Grado en Ingeniería Electrónica Industrial

Escuela Politécnica Superior de Jaén

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior

Trabajo Fin de Grado

SISTEMA PARA INTERACCIÓN POR VOZ HOMBRE-ÁRBOL

Alumno: Javier Ureña Santiago

Tutor: Diego Manuel Martínez Gila

Elisabet Estevez Estevez

Dpto: Departamento de Automática

Septiembre, 2018

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Automática

DIEGO MANUEL MARTINEZ GILA y ELISABET ESTEVEZ ESTEVEZ tutores del Proyecto Fin de Grado titulado: SISTEMA PARA LA INTERACCIÓN POR VOZ HOMBRE - ÁRBOL, que presenta JAVIER UREÑA SANTIAGO, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, SEPTIEMBRE de 2018

El alumno:

JAVIER UREÑA SANTIAGO

Los tutores:

DIEGO MANUEL MARTINEZ GILA
ELISABET ESTEVEZ ESTEVEZ

Resumen

Se presenta un olivo monitorizado a tiempo real por diversos sensores. Estos sensores recogen diversos datos pertenecientes a propiedades que rodean al olivo. Datos como la temperatura ambiente, la humedad del suelo o la humedad ambiente son enviados a la red, y transmitidos al *Cloud* o Nube, donde se almacenan.

El sistema de reconocimiento de voz a desarrollar se encarga de analizar palabras clave como temperatura, o humedad, en una frase dicha por algún locutor cercano. Una vez reconocida la palabra clave, el software recibe el dato pertinente y devuelve una respuesta de forma auditiva, haciendo uso de un árbol de respuestas que permite dar una sensación más humanizada de lo que sería una conversación con el olivo.

El proyecto también ha de tener en cuenta la autonomía completa del modelo, ya que al estar a la intemperie no dispone de una alimentación de red, por lo que se implementará junto a un sistema fotovoltaico autónomo, que, aparte de darle la autonomía necesaria para independizarse de la corriente de red, añade una perspectiva ecológica.

Índice

Índice de ilustraciones:

Índice de tablas:

Índice de gráficas:

1. INTRODUCCION

El reconocimiento de voz es una tecnología que lleva existiendo desde hace decenas de años. Su primera aparición fue en 1952, donde *Bell Labs* creó a *Audrey*, una máquina que era capaz de entender dígitos hablados [12]. A partir de ahí, la tecnología fue avanzando a pasos agigantados, creando máquinas capaces de reconocer más de 20,000 palabras, incorporando la capacidad de predecir palabras, hasta que en 1990 *Dragon* lanzó el *Dragon Dictate*, el primer producto de reconocimiento de voz para consumidores [12]. Desde entonces, empresas como Microsoft, Apple e IBM han estado invirtiendo en esta tecnología hasta lo que se puede disfrutar ahora: un dispositivo en el bolsillo capaz de configurar una alarma sin tener que tocar ningún botón.

Dejando a un lado lo estrafalario que puede llegar a ser esta tecnología, también posee una utilidad especial para aquellas personas con hándicaps físicos o psicológicos, que les ayuda a normalizar un poco más sus vidas, ya sea ayudar a escribir a una persona paralizada, o a hablar a una persona muda (Stephen Hawking hacia uso de un software con Speech-To-Text para poder decir palabras y frases). Y no solo habilita la capacidad de comunicación entre personas, ya que con el reconocimiento de voz se pueden dar órdenes a un ordenador, ayudando de esa manera a hacer uso de casi todas sus funcionalidades. En la Universidad Miguel Hernández de Elche, dos estudiantes de Ingeniería Electrónica Industrial consiguieron desarrollar un software capaz de realizar dibujos haciendo uso de esta tecnología [4].

A lo largo de este capítulo se hablará de la tecnología referente al reconocimiento de voz, y todo lo que le rodea, su impacto en la sociedad y como ha sido capaz de mezclarse en el ámbito agrícola.

Se habla también de la motivación y contexto del proyecto, en que ámbito se ha desarrollado, y de los objetivos que se consiguen alcanzar.

1.1. Reconocimiento de voz: Bases

Si minimizamos el reconocimiento de voz a lo más simple, lo que se puede comprobar es el uso de un motor/software de tipo *Speech-To-Text* (STT, de habla a escritura). Cuando se habla, se provocan ondas que alteran el aire. Estas vibraciones pasan por un conversor Analógico-Digital que traduce la onda en un lenguaje que la computadora puede entender. Por tanto, esta señal se digitaliza y se filtra para eliminar ruido, se normaliza para tenerla entera al mismo volumen o se regula de velocidad para poder compararla correctamente con los modelos guardados en la memoria del dispositivo. La muestra una vez está clara, se divide en muestras aún más pequeñas, del tamaño de centenas de segundo, y compara estos segmentos con los fonemas, que son la unidad más pequeña del lenguaje [5].

Una vez empieza a comparar los distintos fonemas, hace uso de sistemas de modelado estadístico que, a través de funciones de probabilidad es capaz de determinar el resultado más acertado. Esto es necesario ya que los acentos y dialectos, y el hecho de que por ejemplo en España hay 88,000 palabras en la RAE, hacen que la síntesis se complique, sobre todo si estamos sintetizando frases largas.

Por suerte se puede contar con diversos motores STT facilitados para su inclusión en numerosos programas. De los más eficaces se pueden considerar el motor de Google, usado por ejemplo en Cortana, que hace uso de una red neuronal para poder predecir la palabra en base al contexto de la frase.

Esto solo visualizaría el apartado técnico en cuanto al reconocimiento de la voz, pero en este proyecto también se baraja la devolución de datos a través de la voz. A este motor se le llama *Text-To-Speech* (TTS, de texto a habla). Es el encargado de devolver una respuesta auditiva en base a un texto, similar al que usaba por ejemplo Stephen Hawking.

A fin de reproducir el sonido natural de cada lenguaje, un narrador se encarga de grabar una serie de textos donde se encuentran todos los posibles sonidos en un determinado idioma. Estas grabaciones son troceadas y organizadas en una base

de datos, donde se organizarán en difonos, sílabas, morfemas, palabras, frases e incluso oraciones [1].

El sistema TTS hace uso de un sofisticado análisis lingüístico del texto a decir, con la intención de transponer el texto escrito en un texto formado por los fonemas que lo componen, y tras esto, un análisis gramatical y sintáctico se realiza para definir la pronunciación de cada palabra para poder reconstruir asi la oración. A este proceso se le llama *prosodia* y es el que le da el ritmo y entonación a la oración.

Finalmente, el sistema produce información asociando el texto fonético con el tono y la longitud requerida para la correcta pronunciación. Se finaliza el análisis, y se genera un sonido seleccionando las mejores unidades acústicas de la base de datos.

1.2. Reconocimiento de voz en el ámbito agrónomo

Como ya se ha mencionado previamente, esta tecnología se encuentra cada vez más presente en nuestras vidas, en el ámbito privado. Es un hecho que el Reconocimiento de Voz anula varios problemas de una manera cómoda. Por supuesto, plantear esta tecnología en el mundo laboral no es una cuestión. Es el caso de *Montrue Technologies*, una empresa de Oregón, EEUU, que haciendo uso de unos SDK (kits de desarrollo de software) ha desarrollado una aplicación en IOS que ayuda a los facultativos médicos a dictar notas [8].

Siendo este caso un olivo donde se va a implementar el sistema de reconocimiento de voz, no es descabellado analizar cómo ha afectado esta tecnología en el ámbito agrónomo y, sorprendentemente, la cantidad de proyectos establecidos con esta premisa es bastante reducida.

Estudios han demostrado que hay una “división digital” substancial que impide a civiles, particularmente a aquellos poco familiarizados con el uso de TICs, en el ámbito rural, de usar la tecnología emergente. La implementación de un sistema de reconocimiento de voz con propósito agrónomo ayudaría a acelerar esta “transferencia de tecnología”, permitiendo una manera fácil y natural de usar un método de comunicación entre el agricultor y la máquina. Esta es la principal

motivación del *Marathi Interactive Voice Response System* [13], un sistema interactivo de reconocimiento de voz de lengua maratí pensado para la realización de consultas relacionadas con los cultivos para los granjeros indios.

Este sistema ha sido implementado en India, donde el 70% de la población forma parte de la industria agrónoma, y esta tecnología ha supuesto un rol crítico. Se ha observado que los mercados están altamente sesgados contra los productores con prácticas muy restrictivas y opacidad debido a su falta de vinculación con estos y la total falta de acceso a información (el 58.7% de la India rural es indocta). Tiene como objetivo, por tanto, construir una agricultura impulsada por la tecnología y el intercambio de información accesible por la agricultura rural.

Este es un buen ejemplo de lo que se puede conseguir con esta tecnología. Este proyecto, aunque a menor escala, también podría ayudar a algunos agricultores, tecnológicamente ajenos, a desarrollar un cultivo más eficiente.

1.3. Contexto

ISR (Integración Sensorial y Robótica [6]) es una empresa que nació como *spin-off* de la Universidad de Jaén. Su contexto se basa en el Grupo Universitario de Investigación en Robótica y Automática de la Universidad de Jaén. Opta de una amplia experiencia en el desarrollo de soluciones tecnológicas y productos basados en la integración sensorial, automática avanzada y visión por computador, con el objetivo de industrializar y explotar comercialmente soluciones en el campo de la automática y el control. Los sectores en los que trabaja son en control de calidad, sector automovilístico, sector logístico y, finalmente, en el sector agrícola, donde ISR desarrolla nuevas técnicas avanzadas de modelado y control.

Este proyecto surgió como una idea esporádica del CEO de la empresa ISR el cual imaginó la posibilidad de mantener una conversación casual con un olivo, y que este le respondiese acorde a sus propiedades físicas.

El proyecto fue desarrollado durante el segundo cuatrimestre del curso 2017/2018, en el GRAV o Grupo de Robótica, Automatización y Visión por Computador de la Universidad de Jaén, el cual acoge tanto alumnos como

profesores e ingenieros para colaborar con ISR en el desarrollo de soluciones en los distintos ámbitos de investigación. Participar en este grupo supone una ayuda académica para aquellos alumnos que deseen realizar un Trabajo de Fin de Grado con una mención en automática, o simplemente en la participación por mera ambición en los diversos proyectos propuestos. Durante este periodo de tiempo, gracias a la colaboración entre compañeros y el material disponible en el laboratorio, este proyecto se ha podido realizar de manera exitosa.

1.4. Motivación

El acercamiento del Reconocimiento de Voz en un ámbito diferente a lo que se suele estar acostumbrado puede captar la atención de la gente, ya no solo por el apartado técnico que conlleva, sino también porque supondría un proyecto muy mediático. Esto significa un acercamiento e interés de la gente en uno de los pilares de la economía jiennense: el olivo.

Los más de 66 millones de olivos plantados en Jaén traen consigo numerosos puestos de trabajo, y una alta reputación internacionalmente en cuanto a su aceite, convirtiendo a Jaén en la sede del Mercado de Futuros del Aceite de Oliva [15], un mercado oficial supervisado por la Comisión Nacional del Mercado de Valores en el que se negocian los contratos de futuros sobre el aceite de oliva. Reitero así la importancia de este árbol en la provincia de Jaén y uno de los motivos de este proyecto.

El proyecto ayudará a transmitir el interés por el olivo de una manera amigable que cualquiera pueda hacer uso. Su intención es presentarlo en museos, o lugares de interés general, donde cualquiera pueda preguntar información al árbol acercando a la gente en cierta manera al ámbito olivar.

También ayuda a acercar más esta tecnología y favorece la investigación de su uso en posteriores proyectos relacionados con la automática.

1.5. Objetivos

1.5.1. General

El objetivo general consiste en crear un sistema interactivo de pregunta y respuesta auditiva que, dependiendo de una serie de algoritmos relacionados con la toma de decisiones basadas en reglas programadas, sea capaz de mantener cierto nivel coherente de conversación.

1.5.2. ESPECIFICOS

- Investigar la tecnología del reconocimiento de voz en el ámbito agrónomo
- Colaborar con ambientes de GNU/Linux en la Raspberry Pi
- Visualizar las diversas opciones referentes al reconocimiento de voz e implementar la opción más favorable para el proyecto
- Manejar y controlar ThingSpeak y saber incorporarlo en el proyecto
- Desarrollar la programación pertinente para la adquisición de datos y muestreo
- Dimensionar y autonomizar el proyecto para que sea capaz de funcionar desde un sistema fotovoltaico
- Construir un prototipo con la intención de mostrar una versión Demo de cómo funcionaría

2. METODOLOGIA

Hay diversos aspectos a considerar a la hora de elaborar este proyecto. A parte del apartado Software relacionado con el reconocimiento de voz, también se ha considerado pertinente la creación de un prototipo con la intención de implementarlo físicamente y demostrar su funcionamiento. Es por ello, que hay que analizar también el apartado Hardware donde se mencionarán todos los elementos utilizados para su fabricación.

En este capítulo se describirá, por tanto, no solamente el software sino los componentes y su montaje físico. También se hablará del proceso de obtención de datos, abarcando desde el envío de datos a la Nube, hasta su recolección y emisión de forma auditiva.

También se hablará de cómo se ha optimizado el sistema de manera que tenga un consumo mínimo y pueda abastecerse correcta y autónomamente, con la mera ayuda de la energía solar.

2.1. Descripción del setup

En este apartado se procede a explicar el material usado para la creación del proyecto, y se describen los componentes y software usados, tanto los fallidos como los exitosos.

2.1.1. Hardware

A la hora de montar el proyecto, hay que tener varias cosas a tener en cuenta. Como se ha mencionado previamente, el proyecto ha de ser autónomo, por lo que dependerá de energía solar; Ha de estar a la intemperie, por lo que deberá guardarse correctamente; Y ha de devolver una respuesta auditiva con una potencia necesaria como para ser escuchada a ciertos metros de distancia.

Esquemático del circuito propuesto:

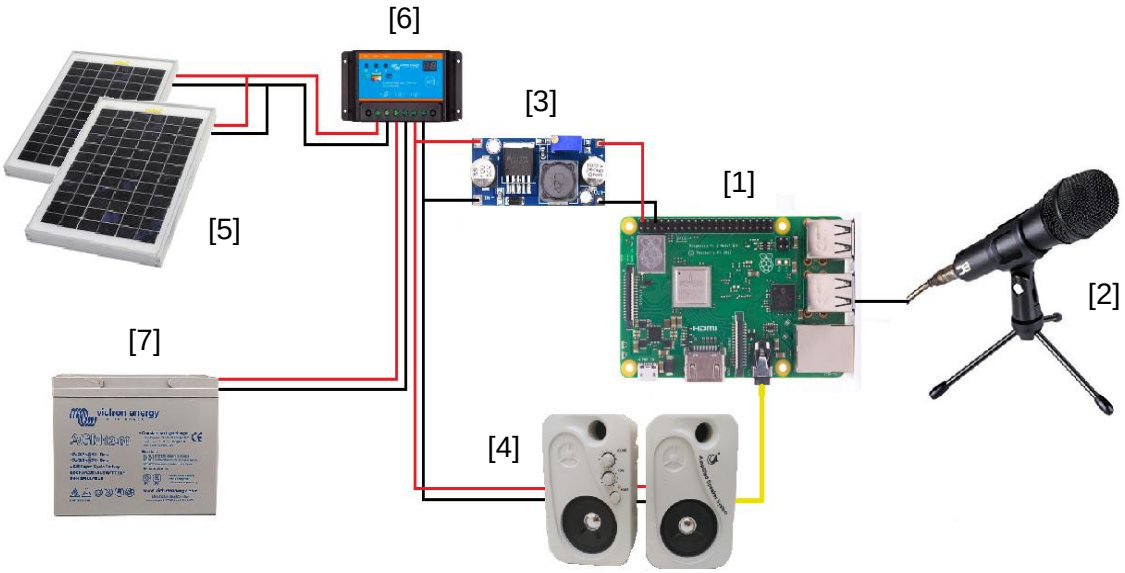


Ilustración 2.1 - Esquemático del montaje del proyecto

Teniendo en cuenta todo esto, se enumeran en lista los siguientes componentes:

[1] Computadora:

Podría decirse que es el cerebro. La Raspberry Pi es básicamente una computadora reducida de bajo coste que permite hacer uso de software libre para pequeños proyectos de enseñanza o personales. Su sistema operativo principalmente es Raspbian (procedente de GNU/Linux) una distribución derivada de Debian optimizada para su Hardware.

Ilustración 2.2 - Raspberry Pi 3 Model B

Posee un procesador renovado Quad-Core Broadcom de 64 bit y 1.2GHz. Tiene una memora RAM de 1GB compartida con la GPU y se diferencia de sus versiones anteriores por poseer módulos Wi-Fi y Bluetooth, evitando así el uso de adaptadores. También posee cuatro puertos USB, uno de los cuales será utilizado para la entrada de audio, y además tiene un conector Jack para salida de audio de 3.5mm que será en gran medida útil para este proyecto [11].

Necesita una alimentación de 5V, la cual se puede introducir tanto por su conector micro USB, como por los pines GPIO. Estos últimos son los que se usarán no solo porque ofrecen más resistencia a golpes, sino que es una manera mucho más cómoda de conectar el regulador de tensión a la Raspberry Pi.

Tiene un consumo de 400 miliamperios, los cuales se tendrán que reducir para evitar el máximo consumo posible.

[2] Micrófono USB:

La capacidad de capturar sonido no viene implementada en la Raspberry Pi, por eso es necesario una manera de grabar frases y traducirlas con el motor STT. De esta tarea se encarga el micrófono USB.

La comunicación es a por USB ya que la conversión A/D se realiza en el mismo micrófono, y es por ello que los mejores micrófonos usados para STT son con este protocolo en vez de un conector Jack, ya que así se elimina el uso de adaptadores.

El micrófono usado para el prototipo es un TONOR, un micrófono de sobremesa omnidireccional de condensador. Es una opción provisional para la realización de pruebas *indoor*:

Ilustración 2.3 - Micrófono omnidireccional TONOR

[3] Regulador reductor de tensión (Buck):

Entre la Raspberry Pi y el controlador de carga solar ocurre una diferencia de potencial que se ha de corregir. La Raspberry Pi necesita 5 V de tensión nominal, mientras que la salida del sistema fotovoltaico es de 12 V. Esto se puede resolver haciendo uso de un circuito reductor DC/DC, también llamado Buck, o *step down*.

El diseño de estos circuitos es bastante simple y barato, pero también hay opciones comerciales bastante baratas y con más funcionalidades. En el proyecto se

ha optado por comprar un módulo reductor de tensión ajustable, que puede funcionar entre 4V~35V, y tiene una salida de entre 1.25V~30V, debido a su precio reducido y comodidad de instalación

La corriente de salida es de hasta 3 A. Su reducido tamaño (43x21x14 mm) también presenta conveniencia, y la simplicidad de su instalación ha resultado oportuna para abreviar el proyecto.

Regulador de tensión Step-Down LM2596:

Ilustración 2.4 - Regulador *Buck* LM2596

[4] Altavoces:

Para la devolución de respuesta se han utilizado unos altavoces estéreo de sobremesa. Disponen de un conector macho Jack como canal de transmisión, y su alimentación es desde la red. Los altavoces, ya instalados en el sistema:

Ilustración 2.5 - Altavoces personales instalados en el montaje

En un principio la intención era diseñar un circuito amplificador Clase D para la señal de audio, pero la idea fue descartada debido a la complejidad que tomó el proyecto, y por la presencia de opciones más plausibles y sencillas.

Los altavoces fueron abiertos y analizados para poder comprobar su funcionamiento interno. Se eliminó el transformador y el cable de red de electricidad, y se ignoró el puente de diodos soldándole unos cables para poder alimentar así el circuito con corriente continua. Para su correcto funcionamiento, es necesaria una alimentación de 12 V, lo cual es conveniente ya que el sistema fotovoltaico tiene una salida de 12 V, por lo que no es necesario una etapa reductora, como en el caso de la Raspberry Pi.

También poseen un regulador de volumen y uno de tono, los cuales han sido configurados para que el motor TTS suene con claridad y suficiente potencia para ser oído por el usuario sin problema.

[5] Panel fotovoltaico

El sistema fotovoltaico dimensionado hace uso de dos paneles fotovoltaicos policristalinos de RS Pro [14], colocados en paralelo:

Ilustración 2.6 - Panel fotovoltaico policristalino RS Pro

Con cada panel, se puede obtener una potencia nominal de 5 W, una tensión nominal de 17.5 V, y una corriente nominal de 0.29 A. La tensión en circuito abierto (Voc) es de 22 V y la corriente en cortocircuito (Isc) es de 0.3 A.

Su reducido tamaño ayuda en cierta manera a camuflar el sistema, además de que da la potencia necesaria para que el proyecto funcione correctamente.

[6] Controlador de carga solar

El controlador de carga solar es el que se encarga de regular correctamente la tensión ofrecida, por los paneles fotovoltaicos, a la carga o a la batería, o a las dos a la vez. Es un dispositivo de control y seguridad, que no solo ayuda a organizar mejor el cableado, sino que además evita el deterioro de la batería por sobredescarga, ya que estas tienden a romperse alcanzado cierto nivel de tensión.

En concreto el controlador usado es un *BlueSolar 12/24V-PWM* [16] de la marca Victron Energy:

Ilustración 2.7 - Controlador de carga solar *Victron Energy Blue Power* 12/24

Es un controlador de PWM de bajo coste que viene provisto de un sensor de temperatura interna y está protegido de cortocircuitos, sobrecorrientes e incorrecta polaridad de paneles solares o batería.

Puede funcionar entre 12/24V y tiene 6 modos de funcionamiento:

0: Cargador de baterías

1-13: Control de luminaria + delay

H: Manual

C: Tensión de la carga depende de la tensión de la batería

L: Funcionamiento entre el anochecer y el amanecer

d: Igual que el modo L pero sin delay

En este proyecto el único modo de funcionamiento que conviene es el modo H, o modo manual, en el cual la carga se activará o desactivará dependiendo de un botón presente en el controlador de carga solar. Esto se debe a que el sistema ha de estar funcionando 24/7, por lo que hemos de evitar que se apague por cualquier circunstancia.

[7] Batería Gel AGM

En este proyecto, para asegurar que el programa está funcionando constantemente, se ha sobredimensionado la batería, al menos para el prototipo. Por lo tanto, se ha optado por usar una batería AGM de Victron Energy [17]:

Ilustración 2.8 - Batería de gel AGM de *Victron Energy*, instalada en el prototipo

Posee una resistencia interna baja, su tensión de funcionamiento es de 12V y posee una capacidad de 60 Ah con 20 C de descarga. A pesar de su reducida dimensión, tiene el inconveniente de su elevado peso (15 kg), propio de este tipo de baterías. Pero no supone un inconveniente mayor, ya que una vez colocado el sistema, la batería permanecerá en su sitio establecido.

2.1.2. Software

Como se ha explicado previamente, el reconocimiento de voz se alcanza con motores STT y TTS. Este tipo de software es muy difícil de crear de cero, ya que es necesario programar un léxico que contenga todas las palabras del lenguaje a sintetizar. Además, es necesario crear un diccionario de fonemas y enlazarlos con cada una de las palabras del léxico. Básicamente, programar un motor STT es muy difícil, y realmente se aleja del objetivo del proyecto. Por ello, se han hecho uso de herramientas ya existentes que facilitarán el trabajo.

A lo largo del proyecto se han considerado múltiples opciones. En el siguiente listado se enumerarán y se explican las ventajas y desventajas de cada una:

- BitVoicer y BitVoicer Server:

[3] Al principio del proyecto, se consideró el uso en Hardware de Arduino, el cual ofrecía a priori una solución simple en cuanto al apartado técnico, debido a su sencilla programación, bajo coste, y simplicidad de instalación. En consecuencia, a sus diversas ventajas, también dispone de numerosas limitaciones en cuanto al reconocimiento de voz. Existen diversos *shields* o placas adaptadas para la memorización de palabras clave, y la realización de comandos en actuadores y numerosos sistemas automáticos. Es por eso que esta elevada limitación de Hardware llevó a considerar BitVoicer, un Software de pago con grandes bibliotecas de palabras y gramática. No requiere de entrenamiento previo, y no hay limitación de memoria ni complejidad a la hora de configurar comandos. Junto un módulo de micrófono en Arduino, esta opción podría haber sido viable.

Por supuesto también presenta sus desventajas. Al ser una aplicación de pago, hace considerar si de verdad se estima su uso. Añadir también que la capacidad de funcionar sin depender de una computadora externa donde funcionase el programa era remota, y las limitaciones de memoria del Arduino aun así presentaban una lacra para el proyecto. Además, no resuelve el apartado del motor TTS, solo el STT (Speech-to-Text), por lo que se acaba llegando a la conclusión de que esta opción es demasiado compleja para su implementación, y se acaba descartando del proyecto.

Aun así, es una muy buena opción para considerar en futuros proyectos relacionados con el reconocimiento de voz. Puede llegar a ser muy útil en relación a proyectos de domótica, o control de pequeños autómatas.

- Julius:

Julius [9] es un firmware que trabaja en Raspberry Pi. Es una aplicación que no depende de conectividad a Internet, por lo que puede llegar a ser más rápida incluso que el reconocedor de voz de Google API.

Similar a BitVoicer, es utilizado para la creación de comandos, los cuales son reconocidos a través de un micrófono en la Raspberry Pi. En cambio, este sólo trabaja en inglés, aunque presenta diversos modelos de lenguaje, entre ellos el español.

Julius es una opción muy recomendada por aquellos expertos en reconocimiento de voz y, aunque tiene potencial, es poco intuitivo de usar y presenta complicaciones en la instalación. Es un firmware preparado para su implementación en proyectos mucho más específicos, y su uso debe ser precedido por una elevada experiencia en el ámbito de programación en GNU/Linux, por ello, se descartó del proyecto.

- Jasper:

Jasper [7][2] es una plataforma de código abierto utilizada para el desarrollo de aplicaciones de escucha y control de voz. Ofrece una guía detallada de su instalación, y hace uso de motores STT importantes como PocketSphinx o la API de Google. Su funcionalidad no se limita sólo en el control por comandos de voz de actuadores, sino que también funciona como un asistente personal programable que puede leer las noticias, reproducir la música que el usuario desee, o incluso programarlo para mantener pequeñas conversaciones con él.

A pesar de su prometedor potencial, Jasper ha resultado ser una herramienta difícil de manejar e instalar, provocando un lapsus de un mes en el proyecto intentando trabajar con él. Aunque la guía de instalación fuese detallada, era muy extensa y lenta. El *troubleshooting* online no era útil, y su escasez de documentación da a entender que Jasper es un Software obsoleto e incompatible en nuevas versiones de Raspbian.

Tras ese tiempo se investigó otras opciones similares a Jasper, pero más nuevas y fáciles. Se visualizaron varias opciones y se concluyó una solución final.

- Jarvis (opción final):

Jarvis [10] es un Software similar a Jasper que presenta numerosas opciones de configuración. Tiene una instalación rápida y sencilla, al contrario que Jasper. La posibilidad de usar motores STT y TTS potentes como el de Google, PocketSphinx o incluso Bing, es una opción plausible con este programa.

También puede trabajar en español, y la cantidad de comandos que se pueden configurar es casi ilimitada, ya que todos estos se almacenan en un archivo de texto el cual ocupa una cantidad de memoria ínfima.

Jarvis es la mejor baza para este proyecto, ya que está preparado para capturar frases completas y analizar las palabras clave de dichas oraciones. Por ejemplo, si se configura el programa para que diga la temperatura que hace, se asigna, en el archivo de comandos, a la palabra TEMPERATURA la respuesta que se desee que dé. De esa forma, preguntes "¿Qué *temperatura* hace?", "¿A qué *temperatura* estamos?", "¿Cuál es la *temperatura* ambiente?", responderá con la respuesta configurada a la palabra TEMPERATURA.

De la misma manera se puede configurar respuestas para diversas palabras, o para conjuntos de palabras. Esto puede ser muy útil para ayudar a realizar diversas preguntas de un mismo tema. Partiendo del ejemplo anterior, se puede configurar Jarvis para que te responda por la temperatura local, pero también se puede configurar para que te responda por la temperatura de otra ciudad.

TEMPERATURA - <respuesta de temperatura local>

TEMPERATURA; MADRID - <respuesta de temperatura en Madrid>

"*Jarvis, ¿a qué **temperatura** estamos?*":

Jarvis: <Respuesta de temperatura local>

"*Jarvis que **temperatura** hace en **Madrid**"*:

Jarvis: <Respuesta de temperatura en Madrid>

También da la posibilidad de crear árboles de respuestas, dando lugar a respuestas que solo se pueden generar dependiendo de las preguntas realizadas previamente.

Por ejemplo, si se le pregunta a Jarvis como está, responderá y preguntará cómo está el usuario de vuelta y, dependiendo de si responde “bien” o “mal” Jarvis dará una respuesta distinta coherente con la respuesta. Esto facilita la capacidad de crear una actitud más “humana” para este proyecto.

A lo largo del capítulo se indagará en cómo se ha configurado y se mostrarán diversos aspectos para el usuario concernientes a los árboles de respuestas que Jarvis puede mostrar, al igual de diversos scripts programados para poder ofrecer la respuesta adecuada en cada momento.

2.1.3. Material de apoyo

Durante el proyecto se han hecho uso de varias herramientas informáticas que han sido de elevada utilidad para poder facilitar la labor, todas ellas relacionadas con el tratamiento y control de la Raspberry Pi:

- PuTTY:

PuTTY es probablemente el programa más útil en cuanto a comunicación con la Raspberry Pi. Establece una conexión a través de SSH (Secure Shell) entre el ordenador personal y la pequeña computadora, de manera que, aunque no ofrece la interfaz de escritorio de la Raspberry Pi, permite acceder al terminal de comandos, en el cual, si se tiene cierta experiencia previa, se pueden realizar todas las acciones que se deseen, sin necesidad de una interfaz de usuario.

Su configuración es relativamente simple. Primero, ha de asegurarse que la Raspberry Pi permite la conexión SSH. La primera vez que se realiza es necesario hacerlo manualmente desde el escritorio de la Raspberry Pi (es necesario, mínimo,

un monitor, un teclado y un ratón). En “Configuración > Interfaz” activas “SSH” y tras ello será posible la conexión con PuTTY.

También se puede hacer desde el terminal de la Raspberry Pi tecleando los siguientes comandos:

```
sudo apt-get install ssh
sudo /etc/init.d/ssh start
sudo update-rc.d ssh defaults
```

Interfaz de PuTTY:

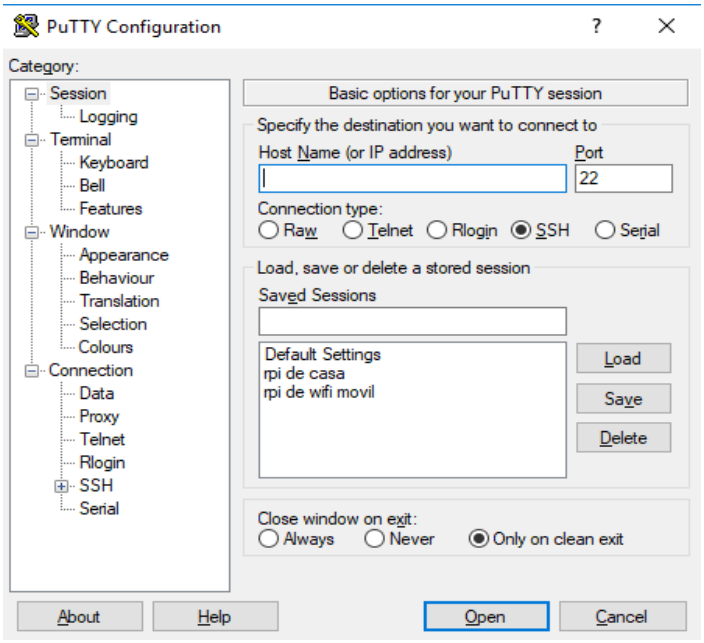


Ilustración 2.9 - Interfaz de PuTTY

Una vez está habilitada la conexión a través de SSH, se puede hacer uso de PuTTY. Es necesario que tanto tu PC como la Raspberry Pi estén conectados a la misma red. Has de conocer también la dirección IP en la que se encuentra la Raspberry Pi, la cual es conveniente fijarla previamente para conocerla de antemano y evitar tener que averiguarla cada vez que se conecte a internet, aunque la mayor parte de las veces siempre se conecta a la misma. En PuTTY se coloca esta dirección IP en “*Host Name*”, y pulsas *Open*. Aparecerá una ventana de terminal que te pedirá usuario y contraseña (si no los cambiaste, suele ser “pi” y “raspberry” respectivamente) y tras esto, se accederá al terminal de la Raspberry Pi.

Desde el terminal se puede realizar cualquier tipo de acción, librando al usuario de tener que depender de un monitor, y de un teclado y ratón USB.

- Angry IP Scanner:

Si no se ha llegado a colocar a tu Raspberry Pi en una IP fija, este programa es útil para saber en qué dirección está, y poder así colocarla en PuTTY para acceder remotamente.

Su funcionamiento es sencillo: realiza un barrido entre dos márgenes seleccionados de direcciones IP, y se dedica a mostrar que dispositivo está conectado en cada una de las direcciones analizadas. La dirección en la que se conecta la Raspberry Pi es fácil de localizar porque aparece con un indicador en el que pone “Pi”. Simplemente se ha de copiar la dirección en la que se encuentra y se pega en PuTTY.

Una vez se conecta la Raspberry Pi a una red Wifi, su dirección IP se mantendrá y para facilitar las cosas se podrá guardar un perfil en PuTTY con las distintas direcciones en las distintas redes a las que se conecte el dispositivo, evitando asi el tener que escanear la dirección a la que se conecta constantemente.

Angry IP Scanner en funcionamiento. Se puede comprobar la dirección en la que se ha conectado la Raspberry Pi, al igual que muchos otros dispositivos conectados a la misma red WiFi:

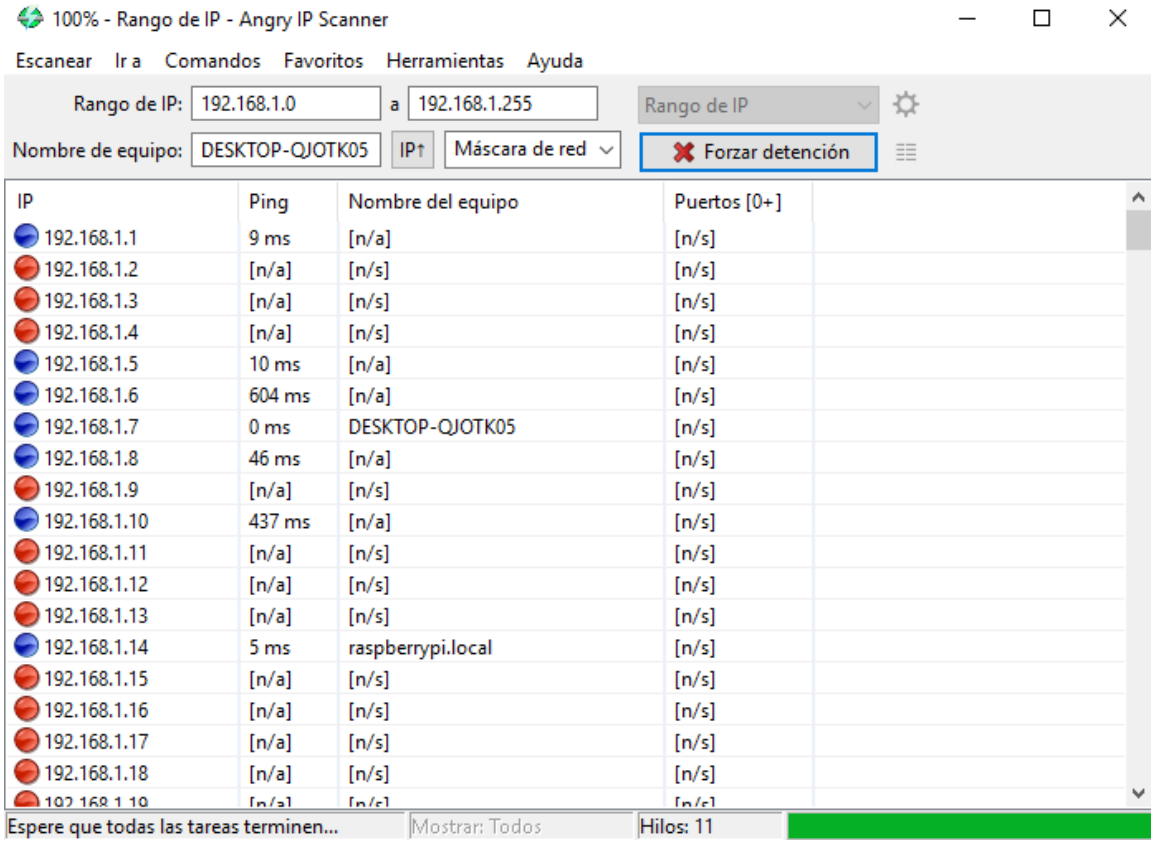


Ilustración 2.10 - Interfaz de Angry IP Scanner

- VNC Viewer (opcional):

Esta herramienta permite ver la interfaz de usuario de la Raspberry Pi desde un ordenador personal. Funcionando a la vez que PuTTY, a través de una conexión SSH es capaz de transmitir el escritorio de la Raspberry Pi, y trabajar desde el.

Funciona de una manera similar al PuTTY, solo que VNC Viewer necesita que PuTTY funcione en segundo plano. Insertando la dirección IP de la Raspberry Pi en el programa permite acceder directamente al escritorio, sin configuración previa.

La desventaja es que es lento, y a la larga se deja de depender del escritorio y se consigue realizar cualquier acción con el terminal. Por ello es completamente opcional. Es mencionada porque al principio del proyecto fue utilizada antes de acostumbrarse a los comandos de Linux en el terminal.

2.2. Jarvis

Jarvis es un asistente de voz ultraligero y multilingüe. Diseñado para la automatización del hogar, puede funcionar con sistemas como Raspberry Pi o macOS.

Su desmesurada flexibilidad de uso permite elegir el motor STT y TTS que se desee y los instala por el usuario. Es extremadamente personalizable, incluso con la adición de numerosos módulos descargables de Internet que permiten desde leer la hora, hasta reproducir música.

En este apartado se hablará de los pasos de instalación y configuración de este Software, además de la creación de comandos.

2.2.1. Instalación

El proceso de instalación de Jarvis es sencillo. Pero antes, han de cumplirse ciertos requisitos de Hardware y Software:

- Instalar en una plataforma compatible, las cuales son Raspbian (en Raspberry Pi) o Mac en OSX. En Windows no es compatible, aunque se puede usar en Debian con una máquina virtual.
- Altavoz o altavoces, para poder oír las respuestas de Jarvis
- Micrófono, preferiblemente USB y de buena calidad
- Botón (opcional) ya que se puede activar el modo escucha pulsando un botón físico. En este proyecto se utilizará una *Magic Word* o palabra de encendido.
- Diodo LED (opcional) para que muestre que el modo escucha está activo.

Una vez se tienen los requisitos mínimos cumplidos, se dispone a la instalación a través del terminal.

Primero es necesario instalar “git”, para poder acceder y descargar todo el contenido desde GitHub de Jarvis:

```
sudo apt-get install -y git # install git if you don't have it already
git clone https://github.com/alexylem/jarvis.git
cd jarvis/
./jarvis.sh
```

A partir de aquí se empezará a instalar automáticamente todos los paquetes de Jarvis. Es posible que aparezca el error de que ciertas dependencias necesarias no se encuentren en la computadora, pero el instalador se encargará de descargarlas e instalar automáticamente todas aquellas que sean necesarias.

Interfaz de Jarvis en el terminal:

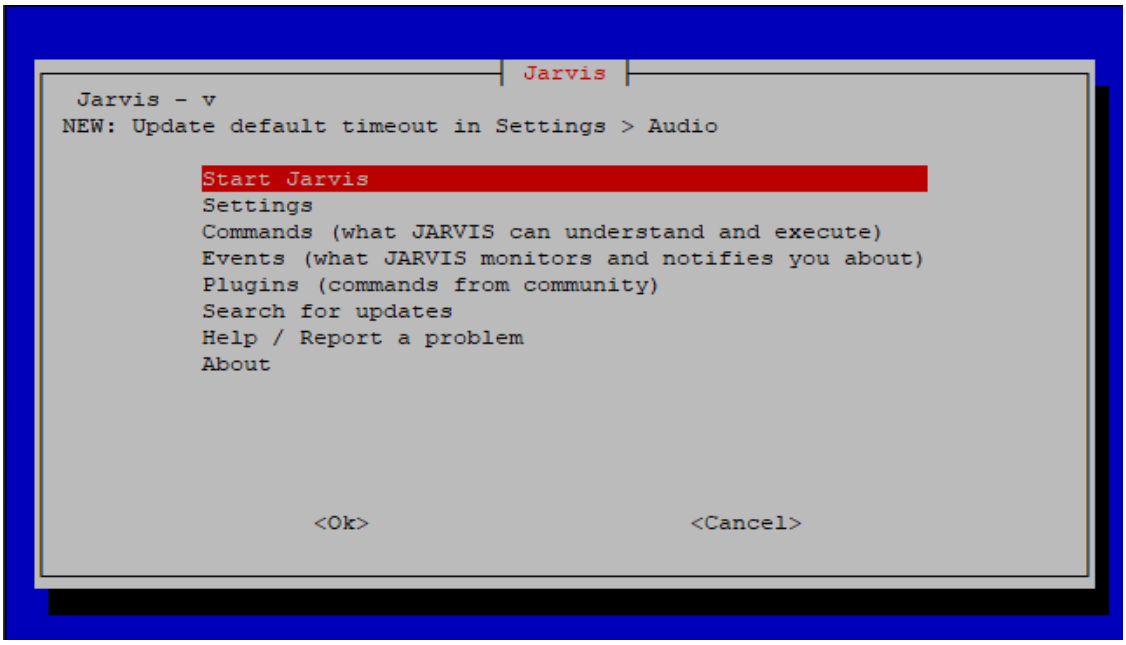


Ilustración 2.11 - Interfaz de usuario de Jarvis en el terminal Linux

Tras unos minutos, ya habrá completado la instalación, y aparecerá un *Wizard* de configuración básica, el cual preguntará por el idioma, el nombre del usuario con el que quiera referirse, y luego empezará a hacer pruebas de audio que se encargan de comprobar si los altavoces y el micrófono funcionan correctamente. Es posible que la primera vez que se instale Debian, este no reconozca ciertos dispositivos de audio. Para ello es necesario conocer la tarjeta de audio en la que se ha colocado el dispositivo, y eso se consigue introduciendo el siguiente comando en el terminal:

```
alsamixer –c 1
```

AlsaMixer es una interfaz en el terminal de Linux bastante conveniente para la selección de tarjetas de sonido y fijación de niveles de ganancia. Pulsando la tecla “f4” se puede ver y editar los dispositivos de captura (micrófonos) y pulsando “f6”, se puede seleccionar la tarjeta de sonido correspondiente al micrófono usado. Una vez seleccionada la tarjeta de sonido, es conveniente regularle la ganancia utilizando las teclas de las flechas “arriba” y “abajo”. Aunque la ganancia también se puede editar desde el mismo Jarvis, a veces su configuración automática no es correcta del todo.

Interfaz de AlsaMixer representando la ganancia del micrófono USB:

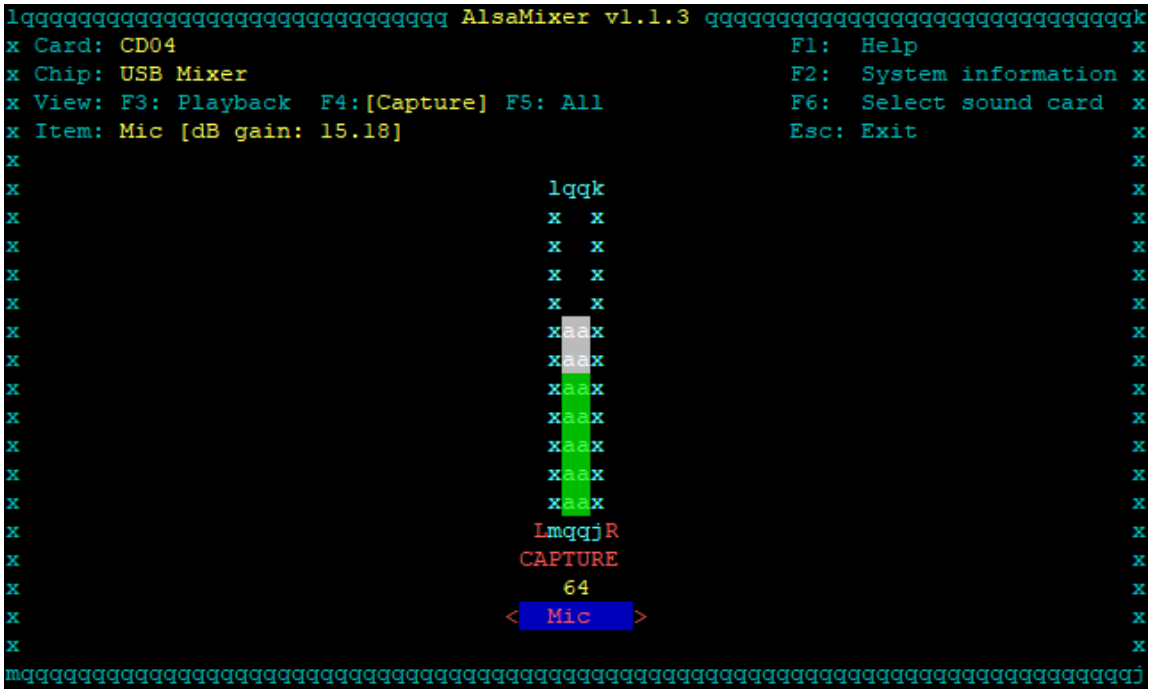


Ilustración 2.12 - Interfaz de AlsaMixer v1.1.3

Durante el *Wizard* de configuración básica también habrá que asignar una palabra clave de activación. Esta palabra es la que hará que el programa pase de “modo oyente” (modo en el cual sólo espera a reconocer la palabra clave) a “modo escucha” (modo en el cual está constantemente interpretando lo que se dice). Básicamente esta palabra es el nombre que se le quiera poner al programa. Al contrario que con los comandos configurables, esta palabra clave utiliza un motor STT diferente, que se ha de elegir. El más recomendado para este es “*Snowboy*”.

Su nombre original es Jarvis, pero por conveniencia de proyecto ha sido cambiado a *OLIVIA*, ya que le da cierta humanización y además es un nombre fácil de decir y entendible por un motor STT en español.

Tras entrenar el software a la palabra clave que se le ha asignado, es necesario elegir el motor STT con el que se desee traducir los comandos. Las opciones a elegir son Bing, Snowboy, Wit y PocketSphinx. En el proyecto se utilizará Bing, el cual necesita de una API Key para poder funcionar.

La obtención de una API Key de Bing es relativamente fácil, al contrario que la de Google. Para ello se ha de seguir los siguientes pasos:

1. Ir a Microsoft Azure y buscar dónde probar los servicios cognitivos (*Cognitive Services*)
2. Iniciar sesión con tu cuenta de Microsoft
3. Suscribirse al servicio **Speech > Bing Speech API**
4. Aceptar los términos y condiciones y pulsar Suscribir
5. Se concederán dos API Keys

Copia una de las dos API Keys en el *Wizard* de Jarvis. Tras esto solo será necesario seleccionar un motor TTS, de los que ofrece el más recomendable es *SVOX Pico*, aunque en el proyecto se decidió usar Google, porque su voz es más natural. Tras esto, Jarvis ya estaría listo para su uso corriente.

El siguiente paso es la configuración y condicionamiento para el proyecto.

2.2.2. Configuración

Tras haber realizado el *Wizard* de configuración básica, se puede repetir este siempre que se desee. También se pueden hacer configuraciones más específicas:

i. Configuración General: En este apartado el usuario puede cambiar varios aspectos del modo de funcionamiento y trato del usuario con Jarvis. En concreto, se puede cambiar el alias con el que quiera que llame al usuario (en este caso será “*Humano*” ya que el trato será de árbol a persona). De las propiedades a destacar se debe mencionar

también la manera en la cual Jarvis pasa de modo oyente a modo escucha, el idioma, entre otros.

ii. Frases: En esta sección se configuran las frases predeterminadas para ciertas situaciones, como son el saludo de bienvenida, la respuesta a la palabra clave, o la respuesta a una pregunta errónea o no configurada.

iii. Audio: Aquí se configura todo lo relacionado con la entrada y salida de audio. Las propiedades más importantes a destacar son la sensibilidad, el volumen y la ganancia del micrófono. También tiene un sistema de autoajustado de niveles para la salida y entrada de audio, el cual se realiza mediante una serie de pruebas en las que el usuario ha de participar hablando con naturalidad.

iv. Reconocimiento de voz: Es el apartado encargado de configurar los motores TTS que se vayan a usar. Si se va a usar Bing o Google para traspasar voz a texto, es necesario una API key como la anteriormente mencionada. Es en esta sección donde se ha de colocar la clave correcta, y renovarla si es necesario.

v. Síntesis de voz: En esta sección se selecciona el motor TTS con el que se desee que Jarvis devuelva las respuestas. Como se ha mencionado antes, en el proyecto será Google debido a la naturalidad de su voz.

vi. Hooks: Este apartado es especialmente útil para acomodar el programa perfectamente a unos requisitos más obcecados. La configuración de estos *Hooks* o "ganchos" no serán necesarios, ya que la configuración de serie es suficiente para el sistema.

Por otro lado, Jarvis ofrece la configuración de lo que denomina "eventos". Estas operaciones son una serie de líneas de comando programables para que ocurran a determinado minuto, hora, dia de la semana o mes, y estos se realizarán automáticamente en el momento configurado.

Estos "eventos" servirán para acciones como decir la hora, felicitar año nuevo, decir el tiempo que hace todas las mañanas, todo ello sin preguntarle previamente, dándole cierta autonomía.

Se han de configurar de una manera especial, y son convenientemente mostrados en la misma ventana de configuración, la cual muestra el orden del código y el tipo de comandos que se pueden poner, junto a algunos ejemplos aplicables.

Pantalla de configuración de eventos:

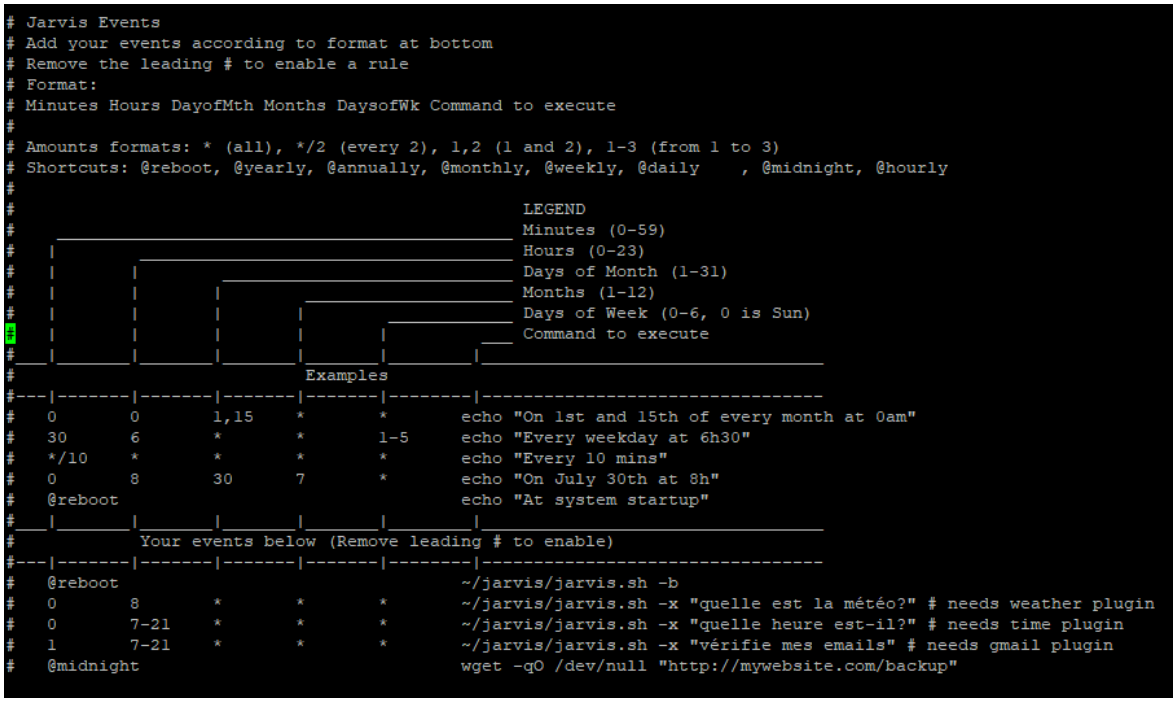


Ilustración 2.13 - Configuración de eventos en Jarvis

2.2.3. Comandos

A continuación, se explica la propiedad más importante, y es la lista de comandos configurables, todos ellos guardados en un archivo de texto llamado “jarvis-commands.txt”.

Desde aquí se pueden ejecutar líneas de código en *Bash Script* directamente, o se pueden hacer llamadas de programas en Java, pero básicamente es donde se configura las respuestas que posteriormente serán salida auditiva. Su orden en el

archivo es importante, ya que es el que define la prioridad del comando sobre otros ya que cabe la posibilidad de que una misma palabra clave se incluya en varios comandos, por lo que esto se puede corregir dependiendo de la posición en la que se encuentren los comandos relativamente en la lista, dando prioridad a los que más arriba se encuentren en ella.

La creación de comandos se basa en una formula muy simple:

PALABRA==say “RESPUESTA”

Antes del “==” se ha de colocar la palabra o conjunto de palabras clave que activen la respuesta deseada. La palabra o frase clave se ha de poner entre asteriscos (*).

Si las palabras van separadas por plecas (|) se pueden crear distintas palabras clave alternativas para la misma respuesta. Aquí dos ejemplos mostrando la diferencia de funcionamiento:

```
*COMO ESTAS*==say “Estoy bien, gracias” #Esta respuesta solo ocurrirá si en la misma pregunta se encuentran ambas palabras clave “como” y “estás”
```

```
*COMO ESTAS*|*QUE TAL ESTAS*==say “Estoy bien, gracias” #Esta respuesta se realizará tanto si preguntas “como estás” como si preguntas “que tal estás”
```

También se puede usar el símbolo “mayor que” (>) para crear comandos anidados. Esto proporciona una sensación de conversación más natural, ya que Jarvis también tomaría el papel de locutor.

Por ejemplo:

```
*COMO ESTAS*==say “Estoy bien, ¿y tú?”

#las siguientes respuestas solo sucederán si previamente se ha realizado el comando anterior:
>*BIEN* ==say “Me alegro”; exit
```

```
>*MAL* ==say “Siento oír eso, ¿puedo ayudarte en algo?”
>>*SI* ==say “Estaré atento a cualquier cosa”; exit
>>*NO* ==say “De acuerdo, avísame con lo que sea”
```

Una conversación con esta configuración podría ser tal que así (las palabras subrayadas son las palabras clave que provocan las respuestas):

Usuario: Jarvis ¿cómo estás?
Jarvis: Estoy bien, ¿y tú?
Usuario: Hoy estoy un poco mal.
Jarvis: Siento oír eso, ¿puedo ayudarte en algo?
Usuario: No, gracias.
Jarvis: De acuerdo, avísame con lo que sea.

Después del “**==**” es donde va la respuesta. Pero no tiene por qué ser siempre una respuesta conversacional. Líneas de código en Shell Script pueden ser incorporadas de manera directa, permitiendo la ejecución automática de comandos, o uso de identificadores:

`$trigger` - Es el nombre asignado a Jarvis. En el sistema será Olivia.

`$username` - Es el nombre del usuario. El configurado es Humano.

`$order` – Envío de orden a un motor de comando externo.

Esto será útil a la hora de recopilar la información procedente desde ThingSpeak, la cual se explicará en el siguiente punto.

El uso de comandos de terminal de Linux también está disponible, por lo que la posibilidad de controlar de manera totalmente vocal la Raspberry Pi es plausible.

```
*CREAR CARPETA EN (*) DE NOMBRE (*) == cd “(1)” && mkdir “(2)” && say “Hecho”

*APAGATE* == say “De acuerdo. Adiós”; sudo shutdown now
```

El primer ejemplo es una línea de comando que permite crear carpetas en el lugar y nombre que se desee. El segundo ejemplo permite apagar la Raspberry Pi sin tener que pulsar ningún botón, ni escribir ningún comando.

Los comandos se pueden habilitar y deshabilitar añadiendo una almohadilla (#) al principio de la línea de código, de esa manera no es necesario eliminarlo entero del archivo. También se pueden colocar comentarios que permiten aclarar el uso del código.

En la siguiente tabla se pueden comprobar todos los comandos configurados para el proyecto.

Las palabras clave solo es necesario ser oídas una vez en la oración para poder activar el comando:

Palabra/s clave	Descripción
AYUDA	Muestra en pantalla la lista de todos los comandos disponibles
*COMO*TE*LLAMAS	Responde con su nombre y la especie de árbol que es
*REPITE (*) Y (*)	Repite en formato auditivo las dos palabras dichas por el locutor
PRUEBA	Hace una pequeña prueba de funcionamiento
HOLA	Saluda y pregunta que puede hacer por el locutor
ADIOS | *HASTA LUEGO* | *NOS VEMOS*	Se despide cordialmente (no cierra la aplicación)
COMO ESTAS | *COMO*SIENTES* | *TAL*ESTAS*	Devuelve una respuesta variable dependiendo de la humedad y temperatura ambientales. Responde acorde con el nivel de confort al que se encuentre
*QUE*PUEDES* | *QUE*SABES* | *QUE*CAPAZ*	Responde con una descripción general de lo que puede hacer
*TEMPERATURA*VIGO*	Accede a un canal público de presente en Vigo y dice su temperatura ambiente
*HUMEDAD*VIGO*	Accede a un canal público en Vigo y dice su humedad ambiente
*TEMPERATURA*CORDOBA*	Accede a un canal público en Córdoba y dice su temperatura ambiente
*HORA*ES*	Responde con la hora actual
DIA	Responde el dia y mes actual

*TEMPERATURA*HACE* | *TEMPERATURA*ESTAMOS*	Hace uso del canal privado de ThingSpeak para responder la temperatura ambiente en Jaén. Su respuesta varía dependiendo del calor o frío que haga
*HUMEDAD*HACE | *HUMEDAD*ESTAMOS* | *HUMEDAD *HAY*	Hace uso del canal privado de ThingSpeak para responder la humedad ambiente en Jaén
OLIVIA	Responde a su nombre como si se le fuera a realizar una petición
CANAL	Es un comando de prueba que accede al canal privado de ThingSpeak para devolverte de forma auditiva los últimos datos recogidos en el canal. Se usa para comprobar que todos los datos le llegan correctamente

Tabla 2.1 – Tabla completa de comandos configurados en el sistema

En alguno de los comandos se aprecia que hay varias opciones. Esto se debe a que para una misma respuesta (como puede ser la despedida) se plantean todas las opciones posibles ya que el vocabulario presenta tantas expresiones con un mismo significado que es necesario prever todas las posibilidades.

Tabla representativa de las respuestas posibles cuando se le pregunta al sistema “que tal está”:

Temperatura → Humedad↓	POCO (<15)ºC	NORMAL (16 ~ 29)ºC	MUCHO (30 ~ 50)ºC	DEMASIADO (>50)ºC
POCO (<10)%	Morir de frio y sed. No apetece hablar	Se está agradable pero le vendría bien un poco de agua	Mucho calor y muy seco. Lo normal en un verano jiennense	Probablemente se haya producido un incendio. Literalmente muriendo.
NORMAL (11 ~ 40)%	Morir de frio y sed. No apetece hablar	Se encuentra ni bien, ni tan mal. Condiciones corrientes.	Agradece la humedad, le hace sobrellevar el calor.	Quejándose del calor. Deseando que acabe ya.
MUCHO (41 ~ 70)%	Queja por congelarse porque hace frio y está mojada	Agradable temperatura, pero demasiada humedad. No nos pasemos	Mucho calor y mucha humedad. Si se alarga demasiado se puede hacer incomodo	Humedad insoportable y calor insoportable. Se queja.
DEMASIADO (>70)%	Acaba de llover, lo agradece pero el frio no ayuda	Probablemente haya llovido. Si sigue asi dará muchos frutos.	Probablemente acaba de llover. Se agradece porque la sensación térmica bajará.	Llueve. Solo espera que el agua después no queme sus hojas.

Tabla 2.2 – Tabla de las distintas respuestas configuradas para *script_feel*

Algunos de estos comandos están configurados directamente en el mismo archivo de texto, pero hay algunas preguntas que requieren la configuración de un gran número de respuestas. Estas están programadas en un programa *Bash Script* contenido en la carpeta del Jarvis, y se llaman desde el mismo programa. Su funcionamiento se explicará en el siguiente apartado.

2.3. Proceso de obtención de datos

Como se ha explicado previamente, el proyecto se basa en la recolección de datos desde la Nube, los cuales pertenecen a sensores implementados en el olivo, que recogen información concerniente a la humedad, temperatura, entre otros.

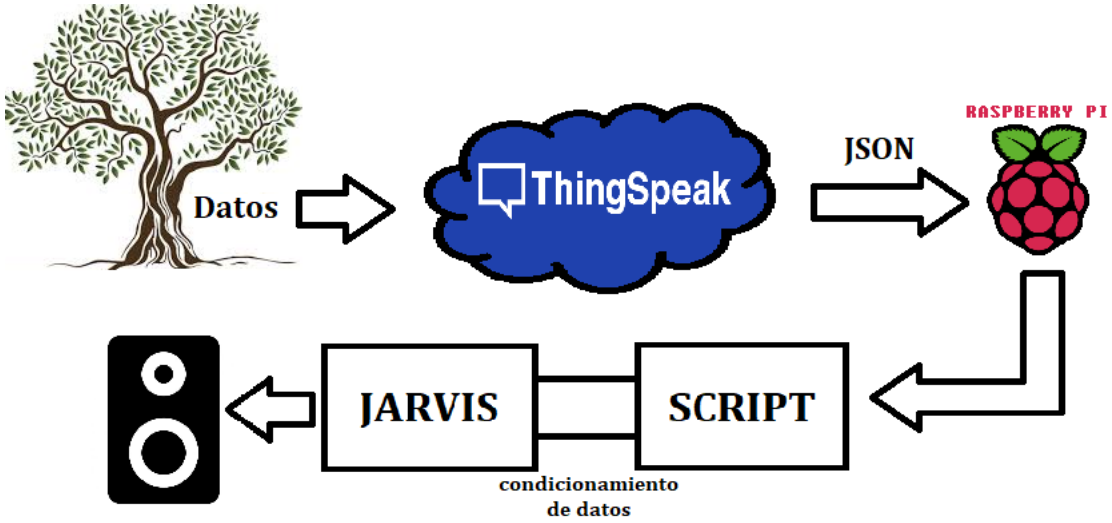


Ilustración 2.14 - Esquema de trayectoria de datos del sistema

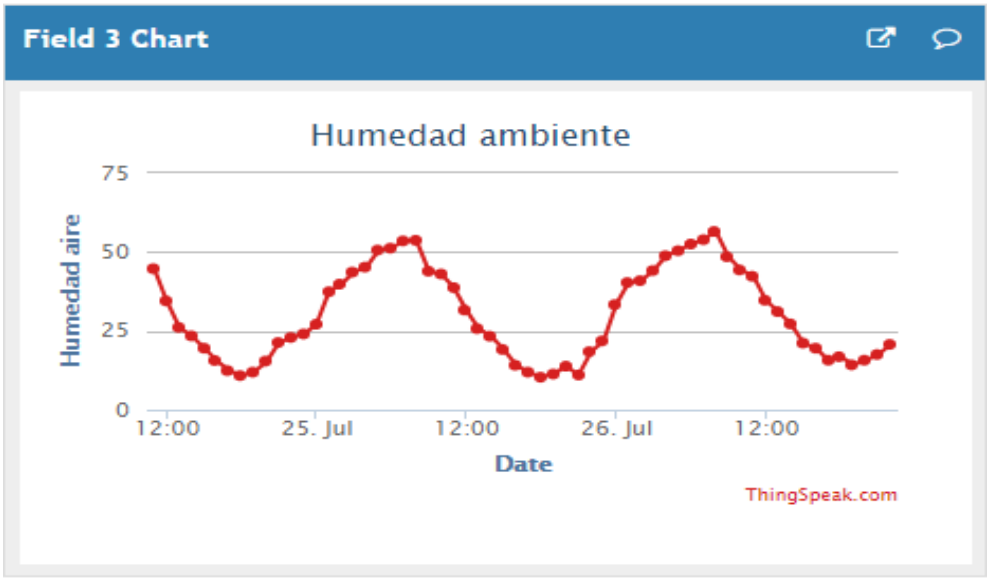
A lo largo del capítulo se mencionan los datos manipulados en la Nube (ThingSpeak), cómo se obtienen e interpretan en el sistema, y cómo se adaptan al software de reconocimiento de voz.

2.3.1. Datos de la Nube (ThingSpeak)

De los creadores de MATLAB y Simulink surge ThingSpeak. Es una plataforma gratuita que permite crear diversos canales, públicos o privados, en los cuales se puede visualizar y analizar la evolución a tiempo real de las condiciones de diversos sensores.

En este sistema, los sensores implementados en el olivo recogen datos ambientales y superficiales, como la temperatura, la humedad, o incluso la tensión restante de la batería que alimenta el sistema. Estos datos se actualizan a tiempo real en la plataforma, con un *delay* de 5 minutos entre dato y dato, pero dando por hecho que los datos son de condiciones ambientales, este retraso no supone un problema, debido a la lenta evolución de las condiciones meteorológicas en un periodo de tiempo tan corto. Desde ThingSpeak, estos datos se enviarán a Jarvis, el cual los diversos scripts configurados se encargarán de interpretarlos como sea conveniente para la situación.

Aquí se muestra un ejemplo de una gráfica representada por ThingSpeak de como varía el porcentaje de humedad ambiente alrededor de un olivo plantado en Córdoba:



Gráfica 2.1 - Gráfica de la humedad ambiente en Córdoba (ThingSpeak)

Para el testeo del prototipo se ha creado un canal privado en ThingSpeak donde se emula la representación de las condiciones que son medidas por los sensores en el olivo, las cuales son temperatura ambiente, humedad ambiente, humedad del suelo y tensión de la batería. Los datos de esta Nube son cifras que son modificadas para probar la correcta coordinación entre el programa y la Nube, y para testear los scripts de Jarvis. El siguiente cuadro muestra el JSON que se


```
{"channel":{"id":481345,"name":"Channel test","description":"Este canal lo estoy utilizando no como recopilador de datos sino como experimento para obtener datos de el.","latitude":"0.0","longitude":"0.0","field1":"Voltaje bateria","field2":"Temperatura Ambiente","field3":"Humedad Aire","field4":"Humedad Suelo","created_at":"2018-04-23T08:54:01Z","updated_at":"2018-08-24T09:31:04Z","last_entry_id":310},"feeds":[{"created_at":"2018-08-24T09:33:13Z","entry_id":310,"field1":"10","field2":"20","field3":"30","field4":"40
```


Cuadro 2.1 - Representación de los datos mostrados en formato JSON

obtiene del canal privado donde se varían los datos para testear a Jarvis:

Si el canal es privado, la modificación y obtención de los datos depende de la *API key* que proporcione ThingSpeak.

2.3.2. Adquisición e interpretación de los datos

Los datos de cada canal son exportados de la Nube desde una URL que te muestra el *Channel Feed*, el cual se representa en formato JSON (*JavaScript Objetct Notation*). Para la obtención de estos mismos datos se han considerado dos opciones: Realizar un programa en Java, o realizarlo en *Bash Script*.

En ambos casos es necesario realizar las siguientes operaciones:

- Conseguir una petición HTTP para poder acceder a la URL de ThingSpeak.
- Obtener el JSON.
- Buscar dentro del JSON el dato que requiramos.

Puesto que cada canal ofrecerá un dato (humedad, temperatura, etcétera) es necesario realizar un programa genérico que sea capaz de funcionar en todos los canales. Por suerte ThingSpeak muestra los datos de todos los canales de la misma manera. El valor útil es el último dato recibido, por eso se ha de condicionar el programa para que siempre recoja este.

El siguiente código muestra el programa creado en Java para la recolección de datos. En él se muestran las herramientas para realizar la petición HTTP, además de las de lectura e interpretación:

```
package json.capturer;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.URL;
import java.nio.charset.Charset;
import org.json.JSONObject;
```

```
import org.json.JSONException;
import org.json.JSONArray;
public class JSONCapturer {
    private static String readAll(Reader rd) throws IOException {
        StringBuilder sb = new StringBuilder();
        int cp;
        while ((cp = rd.read()) != -1) {
            sb.append((char) cp);
        }
        return sb.toString();
    }
    Public static JSONObject readJsonFromUrl(String url) throws IOException,
    JSONException {
        try(InputStream is = new URL(url).openStream()) {
            BufferedReader rd = new BufferedReader(new InputStreamReader(is,
Charset.forName("UTF-8")));
            String jsonText = readAll(rd);
            JSONObject json = new JSONObject(jsonText);
            return json;
        }
    }
    public static void main(String[] args) throws IOException, JSONException {
        JSONObject json = readJsonFromUrl("https://api.thingspeak.com/
         channels/481345/feeds.json?api_key=63JGUHLE166J2JVY&results=1");
        Object feeds = json.get("feeds");
        JSONArray arrayfeeds = json.getJSONArray("feeds");
        String resultado = arrayfeeds.getJSONObject(0).getString("field1");
        int result = Integer.parseInt(resultado);
        System.out.println(result + '\n');
    }
}
```

El programa, denominado JSONCapturer, presenta dos funciones:

- readALL (Reader rd): Es una función encargada de transformar el archivo procedente de la URL en un *String*, permitiendo así al resto del programa Java analizar mejor el dato necesario. Devuelve una variable de tipo String.
- readJsonFromUrl (String url): Como variable de entrada tiene la URL del cual sacar el JSON, que es la que proporciona ThingSpeak. Lee el Buffer de datos procedente de la URL y lo mete en una variable, en la

cual se utiliza la anterior función, readALL y, posteriormente, se transforma el String resultante de la anterior función en una variable de tipo JSON. Conforme se obtiene la variable JSON con todos los datos de ThingSpeak correctamente introducidos en el programa, se procede a interpretarlos en el *main*.

En el *main* se hacen uso de la función readJsonFromUrl. Dentro de una variable de tipo objeto, se introduce la lista "*feeds*" procedente del JSON, que es donde se encuentran listados todos los datos recogidos por el sensor, ordenados cronológicamente, y se convierte en un Array. Posteriormente, dentro del mismo Array, se puede encontrar ya el último dato recogido, ya que, aunque el JSON se vaya actualizando, el Array nunca variará de tamaño. El dato se obtiene en una variable de tipo entero y es devuelta por el programa, el cual finaliza.

Dejando de lado la opción de Java, existe la opción de *Shell Script* (o *Bash Script*), la cual ofrece una solución rápida y perfectamente aplicable en Jarvis.

```
$(curl -s "https:/ /thingspeak.com/channels/377981/field/2.json" | jq '.feeds[99].field2')
```

Lo que se ve en este ejemplo previo es una línea de código en *Bash Script* (o *Shell Script*). En concreto esta es el ejemplo más básico para obtener datos desde la Nube, y está preparada para recoger el último dato de un canal público procedente de Vigo, el cual muestra la temperatura ambiente. El comando `curl` permite acceder a la URL seleccionada, y recoger su contenido. Seguido de `-s`, esto permite evitar que el comando muestre en el output todo el contenido recogido. La pleca permite acceder a los datos recogidos, y tras ello buscar en el JSON el dato pertinente. En este caso busca el array "feeds", en la fila 99, y recoge el dato asignado a "field2", el cual es, en este caso, la última temperatura recogida por el termómetro.

2.3.3. Incorporación de datos en el sistema (Jarvis)

Tras haber parafraseado los datos procedentes desde la Nube, es necesario implementarlos en Jarvis para que este sea capaz de devolverlo en formato auditivo. Jarvis tiene la cualidad de trabajar bien en diversos lenguajes de programación, pero concerniente a los comandos programables de voz, *Bash Script* es la mejor opción.

Como se ha explicado antes, esto se debe a que la incorporación en la lista de comandos de un programa Bash es íntegramente sencilla, ya que el dato devuelto por el programa es capaz de ser interpretado por el motor TTS sin necesidad de depender de un interlocutor tercero.

Volviendo al ejemplo de línea de código en Bash Script del punto 2.3.2, este se puede meter de manera directa como una respuesta a una pregunta que no requiera de varias preguntas, como, por ejemplo: "¿Qué temperatura hace?". Esta pregunta siempre tendrá una respuesta (la cual se va actualizando por el dato procedente de la Nube), y no es necesario programar un abanico de respuestas. Pero cuando se quiere programar varias respuestas a una misma pregunta, colocar el *Bash Script* directamente en el archivo de comandos de voz es demasiado desordenado.

Por ejemplo, este es un pequeño programa, `script_date`, que responde a la pregunta de la fecha actual:

```
#!/bin/bash
dia=$(date +"%d")
mes=$(date +"%B")
case $mes in
    "January") mes=Enero;;
    "February") mes=Febrero;;
    "March") mes=Marzo;;
    "April") mes=Abril;;
    "May") mes=Mayo;;
    "June") mes=Junio;;
    "July") mes=Julio;;
    "August") mes=Agosto;;
    "September") mes=Septiembre;;
    "October") mes=Octubre;;
    "November") mes=Noviembre;;
    "December") mes=Diciembre;;
    *)
esac

echo Hoy es $dia de $mes
```

Debido a que el comando `date` en Linux te devuelve la fecha en inglés, se ha creado este programa para que Jarvis pueda devolver la fecha en español, comparando cadenas de caracteres. Reconociendo la fecha en inglés, es capaz de devolverte, por tanto, en español, la fecha a la que se encontraría en ese momento. Considerando este programa un script de los pequeños, pensar en incluir todos los

scripts en el archivo de comandos supondría un desorden que no se debería alcanzar. Por ello se han programado diversos scripts preparados para cada pregunta que pueda responder de diversas maneras.

Los scripts creados para el prototipo se encargan no solo de recoger el dato y colocarlo en una frase, sino que hace uso del mismo valor para delegar la respuesta de Olivia. Todos ellos permiten apreciar el potencial del proyecto, ya que enseñan que no hay límites en cuanto a las posibilidades de crear preguntas y respuestas para Olivia. Los Scripts creados son:

Nombre	Archivo	Descripción
Fecha	`script_date`	Es el script encargado de devolver la fecha actual. Compara cadenas de caracteres con los procedentes del comando `date` de Linux, y devuelve el día y mes en español.
Datos de canal privado	`script_thingspeakpc`	Recoge los datos procedentes del canal privado de ThingSpeak. Los datos son, en orden, batería, temperatura ambiente, humedad ambiente y humedad de suelo.
Condición física	`script_feel`	Se encarga de devolver una respuesta acorde con las condiciones atmosféricas (ver: Tabla 2.2). Muestra su apreciación por el tiempo, o su descontento.
Temperatura ambiente	`script_temperature`	Devuelve una respuesta acorde con la temperatura ambiente, dependiendo de si hace frío o calor.
Humedad	`script_humidity`	Devuelve los datos de humedad ambiental y humedad del suelo.
Hora	`script_timeofday`	Analiza la hora en formato de 24h y dependiendo de si es por la mañana, por la tarde o por la noche, si le saludas con un “buenos/as dias/tardes/noches” te responderá acorde.

Tabla 2.3 - Scripts creados para la devolución de numerosas respuestas

La creación de estos scripts es sencilla. El primer paso es generar un archivo de texto. Para ello es necesario estar en la carpeta donde se contiene el programa, en este caso, `~/Jarvis`.

Una vez posicionados en la carpeta contenedora, para crear un archivo de texto hay que hacer uso del comando `nano` seguido del nombre del archivo. Este comando es un editor de textos de Linux en el cual se pueden realizar varios programas en cualquier lenguaje deseado.

```
~/jarvis $ nano script_date
```

Una vez dentro del editor de texto, la primera línea de código es indicar el lenguaje de programación que va a ser usado, para que el programa entienda en que idioma está hablando. En este caso el lenguaje de programación es *Bash Script*, por lo que la primera línea de código ha de ser:

```
#! /bin/Bash
```

A partir de aquí ya se puede empezar con la programación del script. Un ejemplo simple es el de la fecha, presentado anteriormente. Este otro ejemplo es un poco más laborioso, y es el que concierne a la devolución de la temperatura, esta vez, de Jaén:

```
#! /bin/bash

temperatura=$(curl -s
"https://api.thingspeak.com/channels/481345/feeds.json?api_key=63JGUHLE166J2JVY&results=1" | jq
'.feeds[0].field2') #Accede a la URL y recoge el dato pertinente

temperatura=${temperatura//[!0-9]/} # Elimina cualquier carácter que no sea número
temperatura=${temperatura%.*} # Elimina los decimales

if [ $temperatura -lt 20 ]
then
        echo "Para ser Jaén, hace frío. Ahora mismo nos encontramos a una temperatura de
        ${temperatura} grados centígrados."
elif [ $temperatura -ge 20 -a $temperatura -le 29 ]
then
        echo "Hace una temperatura agradable. Nos encontramos a unos ${temperatura} grados.
        ¡perfecto para un paseo!"
elif [ $temperatura -gt 29 -a $temperatura -le 40 ]
then
        echo "Estamos a ${temperatura} grados. Hace un poco de calor. Te sugiero que te cobijes
        bajo mi sombra, si quieres"
elif [ $temperatura -gt 40 ]
then
        echo "¡Por Dios! ¡estamos a ${temperatura} grados! te sugiero que te vayas a la piscina o
        te tires a la fuente de la plaza de los pueblos"
fi
```

Este pequeño programa hace uso del dato de la temperatura ambiente proporcionado por la Nube. Al principio se realiza la llamada HTTP y se recoge el

dato pertinente del JSON. Tras ello, el dato se pule para obtener el valor entero con el que se va a trabajar. Luego se hacen uso de los condicionales que son los que devolverán una respuesta u otra. Se puede comprobar que las condiciones presentan expresiones como “-le” (*less or equal*) o “-gt” (*greater than*). Esas expresiones representan respectivamente la condición de “menor o igual que” y la de “mayor que”.

Por otro lado, tenemos las líneas “echo”. Son el equivalente a “return”, y todo lo que vaya después es devuelto por el programa. En este script todos los casos devuelven una cadena de caracteres, o “string”, que es lo que utiliza el motor TTS de Jarvis para poder hablar. En cada respuesta se puede apreciar la expresión “${temperatura}”. Esto se debe a que en *Bash Script* cuando quieres hacer referencia a una de las variables del programa es necesario colocar previamente el símbolo del dólar ($) y para no ser considerado un “string”, se coloca entre corchetes, de esa manera cuando devuelva la respuesta dirá el valor almacenado en esa variable, en este caso, la temperatura ambiente.

Más adelante, en el capítulo 3. Resultados, se indaga más en los diversos comandos programados en Jarvis para mostrar la flexibilidad de su uso.

2.4. Análisis de consumo energético

Como se ha mencionado previamente, el proyecto ha de estar cumplidamente autonomizado. Para ello se ha aplicado energía solar proporcionada por unos paneles fotovoltaicos, los cuales idealmente han de proporcionar la energía necesaria durante el día como para cargar la batería a la vez que alimentar el sistema. Para su correcto funcionamiento es necesario dimensionar el sistema apropiadamente, teniendo en consideración la corriente de la carga (Raspberry Pi y altavoces), y la capacidad de descarga de la batería.

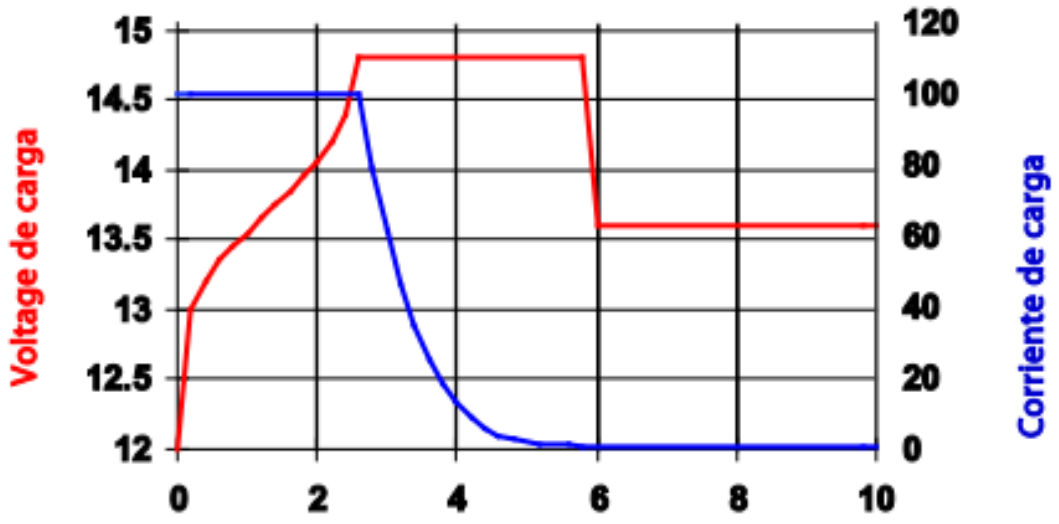
A lo largo de este apartado se mostrarán los cálculos necesarios y los datos prácticos para el correcto funcionamiento en base a los materiales disponibles.

2.4.1. Dimensionado de batería y sistema fotovoltaico

La principal fuente de alimentación será una batería AGM de la marca *Victron Energy*, condicionada para su uso en sistemas fotovoltaicos y preparada para descargas profundas. En concreto esta batería fue usada en un proyecto previo, en el cual su carga también dependía de energía solar.

Hay numerosas propiedades a considerar a la hora de tratar con este tipo de baterías. Por ejemplo, la carga presenta tres etapas en las cuales la corriente y tensión evolucionan. La fase en la cual la corriente suministrada es constante se denomina fase "*Bulk*", tras la cual la corriente empieza a disminuir y la tensión se mantiene constante en las siguientes dos etapas: "*Absorción*" y "*Flotación*".

Este diagrama presenta corriente y tensión de carga a lo largo del tiempo:



Gráfica 2.2 - Representación de la tensión y corriente en el periodo de carga de una batería

Otro aspecto que se puede considerar es la profundidad de descarga de la batería (*Depth of Discharge DOD*), que se refiere al porcentaje de la capacidad total de la batería que se usa durante un ciclo de carga o descarga. Cuanto mayor es el DOD, menos ciclos de uso va a poder dar la batería. El número de ciclos de carga y descarga afectan a la vida útil de la batería, pero este problema no será tratado en este proyecto.

La batería utilizada presenta una tensión nominal de 12 V y tiene una capacidad de C20 = 60 Ah. Esto quiere decir que a lo largo de 20 horas la batería va a proporcionar 60 A, o lo que es lo mismo, en una hora proporcionará 3 A. La fórmula correspondiente es la siguiente:

$$C_N[Ah] = I_N[A] * t\ [h]$$

Por lo tanto, si por ejemplo se somete la batería a una corriente extraída de 5 amperios, siendo la capacidad 60 Ah, el régimen de descarga de la batería sería de 12 horas (teóricamente, ya que hay factores, como la temperatura, que alteran estos resultados).

Por otro lado, tenemos la Raspberry Pi junto con los altavoces. Se puede considerar que los altavoces, aunque estén encendidos, al no estar reproduciendo sonido constantemente su consumo es ínfimo. En cambio, la Raspberry Pi 3 model B tiene un consumo nominal de 480 mA (2.4 W). Considerando que se ha optimizado para consumir menos, y que Jarvis es un programa que requiere de poca memoria RAM, se puede suponer que su consumo teórico es de unos 380 mA.

Tabla de consumo de memoria de la Raspberry Pi con Jarvis funcionando en segundo plano:

```
pi@raspberrypi:~ $ free -m
              total        used        free      shared  buff/cache   available
Mem:            927          86         468          13         372         771
Swap:            99           0          99
```

Tabla 2.4 - Representación del rendimiento de la Raspberry Pi en su máximo uso en el proyecto

Teniendo en cuenta estos datos, es posible calcular cuánto puede durar la batería sin depender de paneles fotovoltaicos:

$$C_N = 60\ Ah\ ;\ A_N = 0.38\ A$$

$$t = \frac{60\ Ah}{0.38\ A} \cong 158\ horas$$

Por lo tanto, con esta batería el sistema podria estar funcionando durante casi una semana sin preocupación de que se apague, lo cual aquí en el sur, donde el sol

está disponible casi todos los dias del año, permite concluir que podría estar funcionando sin problema, siempre y cuando los paneles fotovoltaicos estén correctamente dimensionados y reciban irradiación solar de una manera correcta. Es incluso plausible pensar que la batería está sobredimensionada, por lo que no es mezquino plantear el uso de una batería más pequeña. Lo idóneo para reducir gastos sería hacer uso de una de las baterías que ofrece Victron Energy más reducidas, en concreto la inmediatamente inferior a la utilizada, de 38 Ah, que en comparación cuesta entre 30 y 35 € menos, y proporcionaría 100 horas de autonomía absoluta, que son suficientes.

Por otro lado, tenemos los paneles fotovoltaicos policristalinos de RS Pro. Cada uno tiene una potencia y corriente nominal de 5 W y 0.3 A respectivamente. Siendo el consumo de la Raspberry 0.38 A, para que la batería se cargue durante el día es necesario que los paneles sean capaces de producir más corriente de la que se consume. Es por ello que en principio se ha hecho uso de dos paneles en paralelo, para que ambos apliquen la tensión nominal de 16.8 V, pero el doble de corriente (alrededor de 600 miliamperios). Parte de esta corriente alimentaría por completo la carga (Raspberry Pi más altavoces) y lo restante estaría dedicado a la recarga de la batería, lo que quiere decir que durante el dia la batería se carga a una tensión de 16.8 V y corriente de:

$$I_{carga} = 600\ mA - 380\ mA = 220\ mA$$

Si se supone que el sistema recibe durante el día unas 12 horas de luz solar directa, se puede calcular los vatios-hora que adquiere la batería:

$$Energía\ generada = 16.8\ V(paneles) * 0.22A(corriente\ restante) * 12\ h = 44.35\ Wh$$

En cambio, considerando las otras 12 horas del día totalmente oscuras (de noche) se puede calcular la potencia que consume el sistema:

$$Energía\ consumida = 12\ V(batería) *\ 0.38\ A(carga) * 12\ h = 54.72\ Wh$$

En el hipotético caso de que la noche durase tanto como el día, se puede comprobar que se consume más energía de la que se consigue, planteando el problema de que a la larga la batería se iría lentamente descargando.

Sería pertinente pensar, por tanto, en añadir un tercer panel fotovoltaico, consiguiendo corregir asi este impedimento.

Por supuesto, estos cálculos son puramente teóricos, y no se corresponde con la realidad, en la cual los paneles fotovoltaicos son capaces de generar energía, aunque el sol se esté poniendo, o que la carga no consuma tanto. Se procede por tanto a hacer mediciones experimentales.

La primera medida ha sido el consumo de la carga y se ha observado que, aun con Jarvis funcionando y con los altavoces sonando, la corriente varía entre un mínimo de 0.21 amperios cuando no se está haciendo uso del sistema, hasta picos de 0.28 amperios cuando está funcionando y enviando señal a los altavoces. La tensión procedente del regulador de carga solar, en cambio, no son 12 voltios, sino 13, por lo que la energía consumida en una hora es de:

$$Energía\ consumida = 13\ V * 0.25\ A * 1h = 3.25\ Wh$$

Los paneles fotovoltaicos por otro lado también están proporcionando una energía completamente distinta a la teórica, llegando a producir en mejores horas del día hasta 17 V e incluso 0.7 A. Estos resultados son altamente beneficiosos, ya que en una hora:

$$Energía\ generada = 17\ V * (0.7 - 0.25)A * 1\ h = \ 7.65\ Wh$$

Se comprueba que, en la práctica, si las condiciones meteorológicas son buenas, el sistema funcionaría con energía sobrante, permitiendo asi descartar la posibilidad de fallo del sistema por deficiencia energética.

Cierto es que estas mediciones se han hecho durante uno de los mejores meses del año en cuanto a irradiación solar (agosto), por lo que no es descarado considerar su funcionamiento durante meses peores como diciembre o enero,

aunque teniendo en cuenta la dimensión de la batería, no presentaría un problema a priori.

2.4.2. Optimización de consumo de la Raspberry Pi

A pesar del dimensionado realizado, es ideal encontrar alguna manera de que el dispositivo sea capaz de consumir lo mínimo posible. Es por ello que se han estudiado diversas maneras para reducir el consumo de la Raspberry Pi, relacionadas con la desactivación de ciertos servicios que no son utilizados en el proyecto, o limitando la potencia.

Se visualizan tres posibilidades para la reducción de potencia: neutralización del HDMI, del Bluetooth y la reducción de relojes relacionados con la GPU.

La desactivación del puerto HDMI puede llevar a un ahorro energético de entre 25 y 30 mA. Su desactivación puede ser provocada por comandos, o programada para que se mantenga desactivado en el encendido del dispositivo:

```
/usr/bin/tvservice -o
```

Esta misma línea de código puede introducirse en “/etc/rc.local”, un programa que es el que se encarga de inicializar todos los procesos en el encendido del dispositivo. De esta manera se encarga de que desde el inicio del sistema no sea necesario desactivar los distintos servicios manualmente. Si se desea reactivar el puerto HDMI, solo basta con cambiar en el comando `-o` por `-p`.

La desactivación del Bluetooth es similar. No consume tanto como el puerto HDMI, pero ya que no es un servicio que se use en el proyecto no hay por qué tenerlo activado. El comando `rfkill` se encarga de ello.

```
rfkill block bluetooth
```

Se puede reactivar utilizando `unblock` en vez de `block`.

Por otro lado, el uso del servidor se puede reducir para así minimizar un poco más el consumo de energía y el calor disipado reduciendo los relojes de la GPU. Para ello es necesario modificar el archivo “config.txt” en la carpeta “/boot”. Este

archivo es leído por la GPU antes de que ARM GPU y Linux se inicialicen. En este archivo se pueden modificar diversas propiedades que van desde la memoria del sistema, pasando por cámara, audio, GPIOs y mucho más. En este caso se busca modificar la frecuencia de diversos procesos.

```
gpu_freq=150
h264_freq=25
isp_freq=25
v3d_freq=25
```

Cada una de esas líneas configura, en orden, la frecuencia de la GPU, la del bloque de vídeo de hardware, la del bloque de sensor de imágenes y la del bloque de modelado 3D. Sus valores de serie son 300, y reduciéndolas a 25 se minimizará el consumo total, sin perder rendimiento.

2.5. Control de rango

Funcionando el sistema a la intemperie, es correcto realizar un estudio de funcionamiento en un ambiente más adecuado para su situación, ya que toda la configuración y pruebas de funcionamiento han sido realizadas en el laboratorio, donde los niveles de ruido suelen ser mucho menores que al aire libre. Se ha observado que al sistema le cuesta apreciar las palabras y diferenciarlas cuando los niveles de ruido son ligeramente superiores. Esto se debe principalmente a la calidad del micrófono USB usado, que se ha comprobado que no es capaz de limpiar el ruido de fondo, y mezclarlo con la voz del usuario, provocando comandos erróneos o la incapacidad del sistema para reconocer la palabra mágica. A pesar de ello, para las pruebas se ha utilizado el micrófono del prototipo, y se ha condicionado para que funcionase lo mejor posible.

Por ello el primer paso ha sido instalar el sistema al aire libre:

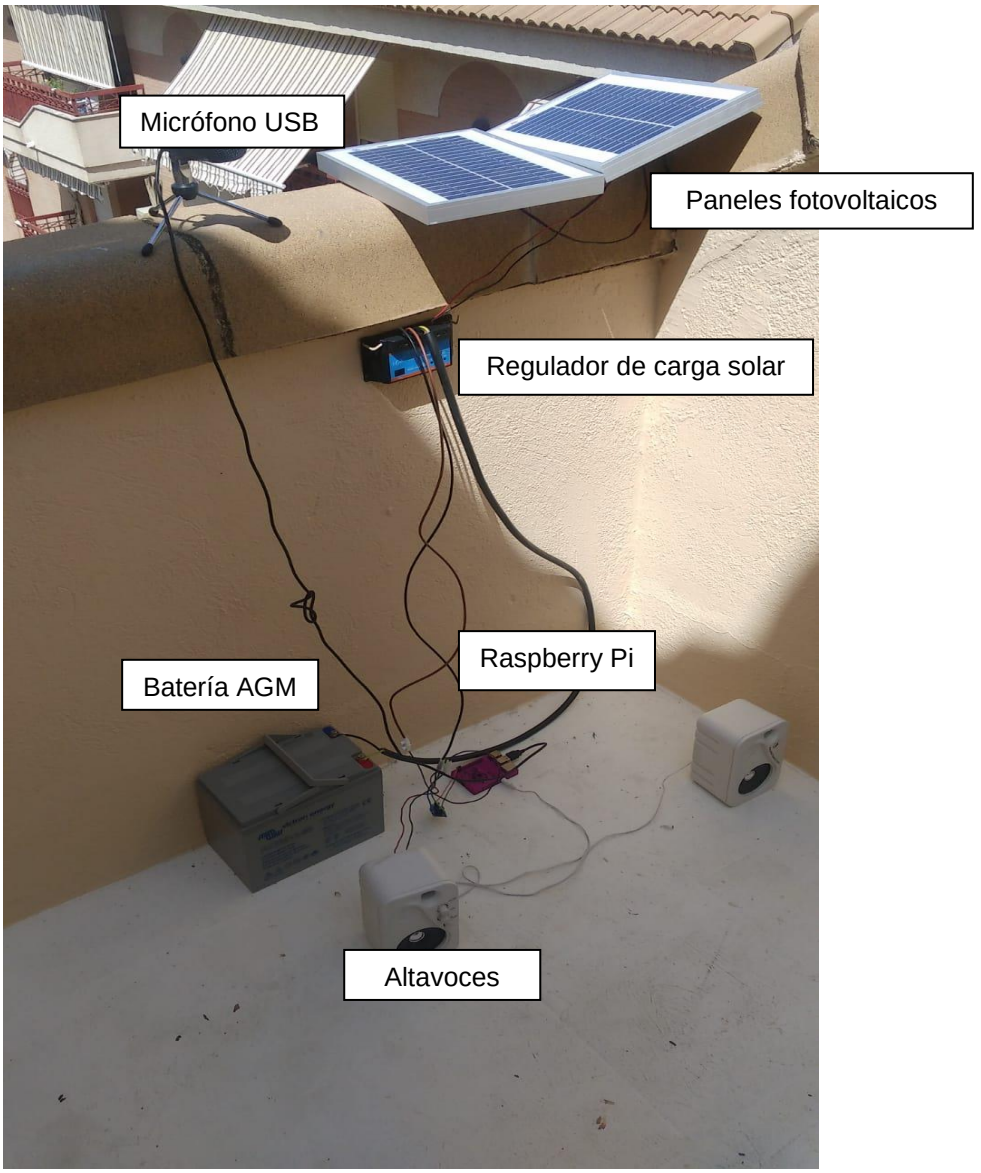


Ilustración 2.15 - Montaje físico del prototipo en una terraza

Esta instalación ha sido realizada no solo para un control de rango auditivo, sino también para analizar el balance energético.

Tras la correcta instalación se procedió a realizar el *Wizzard* de configuración básica de Jarvis, donde regulará automáticamente la ganancia del micrófono acorde con la voz y ruido ambiente.

Como se ha mencionado antes, el micrófono al no ser muy bueno se ha tenido que reducir la ganancia para intentar discernir solamente la voz, e intentar evitar cualquier ruido ambiental. Por supuesto esto también provoca la pérdida de calidad de las palabras escuchadas, llevando a equivocaciones provocadas por el motor

TTS que a veces no es capaz de reconocer ciertas palabras con fonemas poco discernibles. Las paredes y suelo del balcón tampoco ayudan, ya que provocan cierto eco que agrava las palabras.

A pesar de los diversos inconvenientes, se ha realizado el control de rango intentando aliviar cada uno de los problemas de la manera más eficaz posible.

Jarvis tiene un modo de funcionamiento que es "*Troubleshooting mode*", que ofrece la posibilidad de devolver por los altavoces la frase que se ha dicho, para volver a oír el mismo mensaje desde los oídos de la computadora y saber si se ha pronunciado bien, o si el ruido de fondo impide el entendimiento de la oración. Se ha hecho uso de este modo para las pruebas, y aunque el motor TTS de *Bing* es capaz de reconocer frases con un tono de voz normal y sin tener que pararse demasiado en la pronunciación, las pruebas se han realizado alargando y espaciando palabras, además de exagerando sus fonemas, para poder darle la mayor claridad posible.

En esta tabla se muestra el porcentaje de aciertos a la palabra clave "Olivia" con la que pasa de modo oyente a modo escucha:

Distancia (m)	Resultado
1	90 %
2	85 %
3	70 %
4	50 %

Tabla 2.5 - Porcentaje de aciertos en el control de rango

Por otro lado, el reconocimiento de las palabras para la realización de comandos se ve altamente afectado por la distancia y el ruido. Conforme el usuario se aleja del micrófono, las palabras se vuelven más irreconocibles para el motor TTS, por lo que la realización de respuestas no se desarrolla adecuadamente.

3. Presupuesto

El desarrollo del prototipo ha sido visualizado siempre desde el punto de vista económico. Teniendo en cuenta el estudio de optimización fotovoltaico se puede llegar a ciertas conclusiones de cómo podría construirse un modelo más barato, pero también hay aspectos legales los cuales hay que considerar si el prototipo se fabricará con la intención de ser vendido.

En la siguiente tabla se muestra el precio del prototipo usado:

Elemento	Cantidad	Precio
Raspberry Pi 3 model B	1	10.53 €
Micrófono TONOR	1	12.95 €
Altavoces (kit estéreo)	1	19.99 €
Panel RS Pro	2	34,44 €
Batería AGM 60 Ah	1	150 €
Regulador de carga solar	1	31.50 €
	TOTAL:	**259.41 €**

Tabla 3.1 - Presupuesto neto del prototipo

En el caso del hardware, es bastante directo. En el punto 3 se dimensionó el sistema y se llegó a una conclusión en cuanto al precio de ciertos componentes. En cada prototipo son usados dos paneles fotovoltaicos RS. En cambio, cabe la posibilidad de hacer uso de Raspberry Pi Zero W, un modelo más compacto y eficiente que no solo tiene ahorro energético sino también económico. Los altavoces utilizados en el proyecto son unos que no han sido necesario comprar, pero altavoces de sobremesa siguen vendiéndose en el mercado, y su precio es asequible para un proyecto como este.

En el dimensionado fotovoltaico se habló de usar una batería de tamaño más reducido, ya que 60 Ah es una solución exagerada. Una batería de AGM de Victron Energy de 38 Ah supondría un ahorro de entre 30 y 35€.

El micrófono en cambio es una inversión a considerar. En el proyecto se ha utilizado un micrófono USB barato que ha permitido una configuración y testeo del prototipo. Pero como se ha mencionado antes, este micrófono no ayuda en la elaboración de un prototipo adecuado para su uso nominal. El uso de un micrófono mejor supondría un coste mucho mas superior.

En cuanto a los aspectos legales mencionados a principio de este capítulo, hay que considerar que las API Keys de Bing no son gratis. Con cada cuenta nueva creada, *Microsoft Azure* te da la oportunidad de hacer uso de sus API Keys de manera gratuita durante un tiempo limitado de 7 dias y posteriormente es necesario pagar una tasa. Cada 1000 transacciones (usos de la API Key) hay que pagar 3.374 €. Esto ocurre igual con Google, aunque la API de Google se mide en tiempo de uso, no en número de transacciones (cada 15 segundos de uso cuesta 0.006 $, pero los segundos hablados siempre se redondean a 15 si han sido menos, por lo que cada uso costaría realmente 0,006 $ siempre). Es por ello que, aunque parezca poco, si el sistema fuese implementado en un museo, su uso sería extremadamente alto, llevando los costes hasta miles de euros.

Es posible realizar una estimación de lo que puede costar al mes el uso continuado de Olivia con la tarifa de Bing, que es la que se utiliza: se considera un museo con 600 visitantes al dia. Si se toma que de media cada persona hace 5 preguntas a Olivia, eso son 5 transacciones por persona, lo que quiere decir alrededor de 3000 transacciones al día.

3000 transacciones equivalen a 10.122 € al día, por 30 días que tiene el mes, sale por unos 303.66 € al mes. Por supuesto esto es una pura estimación y puede variar de museo en museo, por ejemplo, en el Parque de las Ciencias de Granada saldría mucho más caro, ya que en 2017 tuvo un record histórico de más de 750,000 visitantes.

El coste estimado de Software es:

Elemento	€/transacción	Coste mensual (estimado)
Licencia *API Speech Bing*	0.00374	304 €/mes

Tabla 3.2 - Tabla de coste Software

4. Resultados

El proyecto per se supone una atracción de encanto para la gente, y no es impropio pensar que el sistema es por tanto un éxito, no como un prototipo funcional, sino como una herramienta con el potencial de atraer un interés ajeno por aquello que rodea la provincia de Jaén: el olivo. A lo largo del desarrollo del proyecto durante estos meses, el proyecto ha recibido una atención, podría decirse incluso mediática, por el gran atractivo que supone, una percepción completamente distinta de la que suele tener el ciudadano de a pie sobre el resto de proyectos desarrollados en el GRAV.

La búsqueda del software adecuado para este proyecto supuso durante un tiempo una lacra debido a la falta de información y conocimiento, pero el trabajo se ha visto recompensado tras semanas de investigación, y la solución encontrada ha supuesto vital para el cumplimiento del proyecto. Jarvis ahora forma parte del abanico de opciones a elegir como una herramienta reconocida por el equipo del GRAV para futuros proyectos relacionados con el reconocimiento de voz. Aparte, todos los conocimientos obtenidos relacionados con el reconocimiento de voz, los motores TTS y STT, su funcionamiento, etcétera, son un saber que resultará útil ya no solo para proyectos asignados al GRAV, sino también para proyectos personales que ayudarán a cumplimentar la vida del día a día o incluso participarán en el ámbito laboral de cada uno.

Por otro lado, puesto que el proyecto se ha implementado en un prototipo, el listado de comandos no es tan voluminoso como sería el producto final. A pesar de ello, en el prototipo se ha configurado un amplio abanico de comandos con los

cuales se puede apreciar el correcto funcionamiento del sistema. Todos ellos se encargan de funciones similares, pero específicas, y pueden ser realizados por cualquiera ya que también se ha comprobado la efectividad del motor TTS con diferentes tonalidades de voces, e incluso con diversas personas.

En cuanto al Hardware es pertinente mencionar las diversas limitaciones que ha ocasionado, o que puede ocasionar. Tras el estudio del sistema fotovoltaico no sería mezquino en pensar cómo funcionaría en el mes peor del año. Es posible casi asegurar que el sistema tendrá la suficiente autonomía, y que al no estar funcionando en su máximo constantemente es posible que durante días de penumbra podrá sobrevivir. Considerando que el sistema sería instalado en el sur de España, se dispone de la tranquilidad para pensarlo, pero si el sistema se instalase en una ciudad más lluviosa u oscura, como puede ser en el norte, cabe la posibilidad de que sea necesaria una tercera placa fotovoltaica para poder compensar las pérdidas.

Por otro lado, el micrófono necesita ser mejorado. Se ha comprobado que es necesario una inversión extra para poder incorporar un micrófono de mejor calidad, que sea capaz de eliminar el ruido de fondo y que tenga buen alcance, ya que con el micrófono utilizado se han observado problemas de entendimiento de palabras relacionados con el alcance del locutor, y con el ruido ambiente. Un micrófono omnidireccional de calidad puede resolver este problema. El desarrollador de Jarvis propone un catálogo de soluciones para diversos requisitos. La opción más barata es un micrófono de la marca *Jabra Speak*, omnidireccional y de elevada potencia. Cuenta también con un puerto Jack de 3.5 mm donde se pueden conectar los altavoces externos y por tanto permitiría hacer uso en un modelo deRaspberry: la Raspberry Pi Zero W, que a pesar de ser menos potente que la utilizada, consume menos y es más barata. El precio del micrófono es relativamente caro pero su uso significaría un sistema mucho más efectivo.

Los altavoces han resultado ser una excelente solución. El condicionamiento de estos para el proyecto es sencillo y funcionan a la perfección debido a la elevada potencia que poseen.

5. Conclusiones y trabajos futuros

Se puede concluir que el programa ha sido exitoso, porque se han llegado a cumplir todos sus objetivos. Consigue ofrecerte cierto nivel de conversación, dándote a la vez información de los datos recogidos por los sensores. Los algoritmos creados funcionan correctamente, y la conexión con la Nube es precisa.

El único punto que no se ha llegado a trabajar al final es la creación de un encapsulado para evitar el deterioro del sistema por encontrarse a la intemperie. La creación de un prototipo completamente final y funcional hubiera culminado si se hubiera construido en el olivo, junto a sus diversos sensores que recogen la información. En el laboratorio del GRAV disponemos de los diversos materiales que hubieran hecho falta para poder crear una caja contenedora, que se hubiera condicionado para tener la hermeticidad para evitar el agua, y la claridad para el correcto funcionamiento del micrófono y de los altavoces. La implementación de los paneles es la menos discreta, ya que estos han de ser asomados por encima del olivo para poder captar la mayor cantidad de irradiación solar, además de que habría más de dos paneles, ya que en el sistema hay que alimentar más complementos aparte de Olivia.

En la siguiente foto se muestra uno de los montajes donde se realizará la instalación de Olivia:

Ilustración 5.1 – Foto de uno de los montajes realizados por ISR (cerca)

Ilustración 5.2 - Foto de uno de los montajes realizados por ISR (lejos)

Aun así, el sistema ha demostrado ser eficaz, y su implementación en un sistema real sólo supondría un coste de tiempo. Los resultados del proyecto son positivos y nos demuestra la esperanza de poder usar esta tecnología en el futuro.

La implementación del reconocimiento de voz ha sido en el proyecto completamente primordial, y es una tecnología que se podría barajar más en futuros proyectos. Por ejemplo, la creación de un sistema como el *Marathi Interactive Voice*

Response System, donde los agricultores podrían hacer uso de una terminal con reconocimiento de voz, desde sus casas o desde su puesto personal de trabajo. Y no solo en el ámbito agrónomo, esta tecnología también se puede utilizar en proyectos de automoción, en los cuales se puede hacer uso de comandos de voz para tratar con maquinaria pesada, o para la obtención de datos sin depender de un monitor o una interfaz visual.

En conclusión, el prototipo ha supuesto un inicio en el reconocimiento de voz, y se ha demostrado su utilidad y funcionalidad de manera satisfactoria. Es una tecnología con mucho potencial, y su uso ha resultado altamente gratificante. Se espera con ganas poder ser usado en el futuro con más proyectos.

6. Referencias

[1] Acapela. (28 de Febrero de 2014). *How does it work?* Obtenido de Acapela Group: http://www.acapela-group.com/voices/how-does-it-work/

[2] Ben, E., Graham, M., Andrew, G., Les, P., Russel, B., & John, L. (s.f.). *Linux Voice Raspberry Pi Anthology: The best Raspberry Pi articles from Linux Voice magazine.* Linux Voice.

[3] BitSophia Tecnologia. (s.f.). *BitVoicer Server*. Obtenido de BitVoicer & BitVoicer Server: http://www.bitsophia.com/en-US/BitVoicerServer/Overview.aspx?source=BitVoicer

[4] *Crean un software que permite dibujar con la voz*. (20 de Octubre de 2014). Obtenido de Tendencias21: https://www.tendencias21.net/Crean-un-software-que-permite-dibujar-con-la-voz_a37953.html

[5] Grabianowski, E. (s.f.). *How Speech Recognition Works*. Obtenido de howstuffworks: https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition.htm

[6] ISR. (2016). *Integración Sensorial y Robótica*. Obtenido de Pagina web de ISR: http://isr.es/

[7] Jasper. (s.f.). *Jasper - Home*. Obtenido de Jasper Project: https://jasperproject.github.io/

[8] Knight, W. (30 de Mayo de 2012). *Business Impact: El futuro del reconocimiento de voz*. Obtenido de MIT Technology Review: https://www.technologyreview.es/s/2756/business-impact-el-futuro-del-reconocimiento-de-voz

[9] Lee, A., Kawahara, T., & Kiyoshiro, S. (2001). Julius - an Open Source Real-Time Large Vocabulary Recognition Engine. *Nais Academic Repository*.

[10] Mély, A. (s.f.). *OpenJarvis - Présentation*. Obtenido de OpenJarvis: https://openjarvis.com/

[11] Moya, P. (29 de Febrero de 2016). *Todo sobre la nueva Raspberry Pi 3, más potente y conectada que nunca*. Obtenido de El Español - Omnicrono: https://omicrono.elespanol.com/2016/02/raspberry-pi-3-model-b/º

[12] Pinola, M. (2 de Noviembre de 2011). *Speech Recognition Through the Decades: How We Ended Up With Siri*. Obtenido de PCWorld: https://www.pcworld.com/article/243060/speech_recognition_through_the_decades_how_we_ended_up_with_siri.html

[13] Ram Baheti, M., W. Gawali, B., & Mehrotra, S. (2015). Marathi Interactive Voice Response System (IVRS) using. *International Journal of Computer Applications*, 975-8887.

[14] RS Pro. (s.f.). *MONO & POLY-CRYSTALLINE (12V) Silicone solar cell modules.* Obtenido de RS: https://docs-emea.rs-online.com/webdocs/1587/0900766b815873a3.pdf

[15] *Sitio web oficial del Mercado de Futuros del Aceite de Oliva* . (s.f.). Obtenido de MFAO: http://www.mfao.es/

[16] Victron Energy. (2 de Mayo de 2017). *BlueSolar PWM Light Charge Controller User Manual.* Obtenido de Victron Energy: https://www.victronenergy.com/upload/documents/Manual-BlueSolar-PWM-Light-12-24V-Charge-Controller-EN-A4.pdf

[17] Victron Energy. (s.f.). *Baterias Gel y AGM datasheet.* Obtenido de Victron Energy: https://www.victronenergy.com.es/upload/documents/Datasheet-GEL-and-AGM-Batteries-ES.pdf