Springtrap Animatronic Project

Introduction

This project is based on *Springtrap*, a character from the indie horror game **Five Nights at Freddy's**. The build focuses on a fully articulated animatronic **head** mounted on a motorized neck. The structure is almost entirely **3D printed**, with servos used to replicate lifelike movements such as jaw motion, eye movement, and head tracking. While the ears are purely cosmetic, the rest of the head is designed to be functional, modular, and serviceable.

The project also evolved beyond simple scripted movement. With the addition of a camera module and OpenCV, Springtrap can now **detect and track objects**, creating a much more unsettling (and satisfying) result.

Project Requirements

Mechanical & Structural

- Fully **3D printed parts**, including the head
- Head and endoskeleton frame are **separate**, secured together with screws
- Strong, heavy **baseplate**
- **10 lb cast iron weight** for stability

Actuation

- **SG90 servos** for eye movement
- **MG995 servos** for jaw and neck (high torque)
- Head functionality:
    - Eyes
    - Eyelids
    - Jaw
- Neck functionality:
    - X-axis rotation
    - Y-axis rotation (dual rods, dual servos)

Electronics & Power

- **Raspberry Pi Zero 2 W**
- **PCA9685 16-channel servo driver HAT**
- **Rechargeable battery power**
- Main power switch
- Dedicated Raspberry Pi **shutdown button**

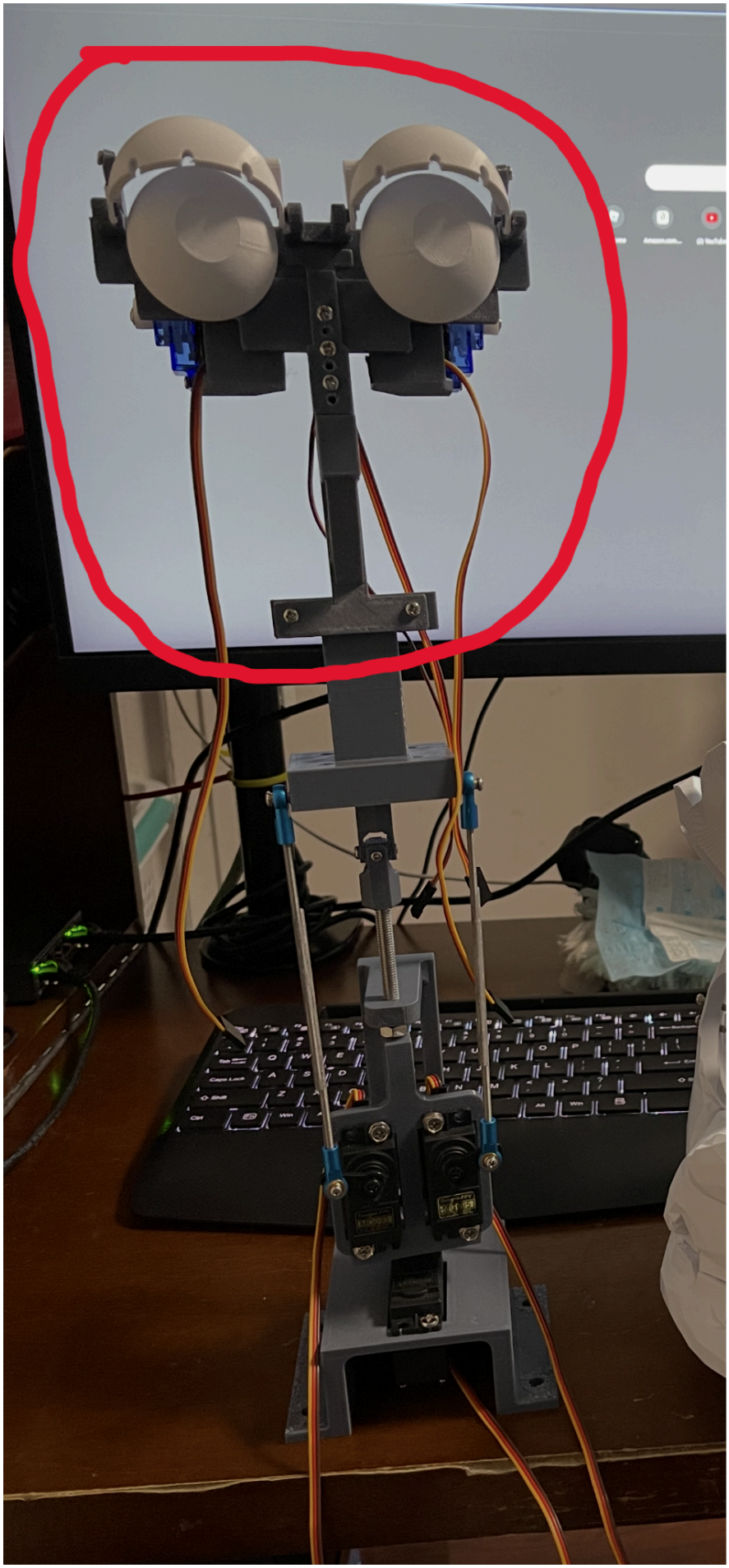
Software

- Make sure the **I2C** serial bus protocol is enabled first
- Servo control using **Python**
- Object detection and tracking using **OpenCV**
- **Arducam Noir IMX219** Pi camera module (new addition)

3D Printing Journey

Endoskeleton Head

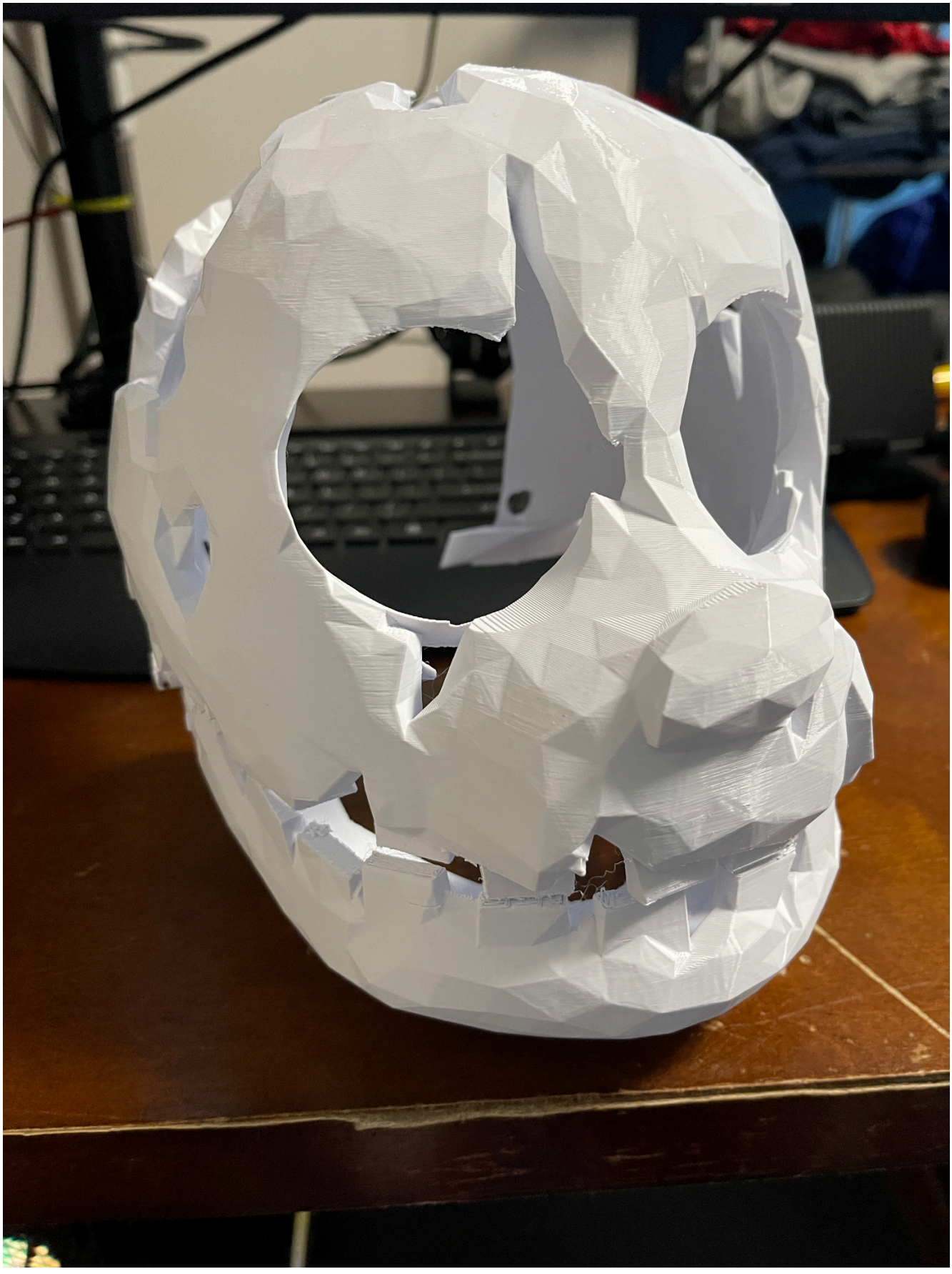
The internal endoskeleton head was printed using **gray PETG filament** for strength and durability. This structure supports the eyes and eyelids while serving as the mounting point for the servos.



Outer Head Shell

The first Springtrap head print (without ears) was primarily used for fitment testing. Attaching the endoskeleton proved harder than expected, and several mounting attempts failed.

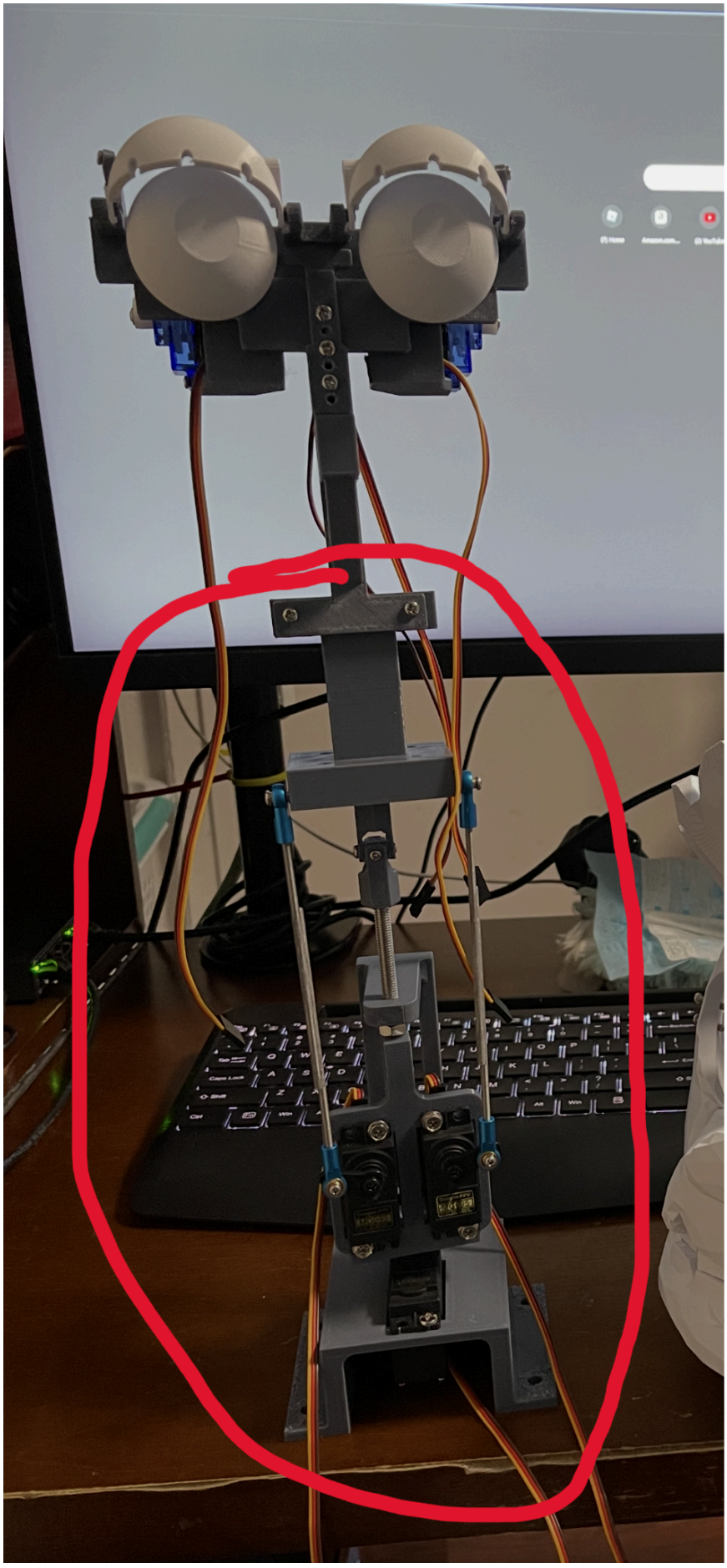
The final solution was to add **two screw mounts on each side of the head**, allowing the endoskeleton to be securely fastened. Due to damage and rough test modifications, both the head and endoskeleton had to be **reprinted**. Also, I mounted the jaw's servo inside the head, as opposed to having it mounted on the endo skeleton.

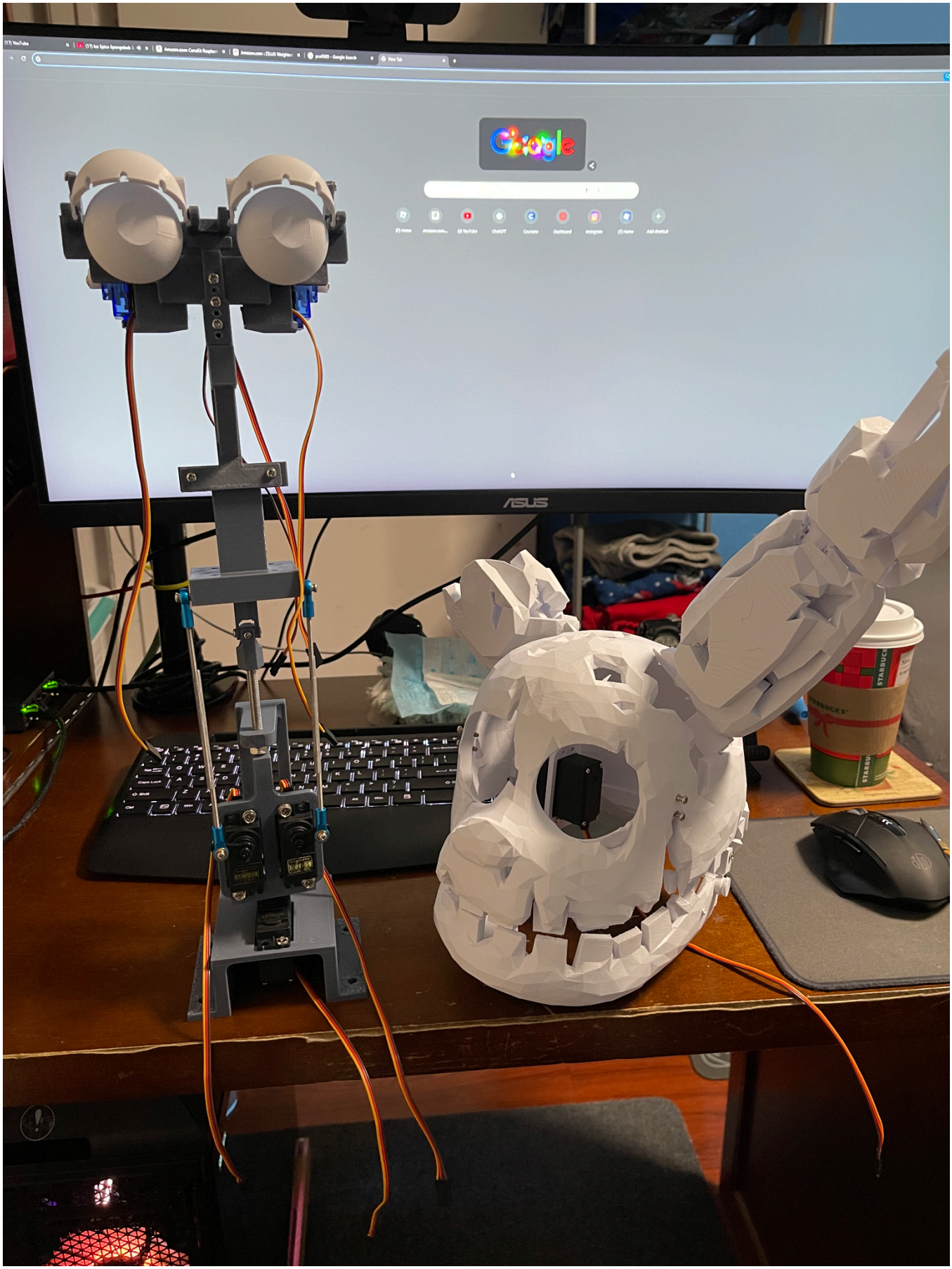


Neck Assembly

To achieve both **X and Y axis movement**, I found a neck design online that closely matched what I needed. A small modification was made so it could properly support the head.

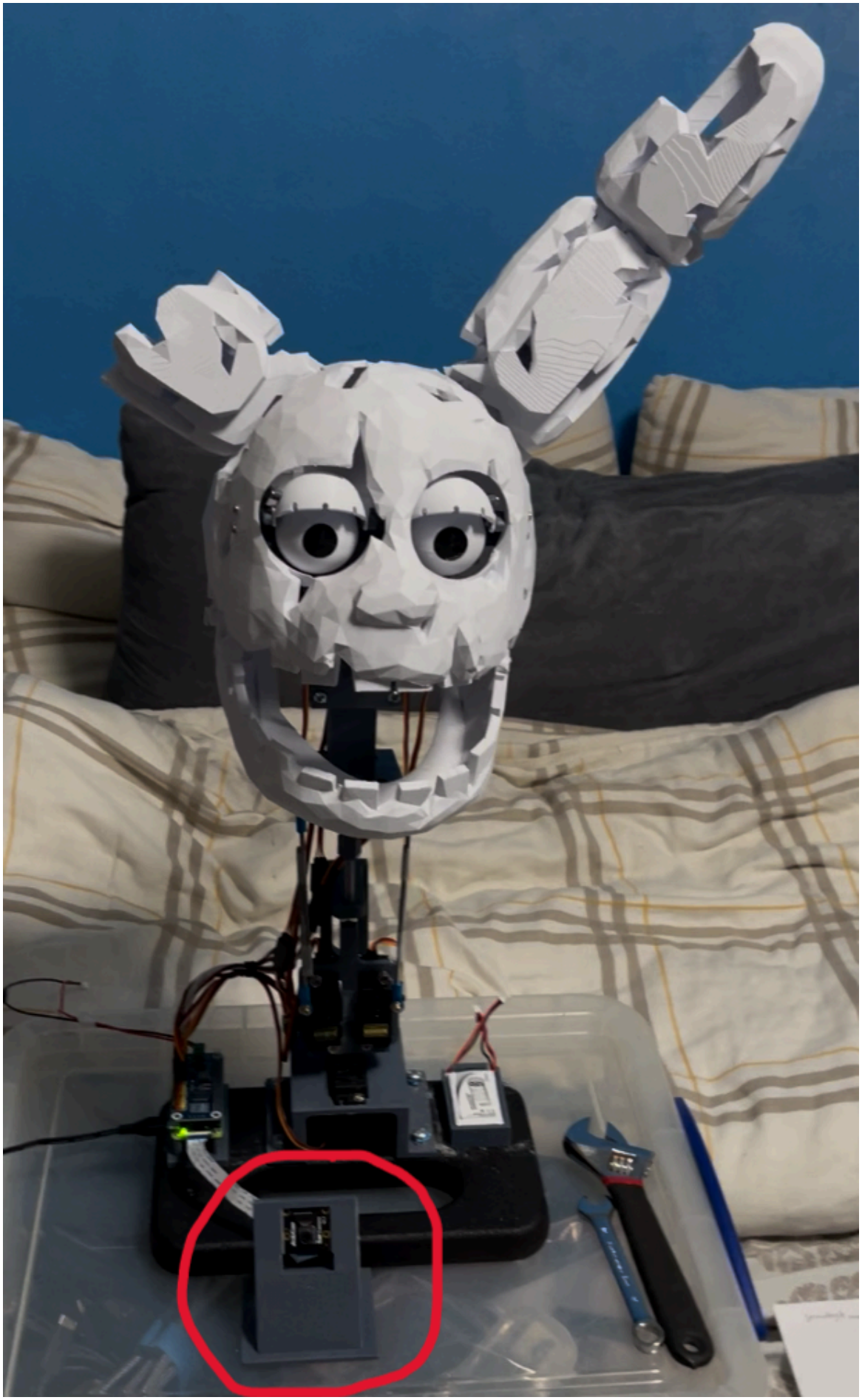
Although the original design recommended a metal universal joint, I opted to **3D print my own** using PETG after struggling to source the correct part. So far, it has held up well.




Google
ASUS
STARBUCKS

IMX219 Noir Camera Stand (New Addition)

To support object detection and tracking, I designed and printed a custom stand for the **Arducam IMX219 Noir camera module**.



Software Setup

Servo Control

A Python virtual environment was created, and **CircuitPython's adafruit_servokit** library was installed to communicate with the PCA9685 servo driver.

Early testing revealed that direct servo commands caused **very fast, aggressive movements**, which risked mechanical damage (and gave Springtrap instant whiplash).

To fix this, I wrote a **custom servo control package** that uses **adafruit_servokit** as a backend but adds:

- Step-based movement
- Adjustable delays between steps

This results in smoother, slower, and more natural-looking motion.

Calibration & CLI Tools

- **helpers.py** contains the **calibration** function, which is used by other scripts to calibrate the servos by moving them to their default positions before any other actions are executed.
- The **CLI.py** script provides a custom command-line interface that allows manual servo control for testing and troubleshooting. For example, typing the command **left face** makes Springtrap turn left by controlling the X-axis servo.

```
oreo-pi@springtrap:~/PythonDir/SpringtrapCodebase $ source CLI.sh
Welcome to the Servos Command Line Interface, you can control the servos/parts via this interface.
Type 1 for help.
Ctrl+c to exit.
>>> 1
Commands:
left face
right face
left incline
right incline
center face
dance
chin down
chin up
chin straight
open jaw
close jaw
jaw half closed
eyes straight
eyes left
eyes right
eye lids normal
eye lids wide
eye lids low
>>>
```

Camera & Object Detection/Tracking (New Addition)

Originally, Springtrap only followed scripted servo loops. This changed with the addition of **object detection and tracking** using a camera module.

I discovered a GitHub project by *CuriousInventors* (the Skellington project) that shared similar goals. While the code was initially complex, extensive review and experimentation allowed me to fully understand and expand upon it.

Problems Encountered

Raspberry Pi OS Changes

- Newer Raspberry Pi OS versions introduced **different camera requirements**
- **libcamera-app** was replaced by **rpicam-app**

Ribbon Cable Mishap

The camera initially refused to work due to the **ribbon cable being inserted incorrectly**. After carefully following Arducam's documentation, the issue was resolved.

OpenCV Camera Access Issues

- **cv2.VideoCapture()** did not work under **Debian 13**
- Solution: switch to the **Picamera2** Python library
- Picamera2 worked flawlessly and integrates cleanly with OpenCV

How Object Detection & Tracking Works

Python Libraries Used

- **Adafruit_servokit** – Servo control via PCA9685
- **cv2 (OpenCV)** – Object detection and contour tracking
- **Picamera2** – Camera access for newer Raspberry Pi OS versions

Detection Logic

1. Camera feed is captured using **Picamera2**
2. Frames are passed into **OpenCV**
3. OpenCV finds **contours** in the image
4. The **largest contour** is selected (e.g. someone walking in camera view)
5. If its area exceeds **400 pixels**, it is treated as a valid target
6. The target's position is remapped to servo angle ranges

Smoothing with EMA

Raw tracking data is noisy and jittery, which is bad for servos. To fix this, an **Exponential Moving Average (EMA)** is applied.

In simple terms:

- EMA blends the new position with the previous position
- This prevents sudden jumps
- Servo motion becomes gradual and more lifelike

The EMA output directly determines the servo angle, giving Springtrap smoother, more natural tracking behavior.

For example, say Springtrap's X-axis servo needs to move from 90° to 160°. Instead of jumping instantly to 160°, the movement is smoothed using the EMA formula inside a loop. Each iteration blends the current servo angle with the target angle, moving the servo about 20% closer to 160° each time. This process repeats until the servo gradually settles at approximately 160°, resulting in smooth, natural motion rather than a sudden snap that'll likely give Springtrap whiplash.

Currently:

- Eyes and neck both track the target
- Control alternates between them using **time-based transitions**

This script runs automatically at startup via **crontab**.

An **important** thing, **if you wish to run CLI.py (or CLI.sh), please make sure that the object tracking script (follower.py) isn't running**; comment out the entry in crontab and reboot.

Power Buttons & Shutdown Safety

The system uses:

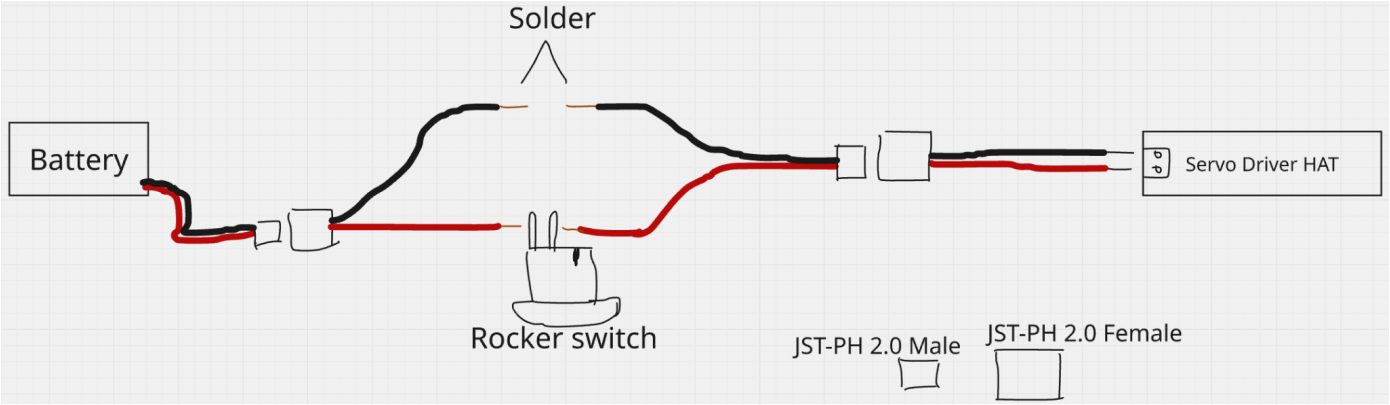
- A **main power switch** for the entire animatronic
- A separate **shutdown button** for the Raspberry Pi

The shutdown button must be pressed **before cutting power** to:

- Prevent SD card corruption
- Ensure all processes terminate cleanly
- Avoid filesystem damage

Only after the Pi has safely shut down should the main power switch be turned off.

Main power switch diagram:



Final Notes

This project started as a simple servo-driven animatronic head and gradually evolved into a much more complex system involving mechanical design, electronics, Linux quirks, and computer vision. The result is a modular, extensible animatronic platform that can continue to grow with more behaviors, sensors, and smarter tracking in the future.

Resources

Original Neck mechanism for animatronics and puppets and YouTube video:
https://www.thingiverse.com/thing:4670841
https://www.youtube.com/watch?v=1NAlPKeRZTM

Original object tracker source code and YouTube video:
https://github.com/curiousinventor/skellington
https://www.youtube.com/watch?v=7m1dUC_UUts

Original Springtrap head:
https://makerworld.com/en/models/522596-five-night-s-at-freddy-s-springtrap-mask#profileId-482156

Picamera2:
https://pypi.org/project/picamera2/0.2.2/

OpenCV:
https://pypi.org/project/opencv-python/

https://pysource.com/2021/01/28/object-tracking-with-opencv-and-python/
https://www.instructables.com/Automatic-Vision-Object-Tracking/

Adafruit-circuitpython-servokit:
https://pypi.org/project/adafruit-circuitpython-servokit/

Github:
https://github.com/JosephG0918/skellington.git
https://github.com/JosephG0918/RPi_Animatronic_CLI_Environment.git