RUB

RUHR-UNIVERSITÄT BOCHUM

EMBEDDED MULTIMEDIA: HEY POLA

Developement of a Voice-Assistant for the RaspberryPi

Philipp Intek, Sebastián Gómez, Aldo Márquez, David Viracachá, Ahmad Aljeniyat und Maurice Brosch

Table of contents

1. Introduction
2. Development Process
3. Software-Architecture
4. Software Modules
5. Product Presentation
6. Future Work

Introduction

Introduction:

- In recent years, many voice assistants have popped up in the consumer electronics market
- Google Home, Amazon's Alexa and Apple's Siri are used in millions of households
- Need for a lightweight, portable and responsive voice assistant
- Boom of AI LLMs enables this approach
- Benefits users that are visually impaired, have hearing loss as well as user that are aware on the topic of data privacy

Development Process: Description

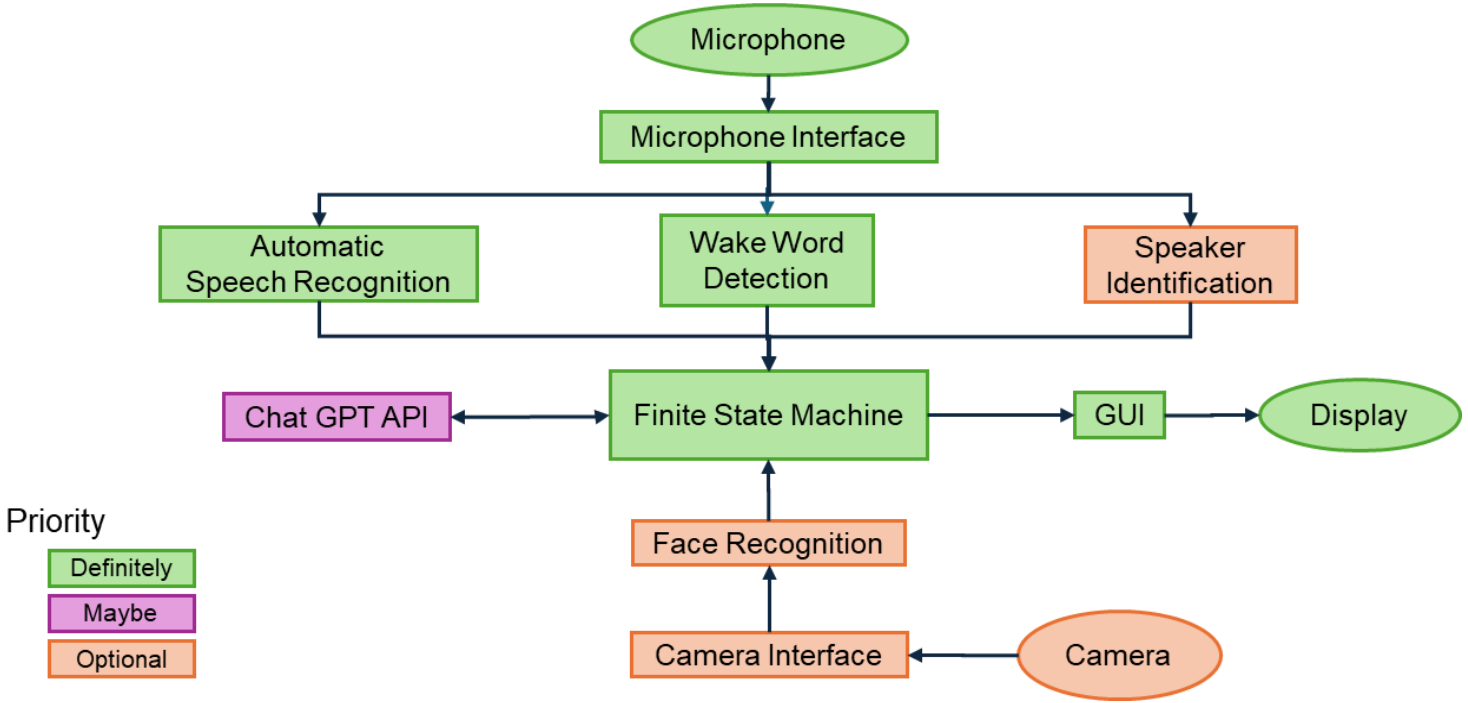
- Voice Assistant
- Activated through specific "wake word" (Hey Pola)
- Capable of transcribing spoken words
- Makes use of LLM to answer user questions
- Questions are transmitted through text and audio

Development Process

Development Process

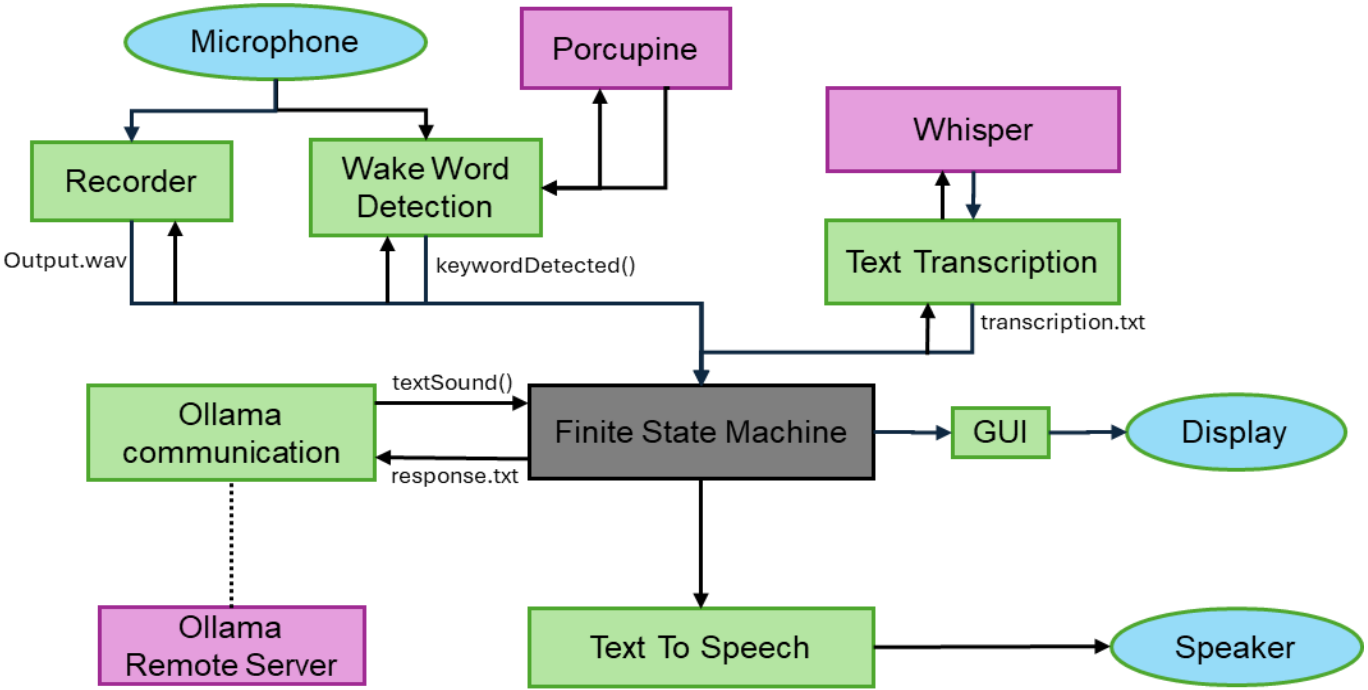
- Identified the different potential use cases for our product
- Set development goals and their priorities
- Defined technical and non-technical requirements
- Team-member responsibilities were chosen based on individual strengths and interests
- Built a modular architecture based on a developed concept top-level diagram
- Weekly team meetings and regular correspondence with team-members
- Individually developed modules were integrated into a final system and tested

Development Process: Proposed Block Diagram



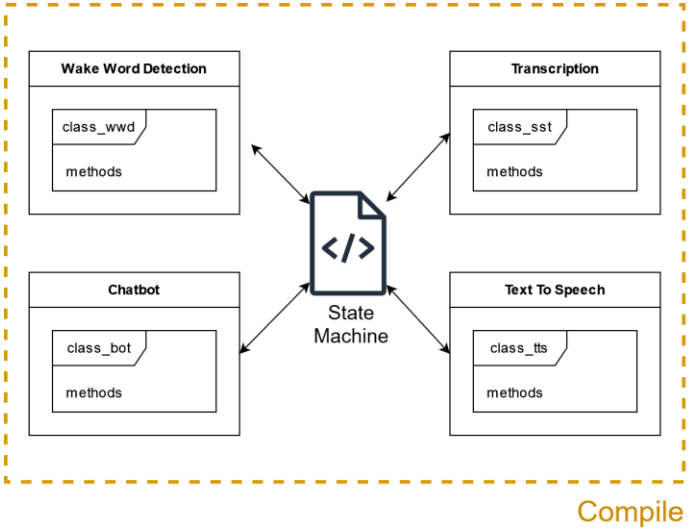
Software Architecture

Software Architecture: Implemented Block Diagram

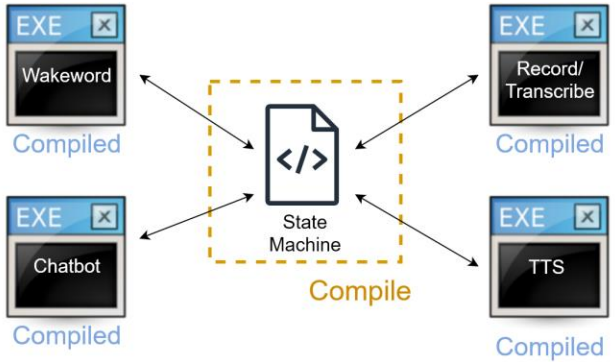


Software Architecture

- The central management program is the finite state machine, which handles the sub-programs
- Two possible types of code linkage:



+ Easier to emit signals from anywhere

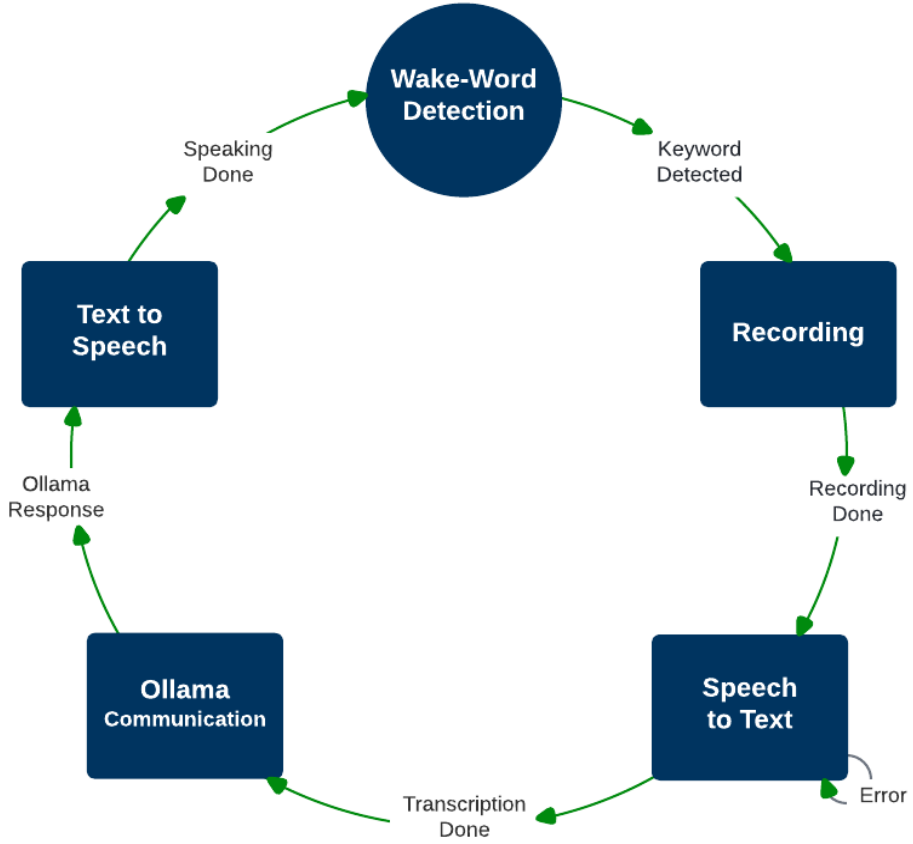


+ Better for parallel development

Software Modules

Software Modules

- Graphical User Interface
- Speech Recognition/ Wake Word Detection
- Speech Recording
- Speech Transcription
- Chat-Bot (Ollama Models)
- Text to Speech


Wake-Word Detection
Keyword Detected
Recording
Recording Done
Speech to Text
Error
Transcription Done
Ollama Communication
Ollama Response
Text to Speech
Speaking Done

Software Modules: Graphical User Interface

- Based on QML (Qt Modelling Language)
- GUI program is linked to the finite state machine

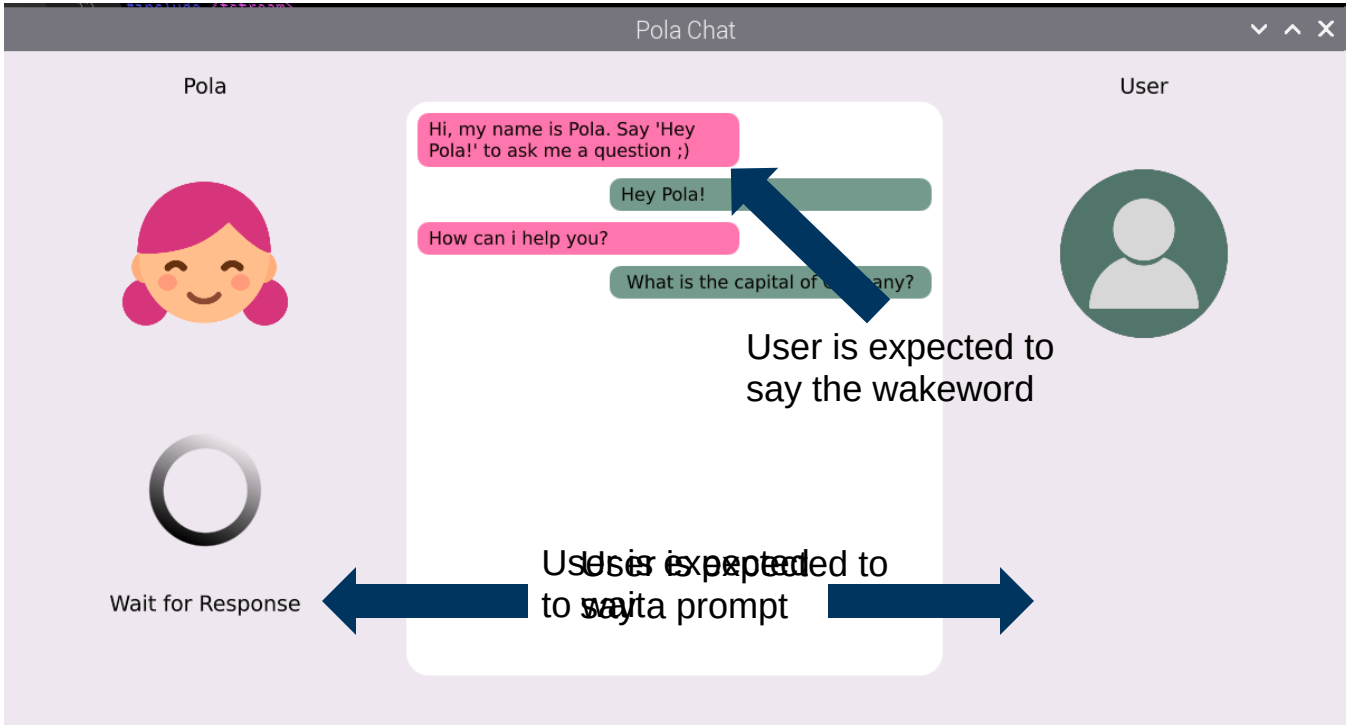
Necessary features:

- None: The fully acoustic assistant would technically be usable without a graphical interface

Wanted features:

- Intuitive information about the current state
- What is currently expected from the user?
- Visual representation of the conversation

Graphical User Interface: Design Choice



Software Modules: Wake Word Detection

- Requirements
  - Always on microphone
  - Circular memory buffer
  - Signalizes when a certain sound is recognized
- Challenges and options
  - Algorithmic and ML approaches
  - How to manage it through time?
- Solution: Use of the Porcupine library

Software Modules: Speech Recording

- Requirements
  - Start capturing by user instruction
  - Save on good quality
  - Run until voice command is ended
- Challenges and options
  - Audio recording duration
  - Format and sample rate

- Solution: Qt QAudioSource Library / Energy level analysis
- WAV Audio format – 16kHz sample rate

Software Modules: Speech Transcription

- Requirements
  - Read Audio Recording
  - Convert Speech to text
  - Language of transcription
- Challenges and options
  - Proper time/accurate processing balance
- Solution: C++ Faster-Whisper Model
- Save transcription in a .txt File

Model	Size
"tiny"	~1GB
"base"	~2GB
"small"	~5GB
"medium"	~13GB
"large"	~26GB

Software Modules: Chat-Bot (OLLAMA)

- Requirements
  - Communicate with remote LLM
  - Send and receive desired text

- Challenges and options
  - Proper time/accurate processing balance

- Solution: JSON data over Python request

Software Modules: Text to Speech

- Requirements
  - Sound reproduction from a given text
  - Seamless reproduction of corresponding sound
- Challenges and options
  - Incompatibilities with design environment
  - Out of the box thinking
  - Correct timing and processing
- Solution: Console implementation

Product Demo

Future Work

Future Work: Corrections

Better use of the Ollama API

- Make use of in-conversational context and system-wide prompts

Ease of use

- Better integration through standardized installation processes
- Only-voice setup

Error handling

- Logging capabilities
- The addition of an error state
- Error explanation to the user

Future Work: Product improvement

Different keywords for various functions

- Instead of "Chatbot", "Home assistant" can be invoked to provide sensor data.

Speaker recognition

- Access control through facial recognition or voice recognition.

GUI enhancements

- Display whether the microphone is detected.
- Visualization of input volume.
- Improved fonts and a more attractive design with additional animations.

GUI enhancements

- Control of lights, thermostats, and other IoT devices.