Design Document

Insani-Tea

Team #C02: Carson Brown, Rajev De Silva, Mark Moran, Hung Nguyen

Team Lead: Carson Brown
Power Lead: Mark Moran
Light/Audio Lead: Hung Nguyen
Mechanical Lead: Rajev De Silva
Processor Lead: Carson Brown

Date: December 6, 2022

Table of Contents

System Design 2
Power Subsystem 5
Processor Subsystem 7
Audio Subsystem 8
Light Subsystem 10
Mechanical Subsystem 11
Software Design 13

Revision Record

Date	**Author**	**Comments**
Sept 21, 2022	Carson	Document Created (software, hierarchical design, interface description)
October 19, 2022	Carson	CDR Modifications (software simulation, subsystem sections)
December 6, 2022	Carson	Final Modifications

System Design

The system design consists of five subsystems including processor, mechanical, power, audio, and light. The processor subsystem includes the Raspberry Pi Zero W. The mechanical subsystem includes a hand and a servo motor. The power subsystem includes a power supply. The audio subsystem includes a speaker and an audio amplifier. The light subsystem includes an LED strip.

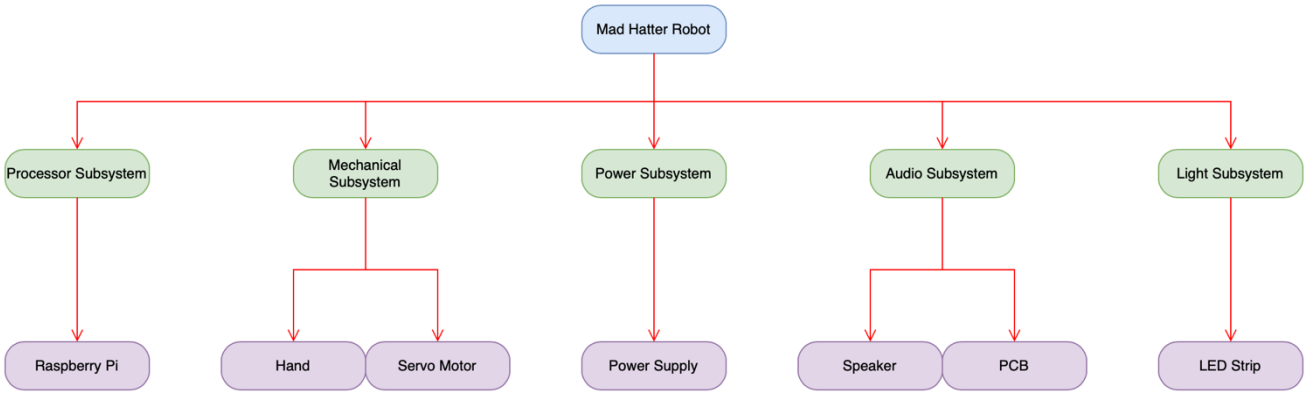

Figure 1: System Diagram

The inputs of the system include 120V AC input power, user input, and director commands. The outputs of this system include light, sound, and movement. Shown below are the interface diagram and description table.

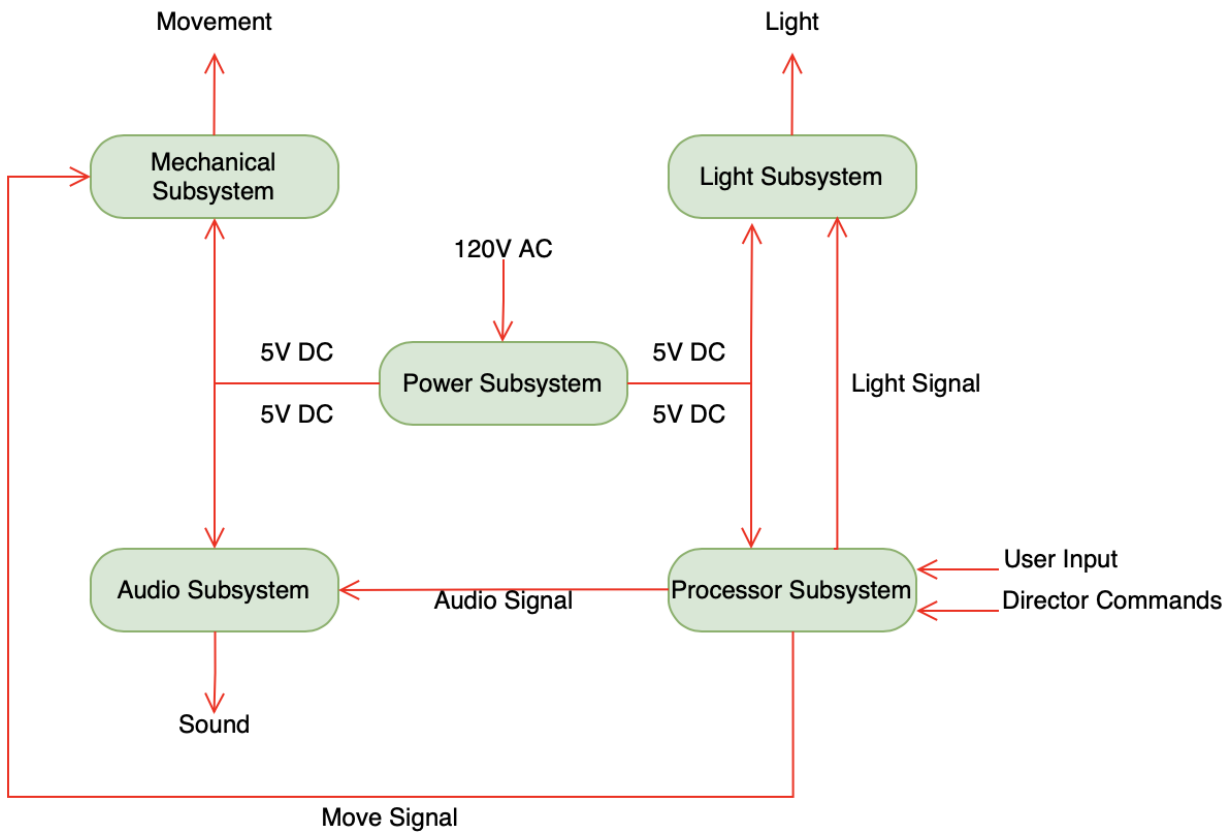

Figure 2: Interface Diagram

Table 1: Interface Description

Input/Output	Interface
120V AC	120V AC from wall outlet
5V DC	5V DC from power subsystem
User Input	Audio, light, and movement test buttons
Director Commands	Instructions communicated by the director server
Light Signal	Digital Out from Pi
Audio Signal	PWM Out from Pi
Move Signal	PWM Out from Pi

Light	Light patterns produced by the LEDs
Sound	Recited lines produced by the speakers
Movement	Hat tip produced by the motor

Shown below are our initial and final robot designs. We went through various iterations across the different subsystems, which will be enumerated later in the document.

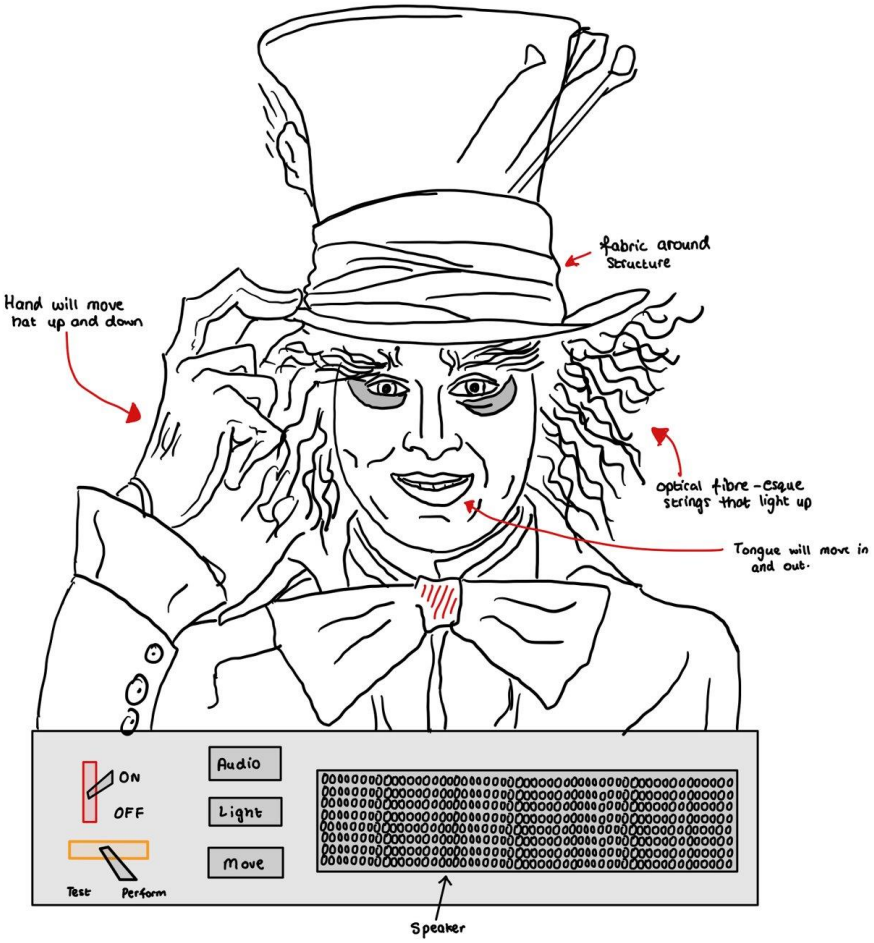

Figure 3: Initial Design

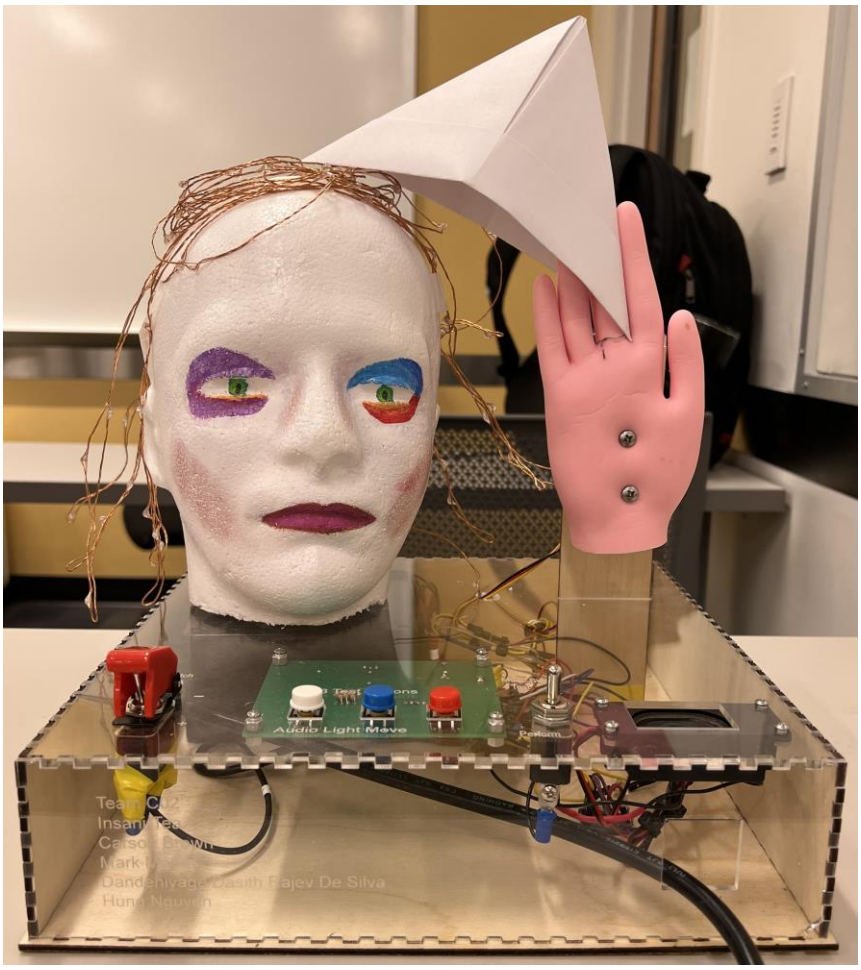

Figure 4: Finalized Design

Power Subsystem

The power subsystem takes in 120V AC from the wall outlet and routes it through a power toggle switch controlled by user input to the power supply. The power supply then outputs 5V DC to the other subsystems.

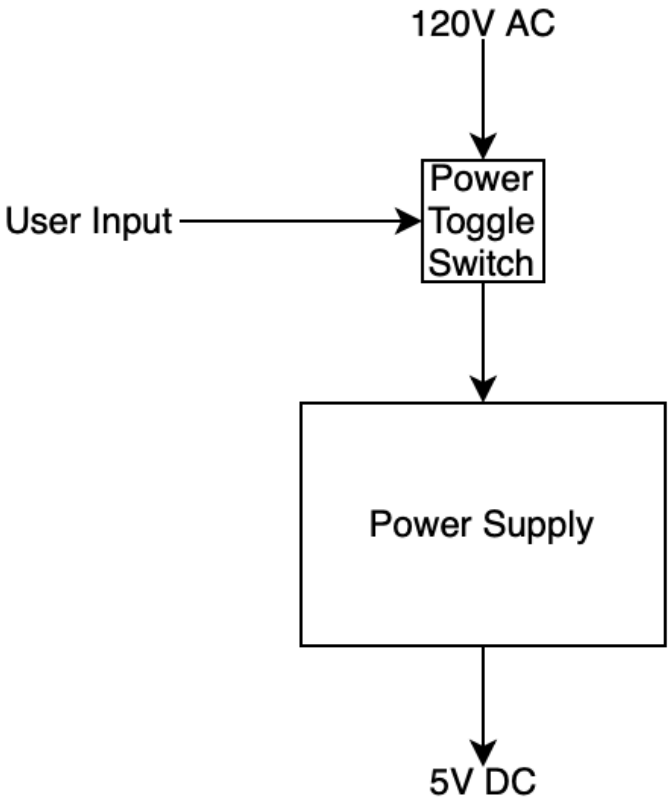

Figure 5: Power Subsystem Diagram

We bought our power supply rather than building it, so the only design work required was soldering the wire connections.

Figure 6: Power Supply and Connections

Processor Subsystem

The processor subsystem takes in 5V DC from the power subsystem to the Raspberry Pi Zero W. The Raspberry Pi also takes in control signals from the mode toggle switch, audio test push button, light test push button, and move test push button, all of which are controlled by user input, as well as director commands. The Raspberry Pi then outputs a PWM out audio signal, digital out light signal, and PWM out move signal to the respective subsystems.

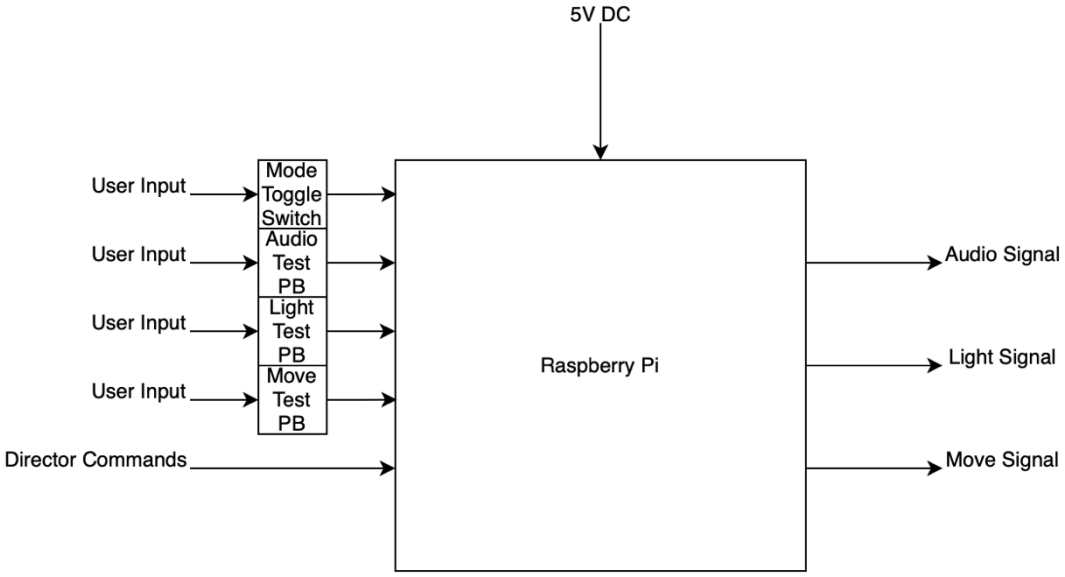

Figure 4: Processor Subsystem Diagram

The primary design challenge was the software, which will be discussed later in the document.

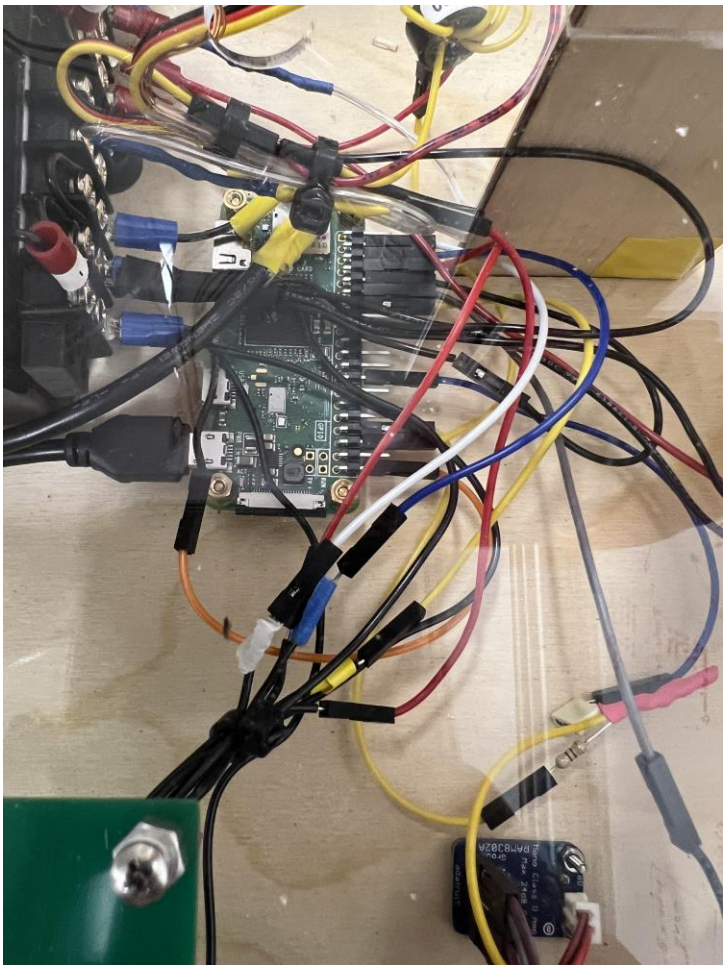

Figure 7: Raspberry Pi and Connections

Audio Subsystem

The audio subsystem takes in 5V DC from the power subsystem to the speaker. The audio amplifier circuit of the PCB takes in a PWM audio signal from the processor subsystem and outputs the amplified signal to control the speaker. The speaker then outputs sound.

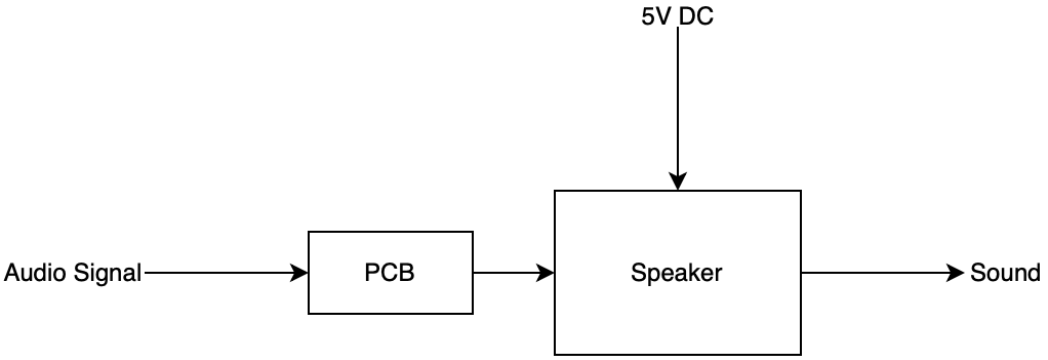

Figure 8: Audio Subsystem Diagram

We had initially planned to use our PCB as an audio amplifier for the speaker. However, the circuit design we utilized on the PCB shown in Figure 9 did not produce a strong enough audio signal.

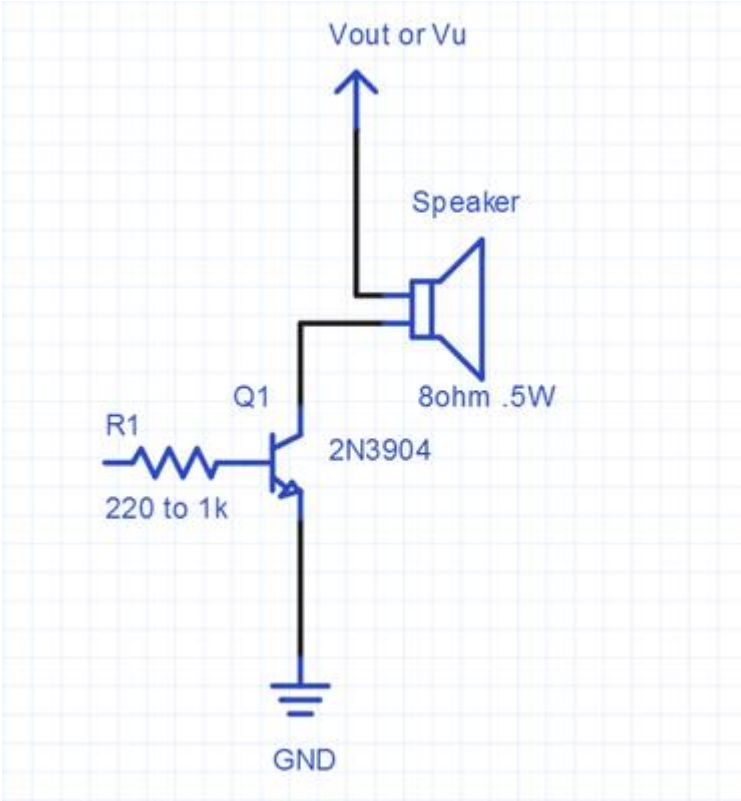

Figure 9. Initial Audio Amplifier Circuit

As such, we had to switch to using a Class D audio amplifier. We also built a low pass filter to clean up the audio signal.

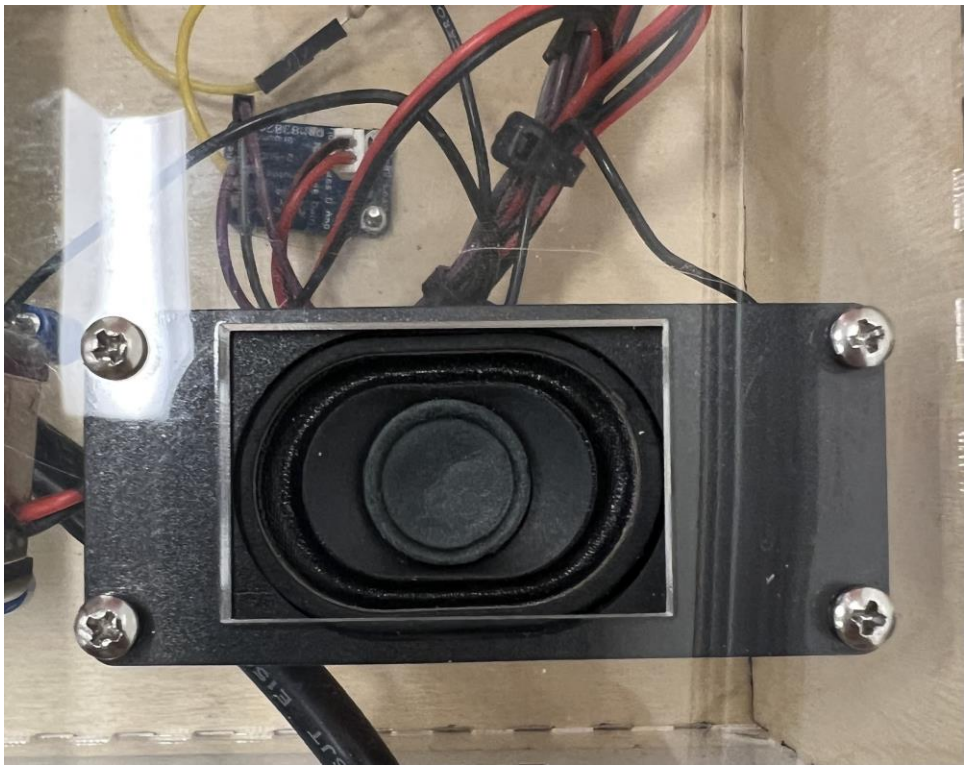

Figure 10: Speaker with Class D Audio Amplifier and Low Pass Filter

Light Subsystem

The light subsystem takes in 5V DC from the power subsystem to the LED strip. The digital light signal from the processor subsystem controls the LED. The LED then outputs light.

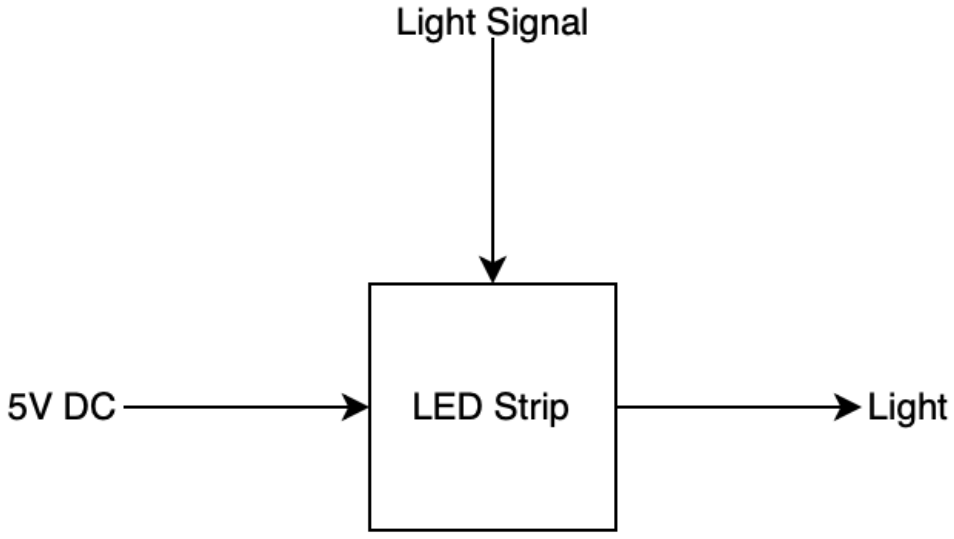

Figure 11: Light Subsystem Diagram

The design process for the light subsystem was straightforward and did not require variations.

Mechanical Subsystem

The mechanical subsystem takes in 5V DC from the power subsystem to the servo motor. The PWM signal from the processor subsystem controls the servo. The servo then outputs movement through the attached hand.

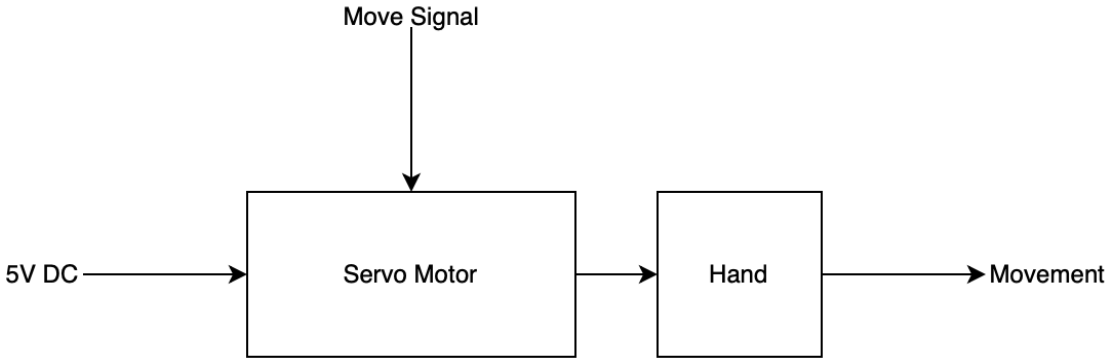

Figure 12: Mechanical Subsystem Diagram

During the design process, we had several iterations of where the servo would be mounted for optimal movement. We initially thought to mount it on the head, on the hand, or on the box, but decided to mount it on an extension protruding from the box so that the hand would have sufficient clearance to move while being able to access the hat.

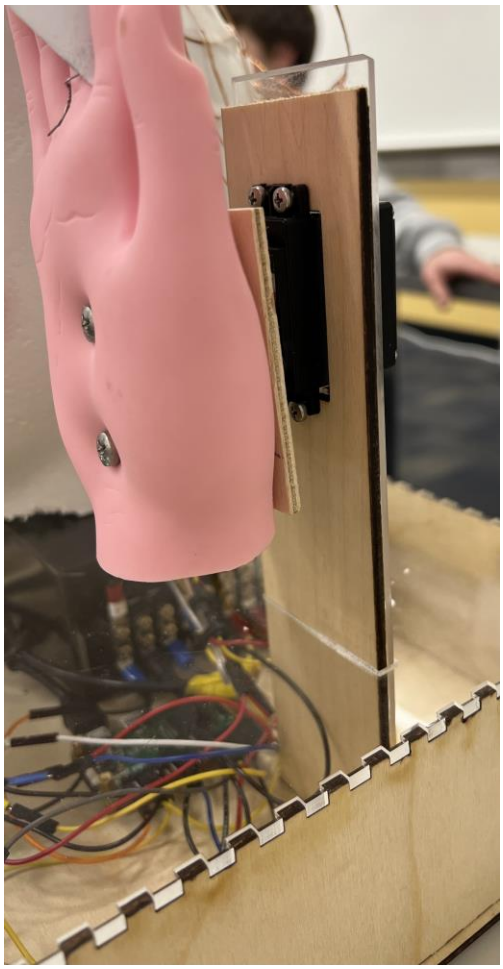

Figure 13: Mounted Servo Motor

Additionally, we had two iterations of our box. The first was entirely made of wood, but for debugging and aesthetic purposes we decided to make the top and one of the walls acrylic. We used a laser cutter to cut and engrave the box components and the holes for mounting our other components.

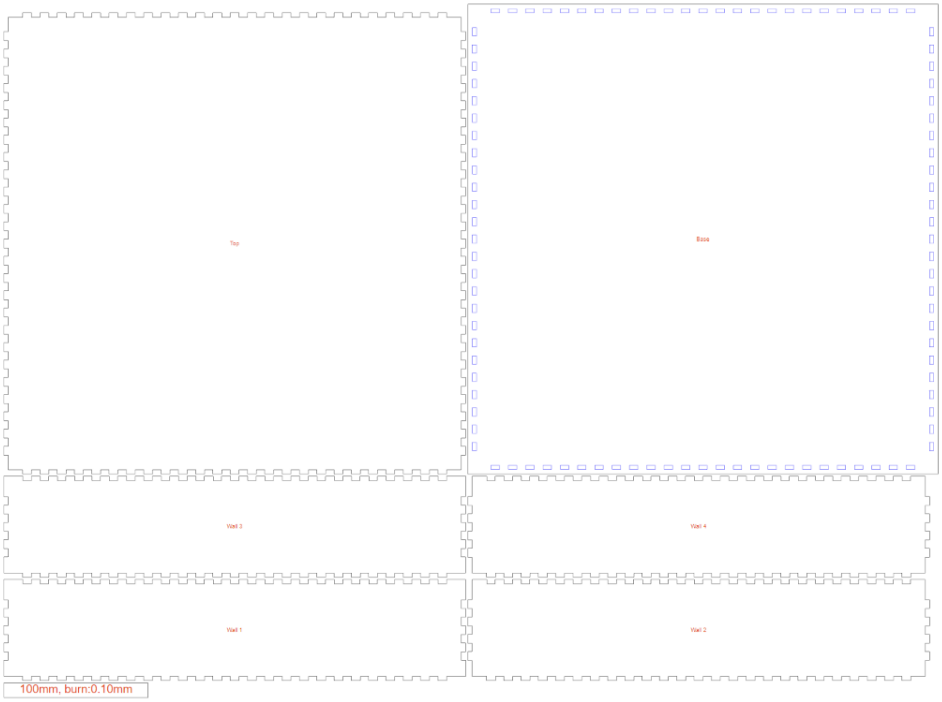

Figure 14: Laser Cutter Files

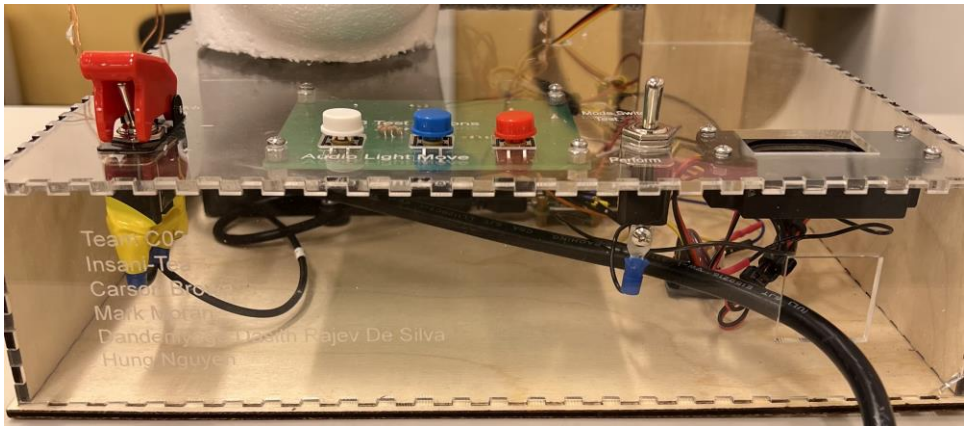

Figure 15. Completed Box

Software Design

Our software design includes 9 different states— off, idle, perform, test, listen, audio/light/move, audio, light, and move shown in Figure 16. Using these states and our state diagram, we designed the software architecture in Figure 17 and Figure 18.

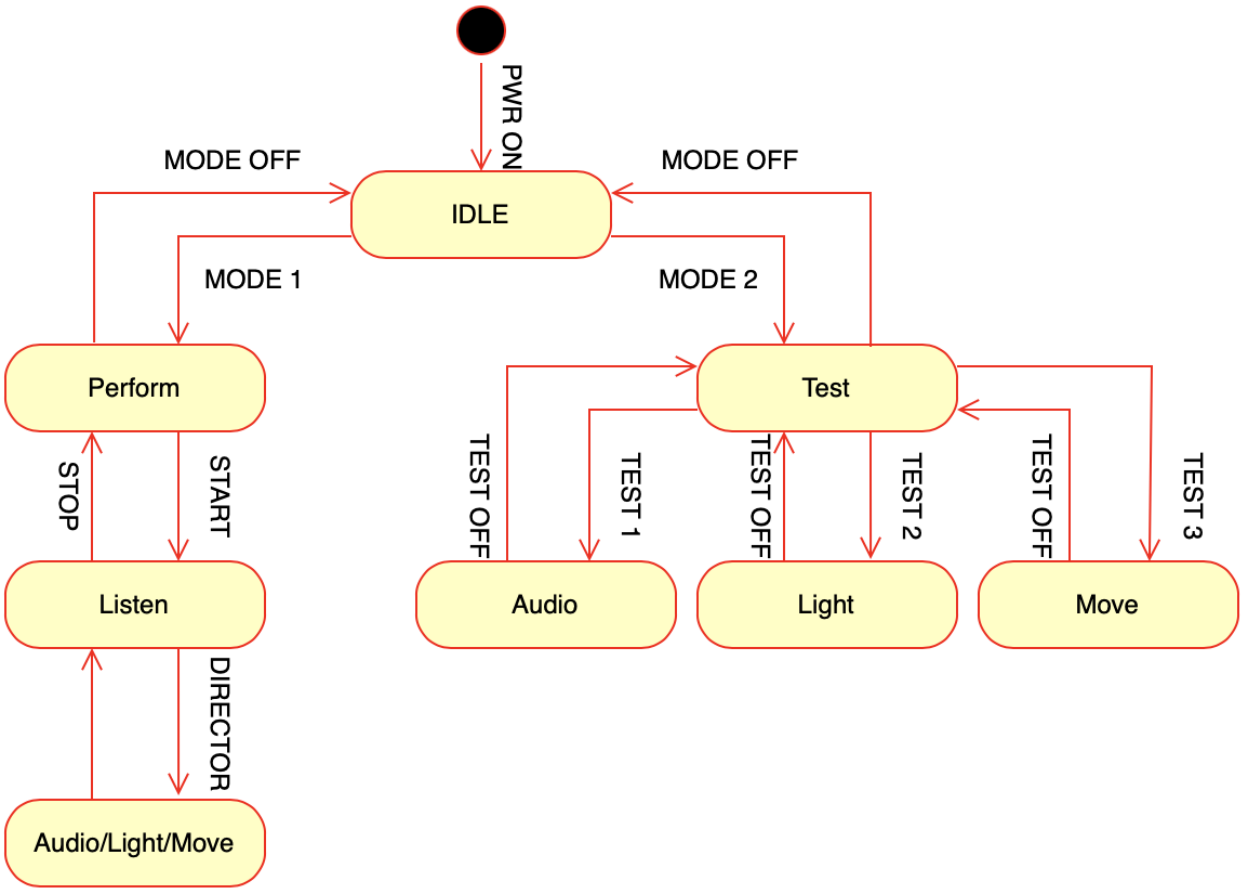

Figure 16: Software state machine with states off, idle, perform, test, listen, audio/light/move, audio, light, and move.

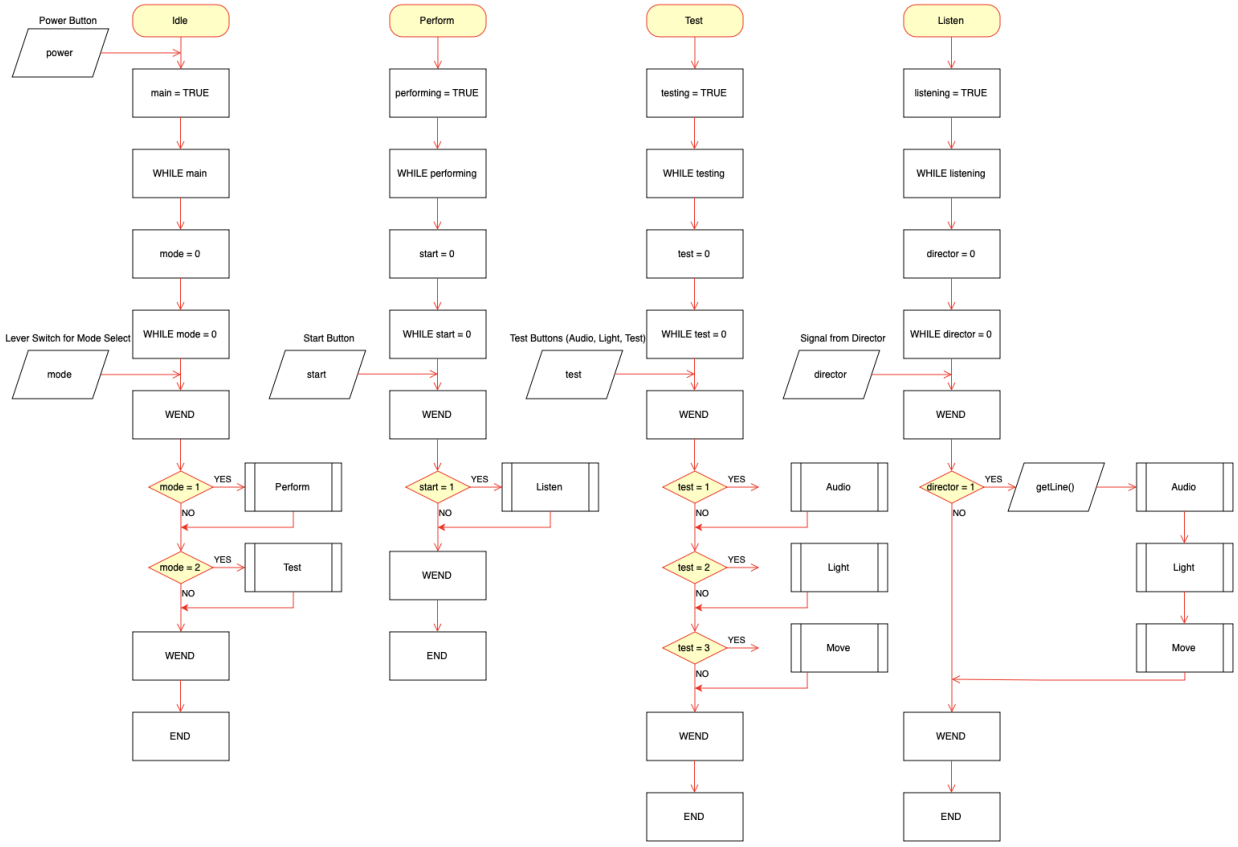

Figure 17: Software flowchart detailing the software architecture for the idle, perform, test, and listen functions.

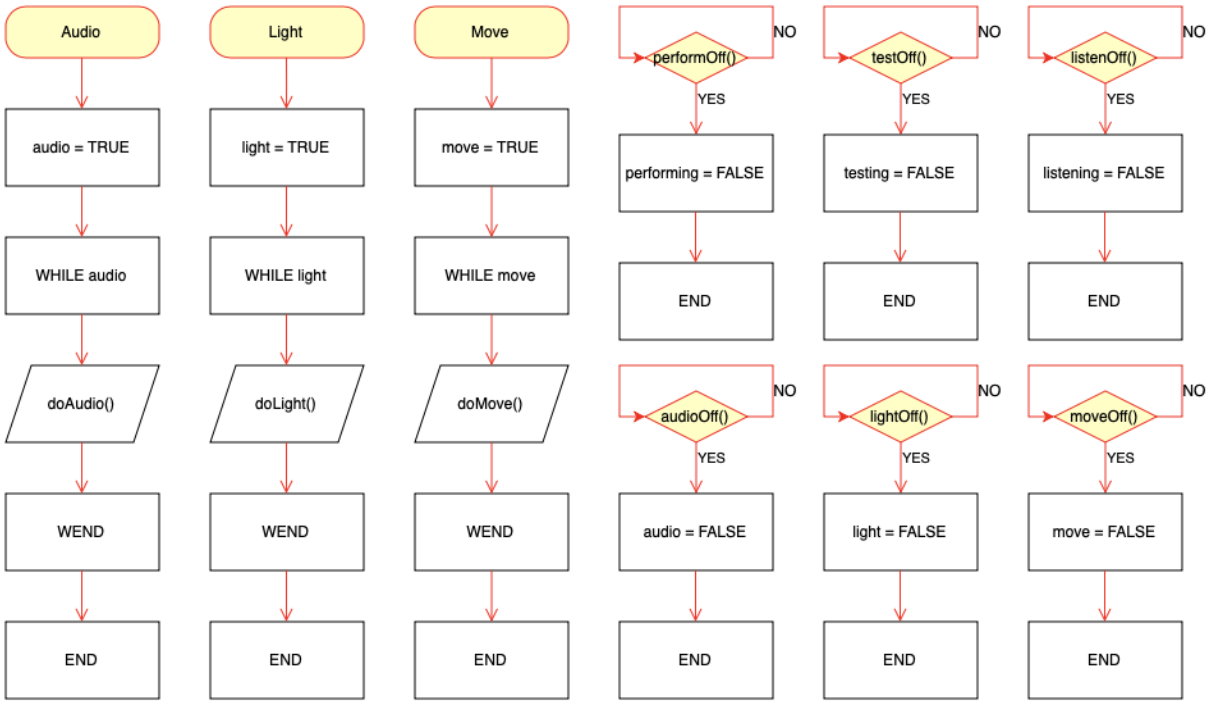

Figure 18. Software flowchart detailing the software architecture for the audio, light, move, and various off functions.

For our software simulation, we tested the functionality of the software architecture with a simulation python script. In our simulation script, we have functions for idle, perform, test, listen, audio, light, move, and various stop functions. Upon running the script, we entered various inputs to control the robot's state and print the corresponding actions. This served as a proof of concept before transitioning to the Raspberry Pi code, and the results were that we found the control logic functions properly and can be implemented in the final software. Sample inputs and outputs can be seen in Figure 19 below.

```
State: off
Waiting for power
0 to stay
1 to power on
Enter power signal: 1

State: idle
Waiting for mode
0 to stay
1 to perform
2 to test
3 to power off
Enter mode: 1

State: perform
Waiting to start
0 to stay
1 for listen
2 to stop performing
Enter start signal: 1

State: listen
Waiting for director
0 to stay
1 for audio/light/move
2 to stop listening
Enter director signal: 1
Enter the line: I am the mad hatter
Line is:  I am the mad hatter

State: audio
I am the mad hatter

State: light
Turning on lights

State: move
Moving hand and hat

State: listen
Waiting for director
0 to stay
1 for audio/light/move
2 to stop listening
Enter director signal: 2
Stopping listening

State: perform
Waiting to start
0 to stay
1 for listen
2 to stop performing
Enter start signal: 2
Perform off

State: idle
Waiting for mode
0 to stay
1 to perform
2 to test
3 to power off
Enter mode: 2

State: test
Waiting for test
0 to stay
1 for audio
2 for light
3 for move
4 to stop testing
Enter test: 1
Enter the line: Would you like some insani-tea
Line is:  Would you like some insani-tea
```

```
State: audio
Would you like some insani-tea

Waiting for audio test to end
0 to continue
1 to end
Enter audio stop signal: 0
Would you like some insani-tea
Enter audio stop signal: 1
Stopping audio test

State: test
Waiting for test
0 to stay
1 for audio
2 for light
3 for move
4 to stop testing
Enter test: 2

State: light
Turning on lights

Waiting for light test to end
0 to continue
1 to end
Enter light stop signal: 1
Stopping light test

State: test
Waiting for test
0 to stay
1 for audio
2 for light
3 for move
4 to stop testing
Enter test: 3

State: move
Moving hand and hat

Waiting for move test to end
0 to continue
1 to end
Enter move stop signal: 1
Stopping move test

State: test
Waiting for test
0 to stay
1 for audio
2 for light
3 for move
4 to stop testing
Enter test: 4
Test off

State: idle
Waiting for mode
0 to stay
1 to perform
2 to test
3 to power off
Enter mode: 3
Powering off

State: off
Waiting for power
0 to stay
1 to power on
```

Figure 19: Software simulation python script with sample inputs and outputs.

For the final software, we wrote a Python script to execute on the Raspberry Pi. This script included audio test, light test, move test, recite, test mode, and perform mode functions. These functions served to control the speaker, LEDs, servo motor, push buttons, and toggle switches. The control logic from our simulations was implemented via signals from the switches and buttons. We also utilized a text to speech library that enabled the robot to read lines from a text file while performing its movement and light effects.