Diving into the Python Scripts

servos_package/__init_.py

Purpose:

As a way to prevent the servo motors from snapping instantly to their target positions, I created my own package, **servos_package**. The main code lives in **__init__.py**, and there's also a **helpers.py** file (which we'll get into later).

Instead of moving the servos directly to their target angles, this package moves them in small steps with short delays between each degree. This creates a smooth, momentum-like motion, making the servos appear slower and more natural.

A bunch of functions have been defined that control the servos. For example, the **full_left_turn()** function.

servos_package/helpers.py

Purpose:

This file contains shared utility functions used across the project. Currently, it defines a single function, **calibration(kit)**.

The **kit** parameter is an instance of **ServoKit**. When called, **calibration()** initializes all servos by moving them to their predefined default positions before any additional actions are executed. This ensures consistent, safe startup behavior and is required to run before other scripts that control servo movement.

CLI.py

Purpose:

Located in the **servos_CLI** directory of the codebase, **CLI.py** serves as the frontend for **servos_package**. As the name suggests, it is a **command-line interface** that allows users to interact with and control the servos directly.

CLI.sh runs the program.

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.

follower.py

Purpose:

Located in the codebase, **follower.py** was a more advanced project of mine, using **object detection and tracking** via a Pi Camera module. It converts the largest detected object's position into servo angles, making it appear as if the animatronic is watching you. Additional lines of code control elements like the eyelids and jaw to make the motion look more lively.

The animatronic periodically switches its focus from tracking with its eyes to moving its neck (X-axis). This transition is time-based, giving a more natural, lifelike motion.

remap() function converts the position of a detected object from camera coordinates into a servo angle.

EMA (Exponential Moving Average) is used extensively in this script. It **smooths out sudden changes in target positions**, giving the servos a sense of momentum. By **gradually adjusting the angles** toward their targets instead of jumping directly, the movement appears more natural and lifelike.

Runs at boot via **crontab**.

shutdown-press-simple.py

Purpose:

I've connected a **push button** to the Raspberry Pi using a couple of GPIO pins. Its sole function is to **safely shut down the Raspberry Pi** when pressed.

Runs at boot via **crontab**.