UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Control domótico por voz

Proyecto Final de Carrera

Ingeniería Técnica En Informática de Sistemas

Autor: Javier Panta Martínez

Director: Sergio Sáez

27 de septiembre de 2012

Palabras clave: Arduino, microcontroladores, domótica, reconocimiento de voz, reconocimiento del habla, EasyVR.

Tabla de contenidos

1 Introducción

El presente escrito documenta la creación de un sistema para controlar un hogar mediante la voz. Concretamente, este documento detalla el proceso de análisis, diseño, implementación y pruebas de un prototipo de interfaz por voz que se conecta a un sistema domótico.

Dicho sistema domótico está en fase de desarrollo pero se supuso su existencia a lo largo de este proyecto. El sistema domótico estaría compuesto de un servidor domótico y partes móviles distribuidas dentro de un hogar. La implementación de estos elementos no forma parte de las tareas de este proyecto.

Existen varios proyectos que usan el reconocimiento de voz en distintos campos. Dentro de la escuela de informática de la UPV algunos de los trabajos más recientes son el desarrollo de una interfaz de usuario sin contacto que utiliza Arduino además del hardware Microsoft Kinetic (Belenguer, 2011); y el desarrollo de un sistema para usar un navegador web mediante la voz en plataformas Linux (Salvador, 2011). Fuera del ámbito de la UPV existen dos proyectos relevantes a este trabajo ya que utilizan el mismo hardware. Estos son el desarrollo de un robot humanoide (Borrello, y otros, 2011) y el desarrollo de unos auriculares inteligentes (Jiang, 2012). Otro trabajo interesante y que ilustra los distintos campos de aplicación de reconocimiento de la voz es el desarrollo y adaptación del reconocimiento de voz en aviones militares (Englund, 11th March 2004).

Este documento se ha agrupado en 6 capítulos. Los capítulos que forman el núcleo de esta memoria y que ilustran el flujo de trabajo seguido durante el desarrollo de este proyecto son: *Presentación del problema*, *Solución propuesta*, *Implementación* y *Resultados*. En la *Presentación de problema* se describe la situación inicial antes de comenzar el proyecto. En el capítulo de la *Solución propuesta* se explican varios aspectos sobre la solución diseñada, como por ejemplo la forma de uso del sistema, la arquitectura y la comunicación entre los distintos elementos. En el capítulo de la *Implementación* se detalla cómo se implementó la solución propuesta. Finalmente el capítulo de *Resultados* se muestra las pruebas a las que se sometió la solución implementada y se discuten los resultados.

Además, antes de los capítulos mencionados anteriormente se presenta el capítulo *Contexto* que expone información sobre el contexto tecnológico actual referente a los principales temas que toca este proyecto: el reconocimiento de voz y los microcontroladores. Finalmente, al final de esta memoria se presentan las conclusiones sobre el trabajo realizado.

2 Contexto

Los dos aspectos tecnológicos con más influencia en este proyecto han sido el reconocimiento de voz y los microcontroladores. Este proyecto trabaja con un sistema domótico, sin embargo no es un proyecto sobre domótica en sí, por tanto no se hablará sobre la domótica en este documento. Si se desea ver información sobre las principales arquitecturas de sistemas domóticos se recomienda leer el texto "Revisión de las arquitecturas de control distribuido" (Poza Luján, y otros, 2009).

2.1 Reconocimiento del habla

El campo del reconocimiento del habla es un campo en investigación constante y que aun presenta grandes desafíos. El problema podría ser abstraído a despejar la ambigüedad de unos datos recibidos, siendo los datos una señal recibida a través de un micrófono. En otras palabras se podría resumir la tarea que debe llevar a cabo el sistema como "dadas n opciones para una entrada de datos ambigua, elegir la más probable" (Jurafsky, y otros, 2009).

Los sistemas de reconocimiento de voz basan su funcionamiento principalmente en los siguientes modelos: las máquinas de estado, sistemas de reglas formales, modelos basados en la lógica de primer orden y los modelos probabilísticos. Los algoritmos que se usan sobre estos modelos son principalmente algoritmos de búsqueda de estados (como por ejemplo la programación dinámica) y algoritmos de aprendizaje de máquinas. Los modelos probabilísticos, los cuales sirven en este caso el propósito de despejar ambigüedad, puedan ser combinados con las máquinas de estados. El resultado de esto son los conocidos modelos de Markov, los cuales son omnipresentes en el campo de reconocimiento del habla. (Jurafsky, y otros, 2009).

En los últimos 10 años el campo del reconocimiento del habla ha dado grandes avances gracias al auge en la investigación en el campo del aprendizaje de máquinas, a una renovada atención a métodos estadísticos y a la gran potencia de los ordenadores actuales (Jurafsky, y otros, 2009). Durante esta última década es cada vez más común ver aplicaciones del reconocimiento del habla. Los usos más frecuentes son agentes telefónicos que interactúan con el usuario únicamente mediante la voz, juguetes, robots, y teléfonos inteligentes. Este último sector es el que está explotando y acercando más al público general el reconocimiento de voz. Compañías como Apple y Samsung están a la vanguardia con sus teléfonos inteligentes que tienen como una de sus principales bazas el reconocimiento de voz y la inclusión de agentes conversacionales (programas que no sólo reconocen el habla sino que interactúan con el usuario de forma oral y natural).

Típicamente existen dos modalidades en el reconocimiento del habla: reconocimiento dependiente del usuario, y reconocimiento independiente del usuario. Lo que estos términos quieren decir es que un sistema puede reconocer el habla de cualquier persona cuando se trata de reconocimiento independiente del usuario. En el caso del reconocimiento dependiente del usuario, el sistema sólo reconoce el habla de una persona determinada. Para este caso el sistema debe de haber sido entrenado con la voz y forma de hablar de dicha persona previamente. Hoy en día los sistemas de reconocimiento dependiente del usuario son

los más comunes. Pocas soluciones ofrecen un reconocimiento independiente del usuario y éstas suelen ser más caras.

Los principales actores comerciales en el campo del reconocimiento del habla son actualmente Nuance y Sensory Inc. Nuance se dedica principalmente a la comercialización de agentes conversacionales que sirvan de operadores en servicios prestados a través del teléfono. Además comercializa el programa Dragon NaturallySpeaking, el cual es el sistema reconocimiento de voz más usado en entornos de ordenadores de escritorio (Kim, 2011). Sensory Inc. ofrece soluciones software para el reconocimiento de voz en dispositivos móviles y otros sistemas electrónicos. Esta empresa también ofrece soluciones hardware en forma de chips dedicados al reconocimiento del habla que pueden ser empotrados en otros elementos (juguetes, robots, etc.). Sensory Inc. es la empresa detrás de los sistemas de reconocimiento de voz de los últimos teléfonos inteligentes de Samsung (Samsung unveils new Android phone with “S Voice” personal assistant, 2012).

2.2 Microcontroladores

Un microcontrolador es un sistema que integra en un mismo circuito integrado una unidad de procesamiento, memoria y pines para la entrada y salida de datos. Se puede pensar en un microcontrolador como un pequeño ordenador.

El primer microcontrolador fue inventado en el año 1971 por Texas Instruments. El nombre del microcontrolador era TMS 1000, trabajaba a 4 bits y tenía un tamaño de 3 × 3,6 mm (Stan Augarten, 1998 - 2009). La Figura 1 muestra una imagen ampliada de este microcontrolador.

Actualmente existe una gran cantidad de fabricantes de microcontroladores. Cada uno de estos fabricantes toma sus propias decisiones en cuanto las características de los microcontroladores que fabrican. Estas decisiones vienen dadas según el sector o propósito al que estén enfocados. Por tanto, existe una gran variedad en cuanto a las arquitecturas, los lenguajes en los que se programan y el tipo de funcionalidades que se incluyen dentro del chip. Actualmente, existen microcontroladores que trabajan en el rango de 4 a 64 bits (Toshiba). E incluso microcontroladores de cuatro núcleos que alcanzan 1600 MIPS (XMOS).

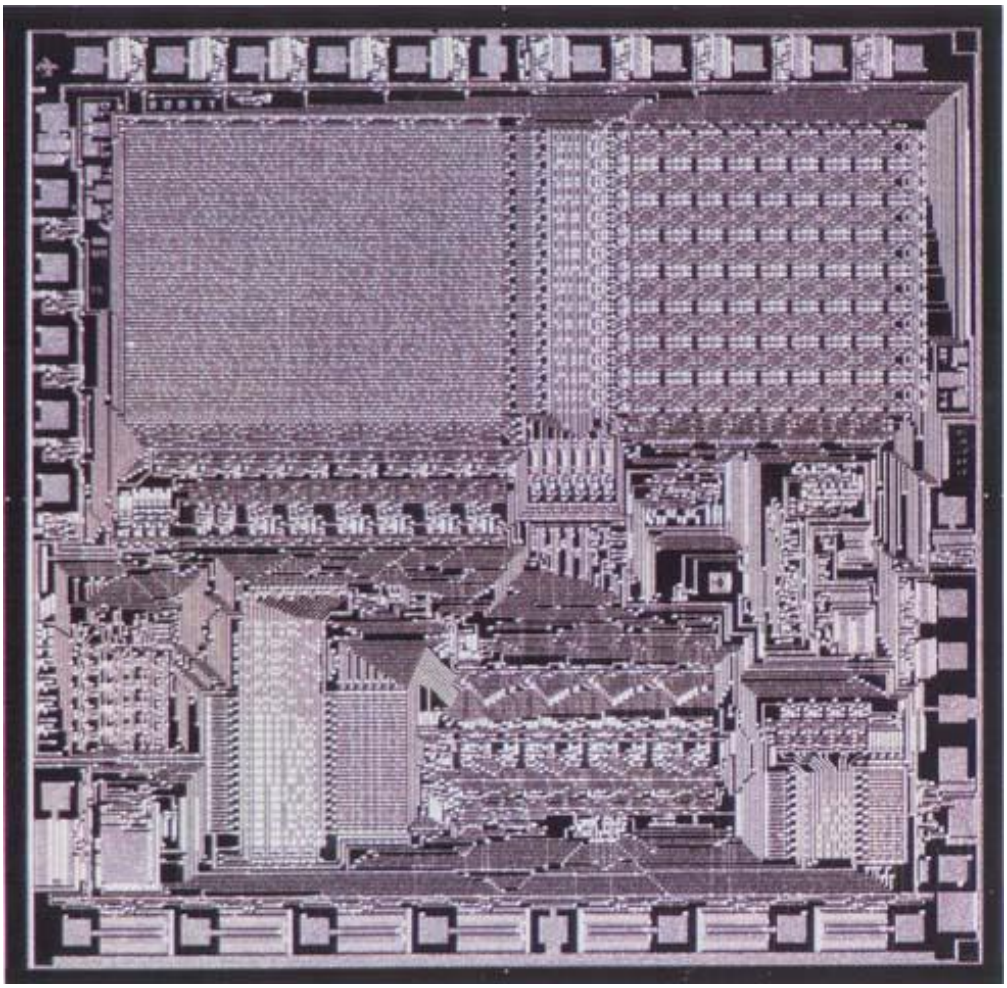

Figura 1: imagen ampliada del interior del microcontrolador TMS 1000. El primer microcontrolador del mundo.

2.2.1 Arduino

Arduino es una plataforma de prototipo de código abierto que nació en Italia en el año 2006. Arduino se compone de una placa que se basa en un microcontrolador, y un entorno de desarrollo para escribir el software para esta placa. Tanto los diseños del hardware como el código de su IDE están disponibles bajo licencias de tipo código abierto.

La documentación de Arduino describe bien su propósito:

> "Arduino es una herramienta para hacer ordenadores que puedan sentir y controlar el mundo físico a su alrededor [...]".

Tanto Arduino como otras plataformas similares ocultan y simplifican los detalles de programación de los microcontroladores en los que se basan a la vez que proporcionan interfaces de entrada /salida más comunes como puertos serie o puertos USB. De esta forma, estas plataformas hacen más sencillo y asequible el iniciarse en el uso de estos pequeños ordenadores.

Otras interfaces de entrada y salida que las placas de Arduino poseen son varios pines. Una de las características más importantes de Arduino es la disposición estándar de estos pines. Esta

disposición estándar permite conectarlo a otros módulos que aportan nuevas características al controlador. A los módulos que integran esta disposición estándar se les llama "Shields".

3 Especificación de requisitos

3.1 Terminología

En esta sección se utilizarán los siguientes términos:

- producto: sistema software a desarrollar.
- sistema o sistema domótico: se refiere al sistema del cual el producto es un subsistema.
- Servidor domótico: ordenador central del sistema domótico y con el cual se comunica directamente el producto.

3.2 Definición del Producto a desarrollar

El producto a desarrollar es una interfaz de reconocimiento de voz para un sistema domótico. La tarea principal de este producto será el de reconocer comandos de voz y convertirlos en comandos domóticos (comandos que entienda el sistema domótico). El principal objetivo de este producto es permitir a cualquier persona con capacidad de expresión oral controlar un hogar mediante la voz. Los comandos que el usuario emita deben contener el objeto que se pretende controlar, la acción que se pretende el sistema realice sobre el objeto, y el lugar donde se encuentra el objeto. Por ejemplo: "abrir persiana cocina". Las aplicaciones de este producto están en el campo de la domótica, siendo de especial interés para personas con movilidad reducida. El principal requerimiento de este producto es que use la plataforma Arduino para interactuar con los usuarios.

Las metas de este proyecto son la creación de:

- El software necesario para controlar el hardware de reconocimiento de voz.
- El software necesario que reciba en el lado del PC los resultados del hardware de reconocimiento de voz y los envíe como comandos domóticos al servidor domótico.
- Una API para controlar el hardware de reconocimiento de voz.
- Un cliente de línea de comandos que haga uso de la API para administrar el módulo de reconocimiento de voz.
- Un fichero XML (definición de la estructura y contenido) a partir del cual se puedan crear comandos de voz que permitan controlar un hogar determinado.

3.3 Descripción Global

3.3.1 Perspectiva del Producto

A fecha de este documento se está planeando la creación de un servidor domótico capaz de controlar un hogar a través de microcontroladores. Dicho servidor domótico forma parte de un completo sistema domótico que en el futuro permitirá controlar un hogar a través de distintas interfaces de usuario. Es aquí donde este producto entra en juego, ya que es una implementación de una interfaz de usuario para dicho sistema domótico. Además de este proyecto final de carrera se han planteado otros proyectos similares para la creación de interfaces a través de SMS y a través del sistema Android. La Figura 2 ilustra el sistema domótico.

El producto a desarrollar se comunica con el sistema domótico a través del servidor domótico. La comunicación entre el producto y el servidor domótico no ha sido definida completamente (en parte porque el servidor domótico aún no existe). De manera provisoria se ha establecido que el producto invocará a un programa que simula una interfaz con el servidor domótico cada vez que reconozca una orden domótica correcta. Además, el servidor debe proveer a la interfaz de un fichero XML a partir de cuyo contenido se pueda crear comandos de voz. La definición de la estructura y contenidos del fichero XML también forma parte de este proyecto.

Algunos aspectos que imponen restricciones son:

- Hardware: el módulo de reconocimiento de voz elegido para este proyecto es el denominado EasyVR (ver sección 4.1). Este hardware tiene las siguientes características que afectan al desarrollo del producto:
    - espacio para 32 comandos de voz.
    - Los comandos de voz se organizan en grupos. Para poder realizar cualquier operación sobre un comando de voz hay que indicar el grupo al que pertenece y la posición del comando de voz dentro del grupo.
    - Los comandos de voz son dependientes del usuario y por tanto precisan de ser entrenados para que puedan ser reconocidos. El sistema por tanto debe permitir entrenar los comandos de voz.
    - Se recomienda no entrenar los comandos de voz más de 2 o 3 veces.
- Interfaces hardware: la comunicación entre la placa de desarrollo Arduino y el PC debe ser llevada a cabo a través de un puerto USB que virtualmente se reconoce como un puerto serie. Por tanto, los protocolos que se vayan a implementar sobre este canal deben usar bytes dentro del rango de los códigos ASCII.
- Interfaces software: el servidor domótico utilizará también una placa de desarrollo de Arduino con un sistema Ubuntu Linux. Por tanto el producto a desarrollar debe poder ejecutarse sobre dicho sistema operativo.
- Operaciones: el producto deberá permitir un modo de funcionamiento en el cual se pueda controlar un hogar y un modo de funcionamiento en el cual sea posible ver, entrenar y recargar los comandos de voz.

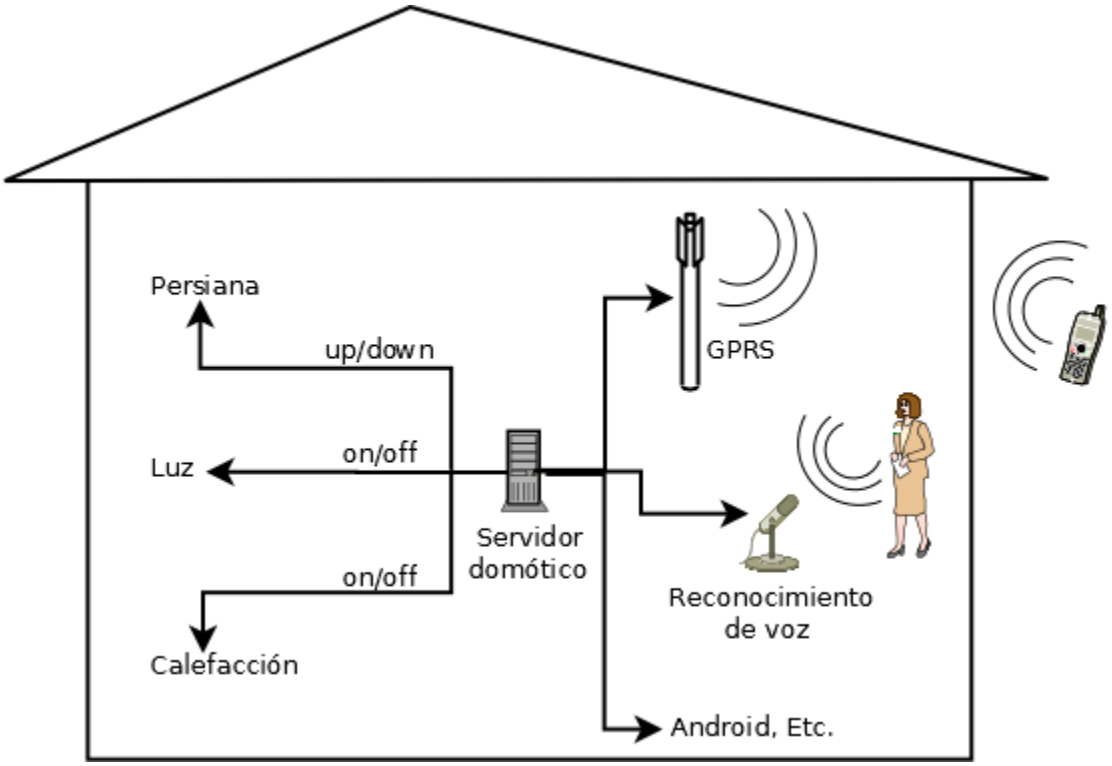

Figura 2: esquema del sistema domótico con algunos ejemplos de interfaces.

3.3.2 Tareas del Producto

Las principales funciones que el producto realiza para controlar los elementos de un hogar son las siguientes:

- realizar el reconocimiento de voz.
- construir el comando domótico a partir de uno o más reconocimientos de voz.
- Verificar la validez del comando domótico.
- Enviar el comando domótico al servidor.

Cuando se requiera entrenar o realizar otras operaciones sobre los comandos de voz se realizarán las siguientes operaciones:

- detener el reconocimiento de voz.
- tomar el control del hardware de reconocimiento de voz y dárselo a alguna aplicación cliente.
- Realizar la operación que el usuario indique (visualizar comandos, entrenar, borrar entrenamientos, recargar comandos).
- devolver el control del hardware de reconocimiento de voz a la placa de desarrollo.
- Reiniciar el reconocimiento de voz.

3.3.3 Perfil de los Usuarios

El público potencial de este producto es muy amplio y general. Se debe tener en cuenta que los posibles usuarios puedan tener las siguientes características:

- conocimiento técnico nulo
- experiencia técnica nula
- cualquier nivel educacional

La característica común de todos los usuarios es que tienen la capacidad para expresar sentencias orales imperativas.

3.3.4 Asunciones

Esta especificación de requisitos se ha basado en las siguientes asunciones:

- toda la información necesaria que el producto necesite del sistema domótico está disponible en el fichero XML.
- La interfaz para que el producto interactúe con el sistema domótico tiene la siguiente forma: *sendcmd objeto estado lugar*.
- el servidor domótico no envía ningún mensaje al producto.

3.4 Requisitos Específicos

3.4.1 Modo normal

3.4.1.1 Interfaces Externas

1. Interfaces de usuario:
   - programa de línea de comandos bash: A través de este interfaz un usuario podrá iniciar la ejecución del producto.
   - Micrófono: a través de un micrófono conectado al módulo de reconocimiento EasyVR los usuarios del producto emitirán órdenes. Por tanto, la naturaleza de la entrada de datos es la voz. Según la documentación del módulo EasyVR el micrófono no debe encontrarse una distancia de más de 30 cm.
2. Interfaces hardware:
   - interfaz UART entre el módulo EasyVR y la placa de desarrollo de Arduino: interfaz a través de la cual la placa de desarrollo Arduino controla al módulo EasyVR y otra vez del cual el módulo EasyVR responde a la placa Arduino. Los bytes permitidos a través de este interfaz son los pertenecientes al rango el código ASCII. El protocolo usado por el módulo EasyVR usa alguno de estos caracteres.
   - Interfaz serie entre placa de desarrollo de Arduino y el PC: las opciones se tiene para las placas de desarrollo se conectan con el PC a través de un cable USB. Virtualmente esta conexión es una conexión serie. Por tanto esto restringe los tipos de mensajes que se puede enviar (códigos ASCII) y el diseño del protocolo a usar.
3. Interfaces de Software:
   - librería del módulo EasyVR para la plataforma Arduino: el propósito de esta librería es proporcionar una forma fácil de comunicar a la plataforma Arduino con el módulo EasyVR. La librería implementa una clase que representa al módulo EasyVR. Los métodos de esta clase finalmente envían caracteres ASCII al módulo EasyVR para poder controlar a este.

- Programa Interfaz entre el producto y el servidor domótico: de forma provisional el producto debe invocar a un programa que representa una interfaz al servidor domótico. Esta interfaz tiene la siguiente forma: sendcmd objeto estado lugar.

4. Interfaces de comunicación:
   - protocolo de comunicación con el módulo EasyVR: este protocolo que debe usarse para la comunicación con el módulo de reconocimiento. La versión a fecha este documento es la que se documenta en el manual del módulo EasyVR versión 3. 2.

3.4.1.2 Requisitos Funcionales

Inicialización/Finalización de la Ejecución del Producto

1. Inicio del Microcontrolador:
   - evento de inicio: cuando se conecta el microcontrolador mediante el cable USB a un PC. A partir de este momento el microcontrolador recibe alimentación empieza ejecutar el código que esté cargado dentro de sí.
   - Funciones:
     - la placa Arduino debe de dar órdenes al módulo EasyVR para que este inicie el reconocimiento de los comandos de voz (presentes en la memoria del módulo EasyVR).
     - Siempre que el módulo EasyVR envía el resultado de un reconocimiento a la placa de Arduino, ésta debe de indicar al módulo EasyVR que inicie el reconocimiento nuevamente.
2. Inicio del software en el PC:
   - entrada: entrada por teclado en una consola bash "*nombre-del-producto start*".
   - Funciones:
     - iniciar todo los componentes que forman parte del software en el lado del PC.
     - Comprobar la comunicación con la placa de Arduino.
     - Demonizar la ejecución del software en el lado del PC. En este contexto se entiende demonizar según la definición de Richard Stevens (Stevens, 2005).
3. Final de la ejecución del software en el PC
   - entrada: entrada por teclado en una consola bash "*nombre-del-producto stop* "
   - Funciones: parar la ejecución de todos los componentes del software del lado del PC.

Utilización

1. Reconocimiento de Comando Domótico:
   - entrada: comando de voz de un usuario a través del micrófono el módulo EasyVR. El comando de voz debe contener un objeto, una acción y un lugar.
   - Funciones:
     - recibir el resultado del reconocimiento en la placa de Arduino.
     - Verificar si resultado es un resultado del éxito.
     - Al ser un resultado correcto, la placa de Arduino debe enviar el resultado al PC.

    - En el PC, verificar el reconocimiento se corresponde a una parte existente del hogar a controlar.
    - Llamar al programa que simula una interfaz con el servidor domótico.

2. Fallo de Reconocimiento de Comando Domótico

- entrada: comando de voz inexistente o incorrecto de un usuario a través del micrófono el módulo EasyVR.
- Funciones:
    - recibir el resultado del reconocimiento de la placa de Arduino.
    - Verificar si resultado es un resultado de éxito.
    - Al no ser un resultado de éxito se descarta el reconocimiento.
    - La placa de Arduino debe indicar al módulo EasyVR se inicia el reconocimiento otra vez.

3.4.2 Modo de administración

En este modo los clientes interactúan con el producto a través de un cliente. Dicho cliente es un programa de línea de comandos que forma parte también de este proyecto.

3.4.2.1 Interfaces Externas

Las interfaces externas son las mismas que están en funcionamiento en el modo normal del producto, con las excepciones de:

- el programa que simula interfaz con el servidor nunca interviene en este modo de funcionamiento.
- La consola bash que se usa en este modo de funcionamiento no tiene por qué ser la misma que la del modo de funcionamiento normal (desde la que se lanza el producto). Puede ser cualquier otra consola bash en un ordenador remoto.

3.4.2.2 Requisitos Funcionales

1. Ver un Listado de los Comandos de Voz:
    - entrada: orden "*listar identificador -grupo*" a través del programa cliente desde una consola bash.
    - Funciones:
        - cliente debe enviar una orden equivalente a la API.
        - la API debe enviar una orden equivalente al módulo EasyVR.
        - la API debe leer la respuesta del módulo EasyVR.
        - La API debe enviar la respuesta al programa cliente
    - salida: impresión de un listado de comandos de voz, del grupo solicitado, por pantalla.
2. Entrenar un comando de voz:
    - entrada: orden "*entrenar identificador -grupo identificador-comando*" a través del programa cliente desde una consola bash.
    - Funciones:
        - cliente debe enviar orden equivalente a la API.
        - La API debe enviar una orden equivalente al módulo EasyVR.
    - Entrada: voz de un usuario a través del micrófono del módulo EasyVR.

- Funciones:
    - el módulo EasyVR debe enviar el resultado a la API.
    - La API debe enviar el resultado al cliente de línea de comandos.
- Salida: impresión del resultado del entrenamiento por pantalla.

3. Borrado del entrenamiento de un comando de voz:
    - entrada: orden "*borrar-entrenamiento identificador -grupo identificador-comando*" a través del programa cliente desde una consola bash.
    - Funciones:
        - cliente debe enviar una orden equivalente a la API.
        - la API debe enviar una orden equivalente al módulo EasyVR.
        - la API debe leer la respuesta del módulo EasyVR.
        - La API debe enviar la respuesta al programa cliente
    - salida: impresión del resultado de la operación por pantalla.
4. Probar los comandos de voz:
    - entrada: orden "*probar identificador -grupo*" a través del programa cliente desde una consola bash.
    - Funciones:
        - cliente debe enviar una orden equivalente a la API.
        - la API debe enviar una orden para que el módulo EasyVR empieza a reconocer cualquier comando de voz del grupo especificado.
    - Entrada: comando de voz del usuario a través del micrófono del módulo EasyVR.
    - Funciones:
        - el módulo EasyVR debe enviar el resultado del reconocimiento a la API.
        - La API debe enviar el resultado al cliente de línea de comandos.
    - salida: impresión por pantalla del comando reconocido por el módulo EasyVR o del fallo de reconocimiento.

3.4.3 Requisitos del Diseño

El diseño del sistema se debe adaptar al esquema básico ilustrado en Figura 3.

Este diseño básico implica que el módulo de reconocimiento de voz se conecta a un ordenador donde existe un programa receptor que "escucha" los reconocimientos del módulo. Este programa es independiente del servidor domótico y su tarea es convertir los resultados del módulo de reconocimiento en órdenes que entienda el servidor domótico. Es decir, serviría de puente entre el módulo de reconocimiento de voz y el servidor domótico.

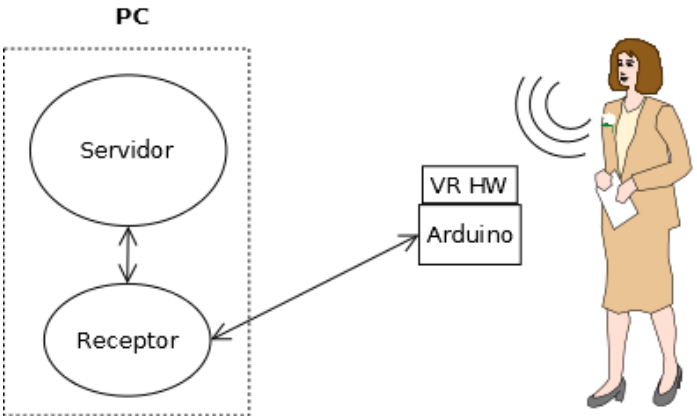

Figura 3: arquitectura básica requerida para el proyecto.

4 Solución propuesta

4.1 Hardware elegido

Se pensó en dos opciones para la plataforma de desarrollo: la placa uno de Arduino y la placa chipKIT Uno32 de Digilent. Esta última emula en factor de forma y funcionamiento a la placa uno de Arduino. Para una comparativa de cada placa se recomienda ver el Anexo A.

La placa elegida para el proyecto fue la placa Arduino UNO debido a dos motivos: la placa chipKIT no promete una compatibilidad total con el cien por cien de shields disponibles para Arduino, además existen pequeñas incompatibilidades en el compilador del IDE que provee el fabricante de estas placas (Burgess, 2011). Finalmente aunque el precio es idéntico para ambas opciones, para este proyecto no hacía falta la mayor capacidad de proceso que ofrecía chipKIT Uno32, ya que el reconocimiento de voz sería llevado a cabo por un chip específico y no por el microcontrolador.

Para el reconocimiento de voz se exploraron 3 opciones: el módulo EasyVR, el módulo SmartVR y el "MPLAB Starter Kit for dsPIC DSC". Para ver una descripción de estas 3 opciones ver el Anexo A.

Al final se eligió el módulo EasyVR en su formato Shield de la marca Veear. Este módulo es el único que está diseñado para trabajar desde el primer momento con placas de Arduino. Aunque es posible conectar el más potente SmartVR con una placa Arduino, la tarea no sería trivial y se requerirían de varios componentes intermedios (cables, breadboard, etc.) además de la adquisición de un micrófono y altavoces. El modo de conexión a través de Shield de EasyVR es simple y directo. Además, esta solución ofrece en una sola placa todos los componentes para crear un sistema de reconocimiento de voz: chip encargado de reconocimiento de voz, micrófono y salida de audio. El precio es otra ventaja de este módulo, al valer sólo 34 €, el precio total junto con la placa UNO de Arduino ronda los 50 €, siendo ésta la opción más económica. El módulo EasyVR sólo permite la creación de comandos de voz dependientes del usuario[1].

4.2 Uso del sistema

El funcionamiento básico del sistema permite a los usuarios controlar un hogar a través de la combinación de comandos de voz. Internamente el reconocimiento de voz llevado a cabo por el módulo EasyVR (controlado por la placa Arduino), los resultados de los reconocimientos de voz son enviados a un programa en el PC para que éste a su vez se los envíe al servidor domótico.

Sin embargo, El módulo EasyVR sólo soporta comandos de voz dependientes del usuario. Es decir, los comandos de voz para controlar un hogar deben de ser entrenados por la persona que vaya a usarlos. De no ser así el módulo de reconocimiento no responderá a nada de lo que diga el usuario. Esto impone el requerimiento de que debe haber una fase de entrenamiento

[1] La única solución comercial actual que permite la creación de comandos de voz independientes del usuario es el paquete SmartVR-DK PRO. Este paquete tiene un valor de 200 € y ofrece la posibilidad de crear comandos independientes del usuario a partir de texto. Es decir, sólo haría falta escribir el texto que se quiere reconocer para la creación de un comando, sin necesidad de entrenamiento.

en algún momento durante la vida útil del sistema. Por tanto se han diseñado dos modos de funcionamiento en el sistema a:

-modo normal: este es el modo de funcionamiento por defecto del sistema. En este modo el sistema está constantemente esperando las órdenes de voz de un usuario para controlar su hogar.

-Modo de administración: en este modo el usuario puede controlar el módulo EasyVR para entrenar y probar los comandos de voz que en él se encuentran. El control del módulo se realiza a través de una interfaz web o una interfaz de línea de comandos.

4.2.1 Cambio de modo de funcionamiento

Se ha pensado el sistema para que sea el usuario quien pueda lanzar un entrenamiento y posteriormente devolver el sistema al modo normal.

El usuario inicia el modo de administración abriendo la aplicación con la que se lleva a cabo el entrenamiento. Tras cerrar la aplicación, el sistema vuelve al modo normal. El cambio es simple y transparente para el usuario.

Para el caso en que se abra la aplicación para realizar el entrenamiento mientras se esté realizando un reconocimiento de un comando de voz, el modo normal tiene prioridad. Es decir, no debe ser posible iniciar el modo de administración mientras el modo normal esté ocupado.

4.2.2 Interacción con el sistema mediante la voz

Un usuario empieza a interactuar con el sistema diciendo una palabra clave. Si el sistema reconoce la palabra clave pasa a esperar la orden domótica que el usuario debe emitir seguidamente.

Para dar una orden domótica el usuario debe decir en una frase una acción y un elemento. Si el reconocimiento tiene éxito entonces el sistema pide al usuario la información del lugar. El usuario dice un lugar y si el reconocimiento tiene éxito se envía el resultado al PC.

Si se establece que la palabra clave es HAL la interacción con el sistema siempre se iniciaría de la siguiente forma:

- usuario: "HAL"
- sistema: "a sus órdenes"

Posteriormente cualquiera de los siguientes casos de uso puede ocurrir:

Caso de uso 1: el sistema reconoce todo lo que el usuario dice

- usuario: "encender luz"
- sistema: "¿dónde?"
- usuario: "cocina"
- sistema: "entendido"

Caso de uso 2: el sistema no reconoce lo que el usuario dice (en cualquier etapa del reconocimiento)

- usuario: *cualquier comando no válido*
- sistema: "no he entendido lo que me quiere decir"

El sistema tolerará esta situación un máximo de 3 veces.

Caso de uso 3: el usuario no dice nada (en cualquier fase del reconocimiento)

- usuario: ...
- sistema: " no he escuchado nada"

En este caso, se presupone que el usuario ha abandonado su intención de emitir una orden y el sistema vuelve al estado de espera, en el que sólo puede reconocer la palabra clave.

4.2.2.1 Justificación

El sistema podría haber reconocido órdenes domóticas en un sólo comando de voz pero Se ha elegido que el sistema use dos comandos de voz por dos razones:

-El sistema se vuelve interactivo. Esto lo hace más amigable frente al usuario.

-Se consigue un uso más eficiente de la memoria. El módulo de reconocimiento de voz EasyVR tiene espacio para 32 comandos de voz dependientes del usuario. Si en un sólo comando de voz se agrupase un elemento, una acción y un lugar el uso del espacio disponible aumentaría rápidamente. Por ejemplo, Para una casa de 4 habitaciones, si se quisieran controlar las luces de todas las habitaciones se tendrían los siguientes comandos:

1. encender luz habitación 1
2. apagar luz habitación 1
3. encender luz habitación 2
4. apagar luz habitación 2
5. encender luz habitación 3
6. apagar luz habitación 3
7. encender luz habitación 4
8. apagar luz habitación 4

En total ocho comandos. Los cuales habría que guardar en la memoria del módulo de reconocimiento de voz ocupando un cuarto de la memoria disponible. Es fácil darse cuenta del uso ineficiente de memoria de este método: se guarda varias veces el mismo dato en varias posiciones de memoria. Por ejemplo, "encender luz" y "apagar luz" se repiten 4 veces cada uno. Además, cada habitación se repite dos veces.

Esta situación empeora si tan sólo se agrega un nuevo elemento que pueda ser controlado por el servidor domótico. Por ejemplo, si se añade el control de persianas de cada habitación de pronto se tendrían ocho comandos más, sumando junto con los anteriores un total de dieciséis comandos de voz que ocuparían la mitad de la memoria disponible.

Separando cada comando de voz en dos: "acción-elemento" y "lugar". Y combinando estos para la generación de órdenes completas se consigue un uso más eficiente de la memoria. Para el primer ejemplo haría falta guardar en memoria seis comandos voz: cuatro habitaciones

y dos acciones-elementos. La combinación de estos daría las ocho órdenes necesarias. Para conseguir controlar las persianas sólo habría que añadir al sistema dos comandos de voz: "subir persianas" y "bajar persiana". Esto haría un total de 8 comandos de voz: cuatro habitaciones y cuatro acciones-elementos. La combinación de estos da un total de dieciséis órdenes domóticas. La mayor cantidad de órdenes domóticas que es posible implementar usando este método sería el de combinar dieciséis acciones-elementos y dieciséis lugares lo que hace un total de 256 órdenes domóticas.

4.2.3 Interfaz para el entrenamiento de los comandos de voz

El entrenamiento de los comandos de voz se puede llevar a cabo a través de dos interfaces de usuario: un cliente de línea de comandos y un cliente web. Ambos clientes permiten visualizar los comandos de voz existentes e iniciar el entrenamiento de cualquiera de ellos. Además del cliente de línea de comandos permite releer la información relativa a los elementos y lugares del hogar que es posible controlar.

Las aplicaciones cliente controlan el sistema de forma remota. El módulo de reconocimiento de voz podría empotrarse en una pared o un mueble alejado de cualquier PC. Por esto es conveniente que un usuario puede controlar el sistema de forma remota. De esta forma por ejemplo, un usuario que quisiese entrenar el sistema podría acercarse al módulo e iniciar el reconocimiento desde su teléfono inteligente.

4.3 Arquitectura

4.3.1 Principales elementos

Existen cuatro elementos principales dentro del sistema:

-módulo EasyVR y placa Arduino: el conjunto de estos dos elementos hardware se encarga del reconocimiento de voz. La lógica implementada en la placa Arduino controla el módulo EasyVR permitiendo un uso del sistema tal como se ha descrito en el apartado 4.2.2.

-Entidad receptor: Esta es una entidad software que reside en el PC. Recibe los resultados del módulo de reconocimiento de voz, convierte estos resultados en órdenes domóticas y se las envía al servidor domótico. Esta entidad entra en funcionamiento durante el modo normal del sistema.

-Entidad API: esta es una entidad software que reside en el PC. La API implementa métodos para controlar el módulo EasyVR de forma efectiva. El propósito principal es poder realizar entrenamientos de los comandos de voz, sin embargo a través de la API se pueden controlar muchos más aspectos.

-Capa de comunicación: esta es una entidad software que reside en el PC. Permite comunicar receptor y a la API con la placa de Arduino.

La Figura 4 muestra la arquitectura propuesta del sistema incluyendo a una aplicación cliente utilizando la API de forma remota:

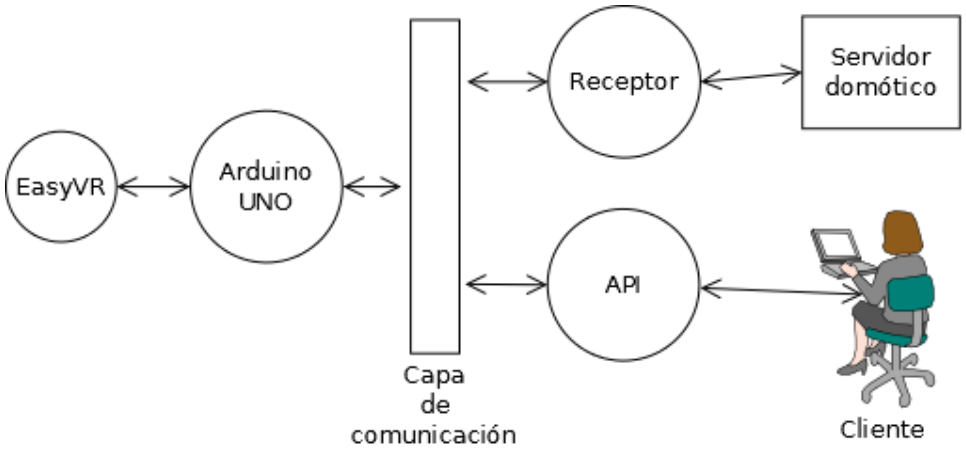

Figura 4: arquitectura propuesta.

4.3.2 Datos

Otro elemento importante en la arquitectura es el "almacén de datos". La tarea de esta entidad es leer y almacenar los datos relativos a los elementos del hogar que pueden ser controlados por el sistema domótico. Como se mencionó antes estos datos son recibidos en formato XML desde el servidor domótico.

El almacén de datos es usado por el receptor y por la API. Por el receptor para verificar la coherencia de los comandos de voz emitidos por los usuarios. Por la API para cargar los datos del fichero XML y ponerlos como comandos de voz en el módulo EasyVR. Esta última acción borrar el contenido anterior de la memoria del módulo.

4.3.3 Comunicación

Como se ha mencionado antes la placa de Arduino se comunica con el PC a través del canal serie. Por tanto se tiene que las dos entidades comparten el mismo canal para comunicarse con un solo elemento (la placa uno de Arduino). Los datos que se reciban a través del canal serie en el PC deben de ir a parar a la entidad correspondiente según el modo de funcionamiento. El canal serie no ofrece ninguna característica para facilitar esa tarea. Es por tanto necesaria la existencia una capa de comunicación en el PC que desvía a la entidad correspondiente lo que Arduino envíe al PC.

En el modo de funcionamiento normal, la capa de comunicación envía todos los datos que recibe del microcontrolador al receptor.

Se consigue entrar y salir del modo de administración gracias a que la capa de comunicación inspecciona los datos recibidos tanto desde la API como desde el microcontrolador. Si la capa de comunicación detecta que la API envía una petición de establecimiento de conexión a Arduino, la capa de comunicación esperará entonces el carácter de confirmación de establecimiento de conexión. Una vez reciba dicho carácter enviará los datos procedentes de la placa Arduino a la API y no al receptor. Si el carácter que se recibe es distinto la capa de comunicación seguirá enviando los caracteres procedentes del microcontrolador al receptor.

Para el cierre de de conexión ocurre el proceso inverso. Si la capa de comunicación detecta que la API ha enviado el carácter de cierre de comunicación al microcontrolador, la capa de comunicación empieza a desviar los caracteres otra vez hacia el receptor.

4.4 Protocolo

Para la comunicación entre los principales elementos del sistema se han diseñado los siguientes protocolos

4.4.1 Modo normal

El módulo EasyVR envía los resultados de los reconocimientos que realiza al programa en la placa de Arduino. Cuando se ha reconocido una acción, un elemento y un lugar; Arduino envía estos datos a la entidad receptor. El receptor no envía nada a Arduino.

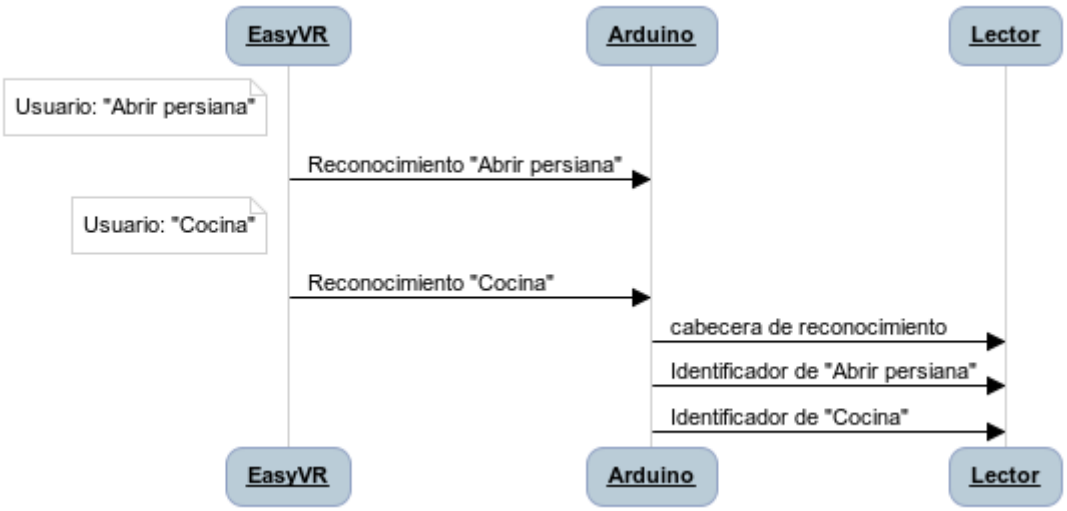

Figura 5: protocolo de comunicación después del reconocimiento de un comando domótico en el modo normal de funcionamiento

En el protocolo entre la placa de Arduino y la entidad receptor se han usado los siguientes caracteres:

- cabecera de reconocimiento: carácter "r". Código ASCII 114
- identificadores: caracteres ASCII en el rango desde la "A" (a mayúscula) al "´" (acento invertido). Códigos ASCII 65 al 96. Estos caracteres son los mismos que envía el módulo EasyVR a la placa Arduino cuando realiza un reconocimiento de voz.

4.4.2 Modo de administración

Las entidades que se comunican en este modo de funcionamiento son la API, el programa en la placa Arduino y el módulo EasyVR. El propósito de la API es que se comunique directamente con el módulo EasyVR. Antes de que esto ocurra la API debe negociar con la placa Arduino para que este le deje comunicarse directamente con la placa EasyVR. Durante la duración de la conexión la API se comunica directamente con el módulo EasyVR. La Figura 6 indica este proceso de establecimiento de conexión.

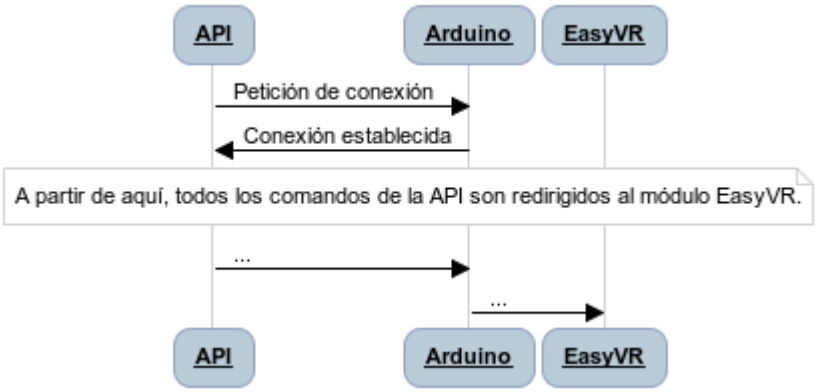

Figura 6: protocolo de establecimiento de conexión entre la API y el módulo EasyVR.

En el caso de que un usuario esté usando activamente el sistema en el modo normal, la secuencia es la siguiente:

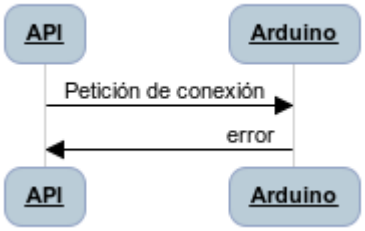

Figura 7: fallo en el establecimiento de conexión.

Cuando el programa cliente (a través de la API) finalice su uso del módulo EasyVR, deberá hacer una llamada a través de la API para negociar el fin de la conexión entre la API y el módulo EasyVR. El protocolo seguido es el que se indica en la siguiente figura:

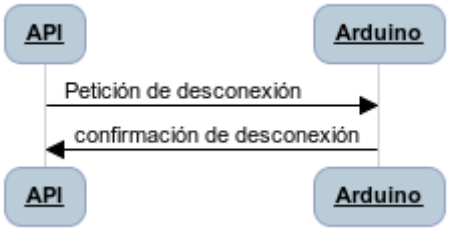

Figura 8: protocolo de cierre conexión entre la API y el módulo EasyVR

Los siguientes caracteres se han usado en el diseño el protocolo:

- Petición de conexión: carácter "!”. Código ASCII 33.
- Confirmación de conexión establecida: carácter "!”. Código ASCII 33.
- Petición de desconexión: carácter "@". Código ASCII 64.
- Error en la conexión/desconexión: carácter '"' (comillas). Código ASCII 34.

Si en el establecimiento de conexión o en el cierre conexión la API no recibe respuesta después un de un cierto tiempo se da por supuesto que la conexión no se ha realizado.

5 Implementación

El código implementado se encuentra en la carpeta code. Se usó el lenguaje de programación python versión 2.7 para el desarrollo del software en el lado del PC. Al ser un lenguaje interpretado, el programa puede ejecutarse en cualquier plataforma sobre la que funcione python. Otras razones para la elección de este lenguaje es que permite un desarrollo muy rápido gracias a las soluciones que trae en sus librerías estándar y a su simple sintaxis.

Se decidió que las entidades descritas en la arquitectura del sistema fueron hilos. Estos hilos son lanzados al inicio del programa por el hilo principal del programa. El código del hilo principal se encuentra en el fichero PC/main.py. El hilo principal del programa también se encarga de crear las colas necesarias para la comunicación del receptor y la API con la capa de comunicación. Concretamente se necesitan 3 colas:

listener_queue: cola utilizada para enviar datos de la capa de comunicación hacia el receptor.

API_in_queue: cola utilizada para enviar datos de la capa de comunicación hacia la API.

API_out_queue: cola utilizada para enviar datos de la API hacia la capa de comunicación.

5.1 Capa de comunicación

La capa de comunicación comunica al programa en el lado del PC con la placa Arduino. Más concretamente desvía los datos provenientes del canal serie a la entidad correspondiente (el receptor o la API).

La implementación de la capa de comunicación se encuentra en el fichero communication.py dentro de la carpeta code/PC. Se ha implementado esta entidad usando dos hilos. Un hilo para encargarse de la lectura de datos provenientes de la placa Arduino y otro hilo para escribir datos a Arduino. La razón por la que se han usado dos hilos es porque se necesitaba una lectura y escritura asíncronas. La lectura de datos provenientes de Arduino es constante, siempre se esperan datos. En cambio, la escritura sólo ocurre cuando un usuario lo solicita a través de la API. En cada hilo se hacen lecturas bloqueantes. Es decir, lecturas que detienen la ejecución del hilo hasta que llega un dato[2].

El establecimiento de conexión entre la API y el módulo EasyVR es el único caso en el que tiene que haber sincronía entre los dos hilos. Esto es así ya que se ha diseñado el proceso de conexión de forma que cuando la API envía una petición de conexión o desconexión esta debe esperar inmediatamente a recibir una confirmación. De no ser así la conexión no se establece.

El pseudocódigo de la capa de comunicación es el siguiente:

Hilo que Lee de Arduino:

while true

[2] Realmente se podría haber usado un bucle que comprobara secuencialmente la existencia de un dato a leer de Arduino o de un dato a leer desde la API. Así fue la primera versión que se hizo de esta entidad pero consumía muchos recursos (50% de la CPU). Esto es probablemente debido a la naturaleza de los bucles en python tal como se puede ver en http://wiki.python.org/moin/PythonSpeed/PerformanceTips#Loops.

- valor = lectura bloqueante desde Arduino
- si petición_API:
    - si valor es confirmación de conexión/desconexión
        - marcar conexión como activa/inactiva
- si conexión con API esta activa
    - enviar valor a la API
- si no
    - enviar valor al receptor

Hilo que escribe a Arduino:

while true

- valor = lectura bloqueante desde API
- si valor es igual a petición de conexión/desconexión:
    - petición_API =true
- enviar valor a Arduino

5.2 Implementación del modo normal

En este modo el sistema espera una palabra clave del usuario. Tras reconocer correctamente la palabra clave el usuario debe decir una frase que contenga una acción y un elemento. Si se reconoce correctamente esta frase el usuario debe decir un lugar. Una vez reconocido el lugar, el microcontrolador envía al receptor tres caracteres ASCII. El primer carácter es la cabecera del mensaje ('r' para comunicaciones correctas). El segundo carácter representa un objeto y la acción que el usuario mencionó. El tercer carácter representa un lugar.

Adicionalmente, si no se reconoce correctamente cualquiera de las dos últimas frases el sistema pide al usuario que repita. El sistema repite este último comportamiento un máximo de 3 veces.

El receptor lee los 3 caracteres que le envía el microcontrolador, verifica que el elemento exista en el lugar y envía la orden domótica al servidor domótico.

5.2.1 Microcontrolador

La implementación de esta parte del sistema se encuentran en el fichero vr.ino, dentro de la carpeta code/Arduino/vr. El fabricante del módulo EasyVR provee una librería para poder trabajar en el código del microcontrolador con un objeto que representa el módulo EasyVR. El código del microcontrolador implementado en este proyecto usa una versión modificada de esta librería. Los dos principales métodos de los que se ha hecho uso son recognizeCommand() y hasFinished(). El primero inicia el proceso de reconocimiento de voz. El segundo sirve para comprobar si el módulo EasyVR ha terminado con el proceso de reconocimiento de voz. Es decir, la llamada para iniciar reconocimiento de voz no es bloqueante, el microprocesador puede desempeñar otras tareas mientras el módulo EasyVR está intentando reconocer un comando de voz.

El comportamiento que se busca implementar tiene un patrón. Para el reconocimiento de un comando domótico el programa realiza tres veces la misma secuencia acciones. Cada una de estas tres repeticiones se puede ver como tres estados:

-primer estado: se entran a este estado cuando el microcontrolador indica al módulo de reconocimiento que reconozca la palabra clave.

-segundo estado: se entra a este estado cuando se ha reconocido la palabra clave y se le indica al módulo de reconocimiento que reconozca una acción.

-tercer estado: se entra a este estado cuando se ha reconocido una acción y se le indica al módulo de reconocimiento que reconozca un lugar.

Como se puede apreciar, los 3 estados realizan la misma operación básica: reconocer algo. Lo que cambia en cada estado es lo que se reconoce y el estado al que se pasa si el reconocimiento ha tenido éxito. Teniendo en cuenta lo anterior se puede modelar el sistema como una máquina de estados:

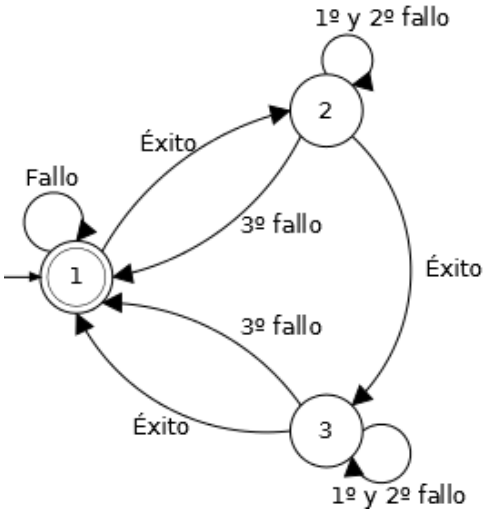

Figura 9: máquina de estados que representa el comportamiento del código en la placa de Arduino.

Para plasmar está modelización se ha hecho que el código realice la misma operación básica (reconocer comandos voz y actuar sobre el resultado) con distintos argumentos según la evolución del programa. Esta evolución del programa es guiada por una variable de tipo entero denominada " step". Los argumentos a usar en cada estado se guardan en vectores. El vector más importante es el vector groups, el cual almacena la información sobre los grupos de comandos de voz a reconocer. Así por ejemplo, cuando el sistema se encuentra en el primer estado la variable step tiene el valor cero. En dicha iteración del programa se utilizará la posición cero del vector groups.

En cuanto al flujo del programa, cuando el programa empieza, el primer reconocimiento de voz es iniciado en la función setup. Para esto, la variable step se inicializa al primer estado del sistema. Posteriores reconocimientos de voz son siempre iniciados al final del bucle loop. Antes de iniciar estos reconocimientos se actualiza el valor de la variable step.

A continuación se puede apreciar el pseudocódigo y la estructura del código:

setup()

- step =1
- iniciar reconocimiento del estado "step"

loop()

- si el módulo de reconocimiento ha terminado:
  - si el reconocimiento tuvo éxito:
    - si el sistema acaba de atravesar el primer estado (se ha reconocido la palabra clave): reproducir mensaje "a sus órdenes"
    - si el sistema acaba de atravesar el segundo estado (se ha reconocido una acción): reproducir mensaje "¿dónde?"
    - si el sistema acaba de atravesar el tercer estado (se ha reconocido un lugar): enviar el resultado al PC y reproducir mensaje "entendido"
    - actualizar la variable step
  - si no:
    - step = 1
- iniciar reconocimiento del estado "step"

El pseudocódigo anterior muestra el comportamiento básico del sistema. Hay dos aspectos que por simplicidad no muestra. Estos aspectos son:

-cuando una acción o un lugar no son reconocidos por el sistema, el sistema pide al usuario que repita. Esto se da un máximo de 3 veces.

-Si el sistema está esperando una acción o un lugar y no escucha nada durante 5 segundos, vuelve al primer estado.

5.2.2 Receptor

La implementación del receptor se encuentra en el fichero listener.py en la carpeta code/PC. El receptor es un hilo que recibe datos y comprueba que estos se correspondan a una combinación válida de acción, elemento y lugar. Si esto es así invoca a un script que envía el comando domótico al servidor domótico.

Los datos recibidos llegan a través de una cola (objeto de python) que el receptor recibe en el momento que es creado. Para comprobar que los datos recibidos forman una combinación válida el receptor usa un método implementado en el almacén de datos.

Para que el script se ejecute en un proceso distinto se hace uso de la librería subprocess. El script recibe como parámetros los identificadores de un objeto, una acción y un lugar.

Los identificadores de los objetos, acciones y lugares, así como el nombre del script son provistos por el almacén de datos a través de sendas funciones.

El pseudocódigo es el siguiente:

while true

- esperar a que la cola de datos tengo un dato disponible y leer un carácter
- si carácter es la cabecera correcta:
    - leer dos caracteres más
    - si es una combinación válida de la acción y lugar:
        - conseguir identificadores de la acción, el elemento y el lugar
        - conseguir el nombre del script
        - ejecutar script con los identificadores como parámetros

5.3 Implementación del modo de administración

En este modo una aplicación cliente puede controlar el módulo EasyVR. Para iniciar este modo el usuario sólo deberá abrir la aplicación cliente. Internamente el sistema se encarga de suspender el modo normal y de dar el control del módulo EasyVR a la aplicación cliente. Al cerrar la aplicación cliente el modo normal vuelve al iniciarse. Si el modo normal ya ha reconocido la palabra clave y está esperando un lugar o una acción el modo de administración no podrá iniciarse.

5.3.1 Microcontrolador

El microcontrolador actúa como un puente de comunicación entre la aplicación en el PC y el módulo EasyVR. Es decir, el código en el microcontrolador envía todo lo que reciba por el canal serie del PC a el módulo EasyVR y viceversa. Este comportamiento se repite en un bucle que se inicia cuando se recibe el carácter de inicio de conexión (!). Este bucle se rompe al recibirse el carácter de cierre conexión (@).

El código que realiza esta funcionalidad se aloja dentro de la función serialEvent() en el fichero vr.ino dentro de la carpeta code/Arduino/vr. Esta función pertenece a la librería para la comunicación por el puerto serie que provee Arduino. El código que se defina dentro de esta función será ejecutado cada vez que se detecte que hay un carácter de disponible en el canal serie de entrada y cuando haya terminado una iteración de la función loop()[3].

Para implementar la prioridad del modo normal sobre el modo de administración se ha hecho uso de una variable llamada module_busy. Esta variable indica que en el modo normal ya se ha reconocido la palabra clave. En el caso de que sea así, se le enviará a la API un mensaje que indica que el sistema está ocupado. El pseudocódigo y la estructura del código se presentan a continuación:

serialEvent()

- leer mensaje del PC
- si el mensaje es "abrir conexión":
    - si module_busy
        - enviar a PC mensaje "ocupado"
    - si no
        - parar modo normal

[3] por tanto, para que este mecanismo funcione de forma eficaz es necesario que cuando el módulo de reconocimiento este intentando reconocer algún comando de voz la ejecución del programa no queda bloqueado. Ver sección 5.2.1.

- while(1)
    - si hay mensaje del PC
        - si mensaje es "fin de conexión": romper bucle
        - si no enviar a EasyVR mensaje del PC
    - si hay mensaje de EasyVR
        - enviar a PC mensaje EasyVR

Para parar el modo normal se llama al método break de la clase EasyVR.

5.3.2 API

La función principal de la API es permitir a los usuarios entrenar comandos de voz a través de programas clientes. La API realmente controla directamente el módulo de reconocimiento de voz por lo que puede realizar más acciones que sólo entrenar comandos.

La implementación de la API se encuentran en el fichero interface.py en la carpeta code/PC . La API es una clase con métodos para controlar fácilmente al módulo EasyVR. Internamente cada uno de estos métodos se basa en los métodos que proporciona una clase llamada EasyVR y que se explica en el siguiente sub-apartado.

La API es expuesta a través de un servidor XML-RPC. Este servidor se ha implementado usando la librería SimpleXMLRPCServer.

Algunos de los métodos que ofrece la API son las siguientes:

-reload: ordena cargar los contenidos del fichero XML del servidor domótico, transformarlos y guardarlos como comandos de voz en el módulo EasyVR.

-list_commands: lista los lugares y los objetos que pueden controlarse a través de la voz. Este es el único método que no trabaja con el módulo EasyVR si no con el almacén de datos.

-list_group (grupo): lista los comandos de voz existentes en el módulo EasyVR del grupo especificado (acciones o lugares).

-train (grupo, comando): entrena el comando especificado dentro del grupo especificado.

-erase_training (grupo, comando): borra el entrenamiento del comando especificado a del a dentro del grupo especificado.

-test (grupo) : inicia el reconocimiento de voz del grupo especificado (acciones o lugares).

Existen además dos métodos que deben ser invocados para conmutar entre los dos modos de funcionamiento del sistema:

-EasyVR_connect: este método debe ser llamado antes que cualquier otro método. Envía un carácter especial al microcontrolador que sirve para iniciar el modo de administración (establece una comunicación directa con el módulo EasyVR según el protocolo explicado en la sección 4.4). Cualquier otro método que se llame antes de este método será ignorado por el microcontrolador y no será enviado al módulo de reconocimiento de voz.

-EasyVR_disconnect: este método sirve para terminar la conexión con el módulo EasyVR. Al igual que el método anterior envía un carácter especial que el microcontrolador interpreta como fin de la comunicación. Después de llamar a este método el sistema entra en el modo normal.

5.3.2.1 Clase EasyVRclass

Esta clase implementa métodos para controlar de forma más atómica el módulo EasyVR. Esta es la clase en la que se basa la API. La implementación de esta clase encuentra en el fichero interface.py en la carpeta code/PC. Internamente cada método envía los caracteres necesarios para dar órdenes al módulo EasyVR según el protocolo descrito en la documentación de este[4].

Los métodos de la clase EasyVR realizan operaciones básicas en el módulo EasyVR. Por ejemplo uno de los métodos es dump_command (group, index), el cual sirve para obtener información sobre el comando "index" del grupo "group". Sucesivas llamadas al mismo método con distintos argumentos podría servir para la creación de un listado de comandos de voz que luego podría ser imprimido por pantalla.

5.3.2.2 Clase cache

La implementación de esta clase se encuentra en el fichero cache.py dentro de la carpeta code/PC. En versiones iniciales del sistema la API siempre se comunicaba con la clase EasyVR para obtener un listado de los comandos de voz. Para conseguir este listado había que hacer tantas llamadas al método dump_command (group, index) como comandos de voz existieran. Dado que cada una de estas llamadas implica al menos la transferencia de 7 caracteres (incluyendo la respuesta) esta operación resultaba muy lenta. Estos retrasos no son aceptables si se tienen además en cuenta los retrasos de las comunicaciones a través del protocolo XML-RPC.

Ya que la información sobre los comandos de voz sólo cambia en tres casos específicos, esta información (etiquetas, cantidad de entrenamientos y conflictos con otros comandos) es guardada ahora en la clase cache.

La información sobre cada comando de voz se guarda en un diccionario (con campos label, count, is_conflict, conflict). Los diccionarios de los comandos que pertenecen a un mismo grupo se agrupan en una misma lista. Finalmente todas las listas de todos los grupos existentes se agrupan dentro de otra lista. Esta última es un atributo público de esta clase y es leída por la API para que esta pueda llevar a cabo sus tareas.

El único caso en que la etiqueta de un comando cambia es cuando se recargan los datos del fichero XML al módulo EasyVR. En cuanto a la información sobre el entrenamiento de los comandos de voz, estos cambian en dos ocasiones:

-Se entrena un comando .

-se elimina el entrenamiento de un comando.

[4] la documentación del módulo EasyVR está disponible en la carpeta a documentation

Cuando el valor de las etiquetas de los comandos de voz cambia se llama al método refresh_group(group_index) desde la API. Esto se hace una vez por cada grupo de comandos de voz que exista en el módulo EasyVR. Este método actualiza toda la información de los comandos del grupo especificado. Es decir las etiquetas, la cantidad de entrenamientos y conflictos con otros comandos.

En los casos en que cambia la información referente al entrenamiento de los comandos de voz, habrá que llamar posteriormente desde la API al método refresh_training (group_index, command_index) para el comando de voz que se haya visto afectado. Este método actualiza la información en la cache sobre la cantidad de entrenamientos y conflictos con otros comandos.

5.4 Fichero XML

El sistema de interfaz de usuario recibe un fichero en formato XML del servidor domótico con información sobre las acciones que éste puede llevar a cabo. Este fichero debe tener el nombre commands.xml para que sea leído por el almacén de datos y debe encontrarse en la misma carpeta que el código del almacén de datos.

Se ha diseñado el fichero para que describa un hogar y para que éste pueda reflejar fácilmente cualquier ampliación de elementos o lugares que pueda controlar el servidor domótico.

Se definen dos entidades básicas en el fichero: los objetos y los lugares. Cada una de estas entidades tiene un identificador y una etiqueta. El identificador es una cadena en inglés, la etiqueta es una cadena en español.

Cada lugar tiene además una lista de objetos que existen en el lugar (por ejemplo, los objetos luz y persiana pueden existir en el lugar cocina). Esta lista está formada por identificadores de lugares, es decir los lugares se referencian no se redefinen.

Cada objeto tiene una lista de acciones que soporta. Cada una de estas acciones tiene también un identificador en inglés y un texto en español.

Existe una etiqueta y un identificador en cada lugar, objeto y acción con el propósito de hacer al sistema fácilmente internacional. El propósito de las etiquetas es el de ser usado por los programas cliente de cara a los usuarios. Por tanto deben describir el elemento al que pertenecen en el idioma de los usuarios. Los identificadores de los objetos en cambio estarán siempre en inglés ya que el sistema usa estos identificadores para operaciones internas.

En el Anexo C se muestra el fichero XML que se ha usado como ejemplo durante las fases de prueba del sistema.

5.5 Almacén de datos

El almacén de datos es una entidad que almacena los datos contenidos en el fichero XML en estructuras especiales y ofrece métodos para que el receptor y la API usen esos datos.

La implementación de esta entidad se encuentra en el fichero data.py. El almacén de datos está implementado como una clase que se ejecuta dentro de un hilo. En la implementación se ha hecho uso de la librería ElementTree. Dicha librería permite representar elementos XML en

una estructura especial llamada Element. Además, permite representar los anidamientos de los elementos XML a través de una estructura en forma de árbol llamada ElementTree.

Al inicio del programa, todo el contenido del fichero XML es guardado en una estructura ElementTree. Un ejemplo de esta estructura con los datos del fichero se muestra en la Figura 10.

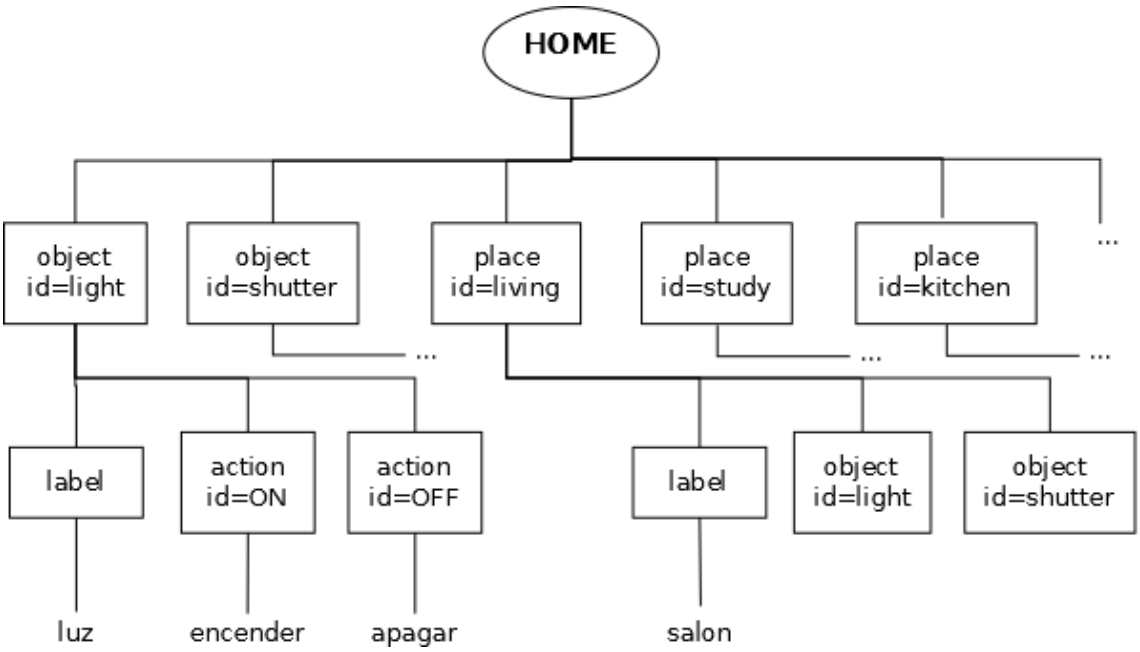

Figura 10: estructura ElementTree con los datos del fichero XML de ejemplo.

Además, referencias a los elementos que representan los lugares y los objetos son guardados en sendas listas. Esto se hace es porque más fácil acceder a los objetos y los lugares si éstos se encuentran en listas que si se encuentran dentro de un árbol el cual hay que recorrer.

Los métodos que el almacén de datos pone a disposición del receptor en el modo normal son los siguientes:

valid_combination (VR_action_index, place_index): recibe dos enteros y devuelve un booleano. Los dos enteros que recibe representan las posiciones de los comandos de voz en el módulo EasyVR. El primer índice representa un comando de voz que agrupa una acción y un objeto. Como por ejemplo "abrir persiana". El segundo índice representa un comando de voz de lugar como por ejemplo "cocina". El valor devuelto es verdadero si la combinación de los dos índices es una combinación válida y falso en caso contrario.

get_entities (self,VR_action_index, place_index): recibe dos enteros y devuelve 3 cadenas. Los dos enteros que recibe representan las posiciones de los comandos de voz en el módulo EasyVR. El primer índice representa un comando de voz que agrupa una acción y un objeto. Como por ejemplo "abrir persiana". El segundo índice representa un comando de voz de lugar como por ejemplo "cocina". Las cadenas devueltas son identificadores que usa el servidor domótico. Respectivamente los identificadores devueltos son identificadores de objeto, acción y lugar.

Para convertir el índice lugar en un identificador de lugar se ha hecho lo siguiente: EL orden de la lista de lugares se corresponde con la tabla de comandos de vos de lugares del modulo EasyVR. Por tanto para conseguir el identificador del comando de voz con índice i solo hay que leer la posición i de la lista de lugares.

Para conseguir los identificadores de un objeto y una acción no pasa lo mismo. Para este caso se tiene que un solo comando de voz de agrupa una acción y un objeto. Para conseguir estos identificadores primero hace falta descomponer el comando de voz en cuestión. Es decir, a partir de un índice se debe conseguir dos identificadores. Para esto se ha usado el siguiente algoritmo.

Argumento a = índice de comando de voz acción + objeto

Si a es par: índice de acción = 0; índice de objeto = a/2

Si a es impar: índice de acción = 1; índice de objeto = (a-1)/2

El índice de objeto indica la posición del objeto en la lista de objetos del almacén de datos. El índice de acción indica la posición de la acción en la lista de acciones (creada al vuelo para cada objeto). Una vez se sabe a qué objeto y acción se está haciendo referencia se extraen sus etiquetas.

Los métodos que el almacén de datos pone a disposición de la API en el método de administración devuelven cadenas que los programas cliente muestran al usuario. Los métodos son los siguientes:

get_actions: devuelve una lista de cadenas. Cada cadena es una concatenación de un objeto y sus acciones correspondientes. Por ejemplo para un objeto persiana cuyas acciones sean abrir y cerrar, se devolverán las cadenas "abrir persiana" y "cerrar persiana". La lista contiene todas las concatenaciones posibles para todos los objetos del sistema.

get_places: devuelve una lista de cadenas. Cada cadena es la etiqueta de un lugar. La lista contiene todos los lugares existentes en el sistema.

list_commands: devuelve un objeto diccionario. Cada entrada del diccionario es identificado por una etiqueta de lugar. Cada entrada contiene una lista de objetos que existen en el lugar correspondiente.

5.6 Aplicaciones Cliente

Para controlar el módulo de reconocimiento y obtener datos de este, las aplicaciones cliente hacen uso de la API del sistema (expuesta a través de un servidor XML-RPC). Para esta comunicación los clientes hacen uso de la librería xmlrpclib, la cual permite la llamada de procedimientos remotos de manera sencilla.

5.6.1 Intérprete de línea de comandos

La implementación de este cliente se encuentra en el fichero client.py . Para la implementación se ha usado la librería cmd.py. Esta librería ofrece en varias facilidades para el

desarrollo rápido de clientes de línea de comandos. Con esta librería el desarrollo se reduce a implementar la lógica para cada comando que exista en el cliente de línea de comandos.

Los siguientes comandos implementados:

- home: permite ver los elementos que es posible controlar en el hogar usando la voz. Esta información se presenta en forma de una lista de las de habitaciones de un hogar y los objetos dentro de ellas que es posible controlar.

- list (actions|places): muestra un listado de los comandos de voz correspondientes.

-train (actions|places) (0..31): inicia el entrenamiento del comando de voz especificado.

-erase_training (actions|places) (0..31): borra el entrenamiento del comando de voz especificado.

-test (actions|places) : inicia el proceso de reconocimiento de voz del grupo especificado.

-reload: vuelve a cargar los comandos de voz del fichero XML. Se leen los datos del fichero XML y se convierten en comandos de voz.

Adicionalmente se ha implementado comandos de ayuda que muestran una descripción para cada comando de los listados anteriormente.

5.6.2 Cliente web

La implementación de este cliente se encuentra en la carpeta django\mysite\EasyVR. El desarrollo se ha llevado a cabo usando el framework Django.

Este cliente es una aplicación web que dota al sistema de una interfaz gráfica. A través de este cliente se puede ver la lista de acciones y lugares existentes en el módulo EasyVR. Además, es posible entrenar y borrar el entrenamiento de estos comandos de voz, y también es posible probar ambos grupos (iniciar el reconocimiento de voz para cada uno). La Figura 11 muestra dos capturas de pantalla de este cliente.

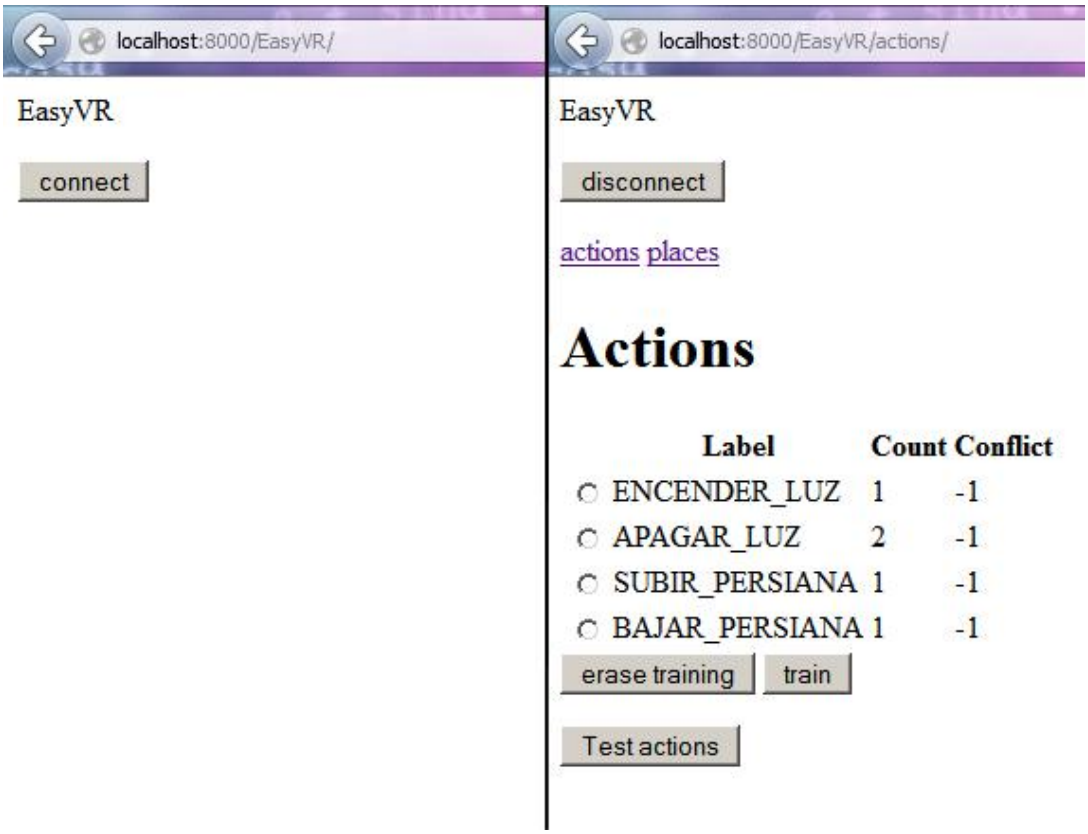

Figura 11: a la izquierda, la pantalla de inicio de la interfaz web. A la derecha listado de acciones después de conectar con el módulo EasyVR.

6 Resultados

6.1 Pruebas

En su versión final el sistema funciona como un demonio, por tanto el sistema no imprime nada por pantalla. Para comprobar la evolución del funcionamiento del programa hay que verificar el log del sistema, y para verificar los comandos domóticos que el sistema produce hay que verificar el fichero de salida que produce el script sendcmd.py. Debido a que el sistema funciona interactuando con los usuarios a través de la voz no era posible automatizar el proceso de pruebas.

6.1.1 Inicio del Sistema

Para que el sistema empieza funcionar hay que invocar al demonio con el argumento *start*. La siguiente figura muestra un inicio del sistema y la información que se genera en el log.

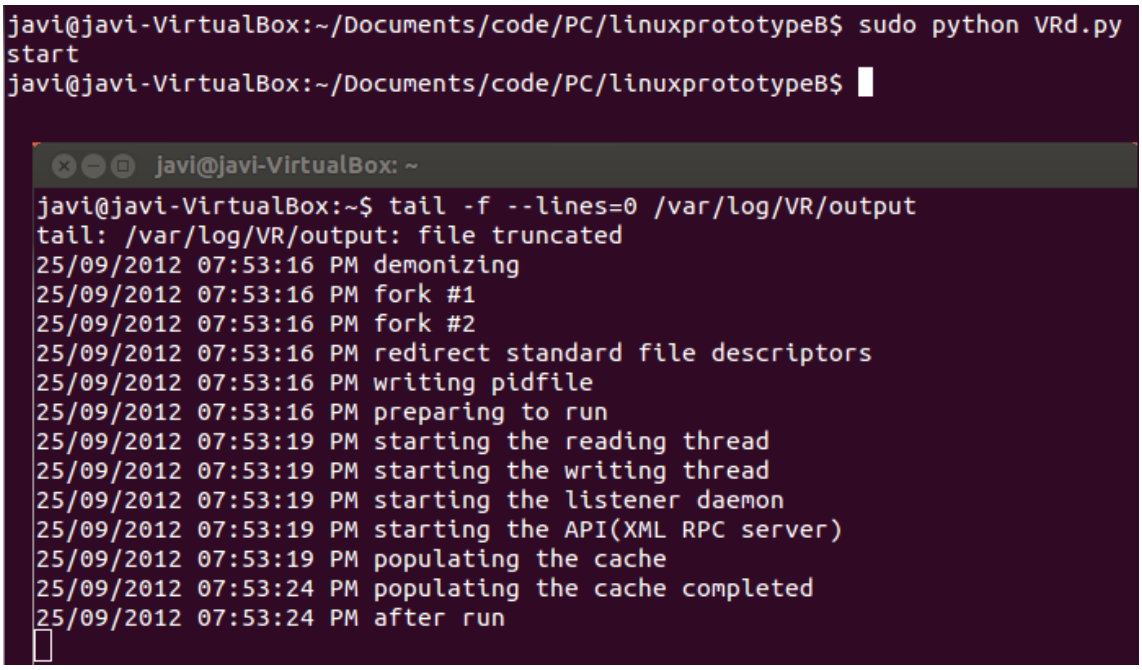

Figura 12: inicio del sistema e información que se escribe en el log del sistema.

6.1.2 Modo normal

Para probar el modo de funcionamiento normal se decían todas las posibles combinaciones de órdenes domóticos al sistema. A continuación se presentan algunos ejemplos:

6.1.2.1 Ejemplo Éxito

En este ejemplo el usuario dijo la siguiente secuencia de comandos de voz:

usuario: "hal”
sistema: "a sus órdenes"
usuario: "subir persiana"
sistema: "¿donde?"
Usuario: "estudio"
sistema: "entendido"

usuario: "hal”
sistema: "a sus órdenes"
usuario: "encender luz"
sistema: "¿donde?"
Usuario: "baño"
sistema: "entendido"

usuario: "hal”
sistema: "a sus órdenes"
usuario: "apagar luz"
sistema: "¿donde?"
Usuario: "cocina"
sistema: "entendido"

El sistema produjo los comandos domóticos que se muestran en la imagen siguiente:

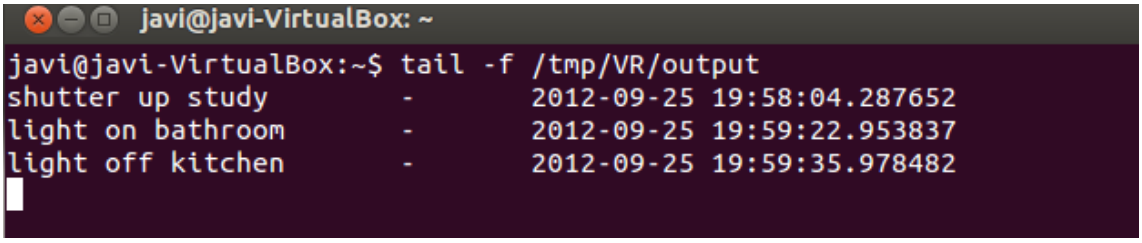

Figura 13: comandos domóticos producidos por el sistema.

En el log del sistema se escribió la siguiente información:

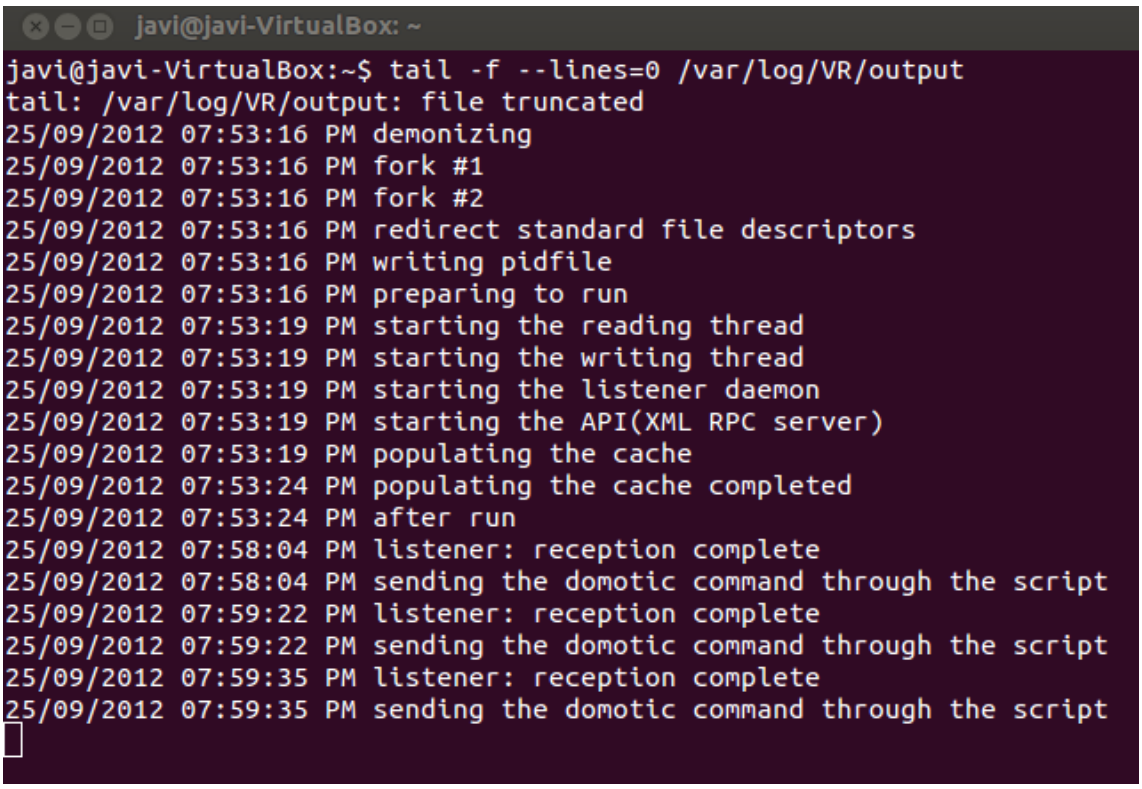

Figura 14: log del sistema durante la generación de comandos domóticos.

6.1.2.2 Usuario abandona su intención de emitir un comando domótico

usuario: "hal”
sistema: "a sus órdenes"
usuario: "subir persiana"
sistema: "¿donde?"
sistema: "no he escuchado nada"

En este caso, al no reconocerse ninguna orden domótica Arduino no envía nada al PC. Por tanto no se genera nada después del la interacción anterior y el fichero de salida del sistema y el log quedan en el mismo estado que se muestra en la Figura 13 y en la Figura 14.

6.1.2.3 El usuario dice un comando que el sistema no entiende

usuario: "hal”
sistema: "a sus órdenes"
usuario: "encender televisión" (comando inexistente)
sistema: "repita por favor"
usuario: "encender televisión"
sistema: "repita por favor"
usuario: "encender televisión"
(El sistema deja de preguntar y vuelve a su estado inicial)

En este caso ocurre lo mismo, no se reconoce ninguna orden domótica y por tanto Arduino no envía nada al PC.

6.1.3 Modo de Administración

Durante el desarrollo del sistema, cada vez que había un cambio mayor se introducía a través del cliente de línea de comandos la siguiente secuencia de comandos:

- list actions
- list places
- home
- test all
- reload
- train all
- erase_training (actions |places) 0..31 //borrar el entrenamiento de una acción o lugar cualquiera
- train (actions |places) 0..31 //entrenar una acción o lugar cualquiera
- test all

Los comandos train all y test all fueron implementados exclusivamente para realizar estas pruebas. El comando train all inicia secuencialmente el entrenamiento de todas las acciones y todos los lugares. El comando test all inicia el reconocimiento primero para las acciones y luego para los lugares en un bucle que se repite 4 veces.

Adicionalmente, se probaban los siguientes casos:

- inicio de las aplicaciones cliente cuando el sistema estaba apagado.
- Apertura de las aplicaciones cliente cuando el modo normal estaba ocupado.

La implementación actual del sistema ha superado todas las pruebas. A continuación se presentan algunos los comandos introducidos y su respectiva salida:

6.1.3.1 Cliente de Línea de Comandos

En la siguiente figura se demuestra la funcionalidad del comando home y el comando reload. En este ejemplo, después de de invocar al comando home (para ver los objetos que se pueden controlar en cada lugar) se modificó el fichero XML. La modificación del fichero XML se puede ver en el Anexo C. Después de dicho modificación se invocó el comando reload. Dicho comando vuelve leer el fichero XML y crea los respectivos comandos de voz en el módulo EasyVR. Como se puede apreciar en la sucesiva llamada al comando home, la información dentro del programa se actualiza.

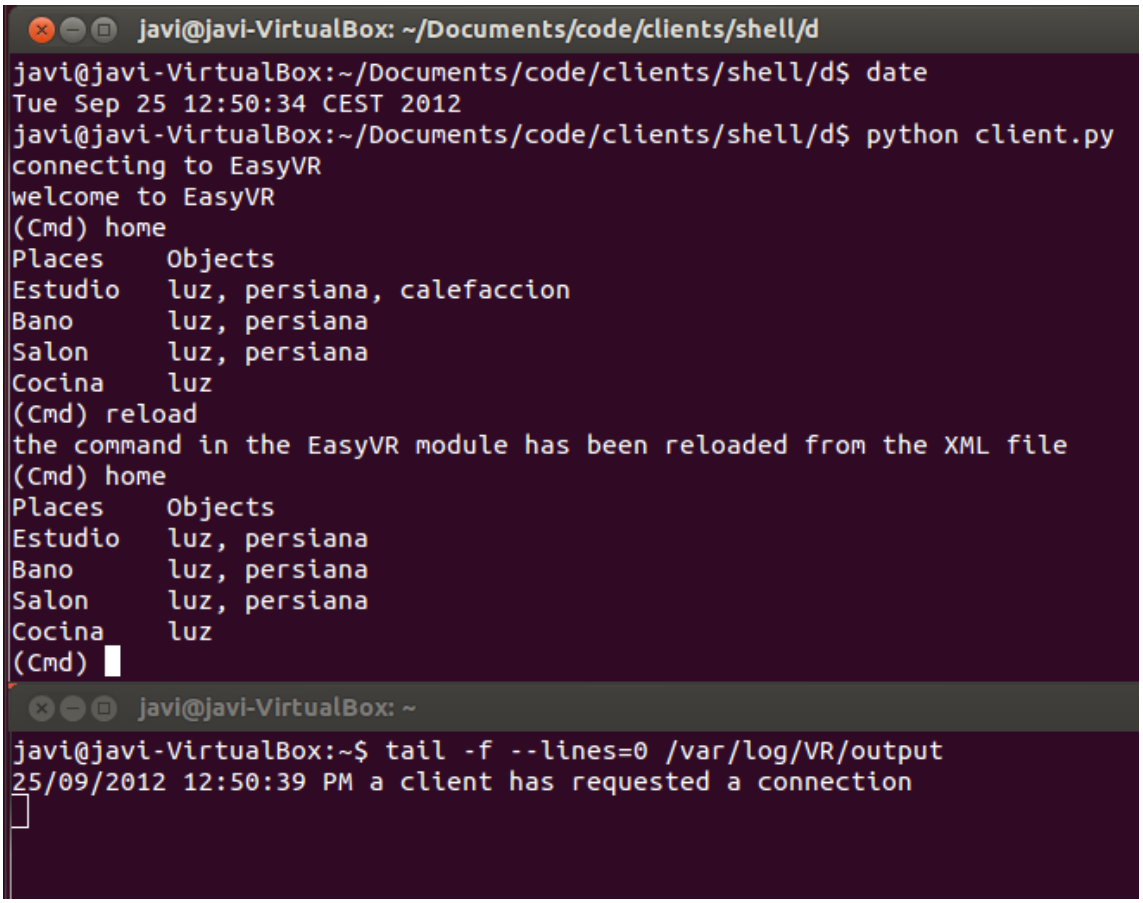

Figura 15: probando el comando "reload" desde el cliente de línea de comandos.

Se puede apreciar además la entrada en el fichero del log que es creada cuando un cliente se conecta al sistema.

La siguiente figura muestra el mismo caso, pero esta vez también muestra los comandos de voz que se crean en el módulo EasyVR. Es posible ver los comandos de voz dentro de dicho módulo con el comando list(actions |places |trigger). Para el caso siguiente se muestran las acciones. Para cada comando de voz se muestra su índice, su etiqueta, las veces que este comando ha sido entrenado y conflictos con otros comandos de voz (lo que ocurre cuando los comandos de voz son parecidos).

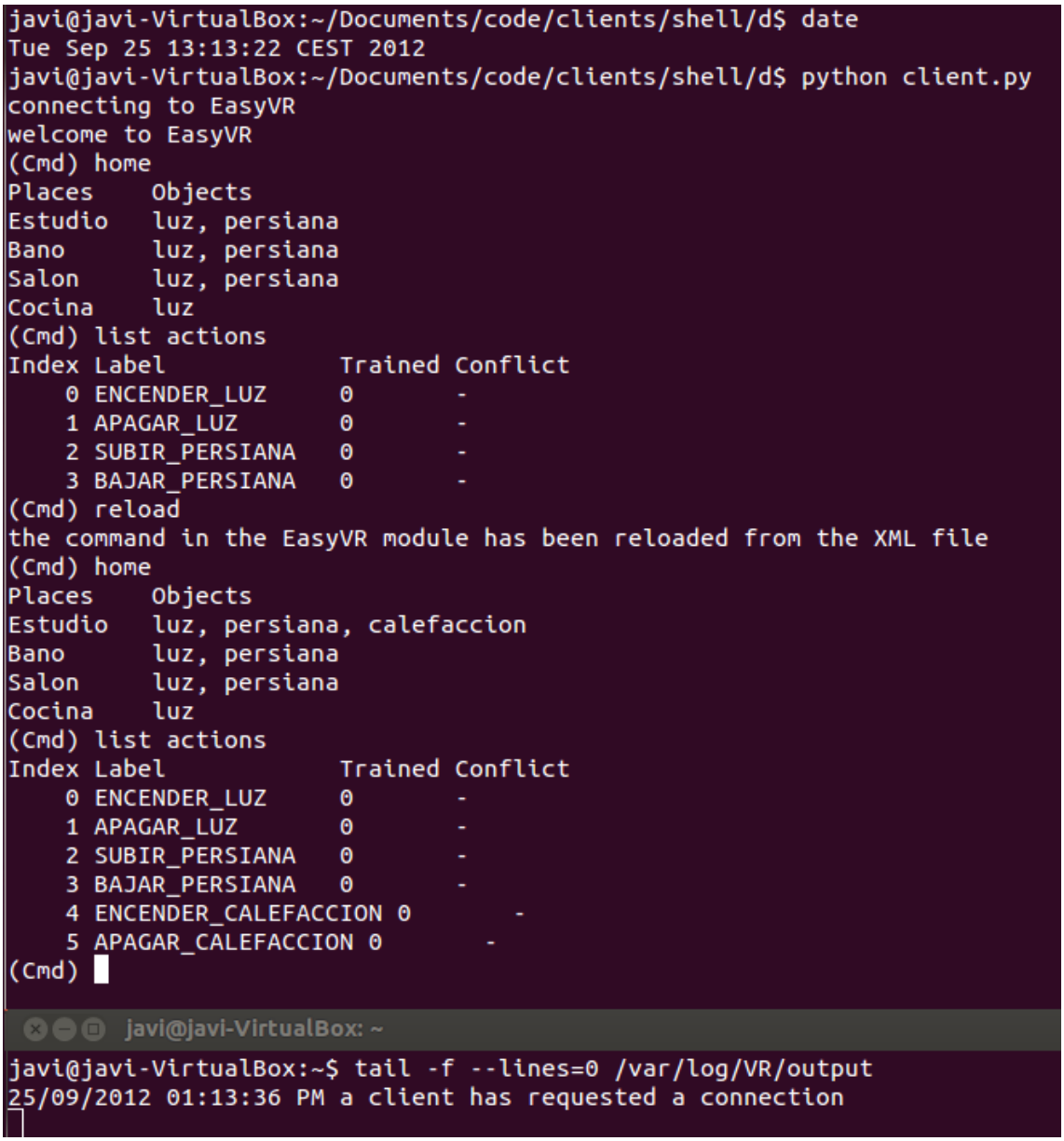

Figura 16: recargando los comandos de voz desde el fichero XML y posterior visualización.

El siguiente ejemplo muestra los comandos ingresados por línea de comandos para realizar el entrenamiento de las acciones. Concretamente se entrenan la acción con índice 0 y luego la acción con índice 1. Posteriormente se muestra el listado de las acciones para ver cómo aumenta la cuenta del número de entrenamientos. Además, se muestra cómo se sale del cliente de línea de comandos y como este cambio queda reflejado en el log del sistema.

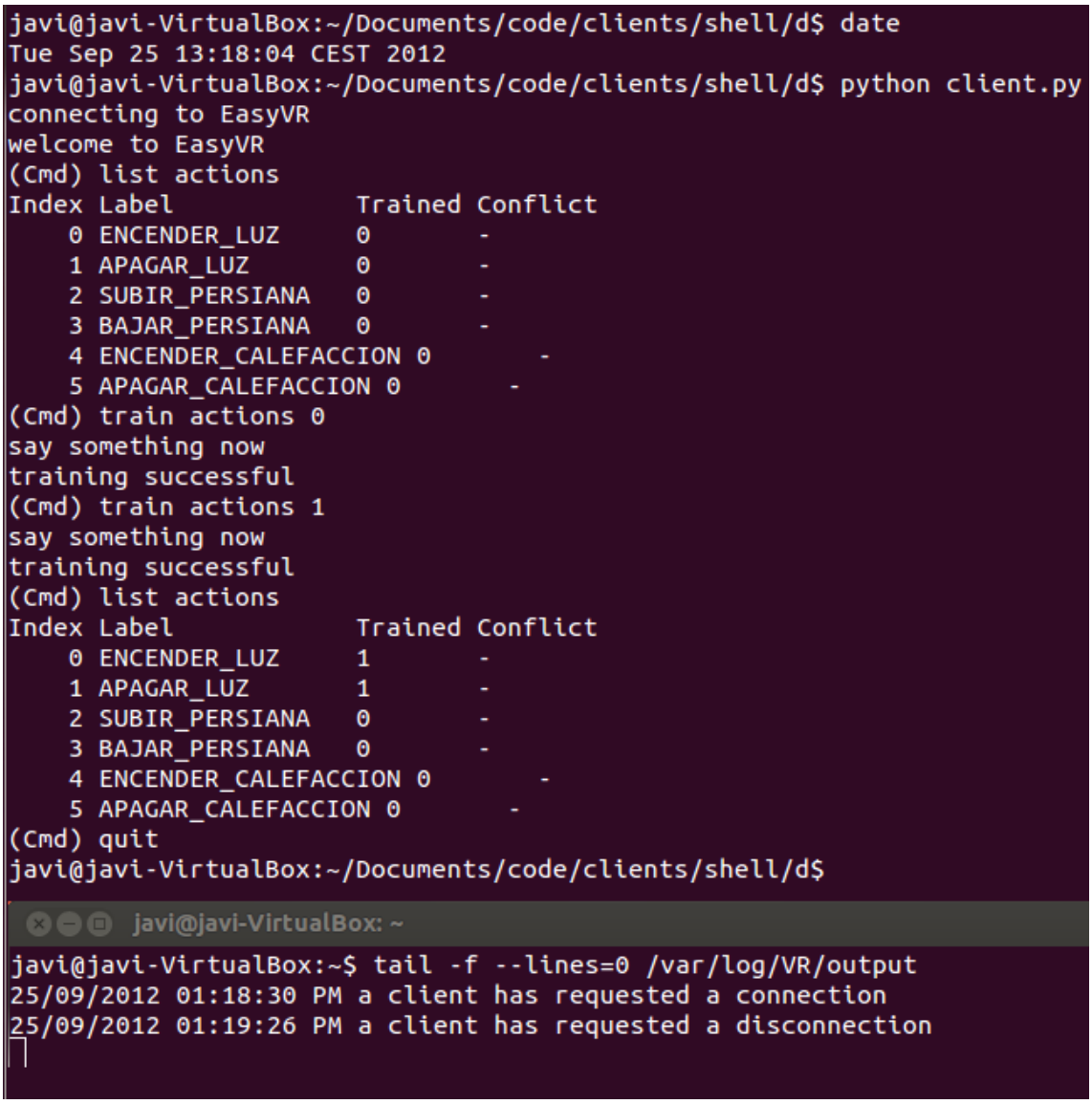

Figura 17: entrenamiento de los comandos de voz desde la línea de comandos.

6.1.3.2 Interfaz Web

Desde la interfaz web es posible entrenar un comando, borrar el entrenamiento de un comando e iniciar el reconocimiento de voz en el módulo EasyVR para probar los entrenamientos.

Primeramente hay que hacer clic en el botón "connect" para establecer la conexión directa con el módulo EasyVR y poder ver los grupos de comandos de voz disponibles.

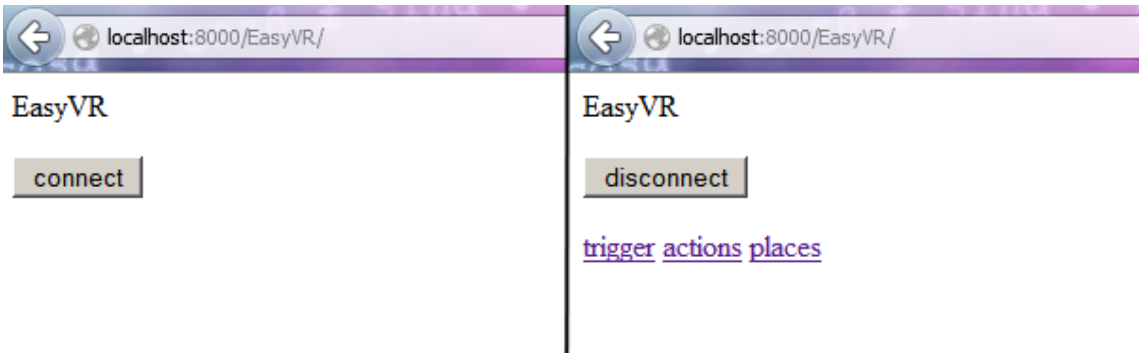

Figura 18: pantallas de inicio de la interfaz web.

Las siguientes capturas de pantalla muestran el antes y el después en el entrenamiento del comando de voz para despertar al módulo EasyVR.

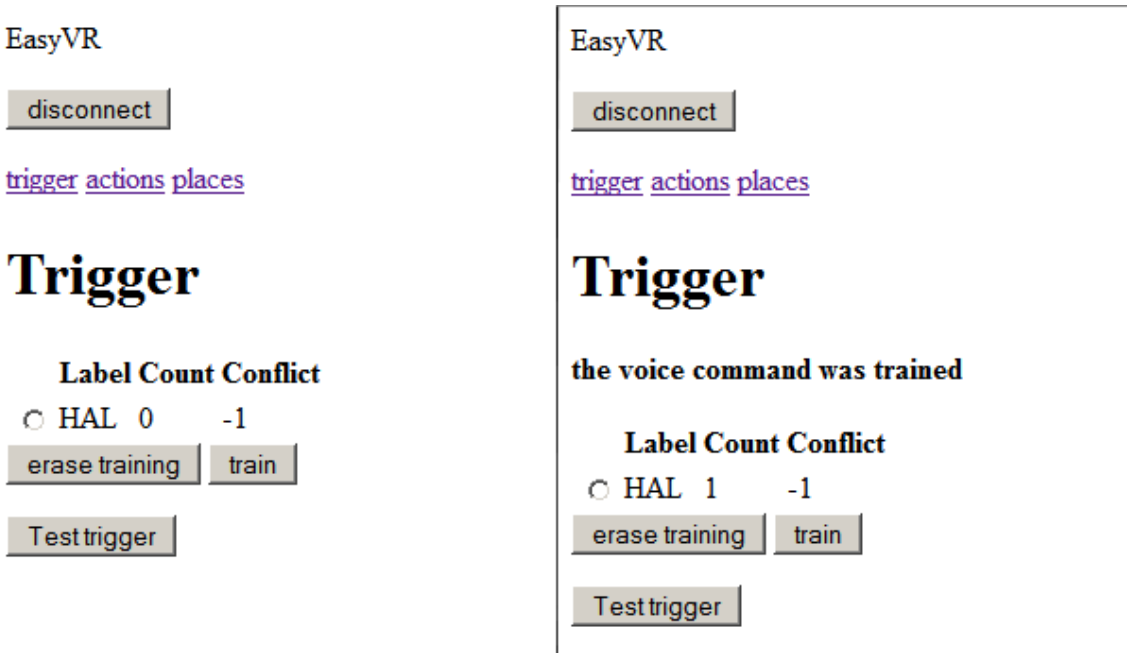

Figura 19: entrenamiento de la palabra para despertar al sistema.

La siguiente captura de pantalla muestra el resultado de decir "encender luz" tras haber iniciado el reconocimiento de voz en el grupo de las acciones.

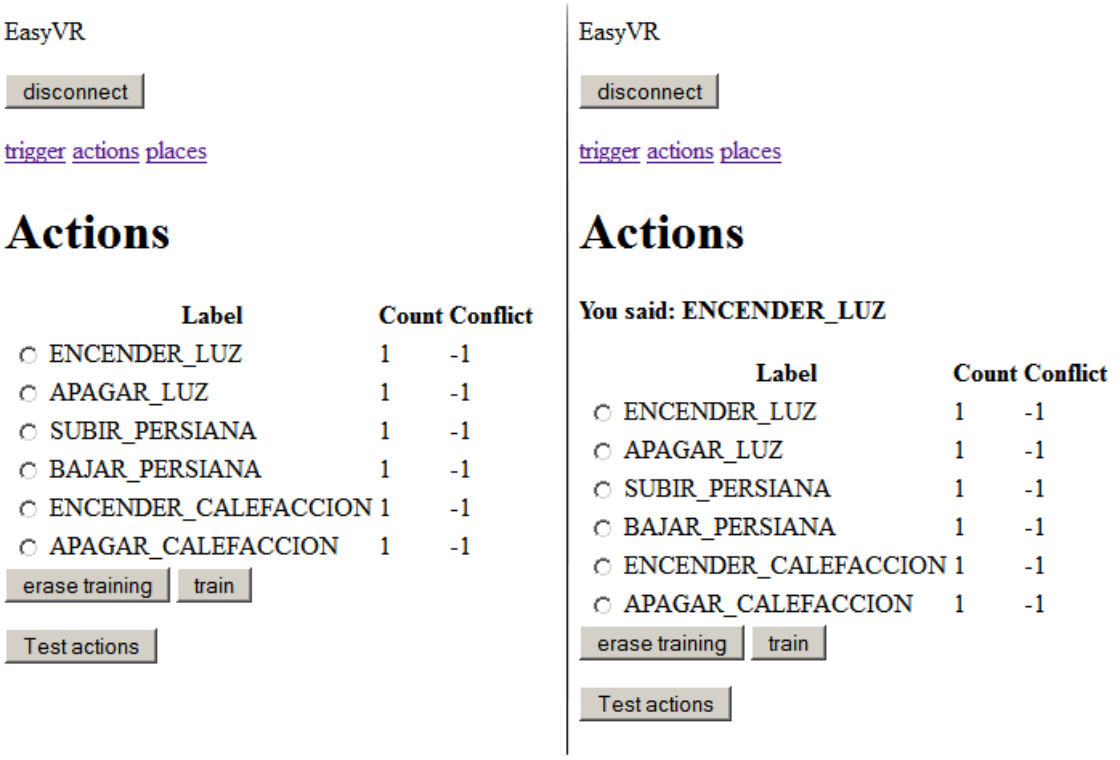

Figura 20: probando el funcionamiento del reconocimiento de voz.

La siguiente captura de pantalla muestra resultado tras haber seleccionado el comando de voz "cocina" y haber presionado el botón "erase training".

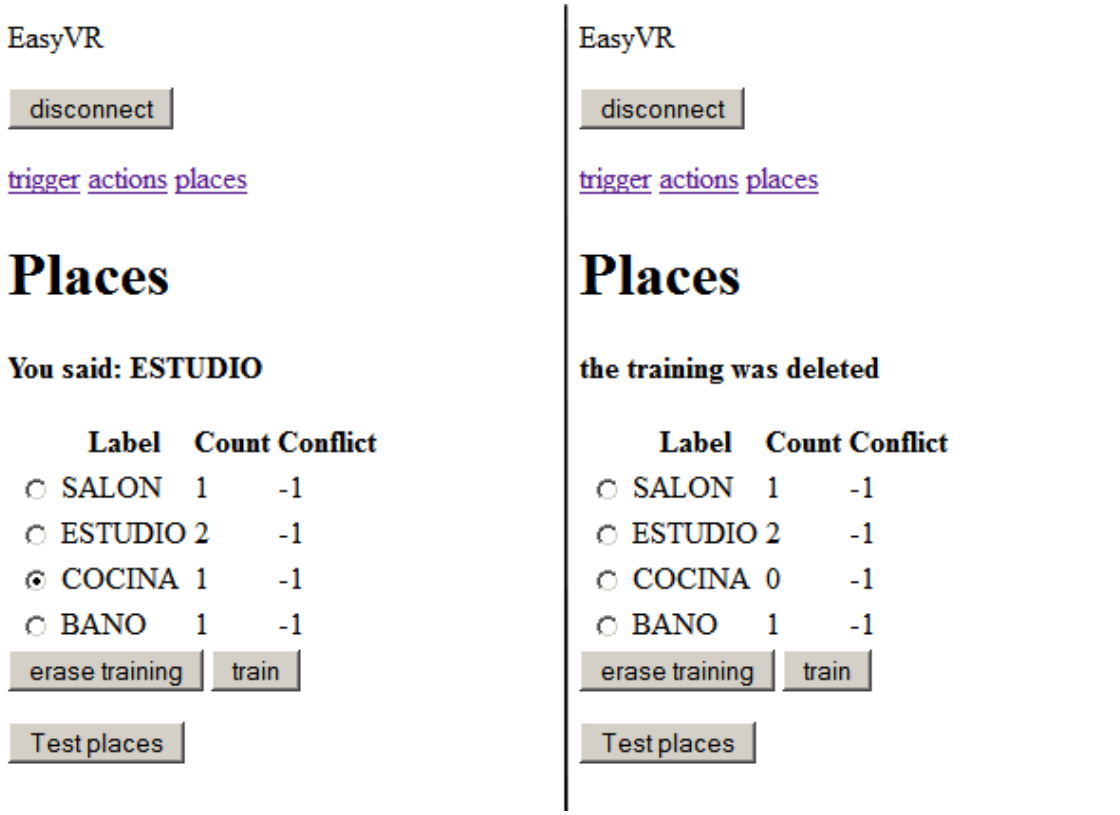

Figura 21: borrado del entrenamiento de un comando de voz.

Cuando el sistema se encuentra ocupado en el modo normal si se presione el botón "connect" el sistema vuelve a la misma página. Cuando el sistema esta pagado y se intenta acceder a la interfaz web se muestra la siguiente pantalla:

Figura 22: mensaje de error cuando el sistema estaba apagado.

6.2 Limitaciones y Trabajos futuros

6.2.1 Limitaciones

Durante el desarrollo de este proyecto se ha podido comprobar algunos fallos de funcionamiento o peculiaridades no documentadas en el módulo EasyVR. En el Anexo B se describen estos casos.

Una de las principales limitaciones del módulo EasyVR que se ha podido observar es que después de que éste reproduce una pista de audio necesita un breve instante para volver al modo de reconocimiento de voz. En las pruebas del sistema se ha comprobado que este instante es lo suficientemente grande para que un usuario empiece a hablar antes de que el módulo esté listo para reconocer. Quizá este aspecto se podría mejorar modificando la librería para la comunicación de la placa Arduino con el módulo EasyVR. Concretamente, modificando los retrasos que se incluyen antes de enviar un dato al módulo EasyVR[5].

6.2.2 Mejoras al sistema actual

Hay una serie de aspectos que permitirían mejorar el estado actual del sistema, esto son entre otros:

-auto-detección del puerto serie en el que se encuentre conectado la placa de Arduino.

-Implementar distintos niveles de verbosidad para la ejecución del sistema.

-En el modo de administración, sería conveniente que los clientes cierren automáticamente la conexión con el módulo EasyVR si se detecta que no ha habido interacción del usuario durante un tiempo determinado.

-Dar a la aplicación web una mejor apariencia (por ejemplo con hojas estilo) y un sistema de autentificación para evitar que cualquier persona con acceso a la red en el que se encuentra el sistema pueda borrar o modificar los comandos de voz.

[5] En un video de demostración del módulo SmartVR (disponible en http://www.youtube.com/watch?v=zH1KjnAwhoE) se puede observar que el comportamiento de este es mejor en este aspecto, no sólo por la rapidez con la que este módulo puede reconocer el habla después de una reproducción de audio, sino también por la mejor calidad de audio que se puede oír.

-si se dotará a la placa de Arduino de un Shield Ethernet, esto proveería al sistema de una mejor forma de comunicarse con el PC. El protocolo TCP/IP ofrece muchísimas ventajas frente a la comunicación por un canal serie. Podría implementarse el protocolo TCP/IP en el receptor y en la API, así estas dos entidades escucharían en dos puertos distintos. De esta forma, el microcontrolador a través del Shield Ethernet sólo tendría que enviar el dato al puerto correspondiente. Esto simplificaría la arquitectura actual sustituyendo la actual solución "ad-hoc" de la capa de comunicación por un protocolo de comunicación ampliamente probado.

-si se desea un mejor desempeño en el reconocimiento de voz del sistema se sugiere adaptarlo para que funcione con el módulo de reconocimiento SmartVR.

6.2.3 Otros proyectos

-Investigar soluciones para que los usuarios puedan dar órdenes de voz en una habitación sin necesidad de acercarse o saber dónde se encuentra el micrófono.

-Un sistema de reconocimiento de voz que funcione independientemente del PC, solo controlado por el microcontrolador, sería adecuado para ciertas situaciones. Por ejemplo, para asistir a personas con un alto grado de inmovilidad para poder encender sus PCs, controlar los elementos de su hogar, etc. Para este tipo de sistemas, el propio fabricante del módulo EasyVR provee un software para la creación de comandos de voz y su entrenamiento. Este software sólo funciona en sistemas Windows, habría que desarrollar otro software para sistemas Linux o adaptar la interfaz web.

7 Conclusiones

El módulo de reconocimiento EasyVR, junto con la placa de desarrollo UNO de Arduino, han resultado ser buenas opciones para el desarrollo de prototipos de aplicaciones que necesiten de reconocimiento del habla.

El módulo EasyVR es un hardware asequible y proporciona unos resultados de reconocimientos de voz aceptables. La dificultad para desarrollar aplicaciones que usen este módulo es trivial, siempre y cuando el sistema que se desarrolla este compuesto únicamente por el módulo EasyVR controlado por la placa Arduino. No era el caso del presente proyecto final de carrera, en el cual el módulo EasyVR debía enviar el resultado al PC y también debía de "dejarse" controlar a través de clientes remotos a petición del usuario. Conjugar estos dos modos de funcionamiento fue lo que planteo la complejidad real del proyecto.

Aproximadamente la distribución temporal del trabajo práctico fue la siguiente: un 25% del tiempo se dedicó al diseño, un 50% se dedicó a la implementación y el 25% restante se dedicó a las pruebas y corrección de fallos del sistema. A lo largo del proyecto la tarea que más dificultad causó tanto en el diseño, la implementación, las pruebas y corrección de errores fue el subsistema de cambio de modo de funcionamiento. Es decir, la parte que permite que existan y se conmute entre el modo de funcionamiento normal y el modo de administración. Otra tarea que tomó mucho tiempo fue la implementación de la API. El tiempo que se tomó para realizar esta tarea fue muchísimo más de lo esperado debido a los comportamientos no documentados del módulo EasyVR.

Los siguientes aspectos proporcionarían una mayor satisfacción del producto final: una mayor velocidad a la que el módulo puede cambiar de reproducir audio a iniciar el reconocimiento del habla, y una mayor distancia a la que es posible hablar para que el módulo pueda reconocer algo (actualmente la distancia máxima es de unos 30 cm).

Bibliografía

Belenguer, Javier Onielfa. 2011. *Implementación de una interfaz de usuario sin contacto.* Escuela de informática, Universidad politécnica de Valencia. 2011. Proyecto Final de Carrera.

Borrello, Charles, Rabnovich, Chris y Jabrucki, Andrew. 2011. *RIT "Tigerbot" Humanoid Platform for Future Expansion.* Kate Gleason College of Engineering, Rochester Institute of Technology. Rochester, New York : s.n., 2011. http://edge.rit.edu/edge/P12201/public/Home.

Burgess, Phil. 2011. chipKIT Uno32: first impressions and benchmarks. *Hack a Day.* [En línea] 27 de Mayo de 2011. [Citado el: 28 de Junio de 2012.] http://hackaday.com/2011/05/27/chipkit-uno32-first-impressions-and-benchmarks/.

Englund, Christine. 11th March 2004. *Speech recognition in the JAS 39 Gripen aircraft - adaptation to speech at different G-loads.* Department of Speech, Music and Hearing, Royal Institute of Technology (Suecia). 11th March 2004. Master Thesis in Speech Technology.

Jiang, Leo. 2012. *Project Proposal for Multifunction Intelligent Headphone System.* Sound Tech Inc. Burnaby : s.n., 2012.

Jurafsky, Daniel y Martin, James H. 2009. *Speech and Language Processing:An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* s.l. : Pearson Prentice Hall, 2009. 0131873210 .

Kim, Ryan. 2011. Nuance buys Vlingo. *GigaOm.* [En línea] 20 de Diciembre de 2011. [Citado el: 27 de Julio de 2012.] http://gigaom.com/2011/12/20/nuance-buys-vlingo-builds-a-voice-technology-giant/.

Poza Luján, José Luis y Posadas Yagüe, Juan Luis. 2009. *Revisión de las arquitecturas de control distribuido.* 2009.

Salvador, Marc Franco. 2011. *Navegación web usando la voz.* Escuela de Informática, Universidad politécnica de Valencia. 2011. Proyecto Final de Carrera.

Samsung unveils new Android phone with "S Voice" personal assistant. **William Meisel. 2012.** Junio de 2012, Speech Strategy News. ISSN 1932-8214.

Stan Augarten. 1998 - 2009. The Most Widely Used Computer on a Chip. *The National Museum of American History.* [En línea] Smithsonian Institution, 1998 - 2009. [Citado el: 14 de Julio de 2012.] http://smithsonianchips.si.edu/augarten/p38.htm. ISBN 0-89919-195-9.

Stevens, Richard. 2005 . *Advanced Programming in the UNIX Environment.* 2005 . ISBN 0201563177.

Toshiba. 64-Bit MIPS-Based Microcontroller With PCI Interface. *Toshiba.* [En línea] [Citado el: 14 de Julio de 2012.] http://uk.computers.toshiba-europe.com/innovation/jsp/news.do?service=UK&year=NONE&ID=00000005a4.

XMOS. Development Kits. *XMOS.* [En línea] [Citado el: 15 de Julio de 2012.] http://www.xmos.com/products/development-kits/xc-3-led-tile-kit.

Anexo A: Opciones Hardware

Microcontroladores

1. Arduino UNO: el modelo uno era el último desarrollo de placas Arduino durante la realización de este proyecto. Son un 25% más pequeñas que una tarjeta de crédito y albergan un microcontrolador ATmega328 con una velocidad de reloj de 16 MHz Dispone de 32 KB de memoria flash. Adicionalmente dispone de 1 KB de memoria no volátil (EEPROM) y 2 KB de memoria volátil (SRAM). El módulo se comunica con un PC a través del puerto USB que integra. El módulo puede alimentarse a través del puerto USB o por medio de una fuente de alimentación externa. Dispone de catorce pines digitales y seis pines analógicos. Los pines digitales pueden servir para el propósito de entrada o salida de datos, mientras que los pines analógicos sirven para la entrada de datos en el microcontrolador. Una de las características de Arduino es la disposición estándar de sus pines, lo que permite conectarlo a otros módulos que aportan nuevas características al controlador. A los módulos que integran esta disposición estándar que permite conectarlos a Arduino se les llama "Shields". El dispositivo se programa usando la IDE gratuita que provee Arduino. El precio de esta placa es de 22 €.
2. chipKIT Uno32: esta placa de la marca Digilent está basada en la placa uno de Arduino y tiene el mismo factor de forma que esta. Esto le confiere compatibilidad con muchos shields pensados para Arduino. La diferencia es que esta placa reemplaza el microcontrolador ATmega328 por un microcontrolador PIC32 de Microchip. Este último es más potente que los ATmega328. Trabajan a 32 bits y a una velocidad de reloj de 80 MHz. Tienen una memoria flash de 128 KB para programas y 16 KB de memoria SRAM para datos. Puede ser alimentada por USB o a través de una fuente de alimentación externa. Esta placa se programa usando una versión modificada del IDE de Arduino y también ofrece la posibilidad de usar el IDE de microchip MPLAB. El precio esta placa es también de 22 €.

Hardware para el Reconocimiento de Voz

Se han explorado tres opciones para el reconocimiento de voz usando microcontroladores de bajo coste:

1. SmartVR: este es un módulo de reconocimiento de la empresa Veear que usa el chip RSC-4128 de Sensory Inc. La utilización de este chip le brindan unas características avanzadas de reconocimiento. Siendo capaz de reconocer comandos de voz independientes del usuario así como comandos de voz dependientes del usuario. Es también capaz de grabar y reproducir voces y de distinguir a usuarios según su voz. Esta solución está pensada exclusivamente para llevar a cabo el procesamiento del reconocimiento de voz. No cuenta con micrófono ni altavoces.
   El precio de esta solución es de 39 €. Sin embargo para programar este dispositivo sería necesario conectarse a él a través de una placa de desarrollo que comercializa la misma empresa. Posteriormente el módulo puede ser utilizado de forma independiente. El precio combinado del módulo de reconocimiento más su placa de desarrollo es de 99 €. Dicha placa de desarrollo incluye conexión a través de USB, posibilidad de alimentación por pilas o fuente externa, micrófono y altavoces, receptor

de tarjetas para ampliar la capacidad de almacenamiento y cuatro botones y leds para prototipos rápidos. Además, junto con el paquete se incluye una IDE para programar el dispositivo y la librería FluentChip de Sensory Inc. para aprovechar al máximo las características del chip RSC-4128

2. EasyVR: este módulo de reconocimiento es el punto de entrada en este campo que ofrece la empresa VeeaR. Este módulo se presenta en dos formatos. Uno para ser conectado a través de la interfaz UART y otro empotrado en un Shield para poder ser conectado fácilmente a un microcontrolador Arduino. Entre sus características el módulo ofrece capacidad de reconocimiento dependiente del usuario, es decir reconocimiento previo entrenamiento, de hasta 32 comandos. Además, trae un pequeño conjunto de comandos que pueden ser reconocidos independientemente del usuario, es decir no hace falta entrenamiento. El módulo también tiene la habilidad de reconocimiento de contraseñas de voz. El módulo tiene capacidad de reproducción de sonidos y trae un conector para altavoces de 8 ohm y un conector tipo jack para conectar auriculares.

   El precio de este módulo es de 29 € y de 34 € si se adquiere en formato Shield. Para su utilización el módulo implementa un protocolo a través del canal serie. Este protocolo permite controlar el dispositivo desde otro elemento. Por ejemplo en el caso del módulo que se presenta en formato Shield, es trivial controlar el dispositivo desde un microcontrolador de Arduino.
3. Familia de controladores dsPIC30F de Microchip: estos controladores procesan señales digitales. Tienen una arquitectura de 16 bits y una velocidad de procesador de 30 MIPS. En cuanto a memoria disponen de entre 6 a 144 kilobytes para memoria de programa y entre 256 y 8192 bytes para datos según el modelo escogido. La capacidad de procesamiento de señales de estos controladores los hace útiles para muchos campos como: automoción, control de motores, iluminación, procesamiento de voz, diseño de fuentes de alimentación inteligentes, etc.

   El precio de estos controladores está entre los 3 y 9 $. Para poder poner en funcionamiento estos controladores necesitaremos de una placa de desarrollo. La opción más básica es adquirir un kit de inicio como el "MPLAB Starter Kit for dsPIC DSC". Este kit trae en una sola placa todo lo necesario para poder empezar a experimentar con estos procesadores en el campo de la voz y el audio (conexión con USB, micrófono, altavoz, botones y leds, IDE, código de ejemplo, etc.). El precio de este kit es de 60 $. Para poder dotar a estos controladores de capacidad de reconocimiento de voz hará falta además el uso de la librería "dsPIC30F Speech Recognition " de Microchip. El precio de esta librería en su versión de evaluación es de 5 $.

Anexo B: Comportamientos no documentados del módulo EasyVR

Operaciones cuando el módulo empieza a funcionar

-situación previa: se inicia el funcionamiento del módulo de reconocimiento (se conecta el cable USB al ordenador)

-desencadenante: se envía cualquiera de los siguientes comandos al módulo: dump, erase, insert.

-comportamiento: el módulo no responde. Si se reenvía el mismo comando el módulo empieza a trabajar normalmente.

Operaciones sobre distintos grupos

-situación previa: se envía cualquiera de los siguientes comandos al módulo para que realicen una acción sobre el grupo X.: dump, erase, insert. El módulo responde normalmente.

-desencadenante: se envía cualquiera de los siguientes comandos al módulo para que realicen una acción, esta vez sobre el grupo Y.: dump, erase, insert.

-comportamiento: el módulo no responde. Si se repite el desencadenante el módulo empieza a trabajar normalmente.

Borrado de comandos

-situación previa: existe un grupo de comandos de voz con n comandos.

-Desencadenante: usando el comando erase se borra un comando con índice i, siendo i menor que n -1 (los índices empiezan en la posición cero).

-Comportamiento: los comandos de voz con índice superior al comando borrado pero menor que n descienden a la posición inmediatamente inferior. Es decir los comandos de voz siempre se agrupan en las posiciones más bajas de sus respectivos grupos.

Añadir comandos

-situación previa: Existe un grupo de comandos de voz con n comandos (el índice que corresponde a un comando n siempre es n-1, es decir los índices empiezan en 0).

-Desencadenante: usando el comando insert se crea un comando con índice i, siendo i mayor que n.

-Comportamiento: el comando falla, el módulo EasyVR envía la respuesta "comando o argumento inválido". El comando insert sólo tiene éxito si se indica un índice menor o igual que n. Una vez más, los comandos de voz siempre se agrupan en las posiciones más bajas de sus respectivos grupos.

Anexo C: commands.xml

Fichero de Ejemplo 1

```
<system>
  <name>HAL</name>

  <object id="light">
    <label>luz</label>
    <action id="on">encender</action>
    <action id="off">apagar</action>
  </object>
  <object id="shutter">
    <label>persiana</label>
    <action id="up">subir</action>
    <action id="down">bajar</action>
  </object>

  <place id="living">
    <label>salón</label>
    <object id="light" />
    <object id="shutter" />
  </place>
  <place id="study">
    <label>estudio</label>
    <object id="light" />
    <object id="shutter" />
  </place>
  <place id="kitchen">
    <label>cocina</label>
    <object id="light" />
  </place>
  <place id="bathroom">
    <label>baño</label>
    <object id="light" />
    <object id="shutter" />
  </place>
</system>
```

Fichero de Ejemplo 2

```
<system>
  <name>HAL</name>

  <object id="light">
    <label>luz</label>
    <action id="on">encender</action>
    <action id="off">apagar</action>
  </object>
  <object id="shutter">
    <label>persiana</label>
    <action id="up">subir</action>
    <action id="down">bajar</action>
  </object>
  <object id="heater">
    <label>calefacción</label>
    <action id="on">encender</action>
    <action id="off">apagar</action>
  </object>

  <place id="living">
    <label>salón</label>
    <object id="light" />
    <object id="shutter" />
  </place>
  <place id="study">
    <label>estudio</label>
    <object id="light" />
    <object id="shutter" />
    <object id="heater" />
  </place>
  <place id="kitchen">
    <label>cocina</label>
    <object id="light" />
  </place>
  <place id="bathroom">
    <label>baño</label>
    <object id="light" />
    <object id="shutter" />
  </place>
</system>
```

Anexo D: Manual de usuario

Manual de usuario

Consideraciones antes de empezar a usar el sistema

Antes de que el usuario empiece a emitir comandos de voz se debe tener en cuenta lo siguiente:

- Habrá que situarse a no más de 30 cm de distancia del micrófono.
- Es recomendable que el micrófono esté orientado hacia la boca del usuario.
- Es necesario conectar al hardware de reconocimiento de voz unos altavoces a través de su jack de conexión ya que el sistema emitirá mensajes de audio.
- El usuario debe esperar a que el sistema termine de emitir completamente un mensaje de audio antes de poder "hablar" al sistema. Si el usuario empezase a hablar aunque sea una fracción de segundo antes de que termine el mensaje de audio, el sistema no escuchará nada.
- Cuando el usuario le indique al sistema que quiere entrenar un comando, el usuario debe esperar un segundo antes de pronunciar dicho comando. Si no se hace esto el sistema no escuchará nada.

Configuración el Sistema

El sistema usa distintos parámetros durante su ejecución. Es posible modificar estos parámetros para que se adapten a las necesidades del sistema/usuario desde un fichero de configuración. Este fichero de configuración se encuentra en la misma carpeta que el código del programa. En los sistemas Linux este fichero se llama config.py. En los sistemas Windows el fichero encargado es configW.py. Estos ficheros deben encontrarse en la misma carpeta que el código para que puedan ser leídos por el sistema.

El fichero config.py (sistemas Linux) tiene la siguiente apariencia:

```
config = {
    'log_file' : '/var/log/VR/output',
    'log_level' : 'info',
    'port' : '/dev/ttyACM0',
    'pid_file' : '/var/run/VRd.pid',
    'XML_file' : '/home/javi/Documents/code/commands.xml',
    'domotic_server' : '/usr/bin/VR/sendcmd.py'
    }
```

Cada parámetro indica lo siguiente:

- 'log_file' indica el fichero de log que usará el sistema.
- 'log_level' indica el nivel de verbosidad del sistema. Los niveles existentes de menor información a mayor son critical, error, warning, info, debug. Es decir si se indica "debug" el sistema imprimirá en el fichero de log toda la información disponible.

Además, si se indica por ejemplo " warning" también se imprimirá la información de niveles inferiores, es decir la información de "critical" y " error".

- 'port' indica el identificador del puerto serie en el que se encuentra conectado la placa de Arduino. Este identificador puede conseguirse leyendo las últimas líneas de la orden de línea de comandos "dmesg | grep tty" después de conectar la placa Arduino a algún puerto USB.
- 'pid_file' indica el fichero donde se guardara el identificador del proceso del sistema (sólo válido para Linux).
- 'XML_file' indica la ubicación del fichero XML que contiene la descripción del hogar a controlar por la interfaz de voz.
- 'domotic_server' indica la ubicación de la interfaz software hacia el servidor domótico.

Todos los nombres de los parámetros y los valores de los parámetros deben encontrarse entre comillas. Los nombres de los parámetros y sus valores deben de ir separados por el símbolo dos puntos (:). El conjunto de todos los parámetros debe encontrarse entre llaves ({}). Cualquier aspecto que se salga de estas reglas generará un error de sintaxis.

Todas las rutas a otros ficheros en el fichero de configuración de Linux deben de ser rutas absolutas. Si se usan rutas relativas el sistema no funcionará.

El fichero de configuración de Windows (configW.py) es similar salvo las siguientes excepciones:

- no es necesario especificar el parámetro 'pid_file'.
- No es necesario especificar rutas completas hacia los ficheros ya que el sistema en Windows no funciona como un demonio.

Iniciar y para del Sistema

Una vez el sistema está configurado a través del fichero de configuración correspondiente se puede pasar a su ejecución.

En Linux, el programa se inicia desde una terminal de línea de comandos. Primero, habrá que dirigirnos a la carpeta donde se encuentre el código del programa:

```
cd /ruta/del/programa/
```

Posteriormente, habrá que escribir el siguiente comando para iniciar el programa como un demonio:

```
sudo ./VRd.py start
```

Casi de forma inmediata, la terminal de línea de comandos nos volverá mostrar el cursor. El programa, al funcionar como un demonio, no imprimirá nada por la salida estándar. Si se desea ver información de funcionamiento del sistema habrá que dirigirse al fichero de lo que el sistema. Para ver los comandos domóticos que el sistema produce habrá que dirigirse al fichero '/tmp/output'.

Para detener el sistema se debe describir a través de un terminal de línea de comandos la siguiente orden:

```
sudo ./VRd.py stop
```

También es posible ejecutar el programa de forma convencional (no como un demonio). De esta forma se podrá ver la información de la ejecución del sistema en la misma consola en la que se inicie el programa. Para iniciar el programa de esta forma se deberá ejecutar el siguiente comando:

```
sudo ./VRd.py foreground
```

Adicionalmente, también puede ejecutarse el siguiente comando:

```
sudo ./main.py
```

En Windows, el programa se puede ejecutar haciendo doble clic sobre el fichero main.py o con la siguiente orden desde la terminal de línea de comandos de Windows (cmd.exe).

```
python main.py
```

Primer Uso

Si es la primera vez que se va usar el sistema, lo primero que se debe hacer es cargar los comandos de voz a partir de los datos que provee el servidor domótico y entrenarlos. Para esto habrá que iniciar el cliente de línea de comandos disponible en la carpeta code\clients\shell a través de una consola de texto:

```
./client.py
```

Una vez dentro del programa cliente debemos ejecutar la orden "reload". Dicha orden será la que lea la información que provee el servidor domótico y la que generará los comandos de voz del sistema. Pocos segundos después de ejecutar la orden anterior el cliente nos confirmará que la operación ha tenido éxito. La siguiente imagen ilustra este proceso:

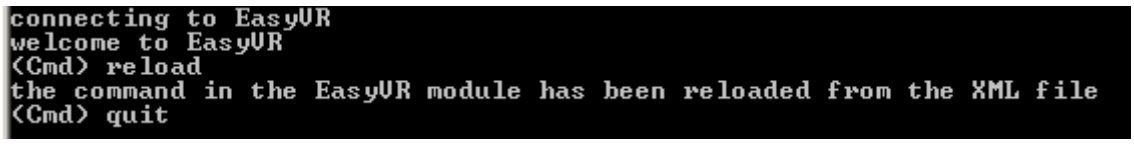

Figura 23: cargando los comandos de voz con la información que proporciona el servidor domótico.

El siguiente paso será visualizar los comandos de voz y entrenarlos. Es posible hacer esto desde el cliente de línea de comandos sin embargo la interfaz web ofrece una interfaz más amigable y más rápida a la hora de manejar. Por tanto en este manual se explicará los pasos para entrenar los comandos de voz desde la interfaz cuenta. Para cerrar el cliente de línea de comandos se deberá escribir la orden "quit" tal como se puede ver en la Figura 23. Si se desea entrenar los comandos de voz es el cliente de línea texto se puede consultar la ayuda del cliente de texto a través del comando "help”.

Para acceder a la interfaz web hemos de abrir un navegador web y dirigirnos a la dirección " http://localhost:8000/EasyVR/". En la pantalla que se nos presentan deberemos hacer clic en el botón conectar tal como se ve en la figura siguiente:

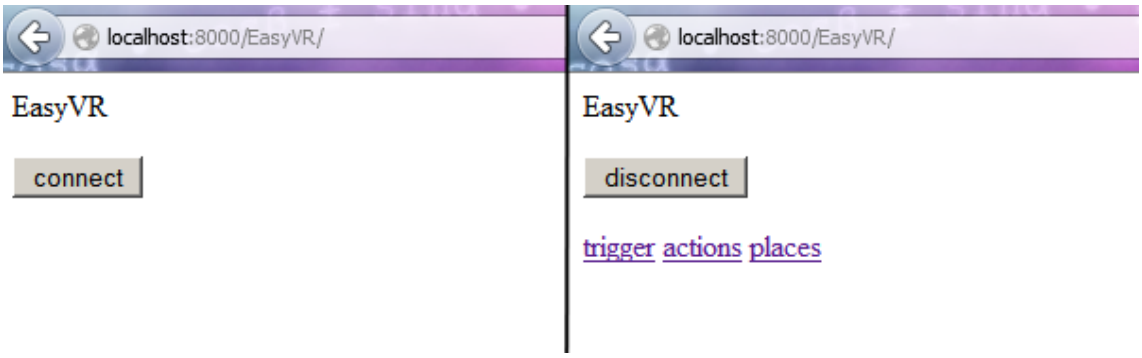

Figura 24: primeras pantallas de la interfaz web.

Los comandos de voz dentro del sistema se agrupan en grupos. Después de conectar, veremos los enlaces de los grupos de comandos de voz existentes. Existen 3 grupos: el grupo trigger, el grupo de las acciones y el grupo de los lugares. El primer grupo, el grupo trigger, contiene la palabra que despertara al sistema, por tanto es una palabra que tiene que entrenarse si se desea usar el sistema.

Deberemos ir a cada grupo y en cada grupo entrenar todos los comandos de voz que existen. Para que el módulo de reconocimiento de voz pueda reconocer de forma correcta lo que un usuario dice no se recomienda entrenar los comandos de voz más de 2 o 3 veces. La siguiente figura muestra el grupo de comandos de voz para despertar al sistema. Como se puede apreciar existe un botón para entrenar, otro para borrar el entrenamiento y otro para probar los comandos de voz. Este último sirve para iniciar el reconocimiento de voz y comprobar que el sistema reconoce correctamente lo que el usuario dice.

Figura 25: captura de pantalla de interfaz web. En la parte inferior se encuentran los botones para entrenar, borrar el entrenamiento y probar el grupo de comandos de voz.

Para entrenar o borrar el entrenamiento de un comando primero habrá que seleccionarlo tal como muestra la siguiente figura:

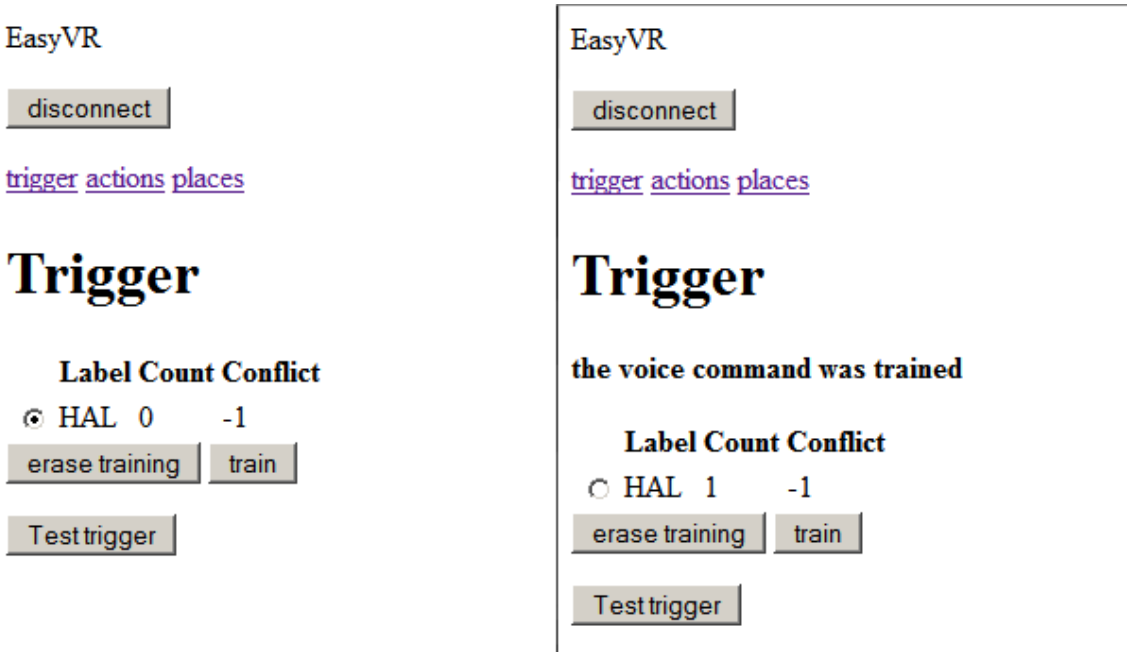

Figura 26: entrenamiento de un comando de voz.

Como se puede apreciar después de un entrenamiento se obtiene un mensaje de confirmación. Lo mismo ocurre para borrar el entrenamiento de un comando de voz tal como muestra la siente figura:

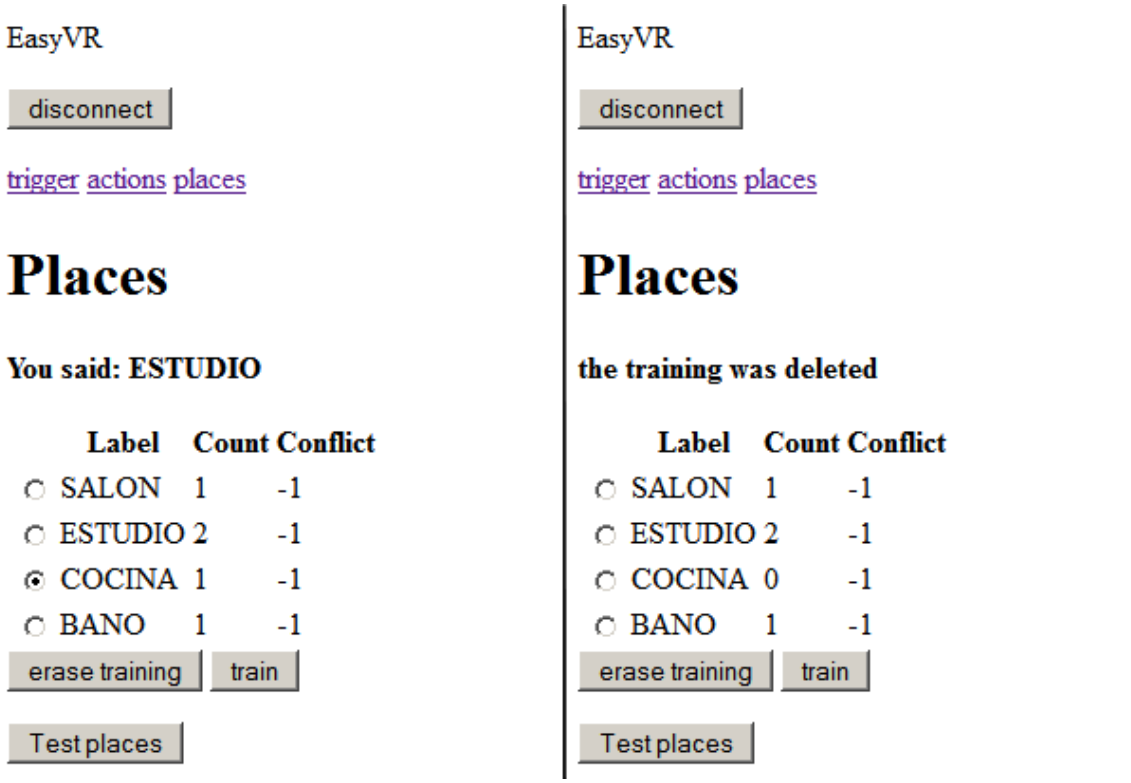

Figura 27: borrado del entrenamiento de un comando de voz.

Para probar el módulo de reconocimiento de voz reconoce correctamente los comandos que hemos entrenado no debemos seleccionar ningún comando de voz. El módulo de reconocimiento de voz reconocerá cualquier comando que el usuario diga dentro de un grupo. Por tanto si nos encontramos en el grupo de lugares e iniciamos el reconocimiento de voz, el módulo de reconocimiento de voz reconocerá cualquier lugar que el usuario diga pero sólo reconocerá lugares y nada más. La siguiente figura muestra el resultado de probar el reconocimiento de voz en el grupo de las acciones después de presionar el botón "test actions”:

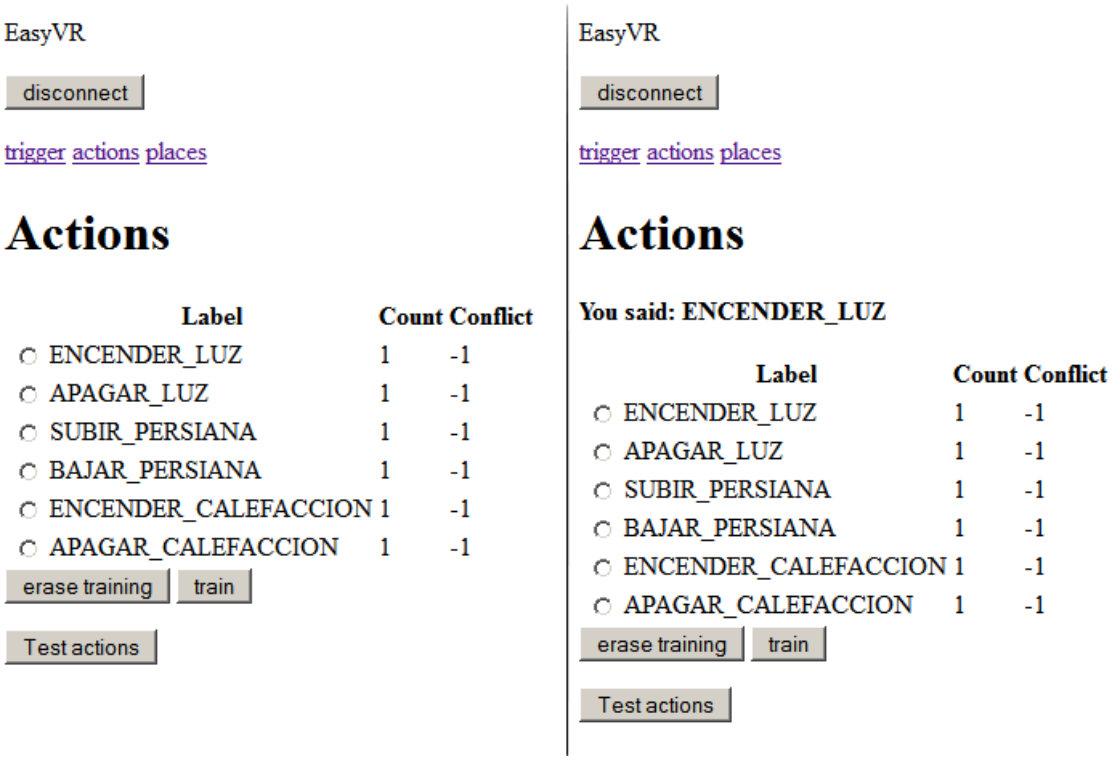

Figura 28: reconocimiento de voz del grupo de las acciones

Una vez todos los comandos de voz presenten en el sistema hayan sido entrenados y sean reconocidas correctamente se podrá empezar a usar el sistema en su modo normal de funcionamiento.

Uso del Sistema

El usuario debe mencionar la palabra que despierta el sistema. Esta palabra es la que se puede ver, cuando se realizan los entrenamientos de voz, bajo el grupo con nombre "trigger”.

Si el sistema reconoce la palabra correctamente responderá al usuario con el mensaje de audio: "a sus órdenes".

El usuario debe mencionar una frase que contenga una acción y un objeto, como por ejemplo "abrir persiana". Las frases que se pueden mencionar en esta fase son las que se pueden ver, cuando se realiza el entrenamiento, bajo el grupo con nombre "actions”.

Si el sistema reconoce la frase correctamente responderá al usuario con el mensaje de audio: "¿donde?".

El usuario debe entonces mencionar un lugar. Los lugares que se pueden mencionar en esta fase son los que se pueden ver bajo el grupo con nombre "places”.

Si el sistema reconoce el lugar correctamente responderá al usuario con el mensaje de audio:

"entendido"

Llegado a este punto los comandos domóticos generados son enviados por el sistema al servidor domótico. El sistema volverá a su estado inicial y el usuario podrá volver a decir la palabra clave para despertarlo.

Después de que el sistema reconozca la palabra que lo despierta es posible que no entienda una acción o lugar debido a una mala posición del micrófono, a la distancia con el micrófono, al ruedo ambiental, etc. en estos casos, el sistema pedirá al usuario que repita lo que ha dicho con el ciento mensaje:

"repita por favor"

Sin embargo, el sistema nunca pedirá a un usuario que repita más de 2 veces. Además, si el usuario no dice nada después de haber mencionado la palabra que despierta sistema, el sistema entenderá que el usuario ha abandonado su intención de emitir un comando domótico y volverá a su estado inicial con el siguiente mensaje:

"no he escuchado nada"