B.3 3D Printed Parts

This project includes several custom 3D printed components that form the mechanical foundation of the pickleball throwing machine, including the launcher, gear train, servo mechanisms, and electronic mounting structures.

All **STL files**, **updated part revisions**, and **print settings** are available in the project's GitHub repository:
(https://github.com/jc2889/OpenPickleballMachine)

Below is a complete list of parts to be printed, including recommended quantities and materials.

B.3.1 3D Printed Part List

File Name	Quantity	Material	Notes
`angle no slide.STL`	2	ASA or PETG	Support frame component
`angle slide.STL`	2	ASA or PETG	Movable guide
`ball disc.STL`	1	ASA or PETG	Feeding mechanism
`ball_mover.STL`	1	ASA or PETG	Rotates ball into firing chamber
`battery strap.STL`	1	ASA or PETG	Battery securing bracket

custom motor gear.STL	2	ASA	Printed PETG in early prototype
custom servo gear.STL	1	ASA	Drives ring gear
custom wheel gear.STL	2	ASA	Printed PETG in early prototype
electrical box.STL	1	ASA or PETG	Houses Raspberry Pi and wiring
left mounting bracket.STL	1	ASA or PETG	Structural support
motor bracket.STL	2	ASA	Holds 775 DC motors
right mounting bracket.STL	1	ASA or PETG	Structural support
ring gear.STL	1	ASA	Controls launch angle via servo gear
roller frame.STL	1	ASA	Frame for launching wheel assembly
screen mount.STL	1	ASA or PETG	Mounts touchscreen to frame
TPU launcher wheel.STL	1	TPU	Pre-printed; provides grip for launch

`tube.STL`	1	ASA or PETG	Ball guide / firing tube

B.3.2 Material Notes

- **ASA** is recommended for parts exposed to sunlight or mechanical load, due to its UV resistance and rigidity.
- **PETG** is acceptable for internal or low-load components.
- **TPU** is used for the **launcher wheel** due to its flexibility and grip.

Please refer to the GitHub documentation for specific **print orientation**, **infill settings**, and **support recommendations**.

B.4 Raspberry Pi Setup (with Software Overview)

B.4.1 Flashing the SD Card

Before setting up the Raspberry Pi, flash the Raspberry Pi OS onto an SD card. For detailed instructions, please refer to the following guide:

- **Flashing Guide:** Official Raspberry Pi Imager Documentation (https://www.raspberrypi.com/documentation/computers/getting-started.html#installing-the-operating-system)

B.4.2 Configuring I2C and Enabling Interfaces

After flashing and booting up your Raspberry Pi, follow these steps to enable I2C communication:

1. **Run Raspberry Pi Configuration:**
   Open a terminal and enter:
   ```
   sudo raspi-config
   ```

2. **Enable I2C:**
   - Navigate to **Interface Options → I2C → Enable**.
   - Exit the configuration tool and reboot if prompted.

3. **Verify I2C is Enabled:**
   After reboot, open the file `/boot/config.txt` by running:
   ```
   sudo nano /boot/config.txt
   ```
   Look for a line like:

```
dtparam=i2c_arm=on
```

If the line is commented out (i.e., starts with a #), remove the #, save the file, and reboot.

4. **Check Wiring:**
    - On a standard Raspberry Pi, **SDA** is physical pin 3 (GPIO 2) and **SCL** is physical pin 5 (GPIO 3).
    - Ensure your PCA9685 module is wired correctly:
        - **SDA** → Pi GPIO 2 (Pin 3)
        - **SCL** → Pi GPIO 3 (Pin 5)
        - Also verify that the **3.3V** and **GND** pins are correctly connected.
5. **Test I2C Communication:**

    Install the i2c-tools if not already installed:

    ```
    sudo apt-get install -y i2c-tools
    ```

    Then run:

    ```
    i2cdetect -y 1
    ```

    This should display the connected I2C devices, including your PCA9685.

B.4.3 Installing Required Libraries

Install the following Python libraries, which are necessary for motor control and UI functionality:

1. **RPI.GPIO:**

   ```
   sudo apt-get install python3-rpi.gpio
   ```

2. **Adafruit CircuitPython ServoKit:**

   ```
   pip3 install adafruit-circuitpython-servokit
   ```

3. **Kivy (for UI development):**

   ```
   sudo apt-get install python3-kivy
   ```

   Alternatively, follow the official Kivy installation instructions if you need a specific version: https://kivy.org/doc/stable/installation/installation.html

B.4.4 Deploying the UI Script

1. **Download the UI File:**

   Ensure that the `UI.py` file (which contains your touchscreen interface code) is downloaded and placed in a known directory (e.g., `/home/pi/`).

2. **Testing the UI:**

   Run the UI manually to confirm it operates correctly:

   ```
   python3 /home/pi/UI.py
   ```

B.4.5 Creating an Autostart .desktop File

To automatically launch the UI on startup:

1. **Create a .desktop File:**

   Open a text editor and paste the following content (adjust paths as necessary):

```
[Desktop Entry]

Version=1.0

Type=Application

Name=Pickleball App

Comment=Launch the Pickleball Control Panel

Exec=/usr/bin/python3 /home/pi/UI.py

Icon=application-default-icon

Terminal=false

Categories=Utility;
```

2. **Save the File:**
   Save it as `PickleballApp.desktop` in the Desktop directory (e.g., `/home/pi/Desktop/`).

3. **Make the .desktop File Executable:**
   Open a terminal and run:
   `chmod +x /home/pi/Desktop/PickleballApp.desktop`

After these steps, on reboot the Raspberry Pi will automatically launch your UI application.

B.4.6 Troubleshooting

- **Package Conflicts or Errors:**
  When installing libraries like `kivy` or `adafruit-circuitpython-servokit`, you may encounter package conflicts or missing dependencies. In some cases, you may

need to uninstall or reinstall Python packages, or use `pip3` with `--break-system-packages` to override system-level restrictions:

```
pip3 install kivy --break-system-packages
```

- **Kivy UI Not Launching:**

  If the touchscreen UI (`UI.py`) doesn't launch:

  - Double-check that the file is executable.
  - Make sure all required Python libraries are installed.
  - Try running it manually in terminal with:
    - `python3 /home/pi/UI.py`

- **I2C Devices Not Detected:**

  If `i2cdetect -y 1` shows no devices:

  - Ensure SDA and SCL are connected to GPIO 2 and GPIO 3 respectively.
  - Re-check that I2C is enabled in `raspi-config`.
  - Confirm that 3.3V and GND are properly connected.

- **Display Not Powering On:**

  If the display causes power issues or the Pi doesn't boot:

  - Make sure the Raspberry Pi is powered through the **step-down converter** rather than the display drawing directly from USB.

- Confirm that the battery voltage is stable and sufficient.

- **General Debugging Tips:**
  - Reboot the Pi after major changes.
  - Use `htop` to check for CPU load if the system freezes.
  - Use `dmesg` or `journalctl -xe` to view system logs for hardware-related errors.