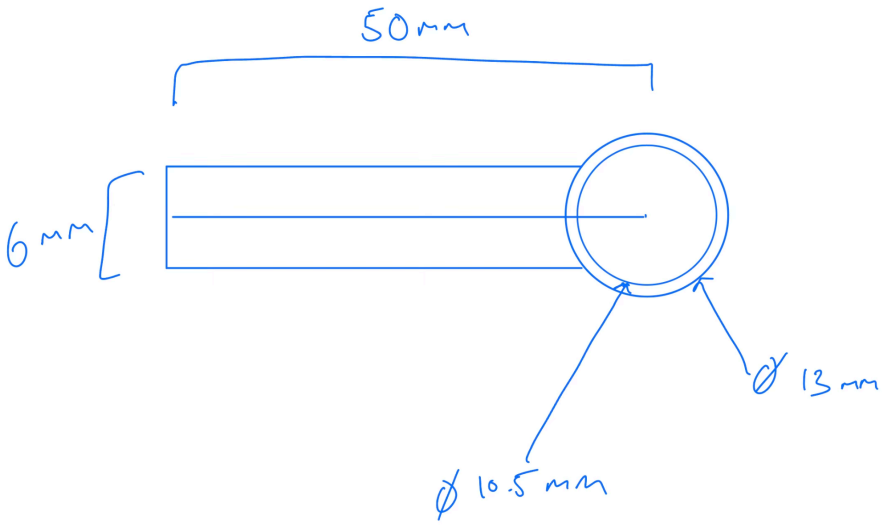

Figure 14. All measurements in mm. A sketch of the external tube holders used in V3. They were built to be long enough such that tubes did not scrape against the side of the centrifuge. The radius of the circle is exactly large enough to mate with a tube.

CAD

The Onshape files for V2 (including the laser cut box) can be found here.

The Onshape files for V3 can be found here.

V2 Designs

Below are screenshots of the internal and external CAD-rendered parts for the V2 cycle. These parts were iterated on for V3, which is indicated below.

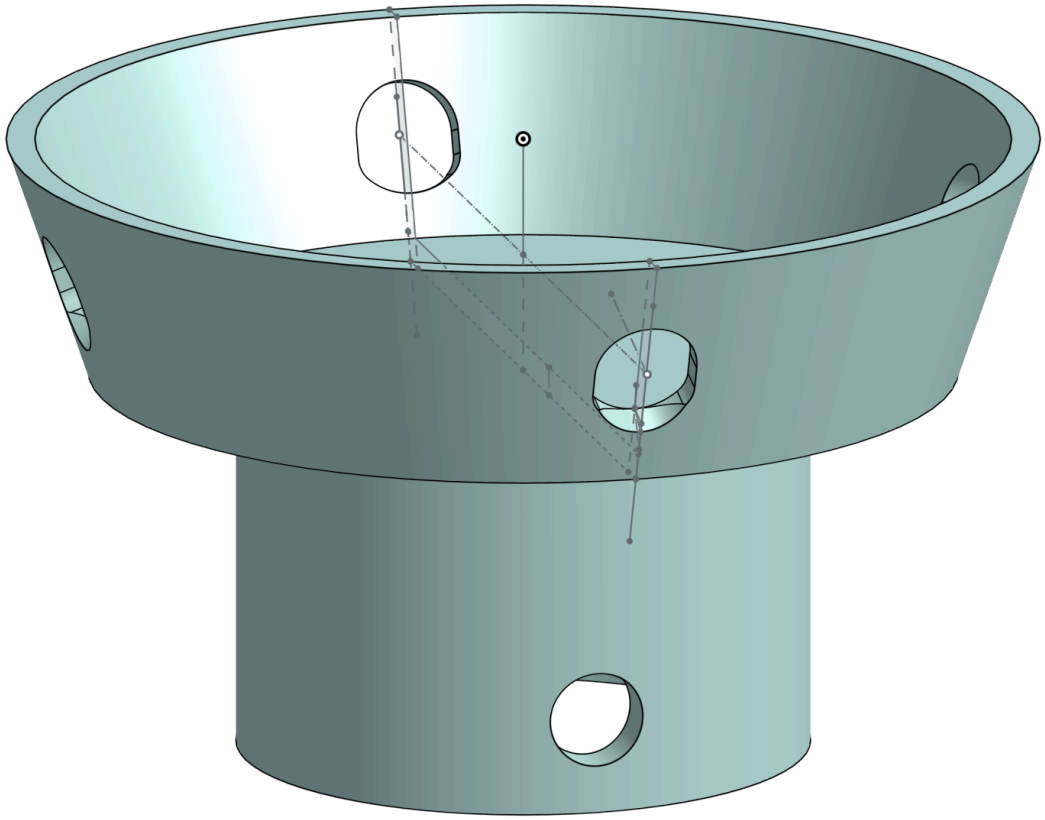

Figure 15: A screenshot of the tube holder + phototransistor complex designed in Onshape. The small hole at the bottom of the base allows light to reach the phototransistor and count rotations. The phototransistor would be attached to the motor inside, and the LED would be attached outside.

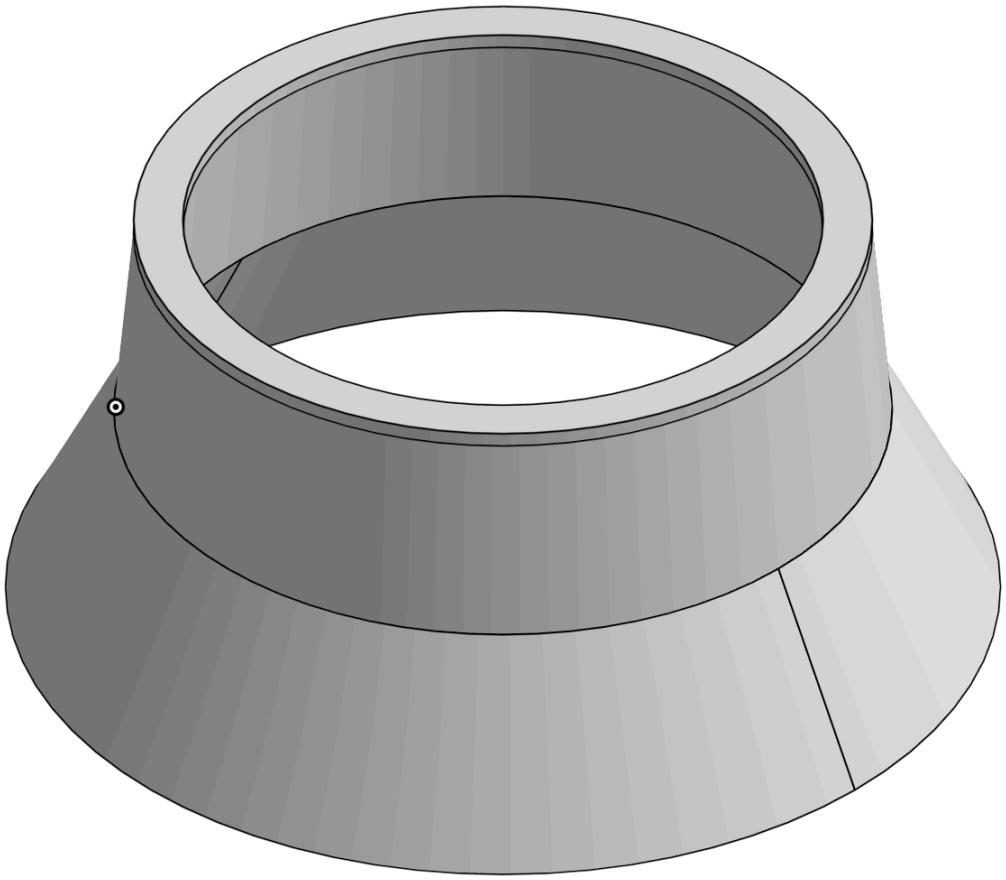

Figure 16: A screenshot of the entire external physical encapsulation, built in CAD. The bottom was made wider both for aesthetics and to accommodate the breadboards and circuits we planned on integrating with the mechanical subsystem.

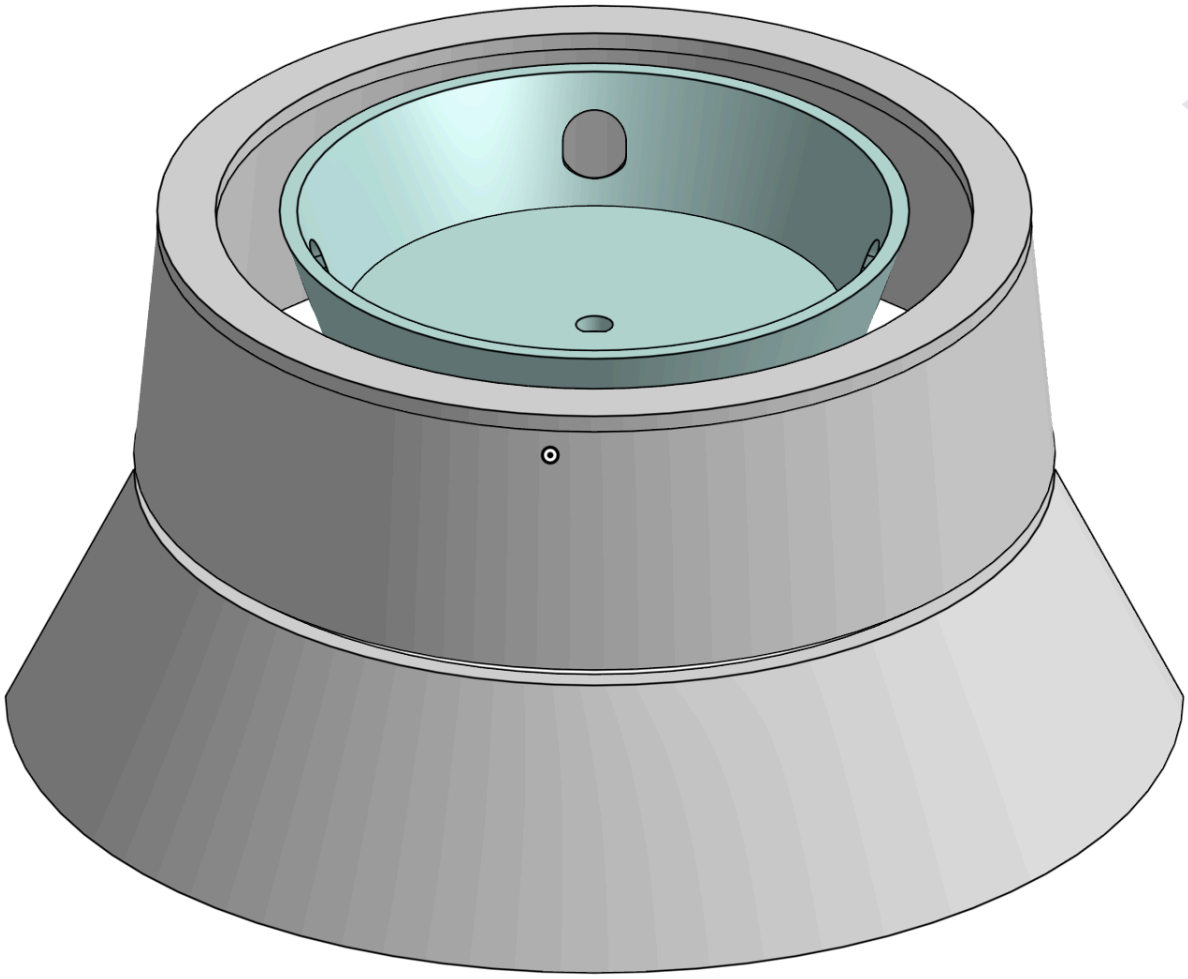

Figure 17: The entire V2 mechanical subsystem, assembled in CAD to test before sending to 3D printing. Extra room was left between the internal and external parts to account for the tubes spinning during centrifugation.

V3 Designs

For the V3 cycle, the general idea behind our mechanical design remained the same. However, 2 main iterations were made in addition to adding tube holders and an acrylic lid to improve the overall functionality of the centrifuge, as shown below with CAD renderings.

The first was a reduction of the radius of the internal tube holders. We hoped that by making this change, the tubes would be able to spin with no risk of touching the external encapsulation. Further, we hoped to increase the maximum speed of centrifugation by reducing radius and mass.

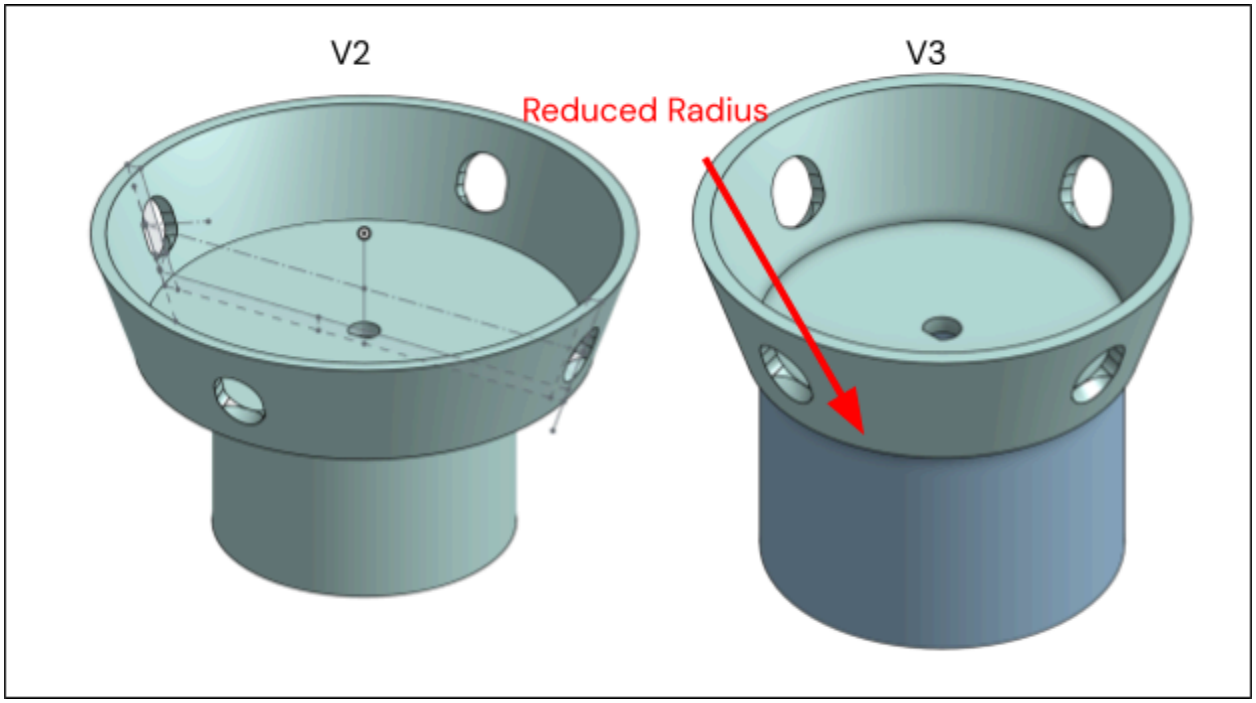

Figure 18. A screenshot showing the iteration of the tube holder + phototransistor complex for V3. The radius was reduced to allow faster centrifugation and more space between the internal and external mechanical parts.

The second was flipping the external encapsulation upside down such that the wider part was facing upwards. This change was made to allow the tubes to spin with more space between the external encapsulation, contributing to a safer overall build.

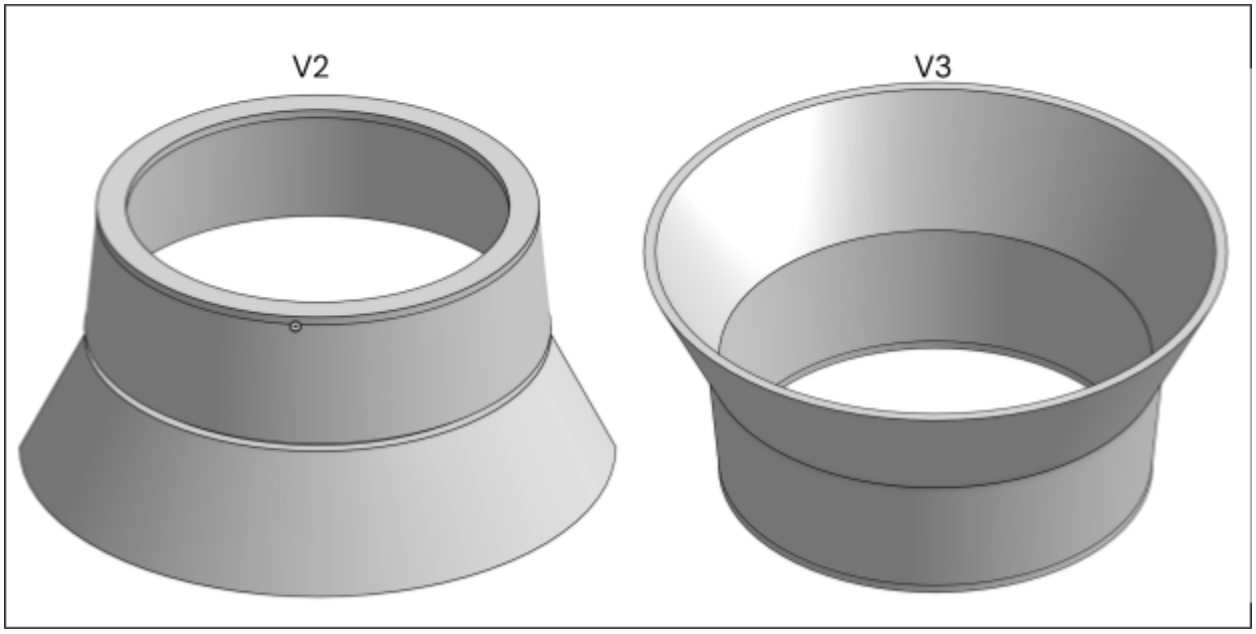

Figure 19. A screenshot showing the iteration of the external encapsulation. It was flipped upside down to allow for more space inside the centrifuge. This was done over printing a new part due to time constraints.

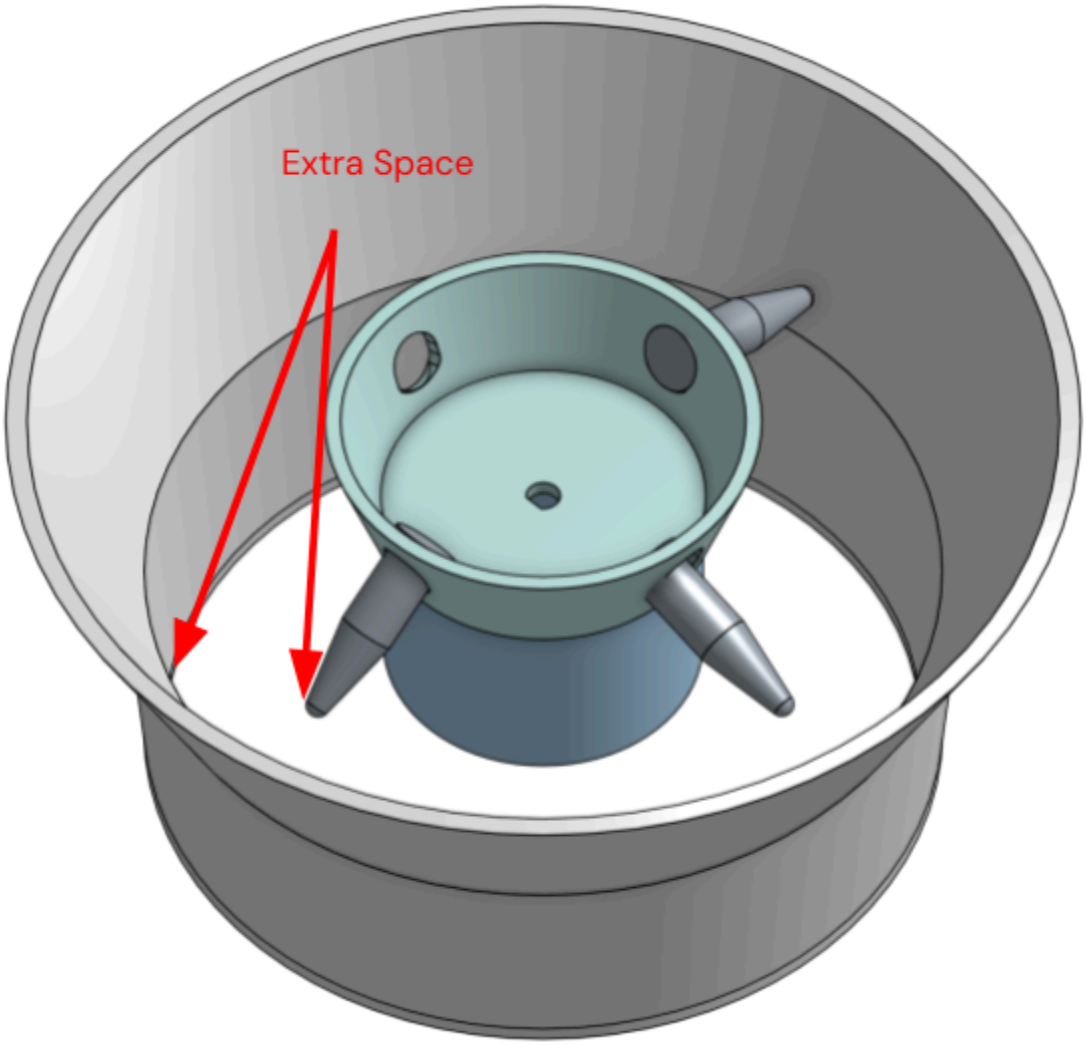

Figure 20. A screenshot of the combined 2 iterations made for V3. With a smaller radius on the inside and a larger one on the inside, the tubes could spin faster with no possibility of touching the external encapsulation.

Accompanying these iterations were also the new parts that we hoped to introduce for V3, namely the acrylic lid (which would accompany a dome that was separately ordered) and the tube holders.

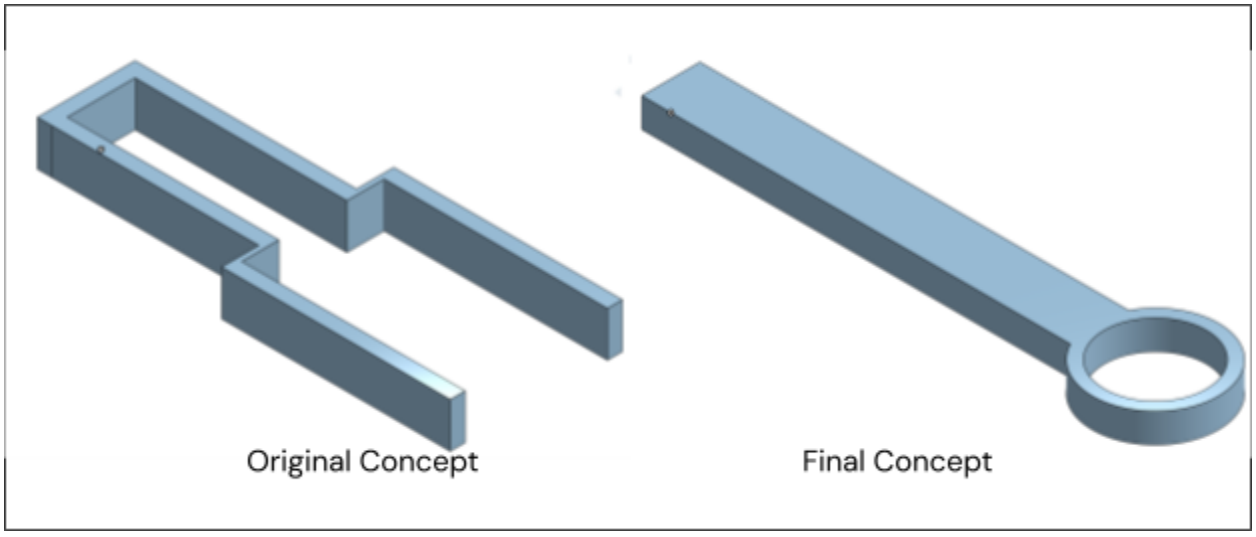

Figure 21. A screenshot of the external tube holders we hoped to implement in our V3 build. On the left is the first iteration of CAD sketching, and on the right is the actual part we ended up printing.

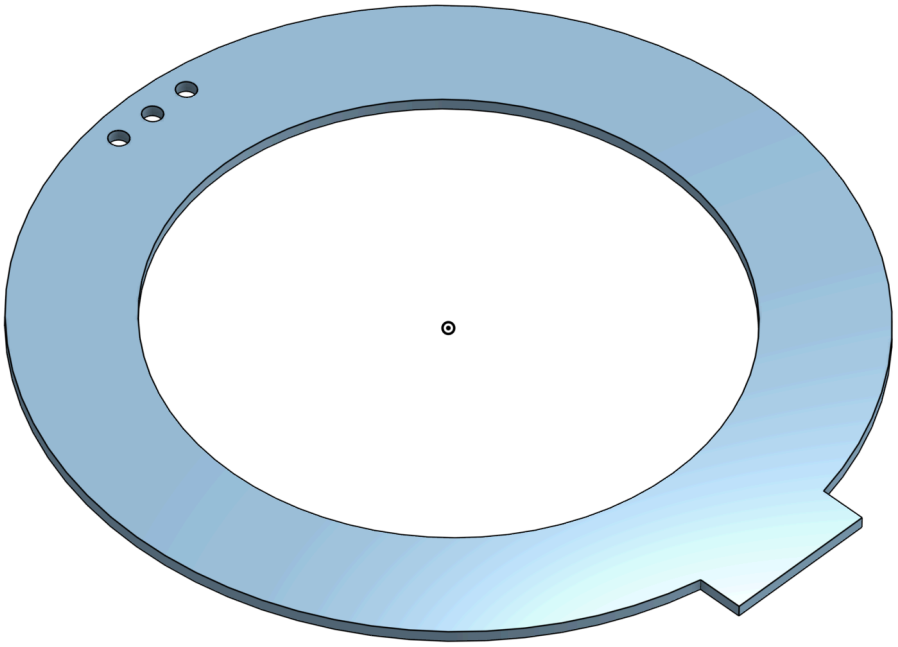

Figure 22. A screenshot of the upper lid we planned to attach to the outer encapsulation. Holes were built for a hinge mechanism, and a lip was added for easy user interaction.

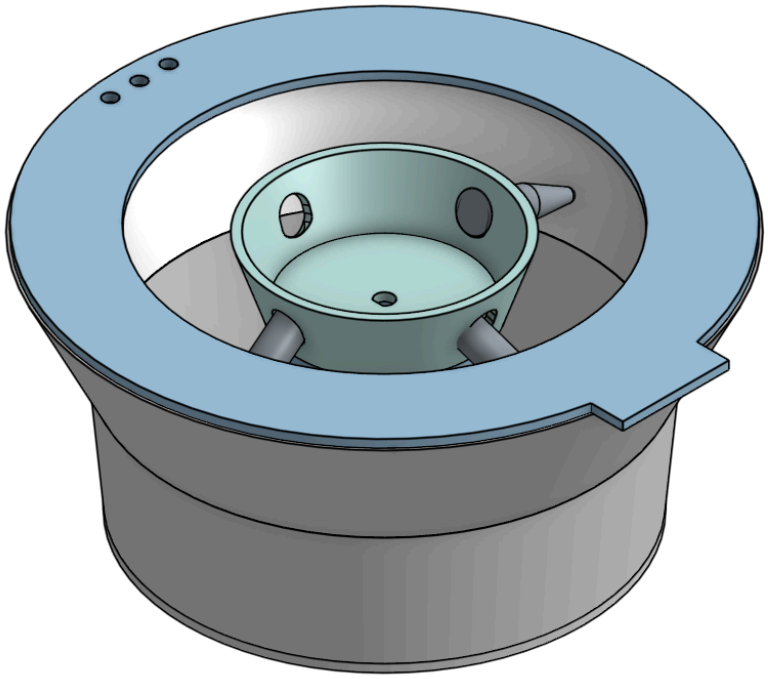

Figure 23. A screenshot of the assembly between the outer encapsulation, lid, and internal tube holders. A dome was separately ordered to fit in the upper hole.

Before 3D printing the upper lid, we also wanted to conduct a sanity check to make sure that the part would fit and satisfy all constraints. Working with a rough physical creation out of styrofoam before printing was a lesson learned after V2 to ensure accurate creation of parts.

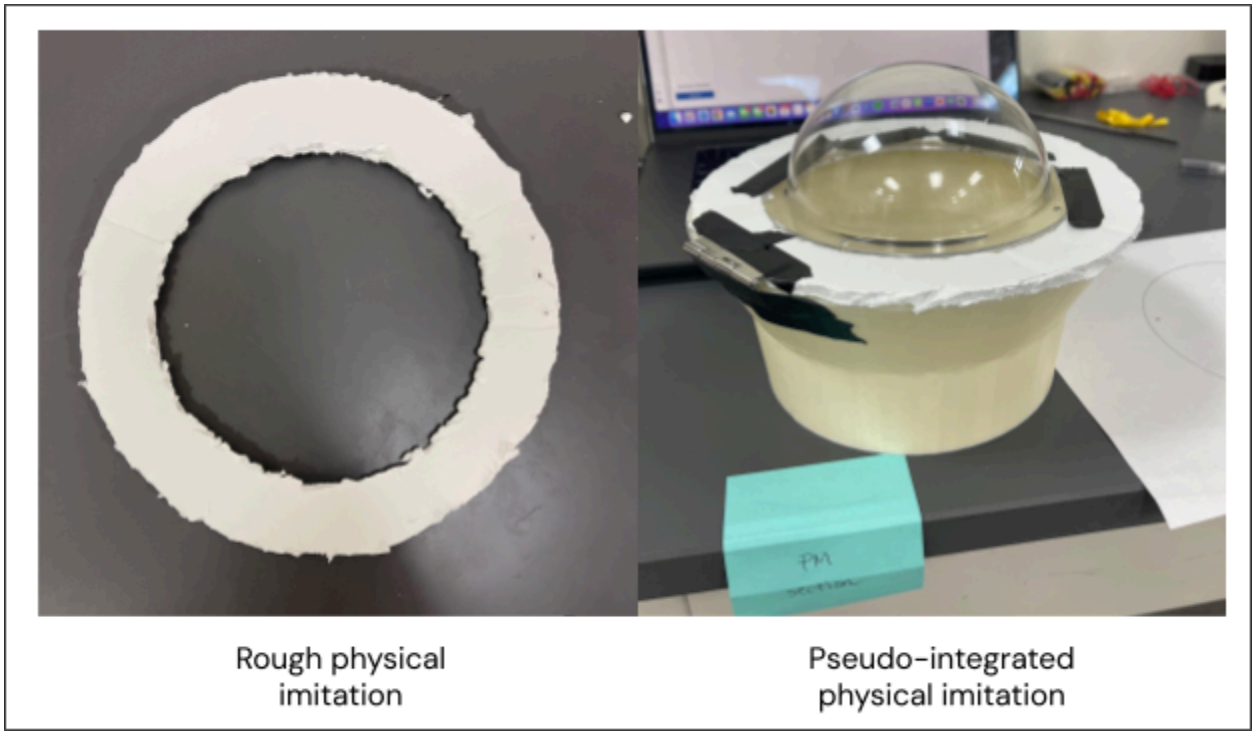

Figure 24. A rough upper lid constructed out of styrofoam found in the lab. This step was conducted to make sure that the lid design was physically feasible before sending it to 3D printing.

3D Printing

Following the sketches and CAD assemblies, we sent the parts in for 3D printing. However, because the external encapsulation was too large for the 3D printer itself, we had to split the parts into two to fall under the limit. It was at this point that the mechanical design process slowed, as the size of the pieces meant that our build was moved to the back of the queue as compared to the rest of the class. The other subparts of V2 took priority during this time. This also informed our decision to flip the encapsulation upside down for V3 rather than print new parts, as we likely would not have had the time to print a new part of similar sizing.

Once the print started completing, another realization we came to was the fact that the motor couldn't be on the ground, not only because it had to be connected to

wires, but also because it had to be stabilized in some way. As such, we used M6 components found in the lab to construct a stable base.

Figure 25. The internal physical complex resting on a base designed from M6 components. The motor is housed inside the shown unit.

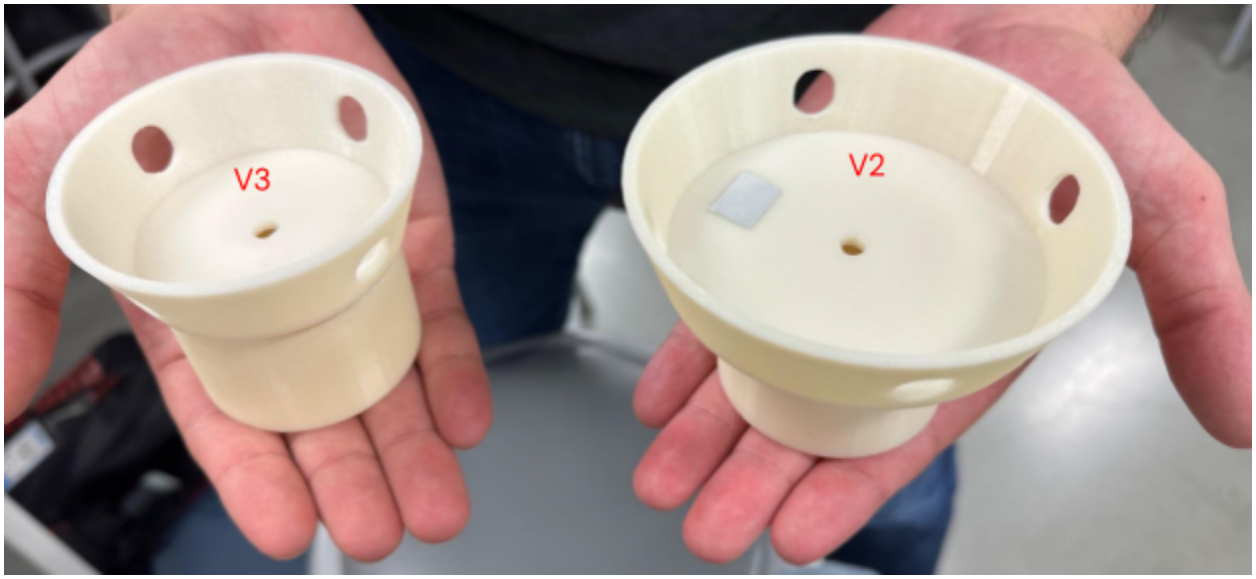

Figure 26. A side by side comparison of the printed V3 tube holders (left) versus the V2 holders (right).

The second conclusion, which went hand-in-hand with our initial V3 iterations, was that the base was simply not wide enough to encapsulate all our circuitry. As such, we constructed a laser-cut box for the contraption to rest on.

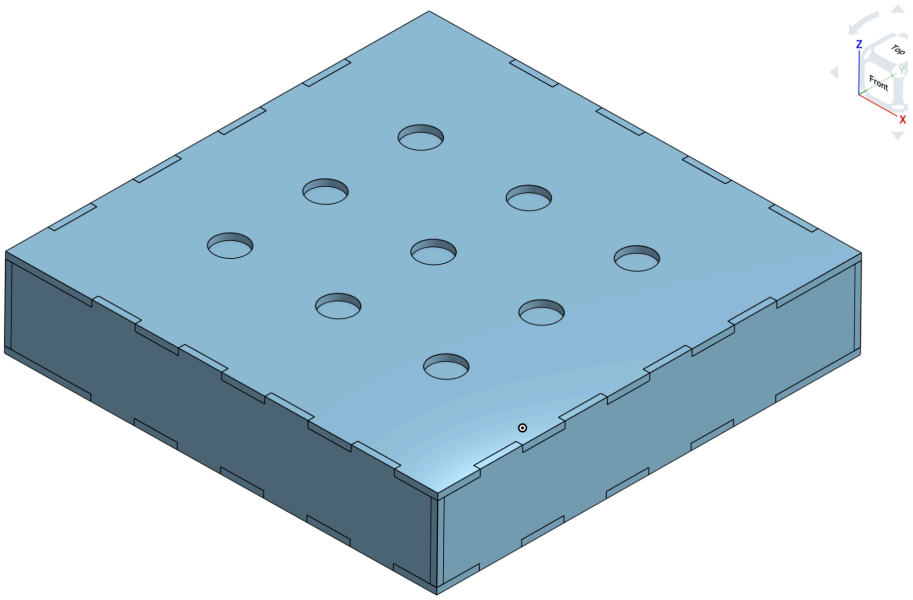

Figure 27. A screenshot of the V2 box the entire physical subunit would rest on. The circuitry would fit within this box, and wires would thread through the top to connect to the motor above.

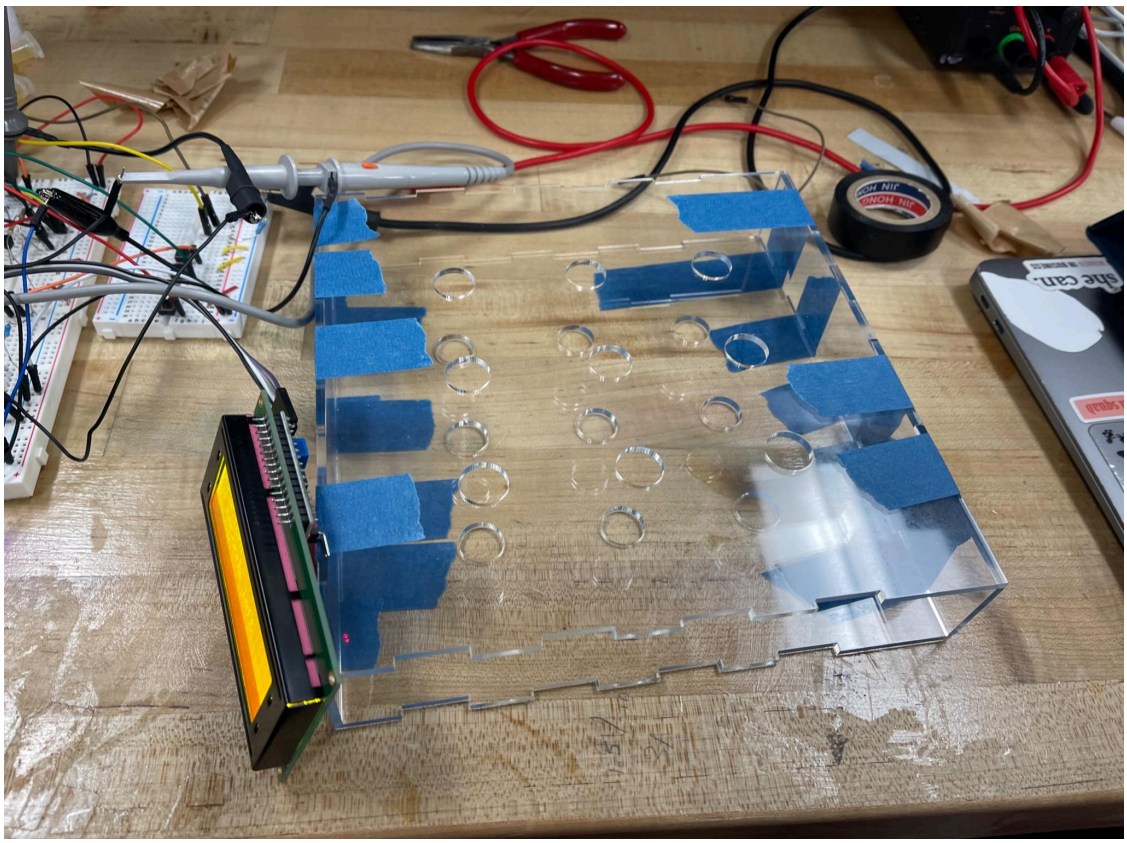

Figure 28: The V2 circuit encapsulation box after laser printing. It will be taped together until the build is finalized.

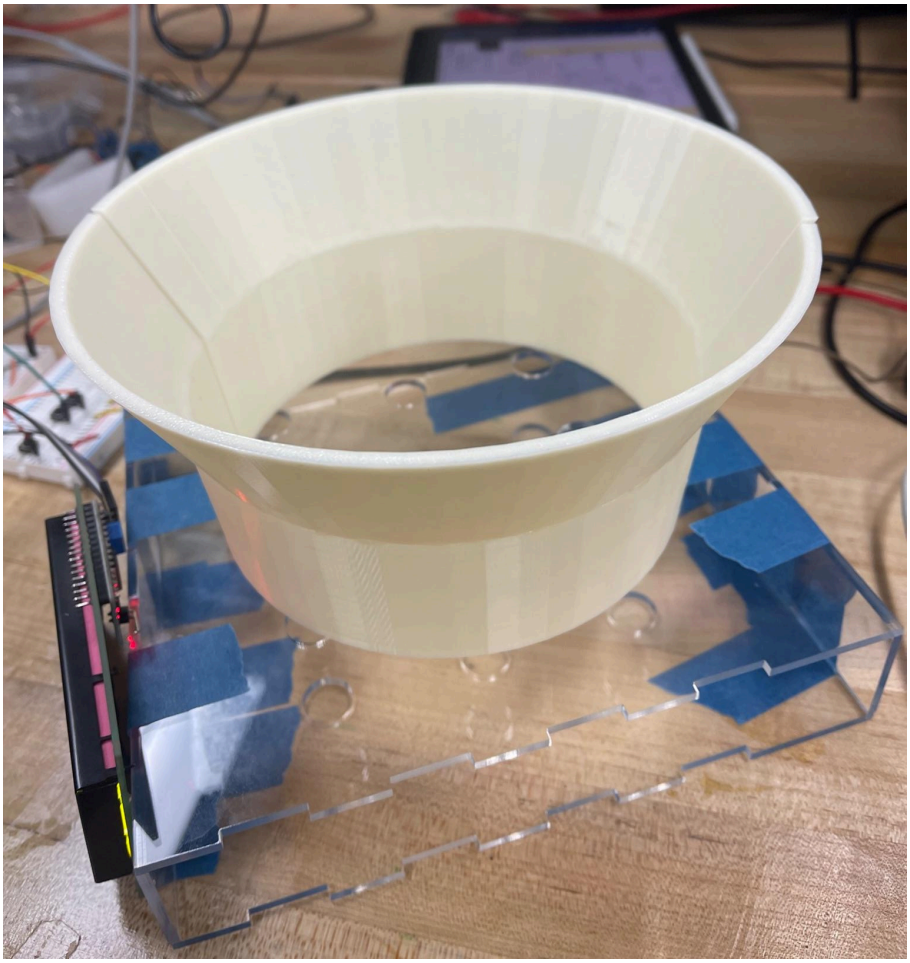

Figure 29. The laser cut box, above which lies the outer encapsulation of the physical subunit. It is turned upside down to accommodate the large radius of the spinning tubes.

For the V3 build, we realized that we wanted the LCD to fit snugly within the laser cut box rather than being outside. This fits with our larger goal of making the entire centrifuge one unified system rather than a collection of semi-integrated parts. Resultantly, we made the height of the box bigger for V3, as shown below.

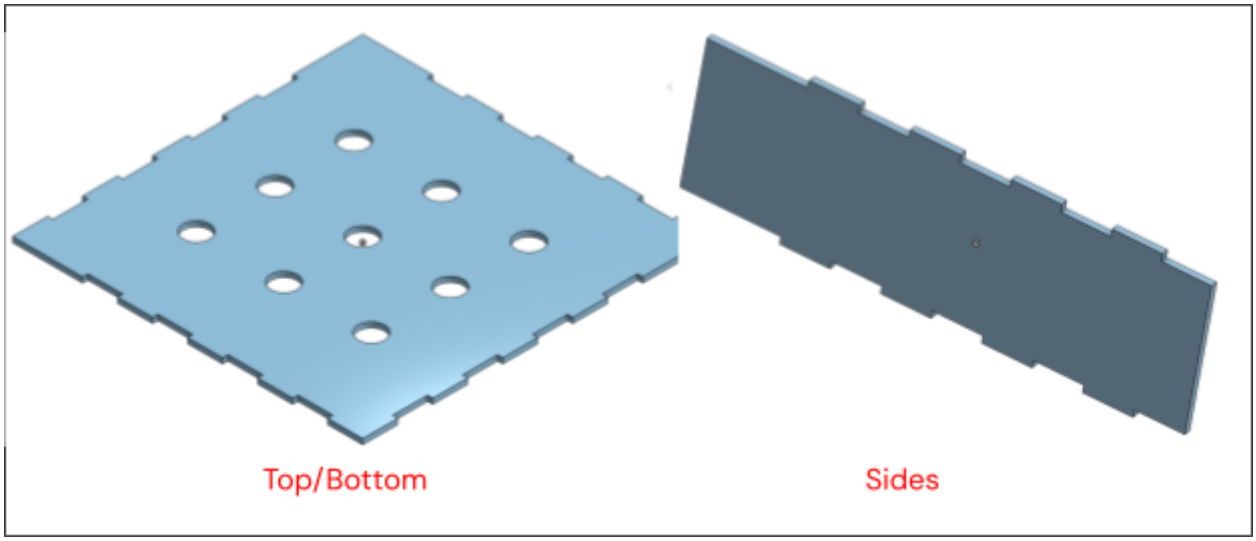

Figure 30. A CAD of the box that was used for V3. The side component height was changed to 60 mm to accommodate the LCD screen.

The final change that we made to the box for V3 was adding in button slots and etched labels. Once again, this falls under our broader target of full integrated functionality. With built in button slots, the circuit would be entirely contained and hidden from the user, adding to the front-end experience.

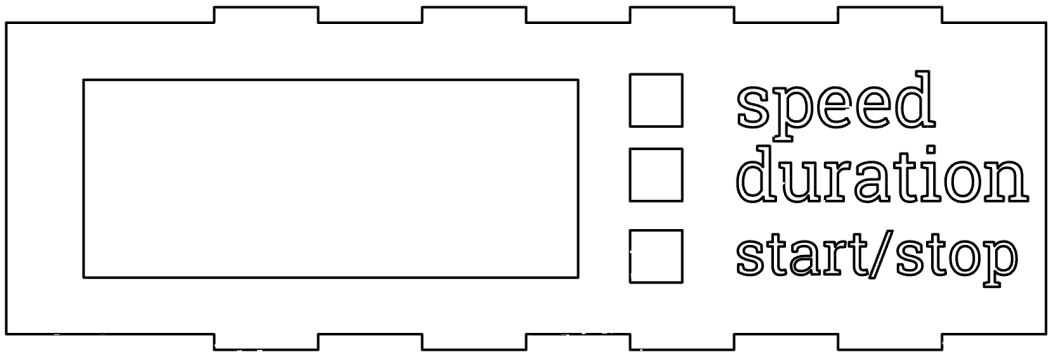

Figure 31. A drawing completed in CAD of the front side of the box for V3. The words were ultimately etched into the acrylic, and holes for buttons and the LCD were also included.

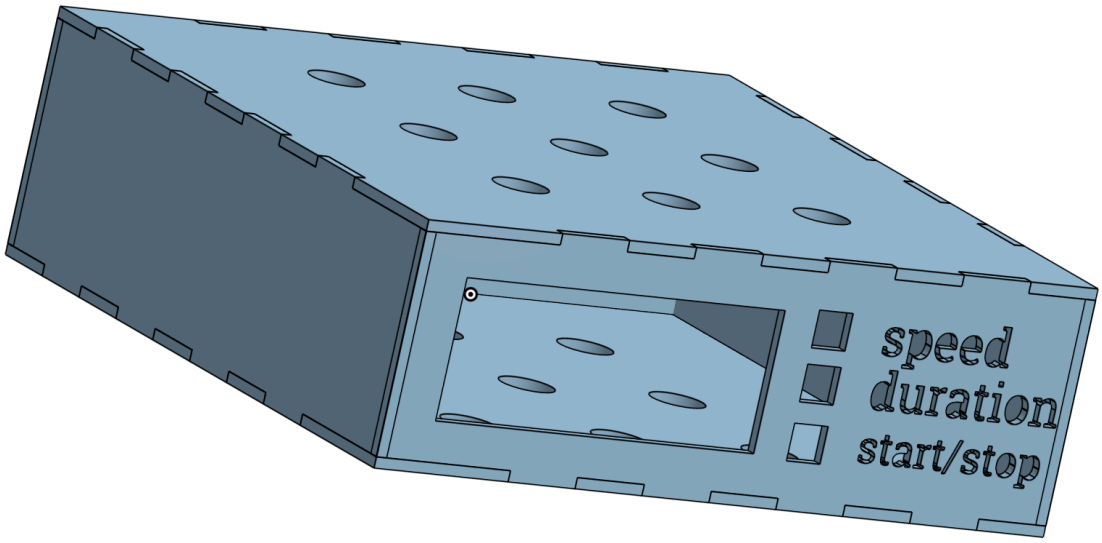

Figure 32. A CAD of the assembly that was ultimately used for V3. The built in spots for the LCD, as well as button inserts, allowed for more complete integration as compared to V2.

Following the completion of all mechanical parts, the subsystem was ready to be integrated with circuit and code.

Integration

This section of our design cycle proved to be by far the most challenging. Though each team member felt confident about the state of their individual builds, integrating the systems was a challenge in and of itself that remained unresolved until shortly before the V3 deadline. Before introducing the mechanical subsystem, we believed the most appropriate first step was to get the code integrated with the circuit to ensure that correct signals were being sent between the actuators, sensors, and control systems. No motor would spin without a correctly functioning circuit and code.

V2 Integration

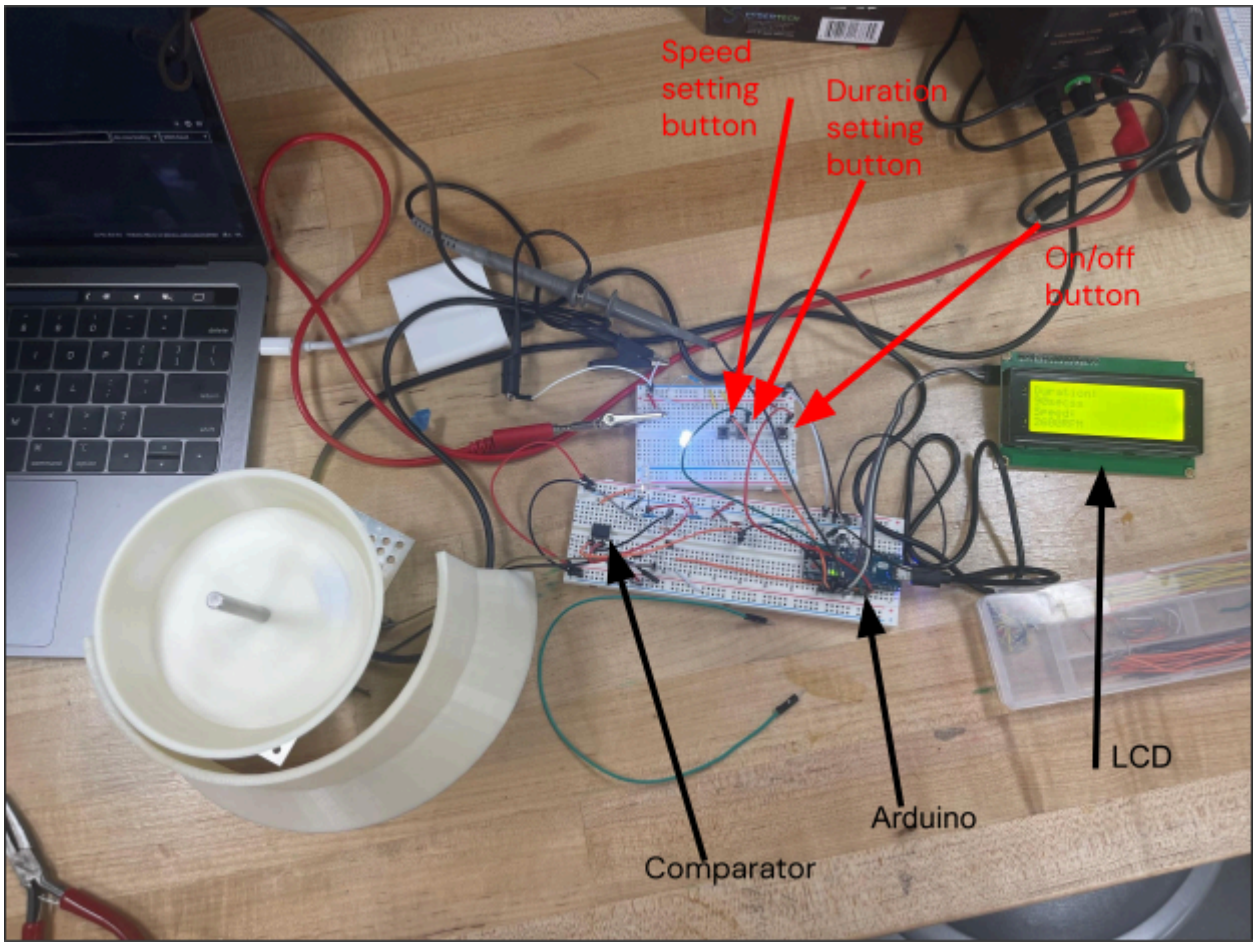

Figure 33: A first pass at the circuit integration with the buttons (red) and LCD (black) to control different inputs and readouts of the motor.

Over the course of integration, a litany of problems arose that required constant mitigation. Below are some of the challenges that proved to be the most difficult and that we made special note to check for in any future builds:

- Connecting grounds to each other between breadboards
    - This includes connecting Arduino ground to other grounds
- Making sure code pins match circuit pins
- Checking the functionality of certain Arduino pins, i.e. is it an interrupt pin?

- Making sure code being pushed to an Arduino doesn't crash it out with an error.
    - E.g. divide by zero error will not cause syntax problems in the IDE, but will short out an Arduino.

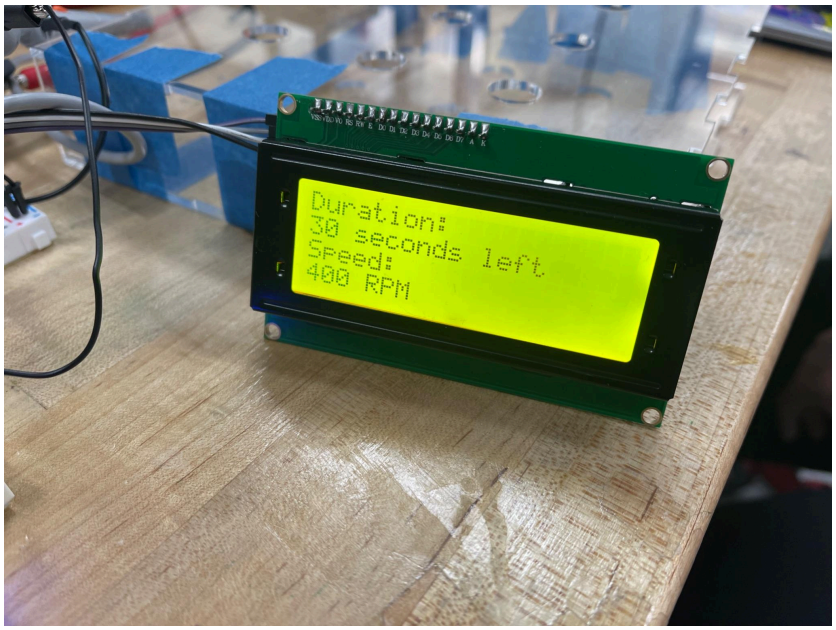

Figure 34: The LCD display and the messages it displays while the motor is running. Seconds count down until 0, at which point the motor stops. The speed is supposed to be displayed live through the phototransistor complex, but never functioned appropriately throughout the build.

After multiple rounds of trouble shooting, we began to integrate the mechanical subsystem in with the circuit + code complex.

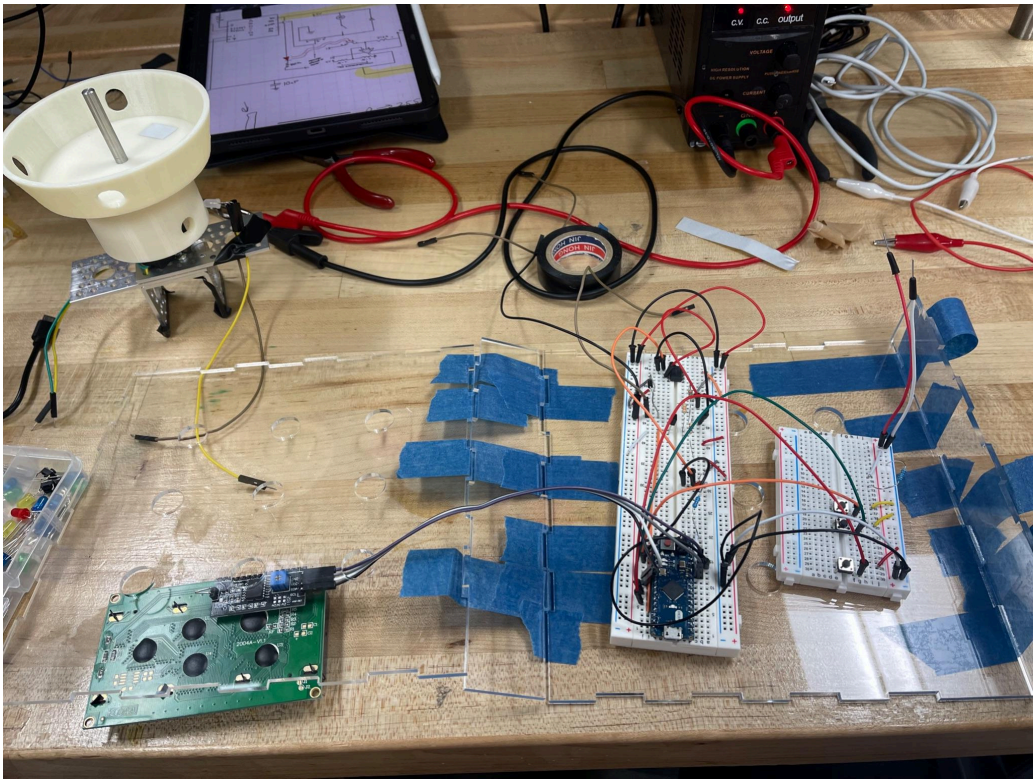

Figure 35. The circuit begins to be integrated into the mechanical subsystem.

Following the full integration of the V2 build, we noticed that the buttons to control speed, duration, and on/off were extremely hard to reach. We proceeded to move them outside the box with the LED for the user oriented part of the centrifugation.

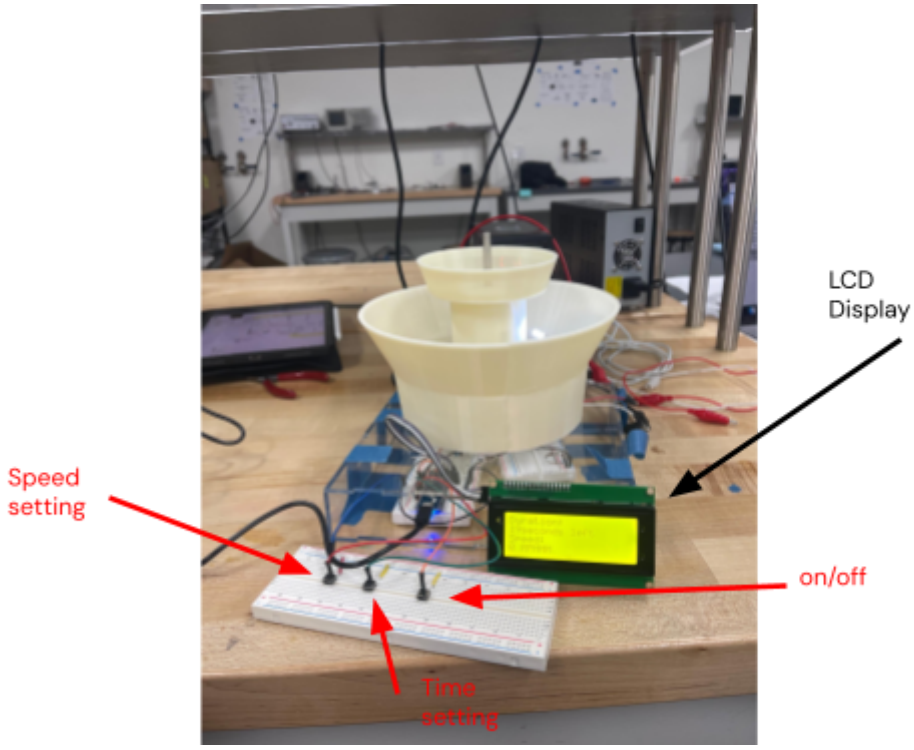

Figure 36. The fully integrated V2 system, with the buttons moved outside for ease of pressing.

At this point in the design cycle, our progress started to plateau significantly. We eventually got nearly all systems functioning appropriately, with one minor malfunction and one major malfunction. Neither issue was resolved by the end of V2, but both were resolved by V3.

- Minor malfunction: Speed setting interrupt button press was not printing consistent outputs on the LCD. Despite checking our code in TinkerCAD and confirming that we had a functioning interrupt scheme to keep track of desired speed, different presses would sometimes add 1000-2000 RPM to the speed instead of the predicted 100 RPM. We hypothesized either that the pin it was attached to wasn't functioning properly, or that it was sending enough voltage through to trigger multiple different inputs at a time.
- Major malfunction: failure of the phototransistor + LED system to measure RPM. This is our primary form of measuring speed, and without it, we had no real ability to accurately test a range of specifications. We checked the

circuit and inputs/outputs via oscilloscope with no glaring errors, leading us to believe the problem may be in the counting part of the code.

V3 Integration & Iteration

Chronologically speaking, we moved directly from V2 integration to V3 integration, **focusing first on the problems listed above that we wanted to remedy before adding in new features.**

Having left V2 with a dysfunctional tachometer, we decided to restart our troubleshooting process by creating a extremely simplified version of our code and test the tachometer subsystem in even smaller software and circuit subsystems: phototransistor output, comparator output, arduino counting, arduino time tracking, and finally accurate speed reading in RPM. We attacked them sequentially with an oscilloscope probe in both hands and our eyes glued to the serial monitor. Bianca was an immense help during this process. Please give her a raise.

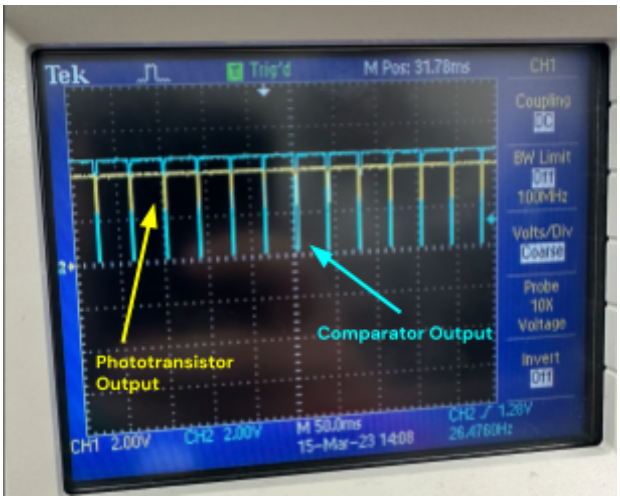

Figure 37. shows an oscilloscope reading of the phototransistor output from the circuit and mechanical integration and the comparator output from the circuit subsystem.

We were quickly able to ascertain the functionality of the circuit component of our tachometer via oscilloscope. We did end up lowering our Vref from ~3.63V to 2.8V just to solidify it between the high and low phototransistor output. Knowing

that the Arduino was receiving clear circuit signals, we transitioned to check the counting with an isolated tachometer code excerpt.

```
volatile unsigned long count = 0;

const int ROTATION = 7;

void setup() {
  pinMode(ROTATION, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ROTATION), accelerometer, RISING);
  Serial.begin(9600);
}

void loop() {
}

// RPM counter ISR
void accelerometer() {
    count += 1;
    Serial.println(count);
}
```

Figure 38. Shows a stripped version of our V3 code used to test the tachometer subsystem. The code includes the arduino setup to received the digital signal from the comparator on pin 7 in a ISR schematic with the ISR housing a counting mechanism.

Our counting functionality worked beautifully in isolation but not in the integrated code. We ended up finding a huge bug in the speed measuring portion of the code. We were only calculating measured speed every 5 seconds with the intention of allowing counts to build up before dividing by elapsed time. Yet, because the arduino loop cycles on an order of microseconds, 5 seconds lasted 15 - 20 loop cycles, resetting the count far too frequently. This meant we weren't always registering counts. Once identified, we were able to apply a state flip system to only measure the speed once as shown in figure 39.

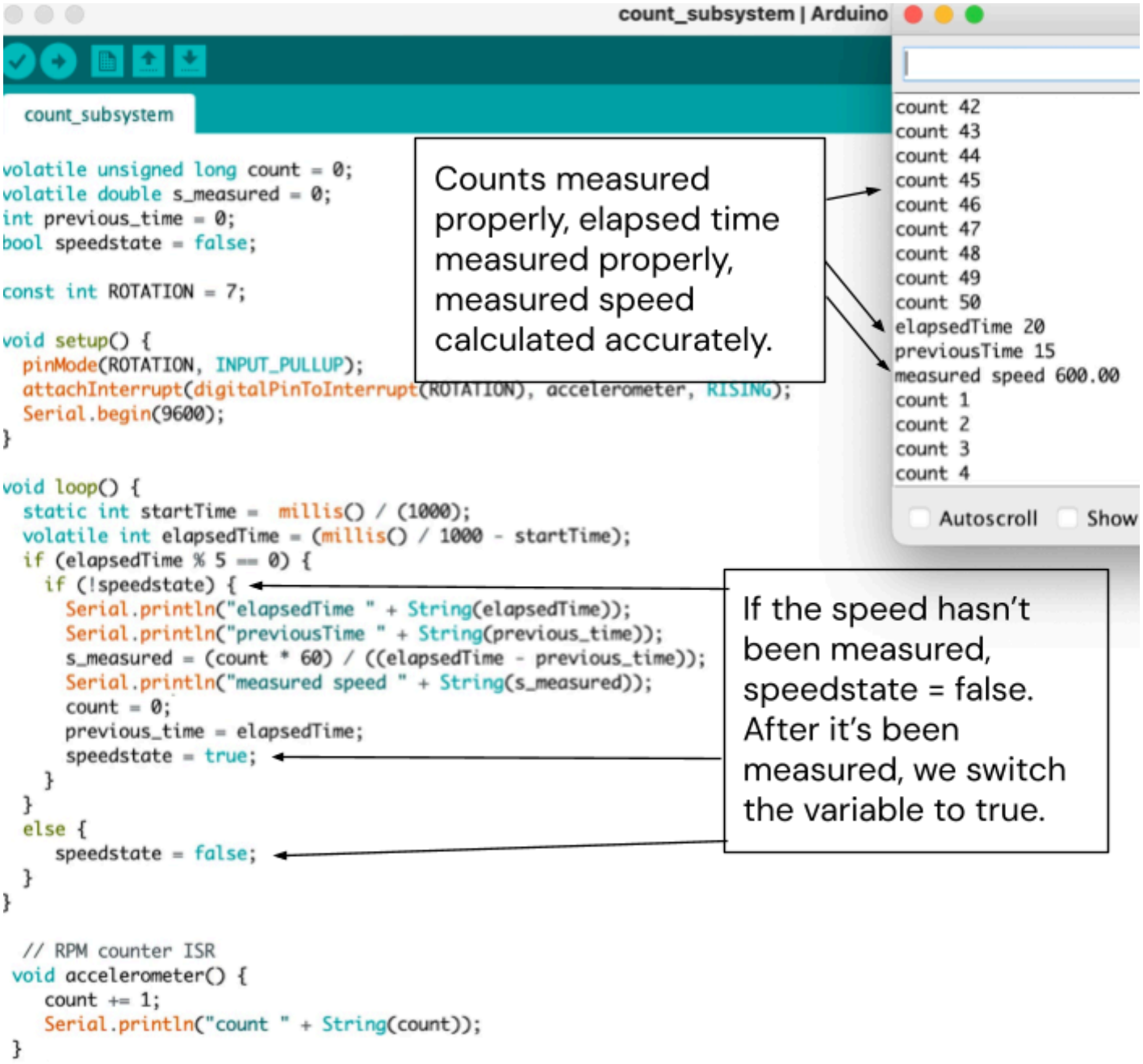

Figure 39. shows our isolated tachometer code with the function generator simulating the comparator output with a 10 Hz square wave. The serial monitor in the upper right corner shows the expected counts read, the expected elapsed time, and the expected measured speed.

Figure 40. verifies our Arduino counting and time tracking subsystems and shows that, in isolation, the tachometer software calculates the expected measured speed. We used the function generator to simulate the comparator output signal

from the phototransistor. With a 10 Hz square wave, we expect to see 600 RPM. We then finally moved to our validation of the tachometer with experimental speed. Replacing the function generator with the comparator output, we create Figure ___.

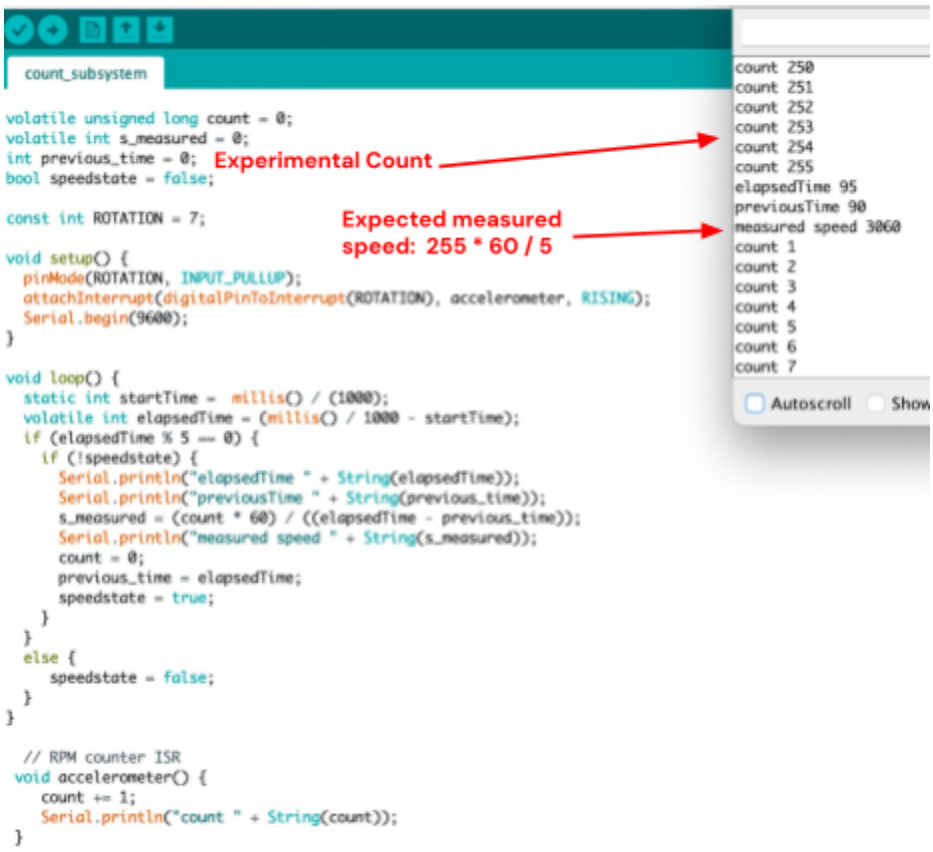

Figure 40. shows the isolated tachometer subsystem successfully displaying the measured speed.

We were then able to incorporate the isolated code into the greater software. Our process of troubleshooting the tachometer subsystem is a microcosm for what we've learned in the class. The ability to subsystems into subsystems and test each individual step was a new form of thinking we learned and practiced during V3. With the ability to accurately measure speed we could now complete a slew of new requirements namely 2.1.3. We also finally implemented and fine tuned our PID control system in the software which we previously left commented out in V2.

The bottom-up troubleshooting technique vastly improved the design and build process in V3 across subsystems but had particular success in software management

With the code malfunctions fixed and the circuit optimized , we moved into physical integration. We budgeted more time for this than we did in V2 because we recognized that it took up most of our time and revealed errors we didn't previously see.

Mechanical integration began with substituting out the larger inner tube holder complex for the smaller one so that tubes could spin at a higher RPM without scraping the outside encapsulation.

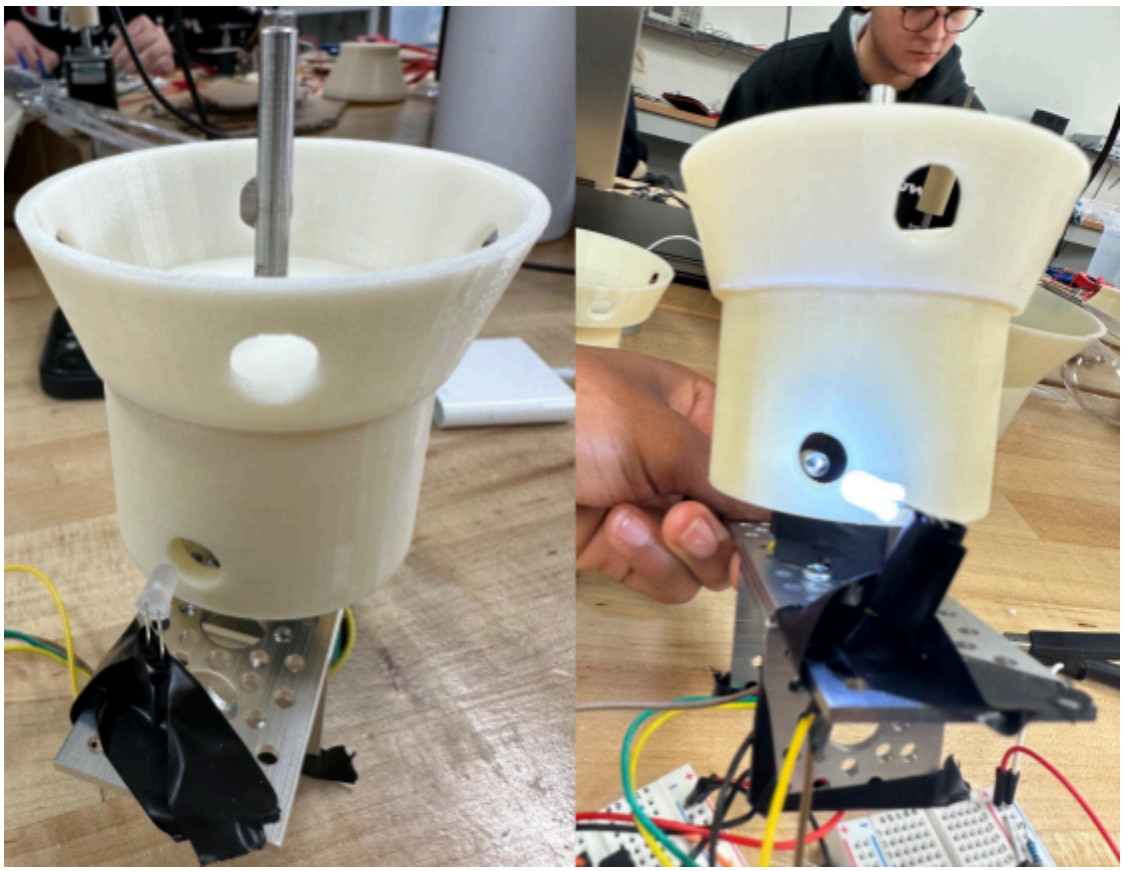

Figure 41. V3 integration began by installing the small tube holder to the motor and checking that the phototransistor complex still worked appropriately.

The next step involved attaching the laser-cut lid to the ordered dome and hinges so that it could open and close freely (see V3 ordering sheet for more details). One shortcoming we ran into was a mismeasurement of the three holes needed to screw in the hinge. Resultantly, only one hole was screwed in; the rest remained open.

Figure 42. Our next step in V3 integration was attaching the laser-cut lid to the preordered dome. On the left is a front facing picture of the system and on the right is a top-down picture that highlights the hinge and motor inside.

From a mechanical perspective, one of the biggest functionality hang-ups with V2 was the integration of the three buttons and the LCD screen. The buttons, which after code tinkering began working smoothly, were too small to comfortably press. The LCD screen also had to lie on the table, which was inconvenient to look at. Thankfully, the new box designed for V3 took both these improvements into account.

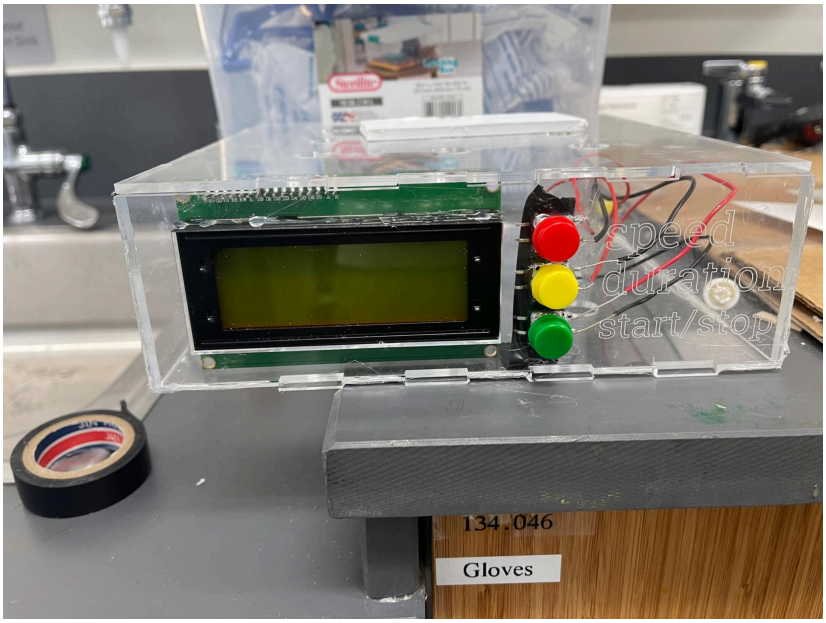

Figure 43. The next big step in V3 integration, where the buttons and LCD screen were placed into the front face slots.

Note that larger buttons were picked out of the lab's supply for ease of user pressing over the smaller buttons.

Another small note made was that the laser-cut box didn't perfectly fit together as a result of edges that were not teethed. This was a minor oversight and the box still functioned perfectly after being hot glued.

One of the integration issues that arose on the fly was the thickness of the acrylic front face as it related to button presses. The acrylic was thick to the point that the button presses were nearly impossible without putting extra force on the button from behind. To remedy this, a drop of glue was placed on the tip of the button receptor. The button was then placed on top and allowed to lock in with a bit more space, as shown in the image below.

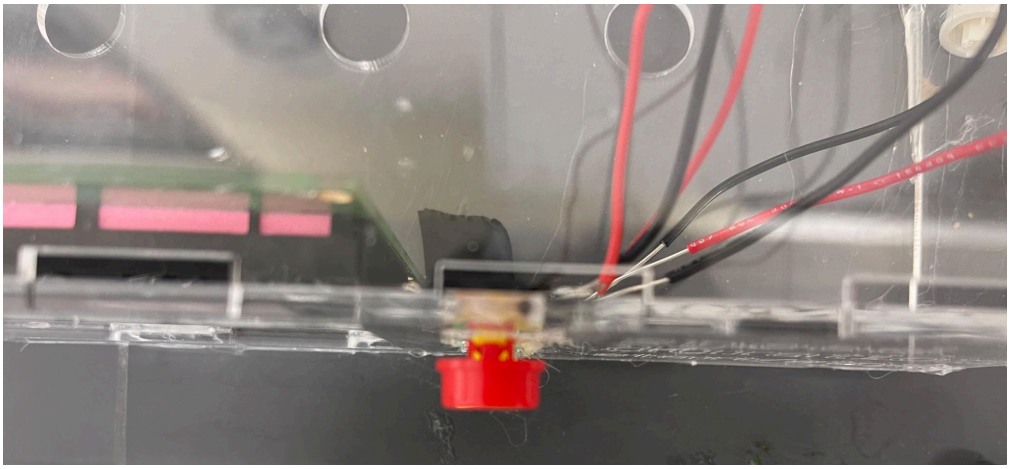

Figure 44. A picture of the buttons integrated into the system. They were not clicked all the way into the protruding button receptor unit; rather, they were glued right at the tip of it to ensure room for full button presses.

Post-system integration with the box, including the LCD and buttons, four tube holders were glued onto the external encapsulation structure. This step was relatively straightforward and took little to no troubleshooting.

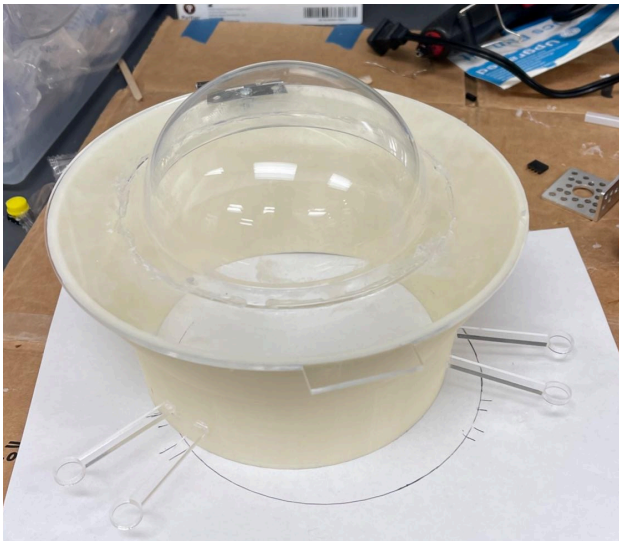

Figure 45. A picture of the tube holders attached to the external encapsulation. There were four tube holders in total that were placed to allow easy access from the front.

Finally, the breadboards were placed inside the box and wires were threaded through the holes to allow the motor to rest on top. One important detail to note is

that because the motor was built on an M6 scaffold, the same pieces had to be added below the external encapsulation to ensure that everything was operating at equal height.

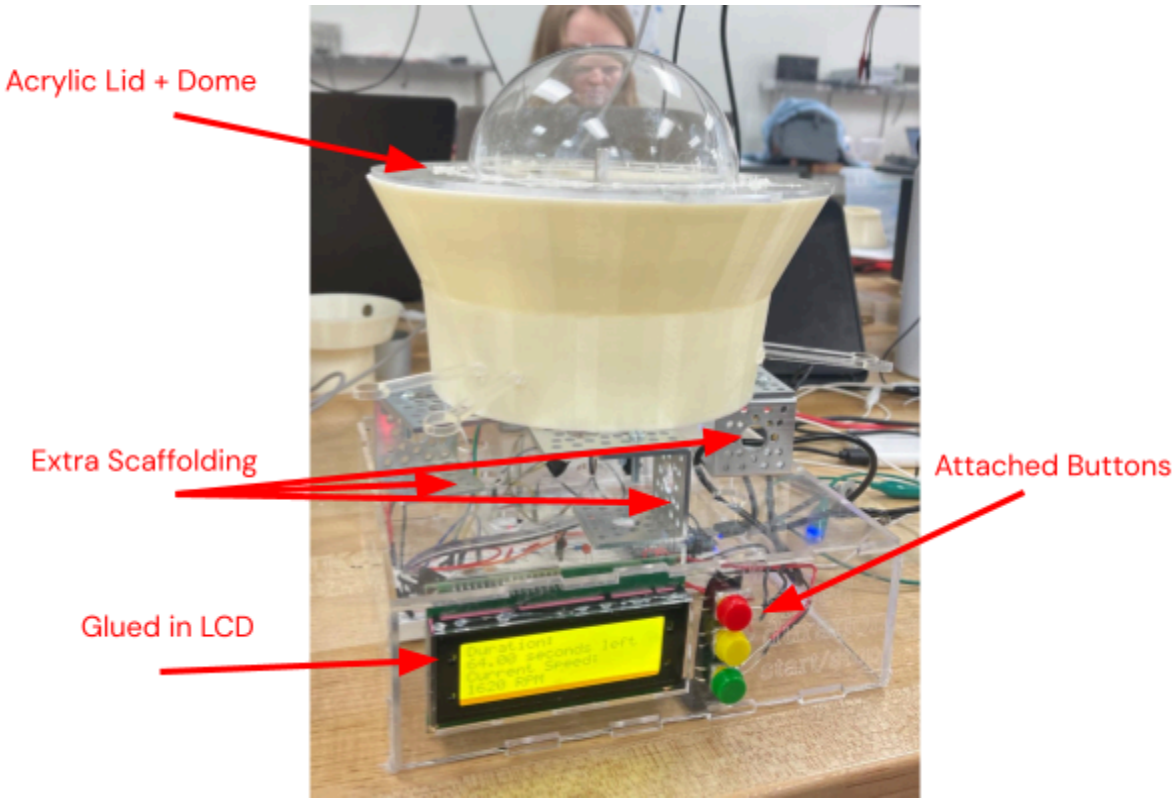

Figure 46. A picture of the fully integrated system, labeled with intermediate steps.

Originally, we had hoped to glue everything down such that the system was fully attached. Because, however, we needed access to the circuit to do oscilloscope testing, we never ended up attaching the encapsulation fully to the box.

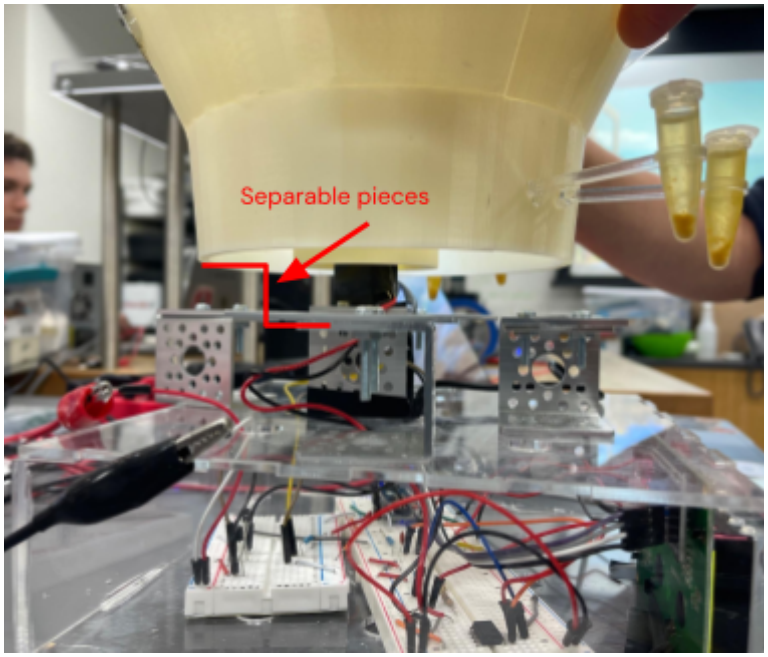

Figure 47. A picture of the physical encapsulation being separable from the rest of the integration. Though not ideal for the overall system integration, this was necessary for testing and assessing purposes.

Soldering: A V3 Add-On

Another change we implemented from V2 to V3 is the use of soldering to clean up circuitry, as mentioned in the aesthetics section above. In addition to making the breadboards look better, we also had some issues with wiring slipping off, especially with both the phototransistor and the larger buttons we used.

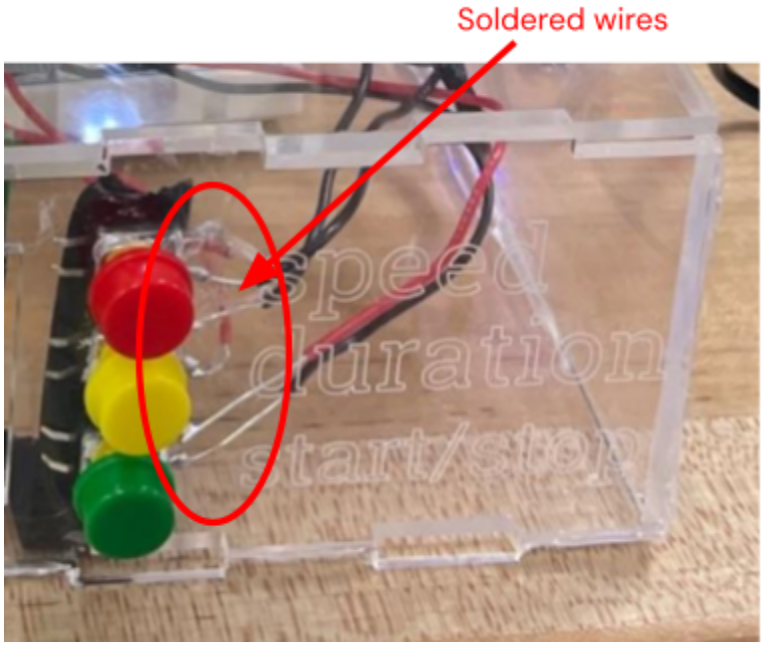

Figure 48. A picture of the soldered wires used to connect the buttons to the circuit. Soldering them eliminated any risk of wires coming disconnected and allowed us to take the buttons off the breadboard completely.

Final Design Touches: Painting & Wall Plugs

Once most of the heavy lifting was done with regards to integration, we moved onto adding final design touches to the centrifuge ahead of its demo day. In keeping with our V1 goal of designing a “celestial centrifuge,” our V3 build was spray painted all black and speckled with stars of various colors to create a galaxy feel.

Figure 49. The V3 external encapsulation during (left) and after (right) its spraypainting. Blue tape was used to cover the domed region so that it remained clear even after being painted.

Though this was technically not a formal part of V3, as it occurred after demo day, we used a 9V wall plug and its accompanying adapter to make our centrifuge operable without a DC bench power supply. With this change, the centrifuge became capable of operating everywhere that a plug and computer (for the Arduino) is available.

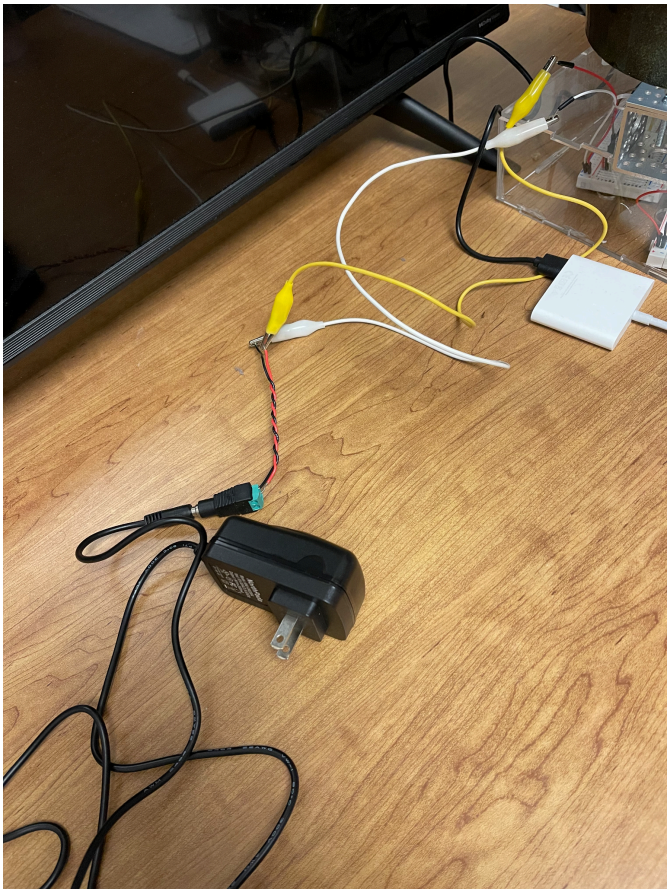

Figure 50. The 9V wall plug used to make the centrifuge function independently from a DC bench voltage supply. A barrel adaptor connected wiring to the plug.

Testing

Trace Matrix Summary

For our V3 design, we successfully completed both **requirements 1.1 and 1.2** once again. Our rotor base was still capable of holding 4 1.5 microcentrifuge samples, the only difference from V2 is that we made the rotor base slightly smaller..

Category 2 requirements included all centrifugation parameters and were completed to varying degrees of success. While we passed all of the category 2 requirements there were some that our team felt could be improved. First for the

requirements we passed that require no need for improvement in another iteration, which includes **requirement 2.1.2, 2.1.4,** and **2.1.5**. These requirements focused on speed setting increments, rotational speed ramp-up, and rotational speed ramp-down, respectively, and all performed great in our tests as can be seen in our test matrix. The centrifuge was able to easily set the desired speed with the buttons on the front panel and see the speed displayed on the LCD. When the on/off button was pressed the centrifuge would start up and stop in a reasonable amount of time.

For **requirements 2.1.1, 2.1.3,** and **2.1.6**, our team marked these as passed but needed improvements. When testing our centrifuge, the motor was capable of reaching 2500 RPM when empty, but when tubes were placed in the rotor base, the centrifuge was no longer able to hit 2500 RPM when the desired speed was set to 2500 RPM and the voltage to 11V. With two samples and the voltage set to 12V, the centrifuge was able to reach 2500 RPM, but running the circuit at 12V is not sustainable and will be too much for the circuit to handle for long periods of time. As a result, our speed was able to be set accurately for most of our speeds except for our max speed. One of our biggest improvements from V2 to V3 was that we were able to read the speed of the centrifuge as it was spinning, which we checked for accuracy by comparing the oscilloscope read outs to the printed speed on the LCD. However, we found that even with our PID controller, the speed was unable to maintain an accurate speed causing it to fluctuate with an error of greater than 5%.

Figure 51. An oscilloscope readout showing the voltage differences in both the comparator and the phototransistor. The matched rise in both indicates that the comparator is successfully interpreting a change in light that the phototransistor receives.

The duration requirements were once again all met. Not much was changed apart from fixing a few minor bugs, as the duration part of the centrifuge was working well in V2. Our duration setting polling scheme software and circuitry worked well and consistently. We passed **requirement 2.2.1** easily throughout successful software integration as well as **requirement 2.2.2 and 2.2.3** using our button counter to incrementally add 30 seconds looping to zero at 600 seconds

(10 minutes). We were also able to measure duration accuracy with our internal countdown system displayed via LCD. We found an average duration percent error between .56% and 1.12%.

Regarding **requirements 3.1 and 3.2,** the on/off button worked very well. What we noticed in V2 was that our on/off button would occasionally need to be reset or would cause the system to reset even though we would click it to start the centrifuge. We hypothesized that there was some oscillating occurring when we clicked the button, so we connected the button to a capacitor. After connecting the button to the capacitor, we no longer experienced any difficulties.

We passed **requirement 4.1** by adjusting our software system. When our centrifuge is complete, a message appears that says “All Done!” and after resets the desired speed and time. This allows us to run the centrifuge again without needing to reset the system.

We were quite happy with the outcomes of our assessments with regards to **requirements 5.1 and 5.2.** After 5 minutes at max speed (see figures in trace matrix below), we saw successful centrifugation of both blood and turmeric powder + water. For the tumeric and water, our centrifuge was able to separate the powder and water much better than our V2 centrifuge. Furthermore, for the pig’s blood, our centrifuge was able to separate the blood into two very distinct layers and a third layer that was harder to see. While the centrifugation could be improved for our blood test, we were very happy with where it was at for our V3 design.

For **requirements 6.1, 6.2, 6.3.1,** and **6.3.2**, we often didn’t create tests directly for these as they had to do with general safety. We failed **6.1** simply because we did not put enough thought into our centrifuge being single-fault safe, instead focusing on building a working V3. We passed **6.2** as when our centrifuge was running, there were no hazardous situations that occurred. We did not test **6.3.1** and **6.3.2** because our system would not have been able to survive these tests. While our system had a lid to protect the motor system and samples, our physical encapsulation was resting on metal scaffolds that were not secured to the circuit encapsulation. Furthermore, our circuit encapsulation box contained holes that were used to feed the wires through to the motor and phototransistor-LED complex,

so if liquid was to escape from our samples in our centrifuge it would be able to seep through the holes and affect the circuit. This is something that could be addressed in a V4 build with more time.

Our centrifuge did very well in regards to **requirements 7.1, 7.2.1,** and **7.2.2**, which corresponded to protection against mechanical hazards. With the addition of our lid, our centrifuge was made significantly safer because the user was no longer able to accidentally touch the centrifuge. For injury to occur it would require the user to actively lift up the lid and touch the spinning centrifuge. As seen in our tests, our centrifuge was also capable of not tipping over when placed at rest on a flat surface, or at an incline. These requirements held true when the centrifuge was put to max speed.

Requirements 7.3.1 and **7.3.2** presented our system with a little more difficulty, as we failed **7.3.2** and passed with needs for improvement on **7.3.1**. For the most part, our centrifuge ensured a significant degree of protection because the motor and rotor base complex were protected from the user, and all of the wires and circuits were contained inside a box. However, our physical encapsulation was not secured to the circuit encapsulation, causing our system to become unstable if someone were to bump it too hard. In addition, the holes in our circuit encapsulation would become problematic if liquid was to spill on the centrifuge, as the liquid would be able to leak into the circuit board.

For V3, our team decided to add two additional requirements, **requirement 8**, which focused on indication of balanced tube weights in the rotor base, and **requirement 9**, which focused on auxiliary, non-spinning tube holders. Unfortunately, we were unable to reach our stretch goal of adding an infrared distance sensor, due to lack of time and focusing on fixing the bugs with our internal tachometer. This caused both **8.1** and **8.2** to fail since we were unable to integrate this aspect into our V3 design. However, **requirement 9.1** was successful! We were able to effectively add 4 external tube holders to our physical encapsulation. We tested the tube holders by placing a tube sample into each of the holders and letting them rest for 20 minutes. All of our tube holders passed the test. We enjoyed this requirement because it gave the user the ability to safely store the tube holders before and after centrifugation. In addition, the external tube holders were a great

way for the user to look at the samples post-centrifugation and detect if there was separation in the different layers of the samples.

High Level Requirements Checklist

Requirement	Failed	Passed (improvement)	Passed (no improvement)
1.1	☐	☐	☑
1.2	☐	☐	☑
2.1.1	☐	☑	☐
2.1.2	☐	☐	☑
2.1.3	☐	☑	☐
2.1.4	☐	☐	☑
2.1.5	☐	☐	☑
2.1.6	☐	☑	☐
2.2.1	☐	☐	☑
2.2.2	☐	☐	☑
2.2.3	☐	☐	☑
3.1	☐	☐	☑
3.2	☐	☐	☑
4.1	☐	☐	☑
5.1	☐	☐	☑
5.2	☐	☐	☑
6.1	☑	☐	☐

6.2	☐	☐	☑
6.3	☐	☐	☐
7.1	☐	☐	☑
7.2.1	☐	☐	☑
7.2.2	☐	☐	☑
7.3.1	☐	☑	☐
7.3.2	☑	☐	☐
8.1 (new)	☑	☐	☐
8.2 (new)	☑	☐	☐
9.1 (new)	☐	☐	☑

Trace Matrix

The trace matrix for Team 12's V3 build can be found here.

Assessment

Summary Discussion

The design cycle process for V3 flowed more smoothly than ever before, which is no surprise given that we had the chance to iterate on our design process multiple times before reaching V3. Though our biggest obstacle was once again the constraint of time, we were generally thrilled with our final integrated product. The below summary discussion begins with a brief high level overview of the V2 build and its associated challenges. The bulk of the discussion, however, surrounds V3 and how we iterated successfully from V2 onward.

V2 & Associated Challenges

Planning, building, testing, and assessing for V2 proved immensely challenging for our team. When the planning process went underway, each of us had a preferred skill set we were looking to bring to the new centrifuge; Benjamin preferred mechanical subsystems and documentation, Montanna excelled with code and upstream design, and Andrew handled downstream design and circuitry. This methodology led us into V2 with a lot of momentum, allowing us to operate significantly ahead of our personal schedule.

N.B. See Our Overall Build/Test Plan here.

As one could imagine, though, this type of vertical knowledge acquisition made our integration process difficult. Though we were all experts with specific subsystems, no one had thought about how these subsystems would work together. We simply assumed that it would work. This was an error. It's important to note that we did predict the integration aspect of our build would take a lot of time. We spent hours in the lab in the days leading up to our final session trying to get the whole system functioning so that we could focus solely on testing and assessing. But the constant stream of problems that arose made the effort feel sisyphean.

Our biggest problem was the failure of our phototransistor + LED complex. Troubleshooting this problem took upwards of 6-7 hours, and even with aid from multiple TA's and instructors, we couldn't make progress.

How V2 Informed V3

Heading into our V3 build, our most pressing concern revolved around fixing our phototransistor + LED complex. Without it, we would be unable to accurately read speeds or utilize a control system to provide any kind of feedback. We were lucky to have done enough diagnostic testing in V2 to know that the mechanical system itself wasn't the problem; as a result, other aspects of V3 could keep moving forward even without a speed readout that fully worked. Thus, Montanna and Andrew worked largely on solving this problem while Benjamin built out a robust integrated mechanical system in the meantime.

Shortly into the V3 design cycle, Montanna and Andrew were able to successfully fix the complex, which ended up being flawed due to measuring time in seconds while looping in microseconds. With that malfunction patched, and with Benjamin having worked on integrating the system, the build stage of our process was dramatically expedited such that we were able to pivot to working on controls. In addition, this opened the door to tests that we were simply unable to conduct in V2 as a result of being unable to measure speed.

The plan and build stages of V3 were robustly planned, meaning that this design cycle was largely characterized by small iterations meant to optimize centrifuge performance rather than major upheavals or malfunctions. This saved both time and energy and allowed us to truly hone in on making a successful final product.

Something important to note is the failure of requirement 8, which called for a centrifuge that could tell whether it was balanced. As touched on above, we did not attempt to solve this new requirement we outlined in our V3 proposal (see Helpful Documentation section). This was simply a matter of time; we focused more on a completed V3 with appropriate documentation rather than trying to add too many new features at the last minute. Future builds would include this requirement.

As has been the case throughout the duration of class, our teamwork was seamless and trouble-free at every turn. We came into V3 with intentions to make sure to leave ample test/assessment time, and we feel confident that we were able to meet that goal. Our trace matrix was notably more quantitative for this design cycle, and we were able to very clearly identify whether we were meeting requirements. As a final anecdotal note, it was deeply satisfying to be able to pick the centrifuge up and walk out with it at the end of class; it seemed to be the most simple specification test proving that we made an entire centrifuge from start to finish.

Hazard Discussion

As elucidated in the design cycle documentation guidelines presented by the teaching team, testing certain safety requirements to failure was infeasible given constraints of both time and laboratory safety. Resultantly, these were assessed slightly more qualitatively in a continuous discussion between the members of Team 12 about whether or not V3 met requirements 6.1, 6.2, 6.3, and 7.1.

In V2, our problems surrounding safety can be reduced to 3 salient points:

- The entire system would break if any liquid spills.
- Spinning parts are touchable, introducing the possibility of user harm.
- The system had to be taped down, meaning it was unstable on its own.

By V3, we were able to solve the latter 2 problems. The acrylic lid and dome complex on top of the external encapsulation meant that the spinning parts were untouchable. An added bonus was that this made the rotating component of the centrifuge a complete layer away from the user, meaning that a physical malfunction wouldn't break the entire system and thus would be single fault safe. Due to the scaffold locking in the inner scaffold, the inner motor complex also became stabilized to the point where it experienced no lateral movement. This helped us with instability requirements.

Despite safety with regards to mechanical malfunctions and instability, our V3 build was still far from spill proof. Though the spinning part was mostly encapsulated from the *outside* (see figure below for counterexample), there were no single fault protective mechanisms built into the *inside* that prevented harm from a spilled sample. Put more simply, if a tube broke or spilled, the solution would touch the motor and drip through holes to the circuit below. To mitigate this, we might consider building in a laser cut disc of sorts that snugly fits the motor and slides directly inside the external encapsulation. Not only would that serve to link the motor with its outer shell and provide stabilization, but this would also prevent internal spills from destroying the entire system.

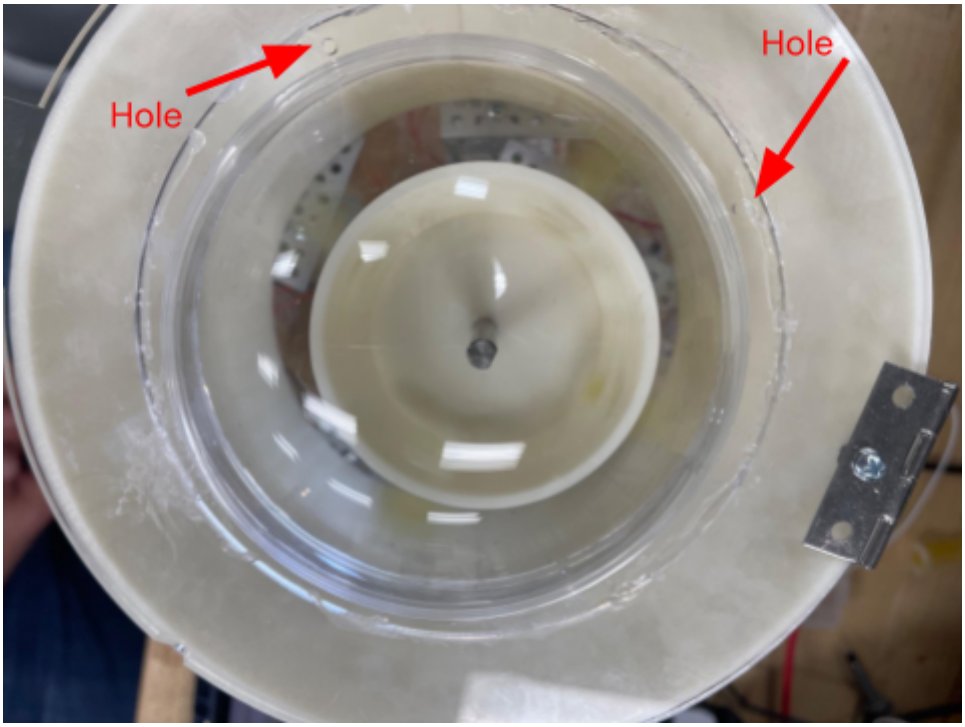

Figure 52: A top down view of the outer acrylic lid + dome complex. The dome came with 3 small holes, 2 of which are clearly pictured and labeled above. This was a small but clear hazard risk, as liquid spilled above the centrifuge may potentially have leaked in and compromised the motor.

Current Shortcomings of V3 Integrated System

Given the above trace matrix, high level requirements checklist, and assessment, we concluded as a team that our V3 centrifuge is a fully functional, integrated system. But that judgment does not preclude the idea that our centrifuge can be continuously improved for future hypothetical builds or iterations. Below is a list of the shortcomings we noted with our current V3 system and a proposed solution if we were to continue working.

Requirement	Shortcoming	Proposed hypothetical solution
2.1.1	Unable to reach 2500 RPM with 4 tubes in the rotor	This was a minor problem that we believe could be fixed with an

	base and at a sustainable voltage.	improved motor or with cutting down the size of the rotor base to create a smaller diameter, allowing it to spin faster.
2.1.3/2.1.6	While we were able to set the speed, the centrifuge was unable to maintain speed within a 5% error when desired speed is set.	Two issues to address. The first is with the button because our button would sometimes not increase the speed. This could be solved by adding a capacitor to the button or rearranging our circuit and connecting the button to an interrupt pin. The other problem was with getting the centrifuge to maintain a stable speed once set. This could be fixed by continuing to troubleshoot our PID system, until we come up with a control system that will fit into the 5% error.
7.3.1	Physical encapsulation was not secured to circuit encapsulation (physical encapsulation was resting on top).	This is an easy fix, as to address this problem would be to create an additional 3D printed part and glue/connect the physical encapsulation and circuit encapsulation together. This was difficult to address in V3 as the 3D printed part we would need would be large and there was a long cue for 3D printing.
6.1/6.3/7.3.2	Circuit encapsulation contained holes, which would allow liquid to leak through if samples are spilled inside the rotor base, or if liquid is spilled outside of the system.	To address this we would need to close off the motor complex from the circuit complex by adding a solid divider between the two. In addition, the above solution would help to cover the holes in the circuit encapsulation and prevent liquid from being able to spill into the circuits. We could also create

		smaller/fewer more specific holes that are closer to the center to address this solution as well.
8.1/8.2	The centrifuge was unable to indicate to the user if it was balanced and even if the centrifuge was unbalanced the system was still able to spin.	The solution to this would be having more time to incorporate the infrared distance sensor into our design. This would require assembling the sensor as described in our V3 proposal. We would include an LED light that would light up when the system is balanced and a connection to our motor, so that unless the light is on, the system will not be able to spin.

Helpful Documentation

Our project proposal for V3 can be found here.

Our order sheet for V3 can be found here.