Technical Module Description: PlaySystemSound.py

Version: 1.0 Date: January 30, 2025

Authors: Henk Stevens, Olaf Mastenbroek, Onno Janssen

Organization: Stichting Oradio

License: GNU GPL

# 1. Purpose and Overview

The `PlaySystemSound.py` module manages the playback of system sounds for the Oradio device. It plays pre-recorded feedback sounds asynchronously while managing the volume levels of other audio streams, such as MPD playback and Spotify Connect. This approach ensures that system notifications are always clearly audible, without interfering with the main music playback experience. Volume levels are temporarily 'ducked' during system sounds, and restored afterward. The sounds played are in depended form the MPD and Spotify as they are “generated” by the alsa sound player aplay. And mixed with the audio sound in the dmix module of the ALSA configuration.

# 2. Key Features

* Asynchronous playback of system sounds using threading.
* Volume ducking: reduces MPD and Spotify volumes during system sounds, then restores them.
* Independent control of system sound volume via ALSA mixer (`VolumeSysSound`).
* Singleton class pattern ensures a single instance of the sound player throughout the system. Insuring that the sound are played in sequence of the calls within the oradio system.
* Batch-ducking support: prevents unnecessary volume restore actions when multiple system sounds play in quick succession.
* Sound selection from a predefined dictionary (`SOUND\_FILES`), supporting user feedback and notifications.
* Standalone test mode for development, including stress testing and custom sound sequence playback.

# 3. System Sound Management

System sounds include feedback such as 'StartUp', 'Stop', 'Play', 'Next', 'USBPresent', and error notifications like 'NoUSB' or 'NoInternet'. The audio files are stored in the directory defined by `SOUND\_FILES\_DIR`. Each sound is mapped to a human-readable key in the `SOUND\_FILES` dictionary, allowing the script to reference them by name.

# 4. Volume Control and Ducking Mechanism

The module reduces the volume of MPD (`VolumeMPD`) and Spotify Connect (`VolumeSpotCon2`) when a system sound plays, using the `amixer` command-line tool. The volumes are lowered to `VOLUME\_MPD\_SYS\_SOUND` and `VOLUME\_SPOTIFY\_SYS\_SOUND`, while the system sound volume is controlled independently at `DEFAULT\_SYS\_SOUND\_VOLUME`.

When multiple system sounds are played in quick succession, the batch-ducking logic prevents premature restoration of the music volume. Volume restoration is delayed by 0.5 seconds after the last system sound to avoid rapid volume changes.

# 5. Standalone Use and Testing

The `PlaySystemSound.py` module includes a standalone test mode that allows developers to verify the playback and volume ducking behavior without requiring the full Oradio system environment. This mode is activated when running the script directly as the main program (`python3 PlaySystemSound.py`).

## Standalone Test Options

* Play individual system sounds by selecting their corresponding number from the menu.
* Perform a stress test: randomly selects and plays sounds over a short duration using multiple concurrent threads to test the robustness of the ducking and restore mechanisms.
* Run a custom sequence test: lets the user specify a sequence of five system sounds to play back-to-back for testing batch volume ducking and restoration logic.

## Example Test Interaction

Example test commands when running the script:  
- `0`: Quit the test program.  
- `1–N`: Play a specific system sound from the list.  
- `99`: Run the stress test.  
- `100`: Specify a custom sequence of sound numbers for batch testing.

This standalone functionality is crucial for verifying both the asynchronous playback logic and the volume ducking feature, helping ensure reliable system behavior before deploying updates into the production Oradio control script.