Proyecto Fin de Master (Master de Prog.LLiure)

MEMORIA:
Reconocedor de Voz para OpenDomo

Autor: Manuel Angel Loureiro Mahia

Consultor: Gregorio Robles Martínez

Tutor externo: Oriol Palenzuela

Empresa asociada: OpenDomo Services S.L

Índice de contenido

1 INTRODUCCION

Este proyecto consiste en introducir la capacidad de gestionar la plataforma OpenDomo por medio de comandos de voz. Para ello, se elige una de las soluciones que ya existen en Software Libre de Reconocimiento de Voz, adaptándola y portándola al entorno OpenDomo.

1.1 Objetivos

En concreto, según el documento de propuesta de proyecto, los objetivos son:

- Recopilar, evaluar y comparar las opciones actuales de reconocimiento de voz mediante software libre
- Elegir una de estas opciones como resultado de la comparativa
- Hacer el *porting* de esta aplicación al sistema operativo OpenDomo (Linux), integrándolo con el *toolchain* actual. Podrán usarse de referencia los *portings* de las otras herramientas, como eSpeak o Festival Lite.
- Integrar la aplicación dentro del sistema operativo OpenDomo, de modo que los comandos por voz puedan desencadenar acciones de control
- Documentar el nuevo módulo, vinculándolo con la documentación oficial de la aplicación elegida

1.2 Motivación

Este campo del reconocimiento de voz me ha interesado siempre desde que estudié en la UPM, y seleccioné como optativa la asignatura de Proceso Digital de la Señal. En este momento, se empezaba a hablar del reconocimiento de voz, y el estado del arte era tal que se consideraba como paradigma el hecho de que se pudiera reconocer la voz de un individuo concreto, pero se estaba a años luz de conseguir un sistema de reconocimiento de voz general, capaz de analizar los fonemas de cualquier individuo en un idioma.

Asociado a este campo se hablaba de términos como Lógica Borrosa e Inteligencia Artificial, que trabajaban con probabilidades para tomar decisiones.

La situación a día de hoy es que este campo ha evolucionado como se esperaba: para realizar un Reconocedor de Voz genérico:

- Es necesario tener el mayor número de muestras posibles.

- Los resultados no son 100% seguros, sino que cualquier reconocedor trabaja con datos estadísticos.

Por otro lado, el mundo de la domótica es otro campo que me apasiona y que creo, firmemente, que cuando termine esta crisis va a tener un auge importante. Ya antes de la crisis se hicieron varias aplicaciones y se presentaron (en el SIMO sobre todo), soluciones como el Hogar Inteligente en la que se controlaban tanto los electrodomésticos como las comunicaciones. Asociado a este mundillo ha nacido el concepto de "redes de sensores" en la que las soluciones se generalizan en un entorno de comunicación global facilitado por las redes Wimax y 3G/4G, generalizando el uso de sensores a servicios ofrecidos a nivel de ciudades enteras...

1.3 Estado del Arte

Completando un poco el punto anterior, voy a centrarme en comentar el estado del arte en el campo de reconocimiento de voz en domótica.

En cuanto a soluciones de domótica existen varias soluciones en el mercado que permiten:

- Gestionar energía
- Gestionar logística
- Gestionar interruptores/reguladores
- Gestionar sistemas de seguridad

Centrándonos en el contenido de este proyecto, existen soluciones de domótica que usan el reconocimiento de voz. En el mundo del software libre, cabe destacar Mister House, que es un sistema escrito en Perl. Algunas características principales:

- Hay distribuciones de Windows y de Linux. En el caso de Windows, para el reconocimiento usa una versión libre de Microsoft VR. En el caso de Linux usa IBM Via Voice.
- Manejo de todo tipo de interruptores (luz, clima, etc) basados en los controladores HW standard: Arduino, X10,...
- Gestión de centrales de seguridad y de cámaras de videovigilancia
- Automatizar la gestión de sistemas de sonido y TV
- Sistema automático de respuestas que realiza acciones concretas al recibir llamadas telefónicas
- Estación meteorológica

Es un proyecto vivo que está disponible también en Github. Las ultimas actualizaciones son de marzo de 2014.

1.4 Estructura de la Memoria

Para mayor claridad se ha ido desarrollando la memoria conforme a los diferentes pasos que se han dado hasta llegar a la solución final. Por ello, se introduce a continuación el contenido de cada capítulo.

Los primeros dos capítulos son la introducción y la licencia.

El siguiente capítulo (capítulo 3: Método de Trabajo) describe cómo se ha organizado el trabajo en cada fase del proyecto, teniendo en cuenta que el proyecto consta de dos fases bien diferenciadas:

- Prospección de Soluciones de software Libre para Reconocimiento de Voz
- Desarrollo /Adaptación de la solución elegida al entorno de OpenDomo

El capítulo 4 (Entorno tecnológico del proyecto) describe el entorno que se ha usado para cada fase del proyecto: máquinas, distribuciones software, herramientas de desarrollo, ...

El capítulo 5 (Seguimiento) describe, dado que es un proyecto realizado con una empresa asociada, la definición de los métodos de comunicación que se usan para cada evento del proyecto: gestión y planificación, definición y aceptación de requisitos, control de errores...

El capítulo 6 entra ya en la solución y describe la solución que se va a usar como referencia (Palaver-API de Reconocimiento de Voz de Google).

El capítulo 7 es la formalización en requisitos que debe de cubrir el sistema: tanto funcionales como en términos de capacidad y funcionamiento. Estos requisitos obviamente han sido acordados con los responsables de OpenDomo.

El capítulo 8, describe el desarrollo realizado, en la que primero se describe las especificidades para cubrir los requisitos, y luego se pasa a describir el software implementado, tanto el software en

tiempo real, como las herramientas usadas para configurar los comandos de voz (VoiceCommands).

El capítulo 9 se centra en los problemas y las adaptaciones que hubo que hacer a raíz de los problemas encontrados en las pruebas.

Los capítulos 10 y 11 hacer referencia a la documentación entregada y el soporte on line que se está dando.

El capítulo 13 incluye las Lesson Learnt, con las cosas que se harían de manera distinta si se empezase de nuevo el proyecto.

El capítulo 14 hace referencia a los planes de evolución, con la información contrastada con los responsables de OpenDomo.

El capítulo 15 detalla las Referencias externas.

Y por último hay un anexo con un snapshot de la documentación y la wiki de soporte.

2 LICENCIA

3 MÉTODO DE TRABAJO

Este proyecto contiene 2 fases claramente diferenciadas:

- Fase 1 (Prospección): Consiste en elegir soluciones de reconocimiento de voz que puedan ser portadas a OpenDomo, realizando pruebas en un entorno genérico.

- Fase 2 (Porting a OpenDomo): Una vez elegida una de las alternativas, consensuadas con OpenDomo, hay que realizar el porting, definiendo los escenarios a reconocer y adaptando la solución al entorno tecnológico de OpenDomo.

3.1 Método de Trabajo en la fase 1 (Prospección)

En esta fase se analizarán las diferentes soluciones existentes de reconocimiento de voz, en un entorno Linux, y que cumplan las premisas siguientes:

- Deben ser licencias de software Libre, preferiblemente GPL
- Deben ser soluciones genéricas que permitan gestionar por comandos de voz los sistemas, independientemente del locutor: no debe ser necesaria ninguna grabación para que el sistema reconozca el comando de voz.
- Debe ser una solución que, además del inglés, permita tener un diccionario en castellano (sería deseable que admitiera más lenguas vernáculas: gallego, catalán, euskera...), pero, dado el estado del arte de esta tecnología, no va a ser posible en el entorno de este proyecto.

Para ello, se utilizarán distribuciones genéricas de Linux (Debian, Ubuntu, Suse, Fedora), donde se probarán dichas alternativas, analizando los pros y contras de cada una de ellas, con el objetivo de seleccionar una de las alternativas, que al final, será la base para el desarrollo del reconocedor de voz de OpenDomo, que se realizará en la siguiente fase.

3.2 Método de Trabajo en la fase 2 (Porting)

Una vez elegida y consensuada con OpenDomo la alternativa más apropiada, en esta fase se hará el porting y adaptación. Para ello, hay que tener en cuenta que OpenDomo es una distribución bastante reducida, que no posee interfaz gráfico (sí acceso web) y cuya toolchain está basada en Busybox, siendo una customización muy específica, no poseyendo herramientas potentes que chequeen las dependencias, como aptitude o yum. Por ello, hay que preparar un paquete completo, empaquetándolo en un .tar.gz.

Una parte muy importante consistirá en definir y customizar las órdenes que se vayan a usar en OpenDomo. Teniendo en cuenta que el acceso de usuario es via web, con lo que los tipos de ordenes consistirán en moverse en los distintos menús y submenús que existen en el interfaz web.

4 ENTORNO TECNOLOGICO DEL PROYECTO

Teniendo en cuenta las dos fases del proyecto, va a haber dos escenarios de tecnológicos en el proyecto:

- *Fase 1 (Prospección):*

En esta fase va a ser necesario el siguiente entorno:

- Distribución Linux para probar las soluciones
- Entorno de Desarrollo para alguna solución. En concreto, hay soluciones basadas en Java, con lo que es necesario un entorno de desarrollo Java (Eclipse).

- *Fase 2 (Porting):*

En esta fase necesitamos:

- Una máquina con OpenDomo
- Una máquina con una distribución Linux genérica para configurar los paquetes y usarla como referencia
- Un entorno de Desarrollo (Eclipse) para las posibles adaptaciones necesarias
- Un repositorio local con un sistema de control de versiones
- Un repositorio web donde ir dejando tanto el software como la documentación del proyecto.

4.1 Entorno tecnológico de la Fase 1 (Prospección)

4.1.1 Distribución Linux para probar las soluciones

Dado que hay soluciones de reconocimiento de voz específicas para una distribución (Simon: Ubuntu, Gnome_Voice_Control: en distribuciones con escritorio Gnome, en concreto, Debian), es necesario tener más de una distribución de Linux.

En mi caso tengo varias distribuciones que he usado para la prueba de las soluciones:
- Ubuntu 13.10 instalada en una partición de 20 Gb
- Debian 7, instalada en una tarjeta SD de 16 Gb (Clase 10, para que no merme la velocidad respecto a disco duro)
- CentOs 6.4 instalada en otra tarjeta SD de 16Gb

4.1.2 Entorno de Desarrollo

Para este apartado, he instalado Eclipse Juno EE. Esto ha sido necesario porque alguna de las soluciones se basan en un applet de Java (Sphinx4).

4.2 Entorno tecnológico de la fase 2 (Porting)

4.2.1 Máquina Virtual con OpenDomo

Estoy usando VirtualBox para cargar la máquina OpenDomo en la versión 1.0.0. Esta máquina correrá sobre alguna de las distribuciones Debian/Ubuntu instaladas en mi portátil.
Aunque, en principio, esta máquina virtual parece ser suficiente para la prueba de la solución de reconocimiento de voz portada, seguramente, para la solución definitiva, una vez probada funcionalmente la solución completa, instalaré una distribución Opendomo en una tarjeta SD, sin virtualizar.

4.2.2 Maquina para configurar el porting

Como se ha comentado, esta máquina es la que se usará para preparar y actualizar la toolchain a usar en OpenDomo.
Para esta máquina voy a usar una distribución Debian, por la facilidad tanto a la hora de probar las soluciones como las facilidades que tiene para probar los scripts que se usarán en la adaptación de la toolchain.

4.2.3 Entorno de Desarrollo

Para este apartado, usaré Eclipse Juno EE. En principio, dada la naturaleza del proyecto, el número de adaptaciones de software debe ser mínima: si el entorno de OpenDomo es equivalente al de la distribución Linux donde se prueba la solución de reconocimiento de voz, no haría falta ningun cambio de código.

4.2.4 Repositorio local: Subversion

Varias son las razones de usar subversion:

- Dada la naturaleza del proyecto, dentro de las diferencias que existen entre CVS y SVN, las siguientes razones fueron contundentes para elegir SVN respecto a CVS:

    1. CVS esta pensado para manejar sobre todo ficheros de texto,mientras SVN maneja todo tipo de ficheros. En el caso de este proyecto, hay una gran variedad de tipos de ficheros: diccionarios, tramas , fonemas, etc... que se soportan mucho mejor con SVN. De hecho, la prueba más evidente es que todos los proyectos que he analizado usan subversion.
    2. SVN es más rápido que CVS cuando hay que mover gran cantidad de información de un repositorio a otro. Los repositorios de los reconocedores de voz son muy grandes, dada la gran cantidad de información vocal que se usa en los reconocedores. En este proyecto es necesario hacer copias locales de los repositorios globales que tienen las diferentes soluciones.
- Otro de los motivos es que SVN se integra fácilmente con Eclipse, lo cual es una ventaja a la hora de hacer las adaptaciones, pudiendo manejar las distintas ramas desde Eclipse directamente, sin ser necesario el uso de los comandos directos desde Linux. De esta forma, basta con configurar Eclipse con Subversion (Subversive) y tendremos un control de versiones subversion, independiente de la plataforma. De hecho, esta solución sería compatible con cualquier distribución que tenga instalado Eclipse con Subversive, incluso en Windows.
- El ultimo motivo es que para albergar en red el proyecto, se ha decidido usar Github, entre

otras cosas, por compatibilidad con el resto de soluciones de OpenDomo.

- Github usa SVN, con lo que usar otro control de versiones no haría más que complicar innecesariamente la gestión del proyecto.

4.2.5 Repositorio en Red: Github

Aunque, como se ha comentado anteriormente, la razón de elegir Github viene forzada por coherencia con el resto de proyectos sobre OpenDomo, ha habido varias razones para decidirnos a usar OpenDomo:

- Casi todos los proyectos que se han analizado están en Github.
- Github permite crear un entrono completo:
  - Ramas de versiones de software (incluyendo código, configuración, scripts, etc...
  - Documentación
  - Control básico de errores
  - wiki, blogs....
- Github es también compatible con Eclipse y SVN.

5 SEGUIMIENTO DEL PROYECTO

En este apartado voy a indicar cómo se va a hacer el seguimiento del proyecto, teniendo en cuenta que, por un lado, es un desarrollo ad-hoc y debe ser validado por los responsables de OpenDomo, y por otro, es un proyecto de software Libre , que va a ser público y, por ello, hay que tener una estrategia clara sobre cómo manejar las contribuciones externas que pueda haber en el proyecto.

5.1 Comunicación

En una primera fase la comunicación se está realizando, casi exclusivamente con personal de OpenDomo, usando el mail privado como herramienta de comunicación. En este proyecto, en el que hay que hacer una customización bastante específica no tiene sentido crear listas de correos específicas, ya que las contribuciones vendrán por el equipo de gestión de OpenDomo y posteriormente, de las contribuciones en Github que pueda haber de la comunidad OpenDomo.

En las ultimas conversaciones con el personal de OpenDomo hemos acordado abrir ya el proyecto en Github, con lo que se creará una wiki en el proyecto para recibir los comentarios/contribuciones externos de la comunidad de desarrolladores. Probablemente, al ser un proyecto muy específico no es previsible que haya muchas contribuciones, salvo las propias de la comunidad OpenDomo. Por ello, se creará un link desde la página OpenDomo hacia el proyecto (como existe en otros subproyectos OpenDomo) de forma que podamos recibir las contribuciones de esta comunidad.

5.2 Control de Errores

Al usar Github, usaremos los mecanismos de gestión de issues de Github. No se considera necesario en este proyecto, que es básicamente una customización, tener un control de errores tipo Bugzilla, dado que el número de errores se espera que sea bajo y muy específico de las adaptaciones que se hagan.

5.3 Documentación

Toda la documentación del proyecto estará incluida tanto en las páginas de OpenDomo como en Github.
La documentación se adaptará a las plantillas/metodología usada en OpenDomo para mantener coherencia con el resto del proyecto.

5.4 Gestión del proyecto

Dado que la solución que se decida portar ha de ser, tal y como se ha descrito anteriormente, una licencia de software Libre (probablemente una variante de GPL), el proyecto tendrá la misma licencia.

Como se ha comentado anteriormente, el proyecto estará en Github, y desde esta página se tendrá acceso tanto a las distintas ramas de software para descargas, como a documentación, control de errores, noticias, wiki, etc. Por otro lado, habrá un link desde la página de OpenDomo hacia el sitio del proyecto, de forma que desde OpenDomo se pueda acceder al proyecto.

La estructura de la página OpenDomo es:

Figura 13-1: Página principal de Opendomo

Como se puede ver hay un link en el lado izquierdo con los proyectos activos en cada momento. Se añadirá el link del proyecto en este apartado, enlazándolo con Github.

En cuanto al empaquetado para la instalación, como se ha comentado anteriormente, se usará la toolchain de OpenDomo, que consiste básicamente en empaquetar el software en formato .tar.gz, usando unos scripts para el control de los paquetes a instalar. Estos paquetes están en un repositorio común de OpenDomo, que se gestionan con unos scripts cada vez que se quiere instalar un nuevo paquete.

6 SELECCIÓN DE SOLUCIÓN DE RECONOCIMIENTO DE VOZ

Siguiendo el esquema de la asignatura “Implantación de sistemas de software Libre”, este capítulo corresponde al apartado de “Análisis de Soluciones en software Libre”.

A continuación se muestra una tabla con las diferentes soluciones analizadas y sus características:

	LICENCIA	MADUREZ	FACIL. USO	IDIOMA	PORTING	COMENTARIOS
SPHINX4	GPLV3	ALTA	Aplicación Java	Ingles OK (**1) Castellano: (**2) Diccionario pobre (mexicano)	Adaptación en entorno Java	Buena solución, pero baja calidad en castellano
SIMON	GPL	ALTA	Pensado para Ubuntu Basado en Sphinx	Ingles OK (**1) Castellano: (**2) Diccionario pobre (mexicano)	Mucha adaptación: muy especifico para Ubuntu KDE	Buena solución en KDE/Ubuntu, pero complejidad de porting y baja calidad en castellano
POCKETSPHINX	GPLV3	ALTA		Ingles OK (**1) Castellano: (**2) Diccionario pobre (mexicano)	Paquete cerrado+ dependencias	Buena solución, pero baja calidad en castellano
GNOME_VOICE CONTROL		MEDIA	Pensado para el desktop GNOME	Ingles OK (**1) Castellano: (**2) Diccionario pobre (mexicano)	Muy específico escritorio Gnome	Buena solución, pero baja calidad en castellano
PERLVOX	GPL	BAJA	Específico Ubuntu	Inglés Solo	Muy específico Ubuntu	Solución pobre
MISTER_HOUSE	GPL	BAJA	Muy específico, asociado a solución Sw	Inglés solo	Solución muy particular de domótica, pero pensado solo para un entorno muy específico	Solución pobre
VEDICS	GPL	BAJA	No funciona bien	Inglés	Calidad muy pobre	Solución pobre
SPEECHLION	GPL	MEDIA	Basado en Sphinx	Ingles OK (**1) Castellano: (**2) Diccionario pobre (mexicano)	Solución similar a Simon pero no tan depurada	Buena solución, pero baja calidad en castellano
PALAVER	GPLV3	MUY ALTA (Google API)	Muy fácil de adaptar, creando Plugins	Inglés OK Castellano OK Hay opciones de catalán /gallego...	Porting muy sencillo: paquete base + creación de plugins específicos	SOLUCION OPTIMA

(**1): Basado en los modelos de voz de VoxForge. Es un proyecto bastante consolidado en inglés, pero en cambio en castellano ha tenido hasta ahora muy pocas contribuciones, las más importantes vienen de la universidad de México, y su calidad es muy pobre.
(**2): Para una aplicación de reconocimiento de comandos por voz, todas estas distribuciones pueden usar las herramientas de sphinxtrain, que permiten, para un interlocutor concreto, reconocer

una serie de comandos específicos con una fiabilidad muy alta. Esto no es una solución para un producto destinado a distintos usuarios, porque obligaría a customizar cada instalación

Después del análisis parece claro que la solución óptima es PALAVER. Funciona correctamente en inglés y castellano, además permite crear plugins específicos para los comandos que se desee reconocer. Hay que tener en cuenta que en OpenDomo los comandos a reconocer son muy específicos de la aplicación: encender luces, manejar sonido, activar cámaras.... por ello, es necesario un sistema que permita flexibilidad en los comandos .

Por ultimo, PALAVER, al usar la tecnología de reconocimiento de Google es la unica solución válida en castellano, dado que Google sí que ha tenido contribuciones suficientes como para tener una base de datos lo suficientemente grande como para distinguir palabras de castellano, independientemente del interlocutor (masculino/femenino, edad, acento...).

6.1 DESCRIPCION DE PALAVER (software de referencia)

Como se ha comentado, Palaver es un software de reconocimiento de voz con Licencia GPLv3.

Está disponible en Github:

https://github.com/JamezQ/Palaver/

Palaver es un sw que se basa en conectarse al API de Google para reconocimiento de voz. Para ello, Palaver está estructurado en los siguientes componentes básicos:

- Diccionario
- Plugins (voice Commands)
- Llamada al API de Google

Para el reconocimiento, y para indicar que cuando comienza una frase a reconocer y cuando acaba, se basa en definir una “hotkey”, un atajo de teclado (por ejemplo, Ctrl l) que marca el comienzo del muestreo de la voz a enviar a Google, terminando cuando se vuelve a teclear la hotkey. Este método no es operativo, y desde luego, no podría usarse en un sistema como OpenDomo. Como se verá en los siguientes apartados, este funcionamiento se ha cambiado totalmente en el paquete realizado para OpenDomo.

6.1.1 Diccionario

El diccionario define la asociación comando de voz – comando a ejecutar.

Su formato es el siguiente:

Frase que dice la persona, pudiendo indicar varias opciones incluyéndolas entre <opt1,opt2>

Comando a realizar

Un ejemplo es:

open <g,G>oogle <c,C>hrome

open 'google-chrome'

En este ejemplo lo que puede decir la persona es cualquiera de las siguientes frases:

open google chrome

open Google Chrome

open Google chrome

open google Chrome

Y el comando que se ejecuta es ==> open 'google-chrome'

Este comando puede ser un comando directo linux, o un script definido por el usuario.

6.1.2 Plugins o voiceCommands

Una de las razones por las cuales se eligió Palaver como software de referencia es su facilidad para implementar nuevas órdenes.

Para ello Palaver implementa lo que se llaman Plugins, que en OpenDomo VR hemos pasado a llamar VoiceCommands ya que el término Plugin ya se utilizaba en OpenDomo.

Los plugins tienen como estructura un directorio en el que por un lado se implementa

- el diccionario de dicho comando: relación comando de voz- script /comando a realizar
- la definición del script que puede ser implementado en Shell, Python, Perl, etc.

Una vez compilado el plugin:

- Se añade el diccionario al diccionario general de la aplicación (main.dic)
- Se añade el comando a ejecutar aun directorio que contiene todos los scripts/comandos a ejecutar.

Todo este sistema se ha adaptado para el reconocedor de voz (OpenDomo VR), dándole mayor flexibilidad (opción multilenguaje).

6.1.3 Llamada al API de Google

Realmente éste es el “core” de la aplicación. Implementa la llamada al API de Google, en la que se pasa un fichero de sonido con la voz, el idioma a reconocer y Google nos devuelve el texto que ha reconocido, así como la fiabilidad del reconocimiento:

Esta llamada está implementada en un pequeño script (snd_speech) :

```
#!/bin/bash


# This should change based on language.
# lang=es or something.
```

```
# Multiple results
#URL="http://www.google.com/speech-api/v1/recognize?
lang=en&client=chromium&maxresults=6"
URL="https://www.google.com/speech-api/v1/recognize?
lang=$LANG&client=chromium"

if [ -z "$1" ];then
    echo ""
    exit 1
fi

wget -qO- --post-file "$1" --header 'Content-type: audio/x-flac;
rate=16000' "$URL" > result.json

RESULT="$(cat result.json | sed 's/^[^[]*\[{\"utterance\":\"\
([^\"]*\)\".*/\1/')"

echo "$RESULT"
```

Como se puede ver la idea es muy sencilla, y nos permite definir el tipo de fichero de voz a enviar, la frecuencia de muestreoy el idioma.
La respuesta nos la devuelve Google en formato Java Script (result.json).

7 FORMALIZACIÓN DE LA DEFINICIÓN DEL SISTEMA (OpenDomo VR)

Siguiendo como referencia la documentación de la asignatura de Implantación de Sistemas basados en software Libre, la siguiente fase, una vez elegida una solución de referencia, es la formalización de la solución a implementar para nuestro caso particular. En este caso, lo que se va a definir en este apartado son las modificaciones a realizar en el software de base (Palaver) para la implementación del reconocedor de voz en OpenDomo, teniendo en cuenta, que dicho sistema es una distribución Linux reducida, que no posee interfaz gráfico (solo interfaz web) y acceso a consola.

7.1 Resumen de requisitos

A continuación se detallan, de manera esquemática, los requisitos que debe satisfacer el sistema:

7.1.1 Requisitos no funcionales

1. Debe permitir entorno CGI para su gestión por interfaz web.
2. No debe de contener ningun elemento gráfico (Palaver tiene ventanas de notificación)
3. Debe de minimizarse el tamaño del software de base, así como de las dependencias, dado que OpenDomo es una distribución mínima
4. El sistema no debe poseer software a compilar o a interpretar, dado el tamaño reducido de la distribución.
5. El reconocedor de voz generará los datos de reconocimiento cuando se arranque la aplicación, de forma que se minimice el consumo de recursos en fase de reconocimiento. De esta forma, si se realiza algun cambio al sistema OpenDomo, debe reiniciarse automáticamente la aplicación para que contemple los nuevos datos.
6. El reconocedor de voz debe ser autoinstalable, de acuerdo con la metodología de instalación de proyectos dentro de OpenDomo. De acuerdo con esta metodología, el proyecto debe empaquetarse en un fichero .tar,gz, que se desempaquetará en cada instalación de cliente.
7. La configuración del reconocedor de voz debe ser bastante sencilla, pensada para usuarios sin conocimientos informáticos.
    - Debe maximizarse el uso de los procesos de autoconfiguración que permitan que el propio reconocedor de voz se configure basándose en información que extraiga el propio sistema OpenDomo
    - Dado que la configuración se hará desde el interfaz web, el procedimiento no puede ser interactivo, sino que debe de poder introducirse los parámetros de antemano, siguiendo una estructura concreta definida en OpenDomo.

7.1.2 Requisitos funcionales

1. Debe de tener opciones multilenguaje (castellano, inglés)
2. El reconocedor debe estar corriendo siempre, y debe autoarrancarse cuando se arranque el sistema.
3. Debe de contener al menos dos estados:

- Identificación: Es es estado inicial. Solo se reconoce una frase llave, por ejemplo: “Hola OpenDomo”. Mientras esta frase no sea reconocida, el reconocedor permanecerá en este estado. En cuanto se reconozca la frase llave pasará al estado “Normal”.
- Normal: Este es el estado de reconocimiento. En él se reconocerán todos los comandos. Para salir de este estado, se usará una frase llave de despedida (por ejemplo, “Adios OpenDomo”) que llevará al sistema al estado de Identificación.

4. Para reconocer una frase, no debe usarse ningún tipo de atajo de teclado para indicar el comienzo y el final de la frase, sino que debe ser capaz el sistema de detectar el silencio, de forma que solo se envié a reconocer las tramas de sonido entre dos intervalos de silencio.
5. Dado que el tipo de comandos a reconocer actuará sobre los diferentes “objetos” del sistema, el reconocedor de voz debe ser capaz de detectar los “objetos del sistema”, no permitiendo actuar sobre objetos no definidos. Por ejemplo, si un comando es “encender luces de salita”, el sistema buscará si existe la habitación denominada “salita” y, solo en este caso, reconocerá y actuará sobre el interruptor de luz de la salita.
6. El reconocedor de voz actuará sobre los elementos del sistema, de forma que modifique el estado de acuerdo a la orden indicada por el usuario. Por ejemplo, en el caso anterior, actuará sobre el puerto del sistema que enciende la luz de la salita.
7. Para comunicarse con el cliente, el reconocedor de voz empleará voz sintética generada con herramientas estandard de síntesis de voz (espeak). Requisitos adicionales:
   - El cliente podrá elegir distintos flavors de voz, por ejemplo: voz masculina o femenina.
   - Los textos a sintetizar deben poder ser modificados de acuerdo con las preferencias del usuario, por ejemplo: confirmaciones formales o informales.

7.2 Definición de Comandos de Voz

A continuación se detallan los comandos de voz que se han acordado con los gestores de OpenDomo. Esta definición de comandos corresponden a la primera fase del proyecto.

```
#PLUGIN: VarClimate
<ajustar,cambiar> termostato (LINE Objeto)
  VarClimate "$Objeto$"
termostato (LINE Objeto)
  VarClimate "$Objeto$"

#END
#PLUGIN: VarLight
<ajustar,cambiar> luz (LINE Objeto)
  VarLight "$Objeto$"
luz (LINE Objeto)
  VarLight "$Objeto$"

#END
#PLUGIN: MusicOFF
desactivar música (LINE Objeto)
  MusicOFF "$Objeto$"

#END
#PLUGIN: MusicON
activar música (LINE Objeto)
  MusicON "$Objeto$"
```

```
#END
#PLUGIN: VideoOFF
desactivar video (LINE Objeto)
  VideoOFF "$Objeto$"

#END
#PLUGIN: VideoON
activar video (LINE Objeto)
  VideoON "$Objeto$"

#END
#PLUGIN: Sensors
informar de (LINE Objeto)
  Sensors "$Objeto$"
#END
#PLUGIN: ClimaOFF
apagar clima (LINE Objeto)
  ClimaOFF "$Objeto$"
apagar clima (LINE Objeto)
  ClimaOFF "$Objeto$"
#END
#PLUGIN: ClimaON
encender clima (LINE Objeto)
  ClimaON "$Objeto$"
encender clima (LINE Objeto)
  ClimaON "$Objeto$"
#END
#PLUGIN: LucesOFF
apagar luz (LINE Objeto)
  LucesOFF "$Objeto$"
apagar luces (LINE Objeto)
  LucesOFF "$Objeto$"

#END
#PLUGIN: LucesON
encender luz (LINE Objeto)
  LucesON "$Objeto$"
encender luces (LINE Objeto)
  LucesON "$Objeto$"
#END
#PLUGIN: OpenDomo_stop
<A,a>diós OpenDomo
  OpenDomo_stop
#END
#PLUGIN: OpenDomo_start
Hola OpenDomo
  OpenDomo_start
#END
```

Este resumen está detallado en formato de diccionario, de forma que, como se puede ver, cada comando está definido en un Plugin, que contiene la frase a reconocer y en la línea siguiente los scripts/comandos a ejecutar.

8 DESARROLLO DE LA APLICACIÓN (OPENDOMO VR)

En este apartado se va a describir el software desarrollado para el reconocedor de voz (OpenDomoVR). Una primera fase va a consistir en detallar que soluciones se adoptaron para implementar cada uno de los requisitos funcionales definidos en el apartado anterior.

8.1 Implementación de los requisitos funcionales

8.1.1 Multilenguaje

Para implementar este requisito, el sistema se basa en leer de la configuración del sistema OpenDomo el lenguaje en un fichero de configuración existente /etc/opendomo/lang.

En fase de configuración el sistema generará los VoiceCommands correspondientes al idioma, organizado en un subdirectorio por cada idioma (/usr/local/opendomo/voiceCommands/en ó /usr/local/opendomo/voiceCommands/es).

En fase de ejecución se seleccionará el lenguaje a enviar al API de Google de forma que Google sepa qué lo que se le envía es voz en castellano o en inglés.

El idioma debe implementarse también en los textos de los mensajes sintetizados a enviar al usuario. Para ello, en fase de configuración se define el parámetros del idioma en un fichero de configuración (espeak.dat) que contiene los parámetros de síntesis de voz:

- Lenguaje
- Personalización (voz masculina o femenina)

8.1.2 Opendomo VR “Always ON”

El reconocedor debe estar corriendo siempre, y debe autoarrancarse cuando se arranque el sistema.

Para ello se ha creado un script de autoarranque (opendomoVRd) que hace que se arranque con todos los niveles de inicialización (2..6). Como comentario general, dado que para arrancar el reconocedor es necesario que estén activos el resto de servicios: sonido, red con conexión, etc, se arranca con update-rc.d en una de las posiciones últimas (99).

8.1.3 Estados del sistema

Como se ha comentado anteriormente, OpenDomoVR debe de contener al menos dos estados:

- Identificación: Es es estado inicial. Solo se reconoce una frase llave. por ejemplo: “Hola OpenDomo”. Mientras esta frase no sea reconocida, el reconocedor permanecerá en este estado. En cuanto se reconozca la frase llave pasará al estado “Normal”.
- Normal: Este es el estado de reconocimiento. En él se reconocerán todos los comandos. Para salir de este estado, se usará una frase llave de despedida (por ejemplo, “Adios OpenDomo”) que llevará al sistema al estado de Identificación.

Para implementar esta feature se han definido:

- Un fichero de configuración llamado MODE que tiene dos estados: Normal, Identification.
- Dos diccionarios: el existente (normal.dic) con todos los comandos a reconocer e identification.dic que solo contiene el comando de identificación (suele ser "Hola OpenDomo"). Cada uno de ellos está replicado por idioma.

De esta forma cada vez que se intenta reconocer voz, se mira en el fichero MODE que modo de identificación está activo lo que implica usar un diccionario u otro para reconocer la voz.

8.1.4 Reconocimiento automático, detectando silencio

Esta feature fue y sigue siendo una de las más problemáticas. El objetivo es poder diferenciar silencio de voz. Para ello, se ha creado un script (autodetect.sh) que está siempre corriendo y analizando muestras de sonido, correlando unas muestra con las muestras del segundo siguiente. Si son iguales, se identifica como silencio. Si son diferentes, ha habido "algo". A continuación se mete en un bucle para detectar el fin de la voz (o sonido): detectar de nuevo silencio, en este caso, desplazando las muestras 0,5 seg.

```
#!/bin/bash

rec recording.flac rate 16k silence 1 0.1 3% -1 3.0 3% &
p=$!
sleep 1
until [ "$var1" != "$var2" ]; do
    var1=`du "recording.flac"`
    sleep 1
    var2=`du "recording.flac"`
done
echo "Sound Detected"
until [ "$var1" == "$var2" ]; do
    var1=`du "recording.flac"`
    sleep 0.5
    var2=`du "recording.flac"`
done
echo "Silence Detected"

kill $p

RESULT="$(./send_speech recording.flac)"
./recognize "$RESULT"
```

8.1.5 Detección automática de objetos a reconocer

Para este fin, se genera en el momento del arranque de la aplicación una estructura de directorios que se almacenan en un directorio específico: /etc/opendomo/speech, con los datos de configuración de cada tipo de puerto. En este directorio se crean ficheros: light.conf, climate.conf, video.conf, music.conf, etc, con los datos de configuración (interruptores de cada tipo distribuidos por habitaciones, controles de temperatura, sonido, cámaras de vídeo...).

8.1.6 Actuación sobre el HW de control de OpenDomo

Lo que debe hacer OpenDomoVR, al final, es actuar sobre los controles de los distintos subsistemas de gestión de la vivienda: luz, climatización, música, cámaras de vídeo, etc. Estos controles son puertos de un tipo de hardware (OD Control, Arduino..) que están accesibles como memoria compartida en el sistema OpenDomo.

Para manejarlo, OpenDomo los declara en /var/opendomo/control donde se define por un lado las zonas (habitaciones) y en cada zona un fichero con los controles disponibles y las posibles acciones. Por ejemplo, hay controles ON/OFF (activar/desactivar luces, por ejemplo), o con valores analógicos(temperatura de un termostato).

Para actuar sobre estos controles, basta con modificar el fichero correspondiente. De esta forma, OpenDomoVR decide primero si el puerto que pide el usuario está accesible (si está definido el control en el fichero correspondiente de /etc/opendomo/speech) y si es así, actúa sobre el fichero correspondiente al puerto.

En caso de que el puerto sobre el que se pretende actuar no exista, el sistema enviará un mensaje vocal de error.

8.2 Consideraciones de diseño de los requisitos no funcionales

4. Debe permitir entorno CGI
5. No debe de contener ningun elemento gráfico (Palaver tiene ventanas de notificación)
6. Debe de minimizarse el tamaño del software de base, así como de las dependencias, dado que OpenDomo es una distribución mínima

4. El sistema no debe poseer software a compilar, dado el tamaño reducido de la distribución.
5. El reconocedor de voz generará los datos de reconocimiento cuando se arranque la aplicación, de forma que se minimice el consumo de recursos en fase de reconocimiento. De esta forma, si se realiza algun cambio al sistema OpenDomo, debe reiniciarse automáticamente la aplicación para que contemple los nuevos datos.
6. El reconocedor de voz debe ser autoinstalable, de acuerdo con la metodología de instalación de proyectos dentro de OpenDomo. De acuerdo con esta metodología, el proyecto debe empaquetarse en un fichero .tar,gz, que se desempaquetará en cada instalación de cliente.

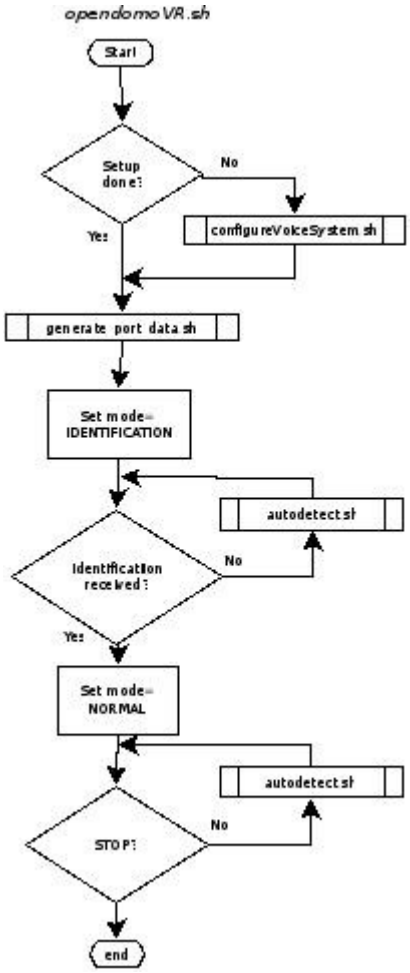
8.3 Descripción funcional del software de la aplicación

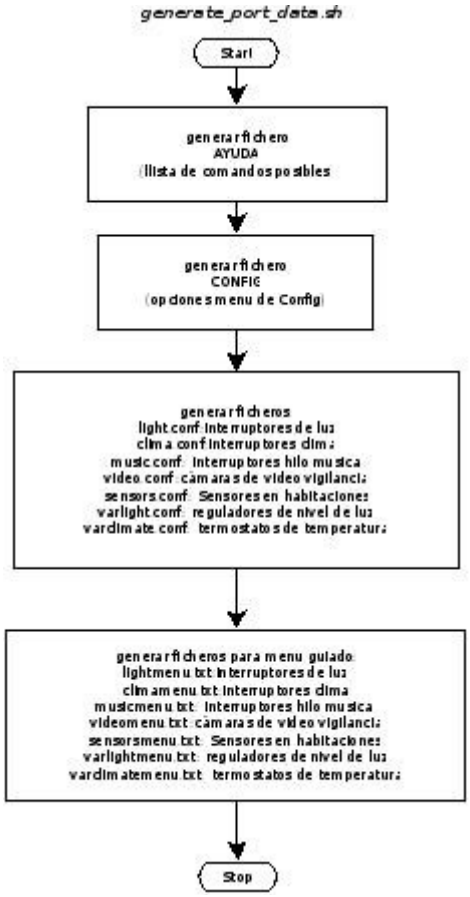
En este apartado se van a detallar por medio de diagramas los elementos del software, de forma que se entienda perfectamente su comportamiento. Este detalle no es exhaustivo: ni están todos los programas ni se detallan todas las instrucciones, sino que se hace un resumen esquemático del funcionamiento de la aplicación.

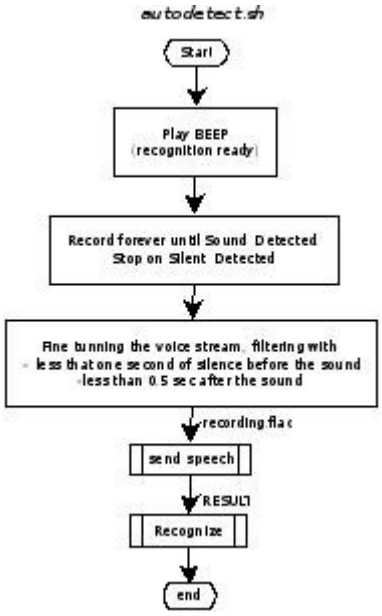
8.3.1 Software operacional

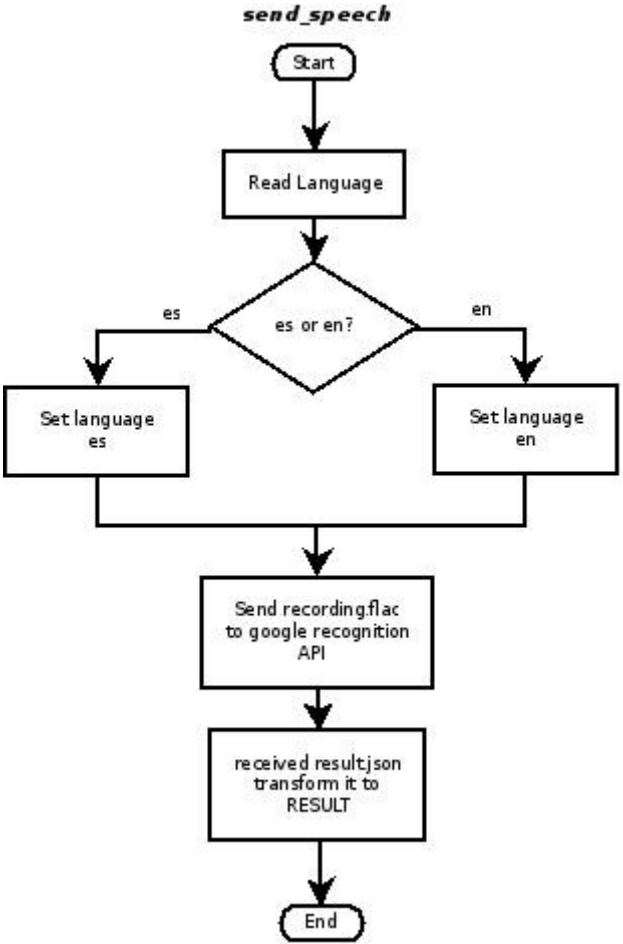
El software operacional consiste en el proceso opendomoVR.sh que está corriendo siempre que esté activo el reconocedor de voz.

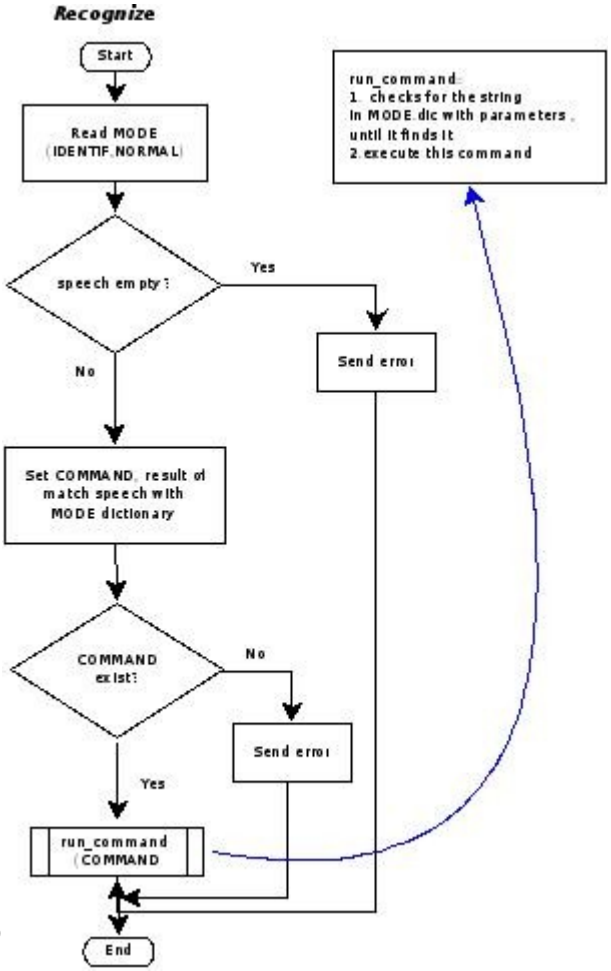
A continuación se muestran los diagramas de flujo de los módulos implicados:


opendomoVR.sh
Start
Setup done?
No
configureVoiceSystem.sh
Yes
generate_port_data.sh
Set mode= IDENTIFICATION
autodetect.sh
Identification received?
No
Yes
Set mode= NORMAL
autodetect.sh
STOP?
No
end


generate_port_data.sh
Start
generar fichero AYUDA (llista de comandos posibles)
generar fichero CONFIG (opciones menu de Config)
generar ficheros light.conf: Interruptores de luz clima.conf: Interruptores clima music.conf: Interruptores hilo musica video.conf: cámaras de video vigilancia sensors.conf: Sensores en habitaciones varlight.conf: reguladores de nivel de luz vardimate.conf: termostatos de temperatura
generar ficheros para menu guiado: lightmenu.txt: Interruptores de luz climamenu.txt: Interruptores clima musicmenu.txt: Interruptores hilo musica videomenu.txt: cámaras de video vigilancia sensorsmenu.txt: Sensores en habitaciones varlightmenu.txt: reguladores de nivel de luz vardimatemenu.txt: termostatos de temperatura
Stop


autodetect.sh
Start
Play BEEP (recognition ready)
Record forever until Sound Detected Stop on Silent Detected
Fine tunning the voice stream, filtering with - less than one second of silence before the sound - less than 0.5 sec after the sound
recording.flac
send speech
RESULT
Recognize
end

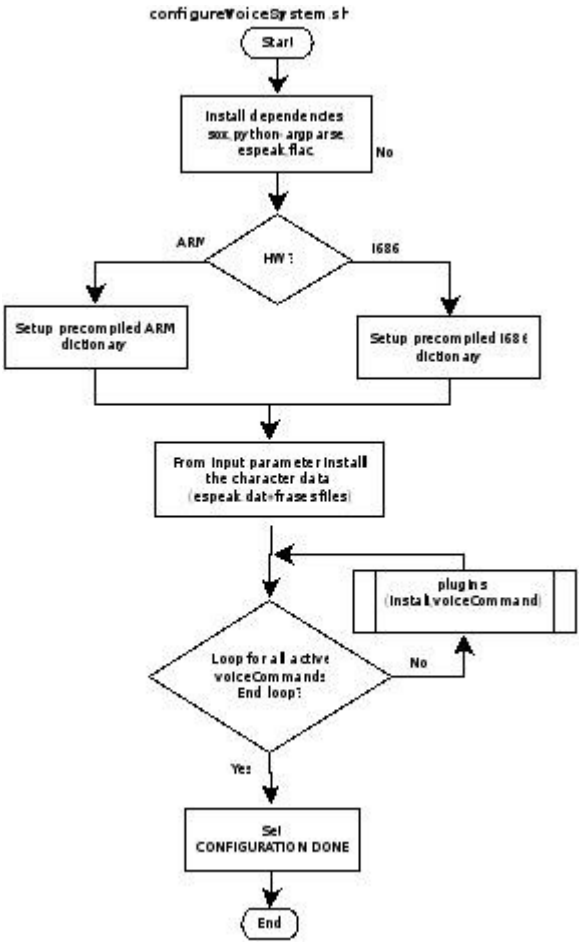

send_speech
Start
Read Language
es or en?
es
en
Set language es
Set language en
Send recording.flac to google recognition API
received result.json transform it to RESULT
End


Recognize
Start
Read MODE (IDENTIF,NORMAL)
speech empty?
Yes
Send error
No
Set COMMAND, result of match speech with MODE dictionary
COMMAND exist?
No
Send error
Yes
run_command (COMMAND)
End
run_command:
1. checks for the string in MODE.dic with parameters, until it finds it
2. execute this command

Como comentario adicional, la parte más crítica en tiempo del reconocedor es , además del tiempo de respuesta del API de Google, la parte de identificación del comando en el diccionario. Esta parte se realiza por un ejecutable en C, llamado “dictionary” que se implementa en la caja “Set COMMAND result of matching speech with MODE dictionary” en el proceso “recognize”. Este ejecutable es muy simple (consiste básicamente en un grep ejecutado en C) pero es crítico ya que el diccionario puede crecer mucho en tamaño y por ello, se pueden prolongar los tiempos de ejecución. Por eso está realizado en C, para evitar dependencias de compilación existe un ejecutable por tipo de arquitectura (x86 y ARM) que se instala en el momento de la configuración del reconocedor.

8.3.2 Software de configuración de la instalación

Para configurar la instalación se usa el script configureVoice System.sh, que tiene como parámetro el personaje que va a contestar a las peticiones del usuario. Los personajes definen el tipo de voz (masculina, femenina), la gravedad del tono vocal, la velocidad de la voz, y las frases de respuesta: Se puede configurar el fichero de frases del personaje de forma que el sistema de una respuesta más formal o más directo.



8.3.3 software de configuración de voiceCommands

Para la configuración de voiceCommands se usan dos programas: sdk y plugins (escritos en Python). Para su funcionamiento, se parte de que existen dos estructuras de carpetas:

- Estructura de carpetas de configuración de voiceCommands:

Esta carpeta se usa para crear y modificar los voiceCommands antes de instalarlos en OpenDomoVR y tiene la siguiente estructura:

VoiceCommand subfolder

VoiceCommand.sp file +plugin.info file

Modes/main.dic

Bin/voiceCommand exe file

Conf/settings.conf

De esta forma en el subdirectorio modes se guarda el diccionario del voiceCommand y en el bin el ejecutable del voiceCommand, dejando en el raíz el empaquetado del voiceCommand (que contiene el diccionario, el ejecutable, el fichero de configuración y el pugin.info empaquetado en .tar.gz)

-Estructura de ejecución:

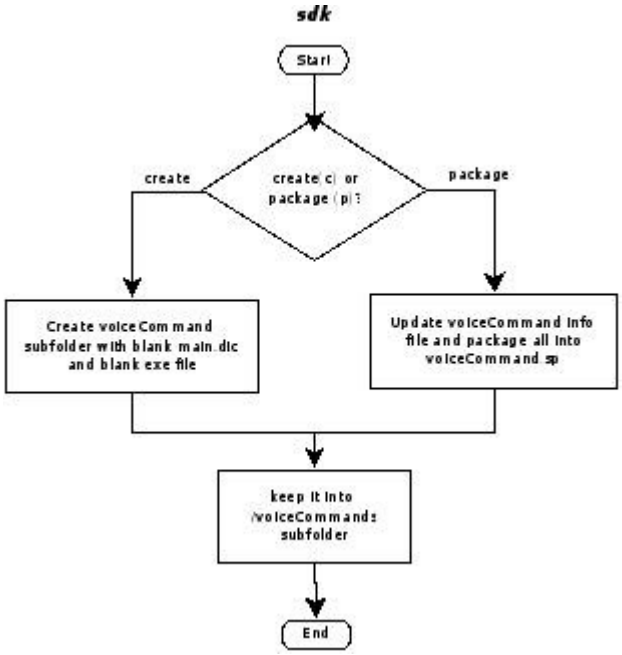
En el directorio vr hay una carpeta:

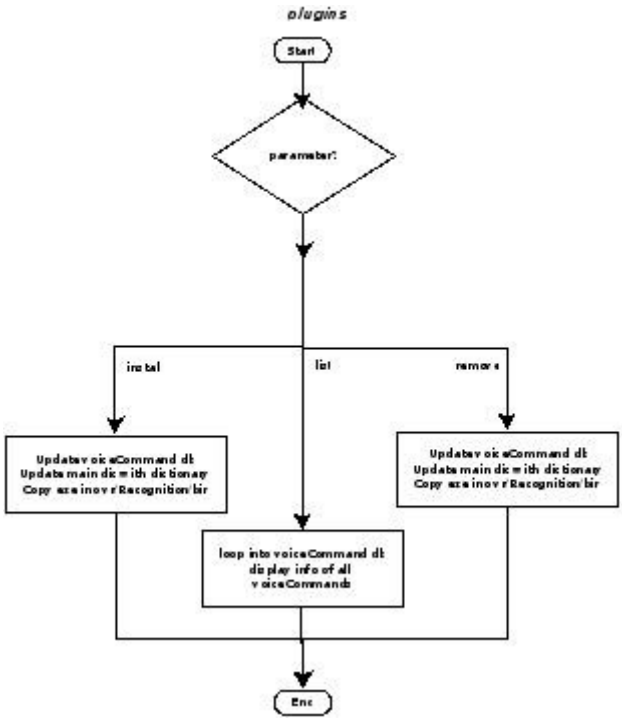
Recognition

Modes/diccionarios por modo (identification.dic y main.dic)

Bin/ejecutables de todos los voiceCommands

De esta forma, cuando se instala un nuevo voiceCommand, se actualiza el diccionario común (main.dic) con el contenido del diccionario del voiceCommand y se añade el ejecutable al directorio /Recognition/bin

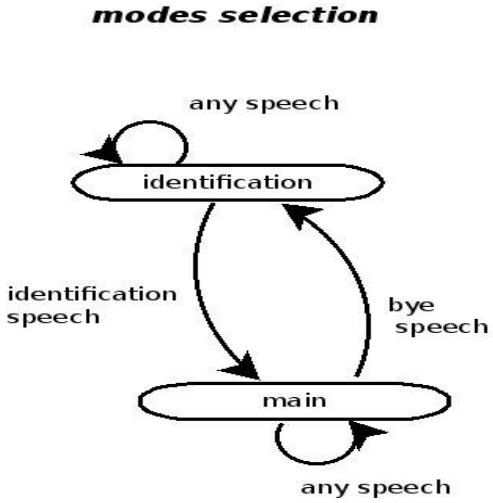




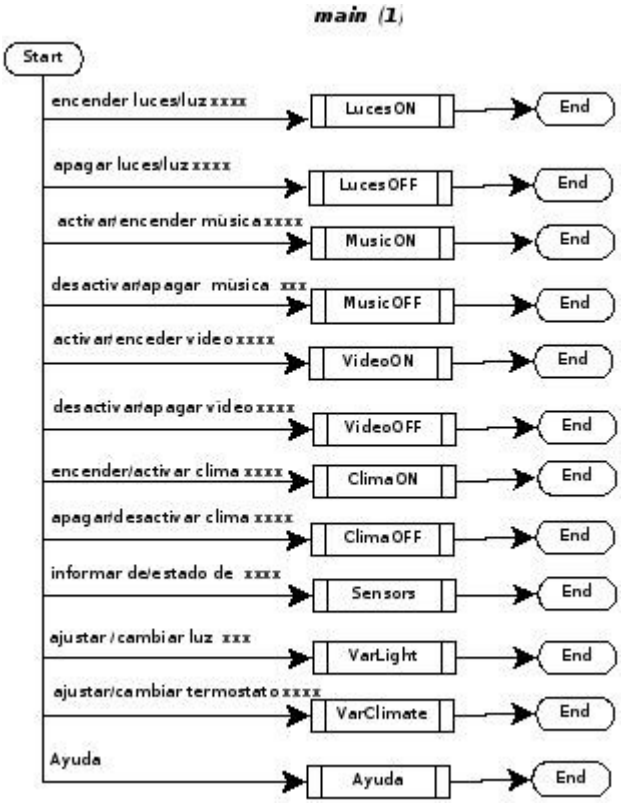
8.3.4 voiceCommands instalados (lógica de OpenDomoVR)

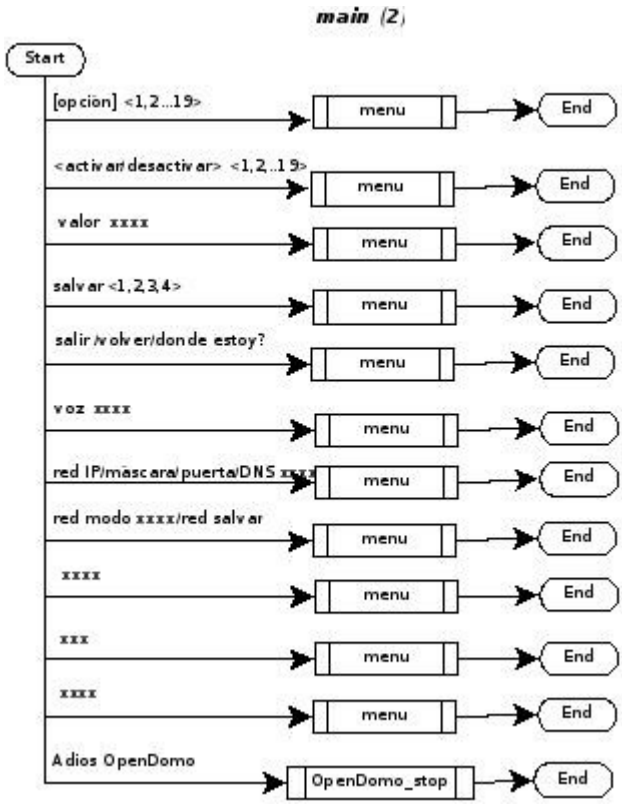
Realmente, la lógica de OpenDomoVR la constituyen los voiceCommands instalados en cada momento. El esqueleto lo forma el diccionario activo en cada momento: identification.dic y main.dic.

El diagrama de estados del sistema es el siguiente:



Esta es la estructura del main.dic (se ha dividido en dos partes el diagrama):





Como los voiceCommand son todos bastante similares, voy a desglosar solo una muestra de ellos. En todos los casos, todos los voiceCommands se basan en que al arrancar el reconocedor, se generan los ficheros con los datos correspondientes a la instalación en el directorio /etc/opendomo/speech, con lo que cada voiceCommand chequea los datos de estos ficheros y con eso genera y valida las posibles respuestas del usuario.

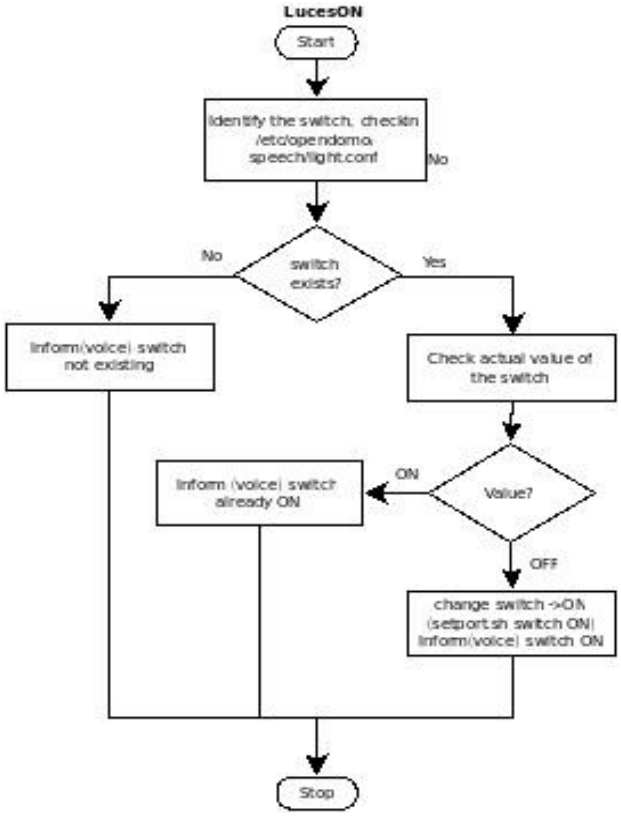
Para que el reconocedor sea utilizable por personas con discapacidad visual, se ha creado el voiceCommand “menu,” que permite hacer las mismas acciones que se harían desde el interfaz web con interfaz vocal. Desde este menú se pueden hacer todas las actuaciones sobre OpenDomo. Dado que el menú de OpenDomo es variable, este menú es también variable ajustándose a las entradas que posee cada instalación.

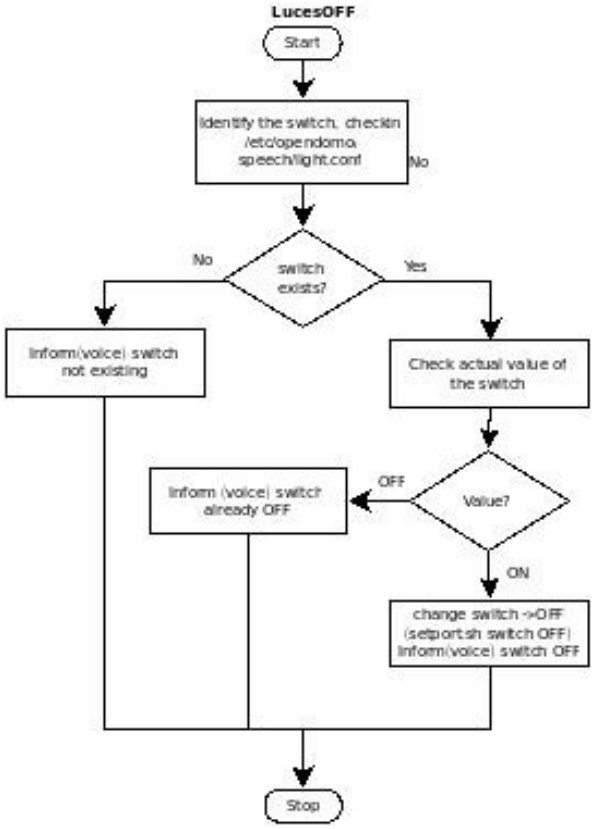
Como se puede ver en el main.dic, hay varios tipos de voiceCommands:

- Activación de Elementos: Luces, Música, Vídeo, Clima.
- Desactivación de Elementos: Luces, Música, Vídeo, Clima
- Comandos para configurar elementos variables: temperatura de una sala, nivel de luz de una sala
- Comandos para obtener información de sensores
- Comandos de apertura y cierre de sesión del reconocedor de voz
- Comandos de ayuda
- Menú guiado

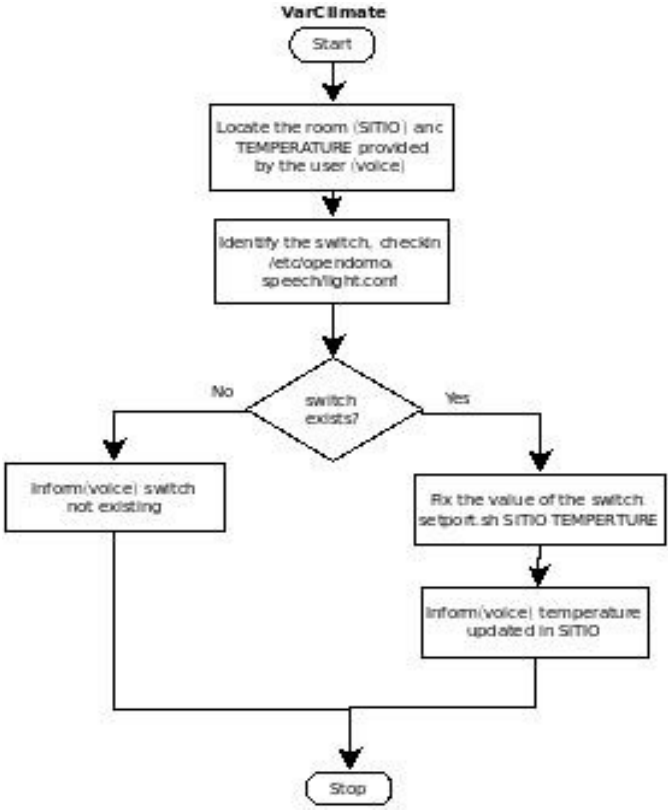
Por ello, voy a describir solo un comando significativo de cada tipo ya que la lógica es igual y solo cambia el elemento sobre el que se actúa.

- Comandos de activación/desactivación de elementos: LucesON y LucesOFF (los comandos MusicON/OFF, ClimateON/OFF, VideoON/OFF, son similares).

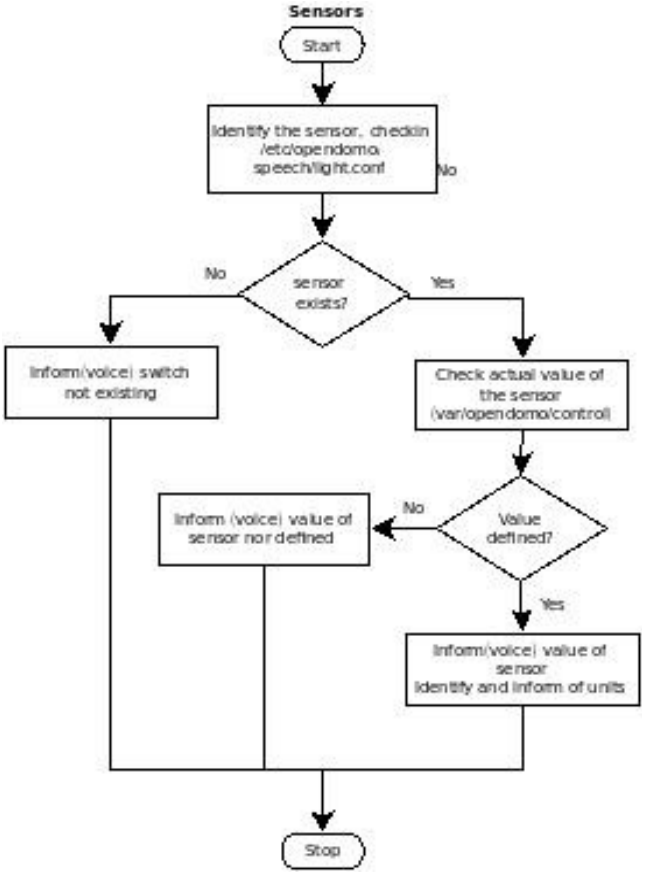




- Comandos de configuración de elementos variables. Se va a describir VarClimate, que ajusta el termostato (el comando varLight que modifica el nivel de iluminación es similar):

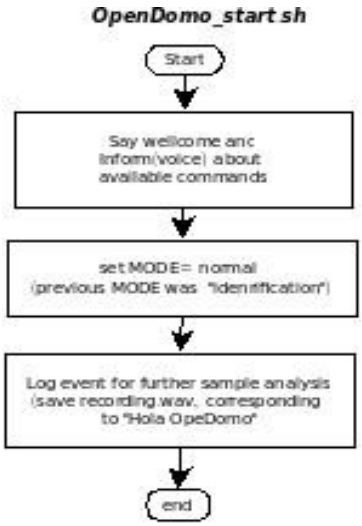


- Comandos para obtener información de sensores. Es un único comando: Sensors. Este comando , debe de leer también la información de unidades: grados, sensor de apertura (abierto, cerrado)... Las unidades están definidas en el fichero de configuración del sensor (en /etc/opendomo/control).



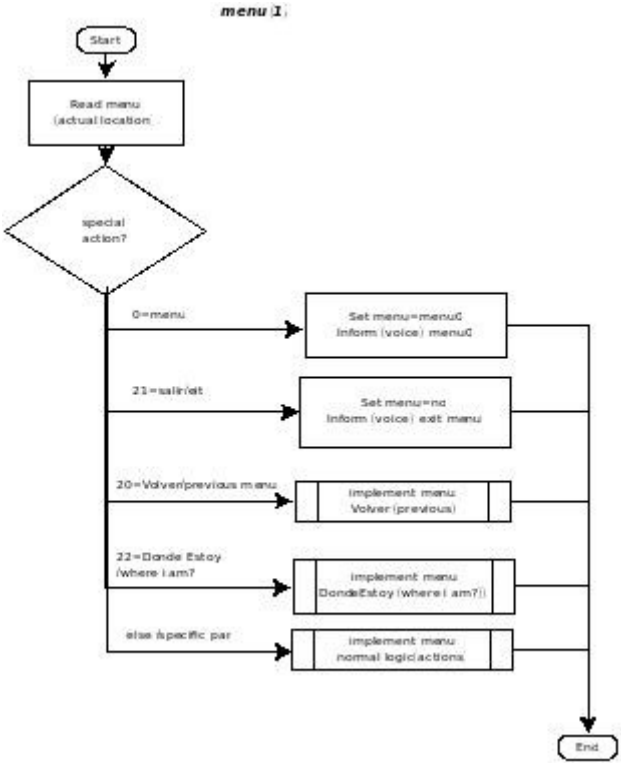
- Comandos de apertura y cierre de sesión del reconocedor de voz Son dos: OpenDomo_start y OpenDomo_stop. OpenDomo_stop es similar a OpenDomo_Start, cambiando el

MODE=identification (no permitiendo así ningún comando hasta que se vuelva a dar el comando de bienvenida (Hola OpenDomo).

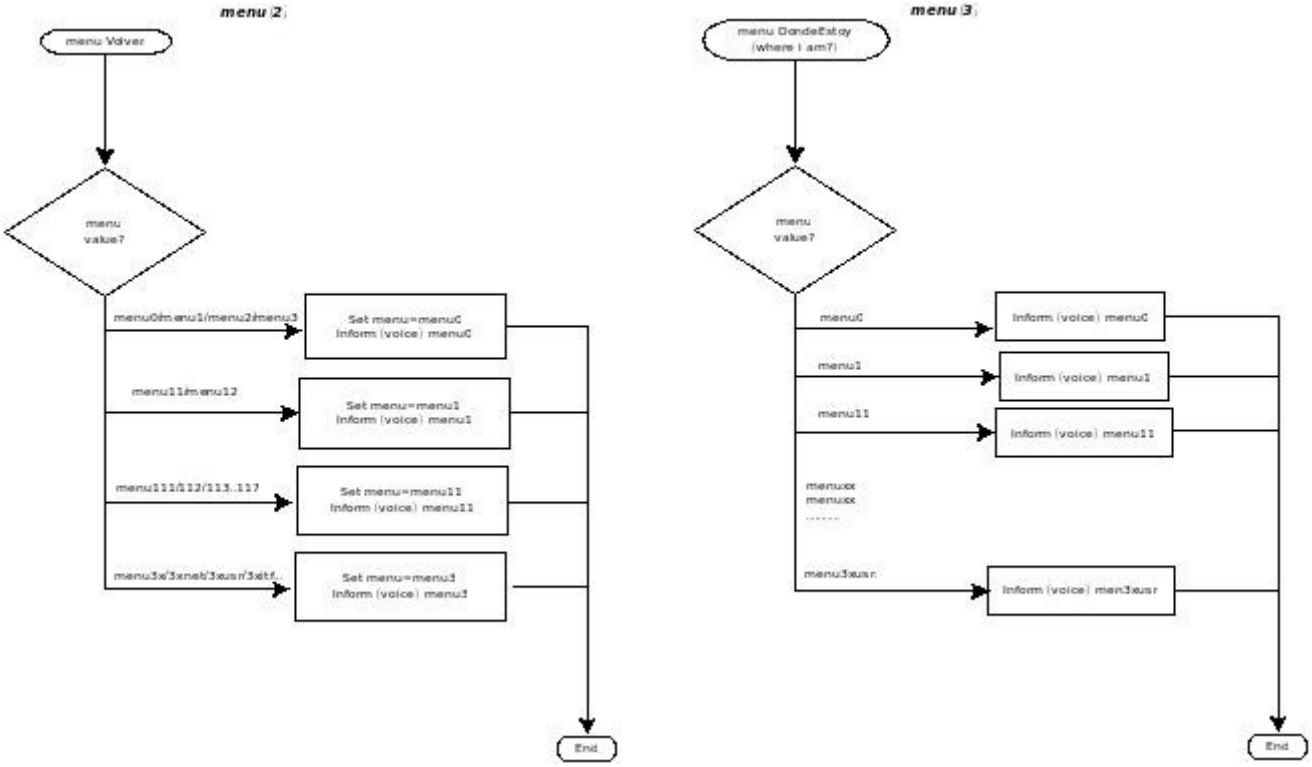


- Comandos de ayuda (Ayuda). Este comando informa de todos los elementos disponibles en la instalación, y los posibles valores. Realmente, toda la lógica está hecha en “generate_port_data.sh” que rellena el fichero /etc/opendomo/speech/AYUDA. El comando Ayuda solo informa por voz del contenido del fichero AYUDA.

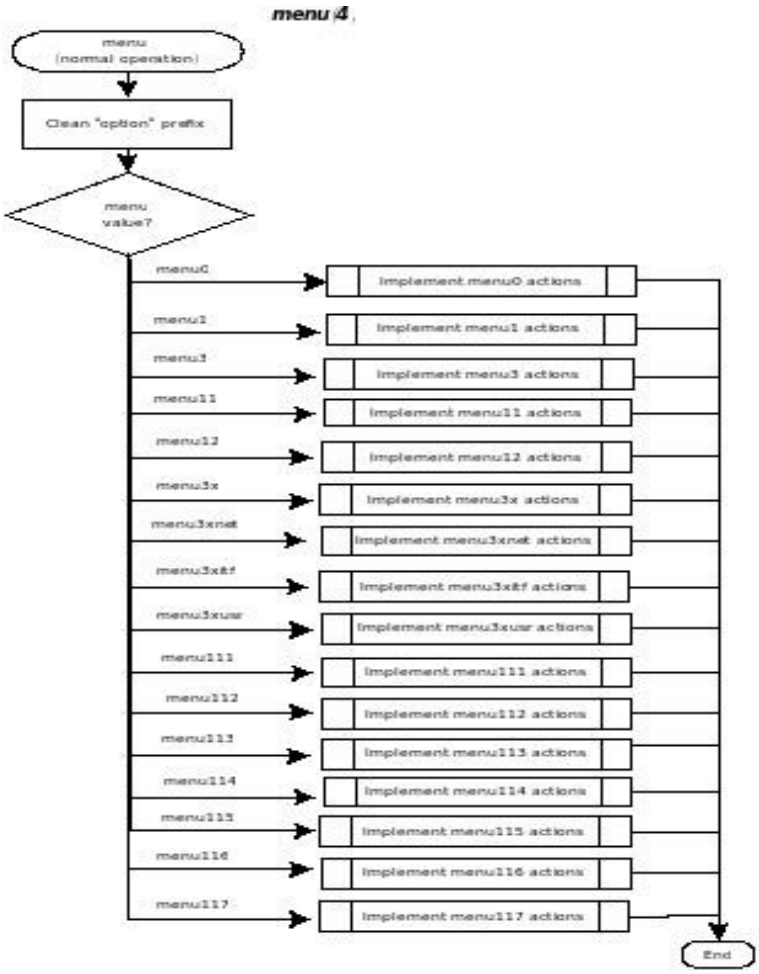
- Menú guiado. Este comando es el macrocomando. Permite cualquier operación sobre la instalación. Dado que este ejecutable es muy grande, se va a dividir en varios diagramas.



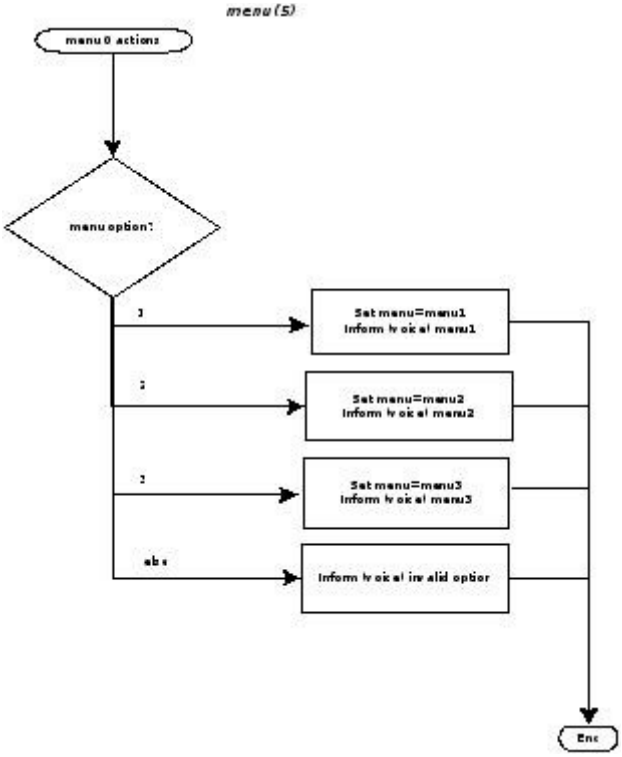
Estos dos menús implementan las funciones volver al menú anterior e indicar “¿dónde estoy?” con las posibles acciones:

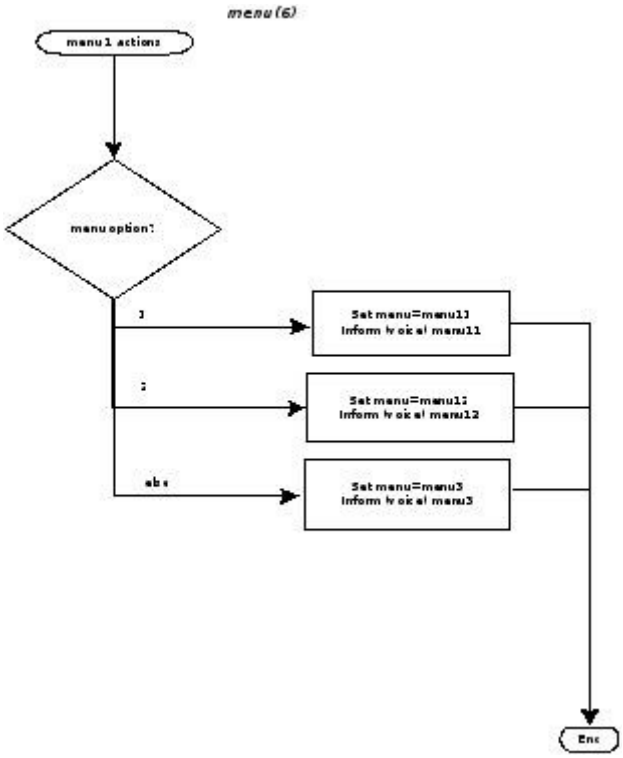


A continuación se describe el funcionamiento normal de menú (son las opciones que implementan las acciones del menú):

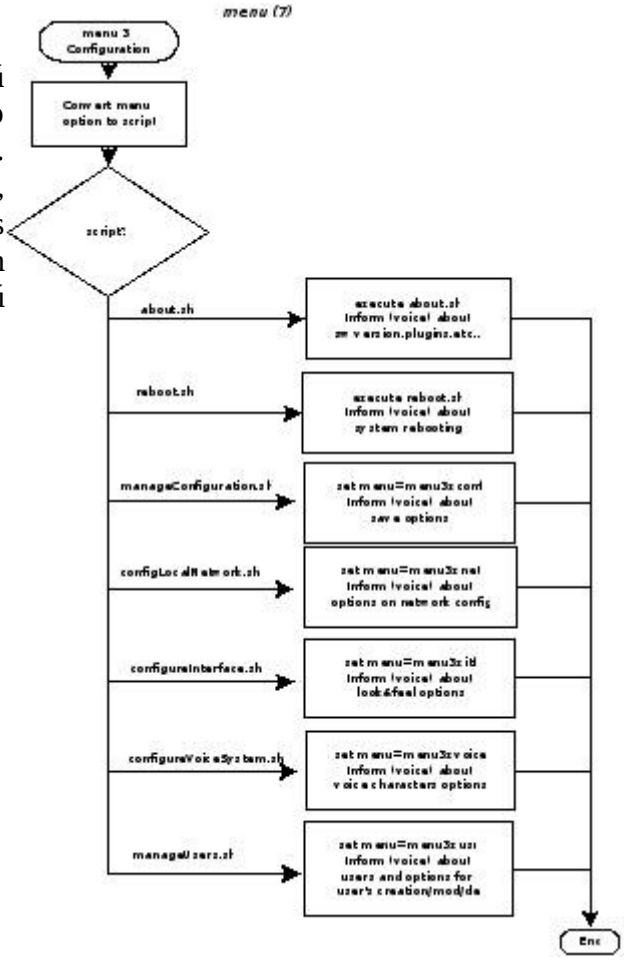


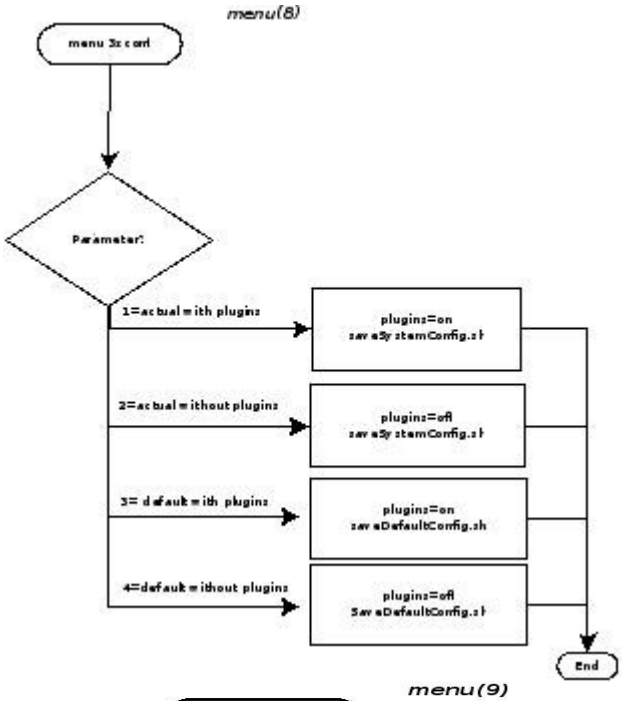
Por último, se va a desarrollar cada una de las entradas de menú:

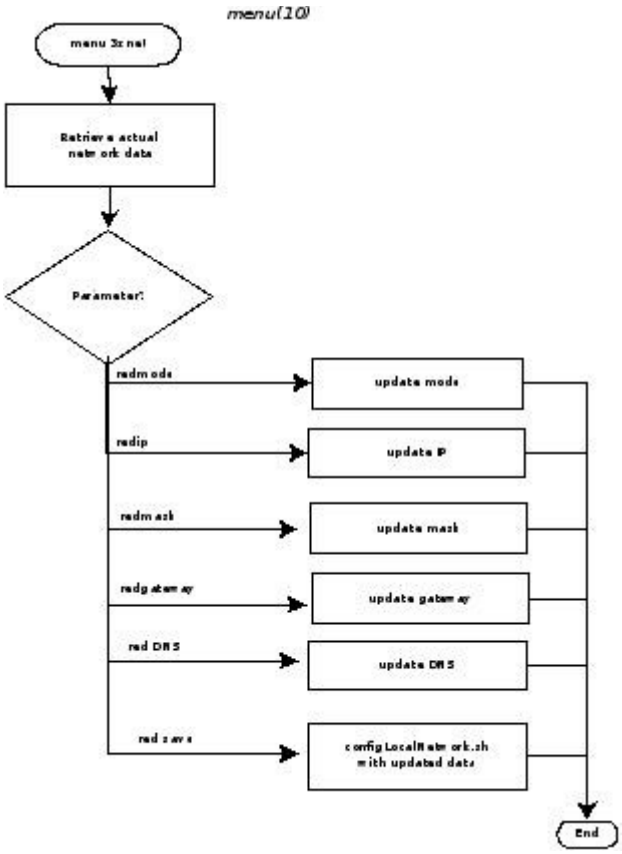


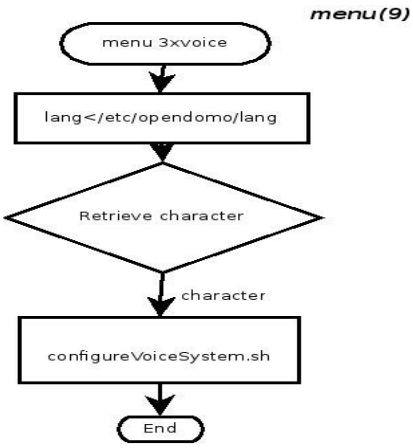


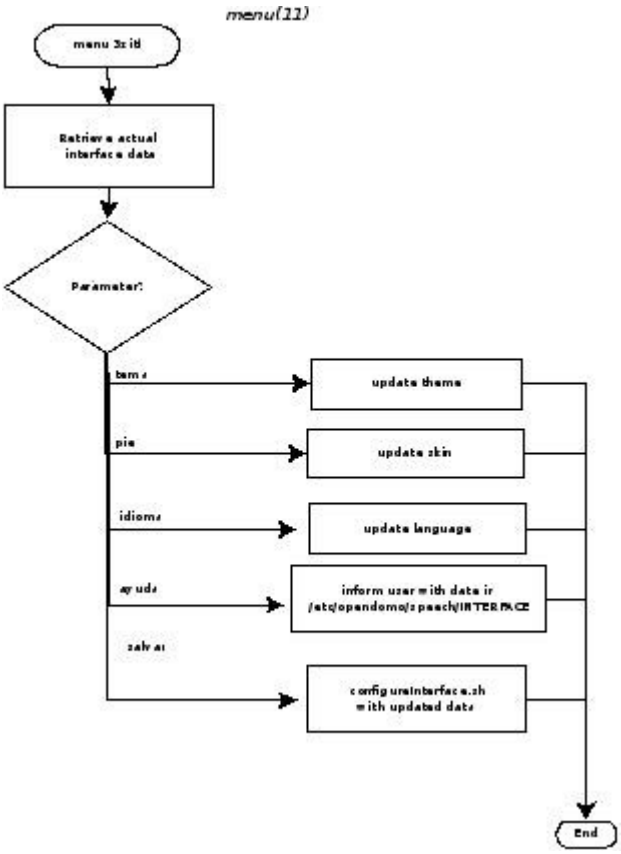
Menú 3 es el menú de configuración. Desde este menú se lanzan los scripts equivalentes a los que el usuario introduciría desde el menú para configurar el sistema. Estos scripts dependen de cada instalación y por ello, lo que se hace es recolectar todos los posibles scripts en generate_port_data.sh, asociando a cada script un número que introducirá el usuario en este menú dependiendo de lo que quiera configurar.

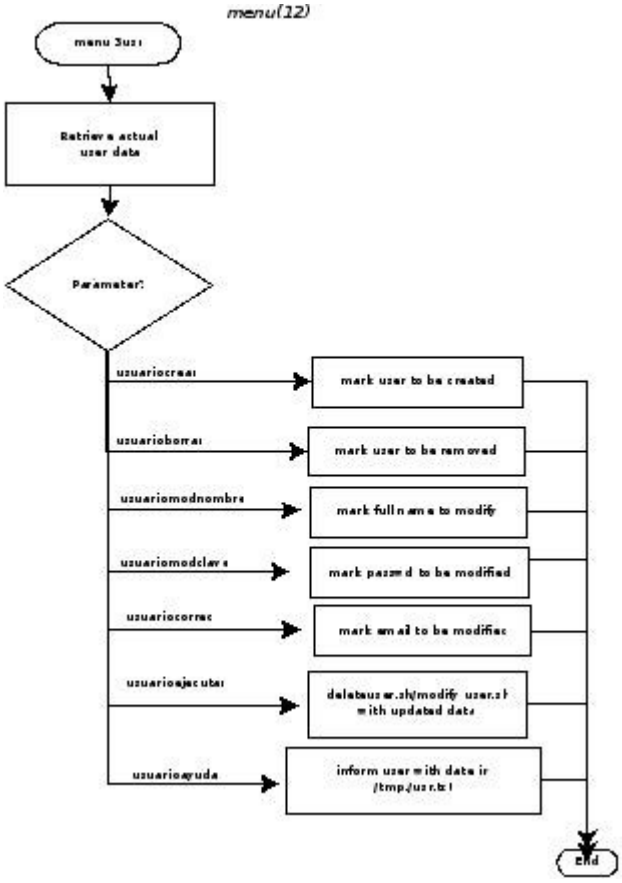


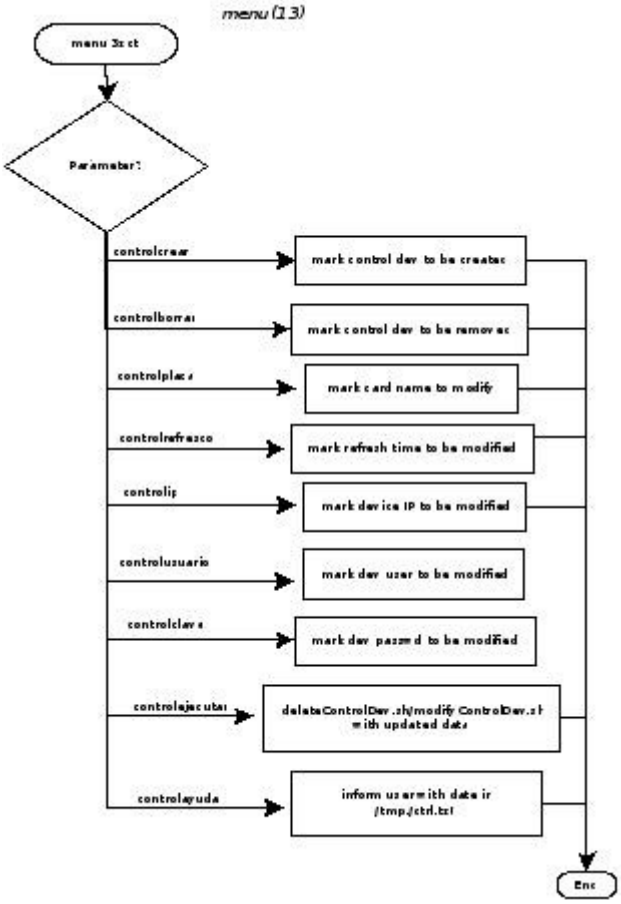

menu(8)
menu 3xcorl
Parameter?
1=actual with plugins
plugins=on
saveSystemConfig.sh
2=actual without plugins
plugins=off
saveSystemConfig.sh
3= default with plugins
plugins=on
saveDefaultConfig.sh
4=default without plugins
plugins=off
SaveDefaultConfig.sh
End

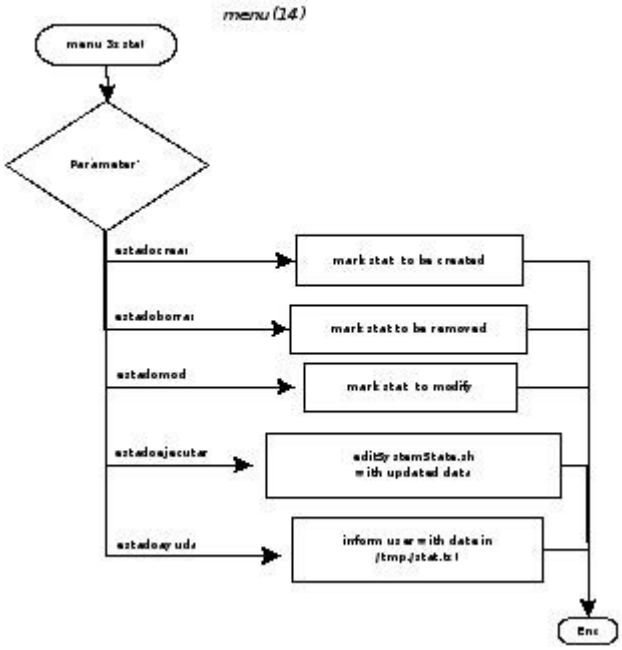

menu(10)
menu 3xnet
Retrieve actual network data
Parameter?
redmode
update mode
redip
update IP
redmask
update mask
redgateway
update gateway
red DNS
update DNS
red save
configLocalNetwork.sh
with updated data
End

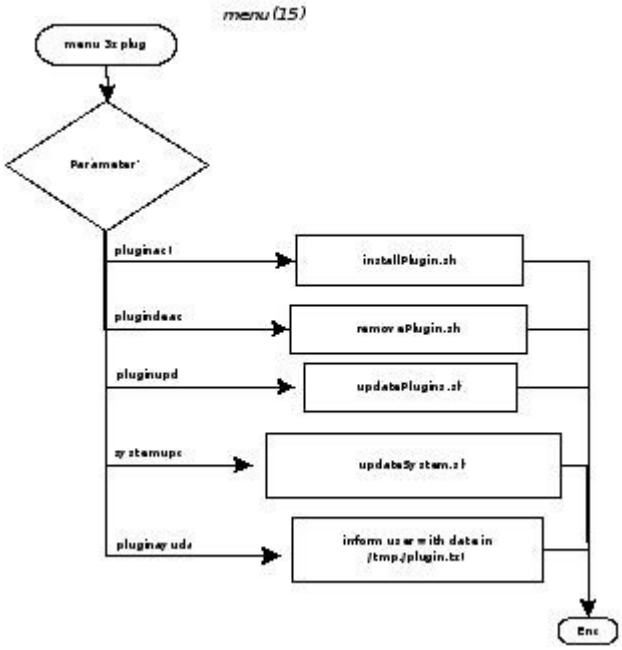

menu(9)
menu 3xvoice
lang</etc/opendomo/lang
Retrieve character
character
configureVoiceSystem.sh
End

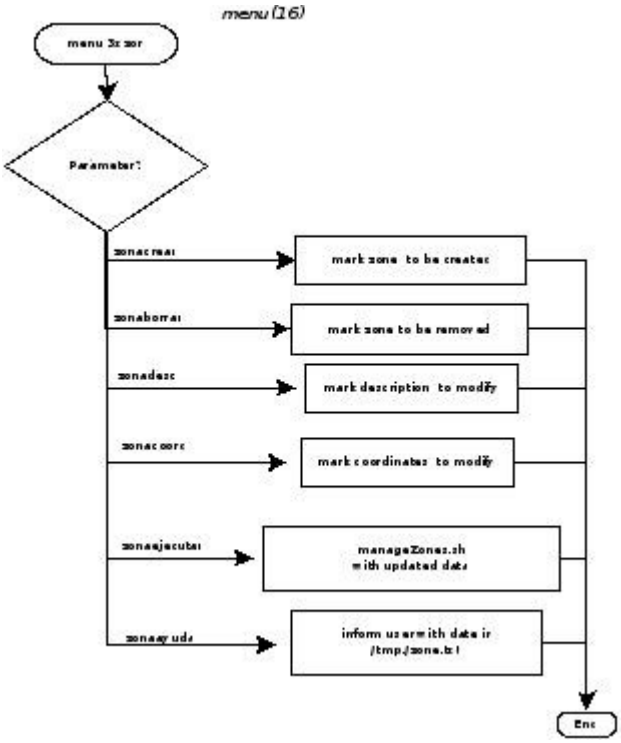

menu(11)
menu 3xitl
Retrieve actual interface data
Parameter?
tema
update theme
piel
update skin
idioma
update language
ayuda
inform user with data in /etc/opendomo/speech/INTERFACE
salvar
configureInterface.sh
with updated data
End


menu(12)
menu 3usr
Retrieve actual user data
Parameter?
uzuariocrear
mark user to be created
uzuarioborrar
mark user to be removed
uzuariomodnombre
mark full name to modify
uzuariomodclave
mark passwd to be modified
uzuariocorreo
mark email to be modified
uzuarioejecutar
deleteuser.sh/modify user.sh
with updated data
uzuarioayuda
inform user with data in /tmp/usr.txt
End


menu (13)
menu 3: ct
Parámetro?
controlcrear
mark control dev to be created
controlborrar
mark control dev to be removed
controlplaca
mark card name to modify
controlrefresco
mark refresh time to be modified
controlip
mark device IP to be modified
controluzuario
mark dev user to be modified
controlclave
mark dev passwd to be modified
controlejecutar
deleteControlDev.sh/modifyControlDev.sh
with updated data
controlayuda
inform user with data in
/tmp/ctrl.txt
Fin


menu (14)
menu 3: stat
Parámetro?
estadocrear
mark stat to be created
estadoborrar
mark stat to be removed
estadomod
mark stat to modify
estadoejecutar
editSystemState.sh
with updated data
estadoayuda
inform user with data in
/tmp/stat.txt
Fin


menu (15)
menu 3: plug
Parámetro?
pluginact
installPlugin.sh
plugindeac
removePlugin.sh
pluginupd
updatePlugins.sh
systemupd
updateSystem.sh
pluginayuda
inform user with data in
/tmp/plugin.txt
Fin


menu (16)
menu 3: zon
Parámetro?
zonacrear
mark zone to be created
zonaborrar
mark zone to be removed
zonadesc
mark description to modify
zonacoord
mark coordinates to modify
zonaejecutar
manageZones.sh
with updated data
zonaayuda
inform user with data in
/tmp/zone.txt
Fin

9 PRUEBAS Y VERIFICACIÓN

En este apartado voy a comentar los cambios que ha habido que hacer a raíz de los problemas encontrados en las pruebas, así como las complicaciones encontradas en el desarrollo del producto.

9.1 Entorno de prueba

Dependiendo de la fecha se ha probado en diferentes entornos:

- Antes de diciembre 2013, se probó en entorno local Debian, simulando las entradas de Opendomo

- Desde diciembre de 2013 hasta febrero 2013, se probó en entorno virtualizado con una distribución de OpenDomo basada ya en Debian, y con la gestión de paquetes de Debian

- A partir de marzo de 2014 se probó en entorno pseudo real, con el primer beta de OpenDomo2.0 basado en Raspberry Pi.

Para las pruebas en este entorno final se preparó un script de forma que se editaba el código en un PC y se pasaba inmediatamente a Github, generando el nuevo paquete desde Github en la Raspberry Pi con el plugin Odevel . De esta forma, el proceso de actualización se agiliza mucho y permitiéndonos probar en entorno real casi inmediatamente, pudiendo corregir los errores casi inmediatamente, sobre el entorno real. El proceso completo después de una modificación de código hasta tenerlo instalado en la Raspberry, tarda del orden de unos 2 minutos.

9.2 Issues detectados en las pruebas

9.2.1 Adaptaciones para el reconocimiento de voz contínuo

Para este apartado, además de desarrollar un sistema basado continuo basado en detectar cambios en el nivel de entrada de audio para diferenciar tramas de voz y silencio, hubo que hacer varias adaptaciones en su funcionamiento, teniendo que cumplir básicamente dos requisitos:

- El sistema ha de detectar comienzo de trama de voz, intentando no perder ninguna trama al comienzo, cuando el usuario comienza a emitir voz.

- Aunque, originalmente las tramas de voz eran cortas: no más de 3 segundos, agilizando el proceso de forma que como máximo en 3 segundos se enviaban tramas al API de Google, debido a la introducción del voiceCommand “menu” fue necesario incrementar los tiempos, dado que en este voiceCommand existen tramas largas. Por ejemplo, “red IP 192 168 1 1”.

De esta forma, se fijaron los tiempos en 0,1 segundos para detectar trama después del silencio (comienzo de trama) y 15 segundos para auditar el fin de trama.

Otro problema que ocurría a veces es que, debido a que el reconocimiento de voz corre en el background, existían casos en que, dependiendo de los tiempos, el sistema se quedaba muerto, no existiendo el proceso del comando “rec” que realiza la exploración del audio de entrada. Para subsanar este problema hubo que crear una auditoría que corre en el background siempre que esté activo el proceso opendomoVR. Esta auditoría (audit_speech_detect.sh) corre periódicamente, actualmente cada 60 segundos, de forma que si detecta que no existe proceso de recording, existiendo proceso de opendomoVR, mata el proceso aurodetect.sh para que opendomoVR abra un proceso nuevo de reconocimiento. Este proceso lo hará mientras exista un proceso de opendomoVR. Cuando no ocurra, el proceso se “suicidará”, dado que el reconocimiento de voz no está activo.

9.2.2 Adaptaciones en la integración con HW real (Raspberry Pi)

Dado que el hardware más común en una instalación OpenDomo es la Raspberry Pi, en un momento dado decidí instalar una Raspberry Pi en casa para hacer la prueba en un entorno pseudo-real. Creo que ésto fue un gran acierto porque de otro modo no se hubieran detectado los problemas bloqueantes existentes en la instalación del reconocedor con Raspberry PI; los problemas que se detectaron estaban relacionados con los siguientes puntos:

- La Raspberry Pi no posee entrada de audio

- La distribución en la Raspberry Pi de una instalación OpenDomo está restringida a un tamaño muy reducido, instalado en una tarjeta SD de 2 Gb.

- La arquitectura de la Raspberry se basa en hardware ARM, por lo que es necesario compilar los programas que son dependientes de la arquitectura. En nuestro caso, el buscador de comandos "dictionary.c" está hecho en C.

Para el primer problema la alternativas estaban en instalar hardware de audio Bluetooth, instalar una tarjeta de audio integrada (con entrada y salida de audio) en USB, o usar un micrófono USB. De estas alternativas, la más adecuada es la de Bluetooth, dado que parece que el uso más común de la instalación debe permitir que el usuario se mueva por las diferentes habitaciones pudiendo emitir comandos de voz y recibir las respuestas del sistema de reconocimiento.

-Las otras dos alternativas siguen siendo válidas pero restringen el uso de los comandos de reconocimiento a la sala donde está situado el sistema.

En ambos casos fue necesario una adaptación del sistema base para satisfacer las dependencias de bluetooth para audio y de los subsistemas alsa para el sonido.

De esta forma el sistema es capaz de interactuar con la mayoría de los periféricos de audio tanto en bluetooth como en conexión directa en USB.

Para el problema de la arquitectura hardware, fue necesario hacer una distribución genérica que no necesitase ser compilada. Por ello, el sistema en fase de instalación reconoce la arquitectura HW (/proc/cpuinfo) y con ello, instala la versión de dictionary.c correspondiente a la arquitectura.

9.2.3 Problemas con el API de reconocimiento de Google

Uno de los ultimos problemas que tuvimos fue que, dado que el API de Google es abierto, a partir del 8 de mayo dejó de funcionar el API v1, debido a problemas de seguridad. A este API se accedía sin ningun tipo de seguridad.

Afortunadamente, existe otra versión del API que se accede con una clave pública (esta es la API v2). El interfaz de la API v2 es totalmente diferente al interfaz del API V1, y por ello, hubo que reestructurar el código creando las versiones V1 y V2 de varios de los procesos: send_speech, recognize y autodetect.sh.

Aprovechando el cambio de interfaz, se ha visto que el interfaz v2, además, devuelve varias posibilidades de reconocimiento: además de la respuesta más probable, devuelve, como mínimo otras 2 posibilidades salvo que el reconocimiento sea 100% fiable. Con ello, se ha enriquecido el reconocimiento de forma que si la primera respuesta no es un comando válido, se hacen dos reintentos con las dos posibilidades que devuelve el API, aumentando la capacidad de reconocimiento.

10 DOCUMENTACION

La documentación de la aplicación, se ha añadido en el proyecto, creando una wiki, que será accesible desde el proyecto OpenDomo.

Este es el link:

https://github.com/loureiromahia/OpenDomo_VR/wiki

En el Anexo 1 se presenta una copia de dicha documentación.

11 SOPORTE

Por otro lado, dado que ya se ha incluido OpenDomo_VR como subproyecto en la nueva versión de Opendomo (Opendomo2.0), se ha abierto un blog desde OpenDomo para las dudas y los problemas que puedan tener los usuarios.

http://es.opendomo.org/newproject-opendomo-vr#comments

Por ahora, solo se ha publicado una beta de OpenDomo 2.0, por lo que el número de comentarios es muy bajo y están relacionados con la implementación del sonido: La recomendación del proyecto es usar Bluetooth (es una opción que hemos probado y adaptado para la Raspberry Pi, que va a ser el hardware más usual para instalar OpenDomo por el precio y la funcionalidad).

12 LESSONS LEARNT

Básicamente, se han ido comentando en los diferentes apartados de la memoria:

12.1 Los desarrollos deben ser ágiles con pruebas frecuentes

Lo que ha facilitado mucho el desarrollo y la consecución del proyecto es hacer desarrollos con poca funcionalidad entregada en cada delivery, pudiendo probar lo antes posible y con un delta de funcionalidad pequeño. Cuando se han hecho entregas grandes con bastante funcionalidad, ha dado lugar a problemas que han sido más difíciles de detectar y corregir

12.2 Entorno de pruebas lo más real posibles

Se ha visto claramente que algo que parecía tan sencillo como adaptar al entorno de la Raspberry Pi dió lugar a varios problemas que no se hubieran visto si se hubiera probado en un entorno simulado: entrada de audio, distribución pequeña, arquitectura hardware.

12.3 Cuidado con los APIs abiertos

Se ha visto claramente que los APIs abiertos en los que se accede libremente pueden estar o no. En un momento dado, pueden desaparecer. Por ello hay que tener prevista una solución de contingencia.

En nuestro caso, la realidad es que ahora funcionan las dos versiones del API (el API v1 ha vuelto a funcionar) . El sistema está preparado para funcionar con las dos versiones del API.

13 FUTURAS ACTUACIONES

Además de enriquecer el proyecto con nuevos VoiceCommnds y depurar los voiceCommands actuales, hay diferentes posibilidades de enriquecer el proyecto:

- Extender OpenDomo_VR a más idiomas: catalán, euskera y gallego, al menos

- Portar todo el código bash shell a Python. De esta forma, se unificaría y compactaría el código, haciéndolo más eficiente permitiendo fácilmente la portabilidad a diferentes arquitecturas.

14 REFERENCIAS

- Digital Signal Processing of Speech Signals. (Lawrence R.Rabiner, Ronald W.Shafer)

- Computers that Talk and Listen: Man Machine Communication by Voice (J.L.Flanagan). Proc. IEEE Vol.64,N4, pp 405-415. April 1976

- Wikipedia: Speech Recognition

- Wikipedia: Voxforge

15 ANEXO I: DOCUMENTACIÓN

public loureiromahia/OpenDomo_VR

Home
loureiromahia edited this page 2 months ago · 10 commits
Pages (5)

- AVAILABLE VOICECOMMANDS
- Home
- HOW OPENDOMO_VR WORKS
- HOW TO CREATE OR MODIFY A CHARACTER (only for advanced users)
- HOW TO CREATE YOUR OWN VOICECOMMAND

Clone this wiki locally

Welcome to the OpenDomo_VR wiki!
CONTENTS:

1.HOW OPENDOMO_VR WORKS

2. AVAILABLE VOICE COMMANDS

3. HOW TO CREATE YOUR OWN VOICE COMMAND

4.HOW TO CREATE OR MODIFY A CHARACTER (only advanced users)

HOW OPENDOMO_VR WORKS
loureiromahia edited this page 2 months ago · 7 commits
Pages (5)

- AVAILABLE VOICECOMMANDS
- Home
- HOW OPENDOMO_VR WORKS
- HOW TO CREATE OR MODIFY A CHARACTER (only for advanced users)
- HOW TO CREATE YOUR OWN VOICECOMMAND

Clone this wiki locally

HOW OPENDOMO_VR WORKS

1.INTRODUCTION

OpenDomo_VR works creating a set of voiceCommands which main components are:

- A dictionary, which contains the association of the recognized voice pattern with the corresponding commands to execute.

- Commands resulting of the recognition of a voice pattern. These commands are executable files written either in bash shell , in python or in another language. In the actual case, all the commands are written in bash shell.

All this logic is organized in what is called voiceCommands which contains a directory structure in which both the dictionary and the command to execute for this particular voiceCommand is defined. All what you have to do to include this voiceCommand in your distribution is “to install” this voice Command.

Installing a voice Command implies two actions:

- Add the part of the recognized speech pattern to the “main.dic” of the installation. (/usr/local/opendomo/vr/Recognition/modes/main.dic)

- Add the command to be executed into the “bin” subfolder of the OpenDomo_VR folder. (/usr/local/opendomo/vr/Recognition/bin)

2.DICTIONARY STRUCTURE

The dictionary (main.dic) is organized in the following way:

SPEECH PERSON SAID

COMMAND TO RUN

For instance:

what is the time

date

In this case, the recognizer will try to recognize the text “what is the time” and if so, the command executed should be a simple “date” command.

There are special characters to improve the recognition:

[optional] Everything between bracket should be optional. The system will consider as valid the recognition a speech pattern containing the main pattern including the optional pattern or not:

For instance:

what is the time [now] —> In this case the system will recognize as valid:

what is the time

what is the time now

<option1,option2,option3…optionn> In this case the system will look for the existance of one of the options.

For instance:

Selected color <white, black, blue, green> —> In this case, the systen will look for one of the folloeing options:

Selected color white

Selected color black

Selected color blue

Selected color green

This option is specially useful for recognizing capitals. In the previous case the first letter “S” maybe that is recognized as “S” or as “s”. In this case, the best pattern is:

<S,s>elected color <white,black,blue,green>

3.HOW TO PASS PARAMETERS

There are mainly 3 ways to pass parameters:

LINE

In this case the structure of the dictionary entry is:

Command (LINE Object)

Executable “$Object$”

LINE means any string of text following the “Command” pattern.

For instance:

Open (Line Object)

Will recognize Open the door ,, Open the window ,, Open the bathroom door , and tte commands that will execute are:

Executable “the door”

Executable “the window”

Executable “the bathroom door”

WORD

In this case the structure of the dictionary entry is:

Command (WORD Object)

Executable “$Object$”

WORD means any single word of text following the “Command” pattern.

For instance:

Open (Line Object)

Will recognize Open door ,, Open window ,, Open bathroom door , and tte commands that will execute are:

Executable “door”

Executable “window”

_Executable “bathroom door” _==> IN THIS CASE THE TEXT WILL NOT BE RECOGNIZED , as the pattern is not fullfilled “Open followed by a single word”. In this case there are 2 words.

SPEECH

In this case the structure of the dictionary entry is:

Command [whatever more]

Executable “$SPEECH$” SPEECH means that all the voice pattern recognized will be passed to the Executable as string parameter.

In this case, the system will recognize any speech pattern like “Command” + “any stream of speech”. For instance: “Command start”, “Command Stop”, “Command Open the window”…

In these cases, what will be executed is :

Executable “Command start”

Executable “Command stop”

Executable “Command open the window”

AVAILABLE VOICECOMMANDS

loureiromahia edited this page 2 months ago · 4 commits

Pages (5)

AVAILABLE VOICECOMMANDS

Home

HOW OPENDOMO_VR WORKS

HOW TO CREATE OR MODIFY A CHARACTER (only for advanced users)

HOW TO CREATE YOUR OWN VOICECOMMAND

Clone this wiki locally

1. INTRODUCTION

In this section i am going to show the list of voicecommands available. This list is the core of the implementation and it is dynamic and can grow when we decide to include in this core new commands.

On the other hand, you can create voiceCommands by your own, using the procedure described in the next section of the wiki.

2. DICTIONARIES AND OPERATIONAL MODES

OpenDomo_VR has two MODEs of operation:

Identification: In this mode OpenDomo_VR only accepts the voice command “Hello XXXX”, where XXXX is the name of the installation. Depending on the language, Opendomo recognizes:

“Hola OpenDomo”

“Hello OpenDomo”

The description of this mode is in the identification.dic

Normal: After a correct Identification, OpenDomo_VR switches to MODE =Normal. In this mode OpenDomo recognizes all the defined voice commands for this installation (switch on/off lights, music control…). OpenDomo_VR keeps in this mode until a “Bye” Voice Command is issued. If this Bye is issued , OpenDomo_VR switch again to Identification MODE, not accepting a new command until a correct identification is issued.

The description of this mode is in the main.dic

Note that all these status are Operational OpenDomo_VR Modes. When the user is not in the

installation, or during the night(while he is sleeping) OpenDomo_VR daemon is not running.

3. MAIN OPERATIONAL VOICECOMMANDS

In this chapter i will describe the contents of actual main.dic, with all the voiceCommands that are recognized and the actions.

VOICECOMMAND MENU

This voiceCommand allows to the user to perform all the posible actions that he/she can do in the opendomo web menu, using simple commands to select one option.

To prepare all the posible entries, OpenDomo_VR looks at the actual data of the installation, checking the switches, cameras, music, …ports in general, in the installation.

OpenDomo_VR will navigate and tell the user the options in the menu, allowing the user choose a submenu or a menu entry, acting as he/she presses an option in the web. For instance , when the user starts the menu mode, saying menu, OpenDomo_VR responds:

“Menu principal. Para acceso a menu de control, diga 1. Para acceso a menu herramientas, diga 2. Para acceso a menu configuracion, diga 3. Para salir del modo menu, diga Salir. Durante todo el menu puede usar el prefijo opcion para aumentar la capacidad de reconocimiento. Es lo mismo decir opcion 2 que decir 2” .

(this is the main menu in spanish).

So, the user can say:

[opcion] <1,2,3>

salir

If he says opcion 1, OpenDomo_VR will respond with the contents of the Control Menu , allowing to choose one of the options on this menu.

And so on…

This is the content of the main dic:

```
#PLUGIN: menu
menú
menu 0
[opción] <1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19>
menu “$SPEECH$”
<activar,desactivar> <1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19>
```

menu “$SPEECH$”

valor (LINE Objeto)

menu “$Objeto$”

salvar <1,2,3,4>

menu “$SPEECH$”

voz (WORD Objeto)

menu “$SPEECH$”

volver

menu 20

salir

menu 21

#END

VOICECOMMAND AYUDA

This voiceCommand shows to the user all the possible actions that he/she can do in the opendomo installation issuing direct commands (not via menu).

To prepare all the posible entries, OpenDomo_VR looks at the actual data of the installation, checking the switches, cameras, music, …ports in general, in the installation.

This is what an example of what the system says in an specific installation when the user says “Ayuda”

Encender luz cocina

Apagar luz cocina

Encender luz cocina

Apagar luz cocina

Encender luz comedor

Apagar luz comedor

Encender luz comedor

Apagar luz comedor

Encender clima comedor

Apagar clima comedor

Encender clima comedor

Apagar clima comedor

Informar de ai01(temperatura)

Informar de ai02 (detector apertura)

Informar de ai01(sensor apertura puertas)

Informar de ai02 (temperatura comedor)

Ajustar termostato comedor valor, grados

Ajustar luz comedor valor, porcentaje

Adios OpenDomo

This is the content of the main.dic:

#PLUGIN: Ayuda

ayuda

Ayuda

#END

VOICECOMMAND VARCLIMATE

This voiceCommand allows to the user change the temperature of a room, adjusting the thermostat to that specific value in ºC.

To issue this command the user has to say:

“ajustar termostato cocina 23 [grados]”

“cambiar termostato cocina 23 [grados]”

“termostato cocina 23 [grados]”

The system will look if there is a thermostat in “cocina”, and if so, will fix the temperature to the value after(23). Note that OpenDomo_VR, will recognize in the following way:

ajustar termostato “string”

And then in the “string” that follows will look for two parameters(minimum):

– string first word. This is the name of the room (in the example “cocina”)

– string second word. This is the value in ºC (in the example “23”)

– Whatever is after these two words is optional, but normally the user will say “grados”

This is the content of the main.dic:

#PLUGIN: VarClimate

<ajustar,cambiar> termostato (LINE Objeto)

VarClimate “$Objeto$”

termostato (LINE Objeto)

VarClimate “$Objeto$”

#END

VOICECOMMAND VARLIGHT

This voiceCommand allows to the user change the level of light of a room, adjusting the variable light regulator to that specific value in %.

To issue this command the user has to say:

“ajustar luz cocina 70 [por ciento]”

“cambiar luz cocina 70 [por ciento]”

“luz cocina 70 [por ciento]”

The system will look if there is a variable light regulator in “cocina”, and if so, will fix it to the value after(70). Note that OpenDomo_VR, will recoginize in the following way:

ajustar luz “string”

And then in the “string” that follows will look for two parameters(minimum):

– string first word. This is the name of the room (in the example “cocina”)

– string second word. This is the value in % (in the example “70”)

– Whatever is after these two words is optional, but normally the user will say “por ciento”

This is the content of the main.dic:

#PLUGIN: VarLight

<ajustar,cambiar> luz (LINE Objeto)

VarLight “$Objeto$”

luz (LINE Objeto)

VarLight “$Objeto$”

#END

VOICECOMMAND MUSICOFF

This voiceCommand allows to the user deactivate the ambiance music in a specific room

To issue this command the user has to say:

“desactivar musica cocina "

“apagar musica cocina "

The system will look if there is a music switch in “cocina”, and if so, will deactivate the music in that room. Note that OpenDomo_VR, will recoginize in the following way:

desactivar musica “string”

And then in the “string” that follows will look for these parameters:

– string first word. This is the name of the room (in the example “cocina”)

– Whatever is after these two words is optional: for instance, the user can say “desactivar musica cocina, por favor”

This is the content of the main.dic:

```
#PLUGIN: MusicOFF
desactivar música (LINE Objeto)
MusicOFF “$Objeto$”
apagar música (LINE Objeto)
MusicOFF “$Objeto$”

#END
```

VOICECOMMAND MUSICON

This voiceCommand allows to the user activate the ambiance music in a specific room

To issue this command the user has to say:

“activar musica cocina "

“encender musica cocina "

The system will look if there is a music switch in “cocina”, and if so, will activate the music in that room. Note that OpenDomo_VR, will recoginize in the following way:

activar musica “string”

And then in the “string” that follows will look for these parameters:

– string first word. This is the name of the room (in the example “cocina”)

– Whatever is after these two words is optional: for instance, the user can say “activar musica cocina, por favor”

This is the content of the main.dic:

```
#PLUGIN: MusicON
activar música (LINE Objeto)
MusicON “$Objeto$”
encender música (LINE Objeto)
MusicON “$Objeto$”

#END
```

VOICECOMMAND VIDEOOFF

This voiceCommand allows to the user deactivate the video camera in a specific room

To issue this command the user has to say:

“desactivar video cocina "

“apagar video cocina "

The system will look if there is a video camera in “cocina”, and if so, will deactivate it in that room. Note that OpenDomo_VR, will recoginize in the following way:

desactivar video “string”

And then in the “string” that follows will look for these parameters:

– string first word. This is the name of the room (in the example “cocina”)

– Whatever is after these two words is optional: for instance, the user can say “desactivar video cocina, por favor”

This is the content of the main.dic:

#PLUGIN: VideoOFF

desactivar video (LINE Objeto)

VideoOFF “$Objeto$”

apagar video (LINE Objeto)

VideoOFF “$Objeto$”

#END

VOICECOMMAND VIDEOON

This voiceCommand allows to the user deactivate the video camera in a specific room

To issue this command the user has to say:

“activar video cocina "

“encender video cocina "

The system will look if there is a video camera in “cocina”, and if so, will activate it in that room. Note that OpenDomo_VR, will recoginize in the following way:

activar video “string”

And then in the “string” that follows will look for these parameters:

– string first word. This is the name of the room (in the example “cocina”)

– Whatever is after these two words is optional: for instance, the user can say “activar video cocina, por favor”

This is the content of the main.dic:

#PLUGIN: VideoON

activar video (LINE Objeto)

VideoON “$Objeto$”

encender video (LINE Objeto)

VideoON “$Objeto$”

#END

VOICECOMMAND SENSORS

This voiceCommand allows to show the status of a sensor in a specific room

To issue this command the user has to say:

“informar de temperatura cocina "

“estado de temperatura cocina "

The system will look for a temperature sensor in “cocina”, and if so, will provide the user with a voice response indicating the actual value of this sensor. Note that OpenDomo_VR, will recoginize in the following way:

informar de “string”

And then in the “string” that follows will look for these parameters:

string, first word: will identify a sensor “desc field”.

string, second word: will identify the room or the zone.

For instance “informar de temperatura cocina”

In this case there will be a zone called “cocina” with a sensor called “temperatura”.

The response contains the actual value of the sensor, and the units (unit field is included in the port information). For instance, in this case the units are “grados”. A posible answer will be :

“Valor actual 23 grados”

This is the content of the main.dic:

```
#PLUGIN: Sensors
informar de (LINE Objeto)
Sensors “$Objeto$”
estado de (LINE Objeto)
Sensors “$Objeto$”
#END
```

VOICECOMMAND CLIMAOFF

This voiceCommand allows to the user deactivate the climate in a specific room

To issue this command the user has to say:

“desactivar clima cocina "

“apagar clima cocina "

The system will look if there is a climate switch in “cocina”, and if so, will deactivate it in that room. Note that OpenDomo_VR, will recognize in the following way:

desactivar clima “string”

And then in the “string” that follows will look for these parameters:

– string first word. This is the name of the room (in the example “cocina”)

– Whatever is after these two words is optional: for instance, the user can say “desactivar clima cocina, por favor”

This is the content of the main.dic:

```
#PLUGIN: ClimaOFF
apagar clima (LINE Objeto)
ClimaOFF “$Objeto$”
desactivar clima (LINE Objeto)
ClimaOFF “$Objeto$”
#END
```

VOICECOMMAND CLIMAON

This voiceCommand allows to the user to activate the climate in a specific room

To issue this command the user has to say:

“activar clima cocina "

“encender clima cocina "

The system will look if there is a climate switch in “cocina”, and if so, will activate it in that room. Note that OpenDomo_VR, will recognize in the following way:

activar clima “string”

And then in the “string” that follows will look for these parameters:

– string first word. This is the name of the room (in the example “cocina”)

– Whatever is after these two words is optional: for instance, the user can say “activar clima cocina, por favor”

This is the content of the main.dic:

#PLUGIN: ClimaON

encender clima (LINE Objeto)

ClimaON “$Objeto$”

activar clima (LINE Objeto)

ClimaON “$Objeto$”

#END

VOICECOMMAND LUCESOFF

This voiceCommand allows to the user siwtchoff lights in a specific room

To issue this command the user has to say:

“apagar luz cocina "

“apagar luces cocina "

The system will look if there is a light switch in “cocina”, and if so, will switchoff light in that room. Note that OpenDomo_VR, will recognize in the following way:

apagar luz “string”

And then in the “string” that follows will look for these parameters:

– string first word. This is the name of the room (in the example “cocina”)

– Whatever is after these two words is optional: for instance, the user can say “apagar luz cocina, por favor”

This is the content of the main.dic:

#PLUGIN: LucesOFF

apagar luz (LINE Objeto)

LucesOFF “$Objeto$”

apagar luces (LINE Objeto)

LucesOFF “$Objeto$”

#END

VOICECOMMAND LUCESON

This voiceCommand allows to the user siwtchon lights in a specific room

To issue this command the user has to say:

“encender luz cocina "

“encender luces cocina "

The system will look if there is a light switch in “cocina”, and if so, will switchon light in that room. Note that OpenDomo_VR, will recognize in the following way:

encender luz “string”

And then in the “string” that follows will look for these parameters:

– string first word. This is the name of the room (in the example “cocina”)

– Whatever is after these two words is optional: for instance, the user can say “encender luz cocina, por favor”

This is the content of the main.dic:

#PLUGIN: LucesON

encender luz (LINE Objeto)

LucesON “$Objeto$”

encender luces (LINE Objeto)

LucesON “$Objeto$”

#END

VOICECOMMAND OPENDOMO_STOP

This voiceCommand switches the mode from NORMAL to IDENTIFICATION

Note that in IDENTIFICATION mode the system will not accept any other command except “Hola OpenDomo” (identification command).

To issue this command the user has to say:

“adios OpenDomo "

This is the content of the main.dic:

```
#PLUGIN: OpenDomo_stop
<A,a>diós OpenDomo
OpenDomo_stop
#END
```

VOICECOMMAND OPENDOMO_START

This voiceCommand switches the mode from IDENTIFICATION to NORMAL

To issue this command the user has to say:

“Hola OpenDomo "

This is the content of the main.dic:

```
#PLUGIN: OpenDomo_start
Hola OpenDomo
OpenDomo_start
#END
```

HOW TO CREATE YOUR OWN VOICECOMMAND

loureiromahia edited this page 2 months ago · 2 commits

Pages (5)

- AVAILABLE VOICECOMMANDS
- Home
- HOW OPENDOMO_VR WORKS
- HOW TO CREATE OR MODIFY A CHARACTER (only for advanced users)
- HOW TO CREATE YOUR OWN VOICECOMMAND

Clone this wiki locally

Creating voiceComamnds in OpenDomo_VR is rather easy. First go into your vr directory.

In my case:

manu@debian:/usr/local/opendomo/vr/characters/es/antonio$ cd /usr/local/opendomo/vr

manu@debian:/usr/local/opendomo/vr$

You can now run the sdk command. For this tutorial I will create a voiceCommand to display the date

manu@debian:/usr/local/opendomo/vr$ ./sdk -c dateDisplay

Creating new plugin

Created plugin in /usr/local/opendomo/voiceCommands/es/dateDisplay/

manu@debian:/usr/local/opendomo/vr$

Inside the folder /usr/local/opendomo/voiceCommands/es/dateDisplay/ there are

three sub folders and a plugin.info file

The three folders: bin, config, and modes will store executables, configuration files and voice commands respectively.

Inside the bin folder is a file named the same as the voiceCommand, in my case dateDisplay.

This is the file that will be executed when your command is said.

Inside the config folder is the file “settings.conf”. More config files may be added.

This file is used to store any settings your voiceCommand needs.

A settings.conf is not required and can be removed later.

Inside the modes folder is the file “main.dic”.

This will store all the commands your program needs.

A program can have more than one dictionary but main.dic is required.

My program will have the command “what is the date”. I have changed the contents of main.dic to:

<w,W>hat is the date

dateDisplay

When the user says “what is the date” the program will execute dateDisplay.

I have changed the contents of dateDisplay in the bin folder to the following:

```
#!/bin/bash
i=$(date)
echo $i >> Microphone/result
./pycmd result
```

This will get the date and write it to the result file that will be displayed after the program is done executing.

I can now go back into the vr folder again and run:

```
manu@debian:/usr/local/opendomo/vr$ ./sdk -p dateDisplay
```

A file will appear, this is the plugin.info file. I have changed mine to the following:

```
#Plugin.info automatically generated
name = dateDisplay
version = 1.0 #Used for updating
file = dateDisplay
configs = #I have left this blank because my program never used settings.conf
author = PEPE
description = Display the current date
actions =
what is the date, display the current date #voice command, description. Each command on it’s own line
```

After you save the file ./sdk displays:

```
Packaged to /usr/local/opendomo/voiceCommands/es/dateDisplay/dateDisplay.sp
```

And finally to install it:

```
./plugins -i /usr/local/opendomo/voiceCommands/es/dateDisplay/dateDisplay.sp
```

Confirm the installation and your voiceCommand is now installed.

TO EDIT A voiceCommand:

You need to follow the steps:

1. Edit the main.dic file or executable file in /usr/local/opendomo/voiceCommands/es/dateDisplay folder

2. In /usr/local/opendomo/vr filder execute: I

./sdk -p dateDisplay

3. In /usr/local/opendomo/vr filder execute:

./plugin -r dateDisplay

4. In /usr/local/opendomo/vr filder execute:

./plugins -i /usr/local/opendomo/voiceCommands/es/dateDisplay/dateDisplay.sp

Note that there is no option to modify a voiceCommand, and so you have to remove it (plugins -r) and reinstall it (./plugins -i)

HOW TO CREATE OR MODIFY A CHARACTER (only for advanced users)

loureiromahia edited this page 2 months ago · 1 commit

Pages (5)

Clone this wiki locally

1.INTRODUCTION

The characters are organized in the following way:

- The main folder is : /usr/local/opendomo/vr/character. There is a folder per language and inside this folder there is a subfolder per character.

manu@debian:/usr/local/opendomo/vr/characters$ ls

en es

manu@debian:/usr/local/opendomo/vr/characters$ cd es

manu@debian:/usr/local/opendomo/vr/characters/es$ ls

antonio marta

In this case, there are two characters defined for es langualge: antonio and marta

- Inside the character folder there are two files :

espeak.dat ==> This file contains the data for the espeak command (pitch, speed, male/female…)

frases ==> This file contains the speech that OpenDomo_VR synthetizes on a particular event

2. CREATING A NEW CHARACTER

To create a new character the best way is to use one of the existing characters and copy the folder of the character to a new folder.

In the example:

manu@debian:/usr/local/opendomo/vr/characters/es$ cp -Rp antonio pedro

manu@debian:/usr/local/opendomo/vr/characters/es$ ls

antonio marta pedro

We have created a new character called “pedro”. Now we have to customize it editing the two files (espeak.dat and frases)

3. CUSTOMIZING VOICE CHARACTERISTICS (espeak.dat)

To perform this operation, it is hardly recommended to read espeak documentation. There you can find how to modify the pitch and the speed of the voice. The main parameters to be used are:

-v specifies the language and the gender (-ves spanish male, -ves+f3 spanish female, -ven english…)

-p pitch of the voice

-s speed of the voice

Actual characters have a mimimum configuration (only basic features of espeak: gender and language, but you can customize playing with espeak).

At the end you have to put in espeak.dat the customized data for espeak.

For instance:

manu@debian:/usr/local/opendomo/vr/characters/es/pedro$ echo “espeak -ves -s 50 -p 90” > espeak.dat

This will create a female voice in spanish, speaking slowly…

There is also the posibility of taking a specific voice as reference.

Anyhow, please, use the manual pages and google to configure the voice as you prefer.

3. CUSTOMIZING THE STYLE OF THE PHRASES (frases)

The frases file is organized in the follwing way:

Identification Field: Content of the phrase

For instance:

manu@debian:/usr/local/opendomo/vr/characters/es/pedro$ more frases

SiEntendido: entendido

NoEntendido: No he entendido, ¿puede repetir?

NoDef: No esta definida la zona

ClimaOFFswoff: Apagando Clima en

ClimaOFFalready: Ya esta apagado el clima en

ClimaONswon: Encendiendo Clima en

ClimaONalready: Ya esta encendido el clima en

LucesOFFswoff: Apagando luces de

LucesOFFalready: Ya esta apagada la luz en

LucesONswon: Encendiendo luces de

LucesONalready: Ya esta encendida la luz en

MusicOFFswoff: Desactivando musica en

MusicOFFalready: Ya esta apagada la musica en

MusicONswon: Activando musica en

MusicONalready: Ya esta activada la musica en

StartBienvenido: Bienvenido a OpenDomo. Comandos posibles

……..

So, to change a phrase for a specific event, you have to modify CAREFULLY ONLY THE FIELD AFTER THE “:”

Example: If you want to modify the phrase that the system says when you try to switch on the light in a room and the light is already switched, you have to go to the line:

LucesONalready: Ya esta encendida la luz en

And modify the text after the “:” (only that!!).

So, you can decide to say “Ya tienes encendida la luz en” instead of “Ya esta encendida la luz en”. In this case the line has to be like this:

LucesONalready: Ya tienes encendida la luz en

DO THIS CAREFULLY TAKING CARE NOT TO MODIFY THE TEXT BEFORE “:” as this text is used by OpenDomo_VR to identify the phrases.

4. CONFIGURE THE NEW CHARACTER TO BE USED BY YOUR INSTALLATION

Finally, after you create the character (creating the folder), modify the voice characteristics (espeak.dat) and the phrases (frases), last thing you have to do is to activate the new character to be used in your installation.

This process is done running ./configureVoiceSystem.sh your_character

To run this command you have to be in : /usr/local/opendomo/services/config.

In the example:

manu@debian:/usr/local/opendomo/services/config$ ./configureVoiceSystem.sh pedro

This process, among some other actions, copies the files frases and espeak.dat to /etc/opendomo/speech. These files of this folder are the noes that define the character to be used in the installation.