# Proyecto final de Ingeniería Electrónica

## Control de luces mediante comandos de voz

**Autor:** Casal, Ricardo (50226)

**Tutores:** Nemirovsky, Nicolás
Orchessi, Walter
Pingitore, Ricardo
Ugarte, Alejandro

**Fecha:** 08/12/2017

# Índice

# Índice de figuras

# Índice de cuadros

# Nomenclatura

**Módulo base o central** Módulo master donde se procesa la voz e indica las acciones a realizar

**Módulo terminal** Módulo esclavo con un relay al cual se conecta la luz a controlar

**Sistema o kit básico** Combinación de un módulo base y dos terminales

**GPIO** General Purpouse Input/Output

**UART** Universal asynchronous receiver-transmitter

**SPI** Serial Peripheral Interface

**I2C** Inter-Integrated Circuit

**SW** Software

**HW** Hardware

# Resumen

Con el avance de la tecnología, los seres humanos buscan facilitar todo tipo de tareas. Una parte importante de esa búsqueda trae como consecuencia la automatización de diversas comodidades del hogar en busca de confort. El control de iluminación mediante comandos de voz es un primer paso hacia un hogar automatizado.

El control requiere de periféricos (micrófonos) capaces de captar las señales de voz humana para su procesamiento. Dicho procesamiento debe ser realizado en tiempo real, lo cual implica la necesidad de una plataforma procesando constantemente la información de los periféricos, tomando decisiones en base a ello y controlando los actuadores. Dichos actuadores deben estar en cons-tante comunicación con la plataforma mencionada. Además, un requerimiento ineludible es el de la escalabilidad vertical (darle nuevas funcionalidades a un sistema ya instalado) y horizontal (agregar componentes a un sistema ya instalado).

El producto propone una primera solución a la búsqueda de confort del usuario, integrando la adquisición y procesamiento de voz y los actuadores mediante un desarrollo que permitiría una evolución sin la necesidad de crear un producto de cero.

# Parte I

# Introducción

## 1. Historia

A comienzos de la década de 1990, surge un gran interés en la domótica en Argentina. Empiezan a surgir algunas aplicaciones parciales, presentaciones en ferias y un gran interés periodístico en las nuevas aplicaciones desarrolladas. Durante esos años, se va incorporando cada vez más el uso de instalaciones de este tipo y nuevas compañías empiezan a adecuar el concepto a sus servicios e incluso a desarrollar nuevas aplicaciones. Este desarrollo decrece durante la crisis del 2001 y resurge paulatinamente hasta que en 2007 se realiza en el país la primera exposición exclusiva de domótica llamada “expo casa domótica” y el primer congreso de domótica.

Luego se crea una comisión de ingenieros especialistas en la provincia de Córdoba, quienes elaboran una guía de contenidos mínimos para la elaboración de un proyecto de domótica. Actualmente, la Comisión de Domótica del Colegio de Ingenieros Especialistas de Córdoba (CIEC) concentra a los profesionales del tema y es uno de los principales reductos dedicados a velar específicamente por la calidad de los servicios que se prestan en el país.

## 2. Justificación del proyecto.

Se realizó una encuesta a través de internet, con la ayuda de la aplicación de formularios de google, para adaptar el proyecto a las necesidades o requisitos que se consideran importantes para el producto. El link de la encuesta fue compartido en facebook. Participaron 200 personas aproximadamente de todas las edades, en donde la mayoría se encuentran entre los 21 y 30 años. Se generó un histograma con los datos obtenidos sobre los tiempos en que la gente está despierta y en actividad en su casa y se observa que la mayoría está entre cuatro y seis horas. Por lo tanto, este producto facilitaría tareas diarias durante ese tiempo y allí se notaría su comodidad rápidamente.

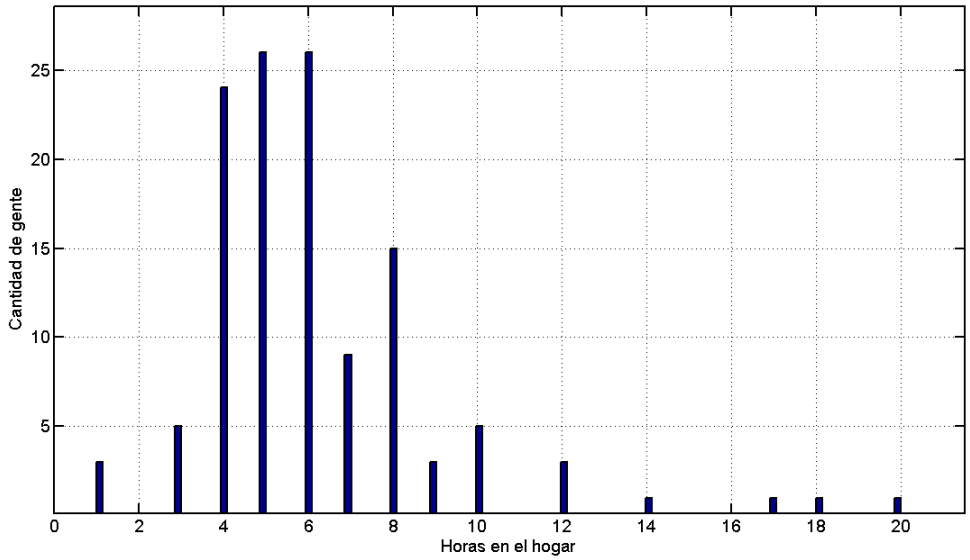



Figura 1: Histograma tiempos en el hogar

La primera pregunta relevante es si el cliente cuenta con un sistema de control autónomo en el hogar. Como se puede ver en el siguiente gráfico, la mayoría no posee ningún sistema de domótica pero tienen interés en el mismo.

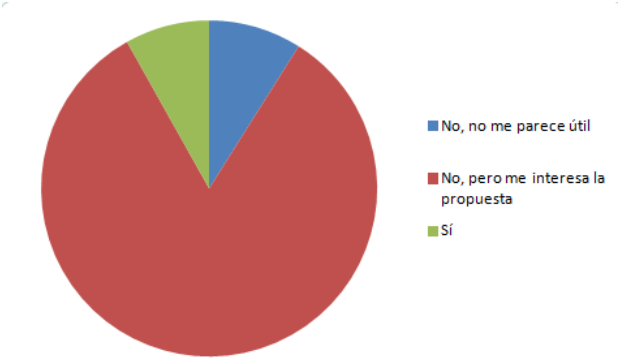



Figura 2: ¿Cuenta usted con algún sistema de control autónomo en el hogar?

En cuanto a preferencia sobre quien puede controlar el sistema, la mayoría prefirió darle acceso a personas individualmente, sin embargo definir el producto de esta manera significaría tener que entrenar el sistema a reconocer las voces requeridas por el usuario. Para esta tarea, el usuario debería proveer varias grabaciones de los comandos requeridos previamente al entrenamiento del sistema de reconocimiento de voz. Debido a que se quiere mantener simple la instalación del equipo se optará por hacer un algoritmo de detección genérico que pueda reconocer a varios tipos de voces. Esta elección permite un desarrollo de un producto único, en el cual se entrena el sistema una sola vez, acelerando el proceso de producción. Si bien se busca simplificar este aspecto, igualmente se va a considerar que el producto pueda escalar y adaptarse a un reconocimiento personal. El marketing del producto tendrá énfasis en su escalabilidad vertical ya que es importante desde la perspectiva del cliente que así sea.

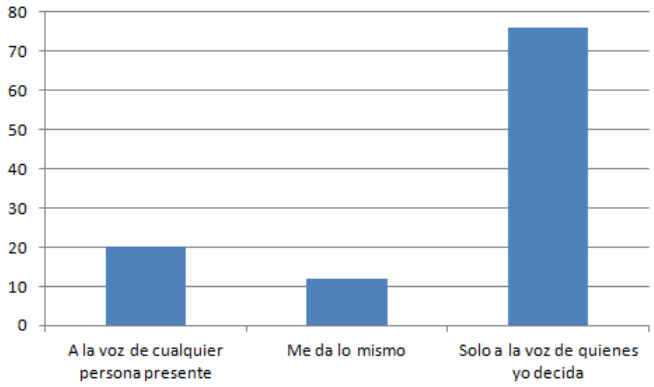



Figura 3: ¿Desea que el sistema responda a su voz, o a la de cualquier persona presente?

El propósito de la próxima pregunta era de definir el hardware a usar con relación de la obtención de la señal. Se eligió utilizar micrófonos colocados en el ambiente debido a que fue la opción más popular. Sin embargo, este tipo de proyecto deberá permitir adaptar el control al celular simultáneamente con los micrófonos, ya que esta segunda opción fue también popular.

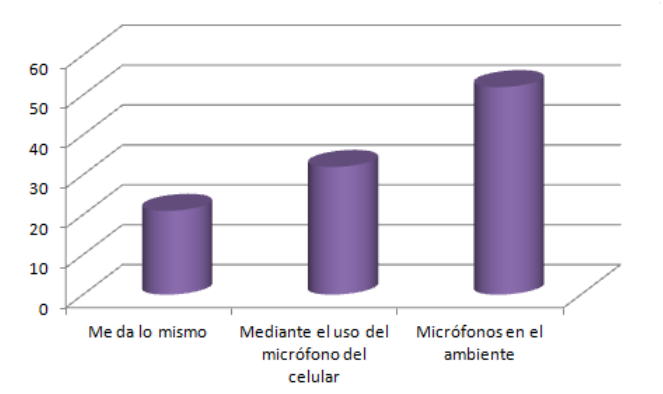



Figura 4: Sistema de reconocimiento de voz

Luego se preguntó si parecería conveniente poder crear comandos propios y pareció ser ampliamente aceptado por la mayoría. Al poder crear comandos particulares, el equipo podría tener una característica más personal. Nuevamente, esto será considerado en la escalabilidad del producto, es decir que el sistema propuesto deberá poder agregar o quitar en un futuro comandos sin modificar el software del producto.

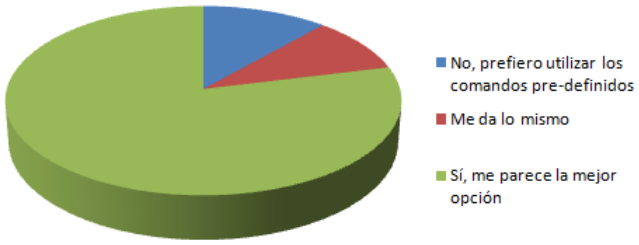



Figura 5: ¿Le parece conveniente poder crear sus propios comandos?

Por último se pidió dar una estimación monetaria del precio máximo por el que están dispuestos a pagar por el sistema. En principio, se decidió tomar un valor entre U$200 y U$500 para adaptar la oferta del sistema a la demanda del mismo. La decisión final se realizará al analizar costos y ganancias, pero el precio objetivo entrará dentro de este rango, teniendo en cuenta no afectar la factibilidad económica.

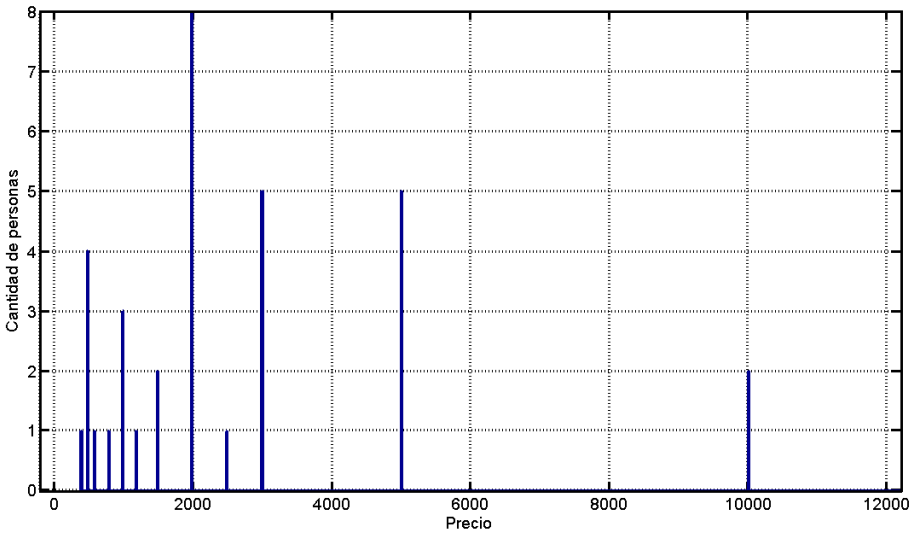



Figura 6: Histograma de precios

# Parte II

# Objetivos

## 2.1. Finalidad del proyecto

El objetivo final del proyecto es desarrollar una plataforma electrónica que integre los campos del procesamiento de señales y la domótica en un único producto orientado al usuario común. Se busca que la plataforma sea el primer paso hacia un hogar completamente autónomo y controlable mediante herramientas tecnológicas modernas. Al mismo tiempo, se busca que mediante características como la facilidad de instalación y uso, el usuario común se vea interesado en el mismo. De ese modo, se obtiene el "feedback" necesario para futuras generaciones del sistema.

## 2.2. Planteo del problema a resolver

El producto proveerá al usuario una solución que incorpora confort en su hogar. Permitirá controlar luces del ambiente sin la necesidad de tener que acercarse físicamente a las mismas, especialmente para las luces en donde el switch está ubicado en un lugar incomodo o difícil de acceder. Si bien el proyecto se limita al uso de luces, es importante considerar que esta idea de confort pueda evolucionar a controlar nuevos dispositivos.

## Parte III

# Definición de producto

## 3. Requerimientos

Los requerimientos representan aspectos importantes a ser considerados a lo largo del proyecto, viéndose reflejados en la definición del producto y finalmente validados luego de la creación de un prototipo. Algunos son tácitos y otros surgen de características propias del consumidor o cliente del producto, factores legales y metas de mercado.

| ID | Descripión | Origen |
|---|---|---|
| REQ-01 | Confialabilidad del sistema* | Tácito |
| REQ-02 | Tiempo que tarda el sistema en realizar una acción menor a 3 segundos | Tácito |
| REQ-03 | Facilidad de uso: No se requieren conocimientos técnicos previos | Cliente |
| REQ-04 | Facillidad de instalación: Debe ser fácil de preparar/conectar | Cliente |
| REQ-05 | Debe ser estética aceptable para el público | Tácito |
| REQ-06 | El precio debe ser menor que 1000 dólares | Cliente |
| REQ-07 | Debe consumir menos que 10 W por módulo y 80W para la luz | Tácito |
| REQ-08 | Debe poder utilizarse en el interior de un hogar con condiciones ambientales normales | Tácito |
| REQ-09 | Escalabilidad horizontal: Debe permitir su expansión para el control de una mayor cantidad de luces | Cliente |
| REQ-10 | Escalabilidad vertical: Debe permitir su expansión para otro tipo de controles y perifericos dentro del hogar | Mercado |
| REQ-11 | Disipación: No debe generar temperaturas peligrosas mayor a 40 grados centigrados | Legal |
| REQ-12 | Debe cumplir con la resolución 0207/2017 de la SECRETARÍA DE COMERCIO del MINISTERIO DE PRODUCCIÓN conteniendo los aspectos necesarios de seguridad eléctrica | Legal |

Cuadro 1: Requerimientos

REQ-01 es uno uno de los requerimientos más importante, el producto debe cumplir su función de manera adecuada. Lamentablemente, debido a la naturaleza del producto, el reconocimiento de voz no podrá tener cero fallas. En consecuencia, se debe buscar minimizar las fallas por reconocimiento de voz independientemente de las fallas eléctricas o de comunicación entre módulos.

Con respecto al reconocimiento de voz, se definirán tres tipos de fallas posibles:

- No detección: Esta falla ocurre cuando el usuario emite un comando pero no hay reconocimiento de ningún tipo. En otras palabras, el comando fue ignorado.
- Falsa detección: Detección de un comando cuando el usuario no tiene la intención de realizar una acción. Agregando un threshold de ruido ayuda a mitigar este efecto para situaciones particulares en donde hay ruido con distribución uniforme.
- Detección errónea: Detección distinta (o errónea) cuando el usuario quiere realizar otra acción. Esta se puede minimizar utilizando un vocabulario restringido a las palabras a utilizar y evitar el uso de palabras prosódicamente similares.

En la práctica, existe una relación de compromiso entre falsas detecciones y detecciones erróneas. La meta será encontrar un balance entre ellas en el cual se minimicen las falsas detecciones pero no presente un gran porcentaje de detecciones erróneas. La razón detrás de esta decisión es que es preferible repetir o corregir un comando a tener los módulos prendiendo o apagándose aleatoriamente.

Las fallas de comunicación pueden ocurrir independientemente del reconocimiento de voz. Es importante que la comunicación tenga redundancia y pueda verificar que los mensajes lleguen a destino.

Las fallas eléctricas pueden ocurrir por fallas en la red eléctrica previa a los circuitos o fallas de los circuitos debido a temperatura o ruido (eléctrico) en el ambiente.

REQ-09 y REQ-10 surgen de analizar la figura 2. En ella, se detecta un gran grupo de gente que no posee actualmente un sistema de domótica. Debido a que el producto podría ser la primer experiencia del cliente con un sistema de domótica se requiere que el usuario no deba tener conocimientos técnicos. Desde el punto de vista del cliente, debe conectar el sistema a la red eléctrica y el mismo debe funcionar al igual que enchufar o desenchufar un módulo terminal no debe afectar el sistema.

REQ-06 surge al tener en cuenta el precio establecido al analizar las encuestas en la figura 6, es probable que haya que tomar decisiones para limitar el costo del producto para que la producción del sistema sea viable.

REQ-10 se establece para que el sistema pueda evolucionar a través del feedback de los usuarios. Una meta en el diseño será que en futuras modificaciones exista compatibilidad hacia atrás. Esto debe contemplar reconocimiento diferenciado de voces, comandos personalizados, posibles migraciones de reconocimiento de voz al teléfono celular y hasta incluso aplicaciones vía Internet.

# 4. Diagrama funcional de Interfaces

El sistema se define con dos tipos de módulos. Ambos tendrán incluídos un transformador 220/5V para su alimentación.

- Módulo base o central. Este debe contener la electrónica necesaria para realizar el reconocimiento y procesamiento de voz, tomar las decisiones y comunicársela a los actuadores.
    - Un sistema embebido con la capacidad de procesamiento suficiente para realizar estas actividades en tiempo real.
    - Comunicación: Debe ser capaz de mantener comunicaciones inalámbricas con los diferentes módulos terminales.
    - En lo que a sensores respecta, debe contar con micrófono/s para capturar la señal de voz. Los mismos deben contemplar el acondicionamiento necesario para su procesamiento.
- Módulos terminales. Cada módulo terminal será el encargado de controlar las luces del ambiente. Debe contar con la capacidad de recibir instrucciones, interpretarlas y realizar las acciones requeridas.

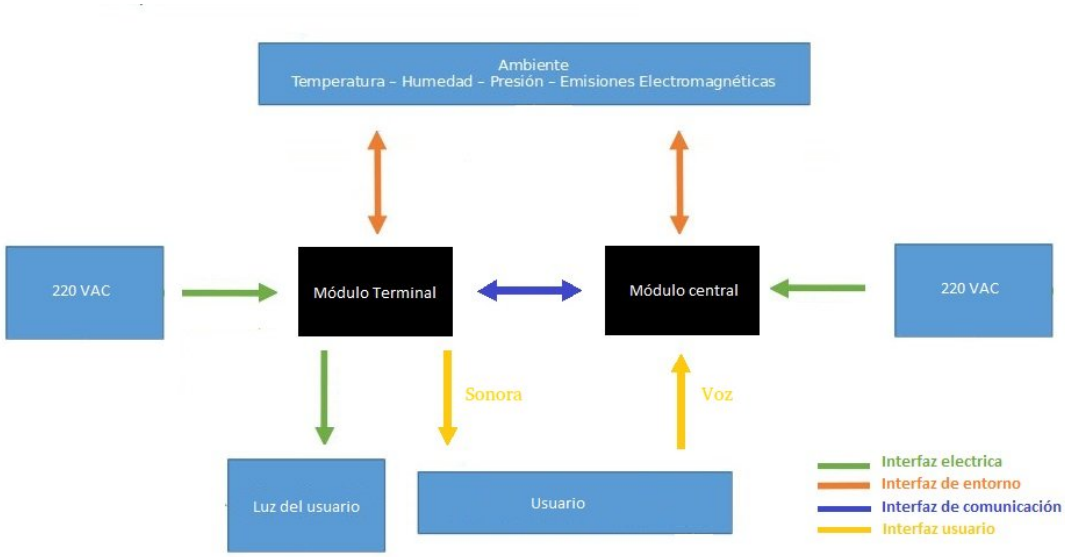



Figura 7: Diagrama de interfaces

# 5. Especificaciones

Teniendo en cuenta los requerimientos anteriores se definen especificaciones que el producto final debe cumplir. El primer grupo de estas especificaciones serán funcionales.

| ID | Descripión | Origen |
|---|---|---|
| FUN-01 | Tasa de errores: Falsa detección < 1 por hora | REQ-01 |
| FUN-02 | Tasa de errores: Detección erronea < 30% | REQ-01 |
| FUN-03 | Tasa de errores: No detección < 40% | REQ-01 |
| FUN-04 | Delay en detección del comando de voz < 3 segundos | REQ-02 |
| FUN-05 | Modularidad: Módulos con funcionalidad independiente | REQ-03, 09 y 10 |
| FUN-06 | Modularidad: Los programas deben andar al conectar los módulos | REQ-03 |
| FUN-07 | Consumo módulo central < 10W | REQ-07 y 11 |
| FUN-08 | Consumo luz módulo terminal < 80 W | REQ-07 |
| FUN-09 | Consumo microcontrolador módulo terminal < 3 W | REQ-07 y 11 |

Cuadro 2: Especificaciones funcionales

FUN-01, FUN-02 y FUN-03 son las tasas de errores surgidas en el requisito REQ-01. Debido a la naturaleza de las fallas FUN-02 y FUN-03 se calcularán como tasa errores sobre el total mientras que FUN-01 se define como fallas por unidad de tiempo. Adicionalmente, se establece 3 segundos como el límite para el tiempo de detección del comando de voz.

FUN-05 tiene su originen en tres requisitos. Por un lado, si cada módulo funciona independientemente se evita la posibilidad de fallas por encenderlos en distintos orden. Por otro lado, si cada módulo es completamente independiente da lugar a que el producto pueda escalar verticalmente con nuevas versiones, manteniendo compatibilidad hacia atrás, y horizontalmente sin la necesidad de cambiar parte del sistema. FUN-06 complementa esto último y presenta la posibilidad que el módulo se pueda mover dentro del ambiente sin generar preocupación de que el mismo se desconfigure. La especificación FUN-08 representa un límite en la potencia de la luz a controlar, teniendo

en cuenta que la gran mayoría de luces en un hogar consumen menos que 80 Watts. FUN-07 y FUN-09 establecen limites de potencia para los módulos, considerando que la mayoría del procesamiento será realizado en el módulo central.

El siguiente grupo de especificaciones definen la interfaz con el usuario:

| Interfaz con usuario | | |
|---|---|---|
| **ID** | **Descripión** | **Origen** |
| INT-US-01 | Indicador de acción sonoro: Switch mecánico | REQ-03 |
| INT-US-02 | Protección de tensión peligrosas en módulo central: gabinete con espesor > 1mm | REQ-12 |
| INT-US-03 | Protección de tensión peligrosas en módulo central: Zocalo de conección | REQ-12 |
| INT-US-04 | Alcance del micrófono > 10 m | REQ-01 y 08 |

Cuadro 3: Interfaz con el usuario

INT-US-01 es necesario para que el usuario tenga un feedback de que su acción se realizó a pesar de que se verá el cambio de la luz. Para el caso particular en el cual la luz no funcione correctamente la indicación sonora pondrá en evidencia que el problema no se encuentra en el módulo terminal. INT-US-04 se establece menor que 10 m ya que se considera una distancia razonable para un ambiente doméstico.

Se procede a definir las especificaciones propias de la comunicación entre módulos.

| Interfaz comunicación | | |
|---|---|---|
| **ID** | **Descripión** | **Origen** |
| INT-COM-01 | Interfaz inalámbrica entre módulos | REQ-03, 04 y 09 |
| INT-COM-02 | Comunicación con redundancia | REQ-01 |
| INT-COM-03 | Delay de transmición < 5 segundos | REQ-02 |
| INT-COM-04 | Alcance de comnunicación para ambos módulos > 15 m | REQ-01, 02 y 08 |

Cuadro 4: Interfaz comunicación

Se establece que la comunicación entre los módulos debe ser inalámbrica, no solo debido a la escalabilidad horizontal, sino también a la facilidad de uso e instalación. No se le puede exigir al usuario que cree un sistema de cables para utilizar el producto. INT-COM-03 establece un límite en el tiempo de retardo en la comunicación y INT-COM-04 establece el alcance de comunicación para ambos módulos en 15 metros.

| Interfaz alimentación | | |
|---|---|---|
| **ID** | **Descripión** | **Origen** |
| INT-VIN-01 | Módulo central: Transformador 220VAC/5VDC | REQ-08,12 |
| INT-VIN-02 | Módulo terminal: Transformador 220VAC/5VDC | REQ-08,12 |
| INT-VIN-03 | Módulo terminal: 220 VAC para la luz | REQ-08 |

Cuadro 5: Interfaz de alimentación

INT-VIN-01 y INT-VIN-02 presentan un rango de tensión típico para alimentar un microcontrolador. INT-VIN-02 y INT-VIN-03 son dos alimentaciones separadas, elección establecida para acotar el proyecto.

| Interfaz de salida | | |
|---|---|---|
| **ID** | **Descripión** | **Origen** |
| INT-VOU-01 | Módulo terminal: Resistencia de salida < 0.5 Ohm | REQ-08 |

Figura 8: Interfaz de salida

| Interfaz ambiental | | |
|---|---|---|
| ID | Descripión | Origen |
| INT-AMB-01 | Operar en temperaturas entre -20 a 40 grados centigrados | REQ-08 |
| INT-AMB-02 | Operar en humedad relativa entre 1 y 100% | REQ-08 |
| INT-AMB-03 | SNR > 20 dB | REQ-08 |

Cuadro 6: Interfaz ambiental

No se utilizará ninguna protección ante liquidos, por lo que el usuario deberá evitar ubicar el equipo en un lugar expuesto. Teniendo en cuenta la especificación INT-US-02, la cual define un grosor de gabinete mayor a 1 mm, y que no habrá protección ante liquidos, el mínimo requisito para el número de IP (Ingress Protection) según el estándar IEC 60529 será 40. El primer número hace referencia al grosor de la protección mientras que el segundo hace refetencia a la protección ante liquidos. Durante la etapa de diseño se analizará utilizar un grosor mayor.

INT-AMB-03 se establece la mínima relación señal a ruido en 20 dB, lo que significa que la señal debe ser 100 veces mayor que el piso de ruido.

| Dimension y peso | | |
|---|---|---|
| ID | Descripión | Origen |
| DYP-01 | Dimensión máxima módulo central: 15x15x5 cm | REQ-05 |
| DYP-02 | Dimensión máxima módulo terminal: 15x10x5 cm | REQ-05 |
| DYP-03 | Peso módulo central < 2kg | REQ-03 |
| DYP-04 | Peso módulo terminal < 2kg | REQ-03 |

Cuadro 7: Especificaciones de dimensión y peso

| Economica | | |
|---|---|---|
| ID | Descripión | Origen |
| ECO-01 | Precio del sistema menor a 1000 dólares | REQ-06 |

Cuadro 8: Especificación económica

| Confialabilidad | | |
|---|---|---|
| ID | Descripión | Origen |
| CONF-01 | MTBF < 1/año | REQ-01 |
| CONF-02 | Vida útil del producto < 5 años | REQ-10 |

Cuadro 9: Especificación de confiabilidad

# 6. Casa de calidad

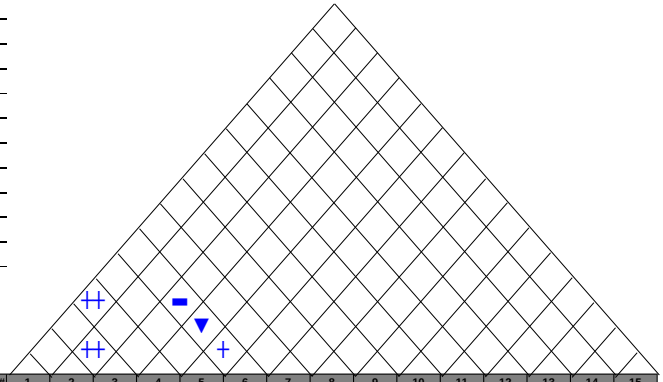
El análisis de la Casa de Calidad permite encontrar las relaciones entre el producto deseado contra las capacidades del mismo. Se comparan dos empresas que producen un sistema similar. La ventaja que se desea respecto a otras empresas es la de tener una gran escalabilidad. A medida que el usuario note que el producto puede escalar fácilmente se espera que genere una sensación de confianza y fidelidad al mismo. Se consideran las siguientes cualidades para el análisis: Calidad de reconocimiento, disipación, conectividad inálambrica, capacidad de procesamiento, indicación de acción, costo, modularidad y consumo.

Title: _____

Author: _____

Date: _____

Notes: _____



| Legend | | |
|---|---|---|
| Θ | Strong Relationship | 9 |
| O | Moderate Relationship | 3 |
| ▲ | Weak Relationship | 1 |
| ++ | Strong Positive Correlation | |
| + | Positive Correlation | |
| − | Negative Correlation | |
| ▼ | Strong Negative Correlation | |
| ▼ | Objective Is To Minimize | |
| ▲ | Objective Is To Maximize | |
| X | Objective Is To Hit Target | |

| Column # | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | Competitive Analysis (0=Worst, 5=Best) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Direction of Improvement: Minimize (▼), Maximize (▲), or Target (x) | | | | | | | | | | | | | | | | | | | | | | | | | |
| Row # | Max Relationship Value in Row | Relative Weight | Weight / Importance | Quality Characteristics (a.k.a. "Functional Requirements" or "Hows") / Demanded Quality (a.k.a. "Customer Requirements" or "Whats") | CALIDAD DE RECONOCIMIENTO | MODULARIDAD | CONECTIVIDAD INALAMBRICA | CAPACIDAD DE PROCESAMIENTO | INDICACIÓN DE ACCIÓN | COSTO | | | | | | | | | | Our Company | VOCCA | VoicePod | | | | Our Company<br>VOCCA<br>VoicePod<br>Column DG<br>Column DG<br>Column DG<br>0 1 2 3 4 5 |
| 1 | 9 | 15.2 | 5.0 | FACILIDAD DE USO | O | ▲ | | ▲ | Θ | | | | | | | | | | | 5 | 5 | 5 | | | | |
| 2 | 9 | 6.1 | 2.0 | ROBUSTEZ | Θ | | | Θ | | | | | | | | | | | | 3 | 3 | 4 | | | | |
| 3 | 9 | 12.1 | 4.0 | SEGURIDAD | Θ | | O | Θ | | | | | | | | | | | | 4 | 3 | 5 | | | | |
| 4 | 3 | 12.1 | 4.0 | PRECIO | O | ▲ | O | O | | O | | | | | | | | | | 2 | 1 | 1 | | | | |
| 5 | 9 | 12.1 | 4.0 | CONFIABILIDAD | Θ | | | Θ | | | | | | | | | | | | 4 | 4 | 3 | | | | |
| 6 | 9 | 6.1 | 2.0 | ESCALABILIDAD | ▲ | Θ | Θ | O | | ▲ | | | | | | | | | | 5 | 1 | 2 | | | | |
| 7 | 9 | 12.1 | 4.0 | TAMAÑO | | Θ | O | | | | | | | | | | | | | 4 | 4 | 3 | | | | |
| 8 | 1 | 12.1 | 4.0 | APARIENCIA | | ▲ | ▲ | | | | | | | | | | | | | 4 | 4 | 3 | | | | |
| 9 | 9 | 6.1 | 2.0 | DISIPACION | ▲ | | | Θ | | | | | | | | | | | | 2 | 2 | 3 | | | | |
| 10 | 3 | 6.1 | 2.0 | CONSUMO | ▲ | | O | ▲ | | | | | | | | | | | | 2 | 3 | 2 | | | | |
| | | | | Target or Limit Value | | | | | | | | | | | | | | | | | | | | | | |
| | | | | Difficulty (0=Easy to Accomplish, 10=Extremely Difficult) | | | | | | | | | | | | | | | | | | | | | | |
| | | | | Max Relationship Value in Column | 9 | 9 | 9 | 9 | 9 | 3 | | | | | | | | | | | | | | | | |
| | | | | Weight / Importance | 372.7 | 203.0 | 193.9 | 403.0 | 136.4 | 42.4 | | | | | | | | | | | | | | | | |
| | | | | Relative Weight | 27.6 | 15.0 | 14.3 | 29.8 | 10.1 | 3.1 | | | | | | | | | | | | | | | | |

# Parte IV

# Análisis de factibilidad

## 7. Factibilidad Tecnológica

### 7.1. Alternativas de diseño

En el proceso de diseño, se deben distinguir diferentes componentes críticos del producto para evaluar alternativas a los mismos, de modo tal de elegir la mejor opción. Una vez planteadas dichas alternativas, se deben enumerar las ventajas y desventajas de las mismas, sin ahondar en detalles de forma tal que la decisión sea independiente de las cuestiones propias de la implementación y así no perder una visión global del producto.

Inicialmente, se analiza la conectividad entre módulos para luego analizar alternativas específicas dentro de cada módulo.

#### Conectividad

Las tecnologías de conectividad Wireless de bajo consumo consideradas son:

- Bluetoorh Low Energy
- ANT/ANT+
- ZigBee
- RF4CE
- Wi-Fi
- Nike+
- IrDA
- NFC

La primera característica analizada es la topología de conexión que cada tecnología permite. Se definen las siguiente cinco topologías:

**Broadcast/Emisión** Comunicación unidireccional. Un componente es el encargado de transmitir acciones pero nunca recibe mensajes

**Malla** Los mensajes son repetidos por cada componente de la red hasta alcanzar el destino deseado.

**Estrella** Un componente central es capaz de comunicarse con varios componentes conectados.

**Escaneo** Busca constantemente recibir una señal de cualquier dispositivo transmitiendo en la misma banda.

**Punto-a-punto** Solo dos dispositivos están conectados en esta configuración.

| | Bluetooth LE | ANT | ANT+ | ZigBee | RF4CE | Wi-Fi | Nike+ | IrDA | NFC |
|---|---|---|---|---|---|---|---|---|---|
| Broadcast | √ | √1 | √1 | x | x | x | x | x | x |
| Mesh | √2 | √ | √ | √ | √ | x | x | x | x |
| Star | √ | √ | √ | √ | √ | √ | x | x | x |
| Scanning | √ | √3 | √ | √ | √ | x | √ | x | x |
| Point-to-Point | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Figura 9: Topología de conexión

Para este proyecto, las dos topologías que tendrían más sentido son emisión y estrella. La primera no permitiría que el módulo central reciba un feedback de que el mensaje llegue a destino y , por lo tanto, se decide utilizar la conexión de tipo estrella. En consecuencia, las tecnologías Nike+, IrDA y NFC no entran en esta categoría.

A continuación se analiza un cuadro generado por la empresa DigiKey, uno de los mayores distribuidores de componentes electrónicos, en el cual recomienda tecnologías en base a la aplicación deseada. 22

| | Bluetooth LE | ANT | ANT+ | RF4CE | ZigBee | Wi-Fi | Nike+ | IrDA | NFC |
|---|---|---|---|---|---|---|---|---|---|
| Remote Control | √ | x | x | √ | x | √ | x | √ | x |
| Security | √ | x | x | x | √ | √ | x | x | √ |
| Health and Fitness | √ | √ | √ | x | x | x | √ | x | x |
| Smart Meters | √ | x | x | x | √ | √ | x | x | x |
| Cell Phones | √ | x | √ | x | x | √ | x | √ | √ |
| Automotive | √ | x | x | x | x | √ | x | x | √ |
| Heart Rate | √ | x | √ | x | x | x | x | x | x |
| Blood Glucose | √ | x | √ | x | x | x | x | x | x |
| Positioning | √ | x | x | x | √ | √ | x | x | x |
| Tracking | √ | x | x | x | √ | x | x | x | √ |
| Payment | x | x | x | x | x | x | x | x | √ |
| Gaming | √ | x | x | x | x | x | x | √ | x |
| Key Fobs | √ | x | x | √ | x | x | x | √ | √ |
| 3DTV | √ | x | x | x | x | x | x | √ | x |
| Smart Applications | √ | x | x | x | √ | x | x | x | x |
| Intelligent Transport Systems | √ | √ | √ | x | √ | x | x | x | x |
| PCs | √ | x | x | x | x | √ | x | √ | √ |
| TVs | √ | x | x | √ | x | √ | x | √ | x |
| Animal Tagging | √ | x | x | x | √ | x | x | x | √ |
| Assisted Living | √ | √ | √ | x | x | x | x | x | √ |

Figura 10: Tecnologías recomendadas para diferentes aplicaciones

Las columnas en amarillo son las tecnologías descartadas debido a su topología de conexión. En azul se encuentran seleccionadas dos aplicaciones que permitirán que el producto sea escalable: Teléfonos celular y PCs. En base a esto se reduce la elección a dos posibilidades, Bluetooth y Wi-Fi.

- Bluetooth: Una opción equilibrada en precio, desarrollo y performance. La conectividad bluetooth se encuentra ampliamente documentada, y hoy en día su precio es bajo; justifica su elección la sencillez de implementación.
- Wi-Fi: La opción más compleja y menos económica. La comunicación tiene mayor redundancia y por ende mayor tiempo de comunicación. Sin embargo, permite la adaptación del sistema, en un futuro, a una aplicación web sin requerir que el usuario instale obligatoriamente software en su celular o computadora personal.

Dado que se prioriza la escalabilidad del sistema se eligió usar Wi-Fi. Nuevas generaciones del producto, en un

futuro, podrán agregar opciones para controlar los módulos terminales desde otras plataformas, como por ejemplo un teléfono celular o computadora. Adicionalmente, permitirá expandir el desarrollo de manera tal que se ofrezca una aplicación de teléfono adicional (sin reconocimiento de voz) en caso que se desee realizar una acción sin emitir sonido.

### 7.1.1. Módulo base o central

Existen dos componentes críticos a analizar en el módulo central: Procesador y sensor/es (adquisición de la señal/es).

#### Procesador

El módulo central se encarga de la parte mas importante del producto: La adquisición y el procesamiento de señales. Se utilizará un SoC (System on Chip), permitiendo que el software utilizado sea portable a teléfonos y computadoras.

Dentro del rango de opciones de circuitos SoC, se definió la Raspberry Pi como procesador a utilizar. Debido a la expertise personal con la misma se reducirá significativamente la curva de aprendizaje para su manejo y configuración.

#### Adquisición de señal

En el procesamiento de señales, la adquisición de señales es un cuello de botella en cuanto a la performance del producto. Existen diversas técnicas (a nivel hardware y software) para mejorar este aspecto. Debido a que a nivel software debe ser considerado el tiempo de desarrollo, este punto se centra en el análisis de factibilidad tecnológica a nivel físico. Las alternativas propuestas son similares en cuanto al sensor utilizado: Micrófono. Sin embargo, debe tenerse en cuenta la cantidad y disposición de los mismos, de acuerdo a la teoría de procesamiento de señales.

- Un micrófono: Es la opción mas económica. El micrófono no podrá detectar de dónde proviene una señal de voz, pero podrá analizarla y procesarla
- Arreglo de micrófonos: La ventaja de tener mas micrófonos es que se puede detectar la presencia de una señal de voz y modificar las ganancias de los micrófonos para maximizarla. Esto puede minimizar ruido generado por otros componentes domésticos.

Una gran limitación en la adquisición y procesamiento de la señal será el tiempo. El arreglo de micrófonos agregaría tiempo de procesamiento y costo. Por esta razón, se optó por un solo micrófono.

### 7.1.2. Módulo terminal

A continuación, se presentan las alternativas propuestas para cada componente importante del módulo terminal: Actuador, Conectividad, Procesador.

#### Actuador

El actuador es el eslabón final, que opera a nivel físico.

Las opciones son un relay o un dimmer.

- Relay: El relay es sencillo de operar, requiere un único pin del procesador (dos estados posibles). Esto se traduce en simplicidad de código (menor tiempo de desarrollo) y un circuito más simple (menor costo).
- Dimmer: El dimmer requiere mayor consideración, tanto a nivel hardware como software, para su correcta implementación. No obstante, resulta más versátil que el relay. En caso de querer expandir la funcionalidad del producto final a un comportamiento más completo (por ejemplo, control de nivel de intensidad de luz), contar con un dimmer permite la actualización del producto sin necesidad de modificar el hardware.

Si bien el dimmer parece una opción más atractiva, debe considerarse el tiempo de diseño y costo adicional que esto significa. Para mantener simplicidad en el proyecto se optó por un relay. Sin embargo, debido al criterio establecido para la escalabilidad vertical del producto, el diseño permitirá introducir módulos terminales con dimmer en un futuro.

#### Procesador

El módulo terminal no requiere demasiada potencia de cálculo, ni un procesador demasiado complejo. Sólo alcanza con el procesamiento necesario para mantener una conexión de WiFi y cambiar de estado un pin GPIO. Es por eso que entre los cientos de alternativas del mercado, se buscará un balance entre precio y calidad del mismo, entendiendo que sus requerimientos no son demasiado pretenciosos.

Debido a que se utiliza una comunicación WiFi, se deben considerar también microcontroladores con módulo Wi-Fi incorporado. La decisión final entre tener ambos en un solo microcontrolador o separados dependerá de los costos que representen cada opción.

## 7.2. DFMEA

La DFMEA es un análisis anticipado de fallas, que busca mitigar el efecto de las mismas al igual que facilitar la manufactura y ensamblaje. Dichas fallas pueden ocurrir con mayor o menor frecuencia, implicar mayor o menor gravedad y el conjunto de ambos aspectos se traduce en un impacto determinado en el producto.

Para la DFMEA del producto, se han identificado dos partes: Módulo central y módulo terminal. Cada uno de ellos

cuenta con tres subsistemas: Para el módulo central se define procesamiento de voz, comunicación y gabinete; para el módulo terminal se define comunicación, actuador y gabinete.

A continuación se presenta el análisis de DFMEA. Se definen los rangos entre 1 a 5 para la severidad, ocurrencia y detección, siendo 1 y 5 el mejor y peor caso respectivamente. El valor de RPN se define como el múltiplo de estos tres valores. Al analizar el RPN se busca realizar una acción para reducir la ocurrencia de valores según los siguientes criterios:

- $RPN \leq 27$: Aceptable
- $RPN > 27$ y $RPN < 48$: Se debe bajar hasta un valor razonablemente práctico
- $RPN > 48$: No aceptable

| Partes | Subsistemas | Falla potencial | Posible efecto | Causa | Severidad | Ocurrencia | Detección | RPN | Control de prevención | Severidad | Ocurrencia | Detección | RPN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Módulo terminal | Comunicación | Acciones no realizadas | Desincronización de la comunicación | Ruido en el medio de comunicación | 2 | 3.5 | 5 | 35 | Contemplar la re-sincronización de equipos en la programación | 2 | 2 | 5 | 20 |
| | | No permite efectuar comandos | Falla conección en encendido | Error al establecer la comunicación | 4 | 2 | 4 | 32 | La información de la red se establece y verifica en el servicio tecnico de instalación | 4 | 1.5 | 4 | 24 |
| | Actuador | Estados incorrectos | Falla encendido/apagado | Falla de circuito de control | 4 | 2 | 4 | 32 | Verificación de soldaduas y continuidad del PCB | 4 | 1 | 4 | 16 |
| | Gabinete | Sistema no funcional | Desconección entre modulo de comunicación y relay. | Movimiento separa ambas placas | 5 | 1 | 2 | 10 | | | | | |
| Módulo central | Procesamiento de voz | Falsa detección | Detección de un comando cuando el usuario no tiene la intención de realizar una acción | Ruido en el ambiente o usuario hablando | 5 | 4 | 2 | 40 | Minimizar detecciones falsas en prototipo | 5 | 2.5 | 2 | 25 |
| | | Detección erronea | Detección distinta (o errónea) cuando el usuario quiere realizar una acción | Algoritmo de detección no es capaz de reconocer la frase correctamente | 3.5 | 4 | 2 | 28 | Buscar un balance que no incremente considerablemente las detecciones falsas pero evite la mayor cantidad de detecciones erróneas | 3.5 | 3.5 | 2 | 24.5 |
| | | No detección | No hay detección alguna cuando el usuario quiere realizar una acción | Threshold de ruido muy alto | 2 | 5 | 5 | 50 | Ajustar el valor de threshold en el diseño de manera que funcione correctamente para varios micrófonos | 2 | 2.5 | 5 | 25 |
| | Comunicación | Falla en funcionabilidad del sistema | Desincronización de la comunicación | Ruido en el medio de comunicación | 2 | 3.5 | 5 | 35 | Contemplar la re-sincronización de equipos en la programación | 2 | 2 | 5 | 20 |
| | | No permite efectuar comandos | Falla conección en encendido | Error al establecer la comunicación | 3 | 1 | 5 | 15 | | | | | |
| | Gabinete | No permite conectar microfono | Raspberry Pi suelta dentro del gabinete | Diseño de gabinete | 5 | 1 | 1 | 5 | | | | | |
| | | Rotura del gabinete | Golpe sometido a stress fisico del gabinete | Diseño de gabinete | 5 | 2 | 3 | 30 | Gabinete robusto, con material antideslizante para evitar caidas accidentales | 5 | 1 | 3 | 15 |

# 8. Factibilidad de tiempos

Para organizar el proyecto se realiza un análisis de tiempos en el cúal cada sección del proyecto está compuesto por dos o más tareas. En primer lugar, se definen las tareas a realizar y se estima el tiempo que llevará a cabo cada tarea. Esto consiste en tres posibilidades: probable, optimista y pesimista.

Debido a que estos valores son subjetivos, se define una distribución de probabilidad BETA para cada actividad. Los parámetros de esta distribución serán:

$$\mu = \frac{t_{optimista} + 4t_{probable} + t_{pesimista}}{6}$$

$$\sigma = \frac{t_{pesimista} - t_{optimista}}{6}$$

Finalmente la combinación de tareas conformará una nueva distribución BETA, en donde se define la media y varianza como la suma de las medias y varianzas respectivamente de cada actividad secuencial.

La próxima figura presenta las tareas definidas junto con sus tiempos estimados y parámetros de sus distribuciones. La unidad temporal son días. Estos incluyen tantos días de la semana como fin de semanas.

| Id | Sección | Tarea | Optimista (días) | Probable (días) | Pesimista (días) | Medio (días) | Std (días) |
|---|---|---|---|---|---|---|---|
| 1 | Objetivos | Elección del producto | 3.00 | 5.00 | 6.00 | 4.83 | 0.50 |
| 2 | | Análisis de mercado | 5.00 | 10.00 | 12.00 | 9.50 | 1.17 |
| 3 | Definición de Producto | Definición de requerimientos | 4.00 | 7.00 | 10.00 | 7.00 | 1.00 |
| 4 | | Casa de calidad | 10.00 | 15.00 | 16.00 | 14.33 | 1.00 |
| 5 | | Especificaciones | 5.00 | 8.00 | 10.00 | 7.83 | 0.83 |
| 6 | Análisis de factibilidad | Factibilidad tecnológica | 7.00 | 15.00 | 30.00 | 16.17 | 3.83 |
| 7 | | Factibilidad temporal | 3.00 | 5.00 | 8.00 | 5.17 | 0.83 |
| 8 | | Factibilidad legal | 12.00 | 14.00 | 15.00 | 13.83 | 0.50 |
| 9 | | Factibilidad económica | 15.00 | 32.00 | 35.00 | 29.67 | 3.33 |
| 10 | Ingeniería en detalle: hard | Diseño hard módulo central | 1.00 | 3.00 | 5.00 | 3.00 | 0.67 |
| 11 | | Diseño hard módulo terminal | 30.00 | 40.00 | 50.00 | 40.00 | 3.33 |
| 12 | Ingeniería en detalle: soft | Diseño soft módulo central | 40.00 | 90.00 | 120.00 | 86.67 | 13.33 |
| 13 | | Diseño soft módulo terminal | 10.00 | 15.00 | 25.00 | 15.83 | 2.50 |
| 14 | Construcción del prototipo | Compra de componentes | 7.00 | 30.00 | 40.00 | 27.83 | 5.50 |
| 15 | | Diseño de PCBs | 10.00 | 14.00 | 15.00 | 13.50 | 0.83 |
| 16 | | Diseño Gabinete | 10.00 | 20.00 | 30.00 | 20.00 | 3.33 |
| 17 | | Soldadura y armado de prototipo | 1.00 | 2.00 | 3.00 | 2.00 | 0.33 |
| 18 | Validación del prototipo | Validación de hard | 1.00 | 2.00 | 3.00 | 2.00 | 0.33 |
| 19 | | Validación de soft | 1.00 | 3.00 | 5.00 | 3.00 | 0.67 |
| 20 | Reporte final | Correcciones | 8.00 | 10.00 | 15.00 | 10.50 | 1.17 |
| 21 | | Conclusiones | 1.00 | 2.00 | 2.00 | 1.83 | 0.17 |

Figura 11: Duración de las tareas

Se define cada color para cada sección del proyecto. Estos se mantienen a lo largo del análisis de tiempos. Utilizando los valores medios de cada actividad se hace una primera estimación de fechas para las tareas requeridas. Se define una fecha de inicio del proyecto en Julio 2016, fecha en que terminé de cursar.

| Id | Sección | Tarea | Comienzo | Fin | Inactividad |
|---|---|---|---|---|---|
| 1 | Objetivos | Elección del producto | 07/01/16 | 07/06/16 | 20 |
| 2 | | Análisis de mercado | 07/06/16 | 07/16/16 | |
| 3 | Definición de Producto | Definición de requerimientos | 08/05/16 | 08/12/16 | 120 |
| 4 | | Casa de calidad | 08/12/16 | 08/27/16 | |
| 5 | | Especificaciones | 08/27/16 | 09/04/16 | |
| 6 | Análisis de factibilidad | Factibilidad tecnológica | 01/02/17 | 01/17/17 | 20 |
| 7 | | Factibilidad temporal | 01/17/17 | 01/22/17 | |
| 8 | | Factibilidad legal | 01/22/17 | 02/05/17 | |
| 9 | | Factibilidad económica | 01/22/17 | 02/23/17 | |
| 10 | Ingeniería en detalle: hard | Diseño hard módulo central | 03/15/17 | 03/18/17 | 100 |
| 11 | | Diseño hard módulo terminal | 03/15/17 | 04/24/17 | |
| 12 | Ingeniería en detalle: soft | Diseño soft módulo central | 03/15/17 | 06/13/17 | 50 |
| 13 | | Diseño soft módulo terminal | 03/15/17 | 03/30/17 | |
| 14 | Construcción del prototipo | Compra de componentes | 09/21/17 | 10/21/17 | 0 |
| 15 | | Diseño de PCBs | 09/21/17 | 10/05/17 | |
| 16 | | Diseño Gabinete | 10/05/17 | 10/25/17 | |
| 17 | | Soldadura y armado de prototipo | 10/25/17 | 10/27/17 | |
| 18 | Validación del prototipo | Validación de hard | 10/27/17 | 10/29/17 | 0 |
| 19 | | Validación de soft | 10/27/17 | 10/30/17 | |
| 20 | Reporte final | Correcciones | 10/30/17 | 11/09/17 | - |
| 21 | | Conclusiones | 11/09/17 | 11/11/17 | |

Figura 12: Fechas de las tareas

Entre algunas secciones hay un “gap” de tiempos dados por inactividad. Esto significa que por razones ajenas, no hay avance del proyecto durante esos días.

Hay dos gaps importantes. El primero, luego de la definición de producto, incluye un período de 4 meses en el que me mudé fuera del país y periodo de fin de año (navidad y año nuevo). El segundo, luego de la ingeniería en detalle, hay un periodo de tres meses en donde realicé un curso intensivo full-time.

## 8.1. Planificación (PERT y simulación de Montecarlo)

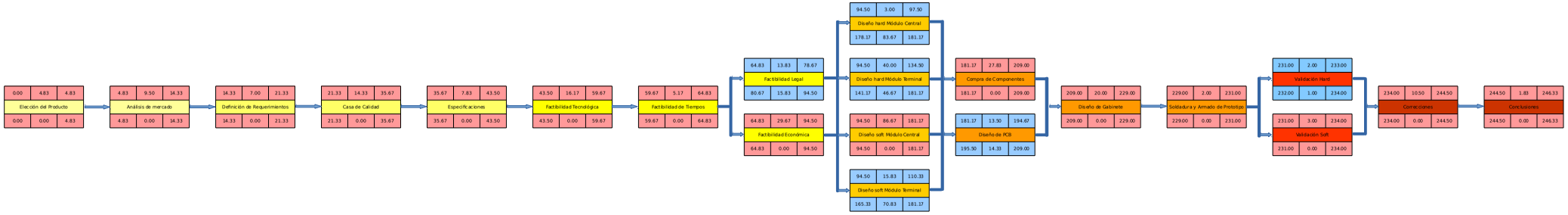
Utilizando los valores anteriores se procede a hacer una estimación de tiempos para la realización del proyecto. Se excluyen los tiempos de inactividad para hacer un análisis del proyecto basado solamente en las tareas del mismo. El diagrama PERT muestra los tiempos relacionados a cada actividad según el siguiente formato:

| Fecha Temprana Comienzo | Duración | Fecha Temprana Finalización |
|---|---|---|
| Tarea | | |
| Fecha Tardía Comienzo | Slack | Fecha Tardía Finalizacion |

Figura 13: Formato PERT

En el cual la duración es el valor medio de duración de la tarea y slack es el tiempo que se puede retrasar una tarea sin afectar la duración total del proyecto.

En el diagrama PERT, se utilizan los colores de sección establecidos anteriormente para el campo de “Tarea” y los otros campos tienen color rojo o azul. El camino de tareas en rojo representa el camino crítico.


0.00 4.83 4.83
Elección del Producto
0.00 0.00 4.83
4.83 9.50 14.33
Análisis de mercado
4.83 0.00 14.33
14.33 7.00 21.33
Definición de Requerimientos
14.33 0.00 21.33
21.33 14.33 35.67
Casa de Calidad
21.33 0.00 35.67
35.67 7.83 43.50
Especificaciones
35.67 0.00 43.50
43.50 16.17 59.67
Factibilidad Tecnológica
43.50 0.00 59.67
59.67 5.17 64.83
Factibilidad de Tiempos
59.67 0.00 64.83
64.83 13.83 78.67
Factibilidad Legal
80.67 15.83 94.50
64.83 29.67 94.50
Factibilidad Económica
64.83 0.00 94.50
94.50 3.00 97.50
Diseño hard Módulo Central
178.17 83.67 181.17
94.50 40.00 134.50
Diseño hard Módulo Terminal
141.17 46.67 181.17
94.50 86.67 181.17
Diseño soft Módulo Central
94.50 0.00 181.17
94.50 15.83 110.33
Diseño soft Módulo Terminal
165.33 70.83 181.17
181.17 27.83 209.00
Compra de Componentes
181.17 0.00 209.00
181.17 13.50 194.67
Diseño de PCB
195.50 14.33 209.00
209.00 20.00 229.00
Diseño de Gabinete
209.00 0.00 229.00
229.00 2.00 231.00
Soldadura y Armado de Prototipo
229.00 0.00 231.00
231.00 2.00 233.00
Validación Hard
232.00 1.00 234.00
231.00 3.00 234.00
Validación Soft
231.00 0.00 234.00
234.00 10.50 244.50
Correcciones
234.00 0.00 244.50
244.50 1.83 246.33
Conclusiones
244.50 0.00 246.33

A continuación se presenta el resultado de una simulación montecarlo. Este se utiliza para estimar la duración del proyecto ante la incertidumbre de la duración de las tareas. Utilizando el lenguaje de programación Python se simuló 10000 veces cada tarea del proyecto y se calculó la duración total del proyecto en cada iteración. Luego, se agupó esta información en forma de histograma normalizado.

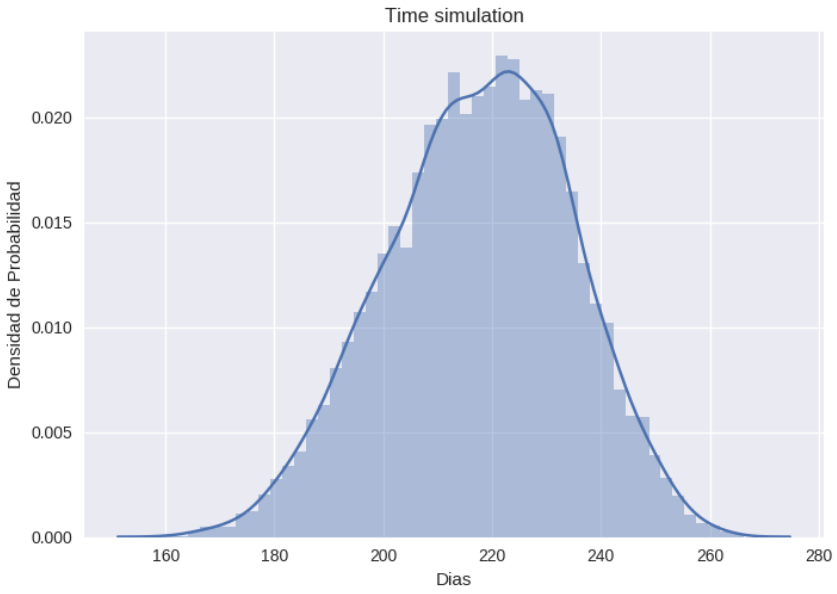



Figura 14: Simulacón Montecarlo

## 8.2. Programación (Gantt)

Finalmente, considerando los tiempos de inactividad, se presenta un diagrama de Gantt. A diferencia del diagrama de PERT, este representa las mismas actividades mostrando las fechas objetivo. El mismo concluye con la finalización de la última actividad el 11/11/17.

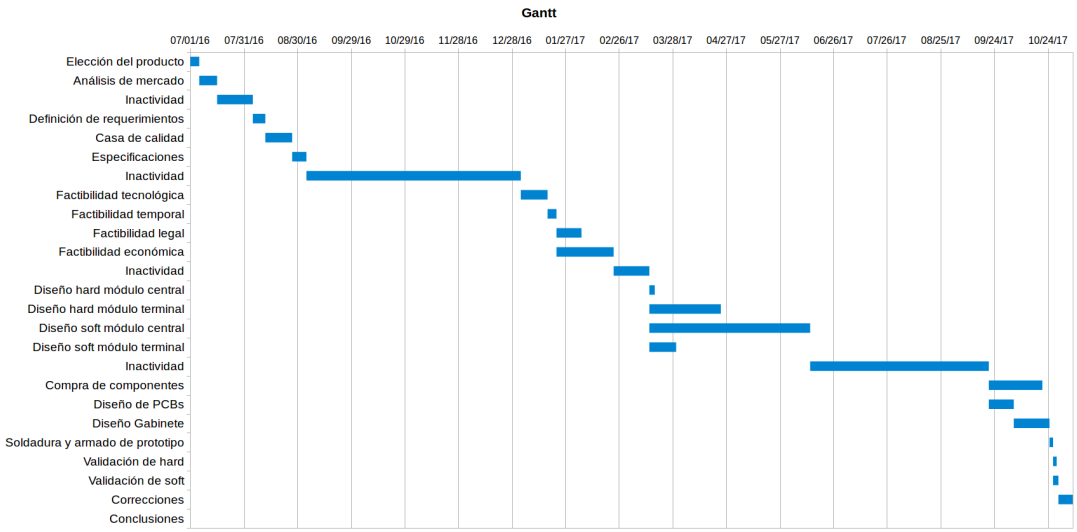



Figura 15: Gant

# 9. Factibilidad económica

En los últimos años, el mercado mundial de los sistemas de domótica ha crecido considerablemente. En 2013, según la organización “Light Control Association”, el mercado era de 5770 millones de dólares y se espera un incremento a 12.81 billones de dólares para el 2020. Estos sistemas no presentan sólo un aspecto de comodidad, sino también de seguridad y ahorro energético. A pesar de que este proyecto se concentra en la comodidad, se debe tener en cuenta posibles expansiones contemplando la eficiencia energética, ya que hoy en día existe una creciente concienciación sobre el tema.

La cantidad de dispositivos a controlar cada vez será más alto. ABI Research estimó la presencia de 17 millones de dispositivos domóticos inalámbricos instalados en hogares en 2013 y proyecta que en el 2018 habrá aproximadamente 500 millones de dispositivos. Esto genera expectativas en cuanto al interés general de la población y las ganancias esperadas.

Es importante la visión a futuro del desarrollo de los sistemas de domótica, debido a que se debe contemplar siempre la escalabilidad de los mismos. La empresa americana Gartner, líder en investigación de tecnología de la información, estima 30 mil millones de dispositivos conectados a la web en 2020, siendo esta la era de la creación de la información. Los hogares crearán información de la mayoría de los dispositivos, ofreciendo una fácil gestión para los usuarios e información cualitativa a los fabricantes para futuras mejoras o productos.

El sistema propuesto estará inicialmente enfocado a un medio ambiente hogareño unifamiliar. El usuario podrá definir qué quiere incorporado en su sistema, permitiéndole expandir la cantidad de módulos que desee.

Habrá dos opciones de compras para los clientes: un kit básico (módulo base más dos terminales) y un módulo terminal adicional. El kit básico tendrá un servicio de instalación sin costo, ya que esta tárea es sencilla, mientras que la instalación de los sistemas terminales adicionales tendrán costo. Además se proveerá un manual por si un usuario experimentado quiere agregar módulos independientemente.

## 9.1. Demanda

El producto empezará su oferta en Buenos Aires con posibilidad de compra sin servicio técnico en otras provincias. Se debe diferenciar el público para poder realizar una estrategia de marketing apropiada para cada uno.

El primer grupo estará compuesto por viviendas bajo alquiler, ya que la instalación del producto no deberá modificar la infraestructura del hogar y es fácil de instalar. Para ello se debe considerar la oferta en los alquileres. Se utiliza como fuente de información al sitio web Reporte Inmobiliario. Es una plataforma digital en la que se puede encontrar información relacionada con el Real Estate de Sudamérica. Indices, estadísticas, herramientas, artículos y la posibilidad de publicar propiedades sin costo son algunos de los beneficios del portal.

ReporteInmobiliario.com nació naturalmente durante el 2003, en momentos de mucha confusión, en los cuales fue necesario contar con diagnósticos objetivos y clarificadores para el mercado inmobiliario argentino

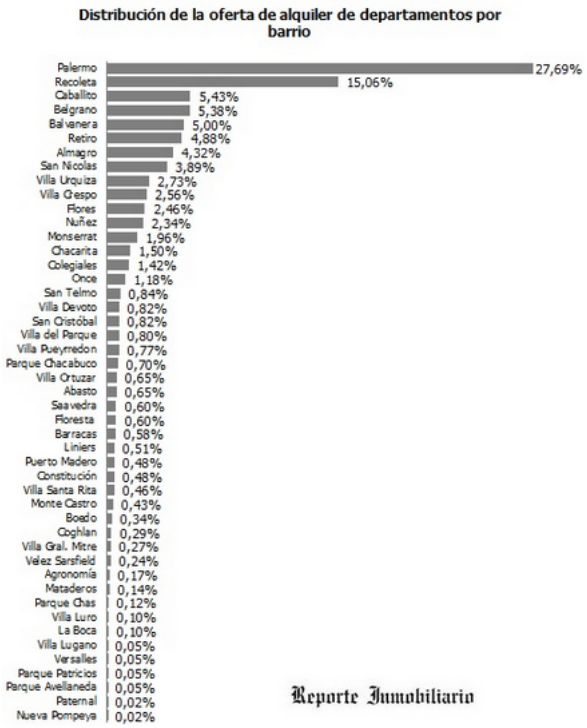



Figura 16: Oferta de alquileres por barrio 2017

Palermo, Recoleta, Caballito, Belgrano y Balvanera serán los barrios enfocados para este grupo. Seriá oportuno buscar este tipo de público mediante marketing digital, aumentando la inversión para realizar publicidad dirigida a estos barrios. La oferta de alquileres acumulada en estos barrios en conjunto se encuentra dentro del 50 y 60 % de la oferta total, según el reporte de 2017 efectuado por Reporte Inmobiliario. 22

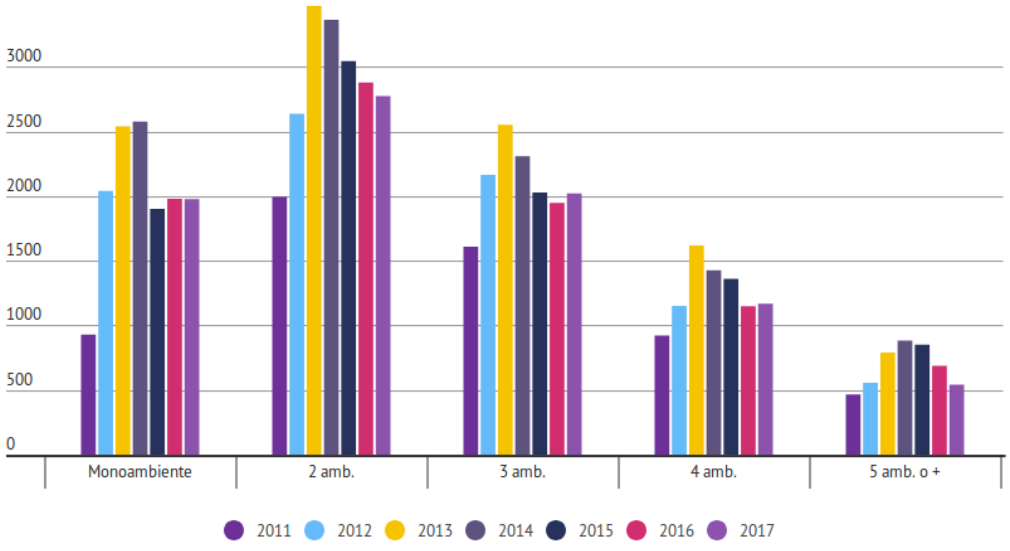



Figura 17: Oferta de cantidad de alquileres por ambiente 2011 a 2017

La figura anterior indica la cantidad de avisos de alquileres en Buenos Aires según Maure Inmobiliaria. Inicialmente, se buscará vender el producto en hogares de 1 o 2 ambientes, en donde haya poca cantidad de gente viviendo

y el producto pueda salir al mercado en sus mejores condiciones de uso: poco ruido interno en el ambiente y baja interferencia de varias voces al mismo tiempo. El público total para estos hogares será 2000 para monoambientes y 2700 aproximadamente para 2 ambientes. En conjunto, es un total de 4700 hogares para los cuales tomaremos el 50 %, representando los barrios de mayor oferta anteriormente mencionados. Por lo tanto el público de este grupo será 2350 hogares. Dado que en las encuestas se vio reflejado un desinterés del producto por un pequeño grupo de gente se decidió estimar que se logrará vender el producto a 1000 hogares en un periodo de dos a tres años. Para tener un margen de seguridad se estimarán 300 por año.

El segundo grupo estará constituido por familias o individuos mudándose a un edificio nuevo, recién construido. Se espera que al mudarse a un nuevo edificio sean más propensos a aumentar su inversión en el nuevo hogar con tecnología nueva e interesante. Una estrategia de marketing para este grupo de personas puede resultar en vender el producto al costo de producción con la meta de que ellos graben videos y saquen fotos del sistema para subirlos a instragram, facebook, twitter o cualquier otra red social.

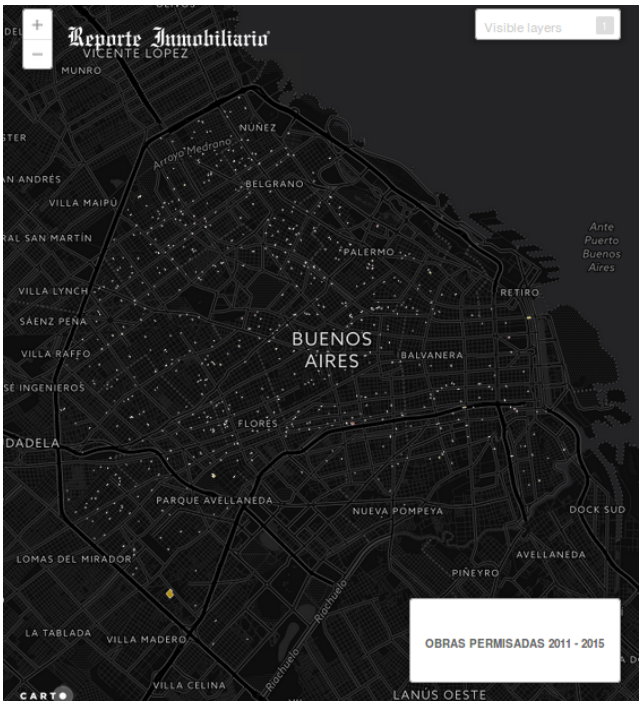



Figura 18: Obras en Buenos Aires

Para ello se debe analizar las obras en la ciudad de Buenos Aires. Se puede ver que las obras están distribuidas equilibradamente dentro de la ciudad. Por lo tanto se buscará contactar directamente a estas nuevas obras, mediante email.

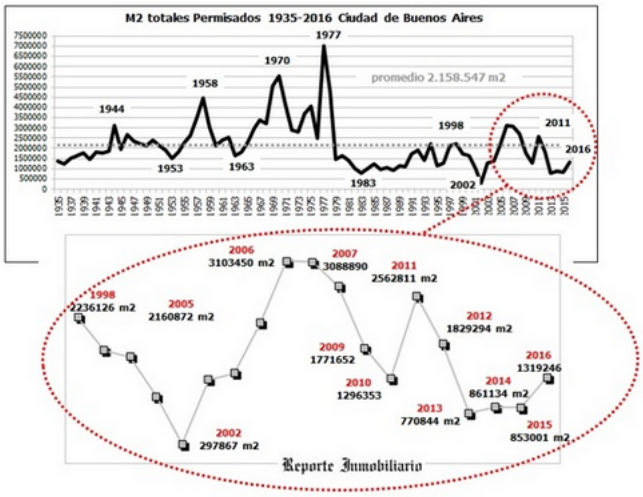



Figura 19: Metros cuadrado de obras en Buenos Aires

Basado en el gráfico anterior, se estiman 1500000 m2 de obras para 2017 y teniendo en cuenta que el tamaño de un ambiente es menor a 100 m2 se considerará como 15000 ambientes. Se estima, manteniendo un margen conservador, que al menos se podrá alcanzar a un 1 % de estos últimos, 150 hogares/ambientes por año.

En base a lo establecido anteriormente, se estima 300 + 150 = 450 hogares o ambientes por año.

Adicionalmente, existirá la posibilidad de comprar módulos terminales independientemente. Se estima que durante el primer año los clientes tendrán sólo el kit básico. Luego de un año, cuando el producto genere una sensación de comodidad, algunos querrán añadir módulos terminales adicionales. Teniendo en cuenta que no todos los clientes tendrán interés en agregar módulos adicionales, se asume que el 20 % aproximadamente comprarán como mínimo un módulo extra. Por lo tanto, se estiman ventas de 450*0.2=90 módulos terminales individuales por año.

## 9.2. Persona jurídica

Se creará una Sociedad por Acciones Simplificada (SAS). La integración dineraria mínima al inscribir la sociedad es del 25 % de dos salarios mínimos, vitales y móviles. Dicho valor a septiembre 2017 es de 4430 pesos.

Al inicio del proyecto, se generará la sociedad con soporte de escribano. Se creará mediante la página del gobierno. 22

## 9.3. Costos

Establecer los costos permite hacer un análisis financiero para finalmente aceptar o rechazar el proyecto. Existen costos relacionados al producto como costos relacionados a la sociedad o persona jurídica. Se analizan todos los costos juntos de forma tal de crear un valor para costos fijos y costos variables.

### 9.3.1. Costos variables

Los costos variables son los cuales dependen de la cantidad de unidad producidas. El costo por unidad de tiempo será la simple multiplicación de Q * P, cantidad * costo por unidad.

Existen dos tipos de costos variables. El primero será llamado costo del sistema básico. El mismo está compuesto de un módulo base o central con dos módulos terminales. Adicionalmente, se permite agregar módulos terminales. En consecuencia, hay un costo variable por unidad de sistema básico y otro por módulo terminal. Como parte de servicio se debe ofrecer asistencia para agregar los módulos pero se le permite al usuario avanzado configurarlo bajo su propio riesgo.

A continuación se detalla los costos en dólares del sistema básico:

| **Kit Básico** | | | |
|---|---|---|---|
| | **Cantidad** | **Precio/unidad (U$)** | **Precio total** |
| Raspberry Pi + Micrófono | 1 | 37 | 37 |
| Tarjeta SD (8Gb) | 1 | 8 | 8 |
| Micros terminal (Adafruit HUZZAH ESP8266 Breakout ) | 2 | 10 | 20 |
| Circuito Relays | 2 | 20 | 40 |
| Transformador 220/5V | 3 | 10 | 30 |
| Gabinete Raspberry Pi | 1 | 8 | 8 |
| Gabinetes Modulo terminal | 2 | 20 | 40 |
| Caja de carton | 1 | 0.7 | 0.7 |
| | | **Total sin impuesto** | 183.7 |

Figura 20: Costo sistema básico

El costo total para la creación del sistema básico es de $183.7 dólares. El precio del producto se establece en $435 dólares.

Los costos variables en dólares por cada módulo terminal individual son los siguientes:

| **Modulo terminal** | | | |
|---|---|---|---|
| | **Cantidad** | **Precio por unidad** | **Precio total** |
| Micros terminal (Adafruit HUZZAH ESP8266 Breakout ) | 1 | 10 | 10 |
| Circuito Relays | 1 | 20 | 20 |
| Transformador 220/5V | 1 | 10 | 10 |
| Gabinetes Modulo terminal | 1 | 20 | 20 |
| Caja de carton | 1 | 0.7 | 0.7 |
| | | **Total sin impuesto** | 60.7 |

Figura 21: Costo módulo terminal

El costo para la creación de un modulo terminal será de $60.7 dólares. El precio final de un módulo terminal adicional, incluyendo el servicio de instalación, se establece en $220 dólares.

### 9.3.2. Costos fijos

Existen dos tipo de costos fijos. El primero será un costo único al inicio del proyecto. Esto constituye el costo de creación de la sociedad, junto con costo de escribano. Adicionalmente, se deberá realizar la compra de una computadora que será utilizada para programación, control de campaña de marketing y manejo de emails.

Estos costos serán:

| Costos fijos únicos | | | | |
|---|---|---|---|---|
| | Precio en Pesos | Precio en Dolares | Cantidad | Total |
| Inversion bienes de uso: Computadora | 11500 | 665.12 | 1 | 665.12 |
| Escribano | 10000 | 578.37 | 1 | 578.37 |
| Costo creación de SAS | 4430 | 256.22 | 1 | 256.22 |
| | | | Total sin impuesto | 1499.71 |

Figura 22: Costos fijos únicos en dólares

A lo largo del proyecto existirán costos fijos en ciertos períodos. Se toma como unidad de tiempo un año. Por ley, se debe realizar un auditoria anual, la cual tiene un costo inherente y se deberá abonar honorarios al abogado a cargo de la misma. Habrá también un costo de contabilidad. Este será anual y contemplará la creación de balances y pagos de impuestos a fin del año fiscal.

Toda actividad económica se realizará a través de una cuenta bancaria. Los gastos bancarios fijos se estiman aproximadamente en $ 12,000 pesos por año. Los gastos bancarios variables serán considerado más adelante.

Para ensamblamiento del producto y todo tipo de servicio se utilizará un espacio físico. Ya que se podrá considerar un Start-up, se utilizarán oficinas compartidas. Precios típicos de alquiler de espacio para Start-ups son de $5000 pesos por mes. Ejemplo de este tipos de ofinicas son: Areatres, Lamaquinita y Wework. ([22], [22] y [22])

Se decide contratar un solo técnico. Este asistirá en tareas de soldadura, verificación de módulos y servicio técnico. Este tendrá un salario empleado (incluyendo carga social) de $20000 por mes.

Existirá un costo periódico debido a certificaciones (CE y FCC). Estas tendrán un periodo de vigencia de 180 días y se realizarán dos veces por año. Este costo se estima en $9000 pesos por certificación.

Para campaña de marketing, se utilizará google adwords. Se estima un costo de $100,000 anuales (costo estimado para pequeñas empresas)

Además, habrá que tener una línea telefónica de contacto para ventas, asistencia técnica y llamadas necesarias para otras actividades, como por ejemplo contacto con contador o abogado.

Los precios de los costos fijos periódicos se encuentran resumidos en la siguiente tabla:

| Costos fijos periodicos | | | | |
|---|---|---|---|---|
| | Precio en Pesos | Precio en Dolares | Cantidad | Total |
| Salario CEO | 360000 | 20821.28 | 1 | 20821.28 |
| Abogado temas legales | 16800 | 971.66 | 1 | 971.66 |
| Contabilidad | 144000 | 8328.51 | 1 | 8328.51 |
| Marketing online | 100000 | 5783.69 | 1 | 5783.69 |
| Gastos bancarios | 12000 | 694.04 | 1 | 694.04 |
| Oficina | 60000 | 3470.21 | 1 | 3470.21 |
| Salario empleado + carga social | 240000 | 13880.86 | 1 | 13880.86 |
| Certificaciones | 9000 | 520.53 | 2 | 1041.06 |
| Costo telefono | 6756 | 390.75 | 1 | 390.75 |
| | | | Total sin impuesto | 55382.07 |

Figura 23: Costos fijos periódicos en dólares

## 9.4. Ciclo de vida

El desarrollo del producto se seguirá durante la etapa comercial del mismo. Esto será para poder generar mejoras, expansiones o adaptaciones, de acuerdo a las necesidades generadas por el proyecto. La vida útil de los componentes del producto se estiman entre 3 y 4 años. La vida del proyecto será de 3 años, dando lugar a realizar una

primera gran modificación a fin del mismo, asegurando la compatibilidad con las versiones anteriores. Durante este tiempo, también puede haber re-lanzamientos o mejoras pequeñas (que aseguren la correcta funcionalidad del equipo).

## 9.5. Indicadores financieros

Para el análisis financiero del producto, se consideró una tasa de inflación del 13.8 % según el INDEC (desde Diciembre 2016) y tasa de interés del 2.68 %, U.S. 30 Year Treasury Bond, según marketwatch.com. La tasa de inflación se utiliza para ajustar los costos fijos, los cuales son en pesos.

Otras tasas tenidas en cuentas son:

- Ingresos Brutos: Es del 3 % de las ganancias de ventas totales. Se representa como gasto de comercialización en el análisis de flujos.
- Impuesto al debito: Es del 0.06 % de los costos totales.
- Impuesto al credito: Es del 0.06 % de las ventas totales.
- Impuestos a las ganancias: Es el 35 % del total depues de considerar los impuestos.

Con respecto al IVA, se consideran dos tasas: 21 % y 10.5 %. Para los componentes, los cuales son importados el IVA será del 10.5 %. Para los costos fijos y ganancias de ventas será el 21 %. Debido a que en el año 0 no hay ganancias, el IVA es un número positivo. Ya que ante valores positivos de IVA se compensa con el resultado del período siguiente, el IVA en el año cero es 0 pero el valor obtenido se descuenta en el ciclo siguiente.

A continuación se presenta el análisis de flujos, el es en dólares:

| Año | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Ventas Módulo Básico | 0.00 | 450.00 | 450.00 | 450.00 |
| Ventas Módulo Terminal | 0.00 | 90.00 | 90.00 | 90.00 |
| Ingreso Ventas MB | 0.00 | 195,750.00 | 195,750.00 | 195,750.00 |
| Ingreso Ventas MT | 0.00 | 19,800.00 | 19,800.00 | 19,800.00 |
| Ingreso Ventas Totales | 0.00 | 215,550.00 | 215,550.00 | 215,550.00 |
| Gasto de comercialización | 0.00 | -6,466.50 | -6,466.50 | -6,466.50 |
| Costos Módulo Básico | -82,665.00 | -82,665.00 | -82,665.00 | 0.00 |
| Costos Módulo Terminal | -5,463.00 | -5,463.00 | -5,463.00 | 0.00 |
| Costos Fijos | -1,499.71 | -55,382.07 | -55,382.07 | -55,382.07 |
| Costos Fijos considerando inflación | -1,499.71 | -63,024.80 | -71,722.22 | -81,619.88 |
| IVA | 0.00 | -22,461.91 | -30,203.83 | -28,125.32 |
| Costos Totales | -89,627.71 | -180,081.21 | -196,520.55 | -116,211.71 |
| Impuesto al debito | -53.78 | -108.05 | -117.91 | -69.73 |
| Impuesto al credito | 0.00 | -129.33 | -129.33 | -129.33 |
| Impuestos y tasas | -53.78 | -237.38 | -247.24 | -199.06 |
| Total antes de ganancias | -89,681.49 | 35,231.41 | 18,782.21 | 99,139.23 |
| Ganancias | 0.00 | 12,330.99 | 6,573.77 | 34,698.73 |
| Total (U$) | -89,681.49 | 22,900.42 | 12,208.43 | 64,440.50 |

Figura 24: Flujos en dólares

Finalmente se realiza el análisis financiero al total. El resultado obtenido mediante un análisis de VAN con tasa de descuento de 2.68 % es de $3725.94 dólares y el análisis de TIR es del 4 %, número mayor a la tasa utilizada. Teniendo en cuenta estos resultados junto con la escalabilidad y oportunidades que podrá generar, se acepta el proyecto.

# 10. Factibilidad legal

## 10.1. Introducción

Todo producto que desee ser comercializado en la República Argentina debe cumplir ciertas regulaciones de carácter obligatorio. Estas normas protegen al usuario frente a posible daños causados por fallas de equipos eléctricos y establece normas básicas de construcción para prevenir cualquier tipo de accidente a usuarios y mascotas domésticas.

## 10.2. Generalidades

El producto debe velar por la seguridad de los usuarios por sobre la del propio dispositivo. Por lo tanto, debe ser diseñado de tal forma que prevea cualquier acción temeraria del usuario que pueda poner en riesgo su integridad. El proceso de diseño debe contemplar entonces dichas acciones (de forma preventiva) y actuar en concordancia para evitarlas.

## 10.3. Resoluciones

**Resoluciones 171/2016 y SC 0207/2017 de la SECRETARÍA DE COMERCIO del MINISTERIO DE PRODUCCIÓN**

La resolución 171/16 establece un régimen para la certificación obligatoria del cumplimiento de los requisitos esenciales de seguridad para los productos eléctricos de baja tensión que se comercializan en el país. El 01/06/17 entra en vigencia la resolución 0207/17, la cual redefine algunos artículos de la resolución 171/16.

El primer artículo de la resolución 171/16 establece:

> *"ARTÍCULO 1° — El equipamiento eléctrico de baja tensión que se comercialice en la REPÚBLICA ARGENTINA deberá contar con una certificación que acredite el cumplimiento de los requisitos esenciales de seguridad que se detallan en el Anexo I que, con DOS (2) hojas, forma parte integrante de la presente resolución."*

Se define equipamiento eléctrico de baja tensión en el artículo 2, sustituido por el artículo 1 de la resolución 0207/17, categoría en la cual entra el producto de este trabajo:

> *"ARTÍCULO 2°.- A los fines de la presente resolución, entiéndese por equipamiento eléctrico de baja tensión a los artefactos, aparatos o materiales eléctricos destinados a una instalación eléctrica o que formen parte de ella que tengan una tensión nominal entre CINCUENTA VOLTS (50 V) y MIL VOLTS (1.000 V) en valor eficaz de corriente alterna senoidal y entre SETENTA Y CINCO VOLTS (75 V) y MIL QUINIENTOS VOLTS (1.500 V) en corriente continua. Estos rangos refieren a tensiones de entrada o salida del equipo, independientemente de tensiones intermedias que ocurran dentro del equipo..."*

El anexo I (171/17) describe los requisitos esenciales de seguridad del equipamiento eléctrico de baja tensión. El mismo está dividido en tres puntos. El primero establece que el equipamiento eléctrico debe contener información acerca de las características fundamentales del producto. Esta información debe ser: "el país de origen, la razón

social del fabricante o la marca comercial registrada, su domicilio legal, la razón social y domicilio legal del importador y del distribuidor en el país y el modelo del producto."

Además, prohibe la aislación de clases 0 y 0I. El módulo terminal, el cual posee tensiones consideradas peligrosas, tendrá aislación de clase 2. Por lo tanto tendrá el símbolo de dicha clase.

Figura 25: Aislación Clase 2

El segundo punto del anexo I establece protección contra los peligros originados en el propio equipamiento eléctrico. El tercer punto establece protección contra los peligros causados por efecto de influencias exteriores sobre el equipamiento eléctrico. Para satisfacer estos dos puntos, ambos módulos tendrán su propio gabinete. Esto permite proteger al usuario del equipamiento y el equipamiento de efectos externos. Para ello se debe considerar que el gabinete sea resistente a golpes y no den posibilidad de electrocutar al usuario. Ya que la alimentación de los micros son de muy baja tensión (5V), la única conexión posiblemente dañina es la luz a conectar. Para ello se utiliza una conexión en la cual el usuario no pueda físicamente tocar la tensión de 220V, mediante un enchufe interno.

El artículo 3 establece:

> *"ARTÍCULO 3° — [...] Podrán ser titulares de las certificaciones del equipamiento eléctrico a que se refiere el Artículo 1° de la presente resolución, las personas físicas con domicilio real y fiscal en la REPÚBLICA ARGENTINA o las personas jurídicas que den cumplimiento a las exigencias referidas en la Ley General de Sociedades N° 19.550, T.O. 1984. [...]"*

La persona jurídica titular de las certificaciones será la SRL mencionada anteriormente.

El artículo 4 establece que los productos deben cumplir los requisitos de seguridad establecido por las normas IRAM correspondientes o, en caso de que tal norma IRAM no exista, la norma IEC aplicable.

> *"ARTÍCULO 4° — Las certificaciones exigidas por la presente medida deberán acreditar que los productos alcanzados por las normas listadas en el Anexo IV que, con UNA (1) hoja, forma parte integrante de la presente resolución, cumplan los requisitos de seguridad establecidos en dichas Normas IRAM. Para el resto de los productos alcanzados por este acto las certificaciones deberán acreditar que los productos cumplan los requisitos de seguridad establecidos por las Normas IRAM o IEC aplicable."*

El artículo 5 define las certificaciones por distinto tipos de sistema y da referencia a la resolución que las define.

*"ARTÍCULO 5° — Los fabricantes nacionales o importadores del equipamiento eléctrico de baja tensión, para demostrar el cumplimiento de la obligación establecida en el Artículo 1° de la presente resolución, podrán utilizar uno de los siguientes sistemas de certificación: Sistema N° 4 (de tipo); Sistema N° 5 (de marca de conformidad) o Sistema N° 7 (de lote), según lo establecido por el Artículo 1° de la Resolución N° 197 de fecha 29 de diciembre de 2004 de la ex SECRETARÍA DE COORDINACIÓN TÉCNICA del ex MINISTERIO DE ECONOMÍA Y PRODUCCIÓN. Para el caso del equipamiento eléctrico de baja tensión enumerado en el Anexo II que, con CINCO (5) hojas, forma parte integrante de la presente resolución, solo podrá utilizarse para la certificación el Sistema N° 5 (de marca de conformidad)."*

El artículo 1 de la resolución 197/2004 define los tipos de certificaciones:

*"a) Sistema N° 4: Ensayo de Tipo seguido de un control (vigilancia) que consiste en ensayos de verificación de muestras tomadas en el comercio y en fábrica. Sólo podrá optarse por este Sistema una vez que la SECRETARIA DE COORDINACION TECNICA del MINISTERIO DE ECONOMIA Y PRODUCCION establezca la vigilancia para el Régimen en cuestión.*
*b) Sistema N° 5: Ensayo de Tipo y evaluación del control de calidad de la fábrica y su aceptación, seguidos de un control (vigilancia) que tiene en cuenta, a su vez, la auditoría del control de calidad de la fábrica y los ensayos de verificación de muestras tomadas en el comercio y en la fábrica.*
*c) Sistema N° 7: Ensayo de Lote, que deberá realizarse sobre muestras representativas tomadas por cada lote fabricado o importado. A los efectos de la realización de la Certificación por Lote, la toma de muestras de cada lote de producción o de importación, por parte del Organismo de Certificación interviniente, se realizará de acuerdo a lo establecido por la norma IRAM 15, cuyos parámetros serán determinados por la entidad certificadora interviniente en función de la dimensión del lote presentado y de la información disponible que acredite su homogeneidad. Los productos certificados por lote deberán ser identificados en forma legible e indeleble indicando el número de lote y el número del certificado emitido por el Organismo de Certificación interviniente."*

Debido al hecho de que se planea ensamblar productos al mismo tiempo que la venta, la certificación para sistema N° 4 será apropiada para el producto. Finalmente, se obtendrán las certificaciones CE (Conformité Européene) para cumplir con los requisitos de seguridad y FCC para cumplir con los requisitos de emisión.

## 10.4. Normas no obligatorias

Hay que tener en cuenta que certificar el producto con normas no obligatorias requieren una mayor inversión, y es necesario evaluar el beneficio que dicha certificación representa.

A continuación se presentan los resultados de una encuesta que sirven para evaluar la importancia de dichas normas para el usuario corriente.

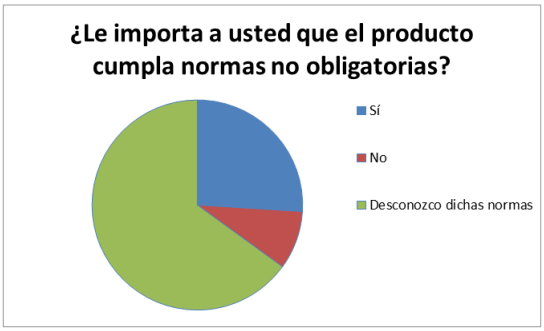



Figura 26: Encuesta correspondiente a la apreciación de normas no obligatorias

Debido a que es muy poco el porcentaje de usuarios que considera un factor clave la certificación de las normas no obligatorias, se optó por no realizar la certificación pero se volverá a considerar en el relanzamiento del producto al fin del ciclo de vida de la primera generación. Dos certificaciones que se querrán lograr en una segunda generación son de eficiencia energética y compatibilidad electromagnética. Para establecer una primera limitación en la EMC del relay, se buscará un switch mecánico que cumpla con la norma IEC 61810. Esta norma establece requisitos generales y de seguridad para relays en equipos de baja tensión.

# Parte V

# Ingeniería en detalle

## 11. Hardware

El hardware consiste en dos módulos: central y terminal. Los mismos son independientes entre sí, permitiendo su acceso por separado y sin necesidad de afectarse entre ellos, incluso para varios módulos terminales. La configuración de la cantidad y disposición de los módulos terminales será a través del programa principal. Es importante asegurar el funcionamiento de cada uno de los módulos para que el sistema funcione correctamente.

### 11.1. Diagrama de bloques

A continuación se presentan los diagramas de bloques de la conexiones básicas de cada módulo con sus componentes.

#### 11.1.1. Módulo central

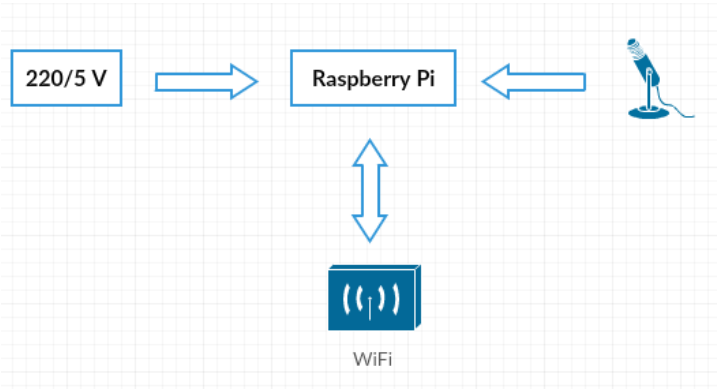



Figura 27: Módulo central

### 11.1.2. Módulo terminal

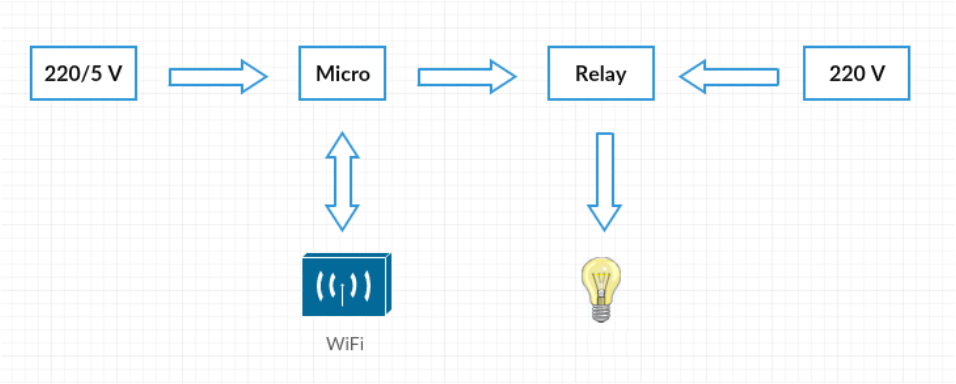



Figura 28: Módulo terminal

## 11.2. Descripción detallada

La interfaz entre ellos será mediante una conexión Wi-Fi. De esta forma se pueden agregar, reemplazar o quitar módulos con gran facilidad y sin necesidad de interrumpir el sistema. Además, se permitirá en un futuro desarrollar el control de las luces mediante una aplicación en el celular sin necesidad de modificar el hardware.

### 11.2.1. Módulo central

El módulo central es la unidad que contiene el programa principal y gobernará el sistema. Está compuesto por una Raspberry Pi 3, con un micrófono y un módulo Wi-Fi (incorporado en la Raspberry Pi). El usuario podrá conectar la Raspberry Pi a la red local mediante cable ethernet directamente al router (recomendado) o Wi-Fi. El módulo central se presentará ante el cliente previamente armado con el software instalado.

El mismo puede alimentarse de dos maneras distintas:

- Adaptador 220/5V - MicroUSB. (Típicamente utilizado como cargador de celular)
- Conexión directa a los pines GPIO.

El método elegido es la utilización del adaptador.

Figura 29: Raspberry Pi 3

### 11.2.2. Módulo terminal

El módulo terminal es el encargado de llevar a cabo las acciones deseadas de encendido o apagado de cada componente del sistema. Está compuesto por un microcontrolador con módulo Wi-Fi incorporado y un relay. Se alimenta con un adaptador 220/5V idéntico al del módulo central y conexión directa a 220VAC para el dispositivo o luz a controlar. Al actuar como esclavo y no producir ningún tipo de procesamiento, el módulo tendrá bajo consumo.

Recibe instrucciones mediante el módulo Wi-Fi, al cual contesta el estado de la ejecución requerida mediante el protocolo TCP/IP. Actúa directamente en la señal de control del releé, encendiendo o apagando la misma.

#### Microcontrolador

Se utiliza el microcontrolador «Adafruit HUZZAH ESP8266 Breakout» 22. Este consiste en un microcontrolador con reguladores de tensión, pines de propósito general (GPIO), módulo Wi-Fi ESP8266, pin de alimentación 3.3V regulado y posibilidades de conexión UART, SPI e I2C.

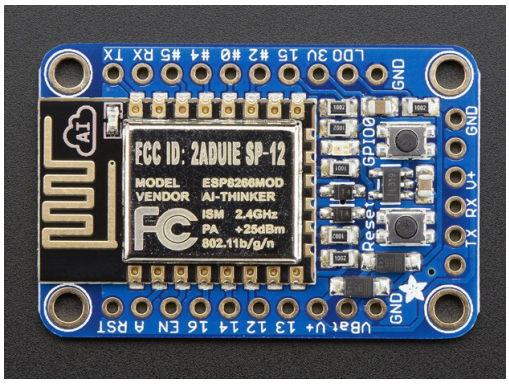



Figura 30: Adafruit Huzzah ESP8266 Breakout

Se utilizan los pines de UART para la programación del microcontrolador inicialmente y un pin GPIO como señal

de control del relay. La tensión de salida de los pines de GPIO es de 3.3V.

**Relay**

Para el relay se utiliza un circuito básico, compuesto de un transistor NPN, resistencias de base, relay o switch mecánico y un diodo de protección.

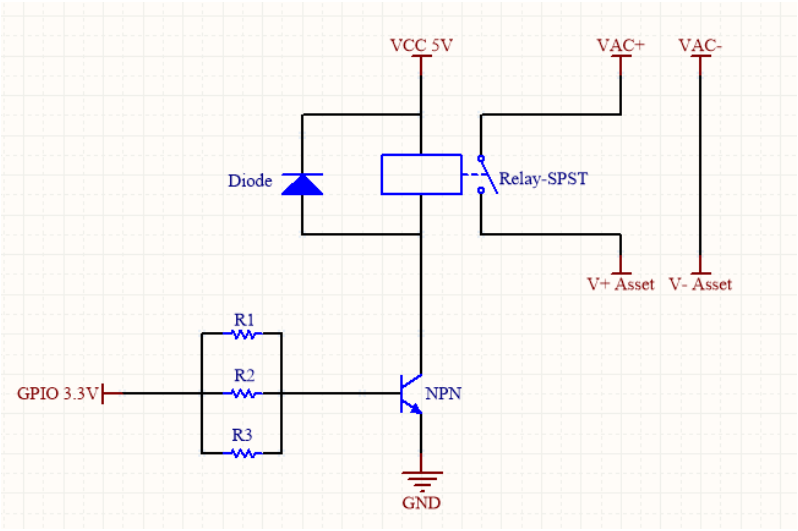



Figura 31: Diagrama del circuito de relay

## 11.3. Detalles de selección y cálculo de los elementos circuitales de cada bloque

### 11.3.1. Módulo central

El módulo central está compuesto por la Raspberry Pi sin elementos circuitales adicionales. El micrófono se coloca directamente en el puerto USB de la Raspberry Pi. La ventaja de no tener elementos adicionales es que se podrá incorporar un gabinete creado especialmente para Raspberry Pi, acelerando el proceso de diseño.

### 11.3.2. Módulo terminal

**Microcontrolador**

El módulo ESP8266 22 es un SoC (System on Chip) de bajo consumo con módulo de comunicación Wireless para aplicaciones de plataforma móvil (teléfono) y provee la habilidad de embeber Wi-Fi en otros sistemas.

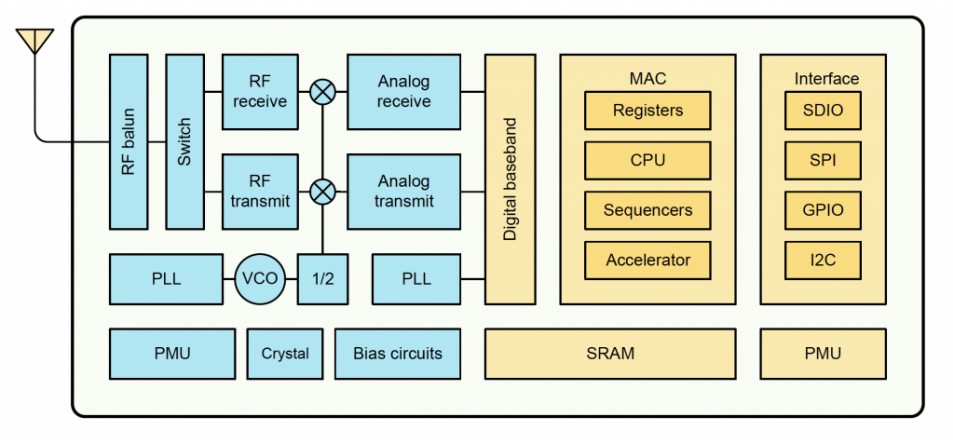



Figura 32: Diagrama de bloques ESP8266

Este chip tiene un firmware que contiene todas las librerías necesarias para cumplir con especificaciones 802.11 de Wi-Fi. En la documentación de Adafruit Huzzah, se recomienda considerar que el módulo ESP8266 puede consumir hasta 250mA. Esto se tomará en cuenta a la hora de establecer la alimentación del módulo terminal.

| Mode | Min | Typ | Max | Unit |
|---|---|---|---|---|
| Transmit 802.11b, CCK 1Mbps, Pout=+19.5dBm | | 215 | | mA |
| Transmit 802.11b, CCK 11Mbps, Pout=+18.5dBm | | 197 | | mA |
| Transmit 802.11g, OFDM 54Mbps, Pout=+16dBm | | 145 | | mA |
| Transmit 802.11n, MCS7, Pout=+14dBm | | 135 | | mA |
| Receive 802.11b, packet length=1024 byte, -60dBm | | 60 | | mA |
| Receive 802.11g, packet length=1024 byte, -70dBm | | 60 | | mA |
| Receive 802.11n, packet length=1024 byte, -65dBm | | 62 | | mA |
| Standby | | 0.9 | | mA |
| Deep sleep | | 10 | | uA |
| Power save mode DTIM 1 | | 1.2 | | mA |
| Power save mode DTIM 3 | | 0.86 | | mA |
| Total shutdown | | 0.5 | | uA |

Figura 33: Protocolos y consumo ESP8266

La placa Adafruit Huzzah consiste en un módulo ESP8266 con los siguientes agregados:

- Botón de reset
- Botón para que el usuario pueda poner el chip en modo de 'bootloading'
- LED rojo
- 'Level shifter' en pines de UART y botón de reset
- 3.3V de salida, con regulador de hasta 500mA
- Dos power inputs con protección con diodos

El esquemático de la placa según la documentación de Adafruit muestra los elementos agregados.

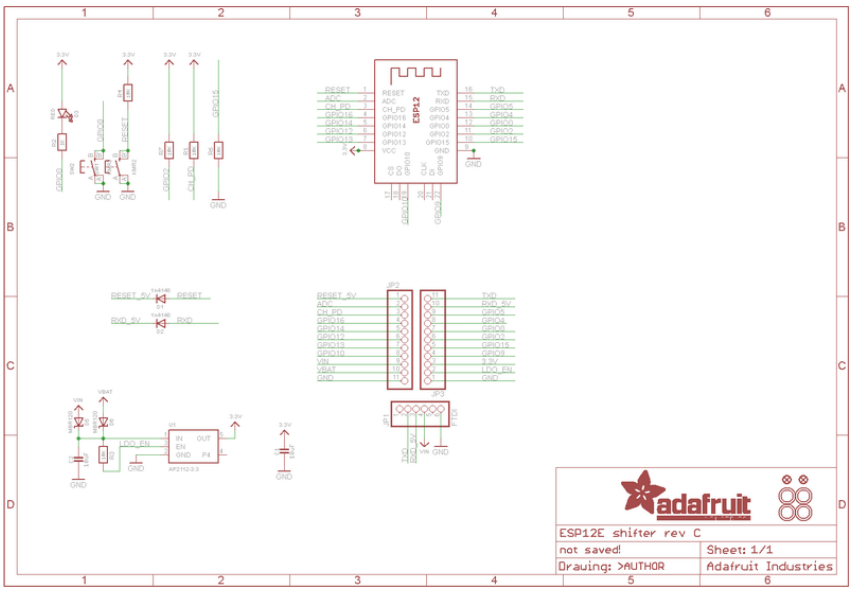



Figura 34: Esquemático Adafruit Huzzah

Una restricción importante es que los pines GPIO, de ESP8266 y Adafruit Huzzah, pueden proveer hasta 12 mA.

**Relay**

El relay se representa mediante el circuito de la figura 31. El transistor se utiliza como una llave. Este tiene dos estados posibles: corte y saturación.

El transistor no permite la circulación de corriente mientras el mismo esté en corte. Esto significa que no circula corriente por el switch mecánico y, en consecuencia, este actúa como circuito abierto en el lado del dispositivo a controlar.

Durante saturación, el transistor tiene corriente circulando mediante el colector y emisor, establecida por la corriente de base según la siguiente ecuación:

$$I_c = I_b hfe_{SAT} \tag{1}$$

donde $I_c$ es la corriente del colector, $I_b$ es la corriente de base y $hfe_{SAT}$ es la ganancia del transistor en estado de saturación.

El switch mecánico tiene una resistencia interna, la cual junto con la corriente de colector produce una caída de tensión en el switch.

$$V_{SW} = I_c R_{SW} \tag{2}$$

donde $V_{SW}$ es la caída de tensión en el switch y $R_{SW}$ es la resistencia interna del switch.

El switch mecánico se controla mediante su tensión. Si la caída de tensión en el switch excede un threshold, el mismo se activa, cerrando un lazo en el lado de la luz. En el otro extremo, la tensión en el switch no puede ser mayor a la tensión de alimentación.

Tomando las resistencias R1, R2 y R3 como una sola llamada R y aplicando la Ley de Kirchoff a la malla desde GPIO 3.3V a GND se obtiene:

$$V_{GPIO} - V_{BE_{SAT}} = V_R$$

donde $V_{GPIO}$ es la tensión del pin de GPIO de control, $V_{BE_{SAT}}$ es la tensión base-emisor de saturación y $V_R$ es la caida de tensión en la resistencia de base.

Ya que $V_R = I_b R$ se despeja una ecuación para calcular $I_b$

$$I_b = \frac{V_{GPIO} - V_{BE_{SAT}}}{R} \tag{3}$$

**Selección de componentes**

**Switch Mecánico OJE-SH-105DM,095** Es un relay de pequeño tamaño para voltajes de 5VDC con carga de hasta 277 VAC. La siguiente figura muestra las características del relay, obtenidas del datasheet. Se encuentran seleccionados los datos referentes al modelo de relay utilizado.

**Coil versions, DC coil, OJ/OJE-D and -H type**

| Coil code | Rated voltage VDC | Operate voltage VDC | Release voltage VDC | Coil resistance Ω±10% | Rated coil power mW |
|---|---|---|---|---|---|
| 003 | 3 | 2.1 | 0.15 | 20 | 450 |
| 005 | 5 | 3.5 | 0.25 | 55.6 | 450 |
| 006 | 6 | 4.2 | 0.3 | 80 | 450 |
| 009 | 9 | 6.3 | 0.45 | 180 | 450 |
| 012 | 12 | 8.4 | 0.6 | 320 | 450 |
| 024 | 24 | 16.8 | 1.2 | 1280 | 450 |
| 048 | 48 | 33.6 | 2.4 | 5120 | 450 |

All figures are given for coil without pre-energization, at ambient temperature +23°C

Figura 35: Características OJE-SH-105DM,095

Los valores de tensión nominal, activación y desactivación son de 5V, 3.5V y 0.25V respectivamente. La resistencia interna es de 55,56$\Omega$ y la potencia nominal es de 450mW.

**Transistor 2N2222** Es un transistor para pequeña señal de control. Datos necesarios para el cálculo de la resistencia de base requerida son $hfe_{SAT}$ y $V_{BE_{SAT}}$.

| | | | | |
|---|---|---|---|---|
| Collector – Emitter Saturation Voltage<br>($I_C$ = 150 mAdc, $I_B$ = 15 mAdc)<br>($I_C$ = 500 mAdc, $I_B$ = 50 mAdc) | $V_{CE(sat)}$ | <br>–<br>– | <br>0.3<br>1.0 | Vdc |
| Base – Emitter Saturation Voltage<br>($I_C$ = 150 mAdc, $I_B$ = 15 mAdc)<br>($I_C$ = 500 mAdc, $I_B$ = 50 mAdc) | $V_{BE(sat)}$ | <br>0.6<br>– | <br>1.2<br>2.0 | Vdc |

Figura 36: Características de encendido

La figura 36, dato sacado del datasheet del transistor, sugiere que para todos los casos de saturación $I_C = 10I_b$. Esto significa que $hfe_{SAT}$ tiene un valor de 10.

Dado que los pines de GPIO del microcontrolador pueden proveer hasta 12mA, surge una primera limitación a la corriente máxima de base. Para tomar un margen de seguridad esta primera limi-tación se establece en 10 mA. Una primera aproximación de la corriente de colector será 100 mA. Esto se usa para estimar la tensión de saturación entre junturas base-emisor y colector-emisor.

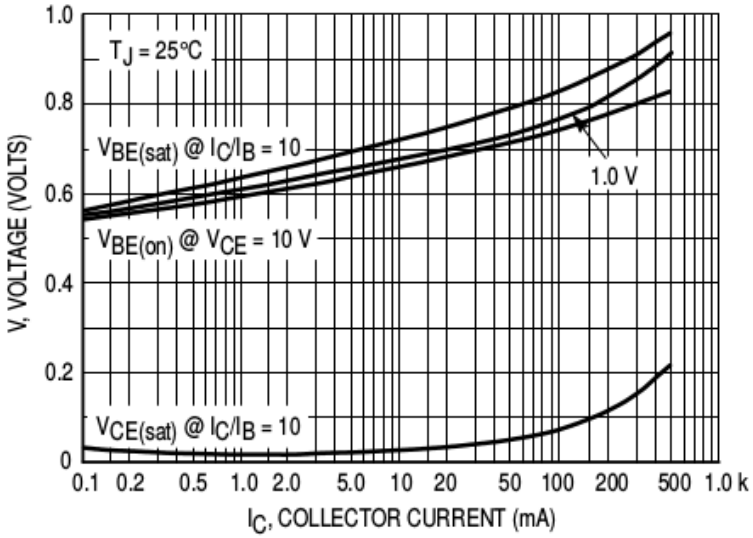



Figura 37: Voltaje de saturación Base-Emisor

Tomando como referencia la figura 37, para una corriente de colector de 100 mA la tensión de saturación base-emisor $V_{BE_{SAT}}$ será aproximadamente 0.8V y la tensión de saturación $V_{CE_{SAT}}$ será menor a 0.1 V.

Una vez estimado valores para $V_{BE_{SAT}}$ y $V_{CE_{SAT}}$ se vuelve a referir la figura 35. En esta se obtiene el valor de la resistencia del switch, siendo de $55{,}6\Omega \pm 10\,\%$. También se observa que el voltaje de activación mínimo es de 3.5V.

En la práctica, el valor de $hfe_{SAT}$ puede variar. Generalmente, el valor real es menor a 10. Se busca elegir un valor de resistencia que funcione para hfe variando entre 8 y 11.

Tomando como restricciones las tensiones mínima y máxima que puede caer en el switch se calcula los valores de hfe. La tensión mínima es 3.5V, tensión necesaria para activar el switch. La tensión máxima viene dado por $V_{cc} - V_{CE_{SAT}} = 5V - 0{,}1V = 4{,}9V$. Utilizando estos valores se calcula el rango de corrientes de colector posibles mediante la ecuación 2.

Dado un valor de R se utiliza la ecuación 3 para calcular la corriente de base. Finalmente, se calcula el rango de valores de $hfe_{SAT}$ mediante la ecuación 1.

La siguiente figura muestra el rango de valores de $hfe_{SAT}$ para distintos valores de resistencia.

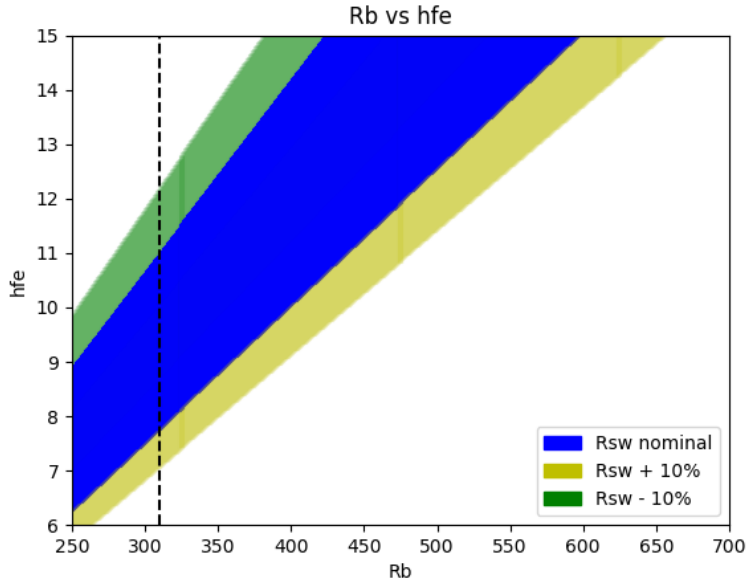



Figura 38: Rb vs hfe

La línea punteada es el valor de resistencia elegido. Este es de 310$\Omega$ y permite un rango de hfe entre 7.6 y 11.

**Diodo 1N4004** El diodo protege el transistor frente a picos de corriente generados en la transición entre saturación y corte. La inductancia incorporada en el switch debe eliminar la energía almacenada en el mismo. El diodo provee un camino para esta corriente, la cual será la misma que circula en por el switch en el momento de encendido, menor a 100 mA.

La elección del diodo consiste en que este tiene baja tensión en directa, soporta grandes picos de corriente y es de bajo costo.

**Alimentación** Se debe considerar que la alimentación del microcontrolador y relay será la misma, es decir, que el transformador 220/5V debe ser capaz de proveer la corriente necesaria. Se estima que el microcontrolador Adafruit Huzzah puede llegar a consumir 250 mA y la corriente del colector en el relay es menor a 100 mA. Por lo tanto, la alimentación debe ser capaz de proveer al menos 350 mA. Al permitir alimentar con cable USB-USB micro, este puede utilizarse desde un transformador similar a un cargador telefónico o puerto de computadora. Los transformadores generalmente utilizados pueden proveer corrientes hasta 1A. El puerto USB de una computadora puede proveer hasta 500 mA, aunque algunos puertos de carga llegan hasta 800 mA.

El método utilizado es un transformador 220/5V (cargador de teléfono).

**Potencia disipada** Finalmente se verifica la potencia máxima disipada en cada componente:

**Resistencia** Dado el valor de R establecido de $310\Omega$ y utilizando la ecuación 3 se obtiene: $I_b = \frac{3{,}3V - 0{,}8V}{310\Omega} \simeq 8{,}064mA$

Asumiento una caída de tensión de 0.8V en la juntura base-emisor la potencia máxima en la resistencia será:

$$P_{R_{max}} = I_b \cdot (3{,}3V - 0{,}8V) = 20{,}16mW$$

Esto significa una sola resistencia debería ser capaz de disipar $\frac{1}{4}$ de Watt mínimo. Como $310\Omega$ no es un valor comercial, se utilizan varias resistencias de $\frac{1}{8}$ de Watt en paralelo.

**Switch mecánico** Utilizando la ecuación 1 y asumiento un valor de $hfe_{SAT}$ de 11, la corriente máxima del colector es de $I_C = 8{,}064mA \cdot 11 = 88{,}709mA$. La corriente en el switch es la misma corriente que en el colector debido a que se encuentran en la misma malla, $I_C = I_{SW} = 88{,}709mA$. Sabiendo que la tensión máxima en el switch es de 4.9V, la potencia máxima disipada será:

$$P_{SW_{max}} = I_{SW} V_{SW} = 88{,}709mA \cdot 4{,}9V = 434{,}67mW$$

Este resultado es menor al máximo valor de potencia disipada en la figura 35, 450mW.

**Transistor 2N2222** Dado que la tensión de threshhold de activación del switch es de 3.5V, la tensión máxima en la juntura colector-emisor puede ser 1.5V. Utilizando la corriente del colector, la potencia máxima disipada será:

$$P_{2N2222_{max}} = I_c V_{CE_{SAT_{max}}} = 88{,}709mA \cdot 1{,}5V = 133{,}06mW$$

Según el datasheet la máxima potencia de disipación a 25 grados centigrados es de 625mW, valor mayor al obtenido.

## 11.4. Plan de pruebas

### 11.4.1. Módulo central

A continuación se detallan las verificaciones necesarias para el hardware del módulo central. Algunas tienen su origen en las especificaciones, mientras que otras surgen de validaciones necesarias para comprobar elementos que componen al módulo.

| Origen | Nombre | Descripción | Valor esperado |
|---|---|---|---|
| FUN-07 | Consumo módulo central | 1. Con un voltimetro con error de +/- 0.05V medir la tensión de alimentación<br>2. Con un amperimetro con error de +/- 5mA medir la corriente de alimentación<br>3. Finalmente cálcular la potencia del módulo como el múltiplo de ambos valores | < 10 W |
| DYP-01 | Dimensión máxima módulo central | Con una regla (con error de +/- 0.5 mm) medir las dimensiones del gabinete | 15x15x5 cm |
| DYP-03 | Peso módulo central | Con una balanza con error de +/- 1 gramo pesar el módulo | < 2kg |
| INT-US-04 | Alcance del micrófono | 1. Grabar sonido<br>2. Reproducir el sonido y verificar que sea audible<br>3. Alejarse del micrófono y repetir hasta que sea inaudible | > 10 m |
| INT-VIN-01 | Módulo central: Transformador 220VAC/5VDC | Con un voltimetro con error de +/- 0.05V medir la tensión de alimentación | 4.5 – 6 V |

Cuadro 10: Plan de pruebas HW con origen en especificaciones - Módulo central

Notar que la prueba con origen FUN-07 no será necesaria ya que la Raspberry Pi consume menos de 6.7 W según el datasheet [22]. Similarmente, la prueba con origen DYP-01 se cumplirá ya que la dimensión de la Raspberry Pi es de 85.60×56.5 ×17 mm.

A continuacións se presentan las pruebas adicionales que se deberán realizar para verificar el micrófono, Wi-Fi y conector Ethernet. Estas se realizarán previamente a las pruebas mencionadas en el cuadro 10.

| ID | Nombre | Descripión | Valor esperado |
|---|---|---|---|
| MC-HW-01 | Verificación micrófono | 1. Grabar un sonido<br>2. Reproducir el sonido y verificar el sonido | Sonido audible |
| MC-HW-02 | Verificación Wi-Fi | Comprobar que se pueda conectar remotamente a la Raspberry Pi a través de la red local | Conexión exitosa |
| MC-HW-03 | Verificación puerto Ethernet | Conectar el puerto ethernet y comprobar que se pueda conectar remotamente a la Raspberry Pi a través de la red local | Conexión exitosa |

Cuadro 11: Plan de pruebas HW adicionales - Módulo central

### 11.4.2. Módulo terminal

A continuación se detallan las verificaciones necesarias para el hardware del módulo terminal. Algunas tienen su origen en las especificaciones, mientras que otras surgen de validación necesaria para comprobar el módulo Wi-Fi de la placa Adafruit Huzzah y correcto funcionamiento del circuito de relay.

| Origen | Nombre | Descripión | Valor esperado |
|---|---|---|---|
| FUN-09 | Consumo microcontrolador módulo terminal | 1. Con un voltimetro con error de +/- 0.05V medir la tensión de alimentación del microcontrolador<br>2. Con un amperimetro con error de +/- 5mA medir la corriente de la alimentación del microcontrolador<br>3. Cálcular la potencia del módulo como el múltiplo de ambos valores | < 3 W |
| DYP-02 | Dimensión máxima módulo terminal | Con una regla (con error de +/- 0.5 mm) medir las dimensiones del gabinete | 15x10x5 cm |
| DYP-04 | Peso módulo terminal | Con una balanza con error de +/- 1 gramos pesar el módulo | < 2kg |
| INT-US-01 | Indicador de acción sonoro: Switch mecánico | - | - |
| INT-US-02 | Protección de tensión peligrosas en módulo central: gabinete con espesor | - | > 1mm |
| INT-US-03 | Protección de tensión peligrosas en módulo central: Zocalo de conección | - | - |
| INT-VIN-02 | Módulo terminal: Transformador 220VAC/5VDC | Con un voltimetro con error de +/- 0.05V medir la tensión de alimentación | 4.5 – 6 V |
| INT-VOU-01 | Módulo terminal: Resistencia de salida < 0.5 Ohm | Con un medidor de impedancias con error de +/- 0.05 Ohm medir la resistencia de salida (con el swith activado) | < 0.5 Ohm |

Cuadro 12: Plan de pruebas HW con origen en las especificaciones - Módulo terminal

La prueba INT-US-01 será cualitativa, ya que espera que el feedback sonoro sea audible. Las especificaciones INT-US-02 y INT-US-03 serán contempladas en la creación del gabinete en una sección posterior.

| ID | Descripión | Descripión | Valor esperado |
|---|---|---|---|
| MT-HW-01 | Verificación Wi-Fi | Verificar que el dispositivo se encuentra en la red local desde una computadora mediante la herramienta nmap de Linux | Conexión exitosa |
| MT-HW-02 | Tensiónes Vsw, Vr, Vbe | Con un voltimetro con error de +/- 0.05V medir las tensiónes en estos puntos | Tensiones y corrientes deben coincidir con la teoria |

Cuadro 13: Plan de pruebas HW adicionales - Módulo terminal

# 12. Software

El software es el desafío más grande del proyecto. El reconocimiento de voz es una tarea difícil y a la vez se debe tener en cuenta de que formas el proyecto puede evolucionar. El desarrollo de software debe permitir cambios o mejoras en cada etapa sin la necesidad de modificar otras.

## 12.1. Desarrollo del software

Existen algunas restricciones impuestas anteriormente. El procesamiento de voz debe realizarse en tiempo real, es decir que tiene que responder dentro de un rango de tiempo razonable. Esto causa limitaciones en el software, debido al límite en cantidad de operaciones por unidad de tiempo. La comunicación entre módulos establecida es mediante protocolo TCP/IP. Esto tiene como ventajas la posibilidad de tener varios equipos actuando como módulo central simultáneamente, redundancia de datos y manejo de errores. Esta elección considera el futuro uso de teléfono celular para control de luces, mediante voz o una aplicación, y una posible expansión a control de módulos desde la nube. En consecuencia, permitirá que el producto evolucione no sólo mediante una conexión remota, pero también a otras áreas, como por ejemplo seguridad del hogar.

Por otro lado, se quiere proveer la posibilidad de utilizar el mismo software con distinto protocolo de comunicación. De esta forma, se podrá ofrecer en un futuro adaptaciones particulares, incluyendo productos producidos por otras empresas, a clientes prometedores. Ejemplos de este tipo de clientes son hoteles o empresas de construcción de edificios.

## 12.2. Diagrama de bloques

A continuación se provee una representación gráfica de la disposición de las secciones de software para el módulo central y un diagrama de flujo del software del módulo terminal. Ambos serán explicados en más detalle en la siguiente sección.

### 12.2.1. Módulo central

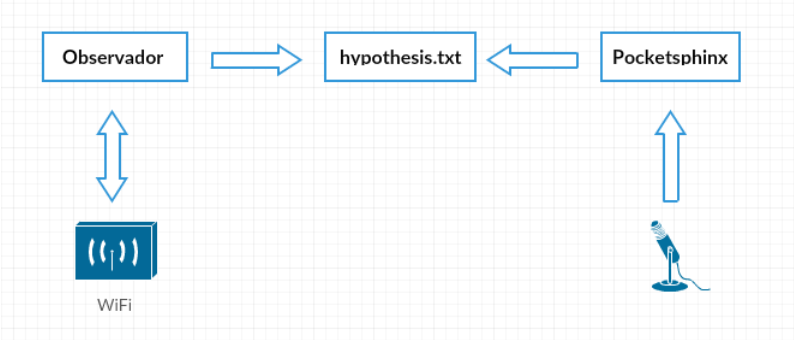



Figura 39: Módulo central

### 12.2.2. Módulo terminal

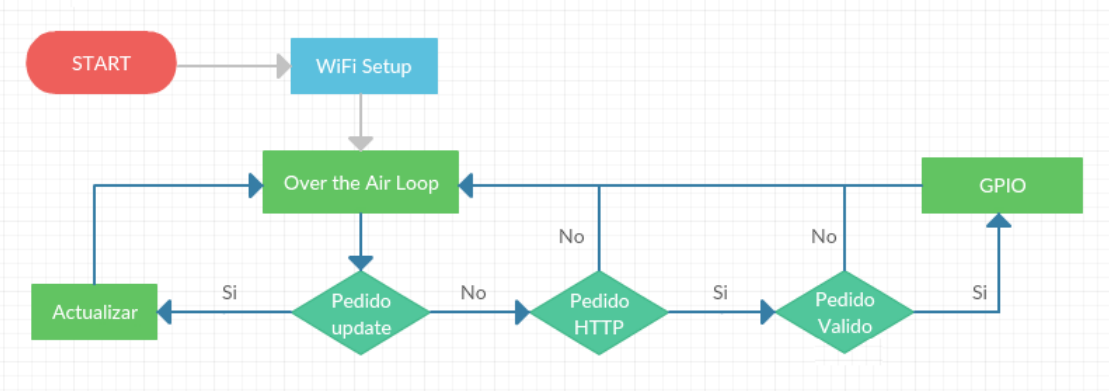



Figura 40: Módulo terminal

## 12.3. Descripción detallada

### 12.3.1. Módulo central

El módulo central está compuesto de una Raspberry Pi con un micrófono USB. La Raspberry Pi fue configurada con un sistema operativo basado en Linux, pero modificado particularmente para usar en una Raspberry Pi. Existen varias distribuciones, la utilizada es Raspbian Jessie, en su última versión lanzada en 05/07/2017.

Es importante utilizar la última versión por razones de seguridad. Los componentes mayormente explotados para ataques cyberneticos de tipo DDoS (Denial of Service) son elementos domésticos. Se utilizarán tres medidas para combatir y prevenir este tipo de ataques.

- Última versión de Raspbian Jessie: La última versión asegura la corrección de vulnerabilidades y bugs explotados conocidos por el equipo de ingenieros a cargo del desarrollo del sistema operativo Raspbian Jessie. Siempre que exista una nueva versión, se enviará un email al usuario para que considere los riesgos de no actualizar y ofrecer gratuitamente una actualización del sistema operativo.

- Cambio de contraseña: El sistema operativo tiene por default un usuario y una contraseña ampliamente conocida. Si bien resulta lógico y obvio el cambio de la misma, existen varios productos hoy en día que se venden con la contraseña default, siendo vulnerables a ataques maliciosos.

- Cierre del puerto Telnet (puerto 22): Este puerto permite el acceso remoto a una Raspberry Pi. Es utilizado frecuentemente y será configurado de forma tal que no acepte conexiones de ningún tipo. Normalmente, no sería accesible ya que el router debería bloquear el acceso externo a la red. Sin embargo, no es sorprendente que se encuentren vulnerabilidades en los routers que permitan el acceso a este puerto. A pesar que el protocolo SSH evita dar respuestas que den evidencia sobre la presencia de dispositivos es vulnerable a ataques de fuerza bruta.

Para la detección y seteo del micrófono se modifica directamente el driver de audio del sistema operativo. Este es llamado ALSA (Advanced Linux Sound Architecture) y es un componente del kernel de Linux. El objetivo del proyecto de ALSA es de hacer posible la configuración automática de tarjetas de sonido y el manejo de múltiples dispositivos de sonido en un solo sistema. Se puede encontrar su documentación oficial online en *www.alsa-project.org* 22.

De esta forma, se define un subsistema de micrófono en donde el sistema operativo se encarga de ajustar la frecuencia de sampleo a la utilizada, 16kHz.

El micrófono USB no ofrece diferente opciones de frecuencia de sampleo. Por esta razón el sistema operativo se encarga de reducir la misma desde 44.1kHz a 16kHz. Existen dos razones para el uso de 16kHz:

- Velocidad de cálculo: Debido a propiedades de la voz, 16kHz permite un análisis sin perdida de información y reduce significativamente el numero de cálculos. Utilizar 44.1kHz no permitiría el reconocimiento en tiempo real.

- Modelo de lenguaje español: El modelo utilizado fue entrenado con audio sampleado a 16kHz. Para utilizar el modelo correctamente se debe utilizar la misma frecuencia de sampleo.

Mediante la creación de un archivo llamado asound.conf, se configuró el subsistema de micrófono mediante la documentación de ALSA

```
pcm.usb {
type hw
card 1
}
pcm.internal {
type hw
card ALSA
}
pcm.!default {
type asym
playback.pcm {
type plug
slave.pcm "internal"
format U16_LE
rate 16000
}
capture.pcm {
type plug
slave.pcm "usb"
}
}
ctl.!default {
type asym
playback.pcm {
type plug
slave.pcm "internal"
}
capture.pcm {
type plug
slave.pcm "usb"
format U16_LE
rate 16000
}
}
```

Figura 41: Archivo asound.conf

En la Raspberry Pi, se utilizan dos programas:

- Observador
- Reconocimiento de voz: Pocketsphinx

Ambos se encuentran configurados de forma tal que al encender el dispositivo se ejecuten en la rutina de encendido del sistema operativo. De esta forma se genera la sensación de «plug and play» del lado del usuario y evitará fallas por cortes de luz.

La interfaz entre ambos programas se realiza mediante un archivo: 'hypothesis.txt'. Para la producción masiva, se crea una imagen de la memoria SD. Esto permite que se pueda preparar el software montando la imagen y sin la necesidad de utilizar la Raspberry Pi. Una vez montado se conecta la memoria y el módulo base está en condiciones de pasar a la etapa de pruebas.

**Observador** El observador es un programa que se ejecuta como un proceso de 'background'. Este se encuentra en modo de espera hasta que se realice un cambio en el archivo *'hypothesis.txt'*. Ante un cambio, observa el contenido del archivo y actúa de acuerdo al mismo.

Es el encargado de comunicar la acción a realizar mediante el protocolo definido, en este caso TCP/IP.

**Pocketsphinx** Este programa realiza el reconocimiento de voz. El proyecto pocketsphinx empezó como un trabajo de doctorado y hoy en día tiene varias personas participando de él. Permite utilizar diferentes idiomas e incluso adaptarlo a otras aplicaciones fuera de reconocimiento de voz.

Existen varias ventajas que llevaron a la elección del proyecto. Es un software de uso libre y por lo tanto no hay incidencia en los costos. Ya que se tiene una base de datos del idioma español completa, permitirá agregar o quitar comandos en el futuro. Actualmente, presenta una adaptación del programa para android. Esto permitirá que en un futuro se puedan emitir comandos de voz desde el teléfono celular. Si bien se optó por un módulo central, un gran número de personas estaban interesadas en utilizar el celular. Un relanzamiento del producto permitirá adaptar el programa fácilmente a este público.

El reconocimiento de voz no es fácil y uno de los mayores limitantes del mismo es la capacidad de procesamiento. Este proyecto tiene en cuenta esto y ofrece muchas funciones optimizadas para reducir el número de operaciones. Las características mas importantes del proyecto Pocketsphinx son:

- Portabilidad

- Simplicidad de implementación

- Eficiencia en utilización de memoria

- Seguridad de los subprocesos

En términos generales, existen dos grandes etapas en el procesamiento de datos: acondicionamiento de la señal y decodificador.

#### Acondicionamiento de la señal

Es extremadamente importante adecuar la señal de audio para su análisis. Mediante su correcto acondicionamiento, se consigue obtener resultados con mayor confianza, debido a que reduce la varianza en la detección final. Se aplican dos técnicas para realizar esta tarea: sustracción espectral y VAC (Voice Activity Classification).

La sustracción espectral se basa en la suposición de que la señal de voz contiene ruido aditivo, es decir, que es independiente de la señal. Además considera un espectro de ruido plano. Esta técnica tiene dos ventajas. Por un lado permite descartar tramas o 'frames' de audio en donde sólo hay ruido en el ambiente, evitando realizar procesamiento innecesario. La segunda ventaja es que elimina ruido en la señal, reduciendo la varianza en las mediciones. Una desventaja es que, si el valor de threshold utilizado es muy alto, puede eliminar o distorsionar la señal de voz. Debido a este efecto, existe un balance entre el valor de threshold y la tasa de fallas.

La detección de actividad de voz o VAC es una técnica muy favorable. Permite descartar tramas en donde no hay señal de voz. No sólo logra obtener una secuencia de estados más estables, sino que también reduce el tiempo de procesamiento.

La figura siguiente muestra el efecto del VAC. Hay una señal de voz con música de fondo. En rojo se observa secciones de la señal en donde hay detección de actividad de voz.

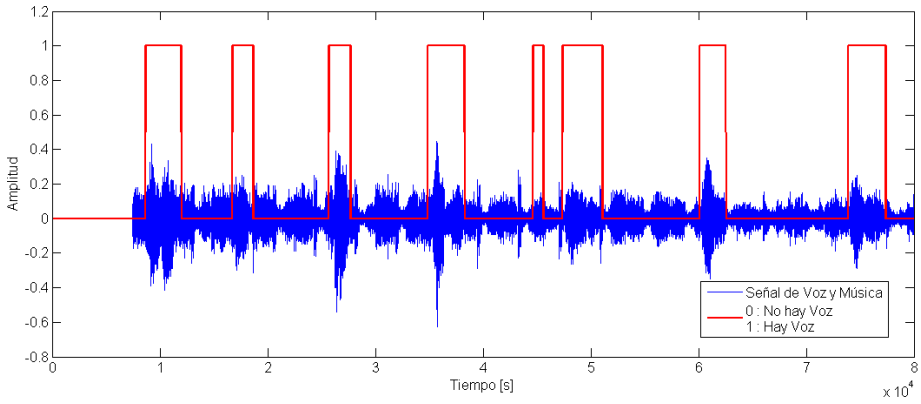



Figura 42: VAC

**Decodificador**

El decodificador posee una estrategia de tres pasadas para el análisis de la voz:

- fwdtree: Búsqueda con algoritmo Viterbi usando un árbol estático de léxicos.
- fwdflat: Búsqueda con algoritmo Viterbi usando un léxico plano
- Best path: Búsqueda 'word graph'

A su vez, se puede dividir al decodificador en tres módulos principales:

- Acustic Modeling
- Foward search
- Word graph search

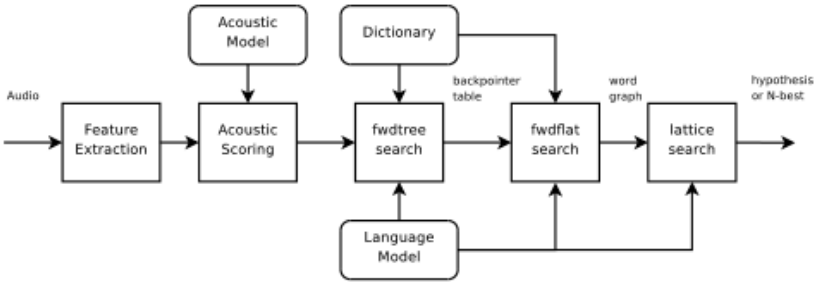



Figura 43: Decodificador

**Acustic Modeling** Modela el lenguaje en características a analizar para la detección de fonemas. Utiliza Gaussian Mixture Model (GMM), compartiendo atributos entre distintas Gaussianas para obtener una mayor eficiencia de computación. Estos modelos atados (llamado así por compartir atributos) se almacenan en un codebook, que se evalúa frente a cada dato de entrada del sistema. Usualmente prevalecen N<5 gaussianas con gran similitud, aunque ese parámetro es un argumento variable.

Para computar las densidades de los datos a evaluar, se toma la menor de las N Gaussianas en un instante o 'frame' anterior como threshold y se computan los nuevos valores. Si ningún valor en el siguiente frame supera el threshold, se asume que es el fin de una palabra. Ya que se puede obtener un mismo fonema alineando los estados de distintas formas, se modelan múltiples 'streams' con sus propios codebooks para luego calcular el puntaje total mediante el múltiplo de ellos. En consecuencia, se pueden representar estos números con menor resolución (8 bits) sin perder eficiencia. Utilizando el logaritmo se puede reemplazar el producto por suma, luego reemplazada por una 'lookup table' para evitar utilizar el CPU para hacer operaciones numéricas. Mediante esto último, el proyecto consigue una alta eficiencia en términos de velocidad.

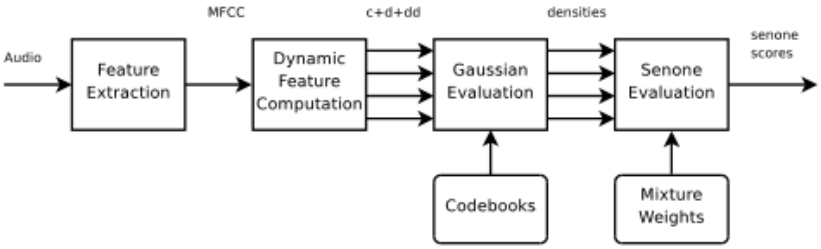



Figura 44: Modelo acústico

**Forward Search** Está compuesto por dos tareas: fwdtree y fwflat del programa pocketsphinx.

**fwdtree** Esta función realiza una búsqueda en un árbol lexical estructurado. Para ello, es necesario entrenar el sistema o bien utilizar una base de datos previamente entrenada. Divide la búsqueda en dos: un árbol lexical, que contiene las características de un modelo oculto de Markov (HMM 22) para representar palabras de multifonemas

y un arreglo de características HMM para fonemas independientes. Este último es un arreglo dinámico de fonemas HMM individuales, representando fonemas únicos y terminadores de palabras compuestas por varios fonemas.

Finalmente, utiliza también una tabla auxiliar que relaciona todas las palabras que comparten el mismo fonema, siempre y cuando no sea el fonema final.

**fwflat** Esta función establece un nuevo puntaje al lattice generado por fwdtree. Sin embargo, no parte del puntaje establecido anteriormente. Utiliza un set de palabras activo durante la primera pasada dentro de un rango de tiempo establecido. Crea una tabla con cada modelo HMM en el cual el fin de la palabra anterior podría ser una variable de entrada para el inicio de la palabra siguiente. Genera una lista corta de palabras que empiezan en cada frame y luego guarda las que persisten por una cierta cantidad de tiempo o frames. Compara también las palabras entradas en frames anteriores y posteriores. Esto sirve de apoyo a la búsqueda de fwdtree y permite detectar distintas frases o palabras a medida que se analiza más información, ya que esto ocurre una vez cada cierta cantidad de frames al mismo tiempo que fwdtree sigue analizando.

**Word graph search** Finalmente se utiliza el algoritmo de Dijkstra para establecer la opción de menor camino, ya que asume que el camino es la suma acumulada de las probabilidades para cada posible frase detectada anteriormente.

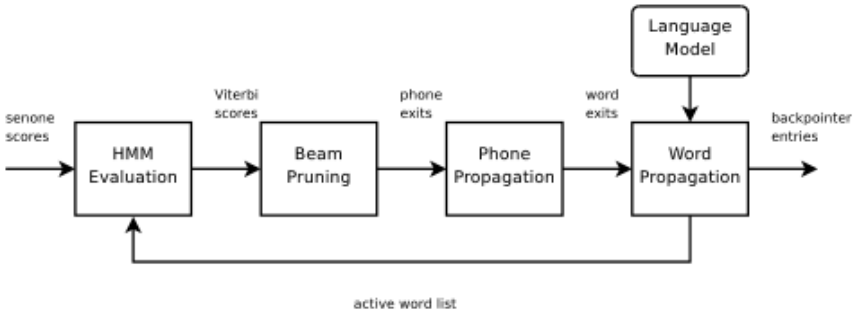



Figura 45: Búsqueda de palabras

Utilizando la API (Aplication Programming Interface 22) y documentación provista, se creó un programa de detección continua. Mediante unas pequeñas adaptaciones en el código y aprovisionamiento de un modelo de lenguaje español se configuró de tal forma que detecte los comandos definidos.

Para reducción de tiempos, se limitó el diccionario a las palabras utilizadas: **prende, apaga, luz, uno, dos y tres**.

Para la generación de las frases se creó un script en formato JSGF (JSpeech Grammar Format) para que el programa busque solo las secuencias requeridas.

*#JSGF V1.0;*
*grammar light_basic; public <light_basic> = <do> (<individual>);*
*<individual> = <object> <number>;*
*<do> = prende | apaga;*
*<object> = luz;*
*<number> = uno | dos | tres;*

Figura 46: Archivo JSGF

Finalmente, al detectar una frase posible según el archivo JSGF se escribirá el contenido en un archivo llamado *'hypothesis.txt'*.

### 12.3.2. Módulo terminal

El programa default en el microcontrolador Adafruit Huzzah tiene la posibilidad de agregar código programado en el lenguaje LUA. Una consideración que se tuvo en cuenta es que para tal acción se necesita una conexión física a los pines de UART. Teniendo en cuenta la visión global del producto, se eligió utilizar otro firmware. Este es provisto por una librería de Arduino y adaptado por Platformio.io 22. Es open source y permite actualizar código mediante Wi-Fi 22.

La instalación de firmware se hará en la oficina. Para ello es necesario utilizar el puerto UART, mediante una conexión física. Este nuevo firmware permitirá cargar nuevas configuraciones a través de Wi-Fi. La configuración final del módulo se ajustará de acuerdo a la configuración de ruteo que el usuario presente. Estos son los parámetros mínimos para que el módulo pueda conectarse a la red local:

- SSID y password: Nombre y contraseña de la red local.
- Gateway: Este numero representa la dirección IP del router y permite la comunicación entre módulos a través del mismo. El usuario puede proveer esta información. Se le asistirá, de ser necesario, mediante servicio técnico.
- Subnet: Generalmente es 255.255.255.0, pero si el usuario tiene una configuración avanzada debe proveer esta información. La mascara de subnet define la cantidad de usuarios o equipos que pueden existir en una red, es decir, el tamaño máximo de la red.

El programa cargado está esperando un pedido de conexión constantemente. Cuando recibe información, en forma de HTTP Get 22, analiza el pedido y realiza la acción pedida.

Por ejemplo: HTTP Get -> '/light1/on'

La respuesta ante este pedido será 200, que según el protocolo HTTP significa 'OK'. En caso de que sea un pedido invalido la respuesta será 404, 'Page Not Found'.

## 12.4. Plan de pruebas

### 12.4.1. Módulo central

Inicialmente, se realizará un plan de pruebas solamente para el reconocimiento de voz (no detecciones, detecciones falsas y erróneas). Para el mismo, se medirá la tasa de errores en las detecciones en un periódo de tiempo mayor a 200 horas. Se propone realizar las mismas pruebas con un grupo de 30 personas en donde debería haber aproximadamente 15 de cada sexo. Dentro de cada grupo, se debe buscar gente de distintas edades con diferente acentos que residan en la provincia de Buenos Aires. Esto tiene en cuenta diferencia de acentos entre barrios y, preferentemente, un pequeño grupo debe haber crecido en Argentina fuera de la provincia de Buenos Aires. Estos últimos presentarán los casos más extremos en cuanto a acento y dialecto.

A continuación se define el plan de acción para el módulo central junto con pruebas adicionales. La interfaz mediante el archivo *'hypothesis.txt'* permite realizar pruebas en el programa del observador y pocketsphinx por separado, al igual que verificar la interfaz entre los mismos.

| Módulo central | | | |
|---|---|---|---|
| **Origen** | **Nombre** | **Descripión** | **Valor esperado** |
| FUN-01 | Tasa de errores: Falsa detección | Medir la tasa de Falsa detección en 350 horas de uso | < 1/hora |
| FUN-02 | Tasa de errores: Detección erroneas | Medir la tasa de Falsa detección para 200 comandos | < 30% |
| FUN-03 | Tasa de errores: No detección | Medir la tasa de No detección para 200 comandos | < 40% |
| FUN-04 | Delay en detección del comando de voz | Medir el promedio de tiempo que tarda desde que se termina de decir un comando a que el programa lo reconozca en 50 comandos | < 3s |

Cuadro 14: Plan de pruebas SW con origen - Módulo central

| **ID** | **Nombre** | **Descripión** | **Valor esperado** |
|---|---|---|---|
| MC-SW-01 | Verificación Programa pocketsphinx | Emitir un comando y se verificar que el archivo de hypothesis.txt sea modificado con el comando deseado | Reconozca un comando y lo escriba en el archivo 'hypothesis.txt' |
| MC-SW-02 | Verificación Programa observador | Modificar el archivo hypothesis.txt directamente y verificar que identifique el cambio y prepare el pedido HTTP | Interprete cambios en el archivo 'hypothesis.txt' |

Cuadro 15: Plan de pruebas SW adicional - Módulo central

### 12.4.2. Módulo terminal

Para este módulo se generará un programa básico que conteste ante pedidos 'HTTP Get' mediante Wi-Fi. Este pedido se puede realizar mediante código o directamente en el explorador. Adicionalmente, se definirá una prueba que verifique que se puedan realizar actualizaciones Over the Air.

| **ID** | **Nombre** | **Descripión** | **Valor esperado** |
|---|---|---|---|
| MT-SW-01 | Actualizar el software Over the Air | 1. Invertir la tensión del PIN GPIO y actualizar el programa por medio de PlatformIO<br>2. Verificar que el sistema se comporte opuesto<br>3. Actualizar el programa con la tensión del PIN GPIO correcta | - |
| MT-SW-02 | Reciba e interprete comandos HTTP | Generar un comando desde el explorador y observar la respuesta en el mismo | - |

Cuadro 16: Plan de pruebas SW - Módulo terminal

### 12.4.3. Sistema completo

Finalmente, se definen ciertas pruebas para el sistema completo.

| Origen | Descripión | Descripión | Valor esperado |
|---|---|---|---|
| FUN-05 | Modularidad: Módulos con funcionalidad independiente | Verificar que al desconectar un módulo el sistema no se interrumpa. | - |
| FUN-06 | Modularidad: Los programas deben andar al conectar los módulos | Desconectar y conectar los módulos verificando que el sistema funcione | - |
| INT-COM-01 | Interfaz inalámbrica entre módulos | - | - |
| INT-COM-03 | Delay de transmición | Medir el delay promedio de transmición al modificar directamente el archivo 'hypothesis.txt' en 50 casos | < 3 segundos |
| INT-COM-04 | Alance de comunicación para ambos módulos | 1. Comprobar que los módulos funcionen<br>2. Alejar los módulos y repetir hasta que no funcionen | > 15 m |

Cuadro 17: Plan de pruebas SW - Sistema completo

# Parte VI

# Construcción del prototipo

La construcción del prototipo consistirá en la creación de un sistema básico anteriormente definido, compuesto por un módulo central y dos terminales. Se verificará el correcto funcionamiento del software y hardware con el fin de poder asegurar que el sistema cumpla con las expectativas de diseño.

## 13. Definición de los módulos

### 13.1. Módulo central

El módulo central está compuesto por la Raspberry Pi con un gabinete diseñado para la misma. Adicionalmente, tiene un micrófono conectado al puerto USB. Este módulo no requiere circuitos adicionales ni diseño de gabinete.

### 13.2. Módulo terminal

El módulo terminal está compuesto por dos circuitos: Adafruit Huzzah y circuito de relay. Se debe crear un gabinete que contenga ambos circuitos y asegure que toda conexión de 220VAC no pueda dañar al usuario bajo ninguna condición, restricción establecida en la factibilidad legal.

## 14. Diseño de los circuitos impresos

### 14.1. PCB del circuito de relay

El diseño del circuito de relay consiste en una capa de 51.05 x 25.65 mm y 1.6mm de espesor. Se considera que por un lado del circuito del relay hay una conexión de 5V de continua y por el otro 220 VAC de alterna. El conector micro-USB para la alimentación de continua debe estar en el borde de la placa. De esta forma, el gabinete podrá tener una apertura para que el usuario pueda conectar la alimentación con facilidad.

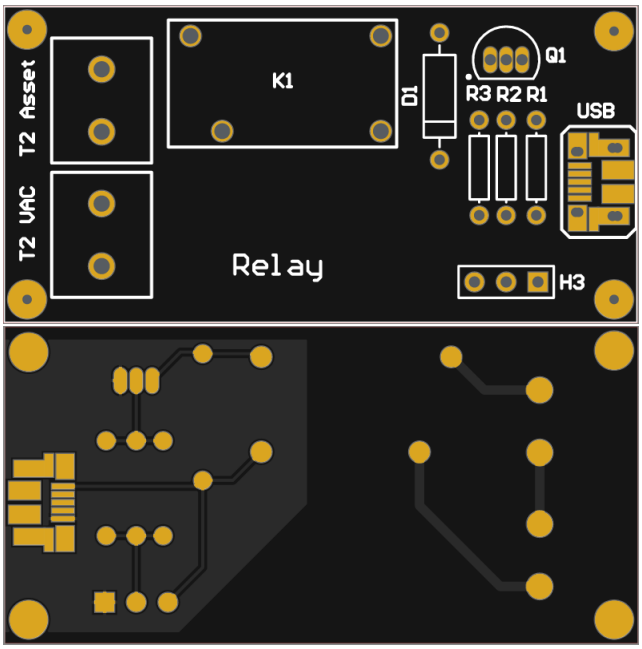



Figura 47: PCB circuito de relay

# 15. Diseño mecánico

## 15.1. Módulo terminal

El diseño mecánico fue realizado en el programa FreeCad. Este consiste en un gabinete de 8.02 x 10.16 cm de 2 mm de grosor. El diseño asegura que los circuitos estén fijos dentro del gabinete y contiene tres aberturas. Por un lado se encuentra el conector micro-USB y por el otro dos conectores para tensión alterna 220VAC y dispositivo a controlar. Estos últimos son dos componentes “snap-in”, es decir que encajan en el agujero establecido.

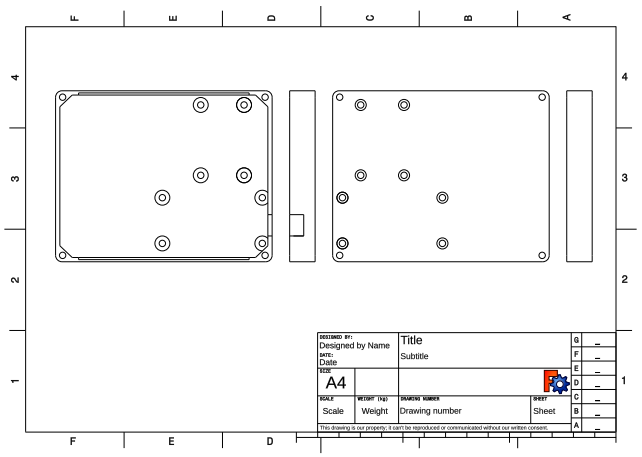



Figura 48: Diseño CAD base

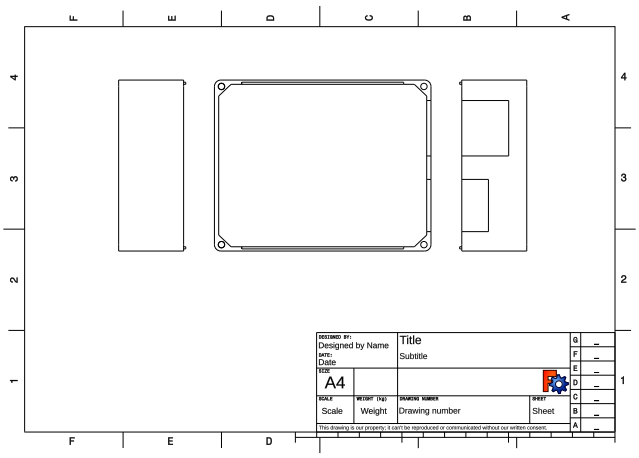



Figura 49: Diseño CAD tapa

El gabinete consiste en dos partes. La base contiene los agujeros para sujetar ambas placas y cuatro agujeros adicionales para unir la tapa mediante tornillos. Adicionalmente, presenta una abertura para la conexión USB. La tapa contiene las dos aberturas mencionadas anteriormente para los componentes snap-in.

Dado que el grosor de los bordes es de 2mm y no presenta ninguna protección en particular frente a exposición de agua, el gabinete presenta un IP de 40 según el estandard IEC 60529.

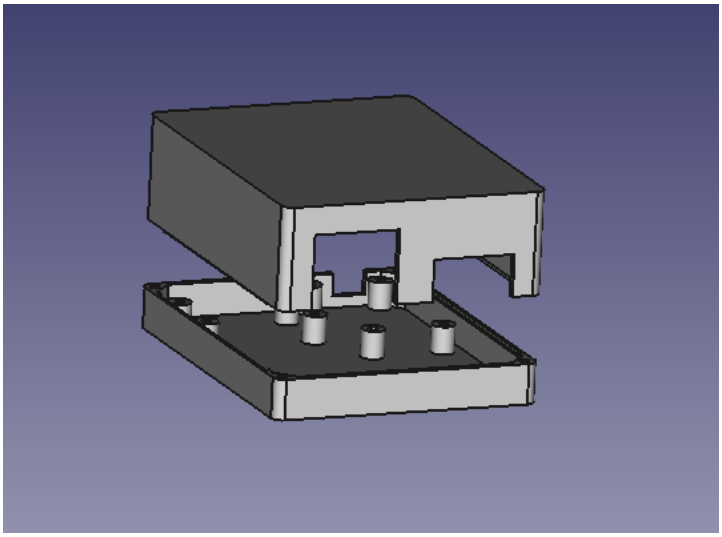


Figura 50: CAD 3D

# Parte VII

# Validación del prototipo

## 16. Validación de hard

### 16.1. Plan y protocolos especiales de medición

#### 16.1.1. Módulo central

Teniendo en cuenta el plan de pruebas establecido en la sección 11.4.1 se realizarón las siguientes mediciones:

| Prueba | Medición | Valor esperado | Validación |
|---|---|---|---|
| Peso módulo central | 17 g | < 2kg | OK |
| Alcance del micrófono | 13 m | > 10 m | OK |
| Módulo central: Transformador 220VAC/5VDC | 5.1V | 4.5 – 6 V | OK |
| Verificación micrófono | Sonido audible | Sonido audible | OK |
| Verificación Wi-Fi | Conexión exitosa | - | OK |
| Verificación puerto Ethernet | Conexión exitosa | - | OK |

Cuadro 18: Validacion HW - Módulo central

#### 16.1.2. Módulo terminal

Similar al caso anterior, se realizaron las siguientes mediciones.

| Prueba | Medición | Valor esperado | Validación |
|---|---|---|---|
| Dimensión máxima módulo terminal | 11x8.5x4.3 cm | < 15x15x5 cm | OK |
| Peso módulo terminal | 12 g | < 2kg | OK |
| Indicador de acción sonoro: Switch mecánico | Audible | Audible | OK |
| Protección de tensión peligrosas en módulo central: gabinete con espesor | 2mm | > 1mm | OK |
| Protección de tensión peligrosas en módulo central: Zocalo de conección | Cumple por diseño | - | OK |
| Módulo terminal: Transformador 220VAC/5VDC | 5.0V | 4.5 – 6 V | OK |
| Verificación Wi-Fi | Conexión exitosa | - | OK |
| Módulo terminal: Resistencia de salida < 0.5 Ohm | 0.4 Ohm | < 0.5 Ohm | OK |

Cuadro 19: Validación HW - Módulo terminal

## 17. Validación de soft

### 17.1. Módulo central

La siguiente tabla resume los resultados obtenidos de la validación. Luego se mencionan detalles contemplados en las fallas relacionadas al procesamiento de voz. Para estas pruebas, se utilizó el sistema en un ambiente de 8 x 5 metros con dos personas, una de cada sexo.

| Prueba | Medición | Valor esperado | Validación |
|---|---|---|---|
| Tasa de errores: Falsa detección | 0.04/hora | < 1/hora | OK |
| Tasa de errores: Detección erronea | 9.50% | < 30% | OK |
| Tasa de errores: No detección | 3.00% | < 40% | OK |
| Delay en detección del comando de voz | 2 segundos | < 3s | OK |
| Verificación Programa pocketsphinx | Exitosa | - | OK |
| Verificación Programa observador | Exitosa | - | OK |

Cuadro 20: Validación SW - Módulo central

**Falsa detección** Para esta falla se coloca la Raspberry Pi con el programa abierto por 350 horas en un ambiente doméstico. No se emite ningún tipo de comando durante este tiempo y se cuentan la cantidad de comandos detectados. En total hubo 45 fallas resultando en una tasa de falsa detección de 0.12 fallas por horas. Tener en cuenta que el archivo SJGF utilizado analiza comandos para tres luces mientras que el sistema básico presenta solo dos (esto permite agregar un módulo sin actualizar el archivo). Además, aproximadamente en la mitad de los casos el comando falsamente detectado no realiza ninguna acción. El usuario no percibirá una falla en donde la falsa detección intenta realizar un comando igual al estado de la luz. Por ejemplo, apagar una luz cuando previamente está apagada. Teniendo en cuenta estas consideraciones, se espera percibir este tipo de fallas en un sistema básico $0{,}12\frac{2}{3}\frac{1}{2} = 0{,}04$ veces por hora.

**No detección y detección errónea** Ambas fallas se pueden validar al mismo tiempo. Se valida la detección de comandos instantáneamente después de emitirlos y se lleva un registro de cuantas detecciones fueron correctas, erróneas y no detectadas. Para lograr un resultado que considere distintas condiciones de ruido se realiza esta validación varias veces en diferentes horarios.

Para 200 comandos emitidos se obtuvieron 19 detecciones erróneas y 6 no detecciones, obteniendo tasas de 0.095 y 0.03 respectivamente. Notar nuevamente que algunas de estas detecciones no realizarán ninguna acción, similar al caso de detección falsa.

## 17.2. Módulo terminal

Se realizan dos pruebas para validad el software en el módulo central:

| Prueba | Medición | | Validación |
|---|---|---|---|
| Actualizar el software Over the Air | Exitosa | - | OK |
| Reciba e interprete comandos HTTP | Exitosa | - | OK |

Cuadro 21: Validación SW - Módulo terminal

A continuación se muestra mas detalle del pedido HTTP mediante el explorador. Se tiene en cuenta que el ip del módulo terminal es 10.0.0.13 y, por lo tanto, un pedido para encender la luz se puede realizar ingresando 10.0.0.13/light1/on y la respuesta ante el pedido será presentada en pantalla. La siguiente imagen pone en evidencia esto ultimo. Para este caso se utilizó el explorador Firefox, aunque el resultado es independiente del mismo.

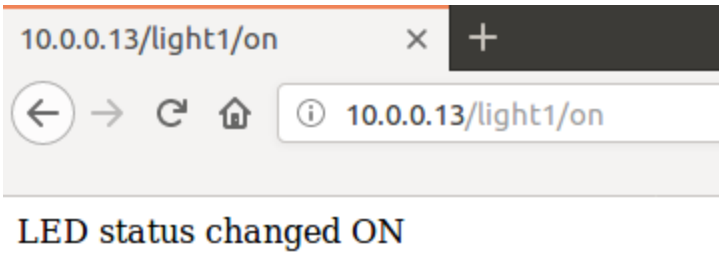



Figura 51: Pedido HTTP desde el explorador

## 17.3. Sistema completo

La siguiente tabla resume los resultados de la validación para el sistema completo.

| Prueba | Medición | | Validación |
|---|---|---|---|
| Modularidad: Módulos con funcionalidad independiente | Exitosa | - | OK |
| Modularidad: Los programas deben andar al conectar los módulos | Exitosa | - | OK |
| Interfaz inalámbrica entre módulos | Exitosa | - | OK |
| Delay de transmición | < 1 segundo | < 3 segundos | OK |
| Alance de comunicación para ambos módulos | 20 m | > 15 m | OK |

Figura 52: Validacion - Sistema completo

Para la interfaz inalámbrica se grabó la siguiente traza con el programa tcpick de Linux. La misma fue el resultado del comando de voz: “prende luz uno”. La ip de la Raspberry Pi en la traza es 10.0.0.8 y la ip del módulo terminal es 10.0.0.12.

```
Starting tcpick 0.2.1 at 2017-10-03 14:22 EDT
Timeout for connections is 600
tcpick: reading from light_one_trace
1       SYN-SENT        10.0.0.8:49640 > 10.0.0.12:http
1       SYN-RECEIVED    10.0.0.8:49640 > 10.0.0.12:http
1       ESTABLISHED     10.0.0.8:49640 > 10.0.0.12:http
GET /light1/on HTTP/1.1
Host: 10.0.0.12
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.9.1

HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE HTML>
<html>
LED status changed ON
</html>
1       FIN-WAIT-1      10.0.0.8:49640 > 10.0.0.12:http
1       TIME-WAIT       10.0.0.8:49640 > 10.0.0.12:http
1       CLOSED          10.0.0.8:49640 > 10.0.0.12:http
tcpick: done reading from light_one_trace

9 packets captured
1 tcp sessions detected
```

Figura 53: Traza

Se observa que el programa establece la conexión 22 y envía en formato HTTP “GET /light1/on”, junto a información adicional específica del protocolo HTTP. La respuesta ante el comando enviado fue 200, que significa OK y

texto adicional “LED status changed ON”.

Finalmente, se realiza una última validación a la modularidad del sistema completo. En primer lugar se verifica su correcto funcionamiento bajo condiciones normales para luego realizar desconexiones a los módulos, de forma tal que el sistema no pierda su configuración.

- Reseteo módulo central: Se desconectó y conectó el módulo base. Luego de esperar 30 segundos para asegurar que haya terminado el proceso de booteo se emitió el comando el “prender luz uno” y se verificó que la luz se encendiera.
- Reseteo módulo terminal: Se desconectó y conectó el módulo terminal. Se emitió el comando de “prender luz uno” y se verificó que la luz se encendiera.
- Por último se emitió un comando de “prender luz uno” con el módulo terminal desconectado. Se volvió a conectar el módulo terminal y la acción no fue realizada. Al realizar nuevamente el comando se prendió la luz exitosamente, demostrando que el sistema no queda en un estado indefinido al no poder realizar una acción.

El sistema completo pasó todas las pruebas exitosamente y comprueba que funciona correctamente.

# Parte VIII

# Estudios de confiabilidad de hard y de soft

# Confiabilidad de hard y de soft

## 18. Hardware

Inicialmente, se define que el sistema no se considera en falla si al menos uno de los módulos terminales funciona. La razón detrás de esto es que la posibilidad de controlar una luz representa la funcionalidad mínima del sistema. Por lo tanto, debe funcionar el módulo base y al menos un módulo terminal.

El modelo propuesto será el conjunto de un modelo serie entre módulo base y terminales, junto con un modelo n,k entre los terminales. Dentro de los módulos terminales se propone un modelo serie entre el circuito de relay y la placa Adafruit Huzza, ya que si falla uno de ellos falla el terminal por completo. Para el caso del sistema básico se tomará en cuenta que funcione al menos uno de los dos módulos terminales, siendo $n = 2$ y $k = 1$.

Para estimar los valores de MTBF se utiliza como referencia la norma MIL-HDBK-217F.

El cálculo del MTBF para el circuito del relay se presenta en la siguiente figura, donde el cálculo de $\lambda_p = \frac{1}{MTBF}$ se obtiene mediante la ecuación $\lambda_p = \lambda_b \prod \pi_i$.

Se definen:

$\lambda_b$ Tasa de fallas base

$\pi_E$ Factor de medio ambiente

$\pi_Q$ Factor de calidad

$\pi_F$ Factor de aplicación para relays

$\pi_{CYC}$ Factor de ciclos

$\pi_C$ Factor de contacto

$\pi_L$ Factor de estrés mecánico

$\pi_T$ Factor de temperatura

$\pi_R$ Factor de resistencia

$\pi_S$ Factor de estrés eléctrico

$\pi_A$ Factor de aplicación para diodos y transistores

| | λp | λb | πT | πS | πQ | πE | πC | πR | πA | πL | πCYC | πF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Diodo | 0.00492 | 0.0038 | 3 | 0.054 | 8 | 1 | 1 | | | | | |
| Transistor | 0.00045 | 0.00074 | 2.1 | 0.11 | 8 | 1 | | 0.47 | 0.7 | | | |
| Resistor | 0.00005 | 0.00015 | | | 0.3 | 1 | | 1 | | | | |
| Relay | 7.21224 | 0.006 | | | 1 | 1 | 1 | | | 4.77 | 36 | 7 |
| **Total** | **7.21766** | | | | | | | | | | | |

Figura 54: Cálculo $\lambda_p$ circuito de relay

Ambos módulos están compuestos por microcontroladores. El cálculo de $\lambda_p$ se encuentra resumido en la siguiente figura, donde cada valor se obtiene como $\lambda_p = (C_1\pi_T + C_2\pi_E)\pi_Q\pi_L$

Esta formula (norma MIL-HDBK217F parte 5.1) define $C_1$ y $C_2$ como tasa de fallas de complejidad y tasa de falla de paquete respectivamente.

| | λp | C1 | πT | C2 | πE | πQ | πL |
|---|---|---|---|---|---|---|---|
| Raspberry Pi | **0.05895** | 0.56 | 0.1 | 0.0059 | 0.5 | 1 | 1 |
| Adafruit Huzzahh | **0.05760** | 0.24 | 0.1 | 0.048 | 0.5 | 1 | 1.2 |

Figura 55: Cálculo $\lambda_p$ microcontroladores

Finalmente se estima el MTBF para cada módulo. En el caso del módulo terminal, el valor final es la suma de los lambdas del circuito del relay y microcontrolador

| | λp | MTBF |
|---|---|---|
| Módulo terminal | **7.27526** | 0.13745 |
| Módulo base | **0.05895** | 16.96353 |

Figura 56: MTBF

Para estimar el MTBF de los terminales se procede a calcular:

$$R_{(t)_s} = \sum_{i=k}^{n} \binom{n}{i} R_{(t)}^{i} (1 - R_{(t)}^{n-i})$$

Donde n=2, k=1 y $R(t) = e^{-t\lambda_{terminal}}$. Reemplazando se obtiene:

$$R_{(t)_s} = 2R_{(t)}(1 - R_{(t)}) + R_{(t)}^2 = 2R_{(t)} - R_{(t)}^2$$

El MTBF está definido como: $\int_0^\infty R_{(t)_s} dt = \frac{2}{\lambda} - \frac{1}{2\lambda} = \frac{3}{2\lambda} = \frac{3}{2} MTBF_{terminal} = \frac{3}{2} 0{,}13745 = 0{,}206175$

Finalmente, ya que se propone un modelo serie entre módulo base y terminales se obtiene la suma de ambos:

$$MTBF_T = MTBF_{base} + MTBF_{terminales} = 16{,}96353 + 0{,}206175 \approx 17{,}17$$

El valor obtenido, de 17.17 fallas cada 10^6 horas, significa que se espera tener $\frac{17{,}17 \times 24 \times 365}{10^6} = 0{,}1504$ fallas por año, valor menor a una falla en el ciclo de vida del proyecto.

# 19. Software

Para obtener una estimación inicial del MTBF se utilizará el modelo de MUSA. La ecuación de predicción es:

$$\lambda_o = kpw_o$$

$k$ es una constante cuantizada para la estructura dinámica del programa y de las máquinas. Un valor para adoptar si no se conoce el mismo es $4{,}2x10^7$

$p = r/SLOC/ER$ estima el número de ejecuciones por por unidad de tiempo, donde r es el promedio de la velocidad de ejecución de instrucciones, SLOC la cantidad de líneas de código fuente y ER es una constante dependiendo del lenguaje, en este caso 2.5 por ser el lenguaje C.

$w_o$ es la estimación del número inicial de fallas, una estimación de la misma esta dada por 6 fallas / 1000 SLOC

En el módulo base, la Raspberry Pi tiene un clock de 700 MHz y se estima que cada línea de código en C tiene 7 instrucciones. Por lo tanto se estiman unas 700 MHz/7 instrucciones por segundo. Similarmente, en el módulo terminal, la velocidad del clock en el ESP8266 es de 80 MHz, obteniendo 80 Mhz/7 instrucciones por segundo.

La cantidad de líneas de código en módulo base y terminal se estiman en 13000 y 490 respectivamente.

Por lo tanto:

$$\lambda_{base} = 4{,}2x10^{-7} \times \frac{700/7}{13000/2{,}5} \times \frac{6}{100} = 4{,}84615x10^{-10}/segundo$$

$$\lambda_{terminal} = 4{,}2x10^{-7} \times \frac{80/7}{490/2{,}5} \times \frac{6}{100} = 1{,}46938x10^{-9}/segundo$$

Llevando los valores anteriores a fallas cada 10^6 horas, se obtiene $\lambda_{base} = 1{,}7446$ y $\lambda_{terminal} = 5{,}2897$. Por lo tanto, la estimación inicial de MTBF será de $MTBF_{base} = \frac{1}{1{,}7446} \simeq 0{,}5732$ y $MTBF_{terminal} = \frac{1}{5{,}2897} \simeq 0{,}189$ para los módulos base y terminal respectivamente.

La estimación de MTBF se seguirá realizando durante el primer año del proyecto para obtener una mejor estimación de la misma. Se propone el modelo de Duane para esta tarea. El modelo se basa en la relación logarítmica entre el tiempo entre fallas acumulativo versus el tiempo acumulativo. Al graficar esta relación, Duane propone que se puede representar mediante la ecuación de una recta. Luego, la cantidad de fallas obtenidas para la recta donde y=1 será la estimación del MTBF.

El tiempo entre fallas acumulativo esta dado por la cantidad de tiempo acumulativo dividido la cantidad de fallas acumulativas. Inicialmente, se buscará obtener al menos dos fallas para poder hacer una primera estimación de una recta.

La pendiente de la misma estará dada por:

$$\alpha = \frac{ln(m_2) - ln(m_1)}{ln(T_2) - ln(T_1)}$$

donde $m_i$ serán los tiempos entre fallas acumulativos y $T_i$ el tiempo acumulativo, ambos para la anotación $i$.

Una vez obtenida la pendiente se puede obtener el valor estimado de MTBF mediante:

$$MTBF = \frac{m_i}{T_i^{\alpha}}$$

Al obtener más valores, se buscará ajustar el modelo mediante regresión lineal.

# Parte IX

# Conclusión

## 20. Objetivos alcanzados

El trabajo presenta el desarrollo de una primera solución a la integración de procesamiento de voz con control de luces en ambientes de tipo domésticos. Más importante, cada parte o módulo del proyecto fue diseñado para que sea independiente y escalable. En consecuencia, permite utilizar y adaptarse frente a diferentes condiciones e incluso rediseñar cada módulo por separado.

Se comprueba la factibilidad del proyecto ante varios aspectos. El análisis legal contiene todo lo necesario para que el producto pueda ser distribuido en Argentina. El análisis económico presenta un plan de la estructura del negocio contemplando costos y ganancias a lo largo del ciclo de vida propuesto. La ingeniería en detalle, construcción y validación del prototipo ponen en evidencia la viabilidad tecnológica del producto definido al inicio del informe. Finalmente, la factibilidad temporal muestra el plan llevado a cabo para la realización completa del trabajo.

## 21. Recomendaciones para futuros diseños

### 21.1. Reconocimiento de Voz

El programa utilizado permite expandir el reconocimiento de voz a nuevo vocabulario, frases e incluso a interpretar lo que el usuario quiere realizar sin establecer un comando específico para tal acción. Cada una de estas mejoras requiere mayor capacidad de procesamiento, lo cual establece una relación de compromiso con el tiempo esperado para realizar cada acción. A medida que la tecnología mejore se podrán incluir mas tareas en el procesamiento de voz.

Por otro lado, hoy en día, existen nuevas técnicas muy populares para la tarea de procesamiento de voz, basadas en redes neuronales profundas (deep neural networks). Estas presentan un mejor rendimiento pero requieren mayor cantidad de datos para entrenar. Una propuesta para futuros diseños consiste en crear y almacenar información (como grabaciones de voces) para poder re-entrenar, re-enforzar o migrar a nuevas técnicas de procesamiento de voz.

### 21.2. Módulo terminal

Una próxima mejora en este módulo sería reemplazar el relay por un dimmer. El producto presentaría un valor agregado extra: permitir dimerizar cualquier luz. Esto significa también agregar comandos para dimerizar como por ejemplo: “Bajar luz uno”

## 21.3. Módulo base

Teniendo en cuenta el resultado de las encuestas en el proyecto, hubo un grupo de gente interesado en controlar luces con su teléfono celular. El programa utilizado está preparado para migrar al teléfono y no sería una tarea dificultosa. Sin embargo, se querrá agregar una opción extra: la posibilidad de controlar las luces desde el celular sin comando de voz.

Adicionalmente, esta expansión podría permitir que el sistema se lleve a la 'nube', es decir, que se puedan controlar las luces desde afuera del hogar.

## 21.4. Sistema completo

Si bien este proyecto se concentra en controlar luces. Es posible expandirlo a cualquier elemento del hogar que requiera prender o apagar. Sería interesante adaptar los controles a otros elementos como por ejemplo persianas, horno, aire acondicionado, etc.

# Parte X

# Anexos

## 22. Referencias


**[1] Tecnologías Digikey: https://www.digikey.com/**

**[2] SAS: https://www.argentina.gob.ar/**

**[3] Areatres: https://areatresworkplace.com/**

**[4] Lamaquinita: https://www.lamaquinita.co/**

**[5] Weword: https://www.wework.com/**

**[6] Hidden Markov Model**

**[7] Pocketsphinx Paper**

**[8] CMUSphinx: https://cmusphinx.github.io/**

**[9] RaspberryPi: https://www.raspberrypi.org/**

**[10] ESP8266: http://espressif.com/**

**[11] Adafruit-Huzzah: https://learn.adafruit.com/**

**[12] ALSA: http://www.alsa-project.org/**

**[13] Pocketsphinx API: https://cmusphinx.github.io/**

**[14] PlatformIO: http://docs.platformio.org/**

**[15] Arduino_OTA (Over the Air): https://github.com/esp8266/**

**[16] TCP/IP**

**[17] HTML**

**[18] Reporte Inmoviliario**

# 23. Módulo terminal

## 23.1. Código Adafruit Huzzah

```
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#define PIN 4
#define HIGH 1
#define LOW 0
// Devices
int N = 1;
char *dev[] = {"/light1"};
// Wifi settings
const char* ssid = "XXXXXXX";
const char* password = "XXXXXXX";
IPAddress ip(10,0,0,13);
IPAddress gateway(10,0,0,1);
IPAddress subnet(255,255,255,0);
// Server instance on port 80
WiFiServer server(80);
void setup() {
Serial.begin(115200);
```

```
Serial.println("Booting");
WiFi.config(ip, gateway, subnet);
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
while (WiFi.waitForConnectResult() != WL_CONNECTED) {
Serial.println("Connection Failed! Rebooting...");
delay(5000);
ESP.restart();
}
// Port defaults to 8266
// ArduinoOTA.setPort(8266);
// Hostname defaults to esp8266-[ChipID]
// ArduinoOTA.setHostname("myesp8266");
// No authentication by default
// ArduinoOTA.setPassword("admin");
// Password can be set with it's md5 value as well
// MD5(admin) = 21232f297a57a5a743894a0e4a801fc3
// ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");
ArduinoOTA.onStart([]() {
String type;
if (ArduinoOTA.getCommand() == U_FLASH)
type = "sketch";
else // U_SPIFFS
type = "filesystem";
// NOTE: if updating SPIFFS this would be the place to unmount SPIFFS using SPIFFS.end()
Serial.println("Start updating " + type);
});
```

```
ArduinoOTA.onEnd([]() {
Serial.println("\nEnd");
});
ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
Serial.printf("Progress: %u %%\r", (progress / (total / 100)));
});
ArduinoOTA.onError([](ota_error_t error) {
Serial.printf("Error[ %u]: ", error);
if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
else if (error == OTA_END_ERROR) Serial.println("End Failed");
});
ArduinoOTA.begin();
Serial.println("Ready OTA");
Serial.print("IP address: ");
Serial.println(WiFi.localIP());
// Start adding startup code here
// prepare GPIO
pinMode(PIN, OUTPUT);
digitalWrite(PIN, HIGH); // on for now....change later bro
// start server
server.begin();
}
void loop() {
ArduinoOTA.handle();
```

```
WiFiClient client = server.available();

delay(5);

if (client) {

//if(client.available()) {

Serial.println("New Client Available");

String req = client.readStringUntil('\r');

Serial.println(req);

String res = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\nLED status changed ";

for (int i=0; i<N ; i++) {

if(req.indexOf(dev[i]) >= 0) {

if(req.indexOf("on") >= 0) {

digitalWrite(PIN, HIGH);

res = res + "ON \r\n</html>\r\n";

}

else if(req.indexOf("off") >= 0) {

digitalWrite(PIN, LOW);

res = res + "OFF \r\n</html>\r\n";

}

else

res = "HTTP/1.1 404";

}

}

delay(5);

client.print(res);

//client.flush();

//}

//client.stopAll();
```

```
//client.flush();

delay(5);

}

}
```

# 24. Módulo central

## 24.1. Código del observador

```
import requests

import pyinotify

file_path = '/home/ricky/Downloads/hypothesis.txt'

phrases = ['prende luz uno', 'apaga luz uno', 'prende luz dos', 'apaga luz dos']

urls = ['http://10.0.0.12/light1/on', 'http://10.0.0.12/light1/off', 'http://10.0.0.13/light1/on', 'http://10.0.0.13/light1/off']

def on_change(ev):

with open(file_path) as f:

txt = f.read()

txt = txt[:-1]

url = ''

if txt in phrases:

url = urls[phrases.index(txt)]

else:

print txt

if url != '':

re = requests.get(url)#[:-1])

wm.add_watch(file_path, pyinotify.IN_MODIFY, on_change)

wm = pyinotify.WatchManager()

wm.add_watch(file_path, pyinotify.IN_MODIFY, on_change)
```

```
notifier = pyinotify.Notifier(wm)

notifier.loop()

print "Done"
```

## 24.2. Código de Pocketsphinx

```
Continuos.c

/* -*- c-basic-offset: 4; indent-tabs-mode: nil -*- */

/* ====================================

* Copyright (c) 1999-2010 Carnegie Mellon University. All rights

* reserved.

*

* Redistribution and use in source and binary forms, with or without

* modification, are permitted provided that the following conditions

* are met:

*

* 1. Redistributions of source code must retain the above copyright

* notice, this list of conditions and the following disclaimer.

*

* 2. Redistributions in binary form must reproduce the above copyright

* notice, this list of conditions and the following disclaimer in

* the documentation and/or other materials provided with the

* distribution.

*

* This work was supported in part by funding from the Defense Advanced

* Research Projects Agency and the National Science Foundation of the

* United States of America, and the CMU Sphinx Speech Consortium.

*
```

```
* THIS SOFTWARE IS PROVIDED BY CARNEGIE MELLON UNIVERSITY “AS IS” AND
* ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL CARNEGIE MELLON UNIVERSITY
* NOR ITS EMPLOYEES BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
* THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
* ====================================
*
*/
/*
* continuous.c - Simple pocketsphinx command-line application to test
* both continuous listening/silence filtering from microphone
* and continuous file transcription.
*/
/*
* This is a simple example of pocketsphinx application that uses continuous listening
* with silence filtering to automatically segment a continuous stream of audio input
* into utterances that are then decoded.
*
* Remarks:
* - Each utterance is ended when a silence segment of at least 1 sec is recognized.
```

```
* - Single-threaded implementation for portability.

* - Uses audio library; can be replaced with an equivalent custom library.

*/

#include <stdio.h>

#include <string.h>

#include <assert.h>

#include <time.h>

#if defined(_WIN32) && !defined(__CYGWIN__)

#include <windows.h>

#else

#include <sys/select.h>

#endif

#include <sphinxbase/err.h>

#include <sphinxbase/ad.h>

#include "pocketsphinx.h"

////////////////////////////// Funcion agregada para intercambio por archivos ///////////////////////

int hyp_to_file (const char *hyp){

FILE *fp;

fp=fopen("hypothesis.txt","w");

if (fp)

{

fputs(hyp, fp);

//fputc("/n", fp);

}

else

{

printf("Error abriendo hypothesis.txt");
```

```
return 0;

}

fclose(fp);

return 1;

}

//////////////////////////////////////////////

static const arg_t cont_args_def[] = {

POCKETSPHINX_OPTIONS,

/* Argument file. */

{"-argfile",

ARG_STRING,

NULL,

"Argument file giving extra arguments."},

{"-adcdev",

ARG_STRING,

NULL,

"Name of audio device to use for input."},

{"-infile",

ARG_STRING,

NULL,

"Audio file to transcribe."},

{"-inmic",

ARG_BOOLEAN,

"no",

"Transcribe audio from microphone."},

{"-time",

ARG_BOOLEAN,
```

```
"no",
"Print word times in file transcription."},
CMDLN_EMPTY_OPTION
};
static ps_decoder_t *ps;
static cmd_ln_t *config;
static FILE *rawfd;
static void
print_word_times()
{
int frame_rate = cmd_ln_int32_r(config, "-frate");
ps_seg_t *iter = ps_seg_iter(ps);
while (iter != NULL) {
int32 sf, ef, pprob;
float conf;
ps_seg_frames(iter, &sf, &ef);
pprob = ps_seg_prob(iter, NULL, NULL, NULL);
conf = logmath_exp(ps_get_logmath(ps), pprob);
printf("%s %.3f %.3f %f\n", ps_seg_word(iter), ((float)sf / frame_rate),
((float) ef / frame_rate), conf);
iter = ps_seg_next(iter);
}
}
static int
check_wav_header(char *header, int expected_sr)
{
int sr;
```

```
if (header[34] != 0x10) {
E_ERROR("Input audio file has [%d] bits per sample instead of 16\n", header[34]);
return 0;
}
if (header[20] != 0x1) {
E_ERROR("Input audio file has compression [%d] and not required PCM\n", header[20]);
return 0;
}
if (header[22] != 0x1) {
E_ERROR("Input audio file has [%d] channels, expected single channel mono\n", header[22]);
return 0;
}
sr = ((header[24] & 0xFF) | ((header[25] & 0xFF) << 8) | ((header[26] & 0xFF) << 16) | ((header[27] & 0xFF)
<< 24));
if (sr != expected_sr) {
E_ERROR("Input audio file has sample rate [%d], but decoder expects [%d]\n", sr, expected_sr);
return 0;
}
return 1;
}
/*
* Continuous recognition from a file
*/
static void
recognize_from_file()
{
int16 adbuf[2048];
const char *fname;
```

```
const char *hyp;
int32 k;
uint8 utt_started, in_speech;
int32 print_times = cmd_ln_boolean_r(config, "-time");
fname = cmd_ln_str_r(config, "-infile");
if ((rawfd = fopen(fname, "rb")) == NULL) {
E_FATAL_SYSTEM("Failed to open file ' %s' for reading",
fname);
}
if (strlen(fname) > 4 && strcmp(fname + strlen(fname) - 4, ".wav") == 0) {
char waveheader[44];
fread(waveheader, 1, 44, rawfd);
if (!check_wav_header(waveheader, (int)cmd_ln_float32_r(config, "-samprate")))
E_FATAL("Failed to process file ' %s' due to format mismatch.\n", fname);
}
if (strlen(fname) > 4 && strcmp(fname + strlen(fname) - 4, ".mp3") == 0) {
E_FATAL("Can not decode mp3 files, convert input file to WAV 16kHz 16-bit mono before decoding.\n");
}
ps_start_utt(ps);
utt_started = FALSE;
while ((k = fread(adbuf, sizeof(int16), 2048, rawfd)) > 0) {
ps_process_raw(ps, adbuf, k, FALSE, FALSE);
in_speech = ps_get_in_speech(ps);
if (in_speech && !utt_started) {
utt_started = TRUE;
}
if (!in_speech && utt_started) {
```

```
ps_end_utt(ps);
hyp = ps_get_hyp(ps, NULL);
if (hyp != NULL)
printf("%s\n", hyp);
if (print_times)
print_word_times();
fflush(stdout);
ps_start_utt(ps);
utt_started = FALSE;
}
}
ps_end_utt(ps);
if (utt_started) {
hyp = ps_get_hyp(ps, NULL);
if (hyp != NULL) {
printf("%s\n", hyp);
if (print_times) {
print_word_times();
}
}
}
fclose(rawfd);
}
/* Sleep for specified msec */
static void
sleep_msec(int32 ms)
{
```

```
#if (defined(_WIN32) && !defined(GNUWINCE)) || defined(_WIN32_WCE)

Sleep(ms);

#else

/* ------------------- Unix ------------------ */

struct timeval tmo;

tmo.tv_sec = 0;

tmo.tv_usec = ms * 1000;

select(0, NULL, NULL, NULL, &tmo);

#endif

}

/*

* Main utterance processing loop:

* for (;;) {

* start utterance and wait for speech to process

* decoding till end-of-utterance silence will be detected

* print utterance result;

* }

*/

static void

recognize_from_microphone()

{

ad_rec_t *ad;

int16 adbuf[2048];

uint8 utt_started, in_speech;

int32 k;

char const *hyp;

int32 score; // Agregadp
```

```
int32 count=0; // Agregado

//clock_t t1=0, t2=0, t3=0;

if ((ad = ad_open_dev(cmd_ln_str_r(config, "-adcdev"),

(int) cmd_ln_float32_r(config,

"-samprate"))) == NULL)

E_FATAL("Failed to open audio device\n");

if (ad_start_rec(ad) < 0)

E_FATAL("Failed to start recording\n");

if (ps_start_utt(ps) < 0)

E_FATAL("Failed to start utterance\n");

utt_started = FALSE;

E_INFO("Ready....\n");

for (;;) {

if ((k = ad_read(ad, adbuf, 2048)) < 0)

E_FATAL("Failed to read audio\n");

ps_process_raw(ps, adbuf, k, FALSE, FALSE);

in_speech = ps_get_in_speech(ps);

if (in_speech && !utt_started) {

utt_started = TRUE;

E_INFO("Listening...\n");

}

if (!in_speech && utt_started) {

/* speech -> silence transition, time to start new utterance */

ps_end_utt(ps);

hyp = ps_get_hyp(ps, &score ); //hyp = ps_get_hyp(ps, NULL );

if (hyp != NULL && score > -2600) { //if (hyp != NULL) {

printf("\n%s\t%d\t%d\n\n",hyp,score,++count); // printf("%s\n", hyp);
```

```
printf("Al txt: %d\n\n",hyp_to_file(hyp));
fflush(stdout);
}
if (ps_start_utt(ps) < 0)
E_FATAL("Failed to start utterance\n");
utt_started = FALSE;
E_INFO("Ready....\n");
}
sleep_msec(100);
}
ad_close(ad);
}
int
main(int argc, char *argv[])
{
char const *cfg;
config = cmd_ln_parse_r(NULL, cont_args_def, argc, argv, TRUE);
/* Handle argument file as -argfile. */
if (config && (cfg = cmd_ln_str_r(config, "-argfile")) != NULL) {
config = cmd_ln_parse_file_r(config, cont_args_def, cfg, FALSE);
}
if (config == NULL || (cmd_ln_str_r(config, "-infile") == NULL && cmd_ln_boolean_r(config, "-inmic") ==
FALSE)) {
E_INFO("Specify '-infile <file.wav>' to recognize from file or '-inmic yes' to recognize from microphone.\n");
cmd_ln_free_r(config);
return 1;
}
```

```
ps_default_search_args(config);

ps = ps_init(config);

if (ps == NULL) {

cmd_ln_free_r(config);

return 1;

}

E_INFO("%s COMPILED ON: %s, AT: %s\n\n", argv[0], __DATE__, __TIME__);

if (cmd_ln_str_r(config, "-infile") != NULL) {

recognize_from_file();

} else if (cmd_ln_boolean_r(config, "-inmic")) {

recognize_from_microphone();

}

ps_free(ps);

cmd_ln_free_r(config);

return 0;

}

#if defined(_WIN32_WCE)

#pragma comment(linker,"/entry:mainWCRTStartup")

#include <windows.h>

//Windows Mobile has the Unicode main only

int

wmain(int32 argc, wchar_t * wargv[])

{

char **argv;

size_t wlen;

size_t len;

int i;
```

```
argv = malloc(argc * sizeof(char *));
for (i = 0; i < argc; i++) {
wlen = lstrlenW(wargv[i]);
len = wcstombs(NULL, wargv[i], wlen);
argv[i] = malloc(len + 1);
wcstombs(argv[i], wargv[i], wlen);
}
//assuming ASCII parameters
return main(argc, argv);
}
#endif
```

Pocketsphinx.c

```
/* -*- c-basic-offset: 4; indent-tabs-mode: nil -*- */
/* ====================================
* Copyright (c) 2008 Carnegie Mellon University. All rights
* reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* 1. Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*
* 2. Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in
* the documentation and/or other materials provided with the
```

```
* distribution.
*
* This work was supported in part by funding from the Defense Advanced
* Research Projects Agency and the National Science Foundation of the
* United States of America, and the CMU Sphinx Speech Consortium.
*
* THIS SOFTWARE IS PROVIDED BY CARNEGIE MELLON UNIVERSITY “AS IS” AND
* ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL CARNEGIE MELLON UNIVERSITY
* NOR ITS EMPLOYEES BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
* LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
* DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
* THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
* (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
* OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
*
* ====================================
*
*/
/* System headers. */
#include <stdio.h>
#include <assert.h>
#ifdef HAVE_UNISTD_H
#include <unistd.h>
#endif
```

```
/* SphinxBase headers. */
#include <sphinxbase/err.h>
#include <sphinxbase/strfuncs.h>
#include <sphinxbase/filename.h>
#include <sphinxbase/pio.h>
#include <sphinxbase/jsgf.h>
#include <sphinxbase/hash_table.h>
/* Local headers. */
#include "cmdln_macro.h"
#include "pocketsphinx.h"
#include "pocketsphinx_internal.h"
#include "ps_lattice_internal.h"
#include "phone_loop_search.h"
#include "kws_search.h"
#include "fsg_search_internal.h"
#include "ngram_search.h"
#include "ngram_search_fwdtree.h"
#include "ngram_search_fwdflat.h"
#include "allphone_search.h"
static const arg_t ps_args_def[] = {
POCKETSPHINX_OPTIONS,
CMDLN_EMPTY_OPTION
};
/* I'm not sure what the portable way to do this is. */
static int
file_exists(const char *path)
{
```

```
FILE *tmp;

tmp = fopen(path, "rb");

if (tmp) fclose(tmp);

return (tmp != NULL);

}

#ifdef MODELDIR

static int

hmmdir_exists(const char *path)

{

FILE *tmp;

char *mdef = string_join(path, "/mdef", NULL);

tmp = fopen(mdef, "rb");

if (tmp) fclose(tmp);

ckd_free(mdef);

return (tmp != NULL);

}

#endif

static void

ps_expand_file_config(ps_decoder_t *ps, const char *arg, const char *extra_arg,

const char *hmmdir, const char *file)

{

const char *val;

if ((val = cmd_ln_str_r(ps->config, arg)) != NULL) {

cmd_ln_set_str_extra_r(ps->config, extra_arg, val);

} else if (hmmdir == NULL) {

cmd_ln_set_str_extra_r(ps->config, extra_arg, NULL);

} else {
```

```
char *tmp = string_join(hmmdir, "/", file, NULL);
if (file_exists(tmp))
cmd_ln_set_str_extra_r(ps->config, extra_arg, tmp);
else
cmd_ln_set_str_extra_r(ps->config, extra_arg, NULL);
ckd_free(tmp);
}
}
/* Feature and front-end parameters that may be in feat.params */
static const arg_t feat_defn[] = {
waveform_to_cepstral_command_line_macro(),
cepstral_to_feature_command_line_macro(),
CMDLN_EMPTY_OPTION
};
static void
ps_expand_model_config(ps_decoder_t *ps)
{
char const *hmmdir, *featparams;
/* Disable memory mapping on Blackfin (FIXME: should be uClinux in general). */
#ifdef __ADSPBLACKFIN__
E_INFO("Will not use mmap() on uClinux/Blackfin.");
cmd_ln_set_boolean_r(ps->config, "-mmap", FALSE);
#endif
/* Get acoustic model filenames and add them to the command-line */
hmmdir = cmd_ln_str_r(ps->config, "-hmm");
ps_expand_file_config(ps, "-mdef", "_mdef", hmmdir, "mdef");
ps_expand_file_config(ps, "-mean", "_mean", hmmdir, "means");
```

```
ps_expand_file_config(ps, "-var", "_var", hmmdir, "variances");
ps_expand_file_config(ps, "-tmat", "_tmat", hmmdir, "transition_matrices");
ps_expand_file_config(ps, "-mixw", "_mixw", hmmdir, "mixture_weights");
ps_expand_file_config(ps, "-sendump", "_sendump", hmmdir, "sendump");
ps_expand_file_config(ps, "-fdict", "_fdict", hmmdir, "noisedict");
ps_expand_file_config(ps, "-lda", "_lda", hmmdir, "feature_transform");
ps_expand_file_config(ps, "-featparams", "_featparams", hmmdir, "feat.params");
ps_expand_file_config(ps, "-senmgau", "_senmgau", hmmdir, "senmgau");
/* Look for feat.params in acoustic model dir. */
if ((featparams = cmd_ln_str_r(ps->config, "_featparams"))) {
if (NULL !=
cmd_ln_parse_file_r(ps->config, feat_defn, featparams, FALSE))
E_INFO("Parsed model-specific feature parameters from %s\n",
featparams);
}
/* Print here because acmod_init might load feat.params file */
if (err_get_logfp() != NULL) {
cmd_ln_print_values_r(ps->config, err_get_logfp(), ps_args());
}
}
static void
ps_free_searches(ps_decoder_t *ps)
{
if (ps->searches) {
hash_iter_t *search_it;
for (search_it = hash_table_iter(ps->searches); search_it;
search_it = hash_table_iter_next(search_it)) {
```

```
ps_search_free(hash_entry_val(search_it->ent));
}
hash_table_free(ps->searches);
}
ps->searches = NULL;
ps->search = NULL;
}
static ps_search_t *
ps_find_search(ps_decoder_t *ps, char const *name)
{
void *search = NULL;
hash_table_lookup(ps->searches, name, &search);
return (ps_search_t *) search;
}
void
ps_default_search_args(cmd_ln_t *config)
{
#ifdef MODELDIR
/* Set default acoustic and language models. */
const char *hmmdir = cmd_ln_str_r(config, "-hmm");
if (hmmdir == NULL && hmmdir_exists(MODELDIR "/en-us/en-us")) {
hmmdir = MODELDIR "/en-us/en-us";
cmd_ln_set_str_r(config, "-hmm", hmmdir);
}
const char *lmfile = cmd_ln_str_r(config, "-lm");
if (lmfile == NULL && !cmd_ln_str_r(config, "-fsg")
&& !cmd_ln_str_r(config, "-jsgf")
```

```
&& !cmd_ln_str_r(config, "-lmctl")
&& !cmd_ln_str_r(config, "-kws")
&& !cmd_ln_str_r(config, "-keyphrase")
&& file_exists(MODELDIR "/en-us/en-us.lm.bin")) {
lmfile = MODELDIR "/en-us/en-us.lm.bin";
cmd_ln_set_str_r(config, "-lm", lmfile);
}
const char *dictfile = cmd_ln_str_r(config, "-dict");
if (dictfile == NULL && file_exists(MODELDIR "/en-us/cmudict-en-us.dict")) {
dictfile = MODELDIR "/en-us/cmudict-en-us.dict";
cmd_ln_set_str_r(config, "-dict", dictfile);
}
/* Expand acoustic and language model filenames relative to installation
* path. */
if (hmmdir && !path_is_absolute(hmmdir) && !hmmdir_exists(hmmdir)) {
char *tmphmm = string_join(MODELDIR "/hmm/", hmmdir, NULL);
if (hmmdir_exists(tmphmm)) {
cmd_ln_set_str_r(config, "-hmm", tmphmm);
} else {
E_ERROR("Failed to find mdef file inside the model folder "
"specified with -hmm ‘%s’\n", hmmdir);
}
ckd_free(tmphmm);
}
if (lmfile && !path_is_absolute(lmfile) && !file_exists(lmfile)) {
char *tmplm = string_join(MODELDIR "/lm/", lmfile, NULL);
cmd_ln_set_str_r(config, "-lm", tmplm);
```

```
ckd_free(tmplm);

}

if (dictfile && !path_is_absolute(dictfile) && !file_exists(dictfile)) {

char *tmpdict = string_join(MODELDIR "/lm/", dictfile, NULL);

cmd_ln_set_str_r(config, "-dict", tmpdict);

ckd_free(tmpdict);

}

#endif

}

int

ps_reinit(ps_decoder_t *ps, cmd_ln_t *config)

{

const char *path;

const char *keyphrase;

int32 lw;

if (config && config != ps->config) {

cmd_ln_free_r(ps->config);

ps->config = cmd_ln_retain(config);

}

err_set_debug_level(cmd_ln_int32_r(ps->config, "-debug"));

/* Set up logging. We need to do this earlier because we want to dump

* the information to the configured log, not to the stderr. */

if (config && cmd_ln_str_r(ps->config, "-logfn")) {

if (err_set_logfile(cmd_ln_str_r(ps->config, "-logfn")) < 0) {

E_ERROR("Cannot redirect log output\n");

return -1;

}
```

```
}

ps->mfclogdir = cmd_ln_str_r(ps->config, "-mfclogdir");

ps->rawlogdir = cmd_ln_str_r(ps->config, "-rawlogdir");

ps->senlogdir = cmd_ln_str_r(ps->config, "-senlogdir");

/* Fill in some default arguments. */

ps_expand_model_config(ps);

/* Free old searches (do this before other reinit) */

ps_free_searches(ps);

ps->searches = hash_table_new(3, HASH_CASE_YES);

/* Free old acmod. */

acmod_free(ps->acmod);

ps->acmod = NULL;

/* Free old dictionary (must be done after the two things above) */

dict_free(ps->dict);

ps->dict = NULL;

/* Free d2p */

dict2pid_free(ps->d2p);

ps->d2p = NULL;

/* Logmath computation (used in acmod and search) */

if (ps->lmath == NULL

|| (logmath_get_base(ps->lmath) !=

(float64)cmd_ln_float32_r(ps->config, "-logbase"))) {

if (ps->lmath)

logmath_free(ps->lmath);

ps->lmath = logmath_init

((float64)cmd_ln_float32_r(ps->config, "-logbase"), 0,

cmd_ln_boolean_r(ps->config, "-bestpath"));
```

```
}
/* Acoustic model (this is basically everything that
* uttproc.c, senscr.c, and others used to do) */
if ((ps->acmod = acmod_init(ps->config, ps->lmath, NULL, NULL)) == NULL)
return -1;
if (cmd_ln_int32_r(ps->config, "-pl_window") > 0) {
/* Initialize an auxiliary phone loop search, which will run in
* "parallel" with FSG or N-Gram search. */
if ((ps->phone_loop =
phone_loop_search_init(ps->config, ps->acmod, ps->dict)) == NULL)
return -1;
hash_table_enter(ps->searches,
ps_search_name(ps->phone_loop),
ps->phone_loop);
}
/* Dictionary and triphone mappings (depends on acmod). */
/* FIXME: pass config, change arguments, implement LTS, etc. */
if ((ps->dict = dict_init(ps->config, ps->acmod->mdef)) == NULL)
return -1;
if ((ps->d2p = dict2pid_build(ps->acmod->mdef, ps->dict)) == NULL)
return -1;
lw = cmd_ln_float32_r(ps->config, "-lw");
/* Determine whether we are starting out in FSG or N-Gram search mode.
* If neither is used skip search initialization. */
/* Load KWS if one was specified in config */
if ((keyphrase = cmd_ln_str_r(ps->config, "-keyphrase"))) {
if (ps_set_keyphrase(ps, PS_DEFAULT_SEARCH, keyphrase))
```

```
return -1;

ps_set_search(ps, PS_DEFAULT_SEARCH);

}

if ((path = cmd_ln_str_r(ps->config, "-kws"))) {

if (ps_set_kws(ps, PS_DEFAULT_SEARCH, path))

return -1;

ps_set_search(ps, PS_DEFAULT_SEARCH);

}

/* Load an FSG if one was specified in config */

if ((path = cmd_ln_str_r(ps->config, "-fsg"))) {

fsg_model_t *fsg = fsg_model_readfile(path, ps->lmath, lw);

if (!fsg)

return -1;

if (ps_set_fsg(ps, PS_DEFAULT_SEARCH, fsg)) {

fsg_model_free(fsg);

return -1;

}

fsg_model_free(fsg);

ps_set_search(ps, PS_DEFAULT_SEARCH);

}

/* Or load a JSGF grammar */

if ((path = cmd_ln_str_r(ps->config, "-jsgf"))) {

if (ps_set_jsgf_file(ps, PS_DEFAULT_SEARCH, path)

|| ps_set_search(ps, PS_DEFAULT_SEARCH))

return -1;

}

if ((path = cmd_ln_str_r(ps->config, "-allphone"))) {
```

```
if (ps_set_allphone_file(ps, PS_DEFAULT_SEARCH, path)
|| ps_set_search(ps, PS_DEFAULT_SEARCH))
return -1;
}
if ((path = cmd_ln_str_r(ps->config, "-lm")) &&
!cmd_ln_boolean_r(ps->config, "-allphone")) {
if (ps_set_lm_file(ps, PS_DEFAULT_SEARCH, path)
|| ps_set_search(ps, PS_DEFAULT_SEARCH))
return -1;
}
if ((path = cmd_ln_str_r(ps->config, "-lmctl"))) {
const char *name;
ngram_model_t *lmset;
ngram_model_set_iter_t *lmset_it;
if (!(lmset = ngram_model_set_read(ps->config, path, ps->lmath))) {
E_ERROR("Failed to read language model control file: %s\n", path);
return -1;
}
for(lmset_it = ngram_model_set_iter(lmset);
lmset_it; lmset_it = ngram_model_set_iter_next(lmset_it)) {
ngram_model_t *lm = ngram_model_set_iter_model(lmset_it, &name);
E_INFO("adding search %s\n", name);
if (ps_set_lm(ps, name, lm)) {
ngram_model_set_iter_free(lmset_it);
ngram_model_free(lmset);
return -1;
}
```

```
}
ngram_model_free(lmset);
name = cmd_ln_str_r(ps->config, "-lmname");
if (name)
ps_set_search(ps, name);
else {
E_ERROR("No default LM name (-lmname) for ‘-lmctl’\n");
return -1;
}
}
/* Initialize performance timer. */
ps->perf.name = "decode";
ptmr_init(&ps->perf);
return 0;
}
ps_decoder_t *
ps_init(cmd_ln_t *config)
{
ps_decoder_t *ps;
if (!config) {
E_ERROR("No configuration specified");
return NULL;
}
ps = ckd_calloc(1, sizeof(*ps));
ps->refcount = 1;
if (ps_reinit(ps, config) < 0) {
ps_free(ps);
```

```
return NULL;
}
return ps;
}
arg_t const *
ps_args(void)
{
return ps_args_def;
}
ps_decoder_t *
ps_retain(ps_decoder_t *ps)
{
++ps->refcount;
return ps;
}
int
ps_free(ps_decoder_t *ps)
{
if (ps == NULL)
return 0;
if (--ps->refcount > 0)
return ps->refcount;
ps_free_searches(ps);
dict_free(ps->dict);
dict2pid_free(ps->d2p);
acmod_free(ps->acmod);
logmath_free(ps->lmath);
```

```
cmd_ln_free_r(ps->config);

ckd_free(ps);

return 0;

}

cmd_ln_t *

ps_get_config(ps_decoder_t *ps)

{

return ps->config;

}

logmath_t *

ps_get_logmath(ps_decoder_t *ps)

{

return ps->lmath;

}

fe_t *

ps_get_fe(ps_decoder_t *ps)

{

return ps->acmod->fe;

}

feat_t *

ps_get_feat(ps_decoder_t *ps)

{

return ps->acmod->fcb;

}

ps_mllr_t *

ps_update_mllr(ps_decoder_t *ps, ps_mllr_t *mllr)

{
```

```
return acmod_update_mllr(ps->acmod, mllr);

}

int

ps_set_search(ps_decoder_t *ps, const char *name)

{

ps_search_t *search;

if (ps->acmod->state != ACMOD_ENDED && ps->acmod->state != ACMOD_IDLE) {

E_ERROR("Cannot change search while decoding, end utterance first\n");

return -1;

}

if (!(search = ps_find_search(ps, name))) {

return -1;

}

ps->search = search;

/* Set pl window depending on the search */

if (!strcmp(PS_SEARCH_TYPE_NGRAM, ps_search_type(search))) {

ps->pl_window = cmd_ln_int32_r(ps->config, "-pl_window");

} else {

ps->pl_window = 0;

}

return 0;

}

const char*

ps_get_search(ps_decoder_t *ps)

{

hash_iter_t *search_it;

const char* name = NULL;
```

```
for (search_it = hash_table_iter(ps->searches); search_it;
search_it = hash_table_iter_next(search_it)) {
if (hash_entry_val(search_it->ent) == ps->search) {
name = hash_entry_key(search_it->ent);
break;
}
}
return name;
}
int
ps_unset_search(ps_decoder_t *ps, const char *name)
{
ps_search_t *search = hash_table_delete(ps->searches, name);
if (!search)
return -1;
if (ps->search == search)
ps->search = NULL;
ps_search_free(search);
return 0;
}
ps_search_iter_t *
ps_search_iter(ps_decoder_t *ps)
{
return (ps_search_iter_t *)hash_table_iter(ps->searches);
}
ps_search_iter_t *
ps_search_iter_next(ps_search_iter_t *itor)
```

```
{
return (ps_search_iter_t *)hash_table_iter_next((hash_iter_t *)itor);
}
const char*
ps_search_iter_val(ps_search_iter_t *itor)
{
return (const char*)(((hash_iter_t *)itor)->ent->key);
}
void
ps_search_iter_free(ps_search_iter_t *itor)
{
hash_table_iter_free((hash_iter_t *)itor);
}
ngram_model_t *
ps_get_lm(ps_decoder_t *ps, const char *name)
{
ps_search_t *search = ps_find_search(ps, name);
if (search && strcmp(PS_SEARCH_TYPE_NGRAM, ps_search_type(search)))
return NULL;
return search ? ((ngram_search_t *) search)->lmset : NULL;
}
fsg_model_t *
ps_get_fsg(ps_decoder_t *ps, const char *name)
{
ps_search_t *search = ps_find_search(ps, name);
if (search && strcmp(PS_SEARCH_TYPE_FSG, ps_search_type(search)))
return NULL;
```

```
return search ? ((fsg_search_t *) search)->fsg : NULL;

}

const char*

ps_get_kws(ps_decoder_t *ps, const char* name)

{

ps_search_t *search = ps_find_search(ps, name);

if (search && strcmp(PS_SEARCH_TYPE_KWS, ps_search_type(search)))

return NULL;

return search ? kws_search_get_keywords(search) : NULL;

}

static int

set_search_internal(ps_decoder_t *ps, ps_search_t *search)

{

ps_search_t *old_search;

if (!search)

return -1;

search->pls = ps->phone_loop;

old_search = (ps_search_t *) hash_table_replace(ps->searches, ps_search_name(search), search);

if (old_search != search)

ps_search_free(old_search);

return 0;

}

int

ps_set_lm(ps_decoder_t *ps, const char *name, ngram_model_t *lm)

{

ps_search_t *search;

search = ngram_search_init(name, lm, ps->config, ps->acmod, ps->dict, ps->d2p);
```

```
return set_search_internal(ps, search);

}

int

ps_set_lm_file(ps_decoder_t *ps, const char *name, const char *path)

{

ngram_model_t *lm;

int result;

lm = ngram_model_read(ps->config, path, NGRAM_AUTO, ps->lmath);

if (!lm)

return -1;

result = ps_set_lm(ps, name, lm);

ngram_model_free(lm);

return result;

}

int

ps_set_allphone(ps_decoder_t *ps, const char *name, ngram_model_t *lm)

{

ps_search_t *search;

search = allphone_search_init(name, lm, ps->config, ps->acmod, ps->dict, ps->d2p);

return set_search_internal(ps, search);

}

int

ps_set_allphone_file(ps_decoder_t *ps, const char *name, const char *path)

{

ngram_model_t *lm;

int result;

lm = NULL;
```

```
if (path)

lm = ngram_model_read(ps->config, path, NGRAM_AUTO, ps->lmath);

result = ps_set_allphone(ps, name, lm);

if (lm)

ngram_model_free(lm);

return result;

}

int

ps_set_kws(ps_decoder_t *ps, const char *name, const char *keyfile)

{

ps_search_t *search;

search = kws_search_init(name, NULL, keyfile, ps->config, ps->acmod, ps->dict, ps->d2p);

return set_search_internal(ps, search);

}

int

ps_set_keyphrase(ps_decoder_t *ps, const char *name, const char *keyphrase)

{

ps_search_t *search;

search = kws_search_init(name, keyphrase, NULL, ps->config, ps->acmod, ps->dict, ps->d2p);

return set_search_internal(ps, search);

}

int

ps_set_fsg(ps_decoder_t *ps, const char *name, fsg_model_t *fsg)

{

ps_search_t *search;

search = fsg_search_init(name, fsg, ps->config, ps->acmod, ps->dict, ps->d2p);

return set_search_internal(ps, search);
```

```
}

int

ps_set_jsgf_file(ps_decoder_t *ps, const char *name, const char *path)

{

fsg_model_t *fsg;

jsgf_rule_t *rule;

char const *toprule;

jsgf_t *jsgf = jsgf_parse_file(path, NULL);

float lw;

int result;

if (!jsgf)

return -1;

rule = NULL;

/* Take the -toprule if specified. */

if ((toprule = cmd_ln_str_r(ps->config, "-toprule"))) {

rule = jsgf_get_rule(jsgf, toprule);

if (rule == NULL) {

E_ERROR("Start rule %s not found\n", toprule);

jsgf_grammar_free(jsgf);

return -1;

}

} else {

rule = jsgf_get_public_rule(jsgf);

if (rule == NULL) {

E_ERROR("No public rules found in %s\n", path);

jsgf_grammar_free(jsgf);

return -1;
```

```
}

}

lw = cmd_ln_float32_r(ps->config, "-lw");

fsg = jsgf_build_fsg(jsgf, rule, ps->lmath, lw);

result = ps_set_fsg(ps, name, fsg);

fsg_model_free(fsg);

jsgf_grammar_free(jsgf);

return result;

}

int

ps_set_jsgf_string(ps_decoder_t *ps, const char *name, const char *jsgf_string)

{

fsg_model_t *fsg;

jsgf_rule_t *rule;

char const *toprule;

jsgf_t *jsgf = jsgf_parse_string(jsgf_string, NULL);

float lw;

int result;

if (!jsgf)

return -1;

rule = NULL;

/* Take the -toprule if specified. */

if ((toprule = cmd_ln_str_r(ps->config, "-toprule"))) {

rule = jsgf_get_rule(jsgf, toprule);

if (rule == NULL) {

E_ERROR("Start rule %s not found\n", toprule);

return -1;
```

```
}
} else {
rule = jsgf_get_public_rule(jsgf);
if (rule == NULL) {
E_ERROR("No public rules found in input string\n");
return -1;
}
}
lw = cmd_ln_float32_r(ps->config, "-lw");
fsg = jsgf_build_fsg(jsgf, rule, ps->lmath, lw);
result = ps_set_fsg(ps, name, fsg);
fsg_model_free(fsg);
return result;
}
int
ps_load_dict(ps_decoder_t *ps, char const *dictfile,
char const *fdictfile, char const *format)
{
dict2pid_t *d2p;
dict_t *dict;
hash_iter_t *search_it;
cmd_ln_t *newconfig;
/* Create a new scratch config to load this dict (so existing one
* won't be affected if it fails) */
newconfig = cmd_ln_init(NULL, ps_args(), TRUE, NULL);
cmd_ln_set_boolean_r(newconfig, "-dictcase",
cmd_ln_boolean_r(ps->config, "-dictcase"));
```

```
cmd_ln_set_str_r(newconfig, "-dict", dictfile);
if (fdictfile)
cmd_ln_set_str_extra_r(newconfig, "_fdict", fdictfile);
else
cmd_ln_set_str_extra_r(newconfig, "_fdict",
cmd_ln_str_r(ps->config, "_fdict"));
/* Try to load it. */
if ((dict = dict_init(newconfig, ps->acmod->mdef)) == NULL) {
cmd_ln_free_r(newconfig);
return -1;
}
/* Reinit the dict2pid. */
if ((d2p = dict2pid_build(ps->acmod->mdef, dict)) == NULL) {
cmd_ln_free_r(newconfig);
return -1;
}
/* Success! Update the existing config to reflect new dicts and
* drop everything into place. */
cmd_ln_free_r(newconfig);
dict_free(ps->dict);
ps->dict = dict;
dict2pid_free(ps->d2p);
ps->d2p = d2p;
/* And tell all searches to reconfigure themselves. */
for (search_it = hash_table_iter(ps->searches); search_it;
search_it = hash_table_iter_next(search_it)) {
if (ps_search_reinit(hash_entry_val(search_it->ent), dict, d2p) < 0) {
```

```
hash_table_iter_free(search_it);
return -1;
}
}
return 0;
}
int
ps_save_dict(ps_decoder_t *ps, char const *dictfile,
char const *format)
{
return dict_write(ps->dict, dictfile, format);
}
int
ps_add_word(ps_decoder_t *ps,
char const *word,
char const *phones,
int update)
{
int32 wid;
s3cipid_t *pron;
hash_iter_t *search_it;
char **phonestr, *tmp;
int np, i, rv;
/* Parse phones into an array of phone IDs. */
tmp = ckd_salloc(phones);
np = str2words(tmp, NULL, 0);
phonestr = ckd_calloc(np, sizeof(*phonestr));
```

```
str2words(tmp, phonestr, np);
pron = ckd_calloc(np, sizeof(*pron));
for (i = 0; i < np; ++i) {
pron[i] = bin_mdef_ciphone_id(ps->acmod->mdef, phonestr[i]);
if (pron[i] == -1) {
E_ERROR("Unknown phone %s in phone string %s\n",
phonestr[i], tmp);
ckd_free(phonestr);
ckd_free(tmp);
ckd_free(pron);
return -1;
}
}
/* No longer needed. */
ckd_free(phonestr);
ckd_free(tmp);
/* Add it to the dictionary. */
if ((wid = dict_add_word(ps->dict, word, pron, np)) == -1) {
ckd_free(pron);
return -1;
}
/* No longer needed. */
ckd_free(pron);
/* Now we also have to add it to dict2pid. */
dict2pid_add_word(ps->d2p, wid);
/* TODO: we definitely need to refactor this */
for (search_it = hash_table_iter(ps->searches); search_it;
```

```
search_it = hash_table_iter_next(search_it)) {
ps_search_t *search = hash_entry_val(search_it->ent);
if (!strcmp(PS_SEARCH_TYPE_NGRAM, ps_search_type(search))) {
ngram_model_t *lmset = ((ngram_search_t *) search)->lmset;
if (ngram_model_add_word(lmset, word, 1.0) == NGRAM_INVALID_WID) {
hash_table_iter_free(search_it);
return -1;
}
}
if (update) {
if ((rv = ps_search_reinit(search, ps->dict, ps->d2p) < 0)) {
hash_table_iter_free(search_it);
return rv;
}
}
}
/* Rebuild the widmap and search tree if requested. */
return wid;
}
char *
ps_lookup_word(ps_decoder_t *ps, const char *word)
{
s3wid_t wid;
int32 phlen, j;
char *phones;
dict_t *dict = ps->dict;
wid = dict_wordid(dict, word);
```

```
if (wid == BAD_S3WID)
return NULL;
for (phlen = j = 0; j < dict_pronlen(dict, wid); ++j)
phlen += strlen(dict_ciphone_str(dict, wid, j)) + 1;
phones = ckd_calloc(1, phlen);
for (j = 0; j < dict_pronlen(dict, wid); ++j) {
strcat(phones, dict_ciphone_str(dict, wid, j));
if (j != dict_pronlen(dict, wid) - 1)
strcat(phones, " ");
}
return phones;
}
long
ps_decode_raw(ps_decoder_t *ps, FILE *rawfh,
long maxsamps)
{
int16 *data;
long total, pos, endpos;
ps_start_stream(ps);
ps_start_utt(ps);
/* If this file is seekable or maxsamps is specified, then decode
* the whole thing at once. */
if (maxsamps != -1) {
data = ckd_calloc(maxsamps, sizeof(*data));
total = fread(data, sizeof(*data), maxsamps, rawfh);
ps_process_raw(ps, data, total, FALSE, TRUE);
ckd_free(data);
```

```
} else if ((pos = ftell(rawfh)) >= 0) {

fseek(rawfh, 0, SEEK_END);

endpos = ftell(rawfh);

fseek(rawfh, pos, SEEK_SET);

maxsamps = endpos - pos;

data = ckd_calloc(maxsamps, sizeof(*data));

total = fread(data, sizeof(*data), maxsamps, rawfh);

ps_process_raw(ps, data, total, FALSE, TRUE);

ckd_free(data);

} else {

/* Otherwise decode it in a stream. */

total = 0;

while (!feof(rawfh)) {

int16 data[256];

size_t nread;

nread = fread(data, sizeof(*data), sizeof(data)/sizeof(*data), rawfh);

ps_process_raw(ps, data, nread, FALSE, FALSE);

total += nread;

}

}

ps_end_utt(ps);

return total;

}

int

ps_start_stream(ps_decoder_t *ps)

{

acmod_start_stream(ps->acmod);
```

```
return 0;

}

int

ps_start_utt(ps_decoder_t *ps)

{

int rv;

char uttid[16];

if (ps->acmod->state == ACMOD_STARTED || ps->acmod->state == ACMOD_PROCESSING) {

E_ERROR("Utterance already started\n");

return -1;

}

if (ps->search == NULL) {

E_ERROR("No search module is selected, did you forget to "

"specify a language model or grammar?\n");

return -1;

}

ptmr_reset(&ps->perf);

ptmr_start(&ps->perf);

sprintf(uttid, "%09u", ps->uttno);

++ps->uttno;

/* Remove any residual word lattice and hypothesis. */

ps_lattice_free(ps->search->dag);

ps->search->dag = NULL;

ps->search->last_link = NULL;

ps->search->post = 0;

ckd_free(ps->search->hyp_str);

ps->search->hyp_str = NULL;
```

```
if ((rv = acmod_start_utt(ps->acmod)) < 0)
return rv;
/* Start logging features and audio if requested. */
if (ps->mfclogdir) {
char *logfn = string_join(ps->mfclogdir, "/",
uttid, ".mfc", NULL);
FILE *mfcfh;
E_INFO("Writing MFCC log file: %s\n", logfn);
if ((mfcfh = fopen(logfn, "wb")) == NULL) {
E_ERROR_SYSTEM("Failed to open MFCC log file %s", logfn);
ckd_free(logfn);
return -1;
}
ckd_free(logfn);
acmod_set_mfcfh(ps->acmod, mfcfh);
}
if (ps->rawlogdir) {
char *logfn = string_join(ps->rawlogdir, "/",
uttid, ".raw", NULL);
FILE *rawfh;
E_INFO("Writing raw audio log file: %s\n", logfn);
if ((rawfh = fopen(logfn, "wb")) == NULL) {
E_ERROR_SYSTEM("Failed to open raw audio log file %s", logfn);
ckd_free(logfn);
return -1;
}
ckd_free(logfn);
```

```
acmod_set_rawfh(ps->acmod, rawfh);

}

if (ps->senlogdir) {

char *logfn = string_join(ps->senlogdir, "/",

uttid, ".sen", NULL);

FILE *senfh;

E_INFO("Writing senone score log file: %s\n", logfn);

if ((senfh = fopen(logfn, "wb")) == NULL) {

E_ERROR_SYSTEM("Failed to open senone score log file %s", logfn);

ckd_free(logfn);

return -1;

}

ckd_free(logfn);

acmod_set_senfh(ps->acmod, senfh);

}

/* Start auxiliary phone loop search. */

if (ps->phone_loop)

ps_search_start(ps->phone_loop);

return ps_search_start(ps->search);

}

static int

ps_search_forward(ps_decoder_t *ps)

{

int nfr;

nfr = 0;

while (ps->acmod->n_feat_frame > 0) {

int k;
```

```
if (ps->pl_window > 0)

if ((k = ps_search_step(ps->phone_loop, ps->acmod->output_frame)) < 0)

return k;

if (ps->acmod->output_frame >= ps->pl_window)

if ((k = ps_search_step(ps->search,

ps->acmod->output_frame - ps->pl_window)) < 0)

return k;

acmod_advance(ps->acmod);

++ps->n_frame;

++nfr;

}

return nfr;

}

int

ps_decode_senscr(ps_decoder_t *ps, FILE *senfh)

{

int nfr, n_searchfr;

ps_start_utt(ps);

n_searchfr = 0;

acmod_set_insenfh(ps->acmod, senfh);

while ((nfr = acmod_read_scores(ps->acmod)) > 0) {

if ((nfr = ps_search_forward(ps)) < 0) {

ps_end_utt(ps);

return nfr;

}

n_searchfr += nfr;

}
```

```
ps_end_utt(ps);

acmod_set_insenfh(ps->acmod, NULL);

return n_searchfr;

}

int

ps_process_raw(ps_decoder_t *ps,

int16 const *data,

size_t n_samples,

int no_search,

int full_utt)

{

int n_searchfr = 0;

if (ps->acmod->state == ACMOD_IDLE) {

E_ERROR("Failed to process data, utterance is not started. Use start_utt to start it\n");

return 0;

}

if (no_search)

acmod_set_grow(ps->acmod, TRUE);

while (n_samples) {

int nfr;

/* Process some data into features. */

if ((nfr = acmod_process_raw(ps->acmod, &data,

&n_samples, full_utt)) < 0)

return nfr;

/* Score and search as much data as possible */

if (no_search)

continue;
```

```
if ((nfr = ps_search_forward(ps)) < 0)
return nfr;
n_searchfr += nfr;
}
return n_searchfr;
}
int
ps_process_cep(ps_decoder_t *ps,
mfcc_t **data,
int32 n_frames,
int no_search,
int full_utt)
{
int n_searchfr = 0;
if (no_search)
acmod_set_grow(ps->acmod, TRUE);
while (n_frames) {
int nfr;
/* Process some data into features. */
if ((nfr = acmod_process_cep(ps->acmod, &data,
&n_frames, full_utt)) < 0)
return nfr;
/* Score and search as much data as possible */
if (no_search)
continue;
if ((nfr = ps_search_forward(ps)) < 0)
return nfr;
```

```
n_searchfr += nfr;

}

return n_searchfr;

}

int

ps_end_utt(ps_decoder_t *ps)

{

int rv, i;

if (ps->acmod->state == ACMOD_ENDED || ps->acmod->state == ACMOD_IDLE) {

E_ERROR("Utterance is not started\n");

return -1;

}

acmod_end_utt(ps->acmod);

/* Search any remaining frames. */

if ((rv = ps_search_forward(ps)) < 0) {

ptmr_stop(&ps->perf);

return rv;

}

/* Finish phone loop search. */

if (ps->phone_loop) {

if ((rv = ps_search_finish(ps->phone_loop)) < 0) {

ptmr_stop(&ps->perf);

return rv;

}

}

/* Search any frames remaining in the lookahead window. */

if (ps->acmod->output_frame >= ps->pl_window) {
```

```
for (i = ps->acmod->output_frame - ps->pl_window;
i < ps->acmod->output_frame; ++i)
ps_search_step(ps->search, i);
}
/* Finish main search. */
if ((rv = ps_search_finish(ps->search)) < 0) {
ptmr_stop(&ps->perf);
return rv;
}
ptmr_stop(&ps->perf);
/* Log a backtrace if requested. */
if (cmd_ln_boolean_r(ps->config, "-backtrace")) {
const char* hyp;
ps_seg_t *seg;
int32 score;
hyp = ps_get_hyp(ps, &score);
if (hyp != NULL) {
E_INFO("%s (%d)\n", hyp, score);
E_INFO_NOFN("%-20s %-5s %-5s %-5s %-10s %-10s %-3s\n",
"word", "start", "end", "pprob", "ascr", "lscr", "lback");
for (seg = ps_seg_iter(ps); seg;
seg = ps_seg_next(seg)) {
char const *word;
int sf, ef;
int32 post, lscr, ascr, lback;
word = ps_seg_word(seg);
ps_seg_frames(seg, &sf, &ef);
```

```
post = ps_seg_prob(seg, &ascr, &lscr, &lback);
E_INFO_NOFN("%-20s%-5d%-5d%-1.3f%-10d%-10d%-3d\n",
word, sf, ef, logmath_exp(ps_get_logmath(ps), post),
ascr, lscr, lback);
}
}
}
return rv;
}
char const *
ps_get_hyp(ps_decoder_t *ps, int32 *out_best_score)
{
char const *hyp;
ptmr_start(&ps->perf);
hyp = ps_search_hyp(ps->search, out_best_score, NULL);
ptmr_stop(&ps->perf);
return hyp;
}
char const *
ps_get_hyp_final(ps_decoder_t *ps, int32 *out_is_final)
{
char const *hyp;
ptmr_start(&ps->perf);
hyp = ps_search_hyp(ps->search, NULL, out_is_final);
ptmr_stop(&ps->perf);
return hyp;
}
```

```
int32

ps_get_prob(ps_decoder_t *ps)

{

int32 prob;

ptmr_start(&ps->perf);

prob = ps_search_prob(ps->search);

ptmr_stop(&ps->perf);

return prob;

}

ps_seg_t *

ps_seg_iter(ps_decoder_t *ps)

{

ps_seg_t *itor;

ptmr_start(&ps->perf);

itor = ps_search_seg_iter(ps->search);

ptmr_stop(&ps->perf);

return itor;

}

ps_seg_t *

ps_seg_next(ps_seg_t *seg)

{

return ps_search_seg_next(seg);

}

char const *

ps_seg_word(ps_seg_t *seg)

{

return seg->word;
```

```
}

void

ps_seg_frames(ps_seg_t *seg, int *out_sf, int *out_ef)

{

int uf;

uf = acmod_stream_offset(seg->search->acmod);

if (out_sf) *out_sf = seg->sf + uf;

if (out_ef) *out_ef = seg->ef + uf;

}

int32

ps_seg_prob(ps_seg_t *seg, int32 *out_ascr, int32 *out_lscr, int32 *out_lback)

{

if (out_ascr) *out_ascr = seg->ascr;

if (out_lscr) *out_lscr = seg->lscr;

if (out_lback) *out_lback = seg->lback;

return seg->prob;

}

void

ps_seg_free(ps_seg_t *seg)

{

ps_search_seg_free(seg);

}

ps_lattice_t *

ps_get_lattice(ps_decoder_t *ps)

{

return ps_search_lattice(ps->search);

}
```

```
ps_nbest_t *

ps_nbest(ps_decoder_t *ps)

{

ps_lattice_t *dag;

ngram_model_t *lmset;

ps_astar_t *nbest;

float32 lwf;

if (ps->search == NULL)

return NULL;

if ((dag = ps_get_lattice(ps)) == NULL)

return NULL;

/* FIXME: This is all quite specific to N-Gram search. Either we

* should make N-best a method for each search module or it needs

* to be abstracted to work for N-Gram and FSG. */

if (0 != strcmp(ps_search_type(ps->search), PS_SEARCH_TYPE_NGRAM)) {

lmset = NULL;

lwf = 1.0f;

} else {

lmset = ((ngram_search_t *)ps->search)->lmset;

lwf = ((ngram_search_t *)ps->search)->bestpath_fwdtree_lw_ratio;

}

nbest = ps_astar_start(dag, lmset, lwf, 0, -1, -1, -1);

nbest = ps_nbest_next(nbest);

return (ps_nbest_t *)nbest;

}

void

ps_nbest_free(ps_nbest_t *nbest)
```

```
{
ps_astar_finish(nbest);
}
ps_nbest_t *
ps_nbest_next(ps_nbest_t *nbest)
{
ps_latpath_t *next;
next = ps_astar_next(nbest);
if (next == NULL) {
ps_nbest_free(nbest);
return NULL;
}
return nbest;
}
char const *
ps_nbest_hyp(ps_nbest_t *nbest, int32 *out_score)
{
assert(nbest != NULL);
if (nbest->top == NULL)
return NULL;
if (out_score) *out_score = nbest->top->score;
return ps_astar_hyp(nbest, nbest->top);
}
ps_seg_t *
ps_nbest_seg(ps_nbest_t *nbest)
{
if (nbest->top == NULL)
```

```
return NULL;

return ps_astar_seg_iter(nbest, nbest->top, 1.0);

}

int

ps_get_n_frames(ps_decoder_t *ps)

{

return ps->acmod->output_frame + 1;

}

void

ps_get_utt_time(ps_decoder_t *ps, double *out_nspeech,

double *out_ncpu, double *out_nwall)

{

int32 frate;

frate = cmd_ln_int32_r(ps->config, "-frate");

*out_nspeech = (double)ps->acmod->output_frame / frate;

*out_ncpu = ps->perf.t_cpu;

*out_nwall = ps->perf.t_elapsed;

}

void

ps_get_all_time(ps_decoder_t *ps, double *out_nspeech,

double *out_ncpu, double *out_nwall)

{

int32 frate;

frate = cmd_ln_int32_r(ps->config, "-frate");

*out_nspeech = (double)ps->n_frame / frate;

*out_ncpu = ps->perf.t_tot_cpu;

*out_nwall = ps->perf.t_tot_elapsed;
```

```
}

uint8

ps_get_in_speech(ps_decoder_t *ps)

{

return fe_get_vad_state(ps->acmod->fe);

}

void

ps_search_init(ps_search_t *search, ps_searchfuncs_t *vt,

const char *type,

const char *name,

cmd_ln_t *config, acmod_t *acmod, dict_t *dict,

dict2pid_t *d2p)

{

search->vt = vt;

search->name = ckd_salloc(name);

search->type = ckd_salloc(type);

search->config = config;

search->acmod = acmod;

if (d2p)

search->d2p = dict2pid_retain(d2p);

else

search->d2p = NULL;

if (dict) {

search->dict = dict_retain(dict);

search->start_wid = dict_startwid(dict);

search->finish_wid = dict_finishwid(dict);

search->silence_wid = dict_silwid(dict);
```

```
search->n_words = dict_size(dict);
}
else {
search->dict = NULL;
search->start_wid = search->finish_wid = search->silence_wid = -1;
search->n_words = 0;
}
}
void
ps_search_base_free(ps_search_t *search)
{
/* FIXME: We will have refcounting on acmod, config, etc, at which
* point we will free them here too. */
ckd_free(search->name);
ckd_free(search->type);
dict_free(search->dict);
dict2pid_free(search->d2p);
ckd_free(search->hyp_str);
ps_lattice_free(search->dag);
}
void
ps_search_base_reinit(ps_search_t *search, dict_t *dict,
dict2pid_t *d2p)
{
dict_free(search->dict);
dict2pid_free(search->d2p);
/* FIXME: _retain() should just return NULL if passed NULL. */
```

```
if (dict) {
search->dict = dict_retain(dict);
search->start_wid = dict_startwid(dict);
search->finish_wid = dict_finishwid(dict);
search->silence_wid = dict_silwid(dict);
search->n_words = dict_size(dict);
}
else {
search->dict = NULL;
search->start_wid = search->finish_wid = search->silence_wid = -1;
search->n_words = 0;
}
if (d2p)
search->d2p = dict2pid_retain(d2p);
else
search->d2p = NULL;
}
void
ps_set_rawdata_size(ps_decoder_t *ps, int32 size)
{
acmod_set_rawdata_size(ps->acmod, size);
}
void
ps_get_rawdata(ps_decoder_t *ps, int16 **buffer, int32 *size)
{
acmod_get_rawdata(ps->acmod, buffer, size);
}
```