Alexa Pi Emulator

Anca Ilienescu, Remus Jeberean, Politehnica University of Timisoara May, 2018

1 Repository

`https://github.com/ankies07/AlexaPi`

2 User requirements

1. Easy setup/installation process.
2. Wake-up word "Alexa".
3. Basic Alexa functionalities.
4. Open for extension eg. adding user-specific functionalities.
5. LED notification indicating the state of the system.

3 System overview

The overview of the system is depicted in Figure 1.

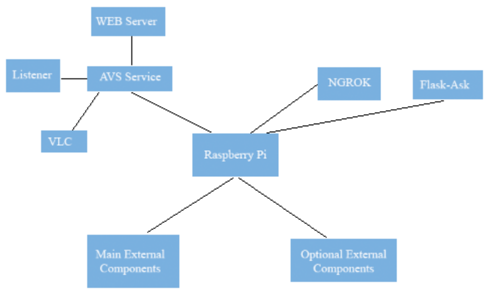

Figure 1: System overview diagram

The entire system runs on the Raspberry Pi.

The required external components are comprised of a pair of speakers and a microphone.

Extensions to the system come under the form of Alexa's Skills. Skills are for Alexa what apps are for iOS or Android devices.

The system runs automatically on startup and requires no further input from the user outside of the first-time setup.

The microphone listens for the wake word "Alexa" and sends the voice input over the Cloud to the Amazon servers, where it is processed. A response is then received and played to the user through the speakers.

The AVS Service runs on startup and handles all Alexa related requests. It uses a small web server to generate the authentication tokens. It also uses a local listener (Pocketsphinx, an open-source engine) to listen for the word “Alexa” in order to start sending requests to the Alexa Voice Service. Responses are played back using VLC media player.

NGROK is a command-line program that opens a secure tunnel to localhost and exposes that tunnel behind an HTTPS endpoint, allowing Alexa to talk to the code on the Pi right away.

Flask-Ask is a Flask extension that helps building Alexa skills, making requests and responses easy to manage. The code using Flask-Ask will run on a while True loop, waiting for requests from the defined Intents received through NGROK. This code runs locally, on the Pi, not on the Amazon servers. This means that any action triggered by Alexa voice commands happens on the Pi, and not on the Amazon servers, allowing for more functional and personal scripts to be run.

Main external components are the two LEDs and the button we may use for this system itself, while the optional ones represent anything else connected to GPIO.

4 Circuit design

The hardware view of the system is depicted below.

The Raspbery Pi 3 comes with everything we need on a fresh Raspbian install.

The only hardware components further required are speakers and a microphone. Any can be used, as long as they can connect to the Pi. This is most easily done with a USB to 3.5 jack adapter.

The wiring between the components can be seen in the figure above. It is important to note that, unless changed in the config file, the LEDs must be connected to GPIO 24 and 25, and the button to 18.

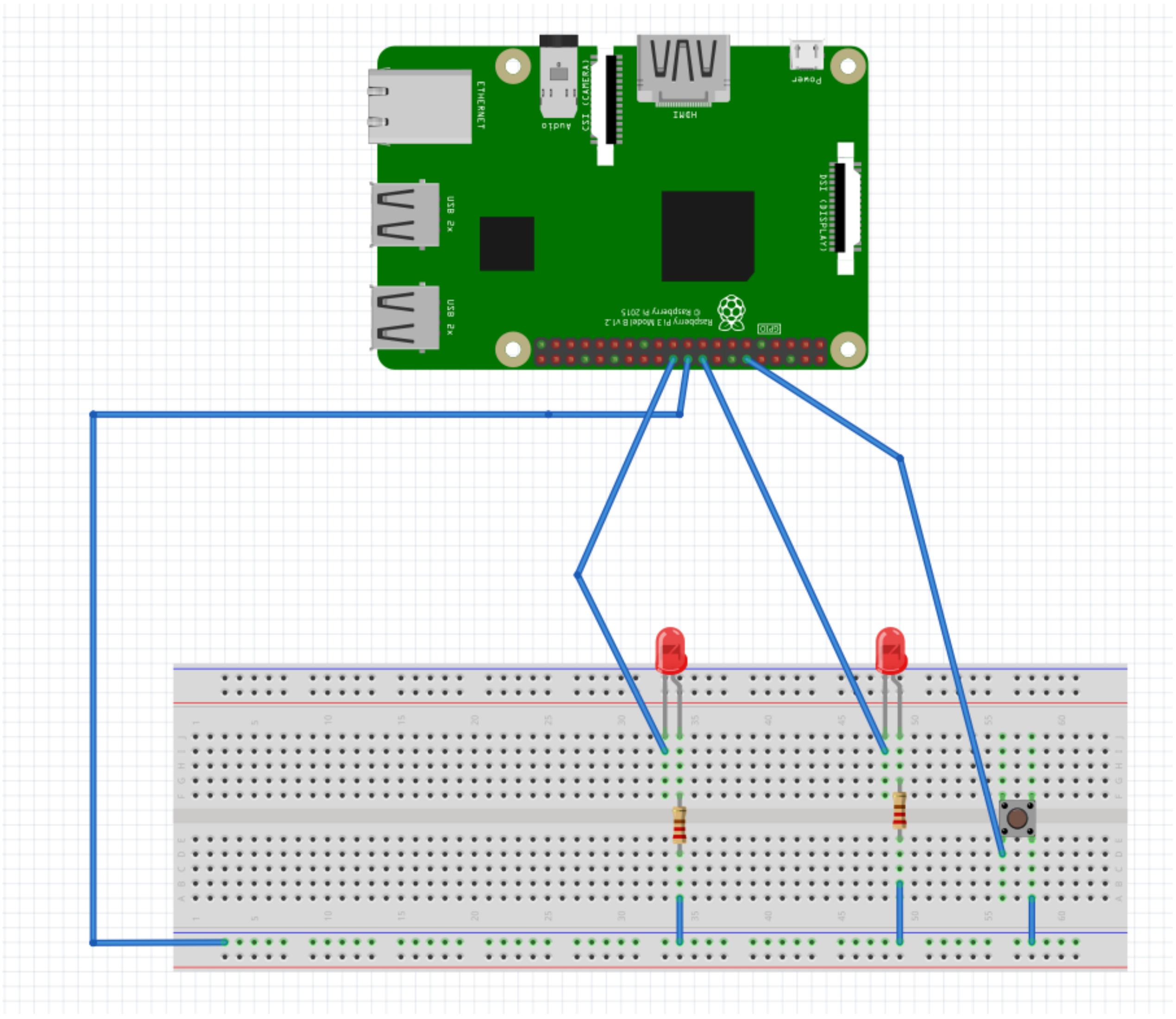
Figure 2: Circuit schematic

5 Software design

The software components and data flow directions are depicted in Figure 3. Each of these will be presented in the following subsections.

5.1 Python modules

auth_web.py: simple web server to generate authentication token using OAuth
main.py: main client, runs on a while True loop waiting for the triggered word "Alexa". Once activated, records audio, sends it to the Amazon servers for processing and plays the response through VLC
__function__.py: local python scripts that may be activated using voice commands. These run using Flask-Ask and communicate with Alexa using NGROK.

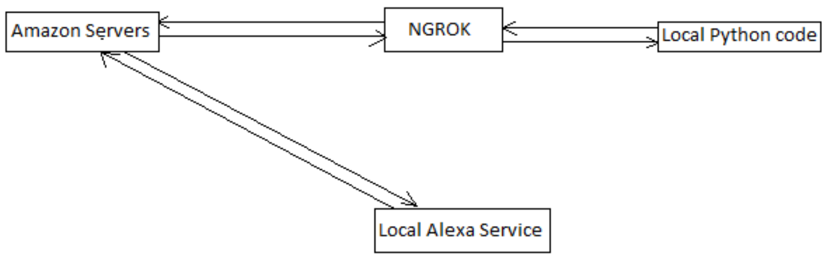

Figure 3: Software entities involved

6 Results and further work

The current version of the project supports the following functionalities:

- Basic Alexa Integration: every command supported by default on Alexa enabled devices
- Automatic Initialization on Setup
- Running custom functions defined by the user
- Configurable with respect to input/output device

The following list of extensions and improvements was identified to be supported in the future:

- Using lambda functions for Alexa, eliminating the need for GROK and Flask-Ask
- New functionality through new scripts