Integration of Home Assistant with an open-source Voice Assistant

Simone Benedetto
Salerno Daniele

Abstract

The Internet of Things is a fast growing technology thanks especially to its application in smart homes. Through the use of voice assistants, the user is provided with the ability to interact with devices without physical interaction and especially remotely. An issue that should not be underestimated is that of privacy and security, given the fact that anything connected to the Internet is vulnerable. In addition to focusing on risks of remote attacks, local risks must also be considered. Hence the need to use access mechanisms to limit the execution of critical functionality to only specific users. Since voice assistants on the market do not implement such functionality, the necessity to adopt a voice assistant that provides maximum flexibility in terms of adding new functionality arises.

1 Introduction

The Internet of Things (IoT) describes a wide range of objects having sensing and actuating devices that collect, analyze and share data among other objects, programs and platforms. Since it is one of the most important technology of this century, the use of IoT devices has grown from 8.7 billion to 50.1 billion [1]. One of the main applications of IoT concepts is the smart home, where objects are connected on a network to collaborate with each other and to be managed remotely via smartphones or voice assistants. The latter are devices that can recognize people's natural language allowing them to receive information (e.g., about the weather, traffic, news of the day) and to give commands (e.g., turn on lights, adjust temperature). Some of the most widely used and popular consumer solutions include Amazon Alexa [2], Google Assistant [3], Apple Siri [4], and Microsoft Cortana [5].

The quick diffusion of voice assistants in the home environment allows for the simplification of performing many everyday actions without a physical interaction with the objects involved in the actions.

The use of IoT devices also makes it possible to perform actions remotely, such as checking surveillance cameras while on vacation.

Examples of home devices that can be controlled include smart bulbs, air conditioners, smart plugs, and any other type of smart device. Through voice assistants, a number of actions can be performed such as:

- the turning on and off of lights;
- the adjustment of home temperature and home thermostats;
- the startup of smart home appliances.

One of the issues that comes with the use of Internet-connected devices is privacy and security. Someone could access home cameras, listen into conversations, or disable the home burglar alarm, all remotely. Security risks can also be local, such as a burglar entering the home and disabling through voice assistant the home alarm instead of using the code. This aspect has not been considered by the manufacturers of major solutions in the market.

Therefore, the need emerges to use mechanisms that can filter and block requests for actions from specific users or specific categories of users in order to avoid undesired and potentially dangerous actions for the individual and the collective. By using these mechanisms, it is possible to avoid the problem of a burglar entering the home and easily disabling the alarm.

The goal of our work is therefore to demonstrate the feasibility of using the Leon voice assistant to manage some home automation devices. Since the latter is an open source project, it provides us with maximum freedom in developing features related to privacy and security.

2 State of the Art

By now, several IT companies are working hard to create the best voice assistant that can satisfy users as much as possible. The most famous are Cortana, Siri, Google Assistant, and Amazon Alexa. A virtual assistant is software that understands natural language and, when properly trained, can communicate with human interlocutors for the purpose of providing information or performing certain tasks. Among these we will focus on the possibility of managing home automation devices, which are of different types: smart bulbs, smart ovens, smart refrigerators, and many others.

There are three main types of voice communications in IoT environments:

- Bidirectional voice communication;
- Single-directional voice communication;
- Voice recognition.

Strategy Analytics [6] thinks that using voice commands is suitable for a variety of IoT applications for the following key reasons:

- **Speech is the natural mode of communication for humans**, and being intuitive makes it easier to transmit voice commands;
- **Speech recognition is useful when the user is engaged in other actions.** In some cases there is an obligation to use voice commands, just think of when driving;
- **Telephony is an efficient medium for bidirectional voice communication.** Virtual voice operators (e.g., telephony operators) are an efficient means of bidirectional communication: the voice assistant can listen and reply without the need for complex commands, simplifying identification procedures (e.g., requesting remaining credit on the sim being used) and limiting the use of operators, leaving only the most complex handling to them;
- **Cost-saving factors**: voice integration could make it possible to eliminate the touch screen on many devices, reducing costs for devices that will be dormant most of the time.

The operation of a Voice Assistant, showed in Figure 1, consists of the interaction of a number of components [7]:

- **User:** interacts vocally with the Smart Speaker, usually activating it with a keyword;
- **Smart Speaker:** equipped with a microphone and Internet connection, it records the user's commands in audio format and sends them to the speech recognition service. It is also capable of communicating the outcome of the request vocally;
- **Speech-recognition service:** converts audio to text, interprets the user's request, and sends the text in a processable format (e.g., JavaScript Object Notation, or "JSON": this is a format suitable for data interchange between client/server applications);

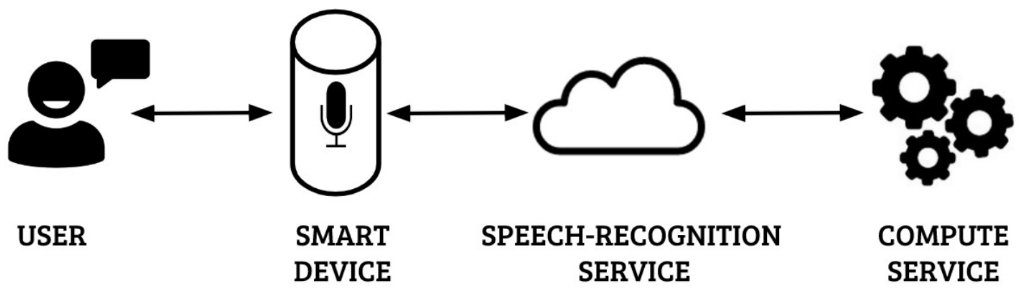

Figure 1: Voice Assistant architecture.

- **Compute service:** implements the computational logic of the requested service. It identifies which function is to be executed and, eventually, interacts with an external server to fulfill the user's request for handling a Smart Object.

Most Voice Assistants offer built-in functions that can be of different types: web search, home automation management, and so on.

Alexa, for example, provides a number of features already built into the service, so-called *"skills"*, to perform music playback, get weather information, search for content on Wikipedia, and so on.

For managing the various smart devices in a home, Alexa allows developers to build ad-hoc skills using *Alexa Skill Kit (ASK)*, a collection of APIs, tools, documentation, and code examples. Depending on the type of functionality you want to implement, you have to choose the most appropriate type of skill. ASK supports the creation of five different skill types:

- **Smart Home Skill**: Allows controlling and managing Smart Home devices, such as lights, thermostats, doors. The interaction model is managed by the Smart Home Skill API. "Intents", software mechanisms that allow users to coordinate the functions of different activities to achieve multiple goals, are processed by an *AWS Lambda Function*, a processing service offered by Amazon Web Services (AWS) [8];

- **Video Skill**: Allows the control of video devices and streaming services. The implementation of this type of skill is very similar to Smart Home Skill, it differs only from the interaction model which in this case is defined by the Video Skill API. With such APIs, it is possible to develop skills that allow customers to control their entire video experience with voice, all achievable in a very simple and intuitive way [9];

- **Music and Radio Skill**: Alexa Music and Radio Skill API is a set of interfaces that enable the selection and control of audio content streamed through an Alexa-enabled device. Interaction and usage are quite similar to the Video Skill API, making content available from the individual provider [10];

- **Custom Skill:** A Custom Skill can handle any type of request, simple or complex. For example, making a purchase on a website, reading e-mail, checking smart devices, and so on. For the implementation, an interaction model needs to be defined, i.e., we need to indicate what requests the skill can handle (intent) and what expressions (utterance) the user needs to use to invoke those requests. A cloud-based service will take care of processing those intents and subsequently return a response. In this case, the cloud-based service can be either a web service implemented ad-hoc for the skill or a function implemented on Amazon Web Service (AWS) Lambda. If you intend to create a new skill that can be used through Alexa, you need to register and configure it on the Amazon Developer Portal. Once this procedure is completed, the skill will be evaluated by Amazon and if it meets the certification requirements it will become available for download through the Alexa App, if not, it will be available only for the account through which it was developed, thus becoming a private skill [11].

In addition to the various Voice Assistants on the market there are several in the embryonic stage. Mycroft [12], Kalliope [13], Open Assistant [15], Jasper [14], and Leon are examples of open source voice assistants.

Leon [16], for example, currently provides a limited number of features called *"modules"* such as :

- Managing personal lists;

- Running a speed test;

- Grab the Github trend repository;

- Download videos from Youtube.

Since this is an open source project it is possible to add new features without any limitations. Creating new modules requires:

- **Define the actions to be performed**: the developer must define the source code related to the features he/she intends to implement;

- **Define dataset**: the developer must define the dataset which is divided into two parts:
  - **Expressions**: are the data used to train the Leon's understanding, i.e., the phrases to be spoken to perform a given action;
  - **Answers**: are the data used by Leon to provide your results binded with the modules outputs.

  Each of these datasets can have different translation, for example into English or French.

Key issue of Voice Assistants, as mentioned in the Introduction, is privacy and security in the use of such devices. Consider the case of the Amazon Alexa voice assistant: this cannot be configured to recognize a particular voice in such a way as to allow specific users to execute certain commands. Thus anyone is enabled to use all the functions of the device, even the critical ones. The only advantage it offers is the ability to receive customized responses based on the recognized user. To make up for the lack of limitations on certain features, Amazon Alexa users recommend changing the device's activation word, but this is not a solution to the problem since only four standard activation words are available. The same problem also affects Google Assistant, which only allows the creation of user profiles to receive personalized content based on the detected voice. In the opposite, an open source Voice Assistant makes it easy to integrate such functionality.

3 Background

This chapter will describe the technologies used to implement our project.

3.1 Leon

Leon is an open-source personal assistant who can live on personal server. To describe Leon's architecture, showed in Figure 2, we will use the following scenario.

1. Client (web app, etc.) makes an HTTP request to GET some information about Leon;
2. HTTP API responds information to client;
3. User talks with their microphone;

4. a) If hotword[1] server is launched, Leon listens (offline) if user is calling him by saying Leon;
   b) If Leon understands user is calling him, Leon emits a message to the main server via a WebSocket. Now Leon is listening (offline) to user;
   c) User said Hello! to Leon, client transforms the audio input to an audio blob;
5. ASR[2] transforms audio blob to a wave file;
6. STT[3] parser transforms wave file to string (Hello);
7. a) User receives string and string is forwarded to NLU[4];
   b) Or user type Hello! with their keyboard (and ignores steps 1. to 7.a.). Hello! string is forwarded to NLU;
8. NLU classifies string and pick up classification;
9. If collaborative logger[5] is enabled, classification is sent to collaborative logger;
10. Brain[6] creates a child process and executes the chosen module;
11. If synchronizer[7] is enabled and module has this option, it synchronizes content;
12. Brain creates an answer[8] and forwards it to TTS synthesizer;

[1]The hotword node is an independent Node.js process which allows you to listen for the Leon hotword. Once Leon hears his name, he listens for your request.

[2]ASR or Automatic Speech Recognition is the use of computer hardware and software-based techniques to identify and process human voice.

[3]STT, or Speech-To-Text, transforms an audio stream (speech) to a string (text).

[4]NLU (Natural Language Understanding) helps computers understand human language.

[5]The collaborative logger helps to improve the Leon's understanding. For each query you will submit to Leon, if the collaborative logger is enabled, it sends an HTTP request to an external Leon's server.

[6]Leon's brain is a major part of his core. This is where he executes his modules, talks, picks up sentences, etc.

[7]The synchronizer allows you to synchronize your content via different methods (Google Drive, on your current device, etc.) restricted by the requested module's offerings.

[8]Answers are Leon's responses. Each package has its own set of answers with different translations.

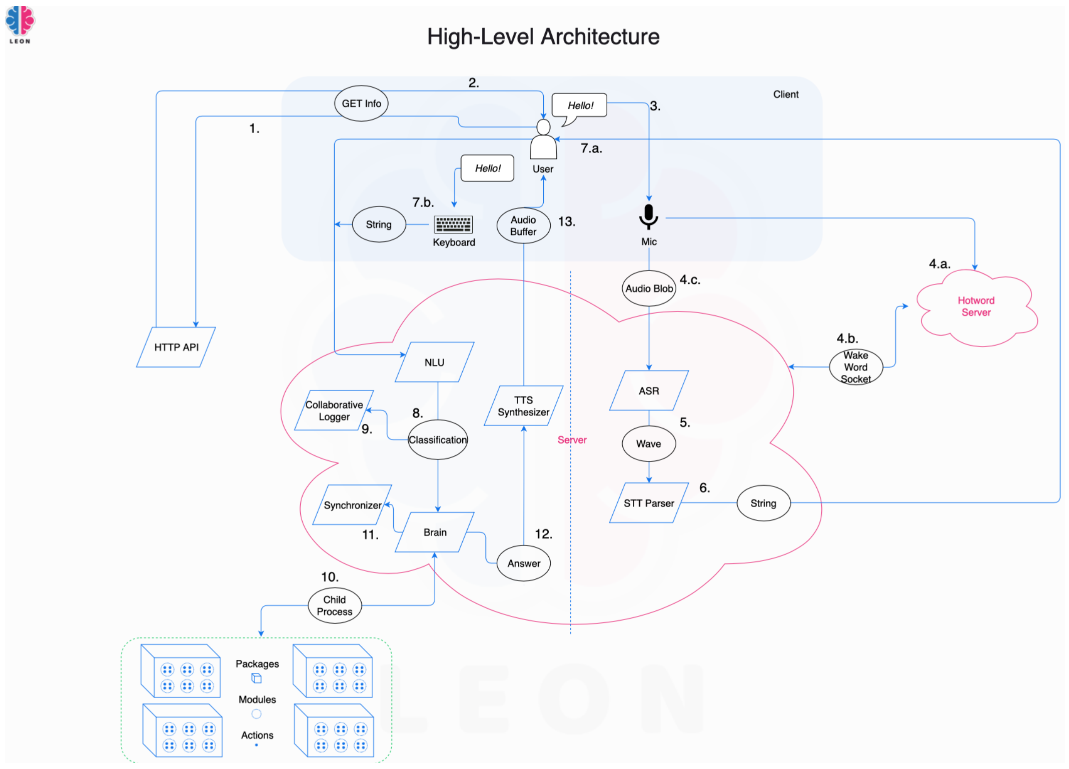

Figure 2: Leon architecture.

13. TTS[9] synthesizer transforms text answer (and send it to user as text) to audio buffer which is played by client.

3.2 Home Assistant

Home Assistant is a home automation platform that allows us to add and manage all the smart devices in our home. It is a software that aims to act as a central control system for smart home devices and can be accessed through a Web-based user interface, or through complementary apps developed for Android and iOS.

All IoT devices, technologies, software, applications, and services are supported by modular integration components, which not only include native integrations for local connectivity protocols such as Bluetooth, MQTT, Zigbee, and Z-Wave, but also support control of proprietary ecosystems if they provide public access via an open API for third-party integrations.

Home Assistant integrates with over a thousand different devices and services. Upon startup, it will automatically scan the network for known devices and enable

[9]TTS or (Text-To-Speech) transforms a string (text) to an audio stream (speech).

easy configuration. It also offers the ability to easily install other applications to implement home management. Being a cross-platform software, it is possible to use the official Home Assistant apps to quickly control all devices. After integrating all the devices in the home, you can create various custom automations such as turning on the lights when the sun goes down. The major advantage of Home Assistant is that since it is not cloud-based like other similar platforms, it has been designed with a greater emphasis on security and privacy. Since Home Assistant communicates with all devices locally, all Smart Home data remains local.

4 Case Study

In this chapter we present our case study in which we perform integration between Home Assistant, using only smart bulbs, and Leon Voice Assistant.

Since we used smart bulbs, we focusing on the actions related to those devices. To do that we implemented, using the API made available by Home Assistant [17], a "*lights*" module, within the "*homeassistant*" package, which is responsible for managing all functionalities related to smart bulbs.

4.1 Turn on and off

The first feature developed was to turn a bulb, or a group of bulbs, on and off using the following voice commands:

- "Turn on/off the x light/group";
- "Can you turn on/off the x light/group?;
- "Please turn on/off the x light/group".

The user will need to replace the letter **x** with the name of the light bulb he/she intends to turn on or off. Upon receiving the voice command, the name that the user has associated with the device will be derived from it, and after searching its **id**, the following APIs will be used to turn the device on or off, respectively:

- http://homeassistant.local:8123/api/services/light/turn_on ;
- http://homeassistant.local:8123/api/services/light/turn_off .

Several checks will also be made to see if the device exists and if the device is available in order to avoid anomalous behavior by the system. The same on/off procedure will also be carried out in the case of a group of bulbs.

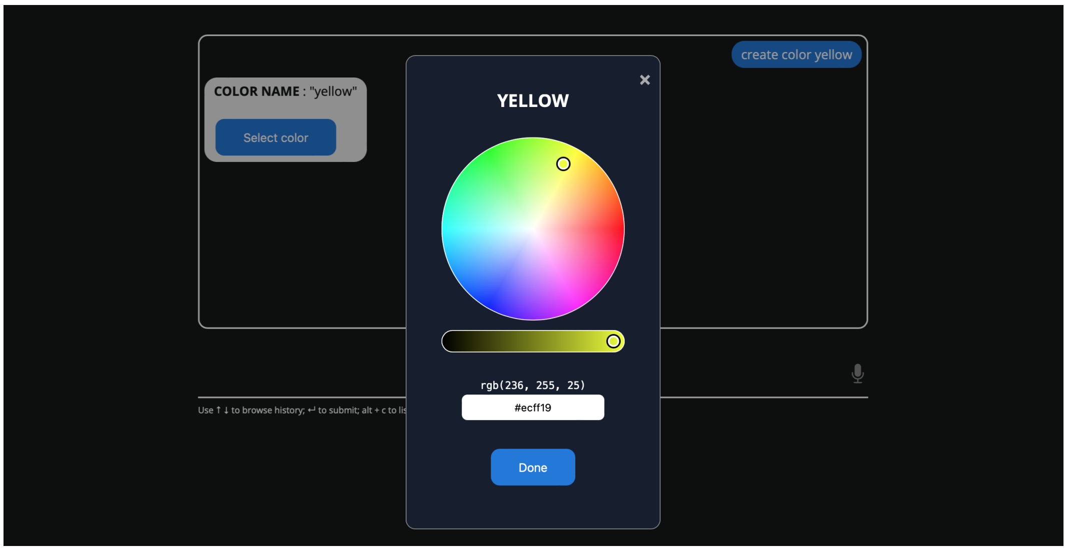

Figure 3: Color picker to select the custom color yellow.

4.2 Color management

Most lighting devices allow the color of the light to be changed according to the user's preferences or needs, and it is generally possible to use a set of colors preprogrammed by the device manufacturers. In the proposed system, it is possible to visualize the list of available colors using the command:

- "Show all possible colors for the lights".

At this point Leon will return in output the list of available color names. Typically such devices work using colors in RGB format, so it is possible to create a custom color using that format. Within the system we have developed it is possible to create a custom color and choose its name. The user can use the following command, making sure to replace the **x** with the name of the color he/she wants to create:

- "Create the color x".

At this point Leon will show to the user a color picker, showed in Figure 3, where he/she can choose the color and later save it.

These colors can later be applied to the smart bulbs using the following command:

- "Change the color of x in y color".

The user will need to replace the letter **x** with the name of the smart bulb whose color he/she intends to change, and **y** with the name of the color he/she intends to select. Upon receiving the voice command, the name that the user has associated with the device and the color identifier will be derived from it. The following API, previously used to turn on lights, can also be used to change the color of a device by specifying the RGB code of the color that he/she intend to use:

- http://homeassistant.local:8123/api/services/light/turn_on .

Several checks will also be made to see if the device exists and if the device is available and also to check if the specified color name exists in order to avoid anomalous behavior by the system.

4.3 Change brightness

In addition to changing the color of smart bulbs, it is possible to change their brightness with the command:

- "Change the brightness of x in y" .

The user will have to replace the letter **x** with the name of the bulb or group and replace the letter **y** with the desired brightness value in a range from 0 to 100. Upon receiving the voice command, the name that the user has associated with the device will be derived from it, and after searching for its **id**, the following API will be used, sending as an additional parameter the brightness value:

- http://homeassistant.local:8123/api/services/light/turn_on .

Several checks will also be made to see if the device exists and if the device is available and also to check if the specified brightness value is between 0 and 100 in order to avoid anomalous behavior by the system.

4.4 Turn on all, turn off all

Suppose the user would like to turn all the lights on or off at the same time. He/She can use the following voice command:

- "Turn x all the light of my house";
- "Please turn x all the light in my house";
- "Can you turn x all the light of my house?" .

The user should replace the letter x with the command "on" or "off" upon receiving the voice command, all available bulbs will be searched and will be respectively turned on or off with the following APIs:

- http://homeassistant.local:8123/api/services/light/turn_on ;
- http://homeassistant.local:8123/api/services/light/turn_off .

Several checks will also be made to see if devices are available to avoid errors within the system.

5 Conclusion and Future Work

In this paper, the feasibility of using an open-source voice assistant to manage smart devices using Home Assistant was demonstrated.

The final system obtained is capable of managing smart bulbs efficiently and the various tests performed were successful. Thus, the work has achieved its goal: Leon can actually be used to manage home automation devices. Currently it can only manage smart bulbs, but in future implementations it will be possible to manage any type of smart device supported by Home Assistant by creating new modules and taking advantage of the available APIs. In addition, it will be possible to implement the various privacy- and security-related features, such as an authentication mechanism to restrict critical functionality. To do this, a new module could be created to perform role-based user profiling, using voice, and insert limitations to certain functionality through appropriate controls. It would then be the administrator user who would specify the list of users enabled to perform certain actions.

References

[1] Koohang, Alex, et al. "Internet of Things (IoT): From awareness to continued use." International Journal of Information Management 62 (2022): 102442.

[2] Amazon Alexa: https://developer.amazon.com/it-IT/alexa

[3] Google Assistant: https://assistant.google.com/

[4] Apple Siri: https://www.apple.com/it/siri/

[5] Microsoft Cortana: https://www.microsoft.com/en-us/cortana

[6] Brown, Andrew. "The role of voice in IoT applications." Whitepaper, Strategy Analytics, Newton (2016).

[7] Voice Assistant Architecture: https://vitolavecchia.altervista.org/caratteristiche-architettura-e-applicazioni-dello-smart-personal-assistant/

[8] Smart Home Skill: https://developer.amazon.com/en-US/docs/alexa/smarthome/understand-the-smart-home-skill-api.html

[9] Video Skill: https://developer.amazon.com/en-US/docs/alexa/video/understand-the-video-skill-api.html

[10] Music and Radio Skill: https://developer.amazon.com/en-US/docs/alexa/music-skills/understand-the-music-skill-api.html

[11] Design your skill: https://developer.amazon.com/en-US/docs/alexa/design/design-your-skill.html

[12] Mycroft: https://github.com/MycroftAI/mycroft-core

[13] Kalliope: https://kalliope-project.github.io/

[14] Jasper: https://jasperproject.github.io/

[15] Open-Assistant: https://openassistant.org/wp/

[16] Leon: https://github.com/leon-ai/leon

[17] Home Assistant - Rest API: https://developers.home-assistant.io/docs/api/rest/