SPEAK-ON

Sistema de comunicación audiovisual para personas con déficit sensorial y traducción comunicativa.

Jesús González Martínez

Grado de Multimedia.

Arduino.

Consultor: Antoni Morell Pérez

Profesor: Pere Tuset Peiró

Me gustaría agradecer a mi mujer Susana y mis dos hijos pequeños Nicolás y Pablo el apoyo e implicación en la exposición del proyecto.
La perspectiva que me ha trasladado mi familia ha sido fundamental para desarrollar el trabajo y perfeccionar la lógica de funcionamiento, donde ha sido reseñable el prisma que me han ofrecido en el apartado de usabilidad del dispositivo.
Finalmente, quiero dar las gracias a las principales empresas implicadas en el proyecto: Microsoft (área de desarrollo cognitivo), Patrick Cauderlier de la empresa Acapela, Yandex, comunidad de Wit.Ai, Cerevoice, IBM, y Danny de Responsive voice, por facilitar el licenciamiento e información en el desarrollo académico.

FICHA DEL TRABAJO FINAL.

Título del trabajo:	Speak-ON: Sistema de comunicación audiovisual para personas con déficit sensorial y traducción comunicativa.
Nombre del autor:	Jesús González Martínez.
Nombre del consultor/a:	Antoni Morell Pérez
Nombre del PRA:	Pere Tuset Peiró
Fecha de entrega (mm/aaaa):	01/2019
Titulación::	Grado de Multimedia
Área del Trabajo Final:	TFG-Arduino
Idioma del trabajo:	Castellano
Palabras clave:	multimedia, cognición, discapacidad.

Resumen del Trabajo:

El proyecto Speak-on ha consistido en el análisis y desarrollo del prototipo de unas lentes de ayuda cognitiva en la comunicación, traducción del lenguaje y síntesis de voz mediante el procesamiento de las conversaciones de un sujeto en siete idiomas: catalán, inglés, alemán, francés, italiano, portugués y español, presentación del texto procesado en una pantalla e integración de la síntesis de voz en el medio audiovisual .En el prototipo se ha utilizado una placa Arduino (capa de presentación) y un procesador del fabricante Onion *(*capa física*),* con el propósito de diseñar un producto con una interfaz web accesible.

Se han analizado diferentes soluciones, lenguajes y métodos de implantación: JavaScript, PHP, Python, Processing, etc. Los diversos algoritmos de cada procedimiento han sido desarrollados integrando una selección previa de determinadas API implicadas en el proyecto: bloque STT *"speech to text "(*habla-texto),bloque de traducción, síntesis de voz, ,TTS "*text to speech*"(texto-habla), además, el examen cuantitativo efectuado, ha sido determinante en la implantación.

Por otro lado, el principal objetivo del proyecto ha sido elaborar una solución que facilite a personas con discapacidad visual o auditiva un sistema de transcripción multilenguaje si un conocimiento previo del medio evitando periodos de adaptación.

Finalmente, se ha concluido que es posible ofrecer un elemento cognitivo ,eficaz y accesible con un impacto económico reducido.

Es importante señalar que se ha construido un modelo del dispositivo, con el objeto de, examinar las limitaciones de tamaño y peso, características que serán recomendables revisar en futuras versiones de comercialización.

Abstract :

The Speak-on project has been consisted of the analysis and development of the prototype cognitive aid lenses in communication, language translation and speech synthesis by processing people's conversations in seven languages: Catalan, English, German, French, Italian, Portuguese and Spanish, presentation of text processed on a screen and integration of speech synthesis into Audiovisual Media. In the prototype has been used an Arduino board (presentation layer) and an onion processor (physical layer) to design a product with an accessible web interface.

Has been analysed different solutions, languages and methods of implementation: JavaScript, PHP, Python, processing, etc.The different algorithms of each procedure have been developed by integrating a pre-selection of certain API involved in the project: STT block (voice-text), Translate block, speech synthesis, TTS (text-to-speech), in addition, the quantitative examination carried out, has been instrumental in the implementation.

In the same way, the main objective of the project has been to develop a solution that provides people with visual or hearing disabilities with a multi-language transcription system, eliminating adaptation periods.

Finally, has been concluded that it is possible to design a prototype cognitive, effective and accessible with a reduced economic impact. It is important to bear in mind that a model of the device has been built to examine size and weight limitations, features that will be recommended for review in future marketing versions

Notaciones y convenciones.

Tipo de contenido	Fuente	Tamaño de fuente	Alineación del texto
Portada.	Arial Negrita.	24,14,12 puntos.	Centrado.
Títulos primer nivel.	Arial Negrita.	20 puntos.	Izquierda.
Títulos segundo nivel.	Arial Negrita.	14 puntos.	Izquierda.
Títulos tercer nivel.	Arial Negrita. Cursiva.	10 puntos.	Izquierda.
Texto del documento.	Times New Roman.	12 puntos.	Justificado.
Código de programación.	Courier New.	9 puntos.	Izquierda.
Cuadros, tablas y figuras.	Arial.	10 puntos.	Izquierda.
Citas.	Times New Roman, cursiva.	12 puntos.	Justificado.
Notas al pie.	Arial.	10 puntos.	Izquierda.

Índice

1. Introducción 11
2. Objetivos 12
2.1 Principales: 12
2.2 Secundarios: 12
3. Contenido 13
4. Metodología 14
5. Planificación. 15
5.1 Intervalo temporal de las fases del proyecto. 15
5.2 Calendario de Inicio de las fases del proyecto. 16
5.3 Diagrama de *Gantt* 16
6. Proceso de trabajo. 18
7. Bloque de conversión voz-texto (STT) 19
7.1 Google *SpeechText* 20
7.2 IBM Watson 20
7.3 Microsoft Speech Cognitive Services. 24
7.4 Wit.AI 26
7.5 Comparativa y selección final. API STT. 31
8.Bloque de traducción. 31
8.1 Microsoft *Translation Text*. Servicios cognitivos. 32
8.2 IBM Translate. Servicio de traducción 35
8.3 Googletrans. Servicio de traducción. 36
8.4 Yandex Translate. Servicio de traducción. 37
8.5 Comparativa y selección final de la Api de traducción 39
9.Bloque de síntesis de voz. TTS 40
9.1 Test comparativo de síntesis del habla. 41
9.2 Cuadro comparativo de características y observaciones TTS. 41
9.3 Selección final en el proceso de conversión texto a voz. 42
10. Soluciones implicadas en el proyecto 42
11. Equipamiento electrónico. 43
11.1 Comparativa de dispositivos 43
11.2 Equivalencia de conexiones: *Omega* y el Microcontrolador Atmega: 44

11.3 Componentes electrónicos. 45
12. Diseño del circuito 46
12.1 Esquema funcional. 46
12.2 Conexionado en *Protoboard.* 46
12.3 Diagrama conectivo de bloques. 47
13. Funcionamiento. 48
13.3 Video de test inicial de comprobación funcional. 48
13.2 Estructura de ficheros. 49
13.3 Algoritmo principal. 49
13.4 Arduino. 54
14. Prototipo 56
14.1 Speak-on V1.0 56
14.2 Speak-on V1.1 57
15. Proyección a futuro 57
16.Viabilidad económica. 58
16.1 Costes 58
16.2 Rentabilidad: 59
16.3 Previsión de evolución de ventas 59
17. Conclusiones 60
Anexo 1. Entregables del proyecto. 61
Anexo 2. Código fuente 62
Anexo 3. Capturas de pantalla 71
Anexo 4. Guía de usuario 74
Anexo 5. Bibliografía 76

Índice de figuras:

Figura 1: Fases del proyecto 15
Figura 2: Diagrama de Gantt (septiembre 2.018-octubre 2.018) 16
Figura 3: Diagrama de Gantt Diagrama de Gantt (noviembre 2.018-diciembre 2.018) 17
Figura 4:Diagrama de Gantt (diciembre 2.018-enero 2.019) 17
Figura 5:Contenido de la API de IBM Watson Speech 21
Figura 6:Aplicaciones activas en la página web de wit.ai para el proyecto 27
Figura 7:Configuración de una aplicación y lenguaje correspondiente. 28

Figura 8:Ejemplo de petición errónea en la API de Wit.AI 30
Figura 9:Resultado de la comparativa en el bloque de conversión voz-texto 31
Figura 10:Resultado de la comparativa en el bloque de traducción 39
Figura 11:Diagrama de relación de pines Arduino Omega 2+. (25) 44
Figura 12:Esquema funcional del circuito electrónico. 46
Figura 13:Conexionado de componentes en placa de pruebas. 46
Figura 14:Diagrama de bloques funcionales. 47
Figura 15:Imágen del prototipo Speak-on V1.1 48
Figura 16:Imagen del sistema speak-on inicializado. 49
Figura 17:selección de idiomas y configuración inicial. 50
Figura 18:Imagenes del estado del algoritmo principal. 51
Figura 19:Proceso de construcción y test de la versión Speak-on 1.0. 56
Figura 20:Imagen del prototipo Speak-on V1.1 57
Figura 21: Imagen de selección de red. 62
Figura 22 Imagen de configuración del sonido 63
Figura 23:Imágenes del prototipo inicial 71
Figura 24 Imágenes del estado funcional del equipo electrónico. 72
Figura 25.Versión 1.1.Test de proyección 73
Figura 26:Conexión a red inalámbrica del sistema 74
Figura 27:Página web de gestión de la herramienta 74
Figura 28: Caja de control. 75
Figura 29: Estados del sistema. 75

Índice de tablas:

Tabla 1: Calendario principal de fases del proyecto. 16
Tabla 2: Muestra base de transcripción. 19
Tabla 3 :Características de la muestra base de transcripción. 19
Tabla 4: Costes Google Speechtext 20
Tabla 5:Limites IBM Watson 21
Tabla 6:Lenguajes soportados: IBM Watson 22
Tabla 7:Latencia en la muestra IBM Watson 23
Tabla 8:Errores en la transcripción IBM Watson 23
Tabla 9:Límites de Microsoft Speech Cognitive Services 24
Tabla 10:Latencia en la muestra: Microsoft Speech Cognitive Services 26
Tabla 11:Errores en la muestra: Microsoft Speech Cognitive Services 26
Tabla 12:Limites de servicio Wit.AI 26

Tabla 13:Latencia en la muestra.Wit.AI 30
Tabla 14:Errores en la muestra.Wit.AI 30
Tabla 15:Servicios de traducción seleccionados en el proyecto. 32
Tabla 16:Texto de muestra para el análisis de traducción 32
Tabla 17:Límites del servicio de traducción de Microsoft Translate 33
Tabla 18:Latencia en la muestra. Microsoft Translate 34
Tabla 19:Errores en la muestra. Microsoft Translate 35
Tabla 20:Limites del servicio IBM Translator 35
Tabla 21:Tabla 22:Latencia en la muestra. IBM Translator 36
Tabla 22: Errores en la muestra. IBM Translator 36
Tabla 23:Límites en el servicio GoogleTrans 36
Tabla 24:Latencia en la muestra. GoogleTrans 37
Tabla 25:Errores en la muestra. GoogleTrans 37
Tabla 26:Límites en el servicio *Yandex Translate* 38
Tabla 27:Latencia en la muestra: Yandex Translate 38
Tabla 28:Errores en la muestra: Yandex Translate 38
Tabla 29:Contenido textual de la muestra .Síntesis de voz 40
Tabla 30:Resumen de latencia y errores en el procesado de la muestra. Síntesis de voz. 41
Tabla 31:Comparativa funcional de las de conversión texto a voz 41
Tabla 32:Soluciones aplicadas en el proyecto Speak-on 42
Tabla 33:Comparativa de dispositivos electrónicos base. 44
Tabla 34:Relación de conexiones Omega y Atmega 328 44
Tabla 35:Listado de componentes electrónicos del proyecto. 45
Tabla 36:Árbol de directorios y estructura de ficheros. 49
Tabla 37:Coste variable unitario. 58
Tabla 38:Estimación producción anual. 58
Tabla 39:Costes de producción anuales. 58
Tabla 40:Coste total unitario 59
Tabla 41:Beneficio/unidad 59
Tabla 42:Estimación inversión anual. 59
Tabla 43:Retorno de inversión 59
Tabla 44:Previsión de ventas 59
Tabla 45:Localización de ficheros de diseño y test 61
Tabla 46:Localización de ficheros del proyecto 61

1. Introducción

En la actualidad, existen numerosas hipótesis sobre el origen del lenguaje, aunque, no hay evidencias que confirmen las teorías ni una garantía científica en las propuestas. Sin embargo, es posible afirmar que en la comunicación de los seres humanos el uso de los signos es una herramienta imprescindible, puesto que, nos permiten transmitir información o representar determinadas acciones.

Los signos utilizan dos vías claramente diferenciadas en la transmisión de los mensajes: la vía no lingüística (sonidos determinados, información visual y gestos) y la vía lingüística, donde el lenguaje verbal y el lenguaje escrito son utilizados como transporte del pensamiento e ideas. Por consiguiente, es posible identificar en los lenguajes dos sentidos relacionados con este modelo comunicativo: la visión y la audición. (1)

Precisamente, desde una perspectiva personal, el TFG se ha centrado en utilizar los signos lingüísticos como eje funcional, donde la existencia de una discapacidad visual o auditiva en el individuo se ha considerado un importante obstáculo en la comunicación que es imprescindible abordar, debido en gran parte a la afectación que provoca esta situación en el bienestar social.

La necesidad de explorar diferentes soluciones ha surgido de un proceso previo de observación del entorno, en el que se ha determinado la magnitud de proporcionar una solución técnica y de aportar un recurso alternativo aplicable a personas que padecen este tipo de discapacidad, con la finalidad de minimizar las barreras, favorecer su integración en la sociedad y aportar un mayor grado de accesibilidad cognitiva.

Asimismo, la accesibilidad de los sistemas de la información es un valor recogido en la convención de la ONU[1] de los derechos de las personas con discapacidad .En particular ,en el artículo cuatro del documento, se ha reflejado la siguiente directriz: ”*Emprender o promover la investigación el desarrollo y promover la disponibilidad y el uso de nuevas tecnologías, incluidas las tecnologías de la información y las comunicaciones, ayudas para la movilidad, dispositivos técnicos y tecnologías de apoyo adecuadas para las personas con discapacidad, priorizando aquellas herramientas con un coste reducido*”. (2)

En definitiva, el estudio y desarrollo del TFG expuesto pretende promover y demostrar que la conjunción de un pequeño procesador y el sostén de una placa electrónica Arduino de bajo coste son elementos suficientes para construir un prototipo de unas lentes de ayuda cognitiva que facilitarán la accesibilidad y el conocimiento en múltiples ámbitos.

[1] Organización de las Naciones Unidas creada, con la finalidad de, proteger los derechos humanos y proyectar la paz a nivel internacional.

2. Objetivos

Los objetivos de la investigación y desarrollo del proyecto expuesto han sido divididos, principalmente, en dos grupos: por una parte, se efectuará un examen cuantitativo en el que se valorarán los parámetros del ejercicio con la ayuda de una perspectiva paramétrica, por otra parte, se elaborará una valoración cualitativa del nivel de los resultados.

2.1 Principales:

- Construir un dispositivo electrónico de visualización que realice una traducción en siete idiomas: catalán, inglés, francés, italiano, alemán, portugués y español mediante una captura previa del audio y posterior procesado.
- Realizar una transcripción automática de una conversación de forma eficaz y fiable con una latencia de respuesta inferior a 4 segundos y con un nivel de error < 10%.
- Traducción textual con un flujo de retorno de la API < 4 segundos.
- Identificar sistemas artificiales con una latencia de síntesis de voz <= 3 segundos/petición.
- Ofrecer una herramienta basado en Arduino de apoyo cognitivo accesible a personas con déficit auditivo/visual con un único elemento de disparo (un botón).
- Presentar la información a nivel audiovisual :voz digital sintetizada y texto en un *display TFT* de 1.44” integrando la solución en unas lentes.

2.2 Secundarios:

- Análisis de la integración de la solución bajo distintas plataformas de desarrollo.
- Investigación de las diferentes soluciones de conversión voz-texto, traducción multiplataforma y síntesis vocal.
- Empleo del lenguaje de programación *Python* como eje operativo del desarrollo del dispositivo. (3)
- Transmisión de lenguaje natural en las voces utilizadas.
- Diseño y elaboración de un prototipo funcional de Speak-On
- Adaptación y optimización de un sistema Linux embebido *Openwrt*[2] en el entorno.

[2] Sistema operativo Linux enfocado a dispositivos integrados.

3. Contenido

El estudio y desarrollo técnico del proyecto Speak-On se ha dividido en cuatro módulos: bloque de conversión voz a texto, bloque de traducción, síntesis de voz y finalmente, presentación audiovisual. En el desarrollo de los bloques, se han generado distintos algoritmos y se han analizado múltiples API[3] *'application programming interface*' (interfaz de programación de aplicaciones) ,con el principal objetivo de optimizar el resultado, reducir el impacto negativo de las peticiones y ofrecer al usuario final una experiencia positiva .Del mismo modo ,la exploración de los diversos bloques se ha elaborado utilizando dos lenguajes de programación, con el propósito de ,identificar que opción es más óptima incluir en las llamadas a las API : PHP '*Personal Hypertext processor* '(preprocesador de hipertexto) gestionado con cURL (URL de cliente que facilita la interconexión con servidores) y *Python* un lenguaje flexible de fácil legibilidad e integración que ha permitido el desarrollo de una solución multiplataforma.

En el primer capítulo, se ha realizado un análisis del bloque STT '*Speech To Text'* (conversión de voz a texto). Su principal función ha sido generar el procedimiento de transcripción de una conversación y almacenar su contenido, con la intención de, ser tratado en el bloque de traducción.

En el segundo capítulo, se ha examinado el modelo lingüístico y se ha efectuado un estudio de los diferentes sistemas de traducción: Microsoft, IBM, Google, Yandex, etc. Del mismo modo, se han obtenido datos objetivos, con el fin de, identificar el principal motor de traducción en el dispositivo.

En el tercer capítulo, se han examinado diversos métodos de síntesis de voz [4], sistemas que han sido capaces de demostrar un lenguaje natural e integridad en las voces ,características esenciales vinculadas al nivel de entonación ,un parámetro esencial en el desarrollo del procedimiento TTP *' text-to-phoneme'* (transcripción fonética) en el que se ha efectuado una conversión de texto y ortografía a una cadena de fonemas.

En el último capítulo, se ha efectuado la integración final del *software* en el bloque de presentación audiovisual. El programa ha sido ejecutado en un sistema *Linux* embebido[5] e integrado en un dispositivo electrónico Arduino.

De igual manera, este último elemento ha sido el responsable de la gestión y explotación del resto de bloques operativos incluyendo la presentación del contenido en un display TFT. '*Thin Film Transistor-Liquid Crystal Display*' (Transistor de película fina, pantalla de cristal líquido).

---

[3] Las API son utilizadas con el propósito de incluir las funcionalidades de un servicio externo.

[4] La síntesis de voz es el proceso de simulación artificial de la voz humana en una computadora.

[5] En un sistema operativo embebido el *software* está vinculado a la arquitectura del *hardware.*

Finalmente, se ha diseñado un modelo funcional, se ha analizado la viabilidad económica de la comercialización del producto y se han expuesto las principales conclusiones del trabajo realizado.

4. Metodología

En la elaboración del proyecto, se ha necesitado incorporar en el documento una investigación cuantitativa, con el principal propósito de, disponer de información relevante de las diversas variables delimitadas ,con la ayuda de parámetros numéricos.

Para ello, se ha realizado un test operativo en las versiones de las API implicadas en el análisis, principalmente en los bloques de conversión voz-texto y de traducción. En la muestra se ha utilizado un texto previamente definido, se han contabilizado las palabras y los caracteres y se han determinado los porcentajes de error en los resultados. Adicionalmente, se ha medido la latencia en segundos de las respuestas proporcionadas por las diversas soluciones utilizando como herramienta un cronómetro web en la siguiente dirección: http://online-stopwatch.chronme.com/.

En los test SST y *Translate* se han empleado dos lenguajes de programación*: Python*[6] y PHP. El uso de estos lenguajes se ha justificado en gran parte debido a la necesidad de medir el tiempo de ejecución y cuantificar el retorno de las respuestas, dos características con un impacto directo en el rendimiento y en la experiencia del usuario. Por lo tanto, se ha descartado la admisión de determinadas soluciones que no han superado el test debido al incumplimiento de las restricciones temporales o el desbordamiento de los errores contabilizados.

En particular, en el bloque de síntesis de voz se ha realizado un estudio híbrido vinculado a la dificultad de ofrecer un resultado matemático, por un lado, se han observado las diferencias de los resultados del bloque de síntesis del habla desde una perspectiva cualitativa, por otro lado, se han cuantificado los retardos en la respuesta de la síntesis artificial. Por último, en la integración final y presentación audiovisual, se han analizado las características técnicas del *hardware* que mejor se adapten al dispositivo Speak-On.

[6] Python es un lenguaje de programación interpretado (no es necesario compilar el código) es procesado por un intérprete integrado.

5. Planificación.

Se ha realizado una planificación temporal vinculada a las fechas propuestas en la entrega del TFG del Grado de Multimedia.

5.1 Intervalo temporal de las fases del proyecto.

Descripción	Inicio	Final	Días	
FASE DE INICIO	**21-09-18**	**27-09-18**	**6.5**	**13.0**
Propuesta	21-09-18	22-09-18	1.5	
Determinación de objetivos.	23-09-18	24-09-18	1.5	
Estimación de recursos necesarios.	24-09-18	25-09-18	1.5	
Analisis de riesgos	26-09-18	26-09-18	1.0	
Estimación del tiempo necesario y coste del ...	27-09-18	27-09-18	1.0	
FASE DE PLANIFICACIÓN	**28-09-18**	**07-10-18**	**10.0**	**20.0**
Analisis de Alcance	28-09-18	30-09-18	3.0	
Definición del plan del proyecto	01-10-18	02-10-18	2.0	
Descomposición de procesamiento vocal	03-10-18	04-10-18	1.5	
Descomposición en los sistemas de traducción	04-10-18	05-10-18	1.5	
Sintesis de voz.Alcance y determinación	06-10-18	06-10-18	1.0	
Presupuestos.Determinación ecónomica	07-10-18	07-10-18	1.0	
FASE DE EJECUCIÓN	**08-10-18**	**20-12-18**	**73.5**	**73.5**
Ensamblaje y conexionado	**08-10-18**	**11-10-18**	**4.0**	**8.0**
Integración estructural.	08-10-18	09-10-18	2.0	
Revisión funcional	10-10-18	11-10-18	2.0	
Logica de registro vocal	**12-10-18**	**16-10-18**	**4.5**	**8.5**
Algoritmos de captura y tratamiento de audio	12-10-18	14-10-18	2.5	
Captación Acustica	15-10-18	16-10-18	1.5	
Lógica de procesamiento I	**16-10-18**	**14-11-18**	**29.5**	**58.0**
Algoritmos de lenguaje natural	16-10-18	30-10-18	14.0	
Algoritmos de tratamiento de conversión	31-10-18	14-11-18	14.5	
Lógica de procesamiento II	**15-11-18**	**08-12-18**	**24.0**	**48.0**
Algoritmos de traducción de texto	15-11-18	23-11-18	8.5	
Algoritmos de presentación visual	23-11-18	01-12-18	8.5	
Implantación de sintesís de voz	02-12-18	04-12-18	2.5	
Logica funcional (Secuencia y flujo de optimi...	04-12-18	08-12-18	4.5	
Test y validación	**09-12-18**	**14-12-18**	**5.5**	**11.0**
Test de procedimientos desarrollados y bloqu...	09-12-18	14-12-18	5.5	
Ingeniería de producto	**14-12-18**	**20-12-18**		**12.0**
Diseño básico del producto	14-12-18	20-12-18	6.0	
Construcción y validación del prototipo final.	14-12-18	15-12-18	1.0	
Rediseño	15-12-18	18-12-18	3.0	
Conclusiones.Evaluación final.	18-12-18	20-12-18	2.0	
FASE DE CIERRE	**20-12-18**	**13-01-19**	**24.5**	**49.0**
Cierre formal del proyecto .Entrega del TFG	20-12-18	03-01-19	14.0	
Presentación y Codigo	03-01-19	13-01-19	10.5	

Figura 1: Fases del proyecto

5.2 Calendario de Inicio de las fases del proyecto.

DETALLE	FECHA	DESCRIPCIÓN
Inicio del proyecto	21-09-2018	Fase de inicio y desarrollo del contenido.
Planificación	28-09-2018	Organización y preparación del flujo del proyecto.
Ejecución	08-10-2018	Desarrollo de la solución en los diferentes bloques y elementos multimedia.
Cierre	20-12-2018	Cierre formal del proyecto y preparación de documentación.

Tabla 1: Calendario principal de fases del proyecto.

5.3 Diagrama de *Gantt*.

Septiembre 2.018- octubre 2.018

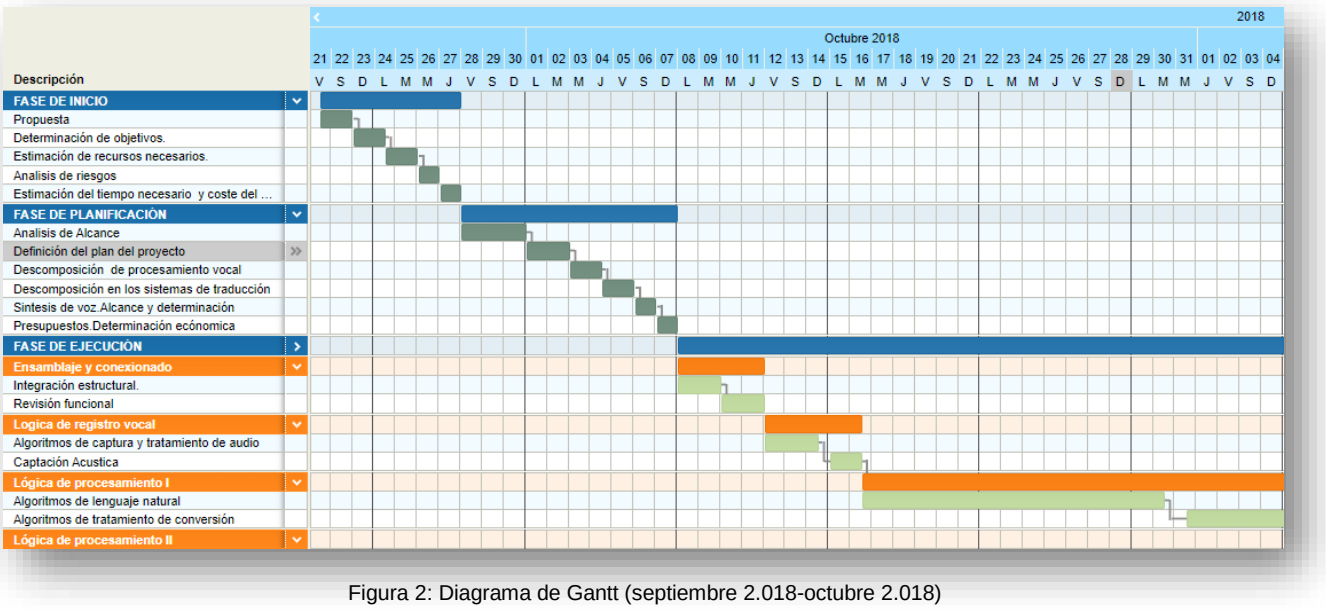

Figura 2: Diagrama de Gantt (septiembre 2.018-octubre 2.018)

Noviembre 2.018 - diciembre 2.018

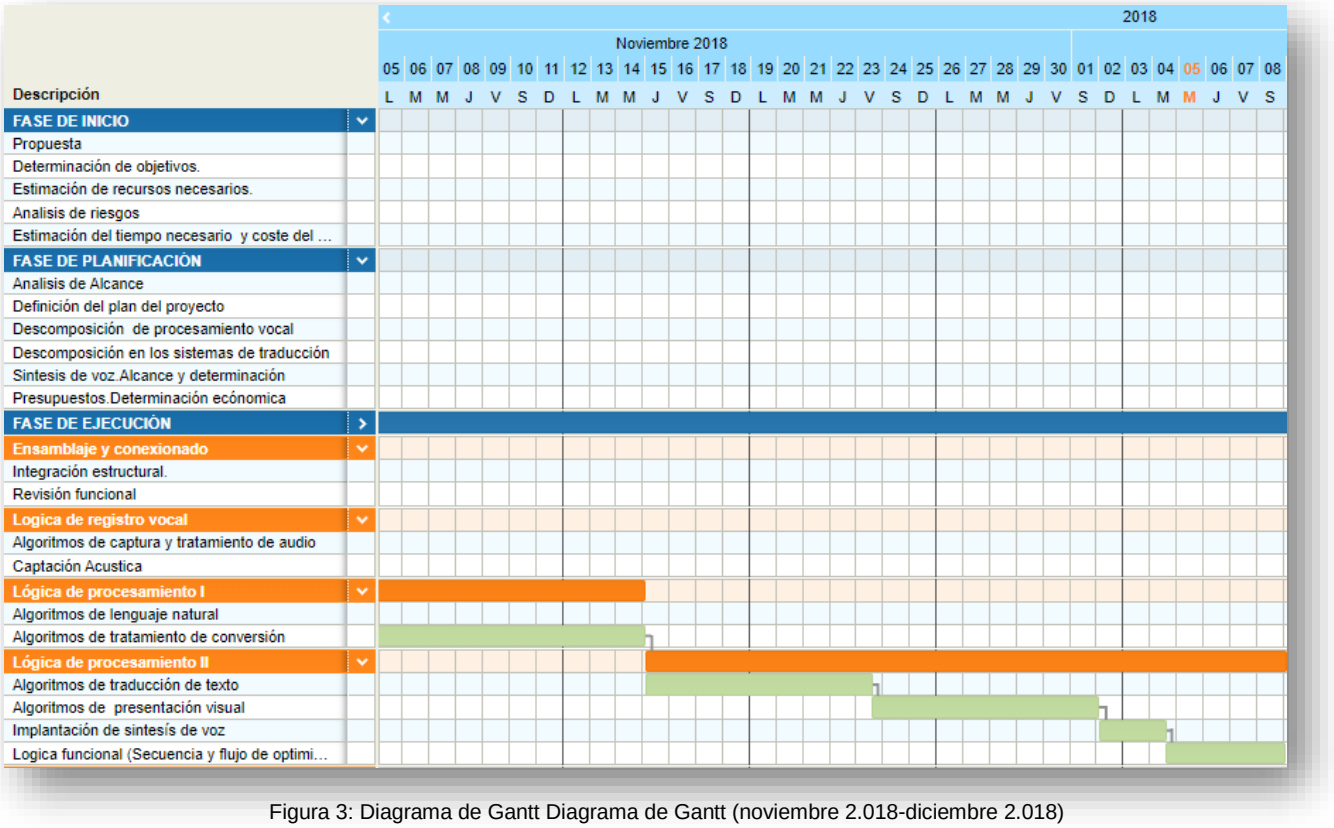

Figura 3: Diagrama de Gantt Diagrama de Gantt (noviembre 2.018-diciembre 2.018)

Diciembre 2.018 - enero 2.019

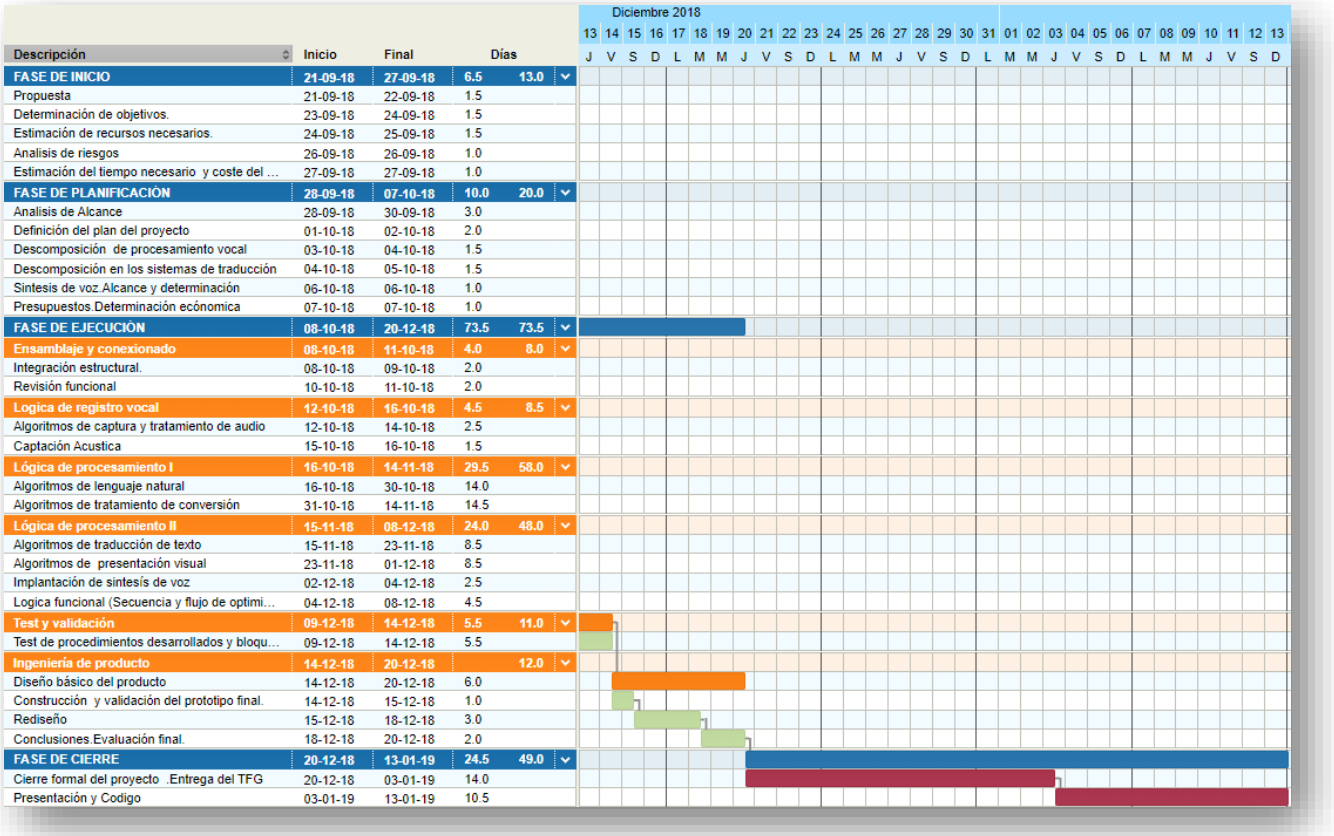

Figura 4:Diagrama de Gantt (diciembre 2.018-enero 2.019)

6. Proceso de trabajo.

El proyecto se ha dividido en varias fases: en primer lugar, se ha elaborado una propuesta del proyecto, se han determinado los objetivos del análisis del contenido y se ha fijado el alcance de cada uno de ellos incluyendo una estimación temporal de la dedicación.
En segundo lugar, se ha planificado la línea de trabajo y se ha estructurado la documentación, principalmente, en cuatro etapas diferenciadas:

- Conversión voz a texto [7](implementación y test de algoritmos).
- Traducción (implementación y test de algoritmos).
- Síntesis de voz (implementación y desarrollo de algoritmos).
- Presentación e integración (implementación).

El principal objetivo de la división del contenido, ha sido facilitar obtener resultados de manera aislada, con el objeto de, interconectar las etapas en el flujo operativo final de la solución. Al mismo tiempo, se han definido los límites y alcance de los sistemas funcionales. Esta medida ha evitado dispersar el estudio y generar falsas expectativas, centrando el contenido en el cumplimiento de los objetivos (en su gran mayoría cuantitativos). Es importante señalar que los periodos y entregables del TFG han estado vinculados a la entrega de las PEC del trabajo final de grado.
En tercer lugar, se ha efectuado la fase de ejecución del proyecto donde se ha ensamblado el prototipo de Speak-On en una placa de laboratorio. En esta etapa, se han realizado dos entregas parciales del contenido del proyecto: por un lado, la adaptación del sistema operativo y algoritmo TTS , por otro lado, se ha desarrollado el proceso de traducción y síntesis vocal.
De igual forma, se ha implementado la lógica funcional y se ha desarrollado el *software* de las diversas soluciones con la ayuda de un proceso cíclico (diseño-prueba-validación) donde se ha simulado un entorno de producción.
Por último, se ha presentado la documentación e investigación del trabajo, donde se ha incluido el código, totalidad de las fases y el cierre formal del mismo.

[7] En el proceso de transcripción se interpretan los fonemas en el contexto de la conversación.

7. Bloque de conversión voz-texto (STT)

El reconocimiento automático del habla (RAH) es un procedimiento ampliamente complejo en el que intervienen múltiples variables: la pragmática, semántica fonética, etc. Es importante resaltar que el resultado de la combinación y el procesamiento de estas variables no siempre ofrecen un resultado satisfactorio, por tanto, se ha trabajado con un porcentaje de error inferior al 10%.

Se han analizado diferentes API de amplio reconocimiento evitando *websockets* [8](ejecución en tiempo real) ya que estos no ofrecen estabilidad suficiente para el proyecto debido en gran parte a la excesiva demanda de hardware. El principal objetivo ha sido identificar una API eficaz que ofrezca un retardo mínimo(<4seg) con una calidad óptima (<10% errores).

Para poder analizar los resultados de la conversión voz-texto se ha registrado un audio en formato WAV con un muestreo de 48Khz y 16 bits. Este audio será el utilizado en el análisis de las distintas versiones. El texto del audio ha sido el siguiente:

MUESTRA UTILIZADA EN EL PROCESO DE TRANSCRIPCIÓN
Esta grabación corresponde a un análisis de pruebas del trabajo final de grado de multimedia. Se comprobarán los errores detectados en el bloque de conversión de voz a texto.

Tabla 2: Muestra base de transcripción.

PALABRAS	**CARACTERES SIN ESPACIOS**	**NUMERO TOTAL DE CARACTERES**
29	145	173

Tabla 3 :Características de la muestra base de transcripción.

Es reseñable que en la petición de conversión se utilizarán dos lenguajes de programación diferentes con el fin de identificar qué resultados son más estables y dinámicos: PHP gestionado por *cURL* y *Python*.

[8] *websocket* es un protocolo que permite la apertura de una canal bidireccional en la web utilizando una única conexión TCP/IP

cURL es una librería que permitirá a PHP conectarse con los servidores de los distintos procesos adaptando la petición a cada estructura presentada por el fabricante, de esta manera, se optimizarán las peticiones y la comparativa final.

Por otro lado, *Python* se ha integrado en varias soluciones del bloque de conversión de voz a texto, además, ha sido uno de los lenguajes definidos en el proyecto ,con el principal objetivo de comunicar los bloques y el flujo de trabajo.

7.1 Google SpeechText

Google dispone de un servicio de implantación de sus principales herramientas, una plataforma en la nube[9] donde facilita a los desarrolladores de software la integración de sus diferentes soluciones. En concreto SpeechText trabaja utilizando redes neuronales y múltiples funcionalidades lingüísticas (120 idiomas) siendo una de las soluciones con mayor proyección existentes en el sector.

No obstante, se han detectado dos características que debilitan su posicionamiento en el estudio e implantación en el modelo: la autorización previa de uso en un dispositivo integrado como es el caso del proyecto expuesto (no se ha obtenido una respuesta) y el límite temporal en modo test en este caso es factible superar la barrera de 60 minutos en las pruebas y configuración inicial de la API.

GOOGLE SPEECHTEXT
Reconocimiento de voz (todos los modelos excepto vídeo) (Máximo 60minutos).Una vez superada esta barrera se facturaría a razón de 0,006 € cada 15 segundos

Tabla 4: Costes Google Speechtext

7.2 IBM Watson

IBM proporciona a los desarrolladores una API con amplias características que permite realizar transcripciones de voz combinando la señal de audio recogida con la información de la estructura del lenguaje, además, dispone de un amplio abanico de formatos de audio. Utiliza los principales interfaces de programación para su integración (*REST HTTP*[10]*, WebSocket*) (4)

Para realizar el análisis, ha sido fundamental crear un servicio *Watson* del tipo *Speech Text* y acceder a la configuración de la API, en el caso del proyecto Speak-On se ha habilitado una cuenta Lite para realizar pruebas de carga. (5)

[9] Entorno productivo de servicios y funcionalidades en red que ayudarán a reducir costes en el desarrollo de soluciones informáticas.

[10] Entorno de intercambio de datos mediante el protocolo de transferencia de hipertexto(HTTP)

IBM Watson. Credenciales de la API.

Una vez registrado el servicio STT en la web de IBM, se facilitará un usuario/contraseña y una URL para realizar las peticiones vía HTTP.

Figura 5:Contenido de la API de IBM Watson Speech

IBM Watson. Principales características.

IBM WATSON	SERVICIO	LIMITACIONES EN EL PROCESAMIENTO
Versión LITE	SpeechText	100 minutos /mes

Tabla 5:Limites IBM Watson

Transcripción del audio en siete idiomas en tiempo real y bajo petición *HTTP: REST HTTP*, *Websocket*, HTTP asíncrono) (6)

El modelo de procesamiento es personalizable (es posible registrar previamente términos de difícil conversión con el fin de adaptar el resultado) ofreciendo una importante mejora en la precisión del contenido. Por otro lado, la plataforma permite enviar grabaciones en formato WAV[11] y también trabajar directamente con el registro de un micrófono, en este caso se ha optado por enviar el fichero previamente registrado.

IBM Watson. Lenguajes soportados.

IBM dispone de dos formatos de audio adaptados a los diferentes algoritmos de procesado:

El formato *Broadband Model* (Banda Ancha) se utilizará para audios con un muestreo superior a 16Khz, optimizado para un entorno en tiempo real.

[11] Formato de audio sin compresión creado por IBM y Microsoft en 1991 '*WAVE form audio file format'* (formato archivo de audio)

El formato *NarrowBand Model* (Banda Estrecha) se aplicará a audios con muestreos a 8 kHz. Esta tasa se ejecutará normalmente en audio telefónico. Los lenguajes disponibles son los siguientes:

Language	Broadband model	Narrowband model
Brazilian Portuguese	pt-BR_BroadbandModel	pt-BR_NarrowbandModel
French	fr-FR_BroadbandModel	Not supported
German	de-DE_BroadbandModel	Not supported
Japanese	ja-JP_BroadbandModel	ja-JP_NarrowbandModel
Korean	ko-KR_BroadbandModel	ko-KR_NarrowbandModel
Mandarin Chinese	zh-CN_BroadbandModel	zh-CN_NarrowbandModel
Modern Standard Arabic	ar-AR_BroadbandModel	Not supported
Spanish	es-ES_BroadbandModel	es-ES_NarrowbandModel
UK English	en-GB_BroadbandModel	en-GB_NarrowbandModel
US English	en-US_BroadbandModel	en-US_NarrowbandModel

Tabla 6:Lenguajes soportados: IBM Watson

IBM Watson. Implantación y test con lenguaje PHP.

Para realizar el test en PHP se ha diseñado una función que recibirá una variable de texto (lenguaje de salida) y retornará el texto traducido (reconocimiento del idioma de entrada de forma automática) donde se decodificará previamente el objeto *json*[12].

[12] '*JavaScript Object Notation*' **(**notación de objeto de JavaScript), formato de intercambio de datos.

En la solicitud a la API de IBM, un bucle concatenará en una variable el contenido final .Esta solución seccionará el texto en varias posiciones del *array*[13]. La solicitud se ha ejecutado bajo terminal utilizando *cURL* en *localhost*.

```
function func_speechtext_uoc ($language){
$json=shell_exec('curl -X POST -u 512dd512-dec9-4ee4-b159-86e4f3f6fc80:SX5WOHYsmyYB -
-header "Content-Type: audio/wav" --data-binary @/root/records/voice.wav
"https://stream.watsonplatform.net/speech-to-
text/api/v1/recognize?model='.$language.'"');
       // Decoding JSON data to PHP associative array
       $obj = json_decode($json,true);

       //var_dump ($obj);

       $string_complete="";

              // Loop through the associative array
                     for($i=0; $i<count($obj['results']); $i++) {
       $string_complete.=$obj['results'][$i]["alternatives"][0]["transcript"]. " ";}
       return  $string_complete;}
$lang="es-ES_BroadbandModel";
echo func_speechtext_uoc($lang);
```

IBM. Watson. Implantación y test con lenguaje Python.

En *python* se ha utilizado *requests*[14] donde se ha recuperado el objeto *json* convertido a cadena de texto.

IBM Watson. Resultados STT.

LATENCIA				
IBM WATSON. SpeechText	**PHP**		***PYTHON***	
	Tiempo(s)	4.3	Tiempo(s)	3.8

Tabla 7:Latencia en la muestra IBM Watson

% ERRORES EN LA TRANSCRIPCIÓN				
IBM WATSON. SpeechText	**PHP**		***PYTHON***	
	Errores %	8	Errores %	8

Tabla 8:Errores en la transcripción IBM Watson

[13] Conjunto de elementos de la misma tipología utilizado en programación.
[14] Librería de Python para la gestión de solicitudes HTTP y envío de datos.

La versión en Python ha sido superior en los tiempos de latencia, por otro lado, se ha detectado un error en 5 caracteres que ha penalizado el resultado del test. (8%).

7.3 Microsoft Speech Cognitive Services.

Microsoft propone una solución muy flexible y estable que permite trabajar con audio de *streaming*[15], asimismo, acepta la transcripción por lotes de voz a partir de grabaciones de audio en tiempo real, reconocimiento del audio procedente del micrófono.

En este caso, también es posible personalizar la API conforme a las necesidades de determinadas transcripciones y datos complejos de identificar. Adicionalmente, contempla la gestión de la detección de fin de voz, el enmascaramiento de palabras soeces y el trabajo con *Language Understanding* (LUIS) una solución que pretende comprender lo que un sujeto desea transmitir con sus propias palabras. (7)

MICROSOFT SPEECH	SERVICIO	LIMITES
Versión FREE	*SpeechText*	300 minutos /mes Máximo enviado 14 segundos por petición.

Tabla 9:Límites de Microsoft Speech Cognitive Services

En las pruebas realizadas, el audio del elemento TEST.WAV se ha codificado a un muestreo de 16KHZ, PCM[16] '*Pulse Code Modulation*'(modulación por pulsos codificados) ,uno de los requisitos que Microsoft ha reportado en su documentación.

En este caso, los algoritmos diseñados se han generado de forma independiente (de manera similar a IBM WATSON), ya que el soporte de las API de Microsoft Cognitive service trabajan únicamente con UWP, C#, C++, NET, JAVA por lo tanto se ha adaptado el diseño a la API de REST tanto en Python como en PHP.

Microsoft Speech Cognitive Services. Lenguajes soportados.

Árabe, catalán, danés, alemán, inglés, español, finlandés, francés, hindú, italiano, japonés, coreano, noruego, holandés, polaco, portugués, ruso, sueco, chino, tailandés.

Microsoft Speech Cognitive Services. Implantación y test con lenguaje PHP.

Se ha diseñado una función que recibe un código de lenguaje y retornará el audio convertido en una cadena de texto. Se ha decodificado el resultado *json* y se ha recuperado el valor de la posición ["`DisplayText`"]

[15] Tecnología que facilita la reproducción del contenido multimedia sin una descarga previa de fichero.
[16] En la modulación PCM se realiza un muestreo de la amplitud de la señal analógica para su digitalización.

```
function func_speechtext_uoc ($language){

	$restAPI="curl -X POST -H \"Content-
type:audio/wav;codec=audio/pcm;samplerate=16000\" -H \"Ocp-Apim-Subscription-
Key:b60420bced564e419521aa7705a2c9bc\" --data-binary @/root/records/voice2.wav
\"https://westeurope.stt.speech.microsoft.com/speech/recognition/conversation/cognitives
ervices/v1?language=".$language.'"';

	$json=shell_exec($restAPI);
	//Decoding JSON data to PHP associative array
	$obj = json_decode($json,true);

	//var_dump ($obj);
	//read value
	  $string=$obj["DisplayText"];
	return  $string;
	}
$lang="es-ES";
 echo func_speechtext_uoc($lang);
```

Microsoft Speech Cognitive Services.. Implantación y test con lenguaje Python

Se ha ejecutado una petición bajo la librería *requests* en Python y se ha rescatado la primera posición del objeto *json* convertido a cadena de texto. Se ha configurado la variable original en el lenguaje español `'es-ES'` para efectuar las pruebas.

```
import json
import sys
import subprocess
url =
'https://westeurope.stt.speech.microsoft.com/speech/recognition/conversation/cognitive
services/v1?language=es-ES'
apiKey = "b60420bced564e419521aa7705a2c9bc"
headers = {'Content-Type': 'audio/wav',"Ocp-Apim-Subscription-Key": apiKey}
audio_path = open('/root/records/voice.wav', 'rb')
response = requests.post(url, data=audio_path, headers=headers)
data = json.loads(response.content)
print (data['DisplayText'])
```

Microsoft Speech Cognitive Services. Resultados STT.

LATENCIA				
MICROSOFT SPEECH COGNITIVE	**PHP**		**PYTHON**	
	Tiempo(s)	4.0	Tiempo(s)	4.5

Tabla 10:Latencia en la muestra: Microsoft Speech Cognitive Services

% ERRORES EN LA TRANSCRIPCIÓN				
MICROSOFT SPEECH COGNITIVE	**PHP**		**PYTHON**	
	Errores %	0	Errores %	0

Tabla 11:Errores en la muestra: Microsoft Speech Cognitive Services

La versión en PHP ha demostrado ser más óptima en el valor de la latencia. Por otro lado, Python ha aportado legibilidad en el código.

7.4 Wit.AI

Wit.AI es un *startup* cuyo objetivo es estandarizar el reconocimiento de voz bajo su plataforma. Actualmente tiene aproximadamente 7.000 desarrolladores implicados en el SDK '*Software Development Kit*' (kit de desarrollo de software) de Wit.AI, principalmente, fortalecida por la adquisición de la plataforma e inyección económica de la compañía Facebook.
La API de WIT.AI soporta una petición/seg. Se ha considerado una latencia óptima para el análisis e integración. Una de las principales características disponibles, es la gestión mediante comandos de voz en determinadas tareas donde es posible elaborar acciones específicas. Asimismo, es importante reflejar que la API proporcionada no está enfocada para trabajar con una gestión natural STT, no es posible ejecutar una petición de reconocimiento de habla con la ayuda del envío de la variable del lenguaje original, sin embargo, se ha realizado una adaptación perfectamente funcional del sistema, con el objeto de, incorporar la librería en el proyecto. (8)

WIT.AI	**SERVICIO**	**LIMITES**
WIT.AI	*SpeechText*	Sin limitaciones. *OpenSource*

Tabla 12:Limites de servicio Wit.AI

Wit.AI STT. Lenguajes soportados.

Los lenguajes soportados por la plataforma WIT.AI son numerosos (entorno a los 60 idiomas), además, son dinámicos (permanecen en constante evolución). Es importante señalar, que Wit.AI dispone de una comunidad de desarrolladores y expertos en idiomas que alimentan la plataforma y depuran los errores detectados. Los principales idiomas son los siguientes: Africano, albanés, árabe, azerbaiyano, bengalí, bosnio, búlgaro, birmano, catalán , chino, croata, checo, danés, holandés, inglés, estonio, finlandés, francés, georgiano, alemán, griego, groenlandés, hebreo , Hindi, húngaro, islandés, igbo, indonesio, inuktitut, italiano, japonés, canarés, kinyarwanda, coreano, lao, latín, letón, lituano, macedonio, malayo, maorí, mongol, nepalí, noruego, pasto, persa, polaco, portugués , Rumano, ruso, serbio, eslovaco, esloveno, somalí, español, etc..

Wit.AI.STT. Adaptación al proyecto.

Para afrontar la integración de las diferentes versiones en Python y PHP se ha necesitado una adaptación funcional:

Wit.AI no dispone de una API con la posibilidad de enviar como parámetro el lenguaje, no obstante, si permite crear en su web una aplicación configurada en un idioma concreto que reconocería el audio. Por lo tanto, para trabajar con los siete idiomas del proyecto, se han tenido que crear varias aplicaciones distintas, cada una configurada en el lenguaje original mediante el uso de *Tokens* [17]independientes.

En consecuencia, las aplicaciones tendrán activa una clave que será utilizada en las peticiones a la URL.

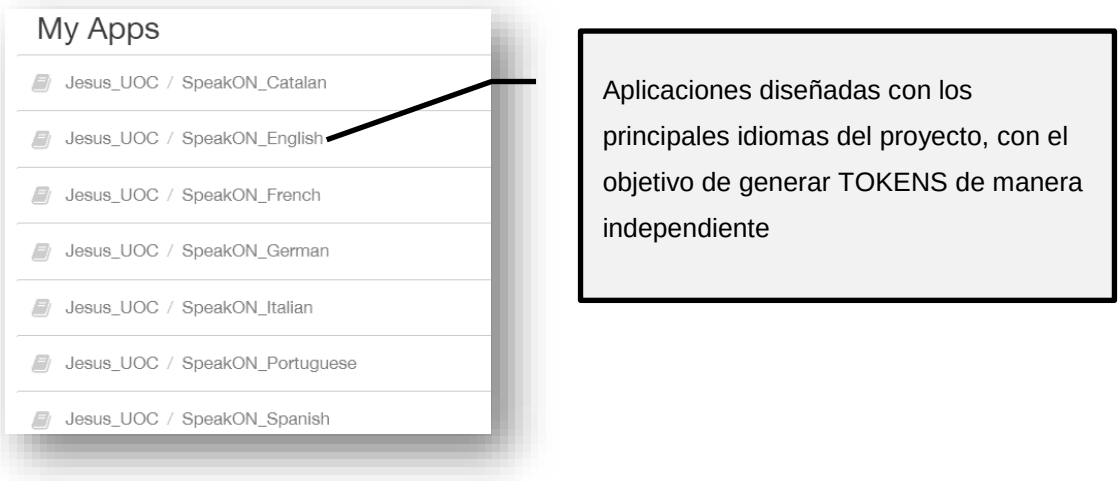

Figura 6:Aplicaciones activas en la página web de wit.ai para el proyecto

[17] *Bearer tokens* (token de portador) es un protocolo de seguridad vinculado a una cadena que identifica a un usuario y su configuración en la plataforma.

Es importante señalar que al crear una aplicación ha sido fundamental configurar la ubicación para trabajar con su clave.

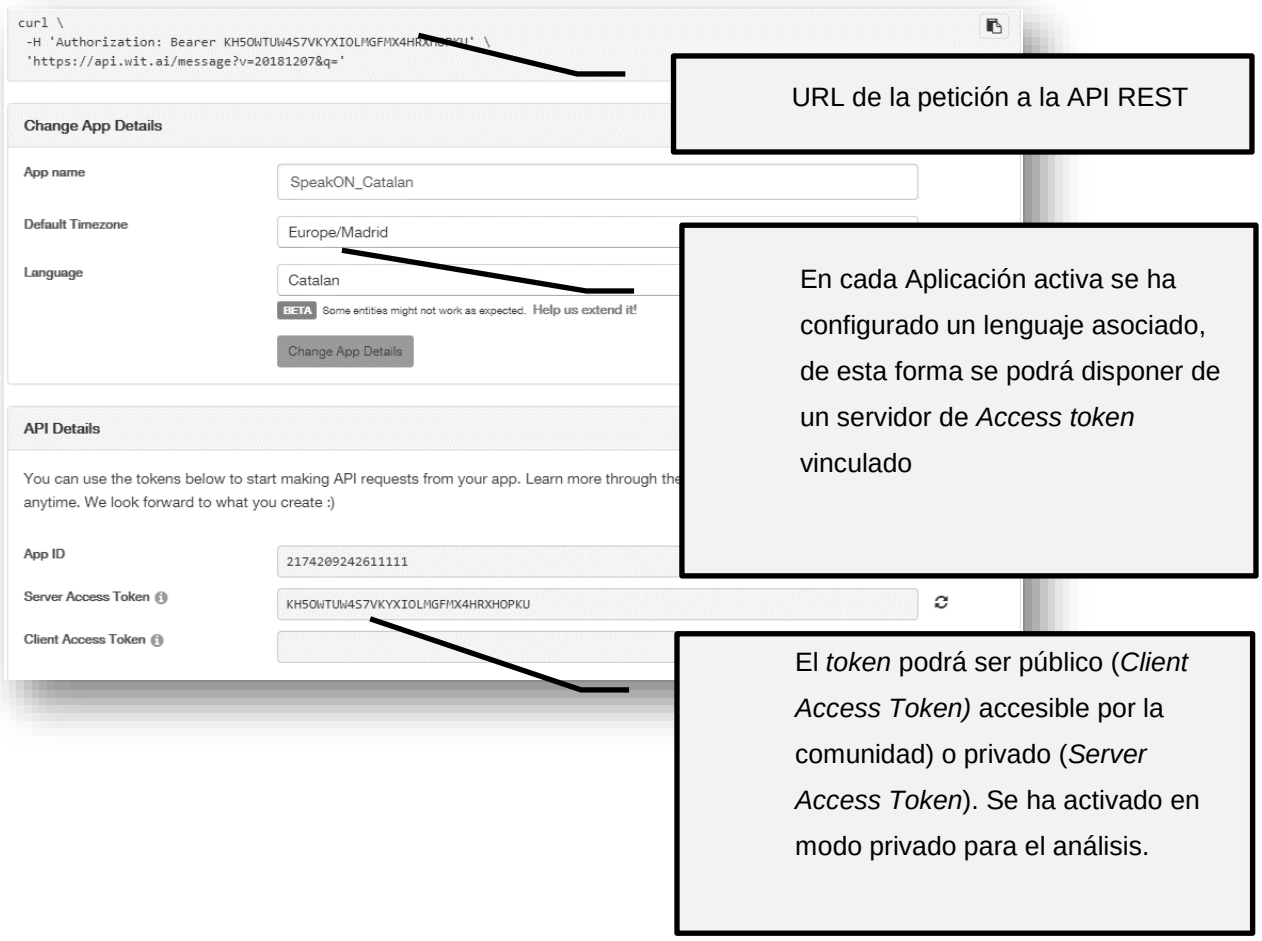

Figura 7:Configuración de una aplicación y lenguaje correspondiente.

Wit.AI.STT. Implantación y test con lenguaje Python

El lenguaje Python no dispone de la sentencia *switch-case*[18], por consiguiente, se ha desarrollado una solución específica de tipo diccionario que retornará el *token* asociado. Para ello, el diseño se ha limitado a los objetivos e idiomas del proyecto: catalán, inglés, francés, italiano, alemán portugués y español. Finalmente, tendrán asociado un significado que será el valor de la clave necesaria en la explotación de las peticiones a la *API REST*. (el *token* creado previamente en *WIT.AI*). El resultado obtenido ha sido correcto.

[18] Estructura de control en los lenguajes de programación sin anidación de escaleras de decisión.

Wit.AI.STT. Función diccionario. Conversión Lenguaje-Token

Código que recibirá un lenguaje (parámetro del formulario principal) y entregará la clave de la cadena personalizada en cada servicio necesaria para la transcripción:

```
def token_speech_uoc(language):
	return {
		"ca" : 'KH5OWTUW4S7VKYXIOLMGFMX4HRXHOPKU',
		"fr" : 'WWWELJGAR5PK4QOM7UPRTNN67VEIKXHR',
		"en" : 'II36VGUP4T2ONZJHN6UAKURDLW5QGZUU',
		"es" : 'DYTZCXVACPL64F56KP44QVYJKQQXZ7NA',
		"it" : 'UALJY7SDLMAUWAZSJLH2HUWWHFKC64WR',
		"pt" : '72BKL3RXGBQRQZGA675HABSQ62WCK3PE',
		"de" : '745JPP7EKFN5S6RNOFBGCDACJJRYLNAS'
	}.get(language)
```

Wit.AI.STT. Implantación y test con lenguaje PHP

Se ha efectuado una llamada *cURL* en *localhost* al fichero PHP incluyendo en la variable $*restAPI* el TOKEN, recuperando, por último, el valor de la posición [“_text”]. Los resultados han sido satisfactorios.

```
<?php
function func_speechtext_uoc ($language){
$restAPI="curl -X POST -H \"Content-type:audio/wav\" -H \"Authorization:Bearer
DYTZCXVACPL64F56KP44QVYJKQQXZ7NA\" --data-binary @/root/records/voice2.wav
\"https://api.wit.ai/speech?v=20181208\"";
$json=shell_exec($restAPI);
//Decoding JSON data to PHP associative array
$obj = json_decode($json,true);
$string=$obj["_text"];
return  $string;
}
$lang="es-ES";
 echo func_speechtext_uoc($lang);
?>
```

Adicionalmente, se han creado dos versiones funcionales en la API: la primera versión utilizará el objeto *request* de Python y la segunda versión la librería WIT disponible en *pip*[19].

[19] Repositorio de paquetes del lenguaje de programación *Python*

Wit.AI. Version 1.

```
def wav_text_wit_uocV1(TOKEN):
	print(" wav_to_text_uoc")
	HEADERS = {'authorization': 'Bearer ' + TOKEN,'Content-Type': 'audio/wav'}
	#Read only mode.
	with open('records/voice.wav', 'rb') as sound:
		data_audio = sound.read()
	#parameters request object
	response=requests.post('https://api.wit.ai/speech?v=20181129',headers=HEADERS,d
ata=data_audio)
	data = json.loads(response.content)
	return data[' text']
```

Wit.AI. Version 2.

```
def wav_text_wit_uocV2(TOKEN):
	text=Wit(TOKEN).speech(open('/www/speakon/records/voice.wav', 'rb').read(),
	headers={'Content-Type': 'audio/wav'})['_text']
	return text
```

Los resultados han demostrado que la versión v2 (librería especifica de Python) es un 15% más veloz que la opción v1 y el elemento *request*. Las dos versiones son perfectamente funcionales, aunque V1 ha entregado errores aleatorios en las respuestas.

```
wit.wit.WitError: Wit responded with status: 400 (Bad Request)
```

Figura 8:Ejemplo de petición errónea en la API de Wit.AI

Wit.AI. Resultados STT.

LATENCIA				
WIT.AI	**PHP**		**PYTHON**	
	Tiempo(s)	5.0	Tiempo(s)	3.0

Tabla 13:Latencia en la muestra.Wit.AI

% ERRORES EN LA TRANSCRIPCIÓN				
WIT.AI	**PHP**		**PYTHON**	
	Errores %	0	Errores %	0

Tabla 14:Errores en la muestra.Wit.AI

Los tiempos de respuesta han sido similares al resto de API analizadas.No se han detectado errores en la transcripción.

7.5 Comparativa y selección final. API STT.

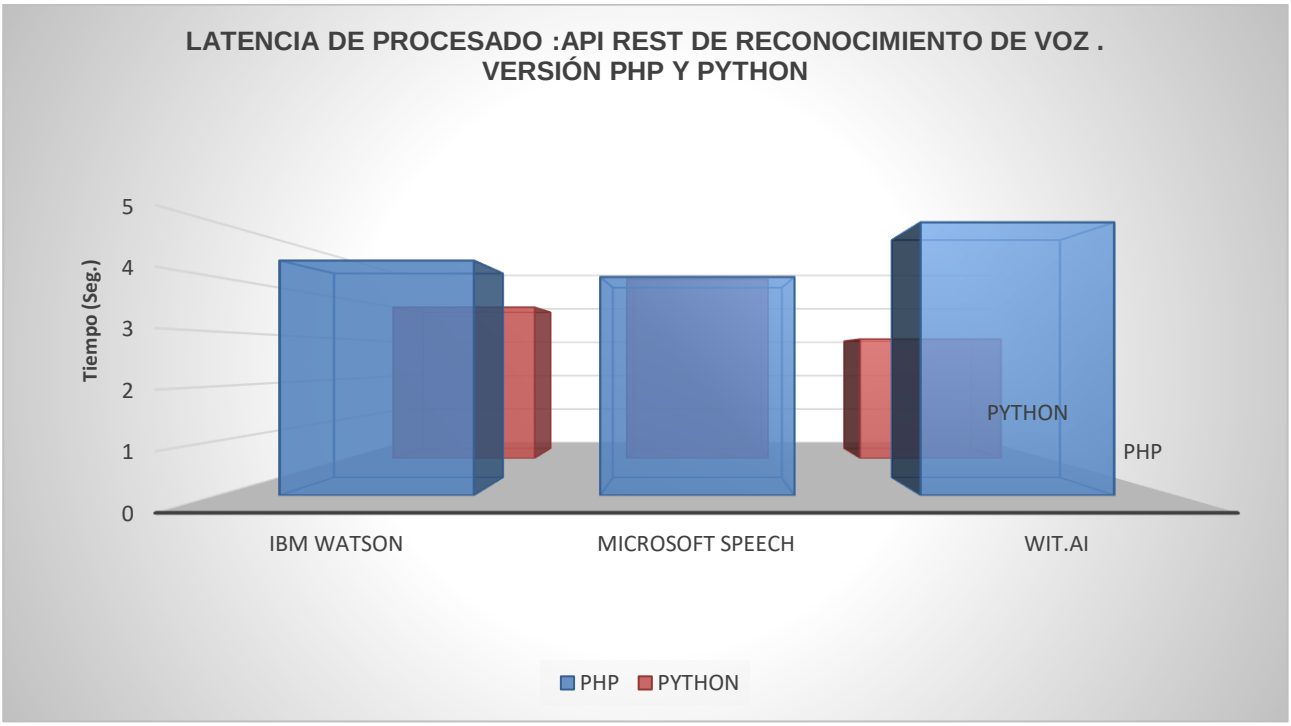

Figura 9:Resultado de la comparativa en el bloque de conversión voz-texto

Una vez analizados los resultados de las diferentes API de reconocimiento de voz se ha determinado trabajar en el proyecto Speak-On con la API wit.ai en versión Python.

Se han considerado los tiempos, la contribución de la comunidad y finalmente el nivel de restricción vocal mensual. Al ser una solución gratuita sin limitaciones, con aproximadamente 60 idiomas diferentes, aportará una experiencia positiva al usuario sin penalizar los tiempos de respuesta del audio procesado ya que en los tres casos han sido muy similares. Por otro lado, la petición HTTP se realizará mediante el algoritmo `wav_text_wit_uocV2(TOKEN)`, este procedimiento ha ofrecido mejores resultados.

8.Bloque de traducción.

Si el usuario ha decidido traducir la conversación, tendrá que activar previamente la opción en el formulario de configuración de la herramienta Speak-On. En consecuencia, el dispositivo trabajará con un nuevo algoritmo funcional: el bloque de traducción.

En este apartado, se han analizado las diferentes opciones para traducir una cadena de texto, se han estimado los resultados y el tiempo de respuesta de cada una de las API.

De igual forma, la calidad de la traducción es un parámetro complicado de cuantificar, el resultado es perfectamente interpretable, en consecuencia, se ha establecido analizar el texto con un margen de error inferior al 10% y se ha cotejado el resultado con el resto de traductores utilizados en el estudio. Finalmente, se ha determinado el sistema más adecuado en el dispositivo Speak-On.
Las API de traducción seleccionadas en el estudio han sido son las siguientes:

API	SERVICIO
Microsoft Translate API	Microsoft Translate
IBM Translate API	IBM Translate.
Google Translate API Python library	GoogleTrans
Yandex API Translate	Yandex

Tabla 15:Servicios de traducción seleccionados en el proyecto.

El siguiente texto ha sido la muestra utilizada en el estudio funcional de las API de traducción:

CONTENIDO DEL TEXTO PARA EL PROCESO DE TRADUCCIÓN:	
Esta grabación corresponde a un análisis de pruebas del trabajo final de grado de multimedia. Se comprobarán los errores detectados en el bloque de traducción de texto.	
Palabras:	**Caracteres sin espacios:**
27	141

Tabla 16:Texto de muestra para el análisis de traducción

Los parámetros serán fundamentales para determinar el nivel de % de error en el resultado. Es importante señalar que se ha utilizado un texto de fácil traducción y se han observado los resultados, simultáneamente, en inglés y francés con dos expertos en estos idiomas.

8.1 Microsoft *Translation Text*. Servicios cognitivos.

URL	https://azure.microsoft.com/es-es/services/cognitive-services/translator-text-api/

Con la ayuda del registro y actividad en la versión gratuita de Microsoft servicios cognitivos, es posible traducir aproximadamente 2 millones de caracteres /mes e incluye traducción de texto, personalización, detección de idioma, diccionario bilingüe y transliteración.

El sistema soporta aproximadamente 60 lenguajes, por lo tanto, permite trabajar de una manera flexible sin limitaciones. (9)

MICROSOFT SPEECH	SERVICIO	LIMITES
Versión FREE	Translator	<2.000.000 caracteres/mes.

Tabla 17:Límites del servicio de traducción de Microsoft Translate

Microsoft Translation Text. Implantación y test con lenguaje Python.

Función adaptada al proyecto. Recibirá los idiomas como parámetros y la cadena de texto y retornará, finalmente, el texto traducido.

```
def translation_microsoft_uoc(text,language_speech,language_personal):
       base_url = 'https://api.cognitive.microsofttranslator.com'
       path = '/translate?api-version=3.0'
       params = '&to='+language_personal+'&to='+language_speech
       constructed_url = base_url + path + params
       apyKey="0fc76450be2f4d30b9ebc2367832623a"
       headers = {
                  'Ocp-Apim-Subscription-Key': apyKey,
                  'Content-type': 'application/json',
                  'X-ClientTraceId': str(uuid.uuid4())
       }
       body = [{
                  'text' : text
       }]
       request = requests.post(constructed_url, headers=headers, json=body)
       response = request.json()
       return (response[0]["translations"][0]["text"])
```

Microsoft Translation Text. Implantación y test con lenguaje PHP.

Algoritmo personalizado y parametrizado extraído de la documentación de Microsoft. Aceptará dos valores:

- Texto a traducir.
- Lenguaje seleccionado en la traducción.

Finalmente, decodificará el fichero *json,* eliminará los espacios en blanco y retornará el texto para su posterior procesado. El código funcional ha sido el siguiente:

```
function func_translate_uoc($text,$language){
             //subscription key.
             $key = '0fc76450be2f4d30b9ebc2367832623a';
             //server and path
             $host = "https://api.cognitive.microsofttranslator.com";
             $path = "/translate?api-version=3.0";
             // Translate
             $params = "&to=".$language;
             //Generates a globally unique identifier (GUID: Globally Unique
Identifier).
             if (!function_exists('com_create_guid')) {
               function com_create_guid() {
                 return sprintf( '%04x%04x-%04x-%04x-%04x-%04x%04x%04x',
                     mt_rand( 0, 0xffff ), mt_rand( 0, 0xffff ), mt_rand( 0, 0xffff )
);}}
/*Translate function build the variable to pass to the url by the post method*/
             function Translate ($host, $path, $key, $params, $content) {
                                $headers = "Content-type: application/json\r\n" .
                                      "Content-length: " . strlen($content) .
"\r\n"."Ocp-Apim-Subscription-Key: $key\r\n" ."X-ClientTraceId: " . com_create_guid()
. "\r\n";
$options = array ('http' => array ('header' => $headers, 'method' => 'POST','content'
=> $content));
$context  = stream_context_create ($options);
$result = file_get_contents ($host . $path . $params, false, $context);
return $result;}
$requestBody = array (array ('Text' => $text),);
$content = json_encode($requestBody);
$result = Translate ($host, $path, $key, $params, $content);
$decode_json= json_decode($result ,true);
//return text translation
$txt=$decode_json[0]["translations"][0]["text"];
trim($txt);
return $txt;}
```

Microsoft Translator Resultados.

LATENCIA				
MICROSOFT	**PHP**		**PYTHON**	
TRANSLATE	Tiempo(s)	3.0	Tiempo(s)	3.0

Tabla 18:Latencia en la muestra. Microsoft Translate

% ERRORES EN LA TRANSCRIPCIÓN				
MICROSOFT TRANSLATE	**PHP**		**PYTHON**	
	Errores %	0	Errores %	0

Tabla 19:Errores en la muestra. Microsoft Translate

8.2 IBM Translate. Servicio de traducción

URL	https://www.ibm.com/watson/services/language-translator/

IBM dispone de un servicio de traducción de texto similar a Microsoft, soporta numerosos lenguajes: árabe, catalán, chino (simplificados y tradicional), checo, danés, inglés, finés, francés, alemán, hindi, italiano, japonés, coreano, polaco, portugués (brasileño), ruso, español, sueco y turco aportando aprendizaje profundo y una mayor precisión. (10)

IBM Translate	**Servicio**	**Límite**
Versión LITE	*Language* Translator-*tq*	<1.000.000 caracteres/mes. Se suprime el servicio tras un mes de inactividad

Tabla 20:Limites del servicio IBM Translator

IBM Translate. Implantación y test con Python.

En Python será imprescindible instalar previamente la siguiente librería:

```
pip install --upgrade "watson-developer-cloud>=2.4.0"
```

A continuación, se configurará la *apikey* [20]activada previamente en el servicio web de IBM, el texto de la muestra y un modelo de lenguaje con una nomenclatura en estructura ISO 639, posteriormente, se ha realizado un volcado de la variable para ver su contenido.

```
from watson_developer_cloud import LanguageTranslator V3
language_translator=LanguageTranslatorV3(version='2018-12-
09',iam_apikey='dIlYFeOWn5ly9-z-zdiyVWZYvdcL7XeqBANQRHD_XHLz')
Translation=language_translator.translate(
text='Hello',
model_id='en_es'.get_result()
print(json.dumps(translation,indent=2,ensure_ascii=False))
```

[20] Identificador proporcionado por el proveedor que permite acceso al servicio especificado.

IBM *TRANSLATE*	**PHP**		**PYTHON**	
	Tiempo(s)	3.0	Tiempo(s)	3.0

Tabla 21:Tabla 22:Latencia en la muestra. IBM Translator

% ERRORES EN LA TRANSCRIPCIÓN				
IBM TRANSLATE	**PHP**		**PYTHON**	
	Errores %	5.6	Errores %	5.6

Tabla 22: Errores en la muestra. IBM Translator

Se ha detectado un error al traducir en el idioma inglés de la palabra **análisis.** Los tiempos han sido similares en los dos tipos de petición.

8.3 Googletrans. Servicio de traducción.

URL	https://github.com/ssut/py-googletrans

Googletrans es una biblioteca de Python gratuita e ilimitada que gestiona el uso de llamadas Ajax en la web de Google. Permite realizar las llamadas a métodos como detectar y traducir.
Asimismo, dispone de un sistema de identificación automática de idioma, soporta HTTP apoyado íntegramente en translate.google.com, de igual forma, no tiene impacto económico.

Googletrans	**Servicio**	**Límite**
Versión FREE	*Translator*	El límite máximo de caracteres en un solo texto es de 15kbytes, por tanto, serán aproximadamente 15360 caracteres (por petición) no existe limites en el número de solicitudes.

Tabla 23:Límites en el servicio GoogleTrans

Googletrans. Implantación y test con Python

Ha sido fundamental instalar previamente, la librería recuperada en el repositorio `PIP (11)`:
`pip install googletrans`
Para realizar el test de la muestra, se ha creado una función para que recibe dos parámetros:

A) Texto a traducir.
B) Lenguaje a seleccionar para la traducción.

La función definida tiene configurada por defecto la identificación automática del lenguaje de origen , por lo tanto ,no ha sido imprescindible enviar el código de idioma.

```
def translate(text,language):
      translator= Translator()
      value=language
      return translator.translate(text,dest=value).text
```

Resultados* Googletrans*. Traducción

LATENCIA				
GoogleTrans	**PHP**		**PYTHON**	
	Tiempo(s)	2.0	Tiempo(s)	2.0

Tabla 24:Latencia en la muestra. GoogleTrans

% ERRORES EN LA TRANSCRIPCIÓN				
GoogleTrans	**PHP**		**PYTHON**	
	Errores %	20	Errores %	20

Tabla 25:Errores en la muestra. GoogleTrans

Inicialmente los resultados han tenido una latencia muy baja, aunque es importante reflejar errores aleatorios en su ejecución (3/15 ejecuciones) no relacionados con el contenido de la variable que ha recibido.

8.4 Yandex Translate. Servicio de traducción.

URL	https://translate.yandex.com/

Yandex (abreviatura de *yet another indexer*) es el motor de búsqueda más utilizado en Rusia.
Uno de sus principales servicios disponibles es una herramienta denominada *Translate* con un sistema de aprendizaje automático. Los desarrolladores tienen la posibilidad de incluir la API en diferentes entornos. En concreto, se ha optado por desarrollar una solución sobre Python.
La traducción está actualmente disponible en más de 90 idiomas: Azerbaiyán, albanés, amárico, inglés, árabe, armenio, afrikáans, vasco, bielorruso, bengalí, birmano, búlgaro, bosnio, galés, húngaro, vietnamita, haitiano, gallego, holandés, Hill Mari, griego, georgiano, gujarati danés, hebreo, indonesio, irlandés, italiano, islandés, español, canarés, catalán, kirguiso, chino, coreano, xhosa, jemer, laosiano, latín, letón, lituano, luxemburgués, malgache, malayo, etc. (12)

Yandex	Servicio	Límites	Costes
API gratuito	*YANDEX TRANSLATE*	1.000.000 caracteres por día, máx. <10.000.000 /mes	0
API Pago	*YANDEX TRANSLATE*	10.000.000 caracteres por día, <50.000.000 /mes	15$ / millón de caracteres.

Tabla 26:Límites en el servicio *Yandex Translate*

Yandex Translate. Implantación y test con lenguaje Python

Ha sido fundamental descargar una librería en el repositorio de Python para trabajar con la API de una manera más eficaz e intuitiva: `pip install yandex-translater`. Además, será imprescindible activar una clave para ejecutar las llamadas a la API. Una vez registrado en la web se activará el identificador necesario utilizado en el test de muestra.

Función diseñada para el proyecto Speak-On:

```
def translation_yandex_uoc(text,language_speech,language_personal):
	tr = Translater()
	tr.set_key('trnsl.1.1.20181026T095357Z.237f66eb2b5589a7.fbac66712861c161c270f3b
532b28b2e0ce5b5ea')
	tr.set_text(text)
	tr.set_from_lang(language_speech)
	tr.set_to_lang(language_personal)
	return tr.translate()
```

Resultados Yandex. Traducción

LATENCIA				
Yandex	**PHP**		**PYTHON**	
	Tiempo(s)	3.0	Tiempo(s)	2.7

Tabla 27:Latencia en la muestra: Yandex Translate

% ERRORES EN LA TRANSCRIPCIÓN				
GoogleTrans	**PHP**		**PYTHON**	
	Errores %	0	Errores %	0

Tabla 28:Errores en la muestra: Yandex Translate

Se han obtenido resultados óptimos en la versión Python. No se han detectado errores en las respuestas.

8.5 Comparativa y selección final de la Api de traducción

Una vez analizadas las diferentes propuestas a nivel de traducción de cadenas de texto, se ha descartado el uso de GoogleTrans. Los tiempos de respuesta de esta solución han sido muy reducidos y su implantación sencilla, pero se han detectado errores y bloqueos debido en gran parte a que Google tiene un servicio de pago en la nube similar, asimismo, la API de manera aleatoria no ha funcionado correctamente al lanzar peticiones en cadena vía AJAX. Adicionalmente, se ha valorado no incorporar en un dispositivo comercial una librería de estas características. En el caso de Yandex Translate los límites de caracteres son muy amplios en la versión básica (10.000.000 millones), su flexibilidad al trabajar con múltiples idiomas y los resultados en los test han sido determinantes.
Por ello, Yandex en Python ha sido la opción seleccionada en el bloque de traducción. Las versiones de Microsoft e IBM Watson han mostrado limitaciones en caracteres traducidos que pueden afectar negativamente a un uso elevado de la plataforma.

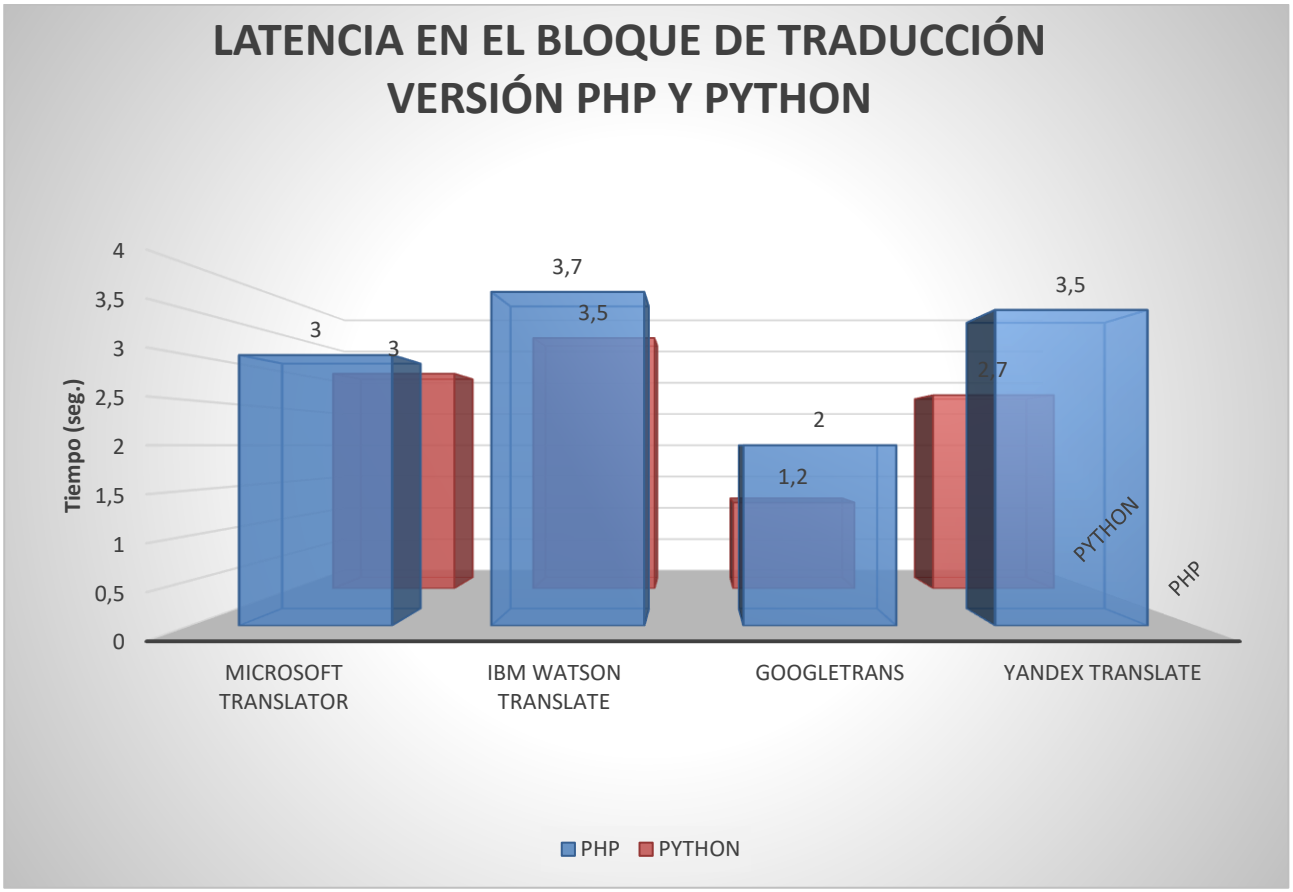

Figura 10:Resultado de la comparativa en el bloque de traducción

9.Bloque de síntesis de voz. TTS

Este bloque ha sido el responsable de realizar la conversión de una cadena de texto a un fichero de audio WAV con una configuración previa de determinados parámetros.

Para realizar la conversión a voz existe la posibilidad de utilizar dos vías distintas:

- una cadena de texto formateada.
- Lenguaje SSML '*Speech Synthesis Markup Language*' (Lenguaje de etiquetado de síntesis del habla),Es un lenguaje propuesto por la W3C[21] ,con el propósito de realizar una síntesis de voz de una forma más natural . (13)

Asimismo, integrar la síntesis de voz en el dispositivo Speak-On, gestionado con un sistema Linux reducido Arduino y unas características técnicas limitadas, requiere un proceso de adaptación. En consecuencia, se ha decidido trabajar con una cadena de texto estandarizada, debido en gran parte a la complejidad de adecuar la investigación al lenguaje SSML. (14) Finalmente, se ha valorado el retardo en el procesamiento, calidad de la voz, costes y el soporte e información del producto.

Las soluciones analizadas ha sido las siguientes:

- Microsoft TTS Cognitive
- CereVoice
- Acapela
- Svox
- Espeak
- Responsive Voice
- Festival

Para efectuar un examen crítico de la síntesis de voz (15), se ha utilizado un texto de muestra en inglés. Este idioma ha estado presente en todas las soluciones descritas con anterioridad. El contenido del texto ha sido el siguiente:

CONTENIDO DEL TEXTO PARA EL PROCESO DE SÍNTESIS DEL HABLA:		
Text to perform the speech synthesis process. Speech synthesis is the artificial production of human speech		
CARACTERES	**PALABRAS**	**% MARGEN ERROR**
107	*16*	*10%(caracteres)*

Tabla 29:Contenido textual de la muestra .Síntesis de voz

[21] '*World Wide Web Consortium*'(Consorcio Mundial de la red) es una comunidad internacional, con el propósito de elaborar directrices operativas en los estándares web.

9.1 Test comparativo de síntesis del habla.

SOLUCIÓN	LATENCIA DE CONVERSION A FICHERO WAV	%ERRORES
Microsoft TTS Cognitive	2 seg.	0
CereVoice (16)	5 seg.	12%
Acapela (17)	3 seg.	0
Svox	2 seg.	0
Espeak	3 seg.	0
Responsive Voice	5 seg.	28%
Festival (18)	3 seg.	0

Tabla 30:Resumen de latencia y errores en el procesado de la muestra. Síntesis de voz.

9.2 Cuadro comparativo de características y observaciones TTS.

SOLUCIÓN	SOPORTE DE LENGUAJES	LÍMITE	OBSERVACIONES
Microsoft Speech	Completo.	<5.000.000 millones caracteres gratis	Nivel Alto en soporte e información. Lenguaje natural.
CereVoice	Parcial. No trabaja con el idioma italiano.	<1.000.000 caracteres/mes 140€	No dispone de Api. Errores detectados
Acapela	Completo.	6€ /min.	Costes elevados. Nivel alto en soporte técnico .
Svox	Parcial. No trabaja con catalán y portugués	No.	Integración en sistemas Linux.
Espeak (19)	Completo.	No.	Sonido artificial, Nivel Bajo, en síntesis.
Responsivevoice (20)	parcial. No trabaja con el idioma catalán.	Si.	Conflictos al incorporar la librería en Python. Nivel Alto en soporte. Errores detectados
Festival	Parcial. Inglés y español.	No.	Elevado consumo de recursos .

Tabla 31:Comparativa funcional de las de conversión texto a voz

9.3 Selección final en el proceso de conversión texto a voz.

Una vez analizadas las distintas soluciones se ha optado por trabajar con la siguiente configuración:

- En alemán, francés, italiano, español, inglés se trabajará con SVOX (21) es una solución que ofrece una calidad aceptable sin costes y de fácil integración, es importante señalar que este módulo no dispone de cobertura para el idioma catalán y portugués, en consecuencia, se ha optado por una solución paralela.
- Catalán y portugués: para estos dos idiomas se ha decidido utilizar la API de Microsoft TTS (22). El fabricante norteamericano ha mostrado resultados óptimos y amplio margen en la versión básica.

Función discriminatoria de la aplicación TTS

Se ha desarrollado una función específica que seleccionará la API vinculada al lenguaje previamente almacenado en la aplicación:

```
def text_voice_uoc (text,language):
	#processing with Microsoft Api Translate
	if language=='ca-ES' or language=='pt-PT':
		#Catalan-Portuguese ,api resolution
		voice_cata_portu_uoc(text,language)

	else:
		#en-US, en-GB, de-DE, es-ES, fr-FR and it-IT.
		p=subprocess.Popen(["pico2wave", "--wave",
"/www/speakon/records/voice_translate.wav",  "-l", language,  text], shell=False)
	p.wait()
```

10. Soluciones implicadas en el proyecto

VOZ A TEXTO (STT)	TRADUCCIÓN	TTS(TEXTO A VOZ)
Wit.Ai	Yandex Translate	Svox / Microsoft Speech Cognitive services

Tabla 32:Soluciones aplicadas en el proyecto Speak-On

11. Equipamiento electrónico.

En la definición del proyecto Speak-On, se ha decidido utilizar un pulsador para la gestión de inicio de grabación del audio y solicitud de transcripción, debido en gran parte, a la accesibilidad que proporciona efectuar todas las tareas desde un único punto, por lo tanto, la interfaz hombre-máquina (IHM) [22] responsable de interpretar las pulsaciones y presentar el contenido en la pantalla *TFT* de 1.44" será una placa electrónica de Arduino. Adicionalmente, se responsabilizará de las llamadas a las API de traducción, síntesis de voz, algoritmos de grabación y entorno conectivo.

En consecuencia, será fundamental disponer de dos elementos diferenciados: por un lado, un sistema operativo conectado a la red de internet, compatible con el lenguaje Python que facilite la integración de los bloques analizados y, por otro lado, una interfaz física que permita la presentación en pantalla y explotación del pulsador mediante una comunicación bidireccional con el módulo Python (Arduino UNO).

Se ha efectuado un estudio de los diferentes dispositivos existentes en el mercado basados en Arduino, con el propósito de, evitar sobrecostes, unificar la solución y optimizar los resultados. Finalmente, se han seleccionado dos placas de procesamiento que permitirán ejecutar el proyecto Speak-On en el mismo dispositivo:

- Arduino YUN rev2 (23)
- Onion Omega2 Plus+ Dock Arduino. (24)

11.1 Comparativa de dispositivos

Dispositivos preseleccionados ,con el objeto de, efectuar las acciones del proyecto:

CARACTERÍSTICAS	OMEGA2 ARDUINO DOCK	ARDUINO YUN REV2
PROCESADOR	ARM 580 MHz	Atheros AR9331,400MHZ
ALIMENTACIÓN	3.3V	3.3V
MEMORIA RAM	128 MB DDR2	64 MB DDR2
MEMORIA FLASH	64MB	32 MB
CONEXIONES ACCESO INTERNET	WiFi: IEEE 802.11b/g/n	IEEE 802.3 10/100Mbit/s WiFi: IEEE 802.11b/g/n
PUERTOS/EXPANSIÓN	USB Type-A: 2.0 Host/Device, Lector de tarjetas: Micro-SD ,GPI,PWM	USB Type-A: 2.0 Host/Device, Lector de tarjetas: Micro-SD
COSTES	31€	67,33€

[22] Elemento que conecta los bloques de procesamiento con el usuario de la herramienta Speak-on.

Tabla 33:Comparativa de dispositivos electrónicos base.

Como se ha podido determinar en el cuadro anterior, las características que ofrece el sistema embebido del dispositivo OMEGA2+ combinado con Arduino, ofrece mejores prestaciones con un coste potencialmente inferior. En concreto, Arduino YUN ofrece un incremento aproximado del 216% ,respecto al valor del modelo del fabricante Onion.

En el apartado de las dimensiones y posibilidades de expansión las dos placas son muy similares: la disposición de pines en el *dock* de Omega2 es idéntica a un Arduino UNO en todos los aspectos a excepción del zócalo que incorpora para el chip de Onion y los diferentes puertos GPIO '*General Purpose Input/Output*' (Entrada/Salida de Propósito General).

Por consiguiente, debido al rendimiento superior, costes y flexibilidad operativa, la elección del dispositivo electrónico para la ejecución del proyecto ha sido el modelo: Onion Omega2+.

Por otro lado, la comunicación entre los dos componentes (chip Omega+) y la placa base de Arduino Dock se gestionará íntegramente mediante el puerto serie de ambos. La conexión UART '*Universal Asynchronous Receiver-Transmitter*' (Transmisor-Receptor Asíncrono Universal) se utilizará para proporcionar comunicación bidireccional entre el Omega y el ATmega MCU *'microcontroller unit'* (unidad de microcontrolador). El puerto serie de ATmega estará conectado al puerto serie UART de *Omega*.

11.2 Equivalencia de conexiones: *Omega* y el Microcontrolador Atmega:

OMEGA PIN	ATMEGA PIN
UART1	Serial PINS
I2C	I2C
GPIO 15	SPI SCK
GPIO 16	SPI MOSI
GPIO 17	SPI MISO
GPIO 19	Reset

Tabla 34:Relación de conexiones Omega y Atmega 328

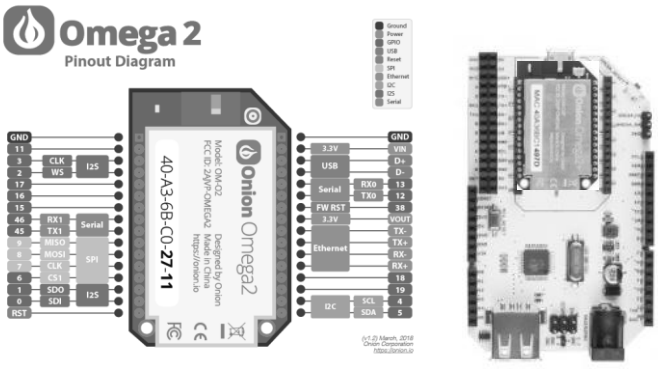

Figura 11:Diagrama de relación de pines Arduino Omega 2+. (25)

11.3 Componentes electrónicos.

En el diseño del prototipo se han seleccionado una serie de elementos electrónicos donde ha destacado fundamentalmente el apartado del procesamiento de audio en sus dos principales roles: el registro de grabación y su posterior reproducción de ficheros del audio generado.

Por lo tanto, se ha determinado trabajar con una tarjeta de sonido USB' *Universal Serial Bus'* (bus serie universal) y una configuración personalizada de 16bits, mono con una tasa de muestreo de 44.100Hz (en un segundo se tomarán 44.100 muestras de la señal analógica de audio) ,con el objetivo de elaborar el proceso de digitalización.

Asimismo, el empleo de un micrófono de alta sensibilidad, con la ayuda de una grabación de sonido de alta calidad ofrecerá un resultado óptimo en el proceso de transcripción, bloque fundamental en el desarrollo del proyecto.

Relación de componentes.

LISTADO DE COMPONENTES:	
KY016	Módulo de Diodos LED RGB. Cátodo Común Vcc 5v.
TFT ILI9163C	Pantalla TFT 128x128 píxeles Interfaz SPI de 4 hilos (A0, SDA, SCK, CS).Tamaño: 1.44 pulgadas
KY004	Modulo interruptor de botón táctil FZ1713 y resistencia de protección. Vcc 5v.
SOUNDCARD USB AQPROX	Tarjeta de Sonido USB , GL3520 + CMI119B chipset , Jack 3.5 mm Audio (output) y micrófono (input)
NGS MS110	Micrófono , 20 - 16 KHz Sensibilidad -62 dB ±3 dB
SH-019	Altavoz de 2W/35 ohmios, diámetro de 41mm.
ARDUINO DOCK R2	DockStation Arduino Uno para Omega2+. Microcontrolador: ATmega328.Vcc: 5v,Pines de Entradas/Salidas Digital: 14 Pines de Entradas Análogas: 6.Memoria Flash: 32 KB (ATmega328)
PROCESADOR OMEGA 2+	Procesador: ARM 580 MHz, RAM: 128 Mb, Flash: 64 Mb, Conectividad: Wifi 802.11 b/g/n,15 GPIO,2 PWM,2 UART,1 I2C,1 SPI,1 I2S,Zócalo para microSD

Tabla 35:Listado de componentes electrónicos del proyecto.

12. Diseño del circuito

12.1 Esquema funcional.

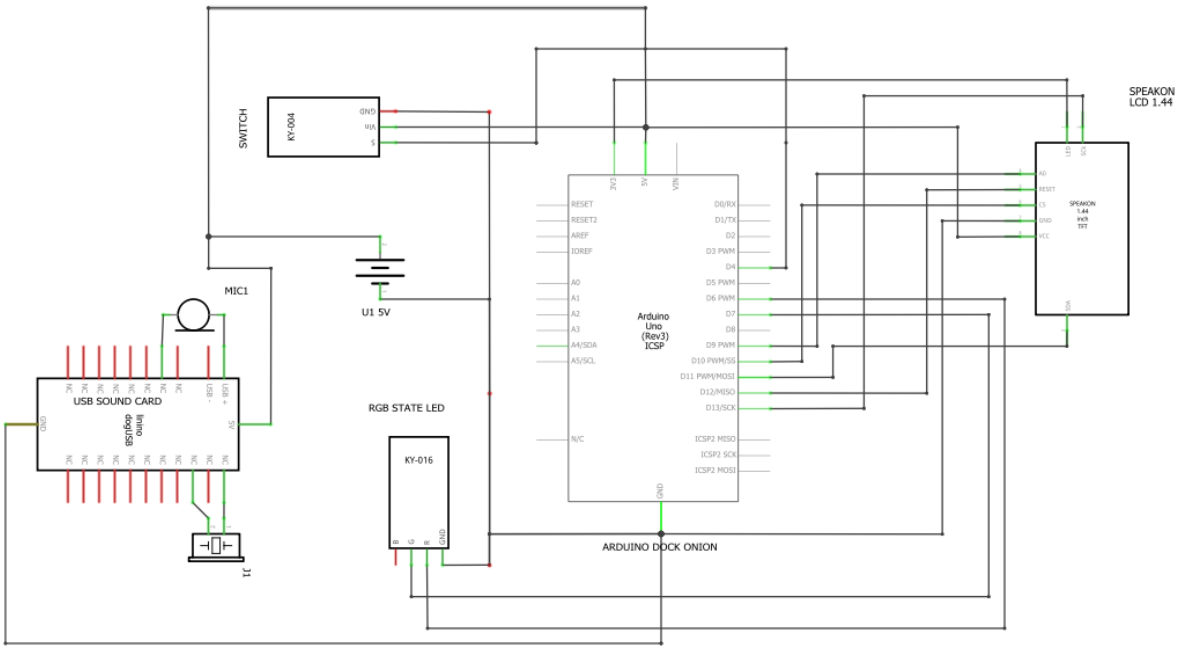

Figura 12:Esquema funcional del circuito electrónico.

12.2 Conexionado en Protoboard.

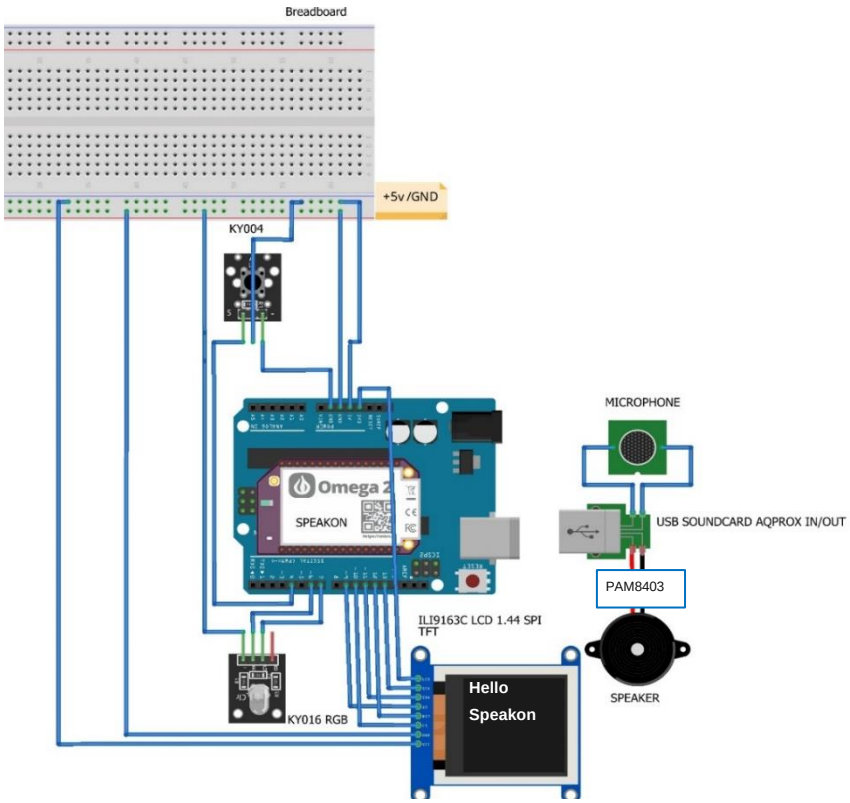

Figura 13:Conexionado de componentes en placa de pruebas.

12.3 Diagrama conectivo de bloques.

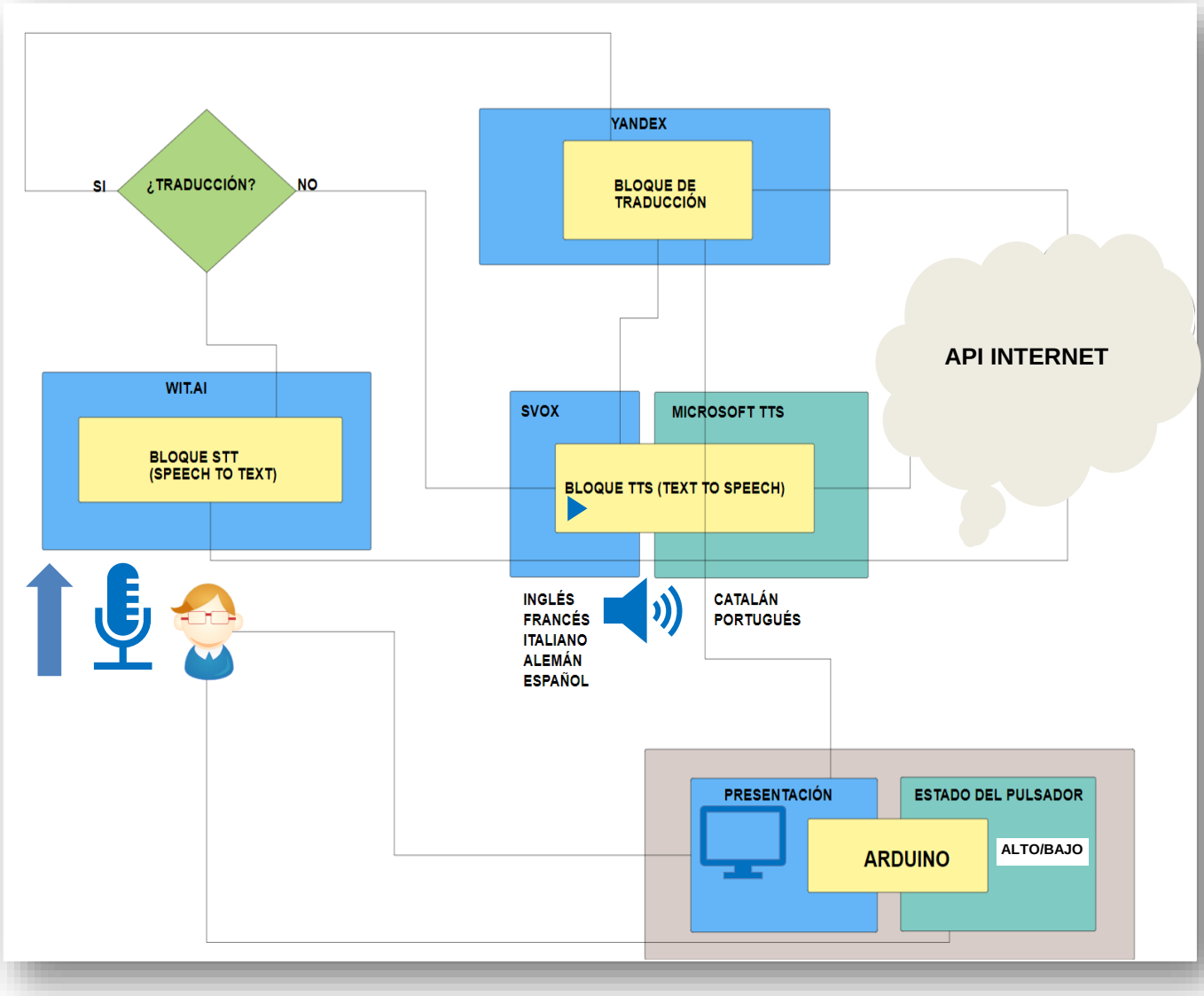

Figura 14:Diagrama de bloques funcionales.

13. Funcionamiento.

Una vez se ha ajustado e iniciado el dispositivo Speak-On, se accederá a la web de configuración donde se seleccionarán los lenguajes y la posibilidad de traducir el contenido (este parámetro será optativo). Se mostrará un mensaje con la confirmación de la escritura de la configuración en el sistema. El equipo con Arduino estará preparado para procesar el audio del entorno registrado por el micrófono direccional. En el *display* se observará el texto *Speakon ready* y se escuchará una locución con el mismo contenido. Seguidamente, si el pulsador es presionado, el diodo LED *'light-emitting diode*' (diodo emisor de luz) D1 ubicado en la caja de conexiones cambiará a color verde(*HIGH*), el proceso de grabación se ejecutará y continuará en este estado hasta que el botón SW1 se active de nuevo. Durante el proceso de grabación la tarjeta de sonido permanecerá registrando el audio a 16bits / 44Khz en formato *WAV* en el subdirectorio `/www/Speakon/records`. Si la grabación se ha detenido, se iniciará el siguiente procedimiento encadenado:

- La grabación de audio se procesará y se convertirá en texto.
- Se realizará la traducción de texto (en caso de que sea solicitado).
- Se presentará el resultado a nivel audiovisual: en el *display* se proyectará el texto traducido /procesado y se sintetizará la voz con el contenido del texto en el lenguaje definido.

Los diversos estados del dispositivo serán transmitidos a nivel audiovisual en la placa Arduino, con el principal objetivo de, generar accesibilidad cognitiva. El usuario percibirá la imagen reflejada en los cristales y el sonido será presentado en su montura.

13.3 Video de test inicial de comprobación funcional.

Test multimedia de bloque operativo.	**https://youtu.be/8kO9l-XEoCw**

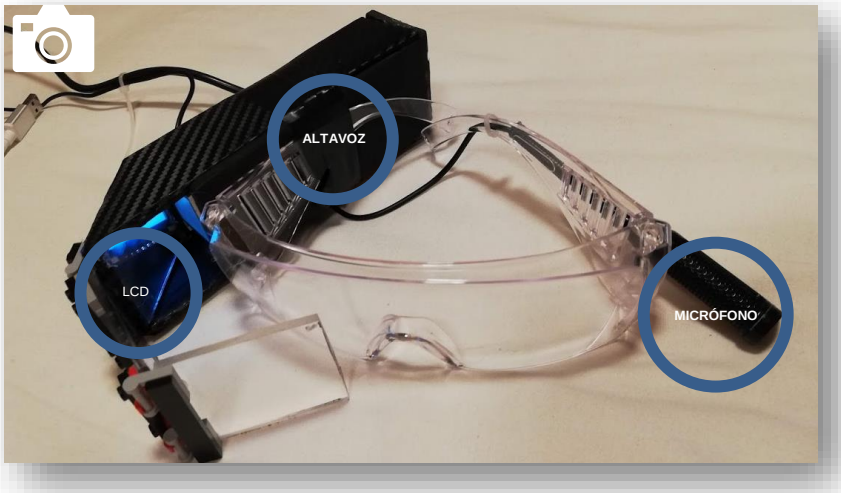

Figura 15:Imagen del prototipo Speak-On V1.1

13.2 Estructura de ficheros.

DIRECTORIO	DESCRIPCIÓN
/www/Speakon/functions	Almacenamiento de funciones PHP de lectura y configuración de la herramienta.
/www/Speakon/options	Almacenamiento fichero config.txt.
/www/Speakon/Python_Speakon	Almacenamiento de los distintos algoritmos desarrollados en Python de Speak-On.
/www/Speakon/records	Almacenamiento de las grabaciones de audio.
/www/Speakon /js	Librerías JavaScript.
/www/Speakon/Index.html	Fichero HTML de la web principal.
/www/Speakon/Speakon.py	Script secuencial que englobará todas las llamadas.

Tabla 36:Árbol de directorios y estructura de ficheros.

13.3 Algoritmo principal.

El sistema Linux se iniciará y la placa Arduino cargará el *sketch* [23] ,mediante la inserción en el fichero `/etc/rc.local` de la ejecución principal del programa se armará toda la estructura de funcionamiento:

```
# LOAD FILE FOR SPEAK-ON
/usr/bin/python3 /www/speakon/speakon.py
exit 0
```

Una vez cargado el script de Python en el *display* aparecerá*: Speakon Ready!*, por lo tanto, el *software* se habrá iniciado correctamente en Omega+2 y la comunicación UART con Arduino se habrá activado, además, se presentará una locución *(Speakon ready*). El Led de estado D1 permanecerá en rojo.

```
write_serial_uoc('#  Speakon\n   ready!')
play_started_uoc ()
```

Figura 16:Imagen del sistema speak-on inicializado.

[23] Denominación del programar cargado en el microcontrolador de la placa Arduino.

El usuario accederá vía WIFI a un portal de configuración del dispositivo en red local. El diseño web será responsivo basado en el *framework* Bootstrap [24]y se adaptará a tabletas, móviles y PC.
http://Speakon.local/Speakon/

En el portal aparecerá un menú de configuración de Speak-On. Por un lado, se configurará el lenguaje del habla que registrará la aplicación, por otro lado, un selector confirmará si se desea realizar una traducción, de esta manera si no se activa, el bloque de traducción no se ejecutará mejorando el tiempo de respuesta de la herramienta. En caso contrario, se desplegará un menú con el lenguaje final de la traducción.

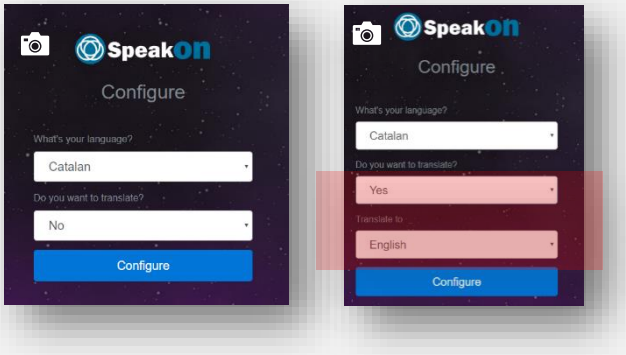

Figura 17:selección de idiomas y configuración inicial.

Las opciones se registrarán utilizando una conversión a objeto `JSON` del contenido de los distintos selectores realizando una petición asíncrona con el método `$.Ajax` de JQuery[25] a un fichero PHP (`func_write_configuration_uoc.php`) que será el encargado de almacenar en un archivo de texto (Speakon/options/config.txt) los valores de configuración del dispositivo en tres filas independientes: Ejemplo del contenido del fichero de configuración `config.txt`:

```
Ca => Lenguaje de la conversación (FILA 0)
En => Lenguaje de Destino para la traducción (FILA 1)
1  => Opción de traducción booleana Y/N (FILA 2)
```

La lectura en Python se realizará con una función que gestionará la apertura del fichero de texto y recogerá el contenido del *array* (`parameters_Speakon_uoc.py`) recibiendo como parámetro la posición de la fila a procesar.
El script revisará el puerto serie cada 1,5 segundos y en el caso de que el botón SW1 se active, el led D1 cambiará de estado y se ejecutará un procedimiento determinado.

[24] Librería para el diseño web responsivo basada en un sistema de rejillas adaptables.
[25] Librería operativa en JavaScript creada, con la finalidad de, optimizar y mejorar la efectividad en el desarrollo web.

En concreto, si **D1** cambia a color verde, Arduino habrá enviado un cambio de estado (`HIGH`), Python recogerá este parámetro lo convertirá a un valor booleano `True` y ejecutará secuencialmente en el bucle `While` lo siguiente:

- Aviso acústico de inicio de grabación de audio (Para personas con déficit visual).

```
play_recorder_uoc ()
```

- Aviso visual de inicio de grabación de audio (Para personas con déficit auditivo).

```
write_serial_uoc('%Recording\n Voice')
```

Es importante señalar que el primer carácter enviado a través del puerto serie será "% "debido a que Arduino discriminará el contenido de la cadena formateando en el display cada estado según el valor recibido en la primera posición:

```
'#'= Ready, '%'= Recording, '$' = Processing,'*' = Clear,'-' = Error.
```

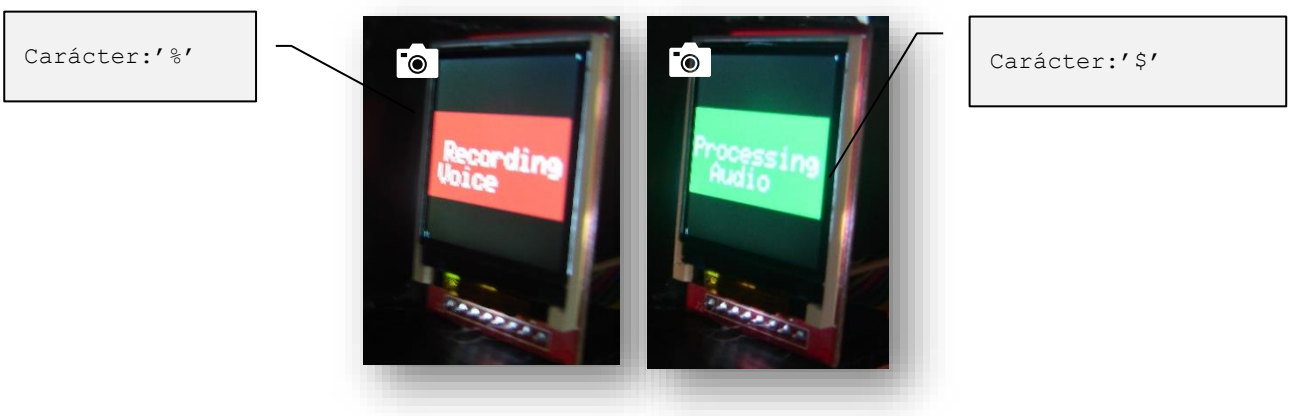

Figura 18:Imagenes del estado del algoritmo principal.

- Inicio de grabación.

```
recording_voice_uoc ()
```

En caso contrario, (el botón SW1 se ha presionado), el estado será 0 (*LOW*) y el led L1 se iluminará en color rojo (se ha ejecutado la orden de transcripción y presentación del contenido). En consecuencia, el flujo de procesamiento será el siguiente:

- Aviso visual de estado procesando audio.

```
write serial uoc('$Processing\nAudio')
```

El símbolo '$' será de nuevo procesado en Arduino

- Aviso acústico del estado en proceso.

```
play_processing_uoc ()
```

- Detención del sistema de grabación.

```
stop_recording_voice_uoc ()
```

- Lectura de variables de lenguajes y opción de traducción.

```
language_speech=read_configuration_uoc(0)
language_personal=read_configuration_uoc(1)
option_translate=convert_string_boolean_uoc (read_configuration_uoc(2))
```

- Selección de *token* con Wit.AI (Idioma del habla).

```
token_use=token_speech_uoc(language_speech)
```

- Algoritmo de conversión del habla a texto.

```
txt_speech_wit=wav_text_wit_uocV2(token_use
```

- Si se decide traducir se recogerá en una variable el texto traducido utilizando Yandex.

```
if option_translate:
txt_translate=translation_yandex_uoc(txt_speech_wit,language_speech,language_pe
rsonal)
```

- También será fundamental adaptar la nomenclatura de los lenguajes con el fin de unificar la aplicación en SVOX/Microsoft/Yandex. En el proyecto se ha trabajado con dos representaciones diferentes en el estándar ISO 639[26].

```
voice_language=convert_type(language_personal)
```

- Si el usuario ha decidido no traducir el idioma, el texto utilizado será el original:

```
#voice without changes
voice_language=convert_type(language_speech)
#no translate text, the text=speech to text
txt_translate=txt_speech_wit
```

- A continuación, se generará el fichero WAV con la síntesis de voz para su ejecución posterior:

```
text_voice_uoc(txt_translate,voice_language)
play_voice_uoc ()
```

[26] Nomenclatura oficial empleada para definir el código regional utilizado en programación vinculado a los diversos idiomas.

En este punto se han utilizado dos funciones distintas que llamarán a SVOX o Microsoft TTS (catalán /portugués). Se adjuntará el código en el Anexo 2.

- Es importante señalar que ocurrirá si se ha almacenado una conversación que superará en el número de caracteres el máximo del *display*. La pantalla utilizada, solo admite 315 caracteres en tamaño (SZ1), por lo tanto, será necesario seccionar el texto para presentar el contenido en varios bloques. Inicialmente, se contabilizarán los caracteres con la ayuda de una función específica:

```
# For Limits the Display LCD 1.44"
nchar=count_uoc(txt_translate)
```

Si la cadena contabilizada ha superado el número máximo de caracteres en la primera presentación, se ejecutará una función que recibirá un multiplicador para dibujar un bucle ,es decir, se mostrará del carácter [0]-[314], esperará 7 segundos (tiempo suficiente para la lectura) y presentará de nuevo el siguiente bloque seccionado: [314]-[xxx] y así sucesivamente hasta el último carácter, en caso contrario ,el contenido será presentado en la primera presentación ,de modo que, se escribirá directamente en el puerto serie para que Arduino lo muestre en la pantalla TFT.

```
if (nchar>315):
       multipl=ceil(nchar / 315)
       split_for_display_uoc(txt_translate,multipl)
else:
       #write text in the serial normalized
       write_serial_uoc(txt_translate)
```

- Función de seccionado de cadena de texto y presentación en el puerto serie.

```
def split_for_display_uoc(text,n):
       for x in range(0, n):
              #clear display
              write_serial_uoc('*')
              #calc position
              ch_start=315*x
              ch_end=ch_start+315
              #split string
              txtprocess = text[ch_start: ch_end]
              #print display
              write_serial_uoc(txtprocess)
              #time for read  text in display 7 seconds
              time.sleep(7)
```

Finalmente, la función de escritura en el puerto serie `write_serial_uoc` solo contemplará los caracteres ASCII *'American Standard Code for Information Interchange*' (Código Estándar Americano para Intercambio de Información) que permitan ser dibujados, por lo tanto, normalizará el texto y eliminará aquellos que no sea posible representar en el *display* controlado por Arduino. (ANEXO 2).

13.4 Arduino.

En el bucle principal de Arduino, se analizará el estado del botón SW1 (alto/bajo) en cada ciclo. Si el botón ha sido activado, el valor de la variable booleana `STATUS` cambiará respecto a su valor anterior. El led RGB *'Red, Green, Blue'* (rojo, verde, azul) se vinculará al contenido de la variable, con el objetivo de, iluminar un color determinado. A continuación, se enviará por puerto serie el digito binario, finalmente, el valor será leído en Python y se ejecutará el *script*[27] principal.

Ahora bien, si un pulsador es presionado en un circuito electrónico digital, será complejo efectuar el cambio de estado de forma natural, debido en gran parte al ruido eléctrico que aparecerá en los flancos[28] de la señal. Estos rebotes generados por ruido (afectaciones externas, mecánicas) podrían provocar falsas pulsaciones que afectarán al sistema y su ejecución.

Para solventar este punto, se ha utilizado una librería específica para Arduino (`BOUNCE2`) que reajustará el tiempo de lectura del estado del pulsador, evitará falsas peticiones al disponer de margen de lectura suficiente.

De forma paralela, la gestión y explotación del TFT se realizará utilizando las librerías del fabricante `TFT_ILI9163C` y `Adafruit_GFX`.

```
#include <Bounce2.h>
#include <SPI.h>
#include <TFT_ILI9163C.h>
#include <Adafruit_GFX.h>
```

Por último, Arduino estará pendiente de la recepción de contenido en el puerto serie. Si se envía información (en este caso, una cadena de texto) leerá la cadena hasta el símbolo "*" `Serial.readStringUntil('*')` y escribirá en el display el valor correspondiente (discriminación del primer carácter).

[27] Composición en un lenguaje de programación que determinará un procedimiento específico.
[28] Los flancos en una señal digital, son la transición de un punto de 5v a 0v (flanco de bajada) o de 0v a 5v (flanco de subida)

Función en Arduino que identificará el primer carácter y formateará el TFT para dibujar el resultado.

```
unsigned long write_LCD(String text) {
  char firstChar=text.charAt(0); //read first char for switch case
  String txt = text.substring(1); // save value the text for display
  unsigned long start = micros(); //number of microseconds from which Arduino board
  tft.setRotation(2); //rotation LCD
  switch (firstChar) { //check first char for Style
    //Speakon Initial Ready
    case '#':
      tft.fillScreen();
      tft.setCursor(5,60);
      tft.setTextColor(WHITE);
      tft.setTextSize(2);
      tft.fillRect(0,20, 144, 100, BLUE);
      tft.println(txt);
      break;
      //System Recording
    case '%':
      tft.fillScreen();
      tft.setCursor(5,60);
      tft.setTextColor(WHITE);
      tft.setTextSize(2);
      tft.fillRect(0,40, 144, 60, RED);
      tft.println(txt);
      break;
    //System Error
    case '-':
      tft.fillScreen();
      tft.setCursor(5,60);
      tft.setTextColor(WHITE);
      tft.setTextSize(2);
      tft.fillRect(0,40, 144, 60, RED);
      tft.println(txt);
      break;
    //txt translated for default
    default:
      tft.fillScreen();
      tft.setCursor(3,3);
      tft.setTextColor(YELLOW);
      tft.setTextSize(1);
      tft.println(text);
      break;}
  return micros() - start;
```

Recibirá una cadena de texto. Leerá el primer carácter y mediante una estructura *switch-case* se formateará y dibujará el contenido.

Se realizará un borrado del TFT y se configurarán los parámetros de :pos.(x,y), color de fuente, tamaño y un rectángulo destacable. Finalmente, se enviará el texto.

Tiempo desde que Arduino comenzó a ejecutar el sketch .Este número se superará (volverá a cero), después de aproximadamente 70 minutos.

14. Prototipo

14.1 Speak-On V1.0

En la versión 1.0 de Speak-On es reseñable destacar el modelo construido, donde se ha utilizado piezas de LEGO © y una lente de x10 aumentos. La pantalla se ha ubicado aprox. a 24cm del ojo. La imagen proyectada ha incidido en un espejo orientado 45º respecto al TFT y de nuevo se ha reflejado en un cristal para corregir la orientación.

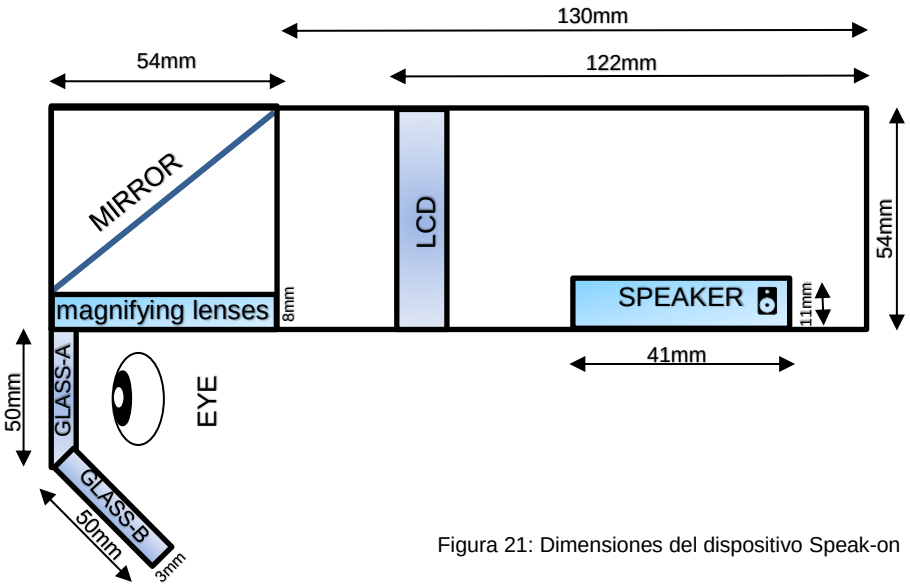

Figura 21: Dimensiones del dispositivo Speak-on

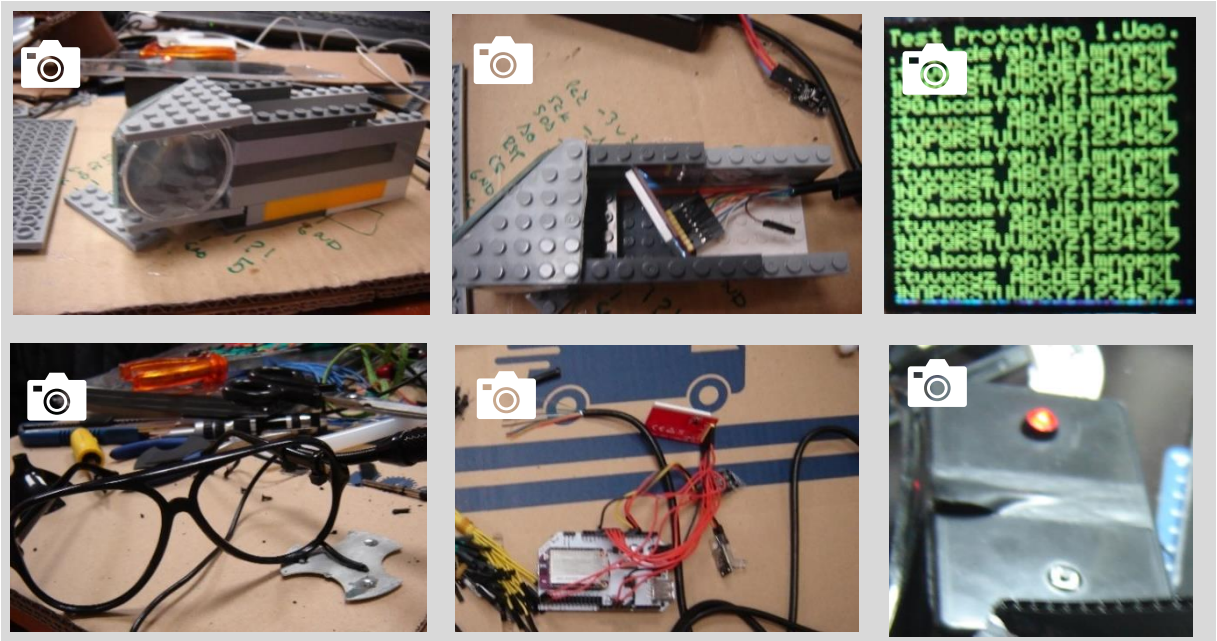

Figura 19:Proceso de construcción y test de la versión Speak-on 1.0

14.2 Speak-On V1.1

Se ha rediseñado el diseño original, con el fin de, mejorar la integración de las lentes y su proyección frontal en los cristales. En consecuencia, se ha trabajado con PVC(policloruro de vinilo), metacrilato y finalmente, piezas de LEGO ©[29] en los adaptadores de los cristales. El *display* se ha fijado a la estructura de las lentes y se ha conectado un altavoz de 35 ohmios/1W para el retorno acústico. Los niveles de audio de línea iniciales han sido corregidos debido a un nivel sonoro deficiente, para ello, se ha modificado el circuito electrónico y se ha conectado en la salida de línea de la tarjeta de sonido USB un circuito integrado PAM8403 (amplificador de 3W y 5 Voltios) alimentado por el *Arduino-dock.*

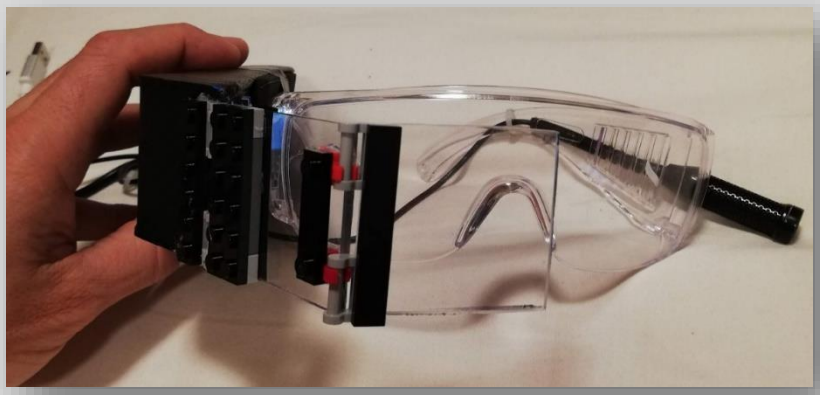

Figura 20:Imagen del prototipo Speak-On V1.1

15. Proyección a futuro

La posibilidad que ofrece Speak-On en los métodos de transcripción, traducción y síntesis del contenido de una conversación, dibuja un nuevo prisma funcional: la educación.

Es factible plantear un entorno educativo estimulado por el dispositivo electrónico donde los alumnos realicen diversas acciones formativas ,con el propósito de, asimilar conceptos lingüísticos asociados con los idiomas del proyecto. El grado de conectividad de la solución ,facilidad en la gestión de configuración y accesibilidad en la interfaz propuesta, ofrecerán un perfil adecuado a diversos grupos de edad sin restricciones.

De igual manera, la estimulación audiovisual proporcionará una importante vía de investigación en pacientes afectados a nivel cognitivo.

[29] Empresa que comercializa diferentes juguetes donde destaca el producto de bloques de construcción.

En concreto ,en las afectaciones auditivas de carácter súbito, las lentes aportarán una herramienta de apoyo elemental ,asociada a la capacidad de comunicación de un sujeto sin la necesidad previa de un proceso de aprendizaje en la lectura de los labios o en el lenguaje de los signos .
Sin embargo, será recomendable abordar importantes modificaciones donde se ha considerado vital reducir las dimensiones y el peso, mejorar la autonomía e industrializar el producto.

16.Viabilidad económica.

En el cálculo de coste del producto Speak-On se ha utilizado el Modelo de Coste Completo en el que se ha repercutido al dispositivo el coste total de producción y los costes no operativos, con el fin de obtener el coste final del producto.

16.1 Costes

- **COSTE VARIABLE UNITARIO:**

Descripción	*Costes	Concepto de impacto en el producto
Costes de producción	35€	Materiales
Costes de Diseño	2€	Diseño en constante evolución con el objetivo de reducir costes
Costes de Fabricación	12€	Mano de obra
Costes de Comercialización	2€	Comisiones de ventas
Costes de Administración	1€	Coste derivado de la función administrativa
Total unitario	52€	
*Relacionados con el volumen de producción		

Tabla 37:Coste variable unitario.

- **PRODUCCIÓN ANUAL:**

Estimación de producción anual:	
Producto	*Unidades /año
Speak-On	1.200
*La producción se ha basado en el trabajo de un operador 5 días / semana en una jornada de 7 horas con una producción aproximada de 5 unidades/día.	

Tabla 38:Estimación producción anual.

- **RESUMEN DE COSTES ANUALES:**

Costes variables /año	*Costes fijos/año	Coste de producción/año
62.400€	2.600€	65.000€
*Estimación variable	*Estimación de gastos en comunicaciones e infraestructura	Totales

Tabla 39:Costes de producción anuales.

- **COSTE UNITARIO:**

Coste Unitario	Total
Costes fijos totales + Costes variables totales) **/** (Número de unidades producidas)	54,16€

Tabla 40:Coste total unitario

16.2 Rentabilidad:

- **MARGEN DE BENEFICIO:**

Coste	Margen de beneficio	Precio venta	Beneficio Bruto
54,16€	30%	70€	15,84€

Tabla 41:Beneficio/unidad

- **INVERSIÓN INICIAL:**

**Inversión Inicial	Total
	15.000€
*Equipamiento necesario para iniciar la actividad empresarial	

Tabla 42:Estimación inversión anual.

- **AMORTIZACIÓN CONTABLE DE LA INVERSIÓN:**

*Previsión inicial de ventas anuales	Beneficio Anual		Unidades necesarias	Recuperación de la inversión
1.100 u.	17.424€		947u.	10,29 meses
*Estimación de ventas en base a los datos recogidos en el estudio del INE (Instituto Nacional de Estadística) (26)				

Tabla 43:Retorno de inversión

16.3 Previsión de evolución de ventas

Año	Estimación /unid.	*Beneficio
1	1.100u.	17.424€
2	1.800u.	28.512€
3	2.400u.	38.016€
4	3.000u.	47.520€
5	4.100u.	64.944€
*No se ha contemplado la posible reducción de costes de producción en el evolutivo anual		

Tabla 44:Previsión de ventas

17. Conclusiones

Los datos analizados en el bloque de conversión de voz a texto, han confirmado que el sistema STT de Wit.AI ha respondido a las necesidades iniciales del proyecto. Los parámetros de latencia y el porcentaje de errores han estado dentro del marco definido, Sin embargo, ha habido peticiones en las que la respuesta de la API ha oscilado de manera aleatoria. (aproximadamente un 15%).

De igual forma, la traducción de texto se ha situado en los márgenes y términos de retardo en la respuesta y retorno del contenido. Por otro lado, los objetivos del bloque de síntesis de voz se han cumplido, la simulación del lenguaje con *Svox* ha ofrecido un nivel de calidad aceptable, asimismo, los parámetros examinados de latencia han sido inferiores al tiempo estipulado. En los idiomas catalán y portugués, Microsoft ha demostrado una alta capacidad, robustez e integración en su API, las voces utilizadas '*HerenaRUS'* y '*HeliaRUS'*, han ofrecido un resultado muy realista.

Adicionalmente, se ha cumplido uno de los objetivos secundarios en el proyecto: utilizar Python como eje principal. Las ventajas que ha aportado Python han sido determinantes, especialmente en la comunicación bidireccional Arduino-Linux supervisada en un único elemento electrónico.

De la misma manera, Arduino, ha ejecutado el código Processing[30] de forma eficaz y no ha demostrado errores en el control de pantalla .La gestión de los rebotes del pulsador no ha manifestado incidencias reseñables con la librería seleccionada. Por otro lado, se ha detectado excesiva sensibilidad eléctrica en el Arduino-Dock, el microcontrolador utilizado ha sido una versión SMD '*surface-mount device'* (montaje en superficie) y se ha necesitado sustituir dos placas durante el ensamblaje debido a la electricidad estática e imposibilidad de sustituir el microcontrolador, sin embargo, se ha solucionado utilizando un brazalete antiestático en la instalación de los componentes.

Igualmente, desde una perspectiva crítica, será necesario examinar el apartado de construcción del modelo. El peso y dimensiones del dispositivo no han sido adecuados. Un elemento de estas características requiere un tamaño reducido con un sistema de anclaje adecuado en cualquier tipo de lentes. Adicionalmente, será inevitable reconsiderar el proceso de conectividad de internet y agregar en futuras revisiones un bloque de comunicación 4G /LTE o 5G. Por último, en el apartado físico del modelo construido, será preciso comprobar y optimizar la presentación visual con doble lente, la distancia focal y luminosidad de la pantalla.

[30] Lenguaje de programación utilizado en las placas electrónicas de Arduino.

Anexo 1. Entregables del proyecto.

Lista de archivos entregados y su descripción.

DISEÑO Y TEST.	
NOMBRE	**DESCRIPCIÓN**
/design/Speakon.fzz	Diseño Fritzing del circuito electrónico
/test/speech/test_Microsoft_PHP_speech.php	Test.Versión Microsoft Speech en PHP
/test/speech/test_Microsoft_Python_speech.py	Test.Versión Microsoft Speech en Python
/test/speech/test_Watson_php_speech.php	Test.Versión Api IBM Speech en PHP
/test/speech/test_Watson_python_speech.py	Test.Versión Api IBM Speech en *Python*
/test/speech/test_Wit_PHP_speech.php	Test.Versión Api Wit.AI Speech en PHP
/test/speech/test_Wit_python_speech.py	Test.Versión Api Wit.AI Speech en *Python*
/test/translate/ translate_microsoft_uoc.php	Test.Versión Api Translate Microsoft PHP
/test/translate/ translate_ibmwatson_uoc.py	Test.Versión Api IBM Watson en *Python*
/test/translate/ translate_microsoft_uoc.php	Test.Versión Translate Microsoft en PHP
/test/translate/ trans_microsoft_yandex_googletr.py	Test Yandex,google,Microsoft en *Python*

Tabla 45:Localización de ficheros de diseño y test

PROYECTO.	
NOMBRE	**DESCRIPCIÓN**
/Speakon/index.html	Fichero principal de configuración HTML .
/Speakon/Speakon.py	Fichero principal de ejecución en *Python*.
/Speakon /options/config.txt	Fichero de configuración.
/Speakon /js	Librerías JS de *bootstrap y jQuery*.
/Speakon /functions/ func_write_configuration_uoc.php	Fichero de escritura de configuración.
/Speakon/python_Speakon/audio_Speakon_uoc.py	Funciones de audio: *play,recorder*,etc..
/Speakon/python_Speakon/communication_Speakon_uoc.py	Fichero de escritura/lectura del puerto serie
/Speakon/python_Speakon/parameters_Speakon_uoc.py	Lectura de config, contador de letras y seccionado de cadena de texto.
/Speakon/python_Speakon/speech_text_Speakon_uoc.py	Función conversión Voz-Texto WIT y diccionario de *token* vinculada al lenguaje
/Speakon/python_Speakon/translate_text_Speakon_uoc.py	Función *Translate* con *Yandex y Microsoft*
/Speakon/python_Speakon/voice_sinth_Microsoft_uoc.py	Voces en Catalán y Portugués con *Microsoft (personalización)*.
/Arduino/CODE_SPEAKON_ARDUINO/ CODE_SPEAKON_ARDUINO.ino	Código *Processing* para el control de la placa Arduino
/Libraries/	Librerías de Arduino utilizadas.

Tabla 46:Localización de ficheros del proyecto

Anexo 2. Código fuente.

Configuración previa del entorno operativo.

Para operar con el diseño propuesto ha sido fundamental abordar una serie de modificaciones en la estructura del sistema operativo de Onion, se han instalado librerías, controladores y se ha optimizado la configuración. Una vez ensamblado la *dockstation* de Arduino y el circuito integrado Onion Omega2+ se ha activado una red inalámbrica con un SSID Omega-xxxx donde las 4 últimas cifras han coincidido con las 4 de la dirección MAC del chip de Onion. El *hostname* inicial para el acceso web ha sido el siguiente:

```
Omega-xxxx. local / 192.168.3.1
username: root
password: onioneer
```

Speak-On ha utilizado la red de internet para las peticiones a las diferentes API del proyecto. Por lo tanto, ha sido necesario conectar la solución a una red inalámbrica.

Figura 21: Imagen de selección de red.

Ampliación de almacenamiento a MicroSD.

Se ha ampliado la capacidad del sistema embebido OS LINUX Openwrt [31] con la ayuda de la utilización de una tarjeta microSD y un archivo de intercambio.

Omega2+ dispone de un almacenamiento de 32 MB, sin embargo, el sistema operativo ha ocupado aproximadamente 10MB ,por lo tanto, el tamaño libre sería muy reducido para incluir las librerías y el proyecto).

[31] Distribución Linux con licencia GPL, optimizada para sistemas con limitaciones de hardware.

En consecuencia, se ha efectuado el procedimiento descrito en la documentación del fabricante y se ha configurado una tarjeta microSD de 16GB .

Se ha desplazado el directorio local `/overlay` a la tarjeta microSD, con el objetivo de, facilitar el despliegue y evitar el desbordamiento de memoria.

En los sistemas embebidos la carpeta `/overlay` es la responsable de almacenar el software y estructura del sistema (tamaño variable) y el directorio `/ROM` tamaño fijo) ofrece el acceso al firmware para mostrar el árbol OS Linux convencional.

Dispositivo de audio.

Se ha ejecutado la instalación de los controladores y *software* de la tarjeta de sonido, con el propósito de permitir utilizar diversos comandos de carácter multimedia: `aplay` (reproducción de ficheros de audio), `arecord` (grabación de audio) etc.

El controlador utilizado ha sido ALSA*' Advanced Linux Sound Architecture'* (arquitectura de sonido de Linux avanzada), herramienta básica con compatibilidad OSS *'Open Sound System' (*Sistema de sonido abierto*).*

```
opkg update
opkg install alsa-utils alsa-lib
```

Calibración inicial del sonido.

El comando `alsamixer` ha facilitado el acceso a los niveles de ajuste de reproducción y grabación.

En la configuración de Speak-On se ha calibrado el micrófono con la ayuda de la generación de un tono de 1Khz a una distancia aprox. de 85cm y se ha regulado el canal de entrada (84) equivalente a 1dB.

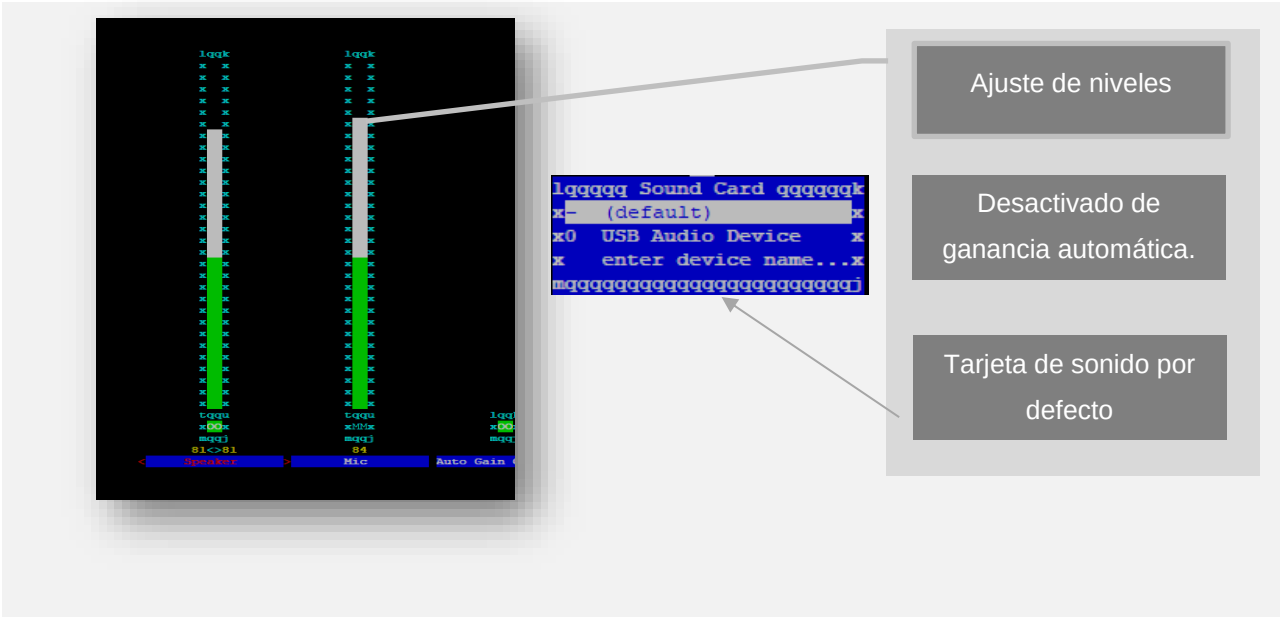

Figura 22 Imagen de configuración del sonido

Estructura de directorios.

El proyecto se ha beneficiado de la existencia de un servidor web en el sistema operativo Linux. Con el objetivo de evitar código innecesario, se ha establecido un directorio dominante (`Speakon)` en la actual estructura web del sistema operativo.

```
mkdir /www/Speakon
```

Asimismo, se ha creado la estructura de carpetas del proyecto:

```
mkdir /www/Speakon/options
mkdir /www/Speakon/functions
mkdir /www/Speakon/records
mkdir /www/Speakon/js
```

Cambio de permisos de forma recursiva

Los usuarios no dispondrán de privilegios para modificar el contenido de los ficheros.

```
chmod 755 -R Speakon
```

Fichero de configuración de lenguajes y traducción

```
touch /www/Speakon/options/config.txt
```

Instalación de PHP.

```
opkg update
opkg install php7 php7-cgi
```

Libreria para trabajar con JSON

```
opkg install  php7-mod-json
```

Activación de PHP en el servidor.

Se ha editado el fichero de configuración del servidor web:

```
vi /etc./config/uhttpd
```

A continuación, ha sido necesario añadir al final del bloque main del fichero `uhttpd` la ubicación de PHP y la página principal a procesar:

```
list interpreter ". php=/usr/bin/php-cgi" option index_page 'index.php'
```

Reinicio del servidor web:

```
/etc/init.d/uhttpd restart
```

Instalación de cURL

Librería que permitirá trabajar con servidores desde línea de comandos en el examen funcional.

```
opkg install curl
```

Instalación de Python

```
opkg install Python
```

Instalación de git

```
opkg install git git-http ca-bundle
```

Gestor de paquetes PIP

```
opkg update
opkg install python-pip
pip install --upgrade pip
```

Comunicación en python-Arduino. Puerto serie.

```
opkg install python-pyserial
```

Librería Googletrans

```
pip install googletrans
```

Librería wit.ai

```
pip install wit
```

Librería para solicitudes HTTP

```
pip install httpclient
```

Algoritmos en Python.

Python*: comunicación puerto serie con Arduino (escritura y lectura).*

```
########################################
#function for read serial port#########
########################################

def read_serial_uoc():
        try:
          serial_p= serial.Serial('/dev/ttyS1',
            baudrate=9600,
            bytesize=serial.SEVENBITS,
            parity=serial.PARITY_EVEN,
            stopbits=serial.STOPBITS_ONE
          )
```

```
          time.sleep(0.5)
          serial_p.isOpen() # try to open port, if possible proceed with 'while True:'
          read = int(serial_p.readline().decode()) #read and convert value to int 0 or 1

        except IOError:
        ser.close()
          time.sleep(0.5)
          ser.open()
          read = int(serial_p.readline().decode()) #read and convert value to int 0 or 1

        return read #return  0 or 1

########################################
#function for write serial port#########
########################################

def write_serial_uoc(text):
        #decomposed normalized form of the unicode string remove accents
        text_normalize=unicodedata.normalize("NFKD",
text).encode("ascii","ignore").decode("ascii")
        text_normalize_special_char=bytes(text_normalize.encode().strip())

        try:
          serial_p= serial.Serial('/dev/ttyS1',
            baudrate=9600
          )
          time.sleep(0.5)
          serial_p.isOpen() # try to open port, if possible print message and proceed
with 'while True:'
          serial_p.write(text_normalize_special_char) #write external var


        except IOError: # if port is already opened, close it and open it again
          ser.close()
          time.sleep(0.5)
          ser.open()
          serial_p.write(text_normalize_special_char) #write external var
```

Python: Síntesis de voz en catalán y portugués.

Función que recibe dos parámetros: lenguaje final y el texto traducido. Finalmente, generará un fichero de audio para su posterior reproducción

```
def voice_cata_portu_uoc (text_captured, language_returned):
        apiKey = "be4aa6990dc846f1a4f92d6a20857678"
```

```
        params = ""
        headers = {"Ocp-Apim-Subscription-Key": apiKey}
        AccessTokenHost = "westeurope.api.cognitive.microsoft.com"
        path = "/sts/v1.0/issueToken"

        # Connect to server to get the Access Token
        print ("Connect to server to get the Access Token")
        conn = http.client.HTTPSConnection(AccessTokenHost)
        conn.request("POST", path, params, headers)
        response = conn.getresponse()
        print(response.status, response.reason)

        data = response.read()
        conn.close()

        accesstoken = data.decode("UTF-8")
        #print ("Access Token: " + accesstoken)

        body = ElementTree.Element('speak', version='1.0')
        body.set('{http://www.w3.org/XML/1998/namespace}lang', language_returned)
        voice = ElementTree.SubElement(body, 'voice')
        voice.set('{http://www.w3.org/XML/1998/namespace}lang', language_returned)
        voice.set('{http://www.w3.org/XML/1998/namespace}gender', 'Male')

        #Catalán
        #"Microsoft Server Speech Text to Speech Voice (ca-ES, HerenaRUS)"
        if language_returned=='ca-ES':
                voice_use=', HerenaRUS)'
        #Portugues
        #"Microsoft Server Speech Text to Speech Voice (pt-PT, HeliaRUS)"
        if language_returned=='pt-PT':
                voice_use=', HeliaRUS)'

        voice.set('name', 'Microsoft Server Speech Text to Speech Voice
('+language_returned+voice_use)
        #read text captured
        voice.text = text_captured
        headers = {"Content-type": "application/ssml+xml",
                                "X-Microsoft-OutputFormat": "riff-24khz-16bit-mono-
pcm",
                                "Authorization": "Bearer " + accesstoken,
                                "X-Search-AppId": "07D3234E49CE426DAA29772419F436CA",
                                "X-Search-ClientID":
"1ECFAE91408841A480F00935DC390960",
                                "User-Agent": "TTSForPython"}
```

```
        #Connect to server to synthesize the wave
        conn = http.client.HTTPSConnection("westeurope.tts.speech.microsoft.com")
        conn.request("POST", "/cognitiveservices/v1", ElementTree.tostring(body),
headers)
        response = conn.getresponse()
        data = response.read()
        f = open('/www/Speakon/records/voice_translate.wav','wb')
        f.write(data)
        f.close()
        conn.close()
```

Audio.

-Conversión a fichero de audio previa discriminación del lenguaje.

```
######################################################################
#function generates a wav file with the selected language and text
######################################################################
def text_voice_uoc (text,language):
        #processing with Microsoft Api Translate
        if language=='ca-ES' or language=='pt-PT':
                #Catalan-Portuguese ,api resolution
                voice_cata_portu_uoc(text,language)

        else:
                #en-US, en-GB, de-DE, es-ES, fr-FR and it-IT.
                p=subprocess.Popen(["pico2wave",                              "--wave",
"/www/Speakon/records/voice_translate.wav",  "-l", language,  text], shell=False)
                p.wait()
```

-Test de Acapela.

```
#Function created to work with the Acapela Group Development speech synthesis
def acapela_speech_uoc (text_captured,language_returned):
tts_acapela=acapela.Acapela("EVAL_VAAS","EVAL_5341578","s9oo8tli",
"http://vaas.acapela-group.com/Services/Synthesizer", "22k", "/www/Speakon/records/")
tts_acapela.prepare(text=text_captured,lang=language_returned,gender="W",
intonation="NORMAL")
        output_filename = tts_acapela.run()
```

Funciones de inicio y detención del procedimiento de reproducción del sonido.

Las invocaciones a los comandos utilizados se han efectuado con la ayuda de subprocesos (27) en Python y con activación de espera de finalización del proceso ejecutado (`p.wait()`) a excepción de la función `recording_voice_uoc ()`.

```
############################################################
#function record Voice with driver ALSA
############################################################
def recording_voice_uoc ():
        #os.popen("arecord -f S16_LE -d 5  -r 48000 test.wav")
        p=subprocess.Popen(["arecord",     "-f",     "S16_LE",     "-r",     "48000",
"/www/Speakon/records/voice.wav"], shell=False)
        #

############################################################
#function stop record
#IMPORTANT.this process will be stopped by killall so we will not use the WAIT () method
############################################################
def stop_recording_voice_uoc():
        p=subprocess.Popen(["killall","arecord"], shell=False)
        p.wait()

############################################################
#function play voice
############################################################
def play_voice_uoc ():
        #en-US, en-GB, de-DE, es-ES, fr-FR and it-IT,ca-ES,pt-PT
        p=subprocess.Popen(["aplay",      "/www/Speakon/records/voice_translate.wav"],
shell=False)
        p.wait()
```

Funciones adicionales.

Conversión a valor booleano (recepción del pulso de Arduino), contador de caracteres, lectura de ficheros de configuración y división del contenido para su presentación.

```
from python_Speakon.communication_Speakon_uoc import write_serial_uoc
import time

#this function opens and reads the txt file for it receives a parameter the line number
to read

def read_configuration_uoc(line_number):
        with open("/www/Speakon/options/config.txt", "r") as f:
            lines = f.readlines()
```

```
            readline= lines[line_number].rstrip('\n')
            return readlin
#this function collects the value of the third row of the txt file and converts the
string to a Boolean value
#It is important to know if you want to translate the content, because the translation
block would not be necessary
def convert_string_boolean_uoc(parameter):
    if parameter == '1':
         return True
    else :
         return False

##############################LCD################################################
#Function that counts letters
#to be able to delimit the number max of chars by
#serial port to the LCD display UOC

def count_uoc (x):
    count = 0
    for letter in x:
            count = count + 1
    return(count)

#This function is important.
#The display used has a maximum of 315 characters in size 1.
#In order to present more characters, a function has been designed
#that uses a for loop to present the text every 7 seconds

def split_for_display_uoc(text,n):
        for x in range(0, n):
                #clear display
                write_serial_uoc('*')
                #calc position
                ch_start=315*x
                ch_end=ch_start+315
                #split string
                txtprocess = text[ch_start: ch_end]
                #print display
                write_serial_uoc(txtprocess)
                #time for read  text in display 7 seconds
                time.sleep(7)
```

Anexo 3. Capturas de pantalla

Imágenes de la construcción del prototipo funcional.

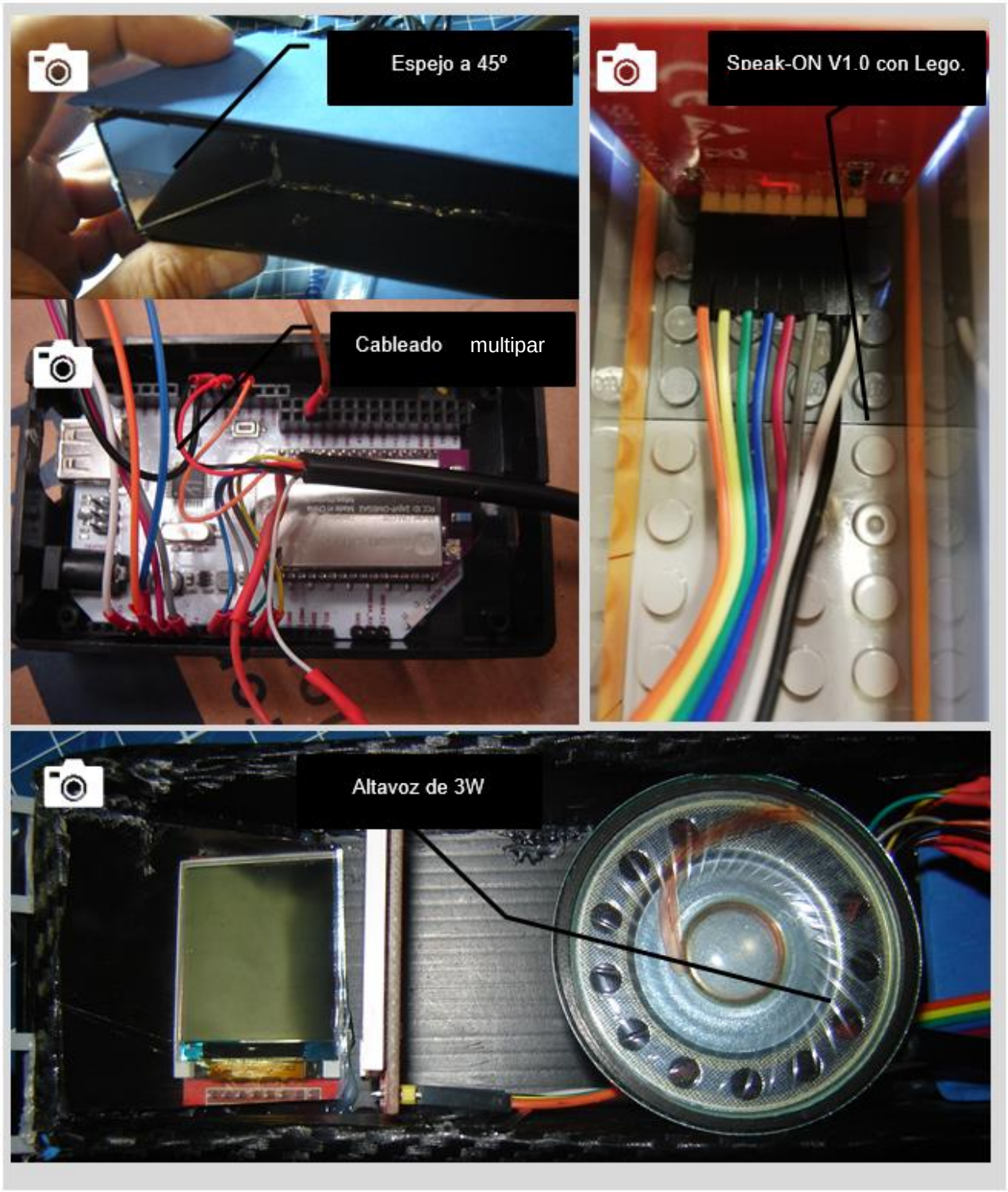

Figura 23:Imágenes del prototipo inicial

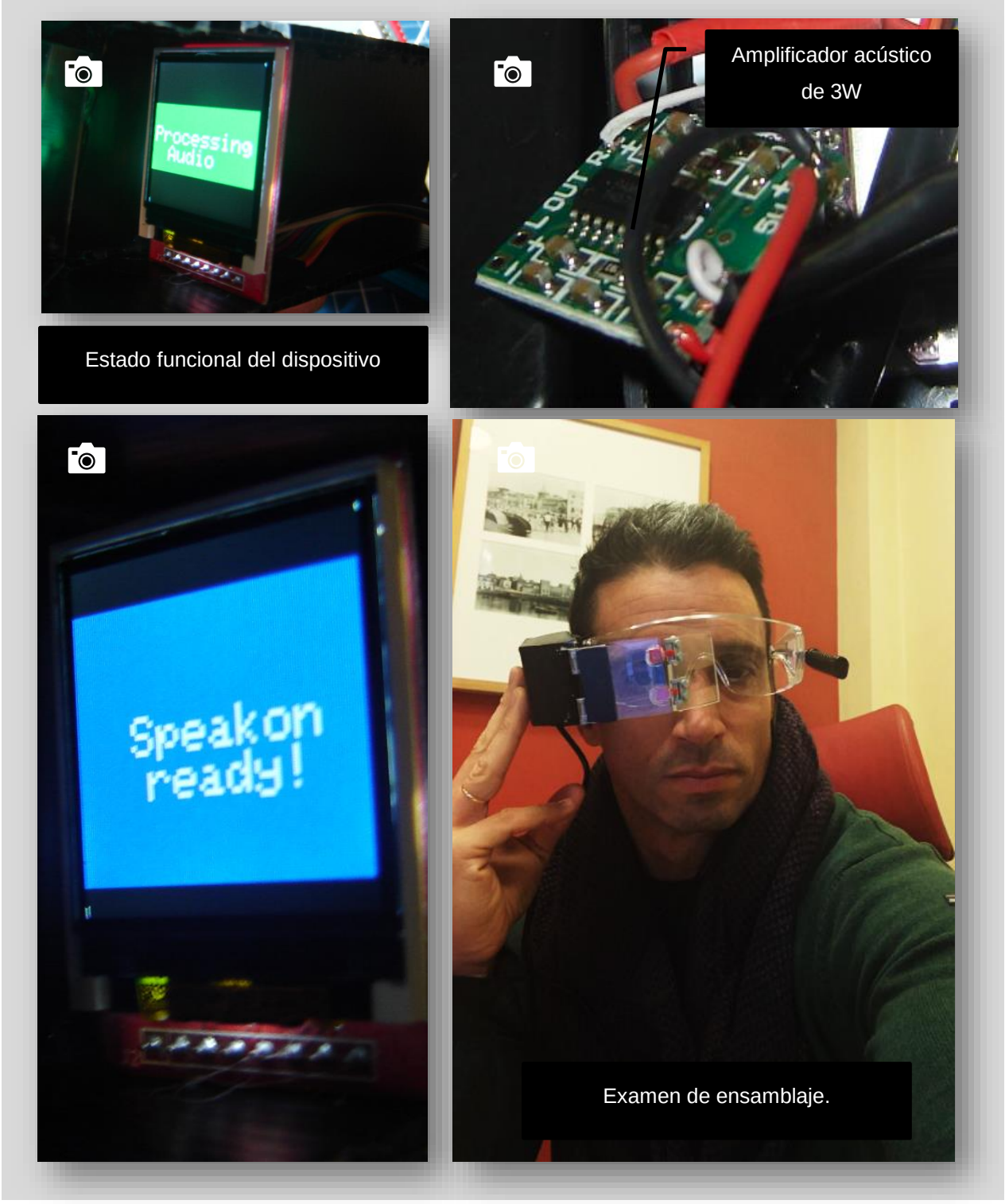

Figura 24 Imágenes del estado funcional del equipo electrónico.

Speak-On V1.1

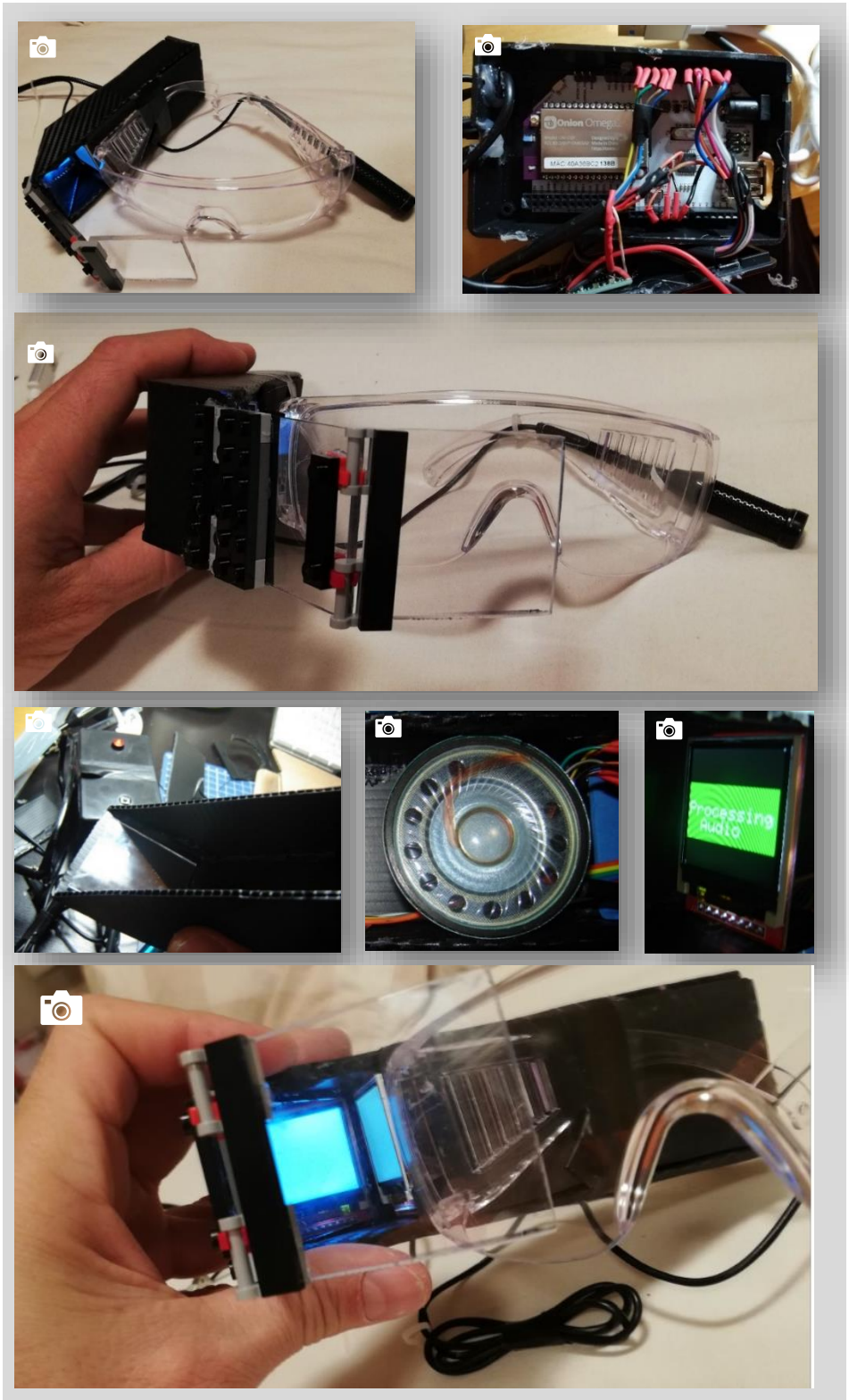

Figura 25.Versión 1.1.Test de proyección

Anexo 4. Guía de usuario

1.Acceso al dispositivo Speak-ON.

Será necesario realizar la conexión WIFI con el dispositivo cognitivo. Para ello, el SSID proporcionado será :**SPEAKON** y la contraseña de acceso :**12345678.**

2.Acceso via web a la configuración.

Una vez conectado al equipo ,se deberá añadir una red inalámbrica disponible en la siguiente URL: http://Speakon.local/cfg/

Figura 26:Conexión a red inalámbrica del sistema

3. Acceso al portal de configuración de la herramienta.

URL de configuración: http://Speakon.local/speakon/

Se determinarán los idiomas del dispositivo y la posibilidad de traducir la conversación registrada. Posteriormente, se almacenarán las opciones en el sistema.

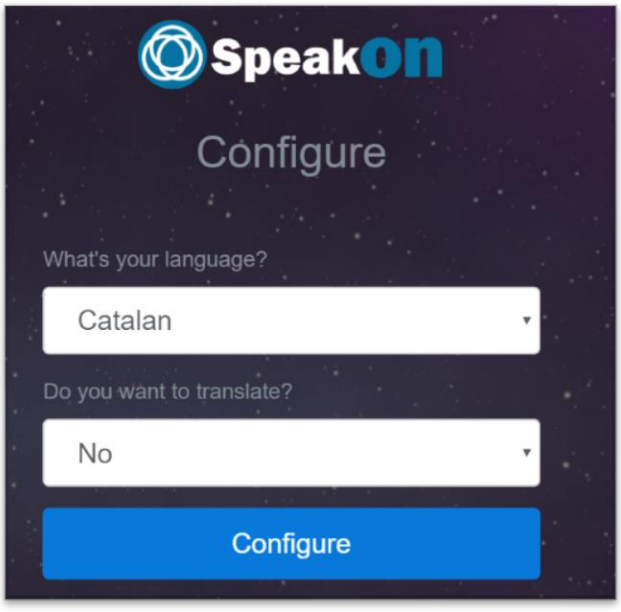

Figura 27:Página web de gestión de la herramienta

4. Control funcional de inicio/detención de Speak-ON.

La gestión del procesamiento de una conversación se efectuará mediante el uso de un pulsador.
Una vez inicializado el dispositivo el led permanecerá en color rojo. Una locución recordará que Speak-ON estará preparado. Además, en la pantalla óptica aparecerá el texto *Speakon Ready*
Para iniciar una grabación será necesario accionar el pulsador, el led cambiará a color verde (acompañado de una locución y un texto de inicio de grabación: *Recording Voice*). La duración máxima será de aproximadamente 90 segundos, con el objetivo de no colapsar el almacenamiento. Si el pulsador es de nuevo presionado, el sistema transmitirá a nivel audiovisual la detención (locución y presentación en pantalla del texto *processing Audio*).Finalmente, se procesará la grabación y se retornará el texto en el *display* y el audio sintetizado.

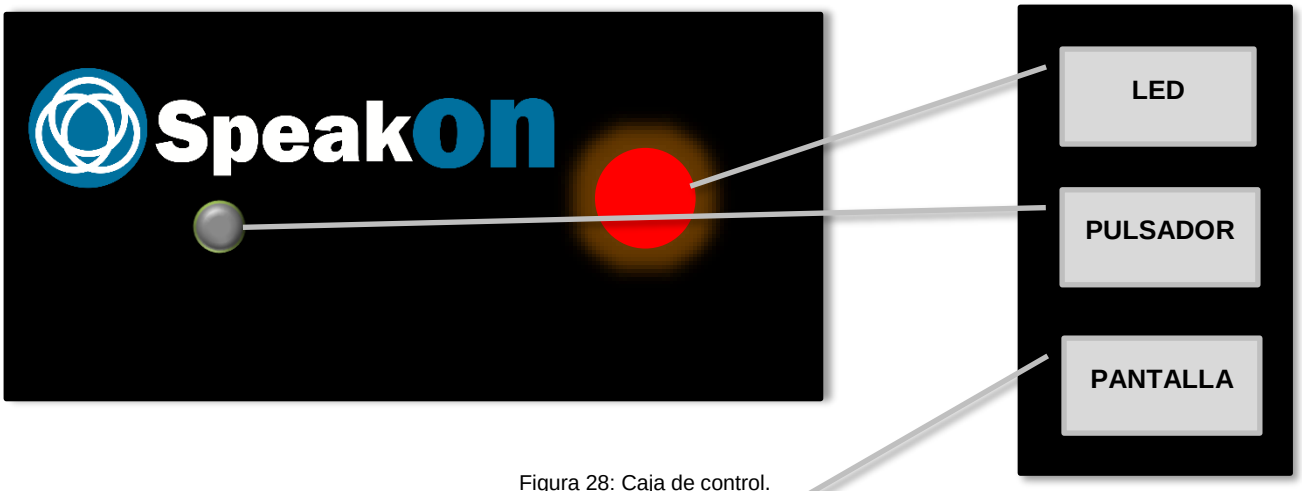

Figura 28: Caja de control.

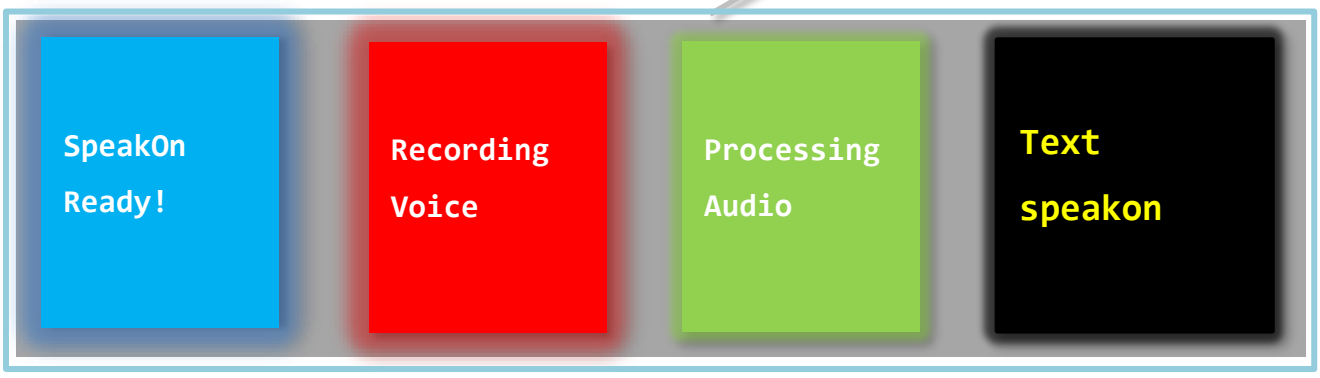

Figura 29: Estados del sistema.

Anexo 5. Bibliografía

1. **Ramírez, Luis Leonardo Yance.** Scielo. [En línea] 13 de enero de 2000. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-21412000000300002.
2. **ONU.** Convención sobre los derechos de las personas con discapacidad y protocolo facultativo. [En línea] 2006. [Citado el: 20 de 12 de 2018.] http://www.un.org/disabilities/documents/convention/convoptprot-s.pdf.
3. **Foundation, Python Software.** python.org. [En línea] 2018. https://docs.python.org/3/installing/index.html.
4. **Melnikov, A.** Internet Engineering Task Force (IETF). [En línea] 2011. http://www.rfc-editor.org/info/rfc6455.
5. **IBM.** Watson Speech to text. [En línea] 2018. https://www.ibm.com/watson/services/speech-to-text/.
6. **bbvaopen4u.** bbvaopen4u. [En línea] 2016. [Citado el: 10 de 12 de 2018.] https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos.
7. **Microsoft.** Speech to text.Microsoft. [En línea] 2018. https://azure.microsoft.com/es-es/services/cognitive-services/speech-to-text/.
8. **Wit.AI.** Wit.AI Speech. [En línea] 2018. https://wit.ai/.
9. **Microsoft.** Microsoft Translator. [En línea] 2018. https://azure.microsoft.com/es-es/services/cognitive-services/translator-text-api/.
10. **Translator, Ibm.** Ibm Translator. [En línea] https://www.ibm.com/watson/services/language-translator/.
11. **Han, SuHun.** Pip Library Python. *Pip Library Python.* [En línea] 2018. https://pypi.org/project/googletrans/.
12. **Translate, Yandex.** Yandex Translate. [En línea] 2018. https://tech.yandex.com/translate/.
13. **Paconaranjo.** Wikipedia. [En línea] 2014. https://es.wikipedia.org/wiki/Speech_Synthesis_Markup_Language.
14. **K. Palacio, J. Auquilla, E. Calle.** Escuela Superior Politécnica del Litoral (ESPOL). [En línea] 2008. http://www.rte.espol.edu.ec/index.php/tecnologica/article/view/145.
15. **King, Simon.** The Centre for Speech Technology Research, The University of Edinburgh, Spain. [En línea] http://loquens.revistas.csic.es/index.php/loquens/article/view/6.
16. **Aylett, Dr Matthew.** Cerevoice. [En línea] 2017. https://www.cereproc.com/en/products/sdk.
17. **Acapela.** Acapela. [En línea] 2018. https://www.acapela-group.com/.
18. **Edinburgh, The University of.** The Centre for Speech Technology Research. [En línea] 2018. http://www.cstr.ed.ac.uk/projects/festival/.
19. **Espeak.** Espeak. [En línea] http://espeak.sourceforge.net/.
20. **LearnBrite.** Responsive voice. [En línea] 11 de 2018. https://responsivevoice.org/.
21. **Jantzen, Volker.** Svox. *Svox.* [En línea] 2001. https://en.wikipedia.org/wiki/SVOX.
22. **Microsoft.** Microsoft Text To Speech. [En línea] Diciembre de 2018. https://azure.microsoft.com/es-es/services/cognitive-services/text-to-speech/.

23. **Arduino.cc.** ***Arduino.cc.*** [En línea] diciembre de 2018. [Citado el: 27 de diciembre de 2018.] https://store.arduino.cc/arduino-yun.

24. Onion. ***Onion.*** [En línea] Diciembre de 2018. [Citado el: 27 de Diciembre de 2018.] https://docs.onion.io/omega2-docs/.

25. Onion.Características. [En línea] 26 de diciembre de 2018. [Citado el: 26 de diciembre de 2018.] https://docs.onion.io/omega2-docs/arduino-dock-2.html.

26. INE. ***INE.*** [En línea] Junio de 2013. [Citado el: 28 de Diciembre de 2018.] https://www.ine.es/metodologia/t22/analisis_epa_epd.pdf.

27. recursospython. ***recursospython.*** [En línea] 2018. [Citado el: 2018 de 11 de 11.] https://recursospython.com/guias-y-manuales/subprocess-creacion-y-comunicacion-con-procesos/.

28. Web, INTERNET YA - Soluciones. INTERNET YA . [En línea] 2018. [Citado el: 11 de 05 de 2018.] https://www.internetya.co/que-es-y-para-que-sirve-una-api/.

29. MarioFinale. Wikipedia. [En línea] Marzo de 2017. https://es.wikipedia.org/wiki/Token_(inform%C3%A1tica).

30. Melkart4k. Wikipedia. [En línea] 03 de 12 de 2018. [Citado el: 24 de 12 de 2018.] https://es.wikipedia.org/wiki/Kit_de_desarrollo_de_software.

31. S., César San Martín. Universidad de La Frontera, Ingeniería Eléctrica. [En línea] 2004. https://scielo.conicyt.cl/scielo.php?pid=S0718-13372004000100002&script=sci_arttext.

32. Schrenk, Michael. *Webbots, Spiders, and Screen Scrapers.* San Francisco : no starch press, 2012.