Design Document

Ratty McBatty (the Dormouse)

Team A07: Gregory Boudreau, Rohith Nibhanupudi, Nasir Christian, Frank Malone

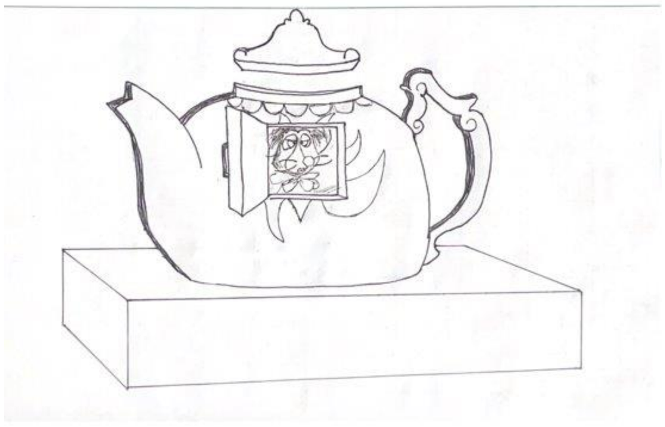
Table of Contents

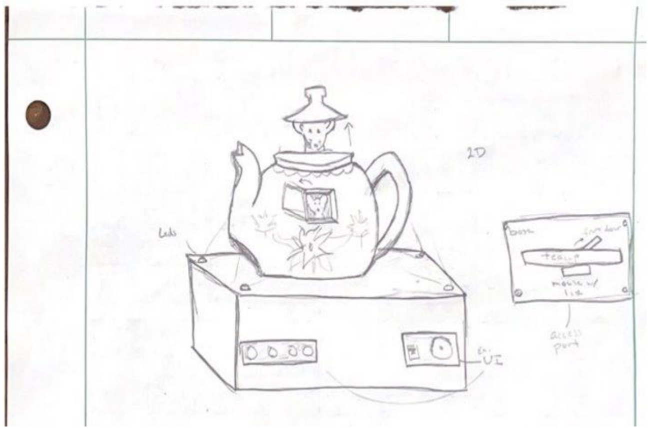
Revision Record

Date	**Author**	**Comments**
September 25, 2022	Gregory Boudreau	Document Created (system design, hierarchical design, and subsystem design)
October 16, 2022	Nasir Christian	Document was edited to fix the Processing Subsystem Section
November 13, 2022	Gregory Boudreau	Finalizing design document for final submission

Project Description

We have designed a 2D teapot which the dormouse will rise out of, lid attached to his head, and play music and lighting according to the cues received from the Director. The box is styled after the Dormouse from Alice in Wonderland. The dormouse will come out of his hiding place within the teapot to music that matches the scene from the movie.





Figures 1 and 2. Our initial design and rendering of the dormouse and his teapot

System Design

The inputs of the system are a 120V AC for input power, user IO buttons and dials, and a Wi-Fi input for the Director's cues. The outputs of the system include the dormouse's movement, the speaker's audio music, and LED lighting around the teapot. There are 5 total subsystems including the power subsystem, movement subsystem, analog audio subsystem, lighting subsystem, and processing subsystem. The power subsystem includes an AC/DC adapter (120V AC to 5V DC), a user input power control switch, and a fuse for safety. The movement subsystem consists of a DC motor and a rack and pinion device for vertical movement. The analog audio subsystem consists of a user input dial for volume control, an amplifier, and a simple speaker for audio outputs. The lighting subsystem includes a series of IO expanders, LED lights, and indicator lights. The processor subsystem supports a series of user IOs for individual testing and control, a Raspberry Pi, and the Wi-Fi connection to the director.

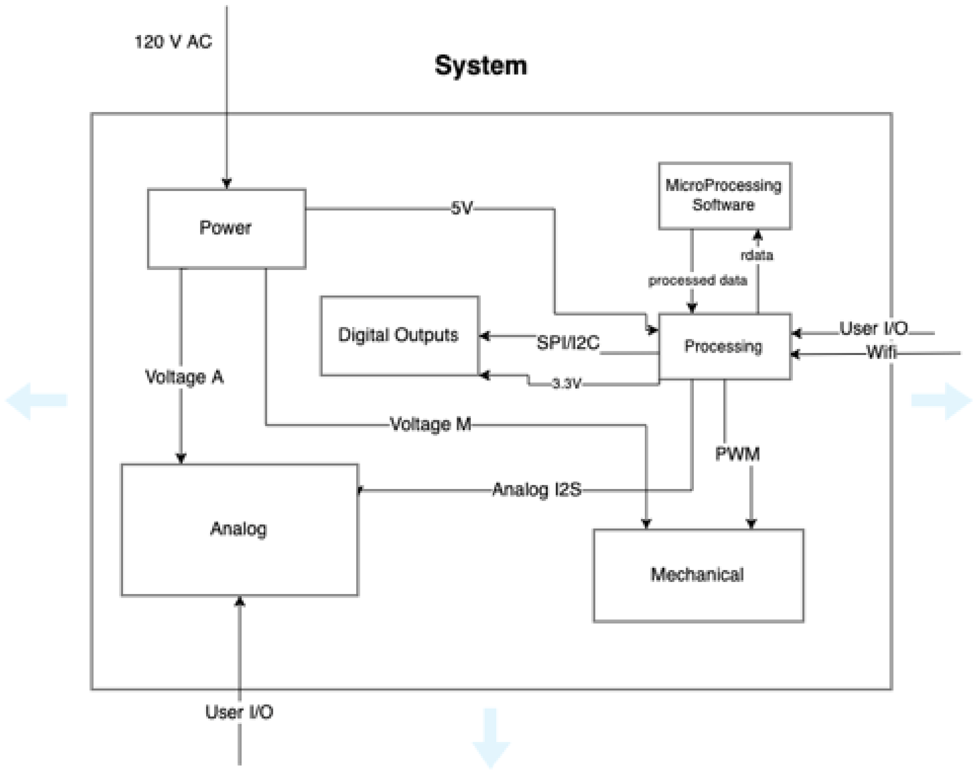
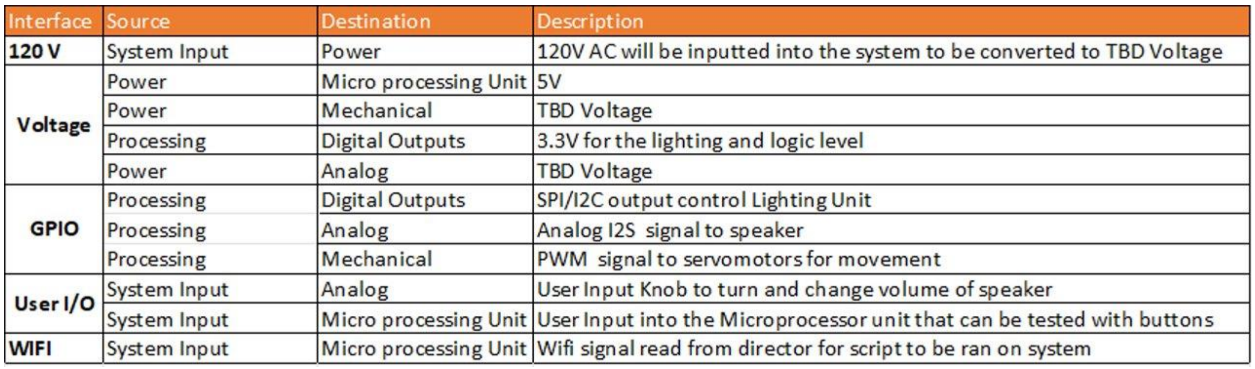


Figure 3. System Diagram



Interface	Source	Destination	Description
120 V	System Input	Power	120V AC will be inputted into the system to be converted to TBD Voltage
Voltage	Power	Micro processing Unit	5V
	Power	Mechanical	TBD Voltage
	Processing	Digital Outputs	3.3V for the lighting and logic level
	Power	Analog	TBD Voltage
GPIO	Processing	Digital Outputs	SPI/I2C output control Lighting Unit
	Processing	Analog	Analog I2S signal to speaker
	Processing	Mechanical	PWM signal to servomotors for movement
User I/O	System Input	Analog	User Input Knob to turn and change volume of speaker
	System Input	Micro processing Unit	User Input into the Microprocessor unit that can be tested with buttons
WIFI	System Input	Micro processing Unit	Wifi signal read from director for script to be ran on system

Table 1. Sub-system inputs and outputs

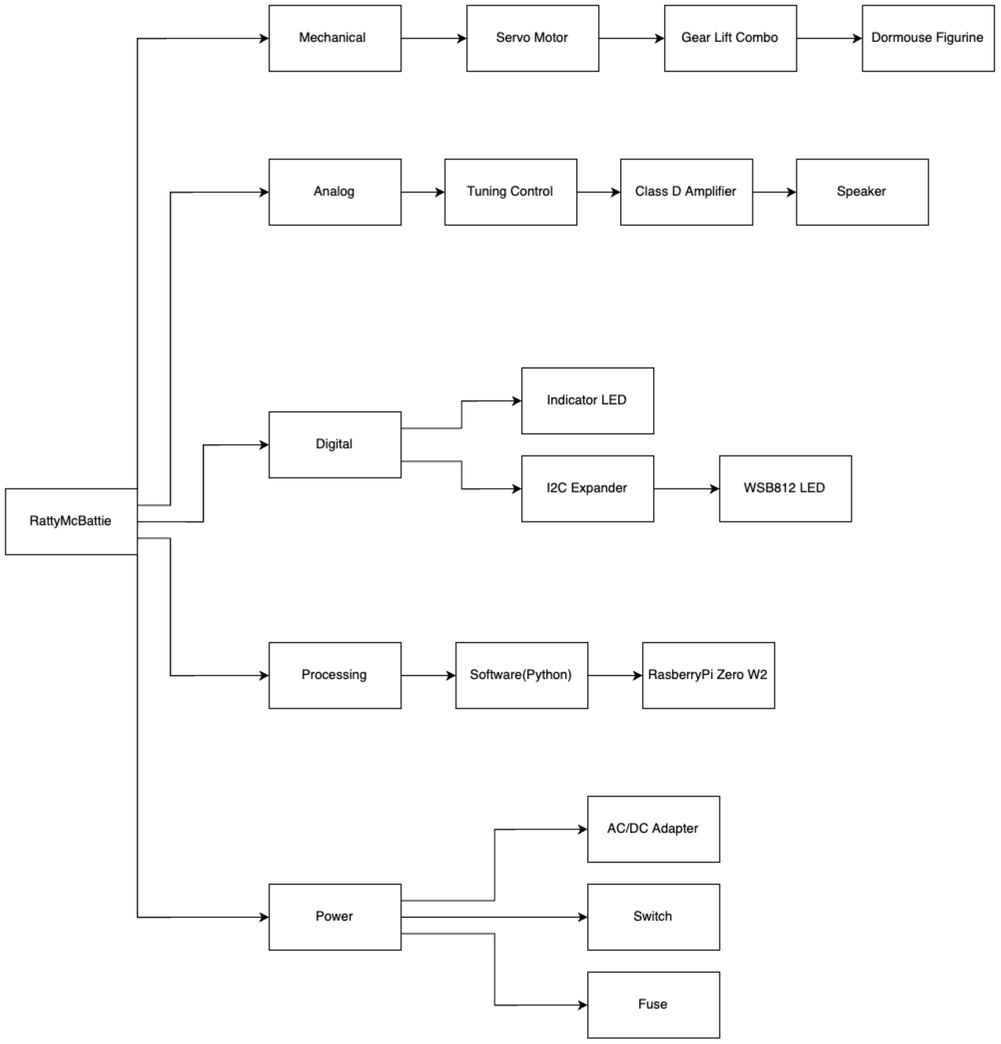


Figure 4. System Hierarchical Design with Components

Sub-System Designs

As described above, our subsystem incorporates five subsystems. The audio processes a digital audio signal from the processing subsystem and converts it to sound for when the movement subsystem raises the dormouse. The processor subsystem acts as an integrating mechanism, bringing together the controls of every subsystem, excluding power, and providing fine-tuned controls to ensure the system as a whole works in tandem.

Power Subsystem

The power subsystem takes in 120V AC from the wall plug and converts it to 5V with 4A current. This system also allows for full user control with a switch to disconnect power at any time. A fuse also ensures that no excess amount of current will endanger the system. The 5V power output is then routed to the processing subsystem, analog audio subsystem, and mechanical subsystem.

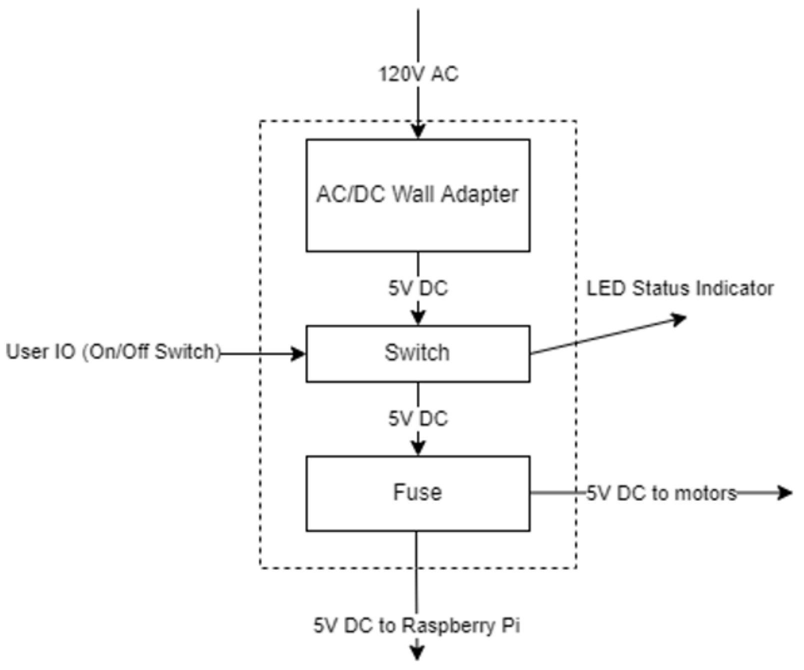


Figure 5. Power subsystem block diagram

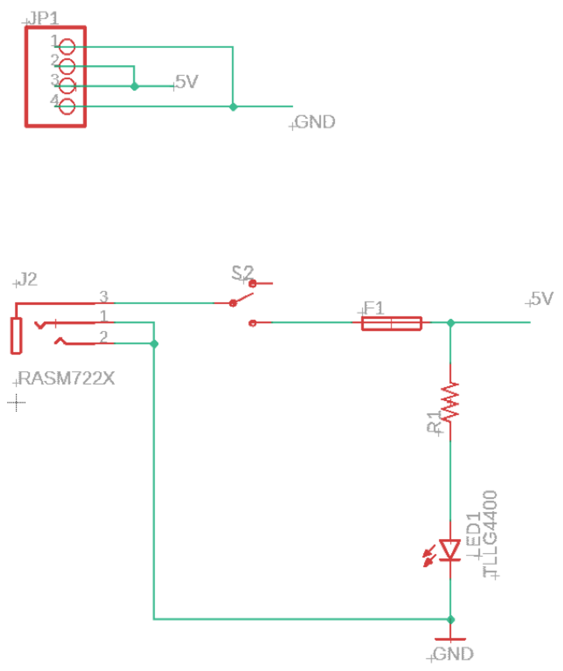


Figure 6. Power subsystem schematic

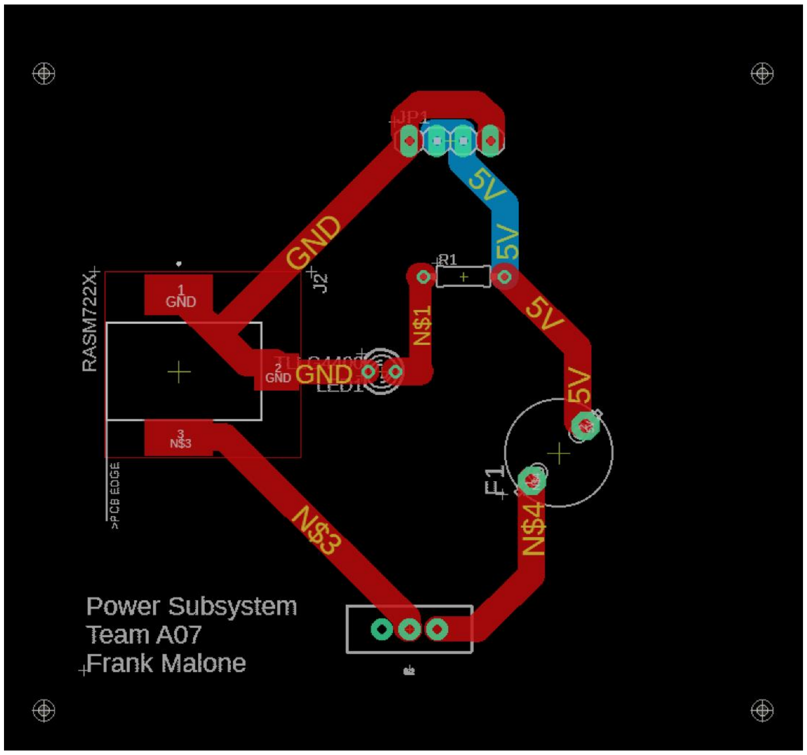


Figure 7. Power subsystem PCB

For the power subsystem, we attempted to reduce potential risk by creating a ‘worst-case’ power draw simulation to ensure that the power supplied would be enough to cover every subsystem being on and drawing max wattage at once. We found that, especially with the larger than expected power subsystem we had, we should have enough power for safe usage but not too much as to waste our budget.

Analog Sub-System

The analog audio subsystem will take in a 5V DC power signal, convert digital signals from the processor to analog audio that plays through the speaker, provide a physical control for volume, trigger the lighting sub-system, and provides testing functionality for toggling on/off.

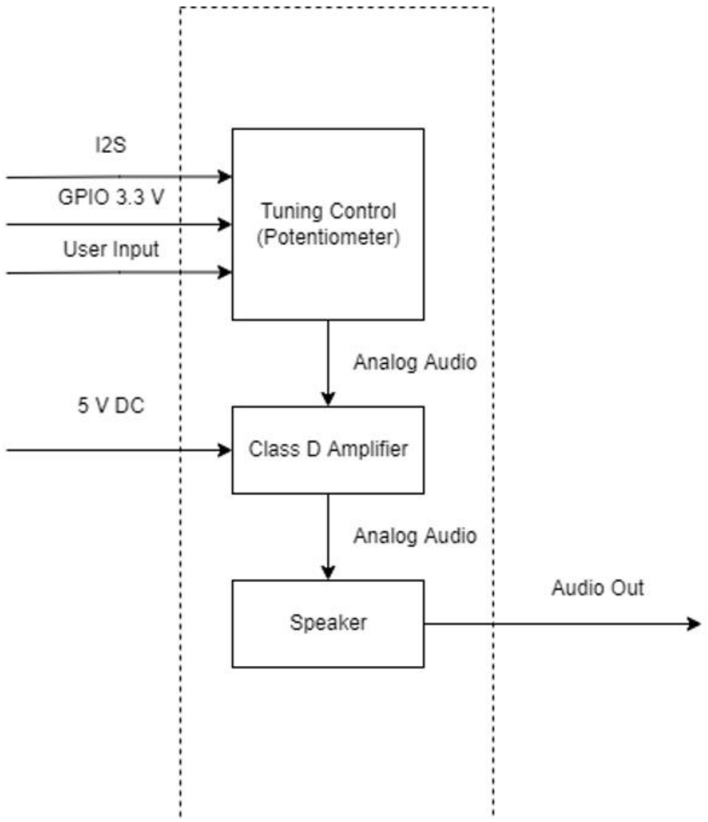


Figure 8. Analog Audio Sub-System Block Diagram

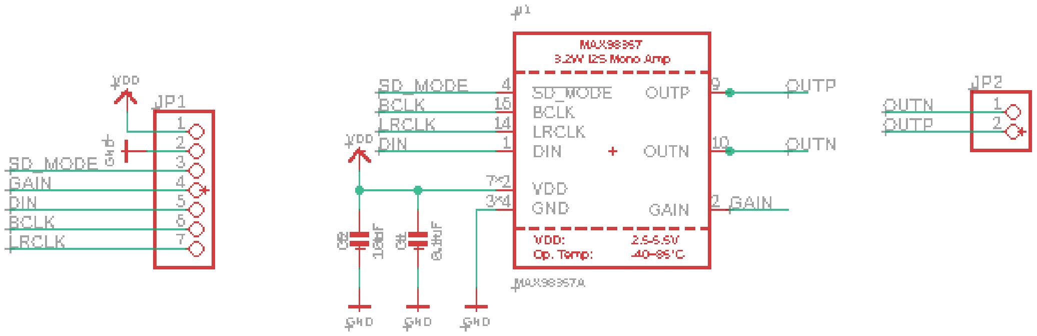


Figure 9. Analog Audio Sub-System Schematic

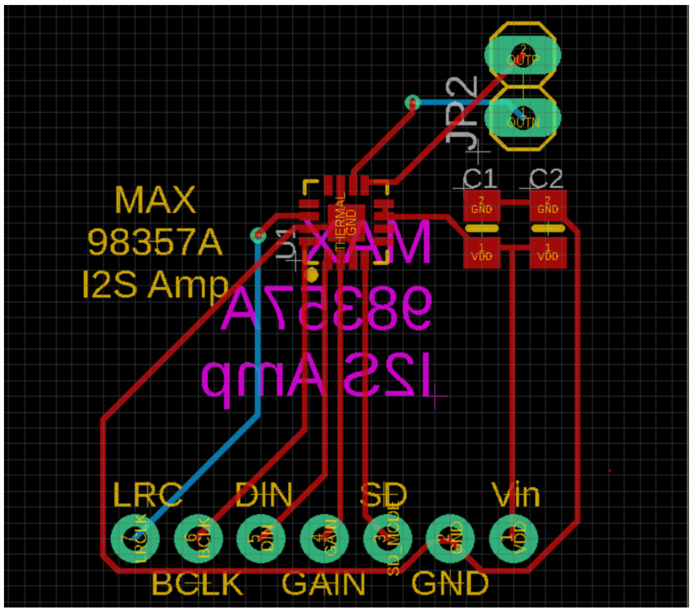


Figure 10. Analog Audio PCB layout

For the analog subsystem, we attempted to reduce potential risk by actively simulating our audio outputs with a Raspberry Pi 3. This was done because we only had a single Raspberry Pi 2 W Zero and had to work with available systems that have very similar structure and outputs. The Raspberry Pi 3 ended up being a risk 'introducer' in that we ended up facing major issues with the I2S driver we were using having had its support removed from the Pi zero on a previous date. Looking back at how the system was being developed, we should have tried to simulate the audio outputs with the board we were using for production or ensured that the driver and tools we were using were actively supported before continuing further.

Lighting (Digital Output) Subsystem

The lighting subsystem will take in 3.3V DC from the processing subsystem, an I2C signal from the processing subsystem, and an SPI signal from the processing subsystem. The I2C signal will control the LED lights on the front of the Box in the shape of a 'D'. The SPI signal will control the indicator LED, Labeled Pi Power i.e., Fig 12, to show the Raspberry Pi is powered and the Dormouse system is on.

We moved away from using an SPI or I2C IO expander chip because we could not receive our intended components and then found a simpler solution. We can actually only use a single GPIO pin because the lights were able to be soldered to one another with wire, this also cut down on cost and made the implementation of the code much easier and made it only have to use one GPIO pin form the Raspberry Pi and a correctly formatted PWM output to the lights.

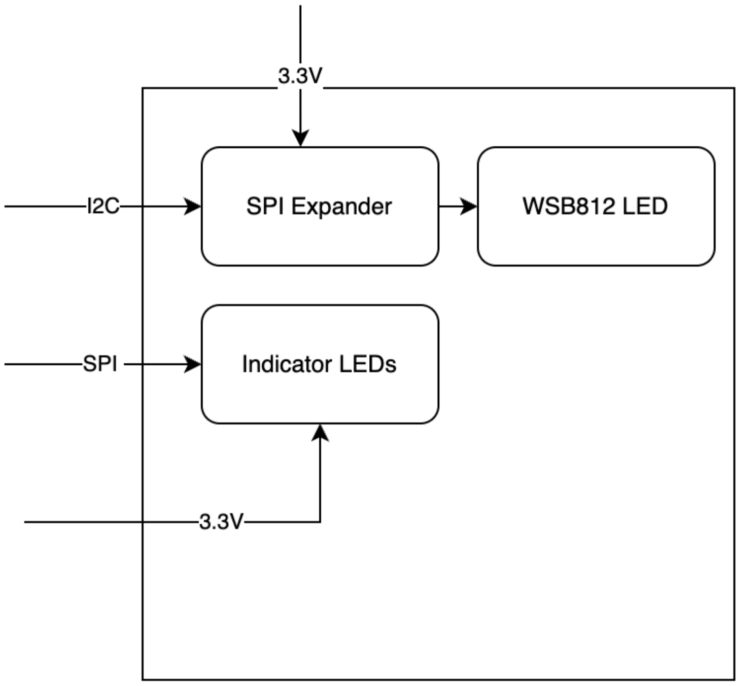


Figure 11. Lighting Sub-System Block Diagram

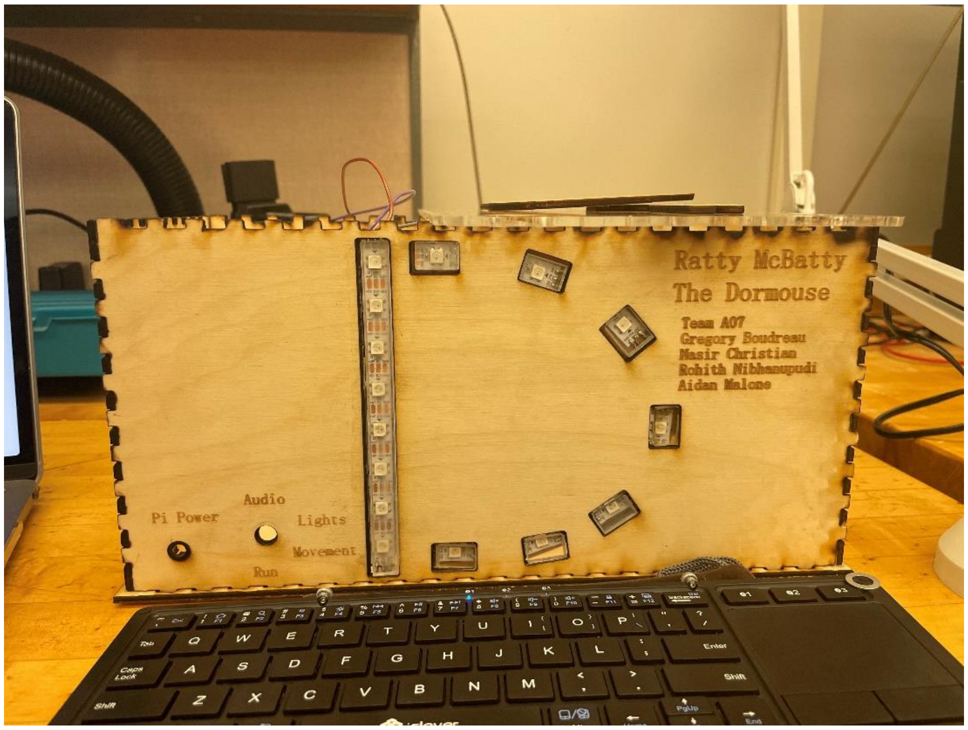


Figure 12. Lighting Diagram

For the lighting subsystem, we ended up designing it to be a visual feedback tool for users using the rotary encoder and as a pleasant lighting effect on the robot box. To reduce potential risks, this subsystem was actively tested and integrated with the software subsystem. It was the first integrated subsystem in the robot's box as it was extremely critical to the actual operation of our robot. The lights were one of our least risky systems as, because it was so closely tied with the software subsystem, it had the longest timeline to address any potential problems.

Mechanical Subsystem

The mechanical subsystem will take in 5V DC from the power subsystem and a PWM input from the processing subsystem to determine the speed and timing of the DC motor which will in turn control the direction of movement for the rack and pinion device. While the exact amount turned will depend on the gear ratio we employ, the ~17.5 oz/in capability the servo provides should hopefully be enough to handle the raising of Ratty McBatty.

The servo we had has been updated because of an issue with the continuous aspect of our previous motor. To hold the mechanical subsystem in place, it will be glued, with a small offset, to the back side of the box's wall. The rack and pinion are held against the gear by a wooden shaft we've attached to the bottom of the box. There is also a small wooden side piece to make sure that the rack and pinion doesn't slide out of the gear. Further, the rack is glued directly to the base and then slotted into place within the teeth.

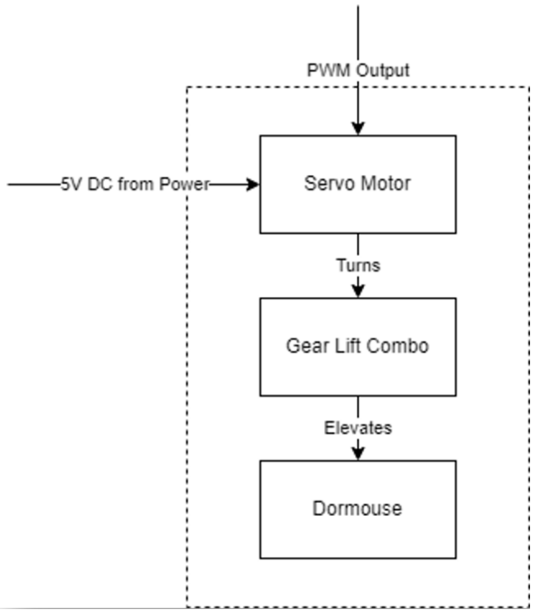


Figure 13. Movement Sub-System Block Diagram

For the mechanical subsystem, we attempted to reduce the risks associated with the system by attempting to test it frequently. This ended up being one of our least risks assessed systems which ended up hurting the team in the long run as the servo ended up NOT functioning as intended with the Raspberry Pi Zero, though it did function well with the mbed (likely due to some driver differences). Because of this, our mechanical subsystem will either eject the Dormouse from the robot OR pull it into the robot and potentially damage the wood by forcing it downwards through the small gap between the wood backing and the gear of the rack and pinion.

Processor Sub-System

The processor subsystem will take 5 Volts DC in addition to GPIO inputs from the Rotary Encoder from user I/O and will output 3.3 Volts DC for the lighting subsystem, digital signals (PWM) for the motor drivers, an SPI signal lighting subsystem, an SPI signal for the lighting subsystem, and an I2C signal for the analog audio subsystem. The schematic and PCB designed for the processing system were made to optimize the placement of wires.

The PCB ended up not working because there were no surface mount I2C expanders available on the market; also, the rotary switch we wanted could not be ordered because it was sold out. These setbacks made us use a PCB raspberry Pi breakout board and a GPIO 40-Pin Ribbon cable to use breadboard for the Pi

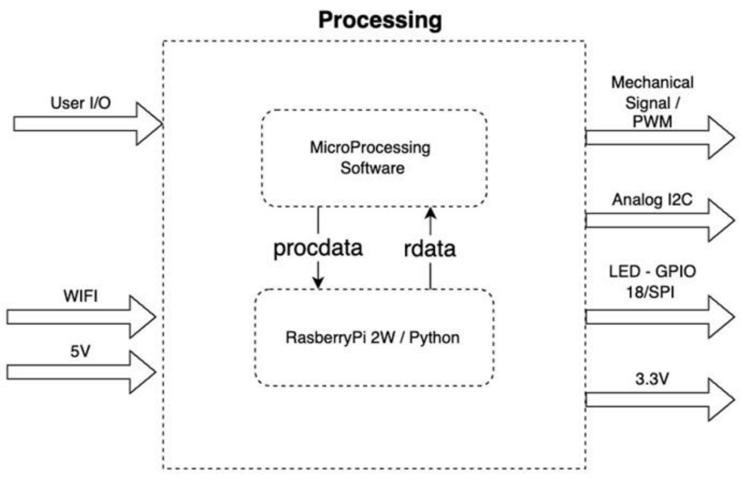


Figure 14. Processor Sub-System Block Diagram

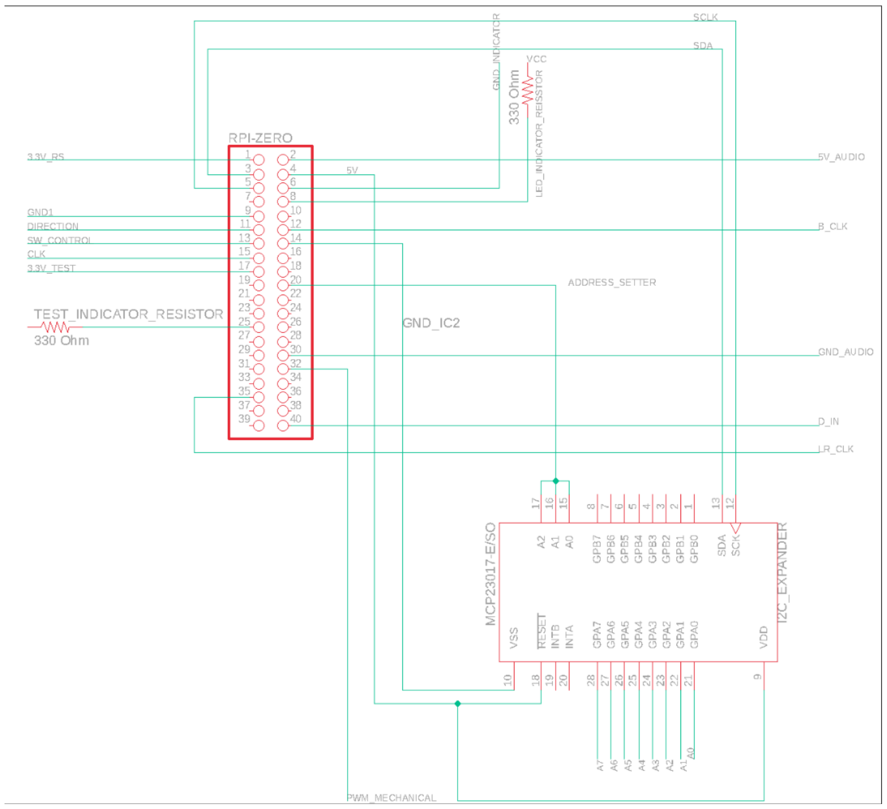


Figure 15. Processor Schematic

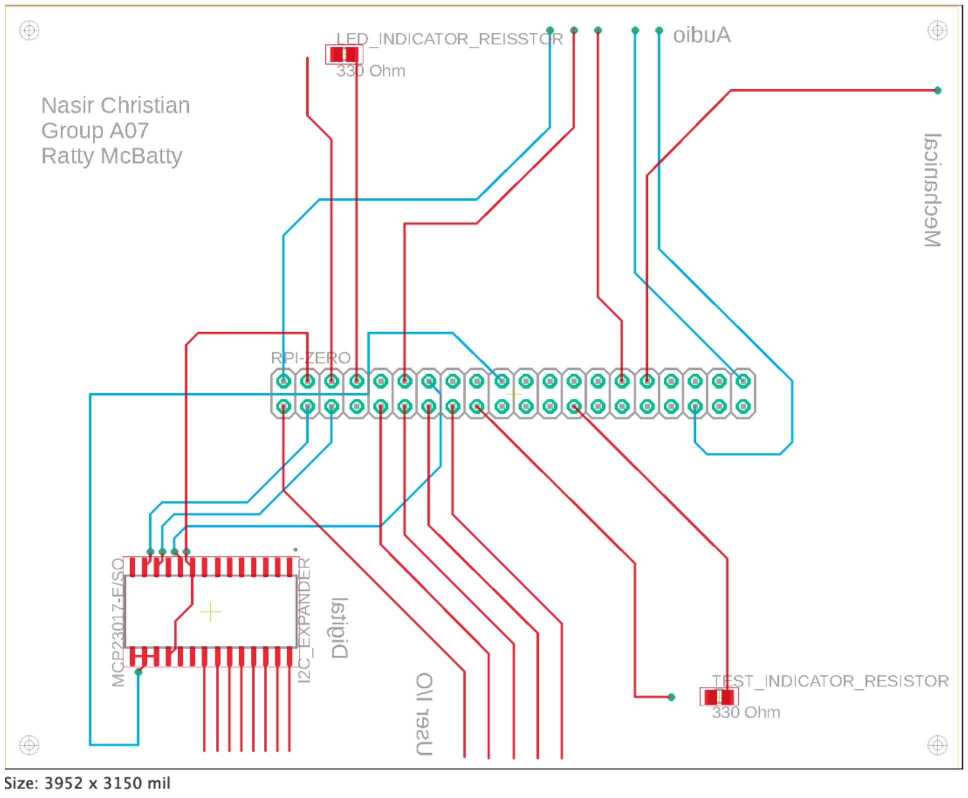


Figure 16. Processor PCB layout

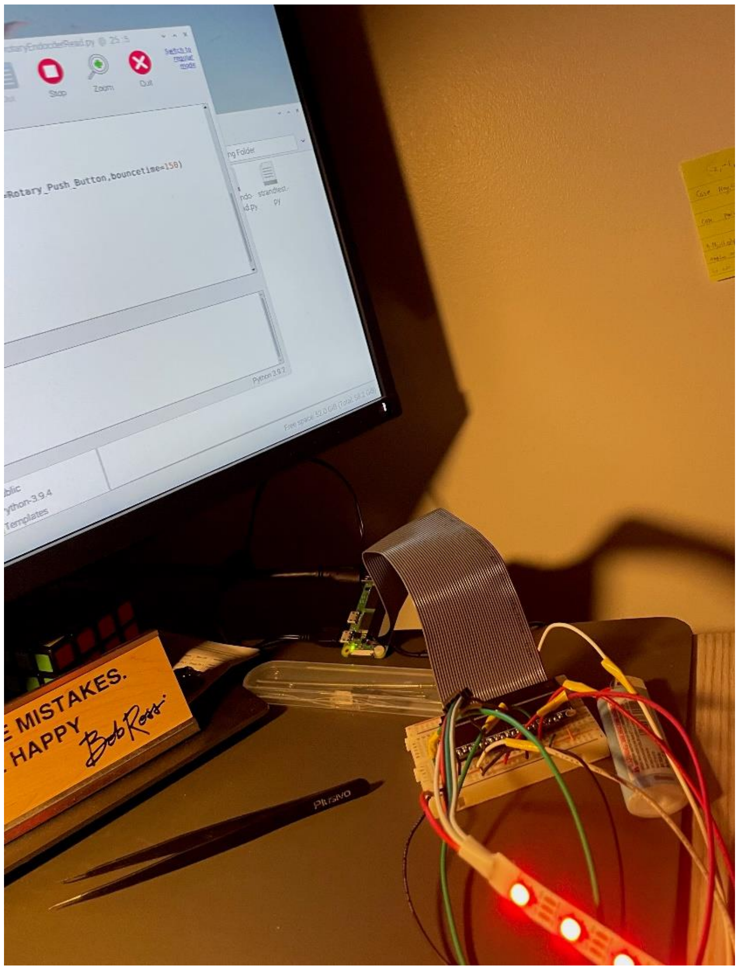


Figure 17. Processor Breadboard layout

The processor subsystem was our by far most robust subsystem. It was actively tested with the hardware it would be run by in the final robot. As such, there wasn't much we actively did to reduce its risks as most of the issues were found and addressed towards the beginning of the design process. The only issue that ended up hurting the system as a hole was our failure to ensure the driver used on the Raspberry Pi 3 for the audio worked on the hardware we had for our processor subsystem.

Constraints, Alternatives, and Trade-Offs

One major constraint we have had to consider is our power subsystem meeting power supply requirement as each subsystem has high demand. Our current power supply expectation of 5V and 4A may change as we finalize component selection and power simulations. Additionally, we were originally going to include a motorized latch to show the dormouse in his ‘hiding’ spot within the teapot, though we have since cut back on the far-reaching goals we set out with. The final major constraint we are facing is the $160 funding limit due to the increase in prices due to the global supply shortage. To address this, we are trying to find the components which can best fit our requirements without exceeding them too far to limit cost. Had costs been at a normal level, we would have used the Raspberry Pi 3 as our production board as we would have been able to create a clean audio and implement a robust threaded process from the software aspect (allowing for concurrent processes on button presses).

Mechanical Design

The mechanical design features a rack and pinion driven by a DC motor to lift the dormouse from behind the teapot. We have decided to 3D print the rack and pinion device as we could not find one that would fit our specific size requirements of both fitting neatly within the box enclosure and raising ~3-4 inches. Shown below (Figure 11) is a sketch of what we want for our rack and pinion component to look like. We will use a DC motor to rotate the gear which will raise or lower the dormouse’s platform. This platform will likely be laser cut from lightweight wood to not provide too much extra weight for the servo motor to lift.

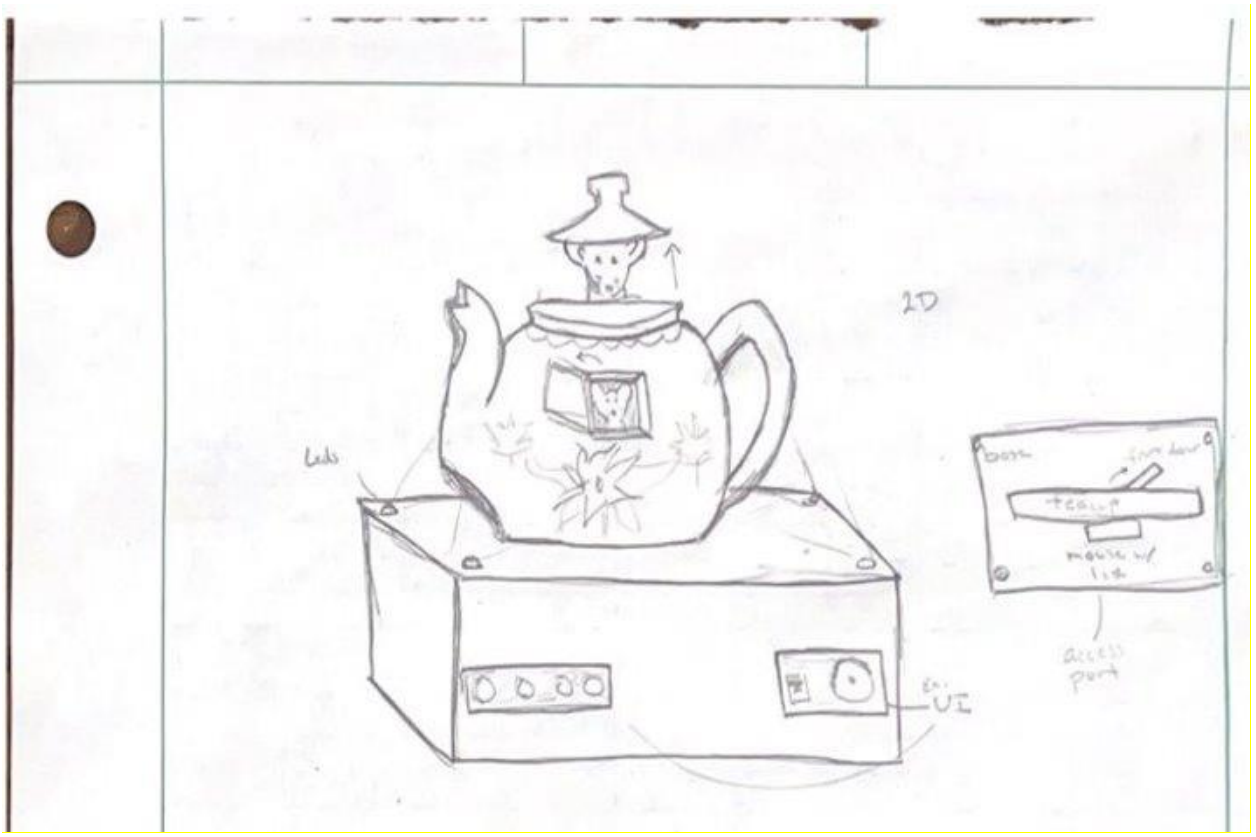


Figure 18. Rough Sketch of the Overall Design for the Dormouse and Teapot

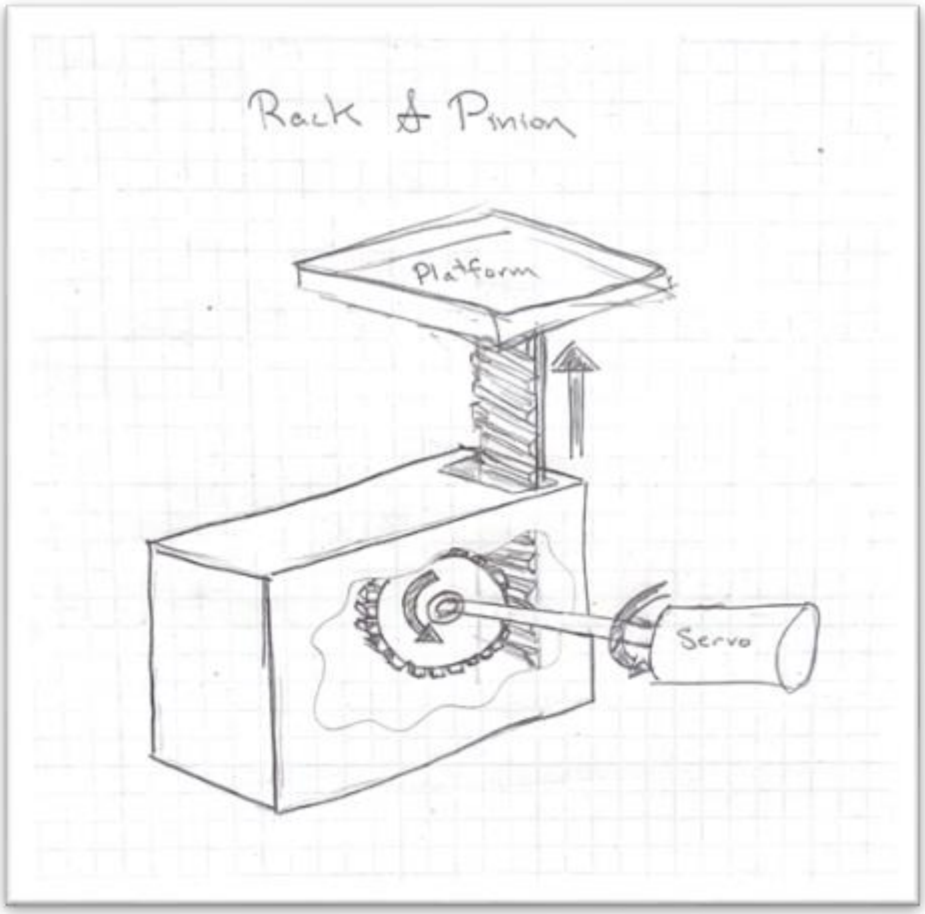


Figure 19. Rough Sketch of the Rack and Pinion Design

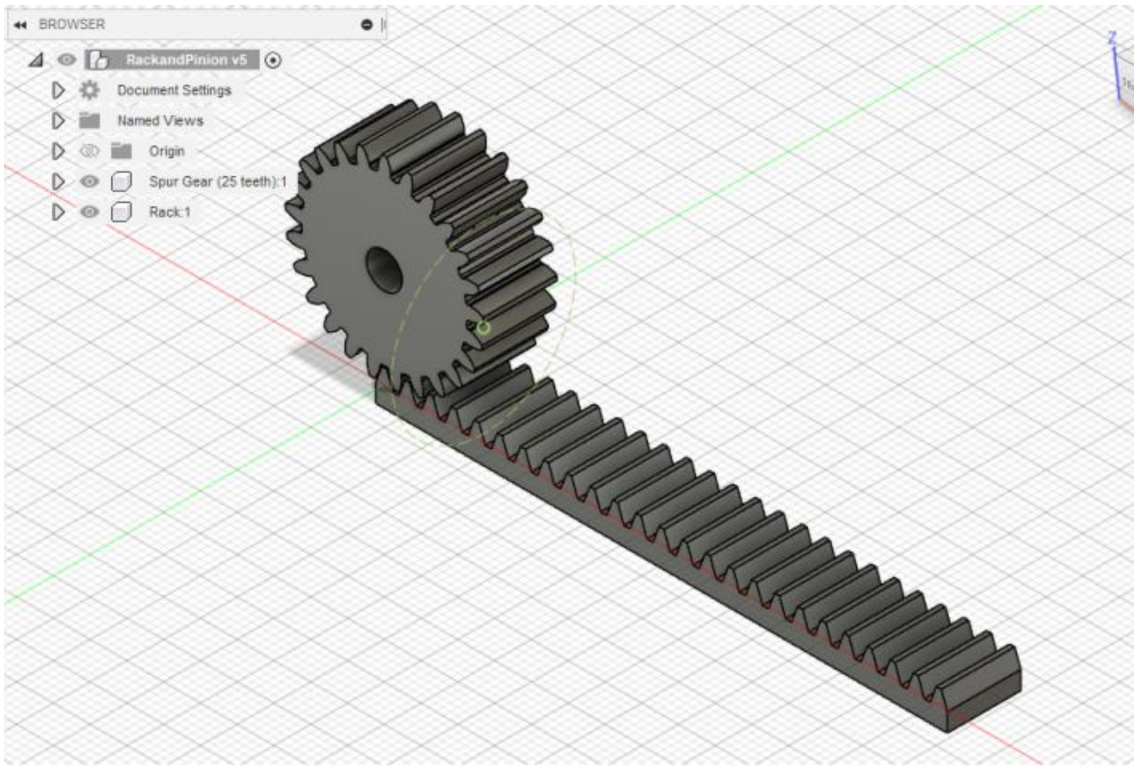


Figure 20. 3D CAD of the rack and pinion in Fusion 360

The roof of the teapot will be attached to the top of the dormouse's head to limit the requirement of additional motors and complexity to the overall design. This also removes the risk of a timing misaligning leading to the dormouse and teapot lid colliding if they were on separate motors. Originally, we intended to have two points of motion, one to raise the dormouse and another to show it 'hiding in the teapot.' However, we decided to real back in the goals we set to be realistic and ensure a robustly made product rather than one we had to rush to complete.

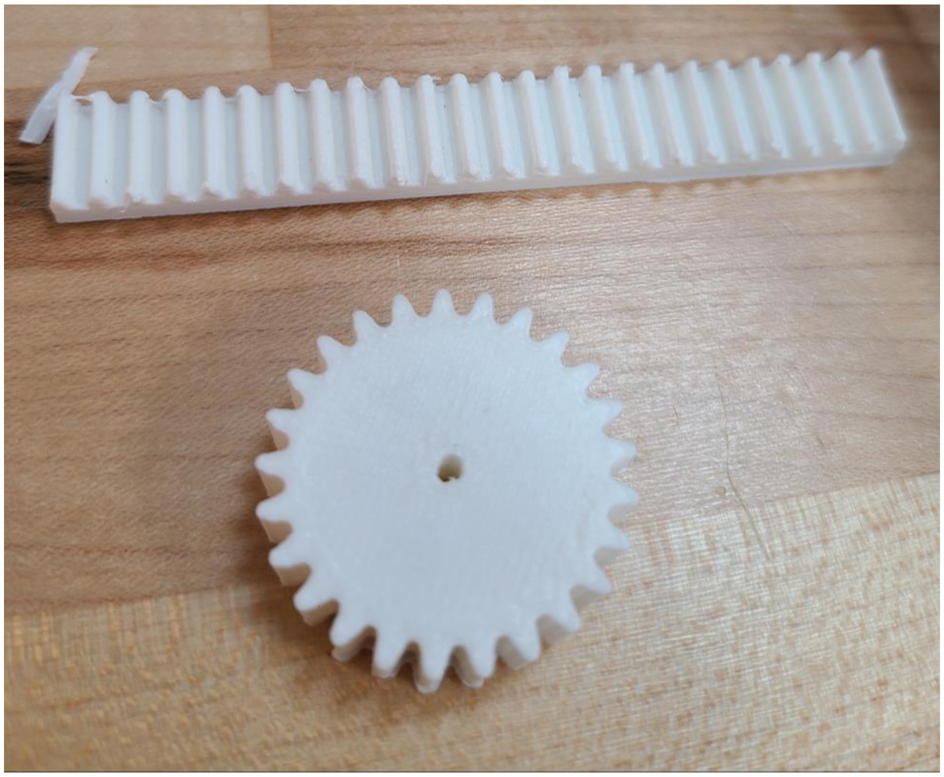


Figure 21. 3D printed rack and pinion for use in final integrated robot

Alongside a design focused specifically on functional implementation, we included some thematic designs on the exterior of the robot. This includes a quote from the Dormouse and a nod to the fact that, in the books and original Disney film, he is always sleeping or tired.

Software Design

Our software design consists of several states that can generally fall under two categories, demo and test. In the test category, the system awaits a user input from several different inputs to run individual or full system tests as coordinated by the processer subsystem. On the other hand, the demo category follows a consecutive flow by connecting to the director and, once connected, processing the lines and going through the whole process as displayed below (Figure 17). Using these states and the diagram below (Figure 13), we were able to design a software architecture to better detail our structure.

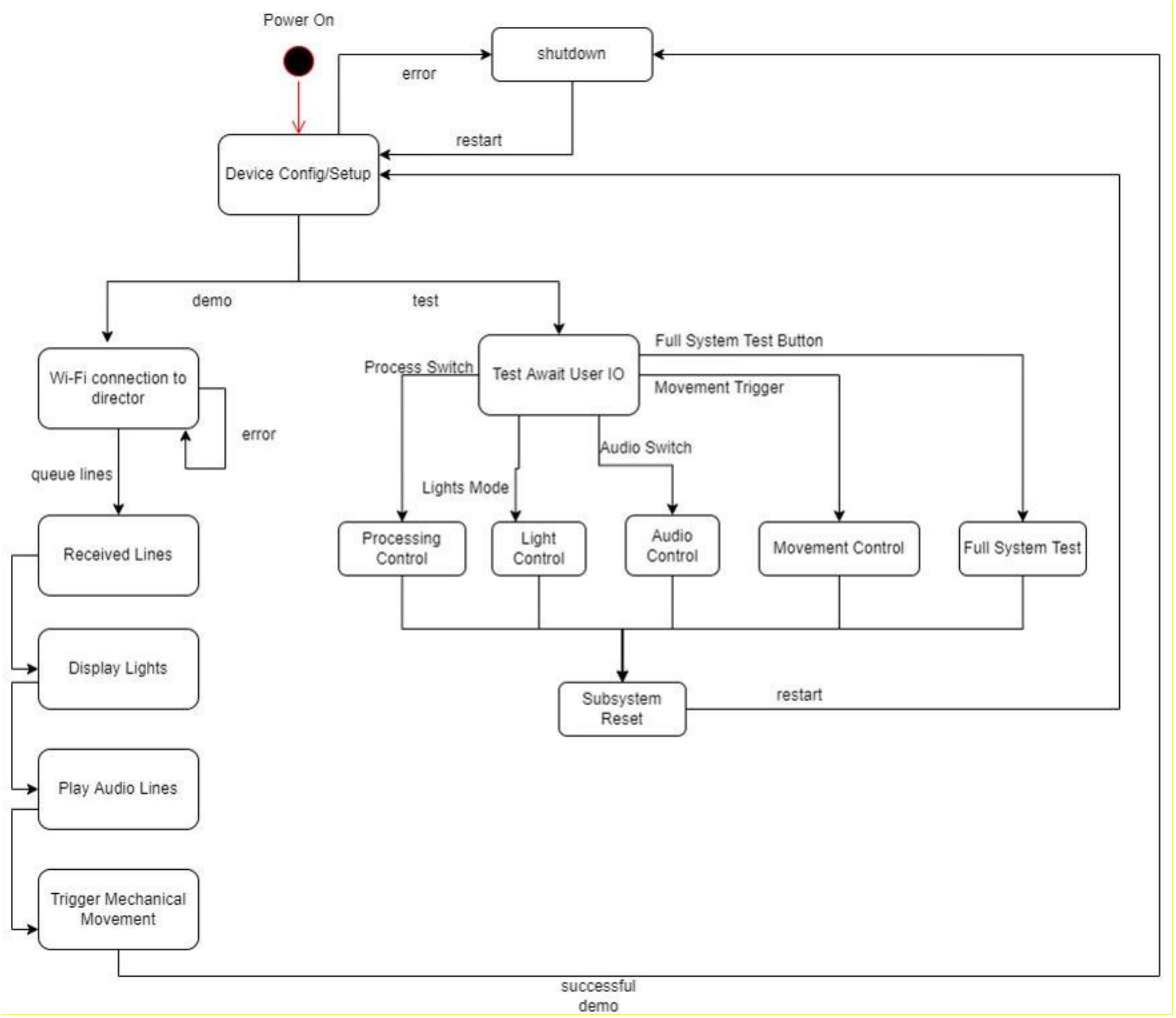


Figure 22. Software State Machine Diagram

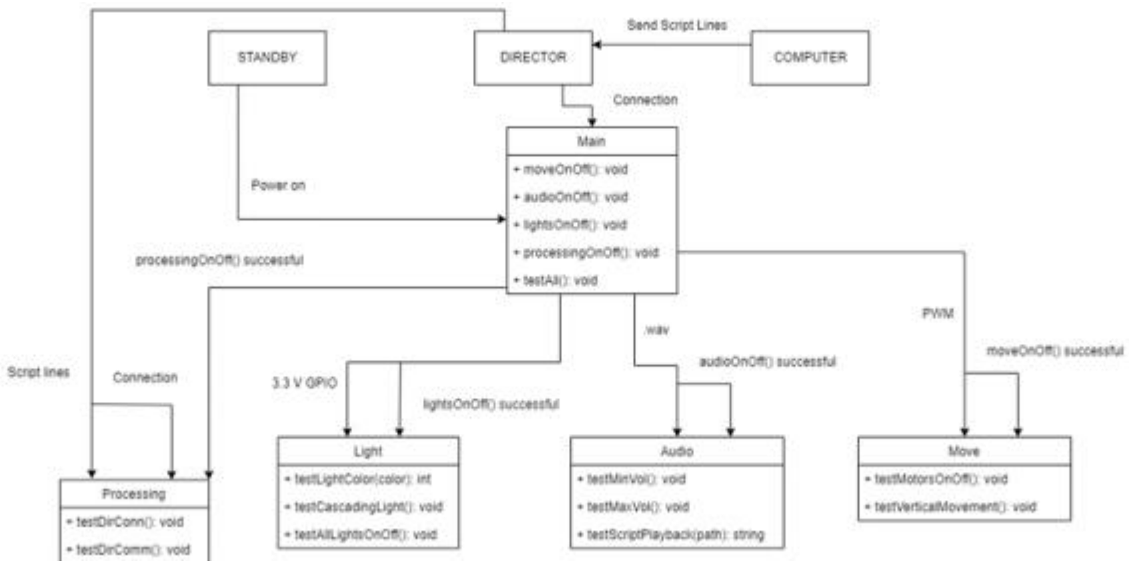


Figure 23. Software Architecture Diagram

Schedule

The schedule for completion of the project is shown in Table 2 where the task lead is indicated on each task.

	Week Number												
Task	1	2	3	4	5	6	7	8	9	10	11	12	13
Brainstorm/Sketch	Aidan	Aidan											
Design Top Level Block Diagram	Nasir	Naisr											
Design Software Block Diagram	Rohith	Rohith											
Requirements Analysis	Gregory	Gregory											
Proposal		■											
Requirements Decomposition			Nasir										
Full Subsystem Breakout Design		Gregory											
Power Circuit Design		Aidan											
Power Circuit Testing Plan			Aidan										
Mechanical Systems Design			Gregory										
Audio System Design			Rohith										
Motor Capabilities Research				Gregory									
Microprocessor Software Design				Nasir									
Microprocessor Software Testing Plan				Rohith									
Processing System Design					Nasir								
Microprocessor Software Simulation					Rohith								
Power Circuit Simulation					Aidan								
PDR Submission Aggregation					Gregory								
PDR					■								
Audio PCB Design							Rohith						
CAD of Mechanical Components							Rohith						
Power Circuit Build										Aidan			
Power PCB Design							Aidan						
Power Requirements/Test Plan						Aidan							
Mechanical Systems Build										Gregory			
Audio System Build										Rohith			
Finalizing of BOM								Gregory					
Full System Schematic Completion							Aidan						
Processing Subsystem Build										Nasir			
Finalizing CDR Documentation							Gregory						
CDR Submission Aggregation								Gregory					
CDR								■					
ALL Parts Received (started earlier)										Nasir			
Audio PCB and Audio System Integration										Rohith			
Processing System Connections Mapping								Nasir					
Exterior Holding Build									Aidan				
Power Circuit Test w/ Load									Aidan				
Mechanical Systems Test										Gregory			
Audio Systems Test w/ Speaker										Rohith			
Processing System Test										Nasir			
Full System Integration and Connection										Aidan			
Full System Test											Rohith		
Full System Video											Nasir		
Final Inspection and Demonstration												■	
Milestones													
Order hardware components (PCB order and other tools)	Aidan												
Finalizing of Software	Nasir	Rohith											
Finalizing of Hardware	Gregory												
Continuous Documentation	ALL												

Table 2. Schedule to complete Ratty McBatty (the Dormouse)

Integration

It is extremely important that every subsystem works together in tandem with the other subsystems to ensure that we have a fully functional system. We will implement a series of simulations and tests to ensure this functionality. First, we will simulate the software using a small-scale breadboard setup to test software functionality using different IO ports on the Raspberry Pi, and then using the Raspberry Pi and software we can test each subsystem individually. Additionally, for the power subsystem, we will simulate maximum (worst-case) power draw from each component and compare against the maximum power supplied by the power subsystem to ensure a healthy supply of power and track usage against our 60W maximum set forth by the customer. By implementing a small-scale breakdown of each system and their testing requirements, we can progress and test at a more constant rate instead of waiting to check a system when it has been fully completed.

Software High-Risk Parts:

- Our by far highest risk component is the Raspberry Pi as it functions as the primary controller of every subsystem, excluding the power subsystem. In the case that the Raspberry Pi fails, the entire machine will no longer be functional. As a result, we have decided to begin by testing the Raspberry Pi and software components first to ensure that this critical component is functional before expanding its connections in physical implementations. Additionally, the Raspberry Pi will power and control an indicator light to show its status to the user and continually confirm functionality after the build has been finalized.

Mechanical High-Risk Parts:

- The highest risk component of the mechanical hardware is the servo motor; As the only moving component of the system, its functionality is critical to the visual aspects of the system. Also, the motor can draw the most power of any component which provides another risk. If the motors fail, Ratty McBatty will not rise out of the teapot and our customers will be left disappointed at a design that does not meet their requirements.
    - An additional high-risk component we've identified is that the servo can severely damage the Dormouse figurine if it turns for too long by either ejecting it from the robot OR pulling it down and into the rack and pinion structure, potentially breaking the wood with the downwards force and small space applied.
- We have a new risk associated with the mechanical system in that we are not certain if the servo we have received can raise the weight of Ratty McBatty. The ~17.5 oz/in rating from the datasheet should be capable of raising the weight of the figurine but this is something that needs to be tested soon.

Analog Audio High-Risk Parts:

- The highest risk component of the analog sub-system is the I2S Class D Amplifier chip. Since it is the core component of the audio PCB, it must be able to decode the digital audio signals and output the correct audio at the audio levels specified by the customer requirements. Moreover, the chip must be able to interface with the physical volume

control to give the users the ability to adjust the volume between the minimum and maximum amounts. If the chip fails, the system will not output sound that meets the customer's expectations.
    - As we found out during integration, we disregarded the potential that the audio I2S driver does not work on our production microcontroller.

Repository Management:

- We have a GitHub that will include all our software files as well as any design files related to the electrical and mechanical systems.
    - Team A07 Github Repository
- We also have a Microsoft Teams OneNote which we are using to track our progress.

Configuration Controls:

- We intend to have several user IOs to allow users to test our device and control both audio volume and power. These will consist of a series of pushbuttons for the testing IO which connect directly to the Raspberry Pi, a dial to control the audio volume connected to a potentiometer on the analog audio PCB, and a switch to turn power off or on in the power subsystem.

Issues with Integration:

- We went through a process of individually building and testing each subsystem to keep our time constraints as modular as possible. This did, however, come back to haunt us during the integration process. Because each subsystem was individually built, there were some simple issues of tying together the connections. For the movement subsystem, it was built and tested with an mbed microcontroller, but for some reason, the PWM input from the raspberry Pi did NOT function as intended (this resulted in us having to switch to a new servo motor). For the audio subsystem, it was built and tested with a Pi 3 and when transitioning back to the Pi zero, it was found that the Pi foundation had removed support for the driver implementation we were relying upon for the I2S audio (this resulted in no audio). For the power subsystem, we had some issues with the switch failing to work (though in the end it does function as intended). For the lighting subsystem, we had a major wire attachment failure during the demonstration on 11/9. One of the channel connections for the A/B signals of the rotary encoder broke off from the solder though it was fixed later that day and functioned as intended during the final video.
- We found that a lot of our issues were due to a lack of focus on integration and a greater focus on individual subsystems. To address this in the future, we believe we should have focused on a more integrated development system and bring together the subsystems at an earlier point in our schedule. The process by which we went left us with little time to address potentially major integration problems.