Module: Integrating Google Assistant with Raspberry Pi

Current online resources and guides predominantly utilise the `google-assistant-library`, which Google deprecated in 2019. This library was replaced by conversational actions, which were subsequently deprecated in June 2023.

For our implementation of Google Assistant, we relied on the deprecated libraries and employed several workaround packaging techniques to avoid significant breakdowns in the assistant library's functionality. While this approach ensures basic operability, it constrains the available features.
Moreover, the Google Assistant SDK now supports only `x86_64` and `ARMv7` architectures, thus limiting use to Raspberry Pi models 3 and 4. Implementing the library even on ARMv7-compatible devices has posed additional challenges.

At present, our device responds to the command `googlesamples-assistant-pushtotalk`; however, manual intervention is required to start and stop the assistant as it lacks the wake word functionality ("Ok Google" or "Hey Google").

Key Implementation Steps

1. **Audio Configuration**: A USB 2.0 MIC Adapter was used as the input, with a 3.5mm jack for output. The audio configuration was defined in a `.asoundrc` file in the home directory (`/home/pi`), and volume adjustments were made using `alsamixer`.
2. **Google Actions Console Setup**: We created a new project on the Google Actions Console and attempted to register the Raspberry Pi device. While downloading the OAuth client secret, we encountered an error but generated a new OAuth client secret on the API page as a workaround.
3. **Python Environment Configuration and Authorization**: The Python environment was set up on Raspberry Pi 4, followed by OAuth authorization.

Python Environment Setup Command

1. sudo apt-get install portaudio19-dev libffi-dev libssl-dev
2. pip install google-assistant-sdk[samples]
3. pip install --upgrade google-auth-oauthlib[tool]
4. pip install pyaudio requests urllib3 six

OAuth Authorization for Raspberry Pi 4
To initiate OAuth authorization, the following command was used:

google-oauthlib-tool --scope https://www.googleapis.com/auth/assistant-sdk-prototype --save –headless --client-secrets /home/khushwant/Desktop/rpi/client_secret_270922874153-jo07dgak4rcrko7vdfc90q8is4lpim0e.apps.googleusercontent.com.json

But `--headless` option has been deprecated in OAuth2.0, we used VNC Viewer to authorise the Raspberry Pi directly through its browser. However, this led to an "Error 400: invalid_request" and device's non-compliance with OAuth 2.0 policies.

To resolve this, in the same project on Google Console we generated a new OAuth client secret for a web application, authorised the device on another Ubuntu system, and transferred the resulting credentials to the Raspberry Pi and it somehow worked.

Code Execution Steps

1. **Audio Test**: Running `googlesamples-assistant-audiotest --record-time 10` resulted in an error due to the deprecated `toarray()` function. We resolved this by replacing `toarray()` with `tobytes()` present in lib/python3.12/site-packages/googlesamples/assistant/grpc/audio_helpers.py
2. **Push-to-Talk Command Execution**: Executing `googlesamples-assistant-pushtotalk --project-id [project-id] --device-model-id [device-model-id]` led to protocol buffer errors. Attempts to resolve this by installing alternative `protobuf` versions were unsuccessful resulting in more protocol buffer errors ; however, setting the environment variable `PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python` mitigated some issues. Additionally, an asynchronous deprecation error was resolved by installing `tenacity==8.0.1`. With these adjustments, `googlesamples-assistant-pushtotalk` now functions on our device.

While `googlesamples-assistant-pushtotalk` works partially on the Raspberry Pi 4, the `googlesamples-assistant-hotword` feature has been fully deprecated, preventing activation via the "Ok Google" wake word. Alternative implementations using Snowboy and Pico Voice, as suggested on several forums, were explored. Unfortunately, Snowboy is outdated, it's been more than 6 years since the last commit on Snowboy, and it's hard to get the code to even run now.

Pico Voice Integration for Hotword Detection We attempted to incorporate hotword detection using Pico Voice for the "Ok Google" command. However, this feature is currently limited to `x86_64` machines, with the Raspberry Pi experiencing issues such as unresponsiveness and segmentation faults during detection.Pico Voice hotword detection was successfully implemented on an Ubuntu `x86_64` machine but failed to operate on the Raspberry Pi. According to the Pico Voice forums, ARM builds require a business partnership to access. Even on Ubuntu, the detection was slower and less accurate than Google Assistant, particularly for non-native accents, despite using a custom-trained hotword detection file for "Ok Google" generated by PicoVoice.

Native Python-based Speech Recognition was also tested for hotword detection but proved significantly less accurate than Pico Voice.