Project : Smart Helmet

KHUSHWANT KASWAN – 241110035 – GROUP 10

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 techniques to avoid significant breakdowns in the assistant library's functionality. While this approach ensures basic operability, it constrains the available features.

At present, our device responds to the command “Google”...executes google assistant and “Stop” to terminate the instance but the accuracy of hotword detection is not as good as google’s functionality ("Ok Google" or "Hey Google").

Key Implementation Steps

1. **Audio Configuration of Pi :** USB mic and 3.5mm earphone
2. **Google Actions Console Project Setup**:Created Project, Registered the device, Enabled Assistant API and downloaded the OAuth client secret.
   (While downloading the OAuth client secret, we encountered an error but generated a new OAuth client secret from the API enable page as a workaround.)
3. **Python Environment Configuration and Device Authorization**
4. **Integrating with Manual Hotword Detection Code**

Code Execution Issues/Debugging

1. **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.json

   Issue : `--headless` option has been deprecated, tried using VNC Viewer to authorise the Raspberry Pi directly through its browser but encountered another authorization error. To resolve this, for the same project on Google Console we generated a new OAuth client secret for a web application, authorised the device on my personal Ubuntu system, transferred the generated credentials.json to the Raspberry Pi home/.config/google-oauthlib/ directory and it somehow worked.

2. **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 …assistant/grpc/audio_helpers.py

3. **Push-to-Talk Command Execution**: Executing "`googlesamples-assistant-pushtotalk`" led to protocol buffer errors. Attempts to resolve this by installing alternative `protobuf` versions were unsuccessful resulting in more errors ; however, setting the environment variable `PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=python` mitigated these issues. Additionally, an asynchronous deprecation error was resolved by installing `tenacity==8.0.1`.
4. "`googlesamples-assistant-hotword`" has been fully deprecated, meaning NO activation via the "Ok Google" wake word.
   Explored alternative implementations using Snowboy and Pico Voice, as suggested on several forums. Unfortunately, Snowboy is also deprecated and it's hard to get its 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 only.According to the Pico Voice forums, ARM builds require a business partnership to access. Pico Voice hotword detection was successfully implemented on my Ubuntu machine but failed to operate on the Raspberry Pi. Experienced unresponsiveness and segmentation faults. Even on Ubuntu, the detection was slower and less accurate than native Google Assistant despite using a custom-trained hotword detection file for "Google" generated by PicoVoice. So it didn't work on Raspberry Pi.

Currently we have implemented Native Python-based Speech Recognition that listens continuously and detects the words "GOOGLE" and "STOP". Although the hotword detection worked but proved significantly less accurate than Pico Voice. Sometimes we will have to say the words multiple times but once activated our assistant works and Responds to STOP which is running as a parallel thread and terminates the assistant.

Execution Flow :
The module is demonstrated by executing the script.sh file present at home/pi/khush/. Instructions appear as we run the script.

Assistant will start only after you say "Google". On successful detection you will hear "Activated" and can start the conversation. To stop the assistant just say "STOP" and will hear "Deactivated" upon successful detection. If not, repeat "STOP" again.
The whole process can be started again just by saying "GOOGLE".....

Unfortunately the assistant cannot be merged with other modules because of the protocol buffer, deprecated libraries we have used to make it work.