DOCUMENTATION

SETUP

The system needs both Python 2.7 and Python 3. Python 2.7 is used for the robot connection code (`body.py`), and Python 3 is used for the voice processing code (`sidecar.py`). Make sure both are installed.

You also need the NAOqi SDK from SoftBank Robotics. Download it, extract it and add its Python 2.7 folder to your PYTHONPATH so the code can find the robot libraries. This lets the system talk to the NAO robot.

Install Python packages for both versions. For Python 2, install Flask with `pip2 install flask`. For Python 3, install several packages: requests, numpy, pyaudio, soundfile, faster-whisper, google-genai, and optionally scikit-learn, seaborn, and matplotlib. These handle audio recording, speech-to-text and AI responses.

You may need system audio libraries. On Mac, install PortAudio with Homebrew. On Linux, install the portaudio development package. On Windows, download a pre-built PyAudio package since building it can be difficult.

Configure three settings. First, set the robot's IP address in `src/body.py` to match your robot's network address. Second, set the microphone device number in `src/sidecar.py` by checking which device your microphone is. Third, add your Google AI API key in `src/format.py` to enable AI-generated responses.

You can also adjust the speech recognition model size. Smaller models are faster but less accurate; larger models are more accurate but slower. Choose based on your computer's speed and accuracy needs.

Before running everything, test each part. Check that you can connect to the robot, that your microphone works and that all Python packages are installed correctly. Once everything is set up, you can start the system with `run.py`, which runs both the robot server and the voice processing together.

ARCHITECTURE

A. Tools and Frameworks

• NAOqi with Custom Modules — Provides direct, low-latency hardware access for audio control. Custom ALModules handle audio capture (ALAudioDevice) and wake-word detection (ALSpeechRecognition).
• Flask REST API — Runs onboard NAO (body.py) to provide robot services via REST endpoints (/talk, /wave_hand, /bow). Ensures easy integration with Python 3 processing and simplified debugging using HTTP tools.
• faster whisper — Executes CPU optimized STT with Whisper Small using CTranslate2. The model is loaded once and reused to minimize overhead.
• Google Gemini Flash 2.5 API — Handles response generation with fast and cloud-hosted LLM inference.
• PyAudio — Captures speech using an external USB microphone for improved signal quality and configurable sampling. Uses 6-second chunks for complete audio capture
• kb.json — JSON knowledge base storing target-domain facts like locations, hours and contacts. Loaded in-memory for deterministic and hallucination-resistant responses.

B. Design Decisions and Rationale

• Deterministic Knowledge Retrieval — All factual queries are resolved using structured JSON lookup prior to LLM usage, ensuring correctness.
• LLM for Response phrasing — The LLM is restricted to generating phrased replies enabling natural and human-aligned responses.
• Intent Driven Gestures — A direct intent-hardware trigger mapping like wave for greeting and bow for closure enhances engagement.
• Real Time Latency — End-to-end and per-component latency (STT, reasoning, TTS, total) are logged to CSV to support optimization.

C. System Data Flow

User speech is captured via an external microphone, transcribed using Speech-to-Text with the Whisper Small model,processed through rule-based intent reasoning using deterministic lookup from kb.json, generating a response using the Gemini Flash 2.5 LLM and finally spoken by the NAO robot with synchronized gesture.

USAGE

To start the system, run `python3 run.py` from the project folder. This starts both the robot connection and the voice processing. You can also run them separately if needed.

Once running, the system listens to your microphone and records 6-second chunks. It transcribes your speech, finds out what you're asking, looks up information in the knowledge base,and has the robot respond with speech and gestures.

The robot behaves differently based on what you ask. If you say "hello" or "hi", it waves and greets you. If you ask where something is, like "where is the robotics lab", it tells you the location. If you ask about a professor, it gives you their office and email. When you say goodbye or thank you, it bows. For other questions, it gives a general receptionist-style answer.

You can also control the robot directly through web requests. Send text for the robot to speak, make it wave, or make it bow. This is useful for testing or connecting to other programs.

The system automatically tracks how fast it responds. It records how long each step takes listening, understanding, finding information, and speaking and saves this to a file so you can check performance later.

To customize the system, edit the knowledge base file to add new rooms, labs, or contacts. The system will use this information when answering questions. You can also change how it recognizes different types of questions or adjust audio settings to fit your needs.