



UBA - Facultad de Ingeniería

75.39

Aplicaciones Informáticas

**Control de dispositivos por comandos de
VOZ**

Profesor: Jorge Ierache

Año 2020

Diego Martín Costa
Padrón: 78189

Indice

| | |
|--|-----------|
| PARTE I | 3 |
| Introducción | 3 |
| Justificación | 3 |
| Historia | 3 |
| Modelos Utilizados | 6 |
| Objetivos | 7 |
| Condiciones de Infraestructura y Equipamiento | 7 |
| Plan de Trabajo | 7 |
| Etapa 1 | 8 |
| Etapa 2 | 8 |
| Etapa 3 | 8 |
| Metodología de desarrollo | 9 |
| PARTE II | 9 |
| Análisis de Soluciones de reconocimiento de voz | 9 |
| Elección de la solución y fundamentación | 10 |
| Casos de Uso | 13 |
| Diagrama Conceptual de la Solución | 17 |
| Diagrama de Secuencia | 18 |
| Diagrama de Componentes | 18 |
| Esquema general de la Arquitectura de la Aplicación Mobile | 20 |
| PARTE III | 23 |
| Casos de Prueba | 23 |
| Demo | 36 |
| ANEXO I | 37 |
| Descripción general del emulador de Switch Sonoff R3 Basic | 37 |
| ANEXO II | 40 |
| Futuras investigaciones y líneas de trabajo. | 40 |
| APENDICE | 43 |
| Bibliografía y fuentes | 43 |

PARTE I

Introducción

El presente capítulo brinda una visión general acerca de la interacción con sistemas y dispositivos a través de comandos de voz, y describe una propuesta de trabajo para el desarrollo de una aplicación informática.

Justificación

El reconocimiento de comandos por voz permite a las personas interactuar con sistemas y dispositivos de manera independiente de teclados, pantallas, mouses y otros dispositivos utilizados comúnmente para operarlos.

Además de ser el método principal de interacción con asistentes virtuales (ej. Alexa, Google, Cortana, Siri), en la actualidad se han incorporado interfaces operables mediante la voz a distintos tipos de dispositivos (controles remotos de televisores, autos, parlantes inteligentes, hubs de casas inteligentes, etc). Adicionalmente, la comodidad de poder operar sistemas y dispositivos sin necesidad de utilizar las manos, también brinda una solución de accesibilidad a personas que no pueden utilizar las interfaces tradicionales debido a algún tipo de discapacidad.

Historia

La tecnología de reconocimiento del habla recorrió un largo camino desde sus inicios, a mitad del siglo pasado, a través del cual fue necesario idear, desarrollar y probar distintos enfoques para obtener los modelos y soluciones comercialmente viables con las que contamos en la actualidad.

- Década de 1950: se desarrollan los primeros proyectos de reconocimiento del habla (ASR: Automatic Speech Recognition). Entre ellos se encuentra el sistema Audrey, desarrollado por Bell Laboratories. El mismo era capaz de distinguir fonemas, y reconocer dígitos del 0 al 9 con una precisión superior al 90%, mediante la comparación de patrones eléctricos previamente tabulados. Entre sus desventajas (además de su gran tamaño y consumo), se encuentra en el hecho de que no era universal, sino que la precisión decaía al procesar la voz de otras personas que no fueran las que tabularon los patrones. Audrey no fue utilizada comercialmente, pero demostró que ‘era posible’. [1](#) [2](#) [3](#)
- Década de 1960: IBM presenta su máquina ‘Shoebbox’, que permitía el dictado de dígitos y operadores matemáticos (y enviar esa información a una calculadora para realizar las operaciones). Los prototipos de esta época seguían funcionando basándose en la comparación de las muestras de sonido contra patrones grabados previamente, con sus consecuentes limitaciones (no eran universales). [1](#) [3](#)
- Década de 1970: la Agencia de Defensa Estadounidense (DARPA) financia un programa llamado ‘Speech Understanding Research’ de 5 años, gracias al cual avanza la investigación en el campo, producto de la cual nace ‘Harpy’ de la Universidad Carnegie Mellon en 1976. Harpy era capaz de entender frases enteras y reconocer más de 1000 palabras. IBM se enfoca más en la utilización de esta tecnología para transcripción de correspondencia empresarial, mientras que Bell se enfoca más en los escenarios de interacción con máquinas. [1](#) [3](#)

- Década de 1980: comienzan a utilizarse técnicas basadas en el modelo oculto de Markov (HMM) para atacar el problema, de manera que el reconocimiento ya no se basa únicamente en la comparación de muestras contra valores tabulados y medición de distancias entre ambos, sino que se aplica un método estadístico de manera de predecir cual fonema tenía mayor probabilidad de aparecer luego de un fonema dado, mejorando muchísimo la precisión. [1](#) [3](#)
- Década de 1990: se empiezan a comercializar los primeros productos, como por ejemplo 'Dragon Dictate' que permitía reconocer 30-40 palabras por minuto (aproximadamente un tercio del ritmo del habla que utiliza una persona), aunque era necesario hablar de manera pausada con una cadencia inferior la que se utilizaría normalmente. [1](#) [3](#)
- Del 2000 en adelante: a principios de siglo, los sistemas de reconocimiento de voz estaban principalmente basados en cadenas ocultas de Markov combinadas con redes neuronales. Hay una nueva serie de programas de investigación patrocinados por DARPA, los cuales ayudan a obtener avances en el campo. Gracias a la mayor inversión (tanto pública por parte de las agencias gubernamentales y privados como Google), la mayor capacidad de procesamiento de los sistemas informáticos más modernos, y nuevas técnicas de entrenamiento de las redes neuronales, finalmente hacia el año 2007 los diseños basados en redes neuronales recurrentes investigados con anterioridad (Long ShortTerm Memory) comienzan a superar las limitaciones técnicas que tuvieron en el pasado y comienzan a obtener un performance superior a los diseños tradicionales basados en cadenas ocultas de Markov. [1](#) [3](#)

En el año 2010 Google incorpora la función de 'búsqueda por voz' en sus motores de búsqueda, la cual le permite entrenar una base de datos gigantesca basándose en las muestras enviadas por los usuarios, mejorando las capacidades de su sistema de manera dramática, hacia 2015, año en el cual la empresa comienza a

incorporar masivamente la utilización de reconocimiento de voz en sus productos y aplicaciones disponibles para smartphones. [3](#)

Modelos Utilizados

- Dynamic Time Warping (DTW): Fueron los primeros en alcanzar un nivel de fiabilidad lo suficientemente alto como para permitir el desarrollo de productos comerciales. El sistema funciona realizando cálculos y comparaciones de distancias entre la muestra y un conjunto de patrones de referencia (las palabras que puede reconocer) con el cual es entrenado previamente. Finalmente, la palabra reconocida será aquella con menor distancia. [4](#)
- Modelo Oculto de Markov (HMM): en lugar de calcular distancias, utiliza un modelo estadístico para cada una de las palabras del vocabulario, con el cual se obtiene una precisión similar al modelo DTW, pero es mucho más eficiente en cuanto a recursos requeridos y los tiempos de respuesta son superiores. Como contrapartida, la fase de entrenamiento es más costosa. [4](#)
- Redes neuronales: Casi abandonados inicialmente, re-surgen a mediados de la década del 2000 a partir de la disponibilidad de algoritmos de entrenamiento eficientes gracias a un mayor desarrollo, interés e inversión en el campo. Inicialmente tenían una gran efectividad para el reconocimiento de fonemas, pero presentaban problemas para la realización de tareas de reconocimiento continuo debido a limitaciones para modelar dependencias temporales, lo cual mejoró a partir de la utilización de arquitecturas de redes neuronales recurrentes LSTM. Utilizadas comúnmente como pre-procesamiento a una tarea de reconocimiento utilizando HMM. [4](#) [5](#) [6](#)

Objetivos

Se propone como objetivo de trabajo el desarrollo de una aplicación tipo IoT que permita poder operar, mediante comandos de voz, cualquier dispositivo eléctrico susceptible de ser controlado con un dispositivo tipo switch (con estados encendido y apagado).

Condiciones de Infraestructura y Equipamiento

Para el presente trabajo, se simulará un switch comercial de marca SONOFF modelo BASIC R3. Dicho dispositivo expone una API REST que puede ser consumida a través de una red de área local (o de internet si el dispositivo se encuentra conectado a una red con acceso a internet), que permite modificar el estado del switch de encendido a apagado y viceversa. El simulador del dispositivo se desarrollará como un servidor web utilizando Node.js, con lo cual será necesario configurar dicho servidor en cualquier computadora tipo PC. El servidor web implementará los protocolos descritos por el fabricante que pueden ser consultados en el repositorio Github del mismo

(https://github.com/itead/Sonoff_Devices_DIY_Tools/blob/master/SONOFF%20DIY%20MODE%20Protocol%20Doc%20v2.0%20Doc.pdf), y

brindará la posibilidad de monitorear el estado del switch mediante una página web que informará el estado del mismo en tiempo real utilizando una conexión de tipo web-socket.

Para el consumo de dicha API (y el control efectivo del switch) se desarrollará una aplicación Android que deberá ser utilizada en un dispositivo Android (smartphone o tablet conectada a la misma red que el switch, con micrófono y versión de sistema operativo Android 6.0 o superior).

Plan de Trabajo

Se realizará el desarrollo en tres etapas:

- Etapa 1

Implementación de Simulador de Dispositivo Sonoff R3.

Se utilizará Node.js con los paquetes Express y Web Socket para implementar un servidor web que simule la API expuesta por el dispositivo respetando el protocolo indicado por el fabricante en la documentación del mismo. Como feedback, el servidor web brindará una página accesible desde cualquier browser en su red, mediante la cual se podrá monitorear en tiempo real el estado actual del switch.

- Etapa 2

Investigación de soluciones que provean reconocimiento de comandos de voz

Para poder interpretar los comandos de voz expresados por el usuario, será necesario un motor de reconocimiento de voz que permita reconocer los comandos que se utilizaran para el control del dispositivo. Se investigarán las soluciones comerciales disponibles dando preferencia a soluciones que permitan operar de manera 'offline' sin necesidad de enviar información hacia servidores de terceros.

- Etapa 3

Desarrollo de aplicación Mobile

Una vez decidida la solución de reconocimiento de voz a utilizar, se implementará una aplicación mobile para dispositivos Android 6.0+ (con micrófono) que brindará la posibilidad de monitorear en tiempo real el estado actual del switch (mediante la utilización de polling contra la API expuesta del mismo). También permitirá, a partir de la implementación de la solución de reconocimiento de voz elegida en la Etapa 2, modificar el estado del switch a través

de comandos de voz, que serán interpretados (una vez reconocidos por motor de reconocimiento de voz) y mapeados de manera de efectuar las llamadas correspondientes a la API expuesta por el switch.

Se compartirá el código en dos repositorios de github, uno para el simulador del switch (<https://github.com/dicosta/SonoffR3Mock>) y otro para la aplicación android (https://github.com/dicosta/7539_AI).

Metodología de desarrollo

El grupo es de un solo integrante, de manera que se orientará el trabajo a desarrollos incrementales, implementando requerimientos de manera de obtener una aplicación con más funcionalidades en cada iteración. Se realizará modelado UML.

PARTE II

El presente capítulo describe las opciones analizadas y presenta la solución de reconocimiento de voz seleccionada, con su correspondiente justificación y criterio de selección, desarrollando sus puntos fuertes y ventajas y desventajas. Luego se desarrollan los Casos de Usos a implementar en la aplicación, los casos de prueba aplicados a la misma y diagramas de secuencia y componentes descriptivos de la aplicación informática desarrollada.

Análisis de Soluciones de reconocimiento de voz

Con el correr de los años se desarrollaron muchos kits de herramientas, soluciones comerciales, soluciones open-source y productos que permiten incorporar la tecnología de voz a los sistemas.

Enfocamos el análisis en las opciones disponibles para utilizar en una aplicación mobile, para reconocimiento de comandos.

| | Speech Recognizer (Android SDK) 10 | Google Cloud Speech AP 11 | Kaldi/Vosk 13 12 | Speechmatics RealTime 9 | IBM Cloud Watson Speech-to-text 8 |
|---|---|--|---|---|---|
| Es Multiplataforma? | No (Solo Android) | Si | Si | Si | Si |
| Costo | Gratuito | Gratuito hasta 60 Minutos (en segmentos de 15 segundos). Luego, tiene un costo de 0.006 dólares cada 15 segundos | Gratuito | Es pago pero no publican los precios ya que depende de la solución seleccionada y donde será desplegada | Plan gratuito hasta 500 minutos con limitaciones. Planes pagos desde \$0.01 el minuto |
| Funciona sin conexión a Internet? | Si | No | Si | No | No |
| Es Open Source? | No | No | Si | no | No |
| Permite desplegarse el motor de ASR en un servidor propio dedicado? | No | No | Si | Si | No |
| Cantidad de lenguajes soportados | ~120 | 120+ | 9 | 31 | 17 |
| Pueden re-entrenarse los modelos? | No | Si | Si | Si | Si |
| Tiene soporte actualmente? | Si | Si | Si | Si | Si |

Elección de la solución y fundamentación

De acuerdo a lo expresado en el plan de trabajo, en la elección de la solución a utilizar se dará mucho valor a la posibilidad de que la misma funcione de manera 'offline' realizando el procesamiento en el mismo dispositivo sin necesidad de enviar la información a servidores de terceros. Debido a que el switch que operará la aplicación es de

propósito múltiple (aunque en nuestra aplicación lo usemos como ejemplo para encender una luz), puede ser importante que el dispositivo manejado por el switch pueda encenderse o apagarse en situaciones donde la conexión a internet no esté disponible, en particular en algún caso donde se maneje un dispositivo importante por una persona imposibilitada de realizarlo de manera manual.

Desde el punto de vista económico, que pueda funcionar de manera offline es importante ya que nos ahorrará costos de servidor propio y costos asociados a servidores cloud de ASR (todo el procesamiento necesario se hará de manera local).

Utilizando estos criterios se evaluaron las dos opciones que funcionan de manera 'offline' (Android SDK y Kaldi/Vosk), encontrándose las siguientes limitaciones para la opción SpeechRecognizer de la SDK de Android ¹⁰:

- No está disponible en todos los dispositivos Android
- No está disponible en todas las versiones de Android
- No siempre puede funcionar de manera offline, a veces requiere conectarse a los servidores de Google.
- Requiere instalación de diccionarios especiales de GBoard de acuerdo al idioma a utilizar desde la configuración de Android, lo cual no es fácil de realizar para un usuario promedio.

Por lo expuesto anteriormente, se decidió utilizar Kaldi/Vosk como solución. ^{13 12}

Vosk actúa como API ofreciendo una interfaz para configurar y utilizar el motor de reconocimiento de voz de Kaldi. Ambos pueden incluirse como librerías tanto en aplicaciones Android, como en aplicaciones iOS, ya que al estar desarrollado en C++ puede correr en múltiples plataformas y funcionar en dispositivos pequeños como teléfonos o raspberries.

Las librerías tienen un tamaño no muy grande (10MB), pero debido a que el procesamiento será local, será necesario incluir el modelo del lenguaje al empaquetar la aplicación android (el peso de los modelos es considerable ya que ronda los 30 - 50 MB por lenguaje).

Para mitigar esto, y dadas las características de la aplicación (solo necesita reconocer algunas palabras y no el diccionario completo), es posible re-entrenar el modelo, utilizando las herramientas de Kaldi, con

las palabras que reconocerá/utilizará la aplicación, lo cual permitiría reducir el tamaño del diccionario considerablemente. Un ejemplo realizado con el modelo inglés re-entrenado para 15 palabras, redujo el tamaño del diccionario de 50MB a ~20MB.

Hay una segunda opción y es la posibilidad que ofrece Vosk de configurar un servidor propio dedicado para el reconocimiento de voz, que utiliza protocolos HTTP (REST o WebSocket) o gRPC. Si el dispositivo a controlar por voz es lo suficientemente potente, podría hostear este servidor y la aplicación delegar el procesamiento de voz en el dispositivo a manejar (el cual ya posee un servidor HTTP ya que permite el control remoto a través de su API REST).

Casos de Uso

| | |
|-----------------|--|
| Nombre | 01 - Iniciar aplicación |
| Descripción | Como usuario quiero iniciar la aplicación para controlar el switch Sonoff R3 |
| Pre-condiciones | <ul style="list-style-type: none"> La aplicación debe estar instalada. |
| Flujo Normal | <ol style="list-style-type: none"> 1. Buscar la aplicación en el teléfono 2. Hacer 'tap' para iniciar la aplicación 3. La aplicación muestra una pantalla de bienvenida durante unos segundos en donde se inicializa y luego va a otra pantalla donde se muestra el estado actual del dispositivo (caso de uso 02 - Monitorear estado del Switch) |
| Excepciones | <ol style="list-style-type: none"> 1. Si la aplicación no se encuentra instalada, el usuario deberá instalar la aplicación 2. Si la aplicación no puede encontrar el dispositivo a controlar en la dirección IP preconfigurada, mostrará un diálogo al usuario para que el mismo pueda configurar la dirección IP del dispositivo que desea controlar. <ol style="list-style-type: none"> a. Si en la dirección IP ingresada por el usuario se encuentra el dispositivo, va al caso de uso 02 - Monitorear estado del Switch b. Si en la dirección IP ingresada por el usuario no puede encontrarse un dispositivo, vuelve al paso 2. |

| | |
|-----------------|--|
| Nombre | 02 - Monitorear estado del Switch en tiempo real |
| Descripción | Como usuario quiero poder monitorear el switch para saber en qué estado se encuentra en todo momento. |
| Pre-condiciones | <ul style="list-style-type: none"> La aplicación debe estar instalada y abierta |
| Flujo Normal | <ol style="list-style-type: none"> Una vez iniciada la aplicación, la misma se conecta con el dispositivo y la pantalla principal muestra el estado del switch en todo momento (con tasa de refresco de 1 vez por segundo), representado con la imagen de una lámpara encendida (en caso de que el switch esté encendido) o una lámpara apagada (en caso de que el switch esté apagado). La pantalla muestra también un botón debajo de la imagen de la lámpara con el texto 'Apagar' o 'Encender' de acuerdo al estado actual del switch En caso de que el switch cambie de estado (desde otra aplicación o de manera manual) el estado se actualizará en la pantalla principal automáticamente. |
| Excepciones | <ol style="list-style-type: none"> Si se pierde conexión con el servidor, la misma será restablecida automáticamente sin necesidad de intervención del usuario. Durante el periodo sin conexión la pantalla principal no actualizará el estado del switch, mostrando el estado de la última actualización. |

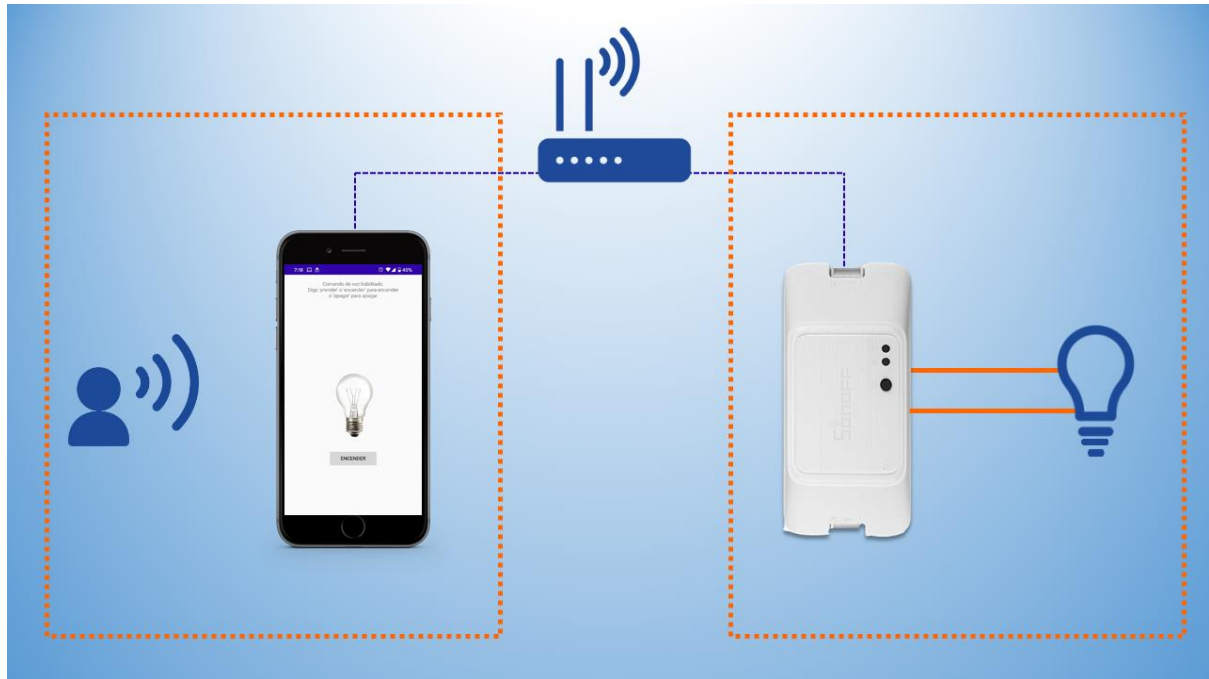
| | |
|-----------------|---|
| Nombre | 03 - Modificar el estado del Switch (modo manual) |
| Descripción | Como usuario quiero poder modificar el estado del switch mediante una interacción manual con la app |
| Pre-condiciones | <ul style="list-style-type: none"> La aplicación debe estar instalada, abierta y conectada con el Switch |
| Flujo Normal | <ol style="list-style-type: none"> La pantalla principal muestra un botón con la leyenda 'Prender' o 'Apagar' (dependiendo del estado actual del switch) que permite al usuario presionarlo y modificar el estado del switch. Al pulsar el botón, el switch cambia de estado, actualizando tanto la imagen como el texto del botón. |
| Excepciones | <ol style="list-style-type: none"> Si se pierde conexión con el servidor, al pulsar el botón, no se efectuará ninguna acción. Solo se modificara el estado del Switch cuando se pulse el botón y haya conexión al dispositivo. |

| | |
|-----------------|--|
| Nombre | 04 - Encender el switch mediante comando de voz |
| Descripción | Como usuario quiero poder encender el switch mediante un comando de voz. |
| Pre-condiciones | <ul style="list-style-type: none"> La aplicación debe estar instalada, abierta y conectada con el Switch. El switch debe estar en estado apagado. |
| Flujo Normal | <ol style="list-style-type: none"> Desde la pantalla principal, pronunciar en voz alta cerca del microfono del telefono el comando para encender el switch (para encender están soportados los comandos 'encender', 'encender luz', 'encender la luz', 'prender', 'prender luz', 'prender la luz'). Una vez recibido y procesado el comando de voz, el switch modifica su estado a encendido, reflejándose en la imagen que indica el estado y en la leyenda del botón. |
| Excepciones | <ol style="list-style-type: none"> Si se pierde conexión con el servidor, al pronunciar el comando, no se efectuará ninguna acción. Solo se modificara el estado del Switch cuando se pronuncie el comando y haya conexión al dispositivo. Si el switch ya se encuentra encendido, pronunciar el comando para encender no tiene ningún efecto. Si estando el switch en estado apagado se pronuncia cualquier otro comando que no sea uno de los comandos indicados en el paso 1 del flujo normal, se mostrará un texto indicando que el comando de voz es desconocido (mostrando el comando interpretado por el reconocedor de voz) |

| | |
|-----------------|--|
| Nombre | 05 - Apagar el switch mediante comando de voz |
| Descripción | Como usuario quiero poder apagar el switch mediante un comando de voz. |
| Pre-condiciones | <ul style="list-style-type: none"> La aplicación debe estar instalada, abierta y conectada con el Switch. El switch debe estar en estado encendido. |
| Flujo Normal | <ol style="list-style-type: none"> Desde la pantalla principal, pronunciar en voz alta cerca del microfono del telefono el comando para apagar el switch (para apagar están soportados los comandos 'apagar', 'apagar luz', 'apagar la luz'). Una vez recibido y procesado el comando de voz, el |

| | |
|-------------|---|
| | switch modifica su estado a apagado, reflejándose en la imagen que indica el estado y en la leyenda del botón. |
| Excepciones | <ol style="list-style-type: none">1. Si se pierde conexión con el servidor, al pronunciar el comando, no se efectuará ninguna acción. Solo se modificara el estado del Switch cuando se pronuncie el comando y haya conexión al dispositivo.2. Si el switch ya se encuentra apagado, pronunciar el comando para apagar no tiene ningún efecto.3. Si estando el switch en estado encendido se pronuncia cualquier otro comando que no sea uno de los comandos indicados en el paso 1 del flujo normal, se mostrará un texto indicando que el comando de voz es desconocido (mostrando el comando interpretado por el reconocedor de voz) |

Diagrama Conceptual de la Solución



En este diagrama se puede apreciar conceptualmente la solución propuesta. A la izquierda tenemos al usuario interactuando mediante comandos de voz con una aplicación móvil, la cual conoce la ruta a un Switch Sonoff dentro de la misma red de área local, ambos vinculados a través de un router. La aplicación será la encargada de reconocer los comandos pronunciados por el usuario, interpretar los mismos mediante su motor de reconocimiento de voz, y realizar las llamadas necesarias al switch para modificar el estado del aparato (en este diagrama una lámpara), que podrá ser 'encendido' o 'apagado'.

Diagrama de Secuencia

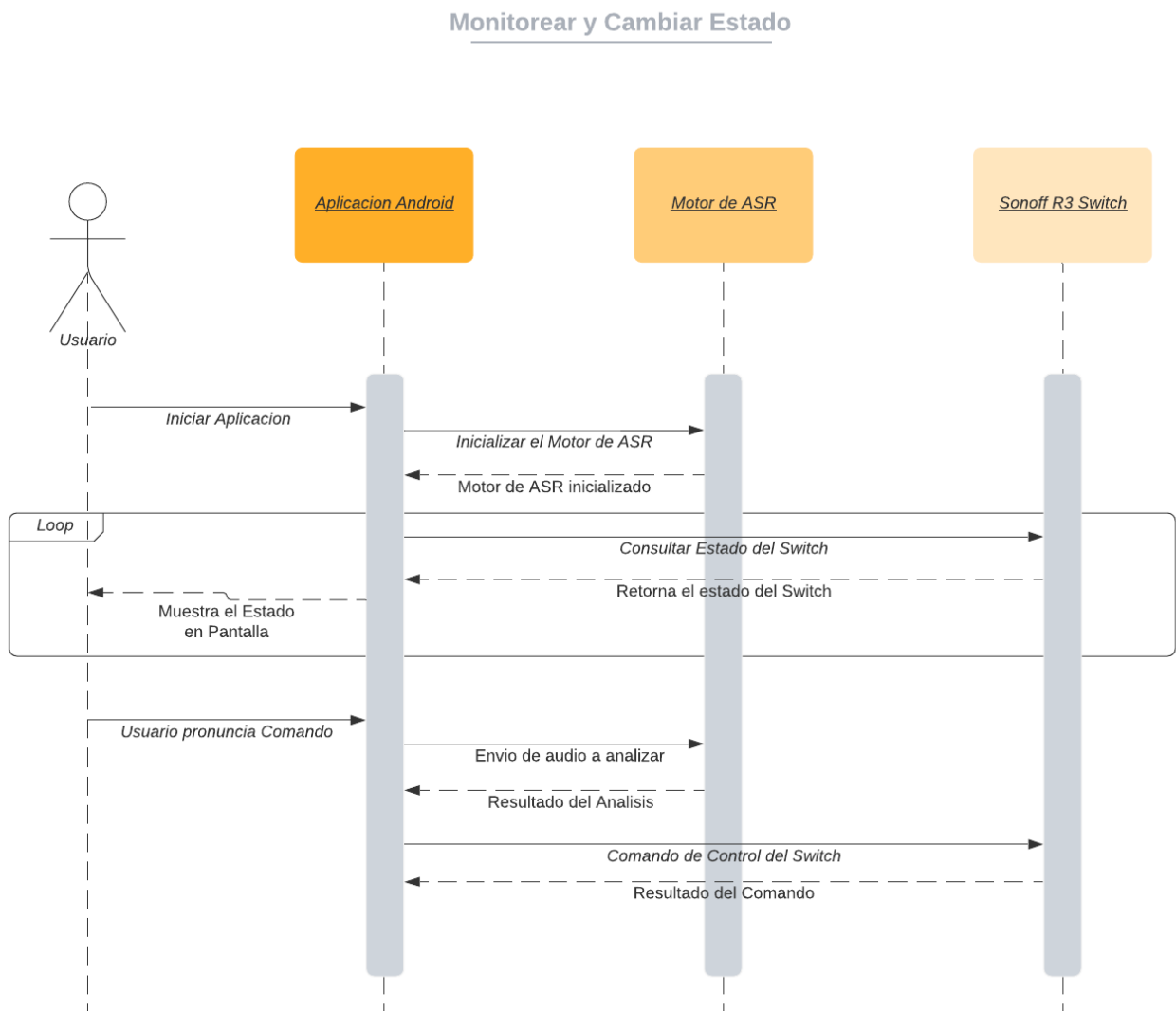
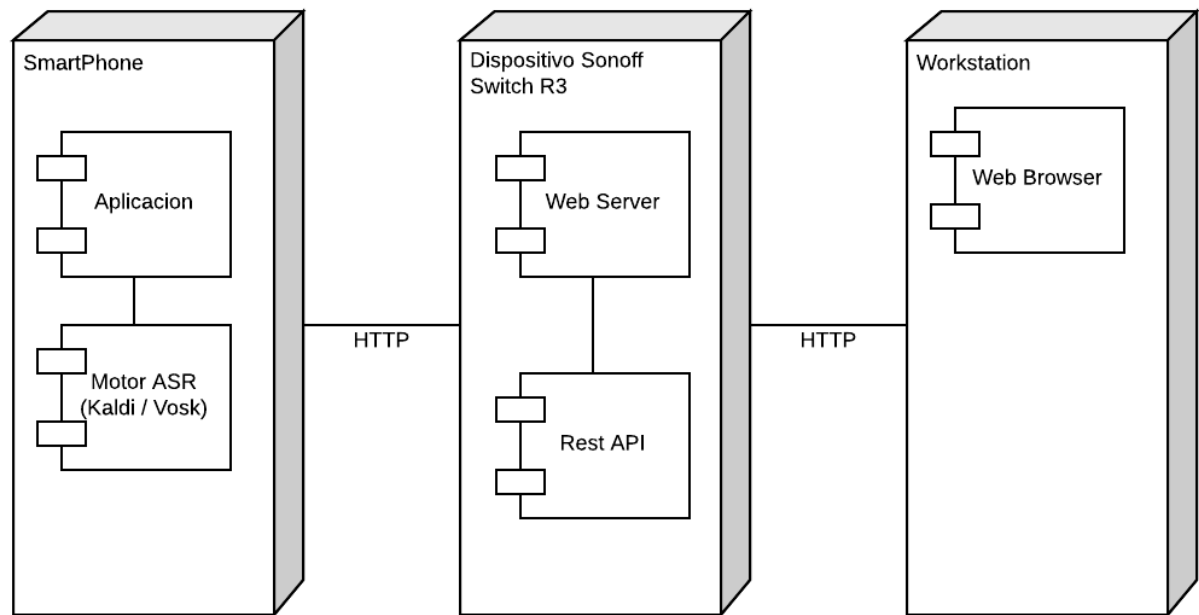


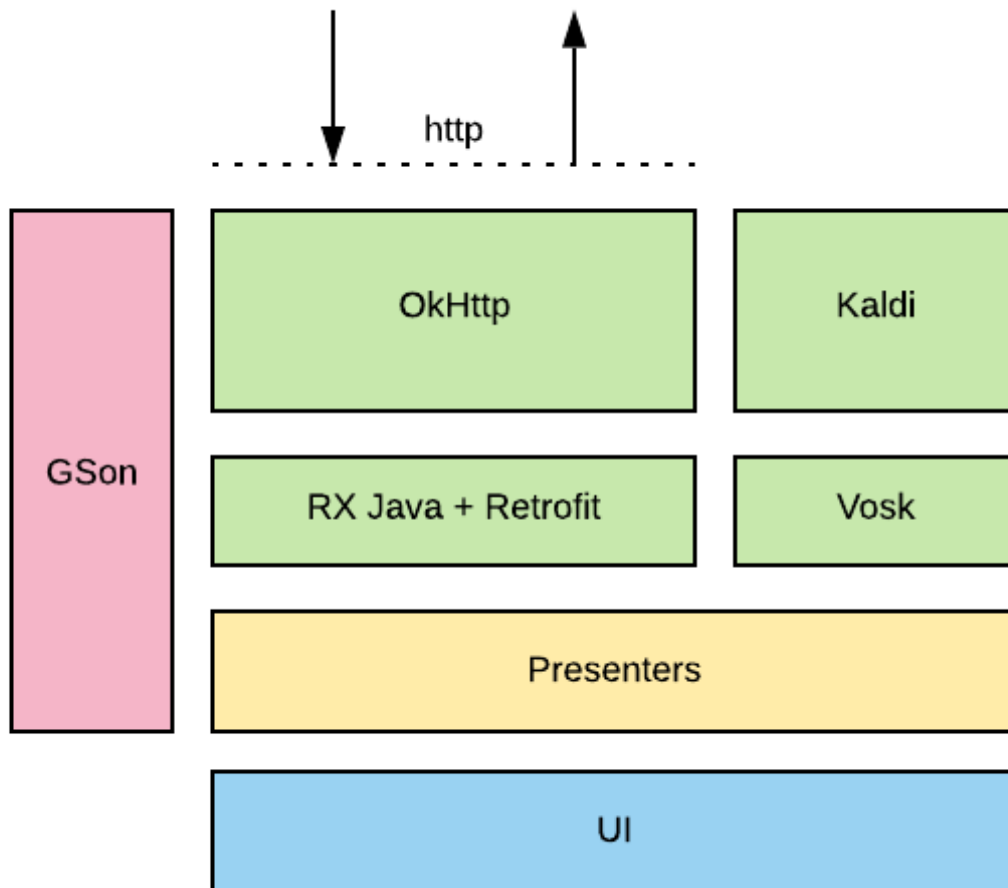
Diagrama de secuencia representando la interacción entre el usuario que inicia la aplicación android, la aplicación android inicializando el motor de ASR (Reconocimiento Automático de Voz) con el modelo del lenguaje correspondiente (español para esta prueba de concepto), y que una vez inicializado, pasa a la pantalla principal donde, mediante un bucle, consulta de manera constante al Switch para obtener el estado actual del mismo y representarlo en la pantalla principal.

También se representa la interacción del usuario con la aplicación a través de un comando de voz, el cual es recibido y enviado para su análisis al Motor de ASR. Una vez se completa el análisis, se reciben los resultados y se interpretan los mismos, y en caso de ser necesario, se envía el comando correspondiente al Switch.

Diagrama de Componentes



Esquema general de la Arquitectura de la Aplicación Mobile



Gson: El intercambio de información entre Vosk y la capa de Presenters, como también entre la el Switch Sonoff y la aplicación mobile se hará a través de mensajes JSON. Se utilizará la librería GSon para la serialización y deserialización de los mensajes

UI: Se utilizaran Activities y Fragments de Android como medio de representación de la información y de captura de interacciones de parte del usuario. Estos componentes conocen e intercambian mensajes con la capa de Presenters.

Presenters: En esta capa se conocen la interfaces expuestas por la capa de UI, de manera de poder enviar mensajes a la misma. A su vez, esta capa también conoce una instancia (implementada como un

Singleton de la aplicación) del motor de reconocimiento de Kaldi, encapsulado dentro de un objeto `SpeechRecognizer` provisto por Vosk (e inicializado al inicio de la aplicación con un modelo del lenguaje español) que ofrece una interfaz mas comoda para trabajar. También posee una instancia de un proxy que realiza las llamadas http necesarias al Switch para modificar u obtener el estado del mismo. De esta manera, esta capa es la encargada de orquestar el monitoreo constante del Switch mediante un polling (utilizando el proxy e informando del estado actual a la capa de UI), y enrutando el audio proveniente del usuario hacia el objeto `SpeechRecognizer`, alimentando el reconocedor de voz, de manera que el mismo informe sobre las palabras detectadas. Luego se analizan dichas palabras y si coinciden con alguno de los comandos, se efectúa la llamada correspondiente a la API del Switch (a través del proxy).

RX Java + Retrofit: Retrofit permite declarar de manera sencilla la interfaz de una API REST e instanciar un proxy (alimentándolo con la definición de la interfaz y la ruta donde vive la API) de manera de simplificar el consumo de dicha API. También se encarga de realizar la serialización/deserialización de la información en formato JSON utilizando el deserializador preferido (en este caso, se utiliza GSON). RxJava permite simplificar la asincronicidad del proceso de consumir una API y obtener los resultados, mediante la conversión a Streams, gestionando los threads de manera eficiente. Retrofit se apoya sobre la capa de OkHttp para consumir la API expuesta por el Switch.

OkHttp: Es una implementación eficiente de un cliente de HTTP, la cual es utilizada en la aplicación para poder consumir la API expuesta por el switch a través del protocolo HTTP.

Kaldi: Es el motor de reconocimiento automático de voz, incluido en la aplicación como librería externa. Su utilización se encuentra encapsulada por la capa de Vosk.

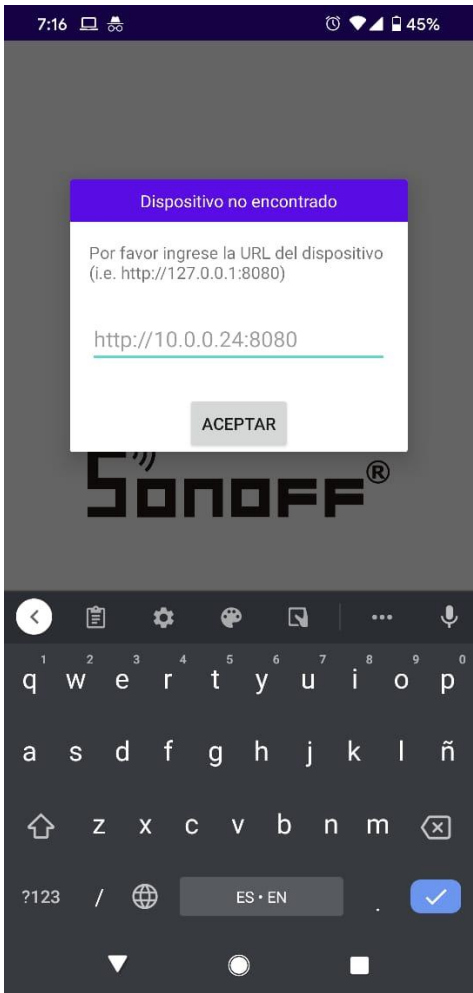
Vosk: Su función es encapsular y proveer una interfaz más amigable a la aplicación y a la capa de presenters para interactuar con el reconocedor automático de voz, ya sea indicando el modelo a utilizar

(en el caso de esta aplicación será un modelo del lenguaje español), como alimentándolo de comandos de voz para que el analizador los interprete y devuelva en modo de texto.


PARTE III

Casos de Prueba


| | |
|--------------------|---|
| Nombre | TC01 |
| Escenario | Al abrir la aplicación, el Switch no se encuentra en la dirección configurada en la aplicación. |
| Pasos para el test | <ol style="list-style-type: none">1. Iniciar la aplicación2. Esperar a que la pantalla de bienvenida termine de inicializar la aplicación |
| Datos para el test | <ul style="list-style-type: none">• Aplicación configurada para conectarse a un switch en la dirección http://10.0.0.24:8080 (default)• El servidor del switch no esta iniciado en la dirección http://10.0.0.24:8080 o esta corriendo en otra dirección |
| Resultado Esperado | La aplicación informa al usuario que no pudo encontrar el dispositivo y muestra un diálogo para que el usuario ingrese la dirección del dispositivo. |

| | |
|---------------------------------|---|
| Resultado Obtenido |  |
| Resultado del Test (Pasó/Falló) | Pasó |

| | |
|--------------------|--|
| Nombre | TC02 |
| Escenario | Al abrir la aplicación, el Switch se encuentra en la dirección configurada en la aplicación. |
| Pasos para el test | <ol style="list-style-type: none"> 1. Iniciar la aplicación 2. Esperar a que la pantalla de bienvenida termine de inicializar la aplicación 3. La pantalla principal se abre automáticamente y muestra el estado actual del Switch. |
| Datos para el test | <ul style="list-style-type: none"> • Aplicación configurada para conectarse a un switch en la dirección http://10.0.0.6:8080 (default) • El servidor del switch esta iniciado en la dirección http://10.0.0.6:8080 • El switch se encuentra en estado 'apagado' |


| | |
|---------------------------------|--|
| Resultado Esperado | La aplicación muestra en la pantalla principal el estado actual del switch, que para esta prueba es 'apagado'. |
| Resultado Obtenido |  |
| Resultado del Test (Pasó/Falló) | Pasó |


| | |
|--------------------|---|
| Nombre | TC03 |
| Escenario | Teniendo la aplicación abierta y conectada al switch, se observan en la misma los cambios de estado del switch en tiempo real |
| Pasos para el test | <ol style="list-style-type: none"> 1. Iniciar la aplicación 2. Esperar a que la pantalla de bienvenida termine de inicializar la aplicación 3. La pantalla principal se abre automáticamente y muestra el estado actual del Switch. 4. Desde otra instancia de la aplicación (en otro |

| | |
|--------------------|--|
| | teléfono) o a través de una llamada a la API del switch realizada a través de alguna herramienta, modificar el estado del mismo. |
| Datos para el test | <ul style="list-style-type: none"> • Aplicación configurada para conectarse a un switch en la dirección http://10.0.0.6:8080 (default) • El servidor del switch esta iniciado en la dirección http://10.0.0.6:8080 • El switch se encuentra en estado 'apagado' |
| Resultado Esperado | La aplicación muestra en la pantalla principal el cambio de estado el switch. Al iniciar la aplicación el switch estaba apagado y luego de modificarse el estado (a través de otra instancia de la aplicación en otro teléfono o con una llamada a la API realizada con alguna otra herramienta), la pantalla se actualiza mostrando el nuevo valor (encendido) |
| Resultado Obtenido | <p>Antes de modificar el estado:</p>  <p>Luego de modificar el estado:</p> |


| | |
|---------------------------------|---|
| |  |
| Resultado del Test (Pasó/Falló) | Pasó |


| | |
|--------------------|--|
| Nombre | TC04 |
| Escenario | Teniendo la aplicación abierta y conectada al switch, se desea cambiar el estado del switch a través de una interacción manual. |
| Pasos para el test | <ol style="list-style-type: none"> 1. Iniciar la aplicación 2. Esperar a que la pantalla de bienvenida termine de inicializar la aplicación 3. La pantalla principal se abre automáticamente y muestra el estado actual del Switch. 4. Presionar el botón debajo de la imagen indicativa del estado actual del switch, para cambiar el estado del switch de 'apagado' a 'encendido'. |

| | |
|--------------------|--|
| Datos para el test | <ul style="list-style-type: none">• Aplicación configurada para conectarse a un switch en la direccion http://10.0.0.6:8080 (default)• El servidor del switch esta iniciado en la direccion http://10.0.0.6:8080• El switch se encuentra en estado 'apagado' |
| Resultado Esperado | Luego de presionar el botón, el switch cambia su estado a 'encendido'. Esto se refleja en la aplicación al cambiar la imagen (de una lámpara apagada a una lámpara prendida) y el texto del botón (de 'encender' cambia a 'apagar'). |
| Resultado Obtenido | <p>Antes de presionar el botón:</p>  <p>Luego de presionar el botón:</p> |


| | |
|---------------------------------|---|
| |  |
| Resultado del Test (Pasó/Falló) | Pasó |


| | |
|--------------------|---|
| Nombre | TC05 |
| Escenario | Teniendo la aplicación abierta y conectada al switch, se desea cambiar el estado del switch de 'apagado' a 'encendido' a través de un comando de voz. |
| Pasos para el test | <ol style="list-style-type: none"> 1. Iniciar la aplicación 2. Esperar a que la pantalla de bienvenida termine de inicializar la aplicación 3. La pantalla principal se abre automáticamente y muestra el estado actual del Switch. 4. Pronunciar en voz alta cerca del microfono del |

| | |
|--------------------|--|
| | telefono el comando de voz para encender el switch (puede ser 'encender', 'encender luz', 'encender la luz', 'prender', 'prender luz', 'prender la luz') |
| Datos para el test | <ul style="list-style-type: none"> • Aplicación configurada para conectarse a un switch en la direccion http://10.0.0.6:8080 (default) • El servidor del switch esta iniciado en la direccion http://10.0.0.6:8080 • El switch se encuentra en estado 'apagado' |
| Resultado Esperado | Luego de pronunciar en voz alta el comando, el switch cambia su estado a 'encendido'. Esto se refleja en la aplicación al cambiar la imagen (de una lámpara apagada a una lámpara prendida) y el texto del botón (de 'encender' cambia a 'apagar'). |
| Resultado Obtenido | <p>Antes de pronunciar el comando de voz para encender:</p>  <p>Luego de pronunciar el comando de voz para</p> |

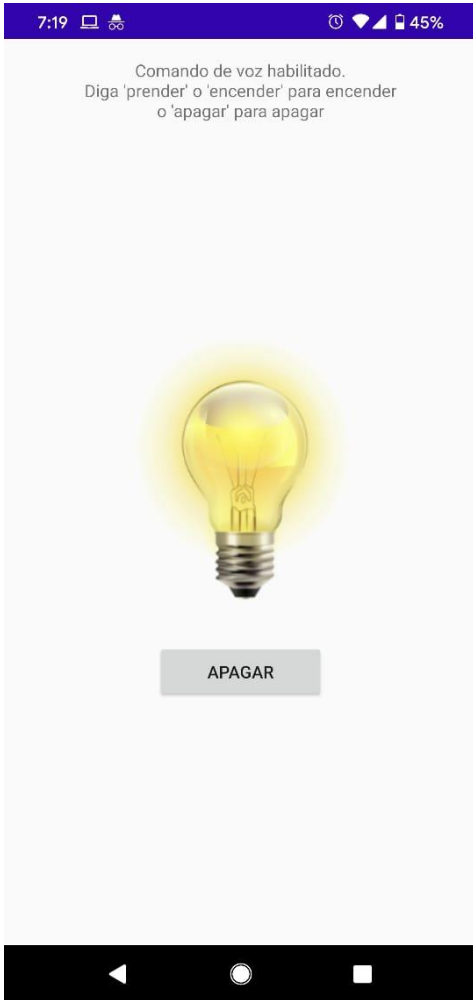
| | |
|---------------------------------|--|
| | <p>encender:</p>  |
| Resultado del Test (Pasó/Falló) | Pasó |


| | |
|--------------------|--|
| Nombre | TC06 |
| Escenario | Teniendo la aplicación abierta y conectada al switch, se desea cambiar el estado del switch de encendido a apagado a través de un comando de voz. |
| Pasos para el test | <ol style="list-style-type: none"> 1. Iniciar la aplicación 2. Esperar a que la pantalla de bienvenida termine de inicializar la aplicación 3. La pantalla principal se abre automáticamente y muestra el estado actual del Switch. 4. Pronunciar en voz alta cerca del microfono del telefono el comando de voz para apagar el switch |

| | |
|--------------------|--|
| | (puede ser 'apagar', 'apagar luz', 'apagar la luz') |
| Datos para el test | <ul style="list-style-type: none"> • Aplicación configurada para conectarse a un switch en la dirección http://10.0.0.6:8080 (default) • El servidor del switch está iniciado en la dirección http://10.0.0.6:8080 • El switch se encuentra en estado 'encendido' |
| Resultado Esperado | Luego de pronunciar en voz alta el comando, el switch cambia su estado a 'apagado'. Esto se refleja en la aplicación al cambiar la imagen (de una lámpara encendida a una lámpara apagada) y el texto del botón (de 'apagar' cambia a 'encender'). |
| Resultado Obtenido | <p>Antes de pronunciar el comando de voz para encender:</p>  <p>Luego de pronunciar el comando de voz para apagar:</p> |

| | |
|---------------------------------|---|
| |  |
| Resultado del Test (Pasó/Falló) | Pasó |

| | |
|--------------------|---|
| Nombre | TC07 |
| Escenario | Teniendo la aplicación abierta y conectada al switch, estando el switch en cualquier estado, al pronunciar un comando desconocido, el switch mantiene el estado |
| Pasos para el test | <ol style="list-style-type: none"> 1. Iniciar la aplicación 2. Esperar a que la pantalla de bienvenida termine de inicializar la aplicación 3. La pantalla principal se abre automáticamente y muestra el estado actual del Switch. 4. Pronunciar en voz alta cerca del microfono del telefono un comando de voz no reconocido por la aplicación (por ejemplo: 'hola como estas') |
| Datos para el test | <ul style="list-style-type: none"> • Aplicación configurada para conectarse a un switch |

| | |
|--------------------|--|
| | <p>en la direccion http://10.0.0.6:8080 (default)</p> <ul style="list-style-type: none">• El servidor del switch esta iniciado en la direccion http://10.0.0.6:8080• El switch se encuentra en estado 'encendido' |
| Resultado Esperado | Luego de pronunciar en voz alta el comando, la aplicación no lo reconoce, por lo tanto el switch no cambia su estado de 'encendido', en el cual se mantiene, e informa al usuario de que el comando no se reconoce |
| Resultado Obtenido | <p>Antes de pronunciar el comando de voz desconocido:</p>  <p>Luego de pronunciar el comando de voz desconocido::</p> |

| | |
|---------------------------------|---|
| |  |
| Resultado del Test (Pasó/Falló) | Pasó |

Demo

Se generó un video con una demo de la aplicación operando con el emulador del dispositivo Sonoff Switch R3 Basic. La misma puede visualizarse en el siguiente link:

<https://drive.google.com/file/d/1AT44K2x22rJBJEVrmnEz-Tmve6BD0dVV/view?usp=sharing>

ANEXO I

Descripción general del emulador de Switch Sonoff R3 Basic

Para el desarrollo y pruebas de la aplicación mobile, es necesario contar con un dispositivo real para poder probar la interacción de la aplicación con la API expuesta por el mismo. Sin embargo esto requiere de la adquisición de un switch (en caso de no contar con uno) y de la configuración del mismo para que funcione en modo programador (DIY). Para simplificar estas cuestiones de infraestructura, se decidió, (como Etapa 1 del Plan de Trabajo), implementar un emulador del Switch, que respete la interface (API) expuesta por el mismo, que se encuentra documentada en la página del fabricante. ⁷ De esta manera es posible independizarse de la necesidad de contar con infraestructura configurada, lo cual puede ser útil para futuros alumnos de la materia que escojan esta opción de TP. El código fuente del emulador se encuentra en el siguiente repositorio que podrá ser público si así se requiriera: <https://github.com/dicosta/SonoffR3Mock>

El emulador está desarrollado con Node.js y es esencialmente un servidor web HTTP similar al que provee el dispositivo físico, implementando el mismo protocolo (estilo REST) descrito por el fabricante. Como agregado de valor, el emulador también soporta protocolo WebSocket, de manera que al recibir una nueva conexión WebSocket, responde con el estado actual del dispositivo, y mediante un mecanismo de notificaciones interno (Event Bus), también envía el estado actual del dispositivo en todas aquellas ocasiones en que el dispositivo cambia su estado. Haciendo uso de esta habilidad de recibir conexiones WebSocket, acompaña al emulador una página web sencilla, a la cual se puede acceder desde cualquier navegador, y la cual muestra en tiempo real el estado actual del Switch (en la ruta /index.html).

Es interesante destacar que el emulador, si bien cuenta con persistencia de los cambios, al terminarse el proceso del servidor web se comporta como si en el switch real se cortara la corriente eléctrica, de manera de que hay un valor persistido que indica que debe suceder con el estado

del switch al restablecerse la corriente eléctrica. El switch puede pasar a estado encendido, o permanecer en estado apagado. Al iniciar el proceso nuevamente, se considera que es como si en el switch 'real' se reestableciera la corriente eléctrica. En tal caso, el switch modificará su estado, acorde al valor persistido.

Las prestaciones/servicios del emulador son:

| Ruta | Descripción |
|--------------------|---|
| '/switch' | Se utiliza para cambiar el estado actual del Switch de 'encendido' a 'apagado' o viceversa. [En caso de haber conexiones WebSocket, las mismas son notificadas mediante el envío del estado actual del switch] |
| '/startup' | Se utiliza para modificar el valor en el cual se inicializará el Switch la próxima vez que el proceso del servidor sea ejecutado (proxima ejecucion del servidor es lo que simula la situación donde el switch se queda sin suministro eléctrico y luego el mismo es reestablecido). [En caso de haber conexiones WebSocket, las mismas son notificadas mediante el envío del estado actual del switch] |
| '/signal_strength' | Utilizado para conocer el valor de la señal de la conexión Wifi en el Switch. En el emulador este valor no tiene un significado relevante, con lo cual se devuelve el valor default con el que se inicializa el modelo (-27) |
| '/pulse' | Activa o desactiva el modo pulso. Al activarlo, también debe configurar el ancho del pulso en milisegundos. [En caso de haber conexiones WebSocket, las mismas son notificadas mediante el envío del estado actual del switch] |
| '/wifi' | Utilizado para indicarle al Switch el SSID y el password de un Access Point al cual debe conectarse. En el emulador esto no es muy relevante, sin embargo se implementa para cumplimentar la |

| | |
|---------------|---|
| | interface. Actualiza el estado del switch con lo cual, en caso de haber conexiones WebSocket, las mismas son notificadas mediante el envío del estado actual del Switch. |
| '/ota_unlock' | Modifica el estado del switch, para permitir posteriores llamadas a '/ota_flash' mediante la cual se podrá instalar un nuevo firmware al Switch. No es relevante para el emulador, sino que se implementa para cumplimentar la interfaz, aunque si cambia el estado del Switch, de manera que si hay conexiones WebSocket, las mismas son notificadas mediante el envío del estado actual del Switch. |
| '/ota_flash' | Permite la instalación de un nuevo firmware en el Switch, que es descargado de la URL provista como parámetro en este método. No es relevante para el emulador, sin embargo se implementa para cumplimentar la interfaz. |
| '/info' | A través de este método se obtiene el estado actual del Switch. Retorna un objeto con todas las propiedades relevantes concernientes al estado actual del Switch. |

ANEXO II

Futuras investigaciones y líneas de trabajo.

La prueba de concepto implementada en la aplicación mobile implica que la aplicación conoce la ruta en la red de área local en la cual se encuentra el switch. Si la misma cambia, es requerida nuevamente al usuario al iniciar la aplicación, lo cual no es lo ideal.

Se podría implementar un mecanismo de Network Service Discovery, mediante el cual, el Switch ‘anuncia’ en la red que provee cierto servicio (identificado por un nombre, un tipo y un puerto) y el cual podría ser buscado por la aplicación mobile al iniciarse, con lo cual no sería necesario que el usuario indique la ruta del Switch, sino que la misma sería buscada y encontrada por la aplicación de manera automática mediante la definición de un protocolo indicando que nombre, qué tipo y que puerto buscar. Un ejemplo comercial de mecanismo de service discovery sería por ejemplo Bonjour, el cual provee librerías y API para implementar tanto el anuncio como la búsqueda en ambos extremos (tanto en el Switch como en la aplicación mobile). ^{14 15 16} [— — —](#)

Demo:

<https://drive.google.com/file/d/1AT44K2x22rJBJEVrmnEz-Tmve6BD0dVV/view?usp=sharing>

Informe:

https://docs.google.com/document/d/1IltZYomd_90nTb3w0It5sh6kRXu7EdeLey0ePO_eIHM/edit?usp=sharing

PPT Presentacion:

https://docs.google.com/presentation/d/1u7in_hdn-DrYWd2ARuap_k582MVvxq4587N17KXLwdA/edit?usp=sharing

Video Presentacion:

<https://www.youtube.com/watch?v=cW1u12VrOuU>

Repositorio Emulador:

<https://github.com/dicosta/SonoffR3Mock>

Repositorio App Mobile:

https://github.com/dicosta/7539_AI

APENDICE

Bibliografía y fuentes

<https://medium.com/descript/a-brief-history-of-asr-automatic-speech-recognition-b8f338d4c0e5> (1)

<https://astaspooks.wordpress.com/2014/10/13/audrey-the-first-speech-recognition-system/> (2)

<https://www.bbc.com/future/article/20170214-the-machines-that-learned-to-listen> (3)

https://en.wikipedia.org/wiki/Speech_recognition (4)

<https://medium.com/descript/the-state-of-automatic-speech-recognition-q-a-with-kaldis-dan-povey-c860aada9b85> (5)

https://en.wikipedia.org/wiki/Long_short-term_memory (6)

https://github.com/itead/Sonoff_Devices_DIY_Tools/blob/master/SONOFF%20DIY%20MODE%20Protocol%20Doc%20v2.0%20Doc.pdf (7)

<https://www.ibm.com/cloud/watson-speech-to-text/features> (8)

<https://www.speechmatics.com/wp-content/uploads/2019/03/Speechmatics-Product-Sheet-Real-time.pdf> (9)

<https://stuff.mit.edu/afs/sipb/project/android/docs/reference/android/speech/SpeechRecognizer.html> (10)

<https://cloud.google.com/speech-to-text> (11)

<https://alphacephei.com/vosk/> (12)

<http://kaldi-asr.org/> (13)

[https://en.wikipedia.org/wiki/Bonjour_\(software\)](https://en.wikipedia.org/wiki/Bonjour_(software)) (14)

https://en.wikipedia.org/wiki/Multicast_DNS (15)

https://en.wikipedia.org/wiki/Zero-configuration_networking (16)