# DISEÑO DE UNA INTERFAZ DE RECONOCIMIENTO DE VOZ PARA EL MANEJO DE UNA SILLA DE RUEDAS, MEDIANTE LA IMPLEMENTACIÓN DE SISTEMAS EMBEBIDOS EN FPGA

JUAN DANIEL PRIETO ALMONACID
JEFFERSON MANUEL GARCÍA PRADA

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA ELECTRÓNICA
BUCARAMANGA
2016

# DISEÑO DE UNA INTERFAZ DE RECONOCIMIENTO DE VOZ PARA EL MANEJO DE UNA SILLA DE RUEDAS, MEDIANTE LA IMPLEMENTACIÓN DE SISTEMAS EMBEBIDOS EN FPGA

JUAN DANIEL PRIETO ALMONACID
JEFFERSON MANUEL GARCÍA PRADA

Trabajo de grado para optar por el título de Ingeniero Electrónico

Director
Fabio Alonso Guzmán Serna

UNIVERSIDAD PONTIFICIA BOLIVARIANA
ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA ELECTRÓNICA
BUCARAMANGA
2016

# Dedicatoria

A Dios que estuvo en todo momento guiándonos, dándonos perseverancia y ánimo en momentos difíciles para lograr el título como ingenieros electrónicos.

A mis padres por apoyarme porque gracias a sus esfuerzos y su amor hicieron lo posible para darme esta oportunidad de formarme como persona y como profesional.

A mi hermano que estuvo pendiente y dispuesto en ayudarme anímicamente para poder alcanzar mis metas.

Leidy, por su gran paciencia entendiéndome en momentos buenos y difíciles, por su apoyo en las decisiones que tomé, en darme consejos que me llenaron de valor para continuar y seguir mis sueños.

Juan Daniel, por su esfuerzo y entrega en el proyecto, también se lo dedico a su familia, que compartieron y me aguantaron como un integrante más en el hogar brindado apoyo, colaboración y tiempo.

Holguer, por su disposición estudiando y guiándonos todos estos meses trabajando y buscando formas para el desarrollo del proyecto y además de enseñarnos su conocimiento en el área.

Ing. Fabio Guzman por confiar en nosotros en lograr conseguir los objetivos, también por brindarnos su colaboración, paciencia y experiencia. Además de instruirnos como profesionales.

A todos los profesores que estuvieron durante mis estudios ya que hacen parte de mis bases para el entendimiento de los temas implementados y experiencias vividas en proyectos trabajados que amplió mi visión.

*Jefferson Garcia.*

# Dedicatoria

A Dios por haberme traido hasta aquí, por levantarme en los momentos difíciles y por ayudarme a encontrar un sentido para mi vida. Por darme el regalo más grande del mundo, y por permitir que mis sueños se sigan haciendo realidad.

A las personas que han dedicado toda su vida en pro de mi bienestar y que nunca me han abandonado. Me han enseñado todo lo que se; todo lo que soy es gracias a ustedes y, a pesar de los errores, nunca han dejado de brindarme todo su cariño y amor. A mis padres que nunca se cansarán de luchar por mi, por hacer todos mis sueños realidad. Este proyecto, que es el fruto de mis esfuerzos, no puede ser de nadie más sino de ustedes.

A mis abuelos que siempre han estado muy pendientes de cada uno de mis pasos. Sus cuidados y consejos me han ayudado mucho en las adversidades a las que he tenido que enfrentarme a lo largo de mi vida. Les entrego mi esfuerzo, mi dedicación y mi trabajo. Espero puedan sentirse orgullosos de mi, con cada una de las etapas que ustedes me han ayudado a construir.

A Andrea, por hacer parte de mi vida. Tú me has enseñado mas de lo que crees. Confias tanto en mi y en mis capacidades que llenas todos mis vacios y no permites que la duda llegue a apoderarse en los momentos de dificultad. Gracias por compartir tanto conmigo, y por apoyarme a lo largo del camino. A ti te dedico un pedacito de mi proyecto, quizá el más especial, porque no concibo este momento de mi vida sin haberte conocido.

A Ellie, por ser la promesa de un futuro mejor. A ti te dedico mi trabajo, mi esfuerzo y mi vida entera. Algún día entenderás lo importante que eres para mi.

A mis hermanos Santiago y Fabio, por comprenderme, animarme, apoyarme y ser mi mas fiel soporte. Ustedes alegran mi vida y me complementan. Con mucho cariño les dedico una parte de todo mi esfuerzo.

A Jefferson, por su paciencia, colaboración y entrega en el desarrollo de este proyecto. Le dedico una parte muy importante de mi esfuerzo pues ha sido uno de los apoyos más grandes que he tenido en los momentos de dificultad. Gracias por todo, pues sin su ayuda y comprensión nada de esto hubiera sido posible.

A Holguer, por su entrega, interés, motivación y compromiso. No imagino los resultados obtenidos hoy día sin su invaluable ayuda. Le dedico este logro, que es producto de toda su dedicación.

A Fabio, demás docentes y todos los que han aportado su granito de arena en mi formación profesional, mil gracias. Esto es para ustedes.

*Daniel.*

# Agradecimientos

A Dios por guiarnos para cumplir los objetivos del proyecto y durante nuestra formación profesional brindarnos perseverancia y dedicación.

A la Universidad Pontificia Bolivariana por el apoyo que nos brindaron durante el desarrollo de este proyecto de grado; por los espacios de trabajo y las enseñanzas durante nuestro periodo de aprendizaje.

Al MEng. Holguer Andrés Becerra Daza por compartirnos su invaluable conocimiento en las áreas de diseño avanzado de hardware, programación de sistemas paralelos y sistemas distribuidos aplicados en FPGA, así como el tiempo dedicado a la solución de incontables problemas que se presentaron en la elaboración del sistema ASR.

Al Ing. Fabio Alonso Guzmán Serna por su guía constante en el desarrollo del proyecto, su interés en el proceso de investigación, sus consejos y dedicación que permitieron consolidar esta tesis de pregrado.

Al PhD. Jhon Jairo Padilla Aguilar por creer en el proyecto y apreciar el esfuerzo detrás de todo el proceso de investigación y desarrollo, además de las diversas sugerencias con el animo de mejorar la calidad y presentación del mismo.

Al MEng. Carlos Gerardo Hernandez Capacho por valorar el alcance del proyecto, por sus consejos y sugerencias tanto en el ámbito académico como personal. Por su interés y motivación en el campo de investigación así como su aprobación.

Al MSc. Alex Alberto Monclou Salcedo por estar presente en todo el proceso educativo, por su disponibilidad en la solucion de diversas inquietudes.

Al PhD. Omar Pinzón Ardila por compartirnos su invaluable conocimiento y experiencia, por sus observaciones que seguramente consideraremos en la elaboración de futuros proyectos.

Agradecemos también a todos los docentes que hacen parte del programa de ingeniería electrónica por las enseñanzas recibidas durante esta etapa de nuestras vidas, tanto en el ámbito académico, como formación personal, pues de alguna manera todos participaron de este logro alcanzado.

A la familia que nos apoyó y promovió en todo momento de nuestras vidas, gracias por permitirnos culminar una etapa más de nuestra formación académica.

A nuestros compañeros y futuros colegas por haber compartido con nosotros esta experiencia a lo largo de los últimos años. Por alegrar nuestro proceso de formación y enriquecer el aprendizaje gracias al intercambio de ideas y opiniones.

## TABLA DE CONTENIDO

## LISTA DE FIGURAS

## LISTA DE TABLAS

## GLOSARIO

ACK: Acknowledgement

ADC: Analog to Digital Converter

ARM: Advanced RISC Machine

ASIC: Application-Specific Integrated Circuit

ASR: Automatic Speech Recognition

AXI: Advanced eXtensible Interface

BJT: Bipolar Junction Transistor

DAC: Digital to Analog Converter

DCT: Discrete Cosine Transform

DSP: Digital Signal Processor

DNN: Deep Neural Networks

FET: Field-Effect Transistor

FFT: Fast Fourier Transform

FIFO: First In - First Out

FIR: Finite Impulse Response

FPGA: Field-Programmable Gate Array

FSM: Finite State Machine

F2H: FPGA-to-HPS Bridge

HMM: Hidden Markov Model

HPS: Hard Processor System

H2F: HPS-to-FPGA Bridge

I2C: Inter Integrated Circuit

JTAG: Joint Test Action Group

LSB: Least Significant Bit/Byte

LWH2F: Lightweight HPS-to-FPGA bridge

MCU: Micro-Controller Unit

MFCC: Mel Frequency Cepstrum Coefficients

MPU: Micro-Processor unit

MSB: Most Significant Bit/Byte

NACK: Negative Acknowledgement

OMS: Organización Mundial de la Salud

PCM: Pulse Code Modulation

PFD: Phase Frequency Detector

PIO: Parallel Input Output

PLL: Phase Locked Loop

PWM: Pulse Width Modulation

RTL: Register Transfer Language

SCL: Serial CLock

SDA: Serial DAta

STFT: Short Time Fourier Transform

TTL: Transistor-Transistor Logic

UART: Universal Asynchronous Receiver-Transmitter

VCO: Voltage-Controlled Oscillator

WER: Word Error Rate

**RESUMEN GENERAL DE TRABAJO DE GRADO**

| | |
|---|---|
| **TITULO:** | Diseño de una interfaz de reconocimiento de voz para el manejo de una silla de ruedas, mediante la implementación de sistemas embebidos en fpga |
| **AUTOR(ES):** | Juan Daniel Prieto Almonacid<br>Jefferson Manuel Garcia Prada |
| **FACULTAD:** | Facultad de Ingeniería Electrónica |
| **DIRECTOR(A):** | Fabio Alonso Guzmán Serna |


**RESUMEN**

Este proyecto plantea el desarrollo e implementación de un sistema que permite manipular una silla de ruedas, endosada con motores de corriente continua, utilizando comandos de voz en habla hispana a través de un ecosistema capaz de realizar reconocimiento de voz por deletreo mediante procesamiento de señales y captación de fonemas en una interfaz embebida programada sobre una FPGA. Se retoma un proyecto existente en la Universidad Pontificia Bolivariana el cual utilizaba un analizador de espectros comercial capaz de identificar muestras de audio pre-grabadas para ejecutar acciones en la silla de ruedas. Se adquiere un board de desarrollo DE1-SoC que satisface las necesidades requeridas. Primero se almacena la voz proveniente de un micrófono en formato WAV, para que esto ocurra se debe superar cierto umbral de intensidad. Esta adquisición de datos ocurre en un módulo ADC que muestrea a 8Khz. Para lograr la detección de la o las palabras pronunciadas, se recurren a los coeficientes cepstrales en las frecuencias de MEL (MFCC), los cuales concentran características propias de los fonemas en bancos de filtros, y desecha información poco valiosa que empobrecen el reconocimiento de voz tales como ruido de fondo, emociones, volumen o tono. El algoritmo implementado utiliza varios tipos de transformadas como Fourier (FFT) y Coseno (DCT), así como filtrado digital (FIR) y modelo oculto de Markov (HMM). Finalmente se añade una capa de seguridad mediante la implementación de sensores de radar ultrasónicos (SRF02), conectados a un bus I2C, para evitar colisiones en el desplazamiento. Al integrar todo el proceso matemático procesado en el ARM con aceleración por hardware se logran resultados satisfactorios siendo capaz de reconocer palabras con tiempos de ejecución reducidos y favorables para la implementación del manejo de la silla de ruedas.



**PALABRAS CLAVE:**

ASR, fonema, FPGA, HMM, MFCC, silla de ruedas.


**V° B° DIRECTOR DE TRABAJO DE GRADO**

## GENERAL SUMMARY OF WORK OF GRADE


**TITLE:** Voice recognition interface design used to handling a wheelchair, by implementing embedded systems in fpga

**AUTHOR(S):** Juan Daniel Prieto Almonacid
Jefferson Manuel Garcia Prada

**FACULTY:** Facultad de Ingeniería Electrónica

**DIRECTOR:** Fabio Alonso Guzmán Serna


## ABSTRACT


This project proposes the development and implementation of a system able to handle a wheelchair, endorsed with DC motors, using spanish speaking voice commands through an ecosystem capable to process speech recognition by using signal processing and uptake phonemes into a programmed interface embedded on a FPGA. We've designed based on Universidad Pontificia Bolivariana project which used a commercial spectral analyzer capable of identifying audio samples pre-recorded in order to execute actions on the wheelchair. A DE1-SoC board was acquired because it can execute efficiently the algorithm proposed. First the voice from a microphone is stored in a WAV file, but it must exceed certain intensity threshold before recording. Data acquisition occurs by an analog-to-digital converter (ADC) which sampling at 8 KHz. the goal is design an algorithm capable to detect and understand the words stored in the audio file. The Mel Frequency Cepstral Coefficients (MFCC) are designed as an approximate human voice scale and allow better representation of speech sounds, it's hugely applied in automatic speech recognition software. MFCC keeps some important phonemes characteristics in filter banks, discarding some useless information that impoverish the speech recognition such as background noise, emotions, volume or tone. The implemented algorithm uses different math transformations like Fast Fourier Transform (FFT) and Discrete Cosine Transform (DCT), as well as digital filters (FIR) and the Hidden Markov Model (HMM). Finally, a security layer is added by implementing some ultrasonic radar sensors (SRF02), connected to an I2C bus, in order to avoid possible collisions along the displacement. The math process running in the ARM works truly fine with the hardware acceleration. Both achieved awesome results being capable of recognize many different words in short periods of time, making the final project able to handle the wheelchair successfully.



**KEYWORDS:**

ASR, FPGA, HMM, MFCC, phoneme, wheelchair.


**V° B° DIRECTOR OF GRADUATE WORK**

# INTRODUCCIÓN

Este proyecto apunta específicamente al desarrollo de un motor de reconocimiento de voz por deletreo en tiempo real construido sobre una FPGA, usando para tal fin una board DE1-SoC de Altera. El manejo de una silla de ruedas a partir de las palabras comprendidas en el deletreo es un pilar secundario que demuestra uno de los muchos usos que puede tener un dispositivo capaz de ejecutar diversas tareas a partir de comandos de voz.

El sistema fue diseñado para ser capaz de reconocer las palabras captadas por un micrófono a lo largo de un tiempo determinado. Para lograr diseñar un motor de reconocimiento de voz que procese la información internamente en tiempo real (excluyendo los poderosos servidores disponibles en internet dedicados a tal fin), se debe garantizar el uso óptimo de los recursos disponibles, y por ello el sistema no debe invertir tiempo de procesamiento hasta asegurar que se produjo un intento de comunicación por parte del usuario.

Tanto la industria como la academia han invertido considerables esfuerzos en este campo de desarrollo a nivel de software y hardware, buscando una solución robusta que permita procesar la voz en tiempo real y usando como motor principal un sistema embebido de bajo costo. Sin embargo, es debido a la larga cantidad de fonemas y acentos que se pronuncian en cada lengua existente, que este campo sigue siendo hasta la fecha un área de investigación y desarrollo.

El reconocimiento de voz, especialmente el procesamiento por deletreo, encuentra en si un gran número de aplicaciones incluyendo algunos campos como el cuidado de la salud, la inteligencia artificial, la interacción optimizada entre el humano y la máquina, los sistemas interactivos con respuesta por voz, los dispositivos militares, entre otros. Generalmente el procesamiento de voz es visto como una herramienta poderosa de seguridad y comodidad que le da un plus de satisfacción a la tarea específica para la que ha sido asignada. Sin embargo, en este tipo de desarrollo reside la posibilidad de restaurar la calidad de vida de muchas personas que padecen diferentes tipos de limitaciones físicas, permitiendo que interactúen con el mundo de un modo más satisfactorio.

Al día de hoy existen varias alternativas para que las personas que padecen de algún tipo de discapacidad motriz recuperen parte de la movilidad ya sea a nivel local o general. Sin embargo, el costo de estas soluciones es muy elevado, lo que restringe el acceso exclusivamente a la clase alta. Logrando desarrollar un motor de reconocimiento de voz en un sistema embebido de bajo costo se puede garantizar que esta tecnología pueda llegar a más personas, incluso con la confiabilidad y seguridad que ofrece una FPGA en diseño de hardware y software.

Este documento describe el principio de un ASR, así como su implementación y desarrollo en una FPGA utilizando un procesador ARM como centro de cómputo, y arquitectura digital para acelerar el proceso y optimizar el uso de recursos. Incluye los objetivos planteados en la etapa inicial del proyecto, así como los desafíos que surgían con el desarrollo del mismo. Para el usuario inicial, se incluye una breve, pero relevante introducción al procesamiento de voz a manera de contextualización.

# 1 DEFINICIÓN DEL PROBLEMA

Actualmente, la silla de ruedas de la universidad cuenta con un analizador de espectros de tipo comercial (shield para arduino), que reconoce muestras personales de voz previamente grabadas, y ejecuta movimientos según el comando que sea pronunciado. Este tipo de sistemas son aptos para un proyecto de carácter demostrativo, más no son recomendados para que lleguen al usuario final, debido a la robustez del mismo.

Al tratarse de un analizador de espectros, exige al usuario grabar previamente su voz en el programa. Ignorando la necesidad de que el ingeniero realice este proceso cada vez que otra persona quiera acceder a la silla de ruedas, esta interfaz no garantiza la seguridad del usuario, pues no es robusto frente a las posibles perturbaciones que puedan afectar las muestras de audio, tales como el estado de animo de la persona, el color de la voz, el acento, entre otros. Una situación de riesgo se presenta cuando la persona que controla la silla siente temor y trata de comunicarle al sistema que se detenga. Espectralmente, la palabra pare pronunciada con o sin miedo es muy diferente, por tanto, el sistema no comprenderá que el usuario desea que el desplazamiento se interrumpa.

Bajo este contexto, se genera la necesidad de reemplazar este sistema comercial existente por un entorno completamente nuevo, diseñado e implementado en un sistema robusto como lo son las FPGA; un algoritmo capaz de realizar procesamiento de voz en tiempo real, que no requiera muestras pregrabadas, ni tampoco se vea afectado por las perturbaciones anteriormente mencionadas. Este algoritmo reemplazará al analizador de espectros y permitirá añadir esa capa extra de seguridad y fiabilidad que requiere el proyecto para convertirse en un producto que pueda llegar al mayor número de personas posible.

# 2 ANTECEDENTES

Una gran sección de la robótica y la inteligencia artificial aplicada, está orientada al desarrollo de herramientas que ayuden a recuperar la movilidad de personas que padecen de algún tipo de discapacidad. Según la OMS, más de mil millones de personas experimentan algún tipo de discapacidad, de las cuales 200 millones padecen de alguna limitación asociada a la motricidad.[1]

En todo el mundo, las personas con discapacidad tienen peores resultados sanitarios, peores resultados académicos, una menor participación económica que conlleva a tazas de pobreza más altas que las personas sin discapacidad.[2] En parte, ello es consecuencia de los obstáculos que entorpecen el acceso de las personas con discapacidad a servidos que muchos de nosotros consideramos obvios, en particular la salud, la educación, el empleo, el transporte o la información.

La discapacidad forma parte de la condición humana. Por sorprendente que parezca, casi todas las personas sufrirán un tipo de discapacidad transitoria o permanente en algún momento de su vida, y las que lleguen a la senilidad experimentaran dificultades crecientes de funcionamiento.

Para lograr retornar la movilidad a una persona que padece una condición de discapacidad motriz aguda deben realizarse muchas reformas en aspectos sociales, políticos y económicos. El modelo de infraestructura usado para la construcción y el desarrollo en las ciudades no es el adecuado, ya lo ha dejado en evidencia la conferencia mundial de discapacidad celebrada en la ciudad de Ginebra, Suiza.[3] Sin embargo, estos cambios demandan tiempo y recursos que a priori, tomarían años en desarrollarse.

La historia del estudio y desarrollo de un sistema apto para realizar reconocimiento de la voz humana se remonta a mediados del siglo XX, donde los laboratorios de AT&T (quienes recientemente habían adquirido Bell Telephone Company, empresa fundada por Alexander Graham Bell) desarrollaban un aparato primitivo que daba las primeras pautas de reconocimiento verbal.[4] Estaba claro que el éxito y la futura globalización de esta técnica se encontraba sujeta al grado de percepción de información verbal compleja, con alta precisión y constancia. Se tienen registros de una conversación entre un ser humano y un ayudante computarizado llamado “Annie”, el cual se trataba de un software que trataba de entender para responder a

[1] The world report on disability: 2011 update, Geneva, Suiza, World Health Organization, 2011. p. 5
[2] La educación inclusiva: el camino hacia el futuro. En: CONFERENCIA INTERNACIONAL DE EDUCACIÓN (48°: 2008: Ginebra). Presentación general de la 48° reunión de la CIE para la organización de las naciones unidas para la educación, la ciencia y la cultura. p. 10-14
[3] The world report on disability: 2011 update, Op. cit., p. 8
[4] CABEZAS, José Antonio. Alejandro Graham Bell: Vidas Ilustres. Barcelona, España : Susaeta Ediciones S.A, 1979. ISBN 84-305-1109-1. p. 18

la pregunta. Fue un fracaso dado su capacidad limitada para la comprensión de la voz humana.[5]

En el año 1960 los científicos de la época se enfocaron en desarrollar un sistema de reconocimiento de voz más complejo. Como primer paso desarrollaron un aparato que podía usar la conversación discreta, es decir, un estímulo verbal puntuado por pausas. Sin embargo, en 1970, es cuando realmente se desarrolló la tecnología de reconocimiento de voz que no requería que el usuario pausara sus palabras. Esta tecnología se convirtió en algo práctico y tangible en los años 80 y sigue siendo desarrollada y pulida hasta el día de hoy.

La ingeniería ha tomado cartas en el asunto a lo largo de la última década. En el año 2005, el instituto de investigación en ingeniería de la Universidad de Zaragoza, España, ha introducido un diseño de una silla de ruedas inteligente controlada por voz bajo el margen del primer congreso internacional de domótica, robótica y tele asistencia.[6] Aquel modelo contaba con un par de procesadores embebidos, un Pentium MMX 266 MHz con 64 MB de memoria RAM con sistema operativo en tiempo real VxWorks. El otro era un Pentium III 850 MHz con 256 MB de memoria RAM que corría Windows 2000. Se comunicaban a través de un puerto serial RS-232 y el uso compartido de una misma red Ethernet. Contaba a su vez con sensores de proximidad que evitaban que la silla de ruedas colisionara con algún objeto del entorno. Reconocía en total 10 palabras, 9 de ellas permitían ejecutar una acción de manejo y la última se usaba como activación.

En la elaboración de dicho reconocimiento de voz, se utilizaron modelos acústicos de entrenamiento. Fue creada una base de datos con repeticiones de las diferentes unidades existentes en la lengua castellana hasta obtener una representación estadística de la señal de voz para cada unidad (de las 10 palabras disponibles). Utiliza entonces un módulo reconocedor adaptado al locutor, es decir, emplea la representación más próxima a la persona que lo va a usar, por tanto, en la mayoría de los casos, la persona que quisiera usar la silla debía donar su voz al proyecto para ser caracterizada previamente.

Más adelante, en el año 2007, se realizó el 6to. Congreso nacional de mecatrónica en México donde la Universidad Politécnica de Pachuca presentó un proyecto similar de una silla de ruedas controlada por voz.[7] Este proyecto reconocía un total de 5 palabras, todas ellas para realizar alguna acción de desplazamiento en la silla de ruedas. Como elemento central de procesamiento, el proyecto contaba con un microprocesador integrado PIC 18X.

---

[5] Ibid., p. 35

[6] ALCUBIERRE, *et al.* Silla de ruedas inteligente controlada por voz. En: Primer congreso internacional de domótica, robótica y teleasistencia para todos. Madrid. 2005. p. 3

[7] PAREDES, Montoto, *et al*. Silla de ruedas controlada por voz. En: 6to congreso nacional de mecatrónica. Soledad de Draciano Sánchez, México. 2007. p. 186

El manejo de los motores se realizaba a partir de un puente H, que permite transformar la información binaria proveniente del sistema de procesamiento, en una señal análoga de tensión que permitiera controlar la velocidad y el sentido de giro.

Este reconocimiento de palabras partió del análisis de las señales de audio provenientes del locutor. Se usó el modelo matemático de la transformada STFT, definido como la proyección de la señal sobre una función de base sinusoidal limitada temporalmente a través de una ventana fija.[8] Finalmente recurre al HMM para interpretar estadísticamente la palabra pronunciada.

La facultad de ingeniería electrónica de la Universidad Pontificia Bolivariana, desarrolló en el año 2014 una silla de ruedas con motores eléctricos de corriente continua con la habilidad de ser manejada mediante comandos de voz.[9] Este proyecto se limita a analizar espectralmente sonidos previamente grabados. Para esta tarea se adquirió una tarjeta comercial EasyVR, en la cual se pueden grabar hasta 26 muestras de audio diferentes. Este módulo viene diseñado para implementación instantánea en Arduino, sin embargo, permite conexión a través de puerto UART con diferentes tipos de microcontroladores. La unidad central de procesamiento utilizada es un Arduino Mega.

Este proyecto evitaba colisiones a través de la detección de obstáculos, para ello usaba sensores de ultrasonido SRF02, los cuales estaban distribuidos de manera estratégica para monitorear las distancias periféricas. La comunicación entre los sensores y la tarjeta Arduino se realizó mediante protocolo I2C. La distancia mínima que puede detectar este tipo de sensores es de 15cm.

El manejo de la dirección y la velocidad del movimiento de los motores lo realiza el módulo RoboClaw 2x30A, el cual permite el control de 2 canales independientes, síncronos y regenerativos de 30A (corriente nominal) con una corriente pico de hasta 60A. Posee 4 modos de configuración: entrada RC, análogo, serial y paquetes seriales. Actualmente se encuentra configurado en el modo de entrada RC.

Actualmente se encuentra en fase de desarrollo (desde el año 2005) un prototipo de silla de ruedas controlada por voz que pertenece al Instituto Tecnológico de Massachusetts, el cual utiliza inteligencia artificial que aprende de su entorno a partir de los comandos de voz que el usuario le suministra. Para el reconocimiento de voz, usa el método MFCC, el cual permite identificar parámetros característicos de cada fonema en una escala que simula el comportamiento de la voz humana. Cuenta con

[8] DAUBENCHIES. The wavelet transform, time-frequency localization and signal analysis. En: IEEE Transactions on Information Theory. Septiembre, 1990. vol. 36, no. 5

[9] VARGAS, Oskar y SÁNCHEZ, Ruth. Manejo de los motores de una silla de ruedas eléctrica mediante comandos de voz. Floridablanca, Colombia, 2014, 61 h. Trabajo de grado (ingeniero electrónico). Universidad Pontificia Bolivariana. Facultad de Ingeniería Electrónica.

un sistema de redes neuronales (DNN) que se encarga de la identificación estadística de los fonemas. Más que reconocer comandos de voz, identifica un gran número de palabras dictadas en una misma oración, apoyándose en un extenso vocabulario. De ahí el título del proyecto: “Spoken Language Interaction with Model Uncertainty: An Adaptive Human-Robot Interaction System.[10]

Es más un algoritmo que es usado para optimizar el dialogo entra una silla de ruedas y un ser humano, a través de un medio de interacción dialectico.

[10] DOSHI, Finale y ROY, Nicholas. Spoken Language Interaction with Model Uncertainty: An Adaptive Human-Robot Interaction System. En: Connection Science Research. 2008. vol. 20.

# 3 JUSTIFICACIÓN

El proyecto tiene como prioridad reemplazar la tarjeta comercial que ejecuta el análisis espectral de las muestras de audio, por un motor de reconocimiento de voz en tiempo real, que permita a un amplio grupo de personas de habla hispana manejar la silla de ruedas con su propia voz sin necesidad de pre-grabar muestras y mediante una interacción mucho más humana, no mediante comandos específicos, sino a través de frases u oraciones complejas.

La tarjeta EasyVR tiene un costo aproximado de $200.000 pesos, sin embargo, posee una gran limitante en el número de configuraciones disponibles a nivel de comandos de voz pregrabados como la cantidad de instrucciones que el usuario puede añadir. Por otra parte, este módulo de instrucciones reducidas requiere de una constante comunicación con el software EasyVR commander mediante un computador cada vez que se requiera su funcionamiento lo cual no es cómodo, practico, ni eficiente, sin mencionar que resta movilidad y autonomía a la silla de ruedas.

Hoy en día las FPGA son ampliamente conocidas e implementadas para diferentes procesos y diseños tanto en las áreas de control, como en instrumentación o las comunicaciones. Resultan ser una herramienta muy eficiente y robusta gracias a la capacidad de ejecutar tareas en paralelo, y a la posibilidad de realizar grandes cambios a nivel de Hardware con solo reprogramar el módulo de desarrollo. Tener un sistema que permite hacer mucho más en cada ciclo de reloj, con la fiabilidad que permite el diseño en hardware, es la solución ideal para tareas que demandan un gran rendimiento, como lo es el procesamiento en tiempo real.

La tecnología Cyclone V resalta por la combinación de un procesador ARM cortex-A9 embebido (HPS) en el mismo chip del diseño de campo FPGA. Ofrece la posibilidad de correr un sistema operativo en el procesador embebido sin necesidad de crear un módulo NIOS, dejando el trabajo secuencial para el procesador y promoviendo el uso de hardware para aceleración del sistema.

Cabe resaltar la importancia del uso de las nuevas tecnologías ya que demuestra un conocimiento que se encuentra a la vanguardia y promueve nuevos proyectos y aplicaciones basados en este tipo de arquitectura. Se pretende así que la silla de ruedas pase a ser un proyecto competitivo con propiedad intelectual de la Universidad Pontificia Bolivariana.

Existe un tipo de reconocimiento de voz ajeno a la mayoría de módulos comerciales pre-diseñados actualmente disponibles conocido como “speech recognition” o reconocimiento por deletreo, el cual es un método de detección de caracteres o palabras dentro una oración, sin necesidad de pausas, solo dictado natural. Este procesamiento requiere un mayor número de recursos pues generalmente usa

análisis de patrones a través de los MFCC, acompañado de una base de datos de fonemas del lenguaje y un método estadístico que permita la identificación de las palabras pronunciadas.

Con esto se pretende mostrar que la implementación sobre la silla de ruedas es solamente una aplicación, que no limita su uso únicamente a ello, pero que si demuestra lo que una herramienta como esta puede hacer, ya que su fácil y practica modificación permite utilizarla como elemento para el control del proceso o la aplicación que lo requiera; Se pretende desarrollar no solo un sistema de manejo por voz de una silla de ruedas, sino un prototipo capaz de ser acoplado a cualquier entorno donde se necesite su servicio para el mejoramiento y la optimización de las funciones.

# 4 OBJETIVOS

## 4.1 OBJETIVO GENERAL

Diseñar e implementar un algoritmo de reconocimiento de voz, para el manejo de los motores eléctricos de una silla de ruedas, sobre un sistema embebido en FPGA.

## 4.2 OBJETIVOS ESPECÍFICOS

- Realizar una revisión bibliográfica sobre los diferentes métodos alternativos para la elaboración de un sistema de reconocimiento de voz.

- Programar un algoritmo dedicado al procesamiento digital de voz, para la ejecución de las tareas de movimiento en la silla de ruedas.

- Implementar el algoritmo de reconocimiento de voz, en un sistema embebido de tipo abierto, que permita innovar en el ámbito del procesamiento de audio con propiedad intelectual de la Universidad Pontificia Bolivariana.

- Acoplar el sistema de motores existente en la silla de ruedas y añadir sensores de proximidad, para evitar colisiones en la trayectoria.

- Realizar pruebas en la comunicación del módulo de manejo con el módulo ROBOCLAW y garantizar el correcto funcionamiento de las rutinas de operación incorporadas en la silla de ruedas.

# 5 MARCO TEÓRICO

## 5.1 DISCAPACIDAD MOTRIZ

La discapacidad motriz se presenta de manera transitoria o permanente alterando el aparato motor, debido a un deficiente funcionamiento en el sistema nervioso, muscular y/o óseo-articular; limitando alguna de las actividades que se puedan realizar.

Las personas con discapacidad motora presentan características clínicas y funcionales diversas. Su discapacidad puede ser de origen congénito, por causas adquiridas o debido a enfermedades degenerativas. La mayoría de estas afectaciones dificultan o imposibilitan la movilidad funcional en una o varias partes del cuerpo. Este es el aspecto más significativo, ya que de manera prioritaria se tienen dificultades en la ejecución de sus movimientos o ausencia de los mismos.

### 5.1.1 Clasificación

La deficiencia motora, comprende una gran variedad de situaciones que permiten describir y clasificar atendiendo a 3 principales grupos: momento de aparición, etiología y localización topográfica.

#### *5.1.1.1 Momento de aparición*

- Prenatal: Se da como consecuencia de una malformación congénita, es decir, antes del nacimiento, en el período de formación del feto.
- Perinatal: Ocurren por malos hábitos durante la primera etapa de la vida, como posturas inadecuadas, o trastornos debido a lesiones en un cerebro inmaduro como la enfermedad motora de origen cerebral (EMOC).
- Postnatal: Toda aquella malformación que sucede por algún suceso particular, que generalmente está asociado a traumatismos, distrofias, tumores, entre otros.

#### *5.1.1.2 Etiología*

- Transmisión genética: Generalmente a nivel del ADN, son producto de un cromosoma genéticamente imperfecto que ha sido heredado de los padres.
- Infecciones microbianas: Discapacidad que se da a partir de un ataque microbiano, ya sea que comprometa la motricidad a nivel encefálico, óseo o muscular.
- Traumatismos: Los traumatismos son situaciones de daño físico al cuerpo. Por lo general involucran heridas serias que ponen en riesgo la vida causando en algunas ocasiones complicaciones secundarias de tipo motriz.

*5.1.1.3 Localización topográfica*

En este grupo se clasifica la imposibilidad de movimiento según una zona específica del cuerpo. Comúnmente se les conoce bajo el nombre de parálisis.

- Monoplejía: Parálisis que afecta a un solo miembro del cuerpo, ya sea superior o inferior.
- Hemiplejía: Parálisis que afecta a un lado del cuerpo, izquierdo o derecho.
- Paraplejía: Parálisis de los dos miembros inferiores del cuerpo. Se debe a lesiones nerviosas en el cerebro o en la médula espinal.
- Cuadriplejía: Parálisis de los brazos, manos, tronco, piernas y órganos pélvicos del cuerpo. Causada por un daño en la médula espinal.

### 5.1.2 Sistema nervioso central

El sistema nervioso central está constituido por dos estructuras: el encéfalo y la médula espinal, ambos se encuentran rodeados por tres capas de membrana denominadas meninges, entre dos de estas capas se sitúa el líquido cefalorraquídeo.

El encéfalo y la médula espinal son los encargados de controlar todas las funciones del organismo, ya sean voluntarias o involuntarias. El encéfalo está integrado por tres estructuras: el cerebro, el cerebelo y el tronco cerebral. Está rodeado por una estructura ósea rígida que se conoce como el cráneo.

*5.1.2.1 Cerebro*

Es la parte más voluminosa del encéfalo y se encuentra situado en el interior del cráneo. Anatómicamente está dividido en dos hemisferios, derecho e izquierdo por un surco central llamado cisura longitudinal. La superficie de cada hemisferio presenta un conjunto de pliegues, que forman depresiones irregulares denominados surcos o cisuras. Cada hemisferio se divide en 4 lóbulos: frontal, temporal, parietal y occipital. El cerebro se encarga de las funciones superiores del ser humano como las capacidades cognitivas (el aprendizaje, la memoria, la conciencia, la imaginación, el pensamiento, etc.) y ciertas respuestas motrices y emocionales.

- Lóbulo occipital:
  Situado en la parte posterior del encéfalo. En él se reciben y analizan las informaciones visuales.
- Lóbulo temporal:
  Interviene en la memoria, el lenguaje y las sensaciones auditivas.
- Lóbulo frontal:
  Situado en la parte anterior del encéfalo. Interviene en las características de la personalidad, la inteligencia, el lenguaje, la escritura, y los movimientos voluntarios.

- Lóbulo parietal:
  Interviene en la identificación de objetos y las relaciones espaciales. Asimismo, se asocia con la interpretación del dolor y el tacto.

*5.1.2.2 Cerebelo*

El cerebelo está situado en la parte posterior del cráneo. Su función es coordinar los movimientos musculares y mantener la postura, la estabilidad y el equilibrio. Estas funciones se regulan de manera automática, es decir, la persona no tiene control voluntario sobre las mismas.

*5.1.2.3 Tronco cerebral*

El tronco del encéfalo está situado en la línea media del cerebro y conecta éste con la médula espinal. En esta zona están situados los centros que controlas las funciones vitales como la respiración, la tos, el ritmo cardiaco, la tensión arterial, la temperatura corporal y la deglución. Otras funciones que controla el tronco cerebral son el movimiento de los ojos y de la boca, la transmisión de los mensajes sensoriales como calor, dolor, ruidos, etc.

*5.1.2.4 Médula espinal*

La médula espinal es un cordón constituido por fibras nerviosas situado en el interior de la columna vertebral. Su función más importante es conducir, mediante las vías nerviosas que la forman, la información de las sensaciones desde todo el organismo hasta el cerebro y los impulsos nerviosos que llevan las respuestas, desde el cerebro a los músculos.

### 5.1.3 Tipos de apoyo a la discapacidad motriz

El apoyo a la discapacidad busca restaurar en la persona la independencia para llevar una vida más plena y satisfactoria que permita un desarrollo personal y una integración social, educativa o laboral. Estos apoyos son concebidos a partir de la intensidad de la intervención, la condición a tratar y el proyecto de vida. El apoyo brindado puede ser a nivel pedagógico, terapéutico e incluso tecnológico.

*5.1.3.1 Apoyo pedagógico:*

Debido al modelo de enseñanza planteado a nivel mundial, el aprendizaje para las personas con algún tipo de discapacidad puede llegar a verse comprometido, hasta el punto de perder totalmente el acceso a educación y formación personal. El apoyo pedagógico busca reintegrar a las personas que padecen discapacidad por medio de la educación y la capacitación en áreas donde su condición no le impida ejercer y prestar un servicio a la sociedad.

El apoyo pedagógico debe estar supervisado por un psicólogo pues se debe combatir los niveles de frustración e impotencia que se suelen desarrollar en las personas con capacidades motrices limitadas.

*5.1.3.2 Apoyo terapéutico:*
La discapacidad motriz causada por traumas suele ser la condición que más se beneficia de este tipo de apoyo que está ligado al campo médico y farmacológico. Debido a que el causante de la limitación de movilidad se debió a una alteración por maltrato físico o herida seria, es posible tratar el área afectada por el trauma para intentar la recuperación parcial o total de la movilidad en la zona afectada.
Sin embargo, existe la posibilidad de incentivar a través de terapia física tanto al sistema muscular como al sistema central.

*5.1.3.3 Apoyo tecnológico:*
La reintegración total de una persona con discapacidad motriz sigue siendo utópica aun con el desarrollo tecnológico actual. Sin embargo, si se puede llevar la calidad de vida a un nivel satisfactorio mediante el uso de las nuevas tecnologías que buscan principalmente restaurar la movilidad y el control de entornos.
Existen sistemas que mejoran la movilidad personal, como brazos artificiales, soportes articulados y conmutadores adosados a sillas de ruedas. Todos ellos cuentan con el respaldo de la tecnología como motor de funcionamiento principal.

También se busca restituir en la persona la capacidad de controlar su entorno externo, es decir, el control sobre dispositivos de uso doméstico. En este campo se elaboran sistemas con reconocimiento de voz, que permitan restaurar la comunicación y la manipulación mediante habilidades que posea la persona como el uso de su propia voz.

Respecto al desplazamiento, la barrera más grande con la que se debe tratar es de orden arquitectónico, pues se requieren más rampas, elevadores y carriles exclusivos para el acceso de sillas de ruedas controladas electrónicamente.

## 5.2 PROCESAMIENTO DIGITAL DE VOZ

El análisis de la señal de voz y su posterior reconocimiento deben superar algunos problemas que en principio parecen triviales ya que son superados de forma sencilla por los seres humanos, algunos de ellos son, la correcta elección y extracción de las características de la señal de voz, tratar con corrección las variaciones inherentes a género, velocidad de emisión, pronunciación y acentos, tamaños de los vocabularios a reconocer, ruido y distorsión de los entornos donde se utilizan, inclusive hasta el estado de ánimo del locutor.

### 5.2.1 Conceptos básicos

En la literatura del procesamiento digital de voz, existen algunos conceptos básicos para la comprensión del modelo completo. Algunos de ellos se listan a continuación:

- Fonema: es la unidad lingüística más pequeña con capacidad de diferenciar significados[11], también interpretada como la articulación mínima de un sonido vocálico o consonántico. Son unidades teóricas básicas postuladas para estudiar el nivel fonológico de una lengua humana.[12]

- Fono: también conocido como un sonido lingüístico, está compuesto por una serie de rasgos fonéticos y articulatorios. Es la manifestación acústica concreta de un fonema. "sonido del habla".[13]

- Rasgo Distintivo: propiedad acústica o articulatoria de los segmentos fónicos lingüísticamente relevante y que puede presentar dos o más posibles valores contrastivos.[14]

- Dictado automático: es una vertiente de los sistemas de reconocimiento digital de voz, que permite identificar varias palabras que el usuario proporciona por medio de un deletreo. Es una interpretación computacional que se le da al habla de la forma más natural posible.

- Codebook: base de datos que usa un motor de reconocimiento de voz, donde se encuentran almacenados los vectores cuantificados de información de entrenamiento sobre los símbolos referentes a cada fonema en el lenguaje deseado.

- HMM: modelo de decisión estadístico de inteligencia artificial basado en aprendizaje y entrenamiento por ramificaciones sintetizadas de forma paramétrica.[15] Actualmente el más utilizado en sistemas de reconocimiento de voz.

### 5.2.2 Muestreo y cuantificación

Según Nyquist sabemos que es necesaria una frecuencia de muestreo de por lo menos el doble del ancho de banda de la señal a caracterizar, sobre esta base y para un análisis mínimo (en lo que respecta a frecuencia) de la señal de voz se utiliza una frecuencia de muestreo $F_s$ de 8khz, aunque se suele usar 16khz si se desea obtener mayor detalle en frecuencia lo que mejora la resolución para tratamiento de la señal. La cuantificación más comúnmente usada, es de 8 bits,

[11] FERNÁNDEZ, Juana. Fonética para profesores de español: de la teoría a la práctica. Madrid: Arco/Libros, 2007. 614 p. ISBN 847-63-5645-5. p. 540

[12] GUIRAO, M y JURADO, M. A. G. Estudio estadístico del español. En: Laboratorio de investigaciones sensoriales CONICET. Buenos Aires, 1993.

[13] FERNÁNDEZ, Op. cit., p 541

[14] FERNÁNDEZ, Op. cit., p 544-545

[15] WATTS, Oliver, *et al.* From HMMs to DNNs : where do the improvements come from?. En: Centre for speech technology research. [en Linea] <http://homepages.inf.ed.ac.uk/ghenter/pubs/watts2016hmms.pdf> [Citado en: Septiembre, 2016].

mínimo requerido para una calidad baja, puede mejorarse su S/R con una técnica no lineal de cuantificación, se obtienen excelentes resultados aumentando la cuantificación a 16 bits.

## 5.3 SPHINX

El sistema SPHINX fue desarrollado en la CMU (Carnegie Mellon University) por Lee *et al*.[16] Es el entorno pionero en reconocimiento de voz por deletreo continuo de extenso vocabulario, y es considerado actualmente como el sistema más preciso en este aspecto. A continuación, se describirán brevemente los diferentes bloques que componen este sistema, haciendo énfasis en el procesamiento de señales. El diagrama de bloques que muestra las etapas de SPHINX se muestran en la Figura *1*.

### 5.3.1 Procesamiento de señales

Todos los sistemas de reconocimiento de voz utilizan una representación paramétrica en lugar de la forma de onda. Por lo general, los parámetros llevan la información en la envolvente del espectro. SPHINX utiliza paquetes de frecuencia cepstrales LPC (linear predictive coding) como su conjunto principal de parámetros, los cuales se calculan teniendo en cuenta las siguientes condiciones de funcionamiento:

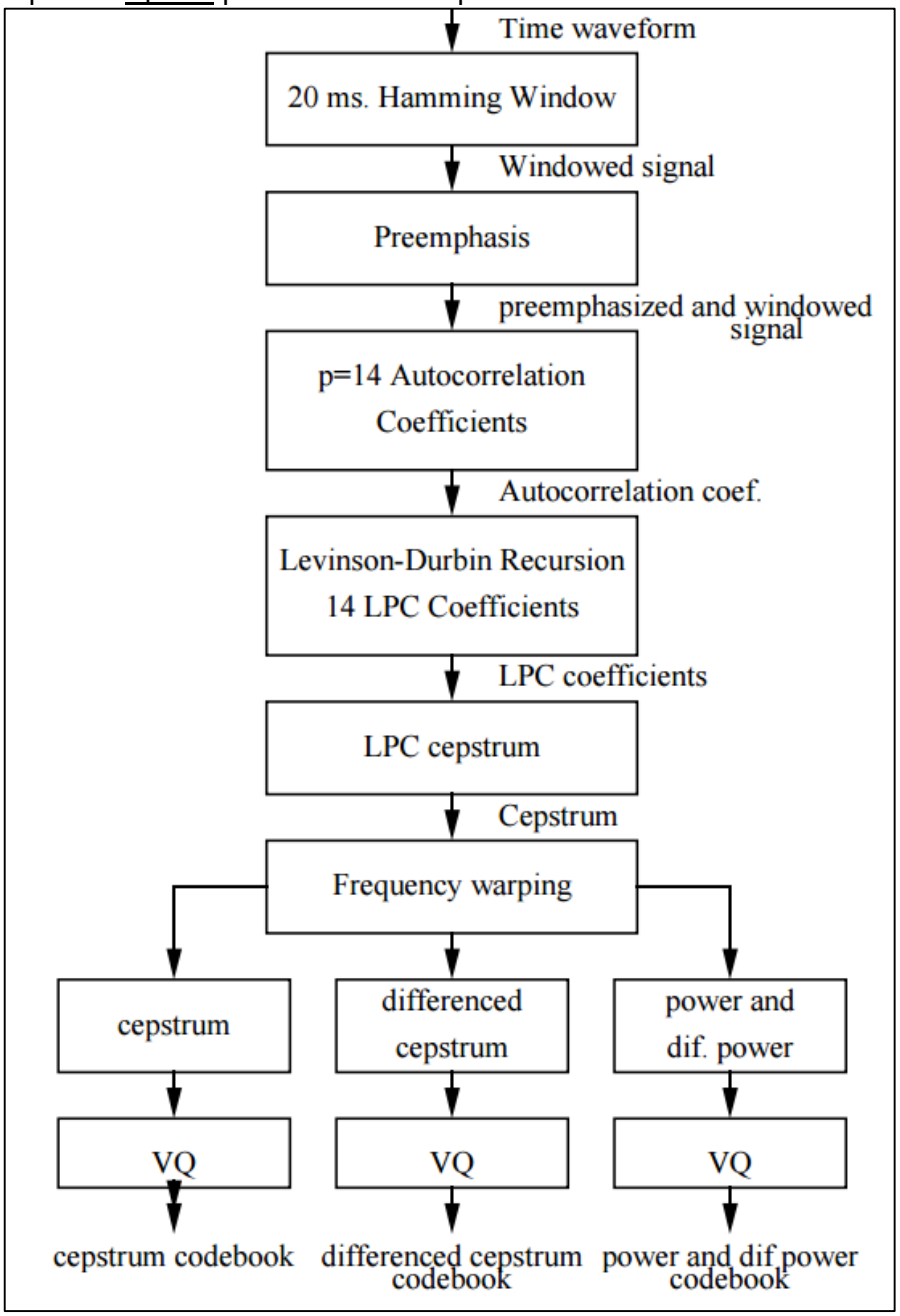
- El deletreo es digitalizado a una frecuencia de muestreo de 16 KHz.
- Una ventana Hamming de 320 muestras es utilizada cada 10 ms.
- Se aplica un filtrado de pre-énfasis $H(z) = 1 - 0.97\, z^{-1}$
- 14 coeficientes de auto correlación son calculados.
- 14 coeficientes LPC son derivados a partir del algoritmo de Levinson.
- 32 coeficientes cepstrales son calculados usando el algoritmo de recursión.
- Los coeficientes cepstrales son modificados en el dominio de la frecuencia usando la transformación bilineal para obtener 12 coeficientes cepstrales LPC en frecuencia.

Aunque en procesamiento de señales, los cuadros de muestras están correlacionados entre sí, el sistema SPHINX asume que cada uno de ellos es estadísticamente independiente de los demás.

Además de la información estática proporcionada por la escala Cepstral de MEL, SPHINX también utiliza la información dinámica representada por la diferencia entre los vectores cepstrales de primer orden: $d_i = x_{i+2} - x_{i-2}$

[16] LEE, Kai Fu, *et al*. Large-vocabulary speaker-independent continuous speech recognition: The SPHINX System. Pittsburgh, 1988. 289 h. Ph.D. Thesis. Carnegie Mellon University.

Figura 1. Etapas de Sphinx para el análisis cepstral de una señal de audio.




Fuente: ACERO, Alejandro. Acoustical and Environmental Robustness in Automatic Speech Recognition. Pittsburgh, 1990. 141 h. Ph.D. Thesis. Carnegie Mellon University. p. 17.

### 5.3.2 Cuantificación vectorial

Cuantificación vectorial (VQ) es una técnica de reducción de datos que mapea un vector real en un conjunto más pequeño de valores discretos. A pesar de que este método fue propuesto originalmente para la codificación de voz, recientemente se

ha ganado una gran popularidad en los sistemas de reconocimiento de voz. Un cuantificador vectorial está definido por un codebook y una medida de distorsión:

- El codebook es un alfabeto discreto que contiene L cantidad de vectores, y es una representación cuantificada del espacio vectorial.
- La medida de distorsión calcula el grado de proximidad de dos vectores. Un vector de entrada es mapeado a un símbolo de este alfabeto eligiendo el codebook más cercano. En SPHINX la medida de distorsión usada es la distancia Euclidiana.

SPHINX utiliza tres codebooks diferentes: uno para el cepstrum, otro para el primer termino de diferencia de los vectores cepstrales y el último para el primer termino de diferencia de la potencia.[17] Cada codebook contiene 256 vectores. Cada trama de la señal de voz es comprimida en 3 bytes. La medida de distorsión utilizada es la distancia Euclidiana. Los vectores prototipo se estiman a través de un algoritmo de agrupamiento jerárquico similar al algoritmo Keas desarrollado por Linde *et al*[18], el cual es un método aproximado de máxima verosimilitud.

### 5.3.3 Modelo oculto de Markov

HMM, la tecnología que predomina actualmente en la constitución de los sistemas de reconocimiento de voz continuos, compone el motor de reconocimiento utilizado en SPHINX.

Un HMM es un proceso doblemente estocástico, en donde existe un primer proceso estocástico no observable (que se encuentra oculto), pero puede ser observado a través de otro conjunto de procesos estocásticos que producen una secuencia de símbolos observados.[19]

Para entender el concepto del HMM, Rabines y Juang[20] consideran el siguiente ejemplo simplificado. Un observador se encuentra en una habitación con un bloqueo visual en frente (por ejemplo una cortina), que impide apreciar lo que está ocurriendo. Mientras tanto, en el otro lado de la barrera se encuentra una persona realizando un experimento de lanzamiento de una moneda (o múltiples monedas). Esa otra persona no le dirá nada sobre lo que está haciendo exactamente; solo le dirá el resultado del lanzamiento de cada moneda. Por lo tanto una secuencia

[17] ACERO, Alejandro. Acoustical and Environmental Robustness in Automatic Speech Recognition. Pittsburgh, 1990. 141 h. Ph.D. Thesis. Carnegie Mellon University. p. 18
[18] LINDE, Y, *et al*. An algorithm for Vector Quantizer Design. En: IEEE Transactions on Communication. vol. COM-28, no. 1, p. 84-95. Enero, 1980.
[19] PICONE, Joseph. Continuous Speech Recognition Using Hidden Markov Models. En: IEEE ASSP Magazine. Julio, 1990. p. 32
[20] L.R. Rabiner y B.H. Juang. An introduction to hidden Markov models. En: IEEE ASSP Magazine. vol. 3, no. 1, p. 4-16. Enero, 1986.

resultante de un experimento oculto se lleva a cabo y el observador solo tiene acceso a los resultados:

$$O = \chi\,\chi\,\tau\,\chi\,\tau\,\tau\,\chi\,\chi\,\ldots\,\tau \quad (1)$$

$$O_1 O_2 O_3 \ldots\ldots\ldots\ldots\ldots\ldots O_\tau \quad (2)$$

Donde $\chi$ se refiere a cara y $\tau$ a sello.

Dado el anterior experimento, el problema es cómo construir un HMM para explicar las secuencias de cara y sello observadas. Uno de los posibles modelos se muestra en la *Figura 2-A*. Se conoce como el modelo de una moneda imparcial. Hay dos estados en este modelo, pero cada uno de ellos está asociado únicamente a cualquiera de las 2 posibilidades, cara (estado 1) o sello (estado 2). Por esta razón, este modelo no es oculto porque la secuencia que se observa define de forma exclusiva el estado. El modelo representa a una moneda imparcial porque la probabilidad de obtener cara (o sello) después de una cara (o sello) es de 0.5; por lo tanto no existe un sesgo en esta observación. Este ejemplo sin embargo muestra como una muestra de pruebas independientes, como lanzar una moneda, se puede interpretar como un conjunto de eventos secuenciales.

Un segundo HMM posible para explicar la secuencia observada de los resultados del lanzamiento de la moneda se observa en la *Figura 2-B*. Se conoce como el modelo de dos monedas imparciales. Existen dos estados en este modelo, pero ninguno está asociado únicamente a cara o sello. La probabilidad de obtener cara (o sello) en cualquiera de los dos estados es de 0.5. También la probabilidad de abandonar (o permanecer) en cualquiera de los dos estados es de 0.5. Así, en este caso, se asocia cada estado con una moneda independiente. Se dice que este modelo es oculto pues las secuencias de datos que se obtienen son independientes de la transición de los estados, es decir, no se puede saber el dato que se obtiene en la cadena a cuál de las monedas corresponde.

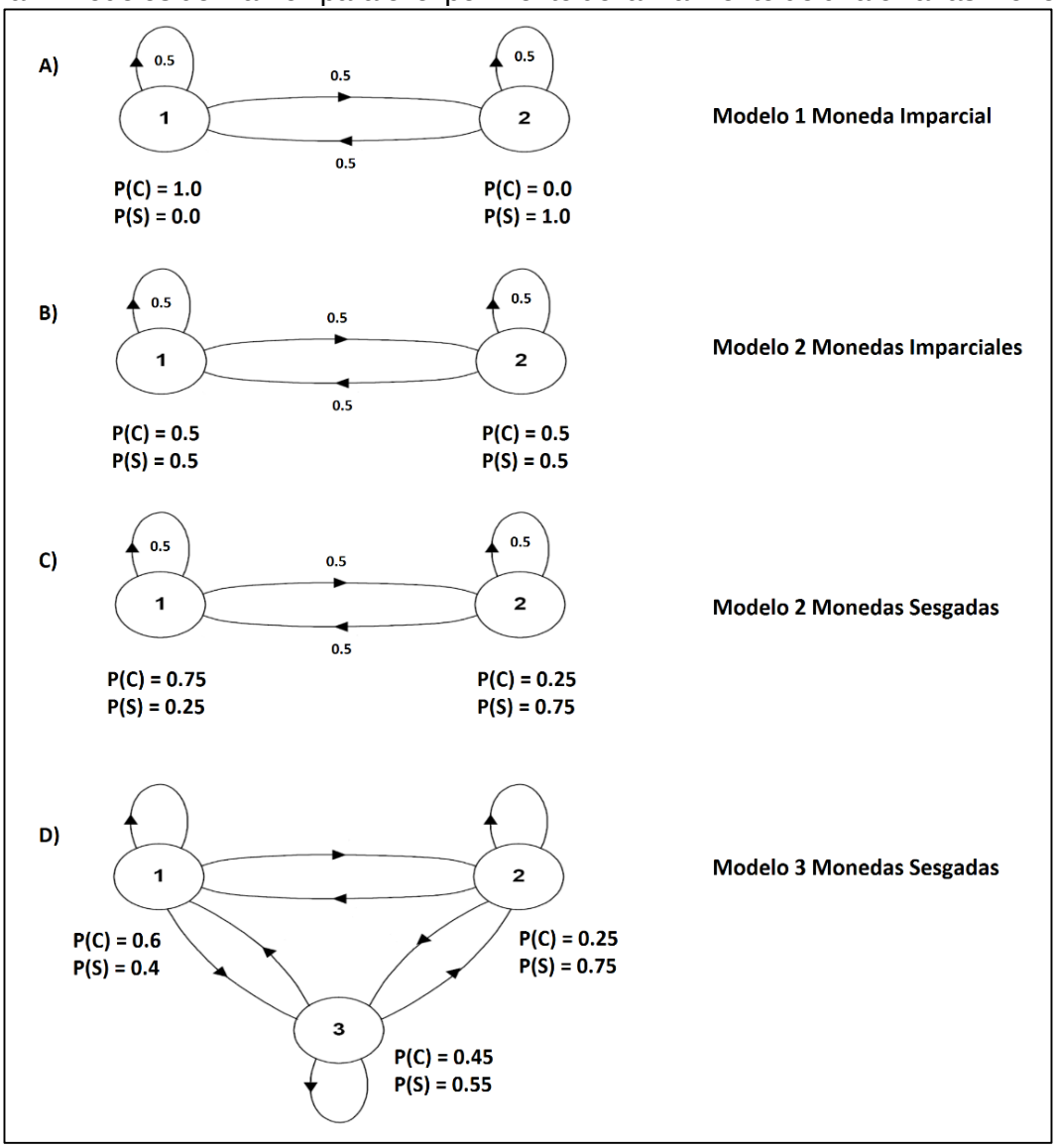
Las *Figura 2-C* y *Figura 2-D* muestran otros dos posibles modelos de HMM. La figura 2-C muestra el modelo de las dos monedas sesgadas, compuesto por dos estados (correspondiente a dos diferentes monedas). En el estado 1 la moneda está fuertemente sesgada hacia las caras, mientras en el estado 2 la moneda está fuertemente sesgada a los sellos. Muestra un modelo oculto de Markov donde en realidad se usan 3 monedas. Dos de ellas fuertemente sesgadas inclinadas hacia cara o sello y la tercera moneda es lanzada para decidir cuál de las dos monedas sesgadas se va a utilizar en cada lanzamiento.

El modelo de la *Figura 2-D*, llamado modelo de las 3 monedas sesgadas, está compuesto por tres estados (correspondientes a tres monedas diferentes). La moneda 1 esta sesgada hacia la cara, las monedas 2 y 3 hacia el sello. Cada una sesgada con un valor diferente y desconocido para el observador. Sin embargo en

este caso es donde se ejemplifica mejor el modelo oculto de Markov. El observador no conoce la cantidad de monedas, tampoco el orden en el que son lanzadas estas monedas, solo obtiene la información de los resultados en general de las monedas lanzadas.

Es por eso que en el momento de diseñar un HMM, una de las condiciones más difíciles de inferir es el número de estados del cual va a estar compuesto el modelo oculto. Si no se tiene una información inicial, el proceso de selección del número de estados se convierte en ensayo y error hasta encontrar el tamaño más apropiado para el modelo.

Figura 2. Modelos de Markov para el experimento de lanzamiento de una o varias monedas.




Fuente: L.R. Rabiner y B.H. Juang. An introduction to hidden Markov models. En: IEEE ASSP Magazine. Vol. 3, no. 1, p. 4-16. Enero, 1986. p. 3.

Para estimar las probabilidades en el modelo oculto de Markov, existe el procedimiento conocido como adelante-atrás y está basado en el hecho de que las probabilidades de los diferentes símbolos se calculan como la suma de las probabilidades de todos los caminos que podrían haber producido la secuencia de observación. Picone[21] plantea el cálculo del procedimiento adelante atrás resumido de la siguiente manera:

$$P\left\{\frac{O}{G}\right\} = \sum_{i=1}^{N} \alpha_T(i) \quad (3)$$

Donde

$1 \leq i \leq N$

$$\alpha_1(i) = \pi_i b_i(O_1) \quad (4)$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \quad (5)$$

Donde

$$t = 1,2,\ldots,T-1 \qquad 1 \leq j \leq N \quad (6)$$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad (7)$$

Donde

$1 \leq i \leq N \qquad \beta_T(i) = 1$

$$t = T-1, T-2, \ldots, 1 \qquad 1 \leq j \leq N \quad (8)$$

El aspecto más significativo de los HMM en relación con el procesamiento de voz, según Rabiner[22] plantea en su publicación sobre aplicaciones del HMM, es la existencia de un procedimiento iterativo en el que los parámetros de un modelo oculto de Markov pueden ser ajustados para representar mejor las estadísticas de una base de datos de entrenamiento.

Así mismo, Picone[23] resalta que el método de Baum-Welch se basa en el principio de máxima probabilidad. Un nuevo modelo es calculado garantizando la mejora de la probabilidad de la secuencia observada. El procedimiento de Baum-Welch se puede resumir de la siguiente manera:

$\pi_1 =$ expected no. of times in $s_i$ at $t = 0$

[21] PICONE, Joseph. Op. cit., p. 33
[22] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. En: Proceeding of the IEEE. vol.77, no. 2, p. 257-285. Febrero, 1989.
[23] PICONE, Joseph. Op. cit., p. 33

$$\pi_1 = \alpha_1(i)\beta_1(i) \tag{9}$$

$$a_{ij} = \frac{\text{expected no.of transitions from } s_i \text{ to } s_j}{\text{expected no.of transitions from } s_i} \tag{10}$$

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)} \tag{11}$$

$$b_j(k) = \frac{\text{expected no.of times in } s_j \text{ and observed symbol } v_k}{\text{expected no.of times in } s_j} \tag{12}$$

$$b_j(k) = \frac{\sum_{\substack{t=1 \\ ot=v_k}}^{T-1} \alpha_t(i)\beta_t(i)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)} \tag{13}$$

### 5.3.4 Unidades de habla

Aunque para los seres humanos la unidad básica de una frase deletreada es una palabra, en un sistema de reconocimiento de voz no es viable plantear cada palabra como la unidad básica del sistema. Esto debido a la gran cantidad de palabras que se pueden encontrar en cada lenguaje, sin mencionar que se requeriría de numerosas repeticiones de cada una de ellas para realizar el entrenamiento adecuado del sistema.

Alejandro Acero[24] sugiere que en los sistemas de reconocimiento de voz es mucho más sencillo interpretar y entender fonemas que palabras debido a la cantidad existente de estos en un lenguaje (unos 26 fonemas componen el idioma español) y a su facilidad para ser entrenados en el motor de reconocimiento. Típicamente cada fonema está definido como un modelo HMM de tres estados.

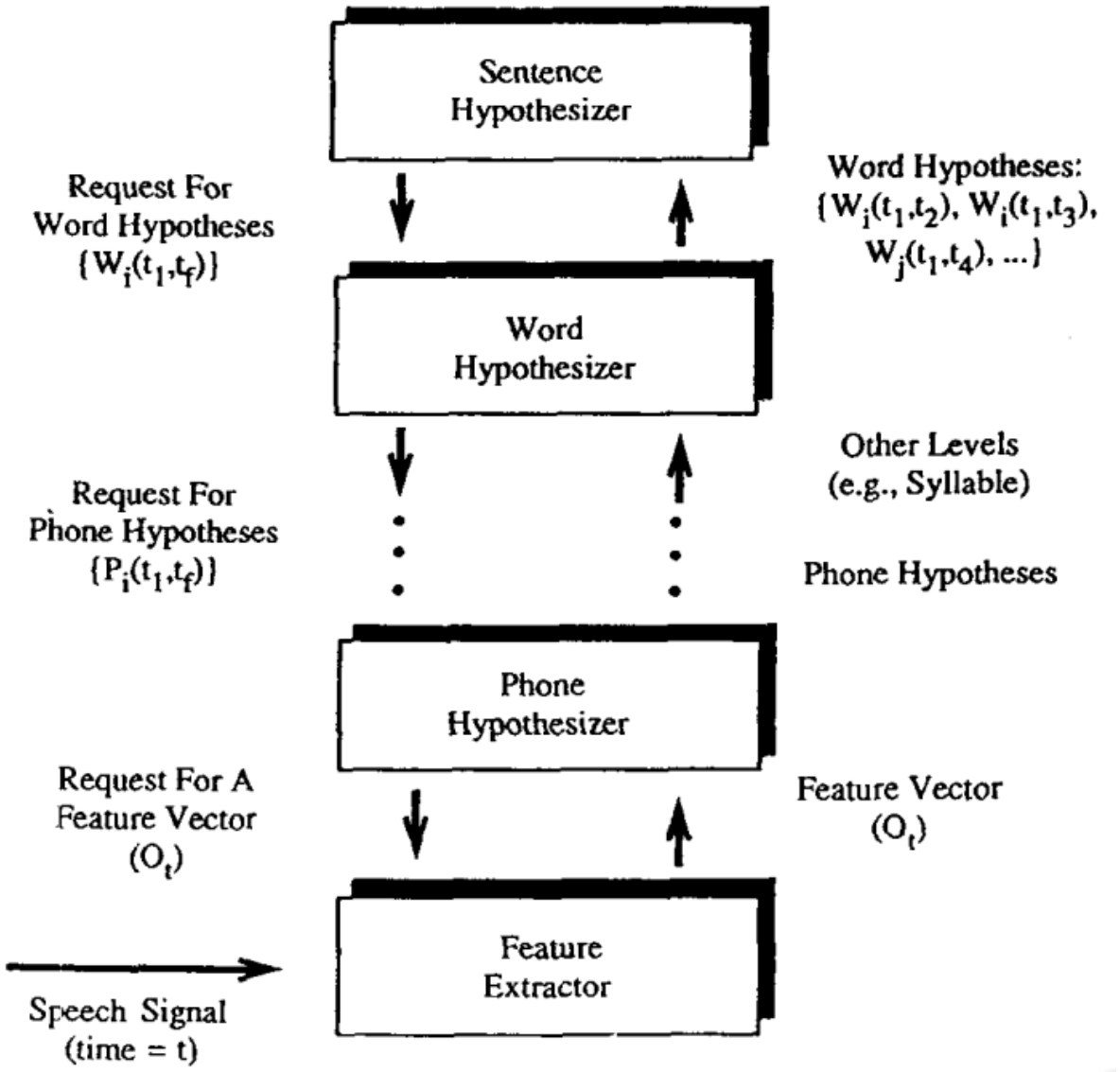
Cada fonema usado en diferentes contextos puede ser estructuralmente muy diferente. SPHINX[25] usa subgrupos generalizados compuestos por tres fonemas, para acelerar el proceso de reconocimiento. Por otra parte, la identificación de una palabra sucede en el proceso de concatenación sobre los modelos apropiados de los tri-fonemas. Finalmente, el modelo de identificación de la expresión o frase completa se logra a partir de la concatenación de los modelos de palabras comprendidas.

Como se aprecia en la *Figura 3*, el funcionamiento del motor de reconocimiento de voz siempre va componiendo el resultado de menos a más, de la unidad de reconocimiento mínima, a la expresión completa. Cada proceso del sistema puede ser representado como un modelo de Markov. La mayoría de sistemas actuales utilizan dos niveles de reconocimiento cuando usan unidades de reconocimiento de

[24] ACERO, Alejandro, Op. cit., p. 19
[25] LEE, Kai Fu, Op. cit., p. 79

palabras enteras. SPHINX usa tres niveles incluyendo la unidad de reconocimiento de fonos y fonemas. La inclusión de una arquitectura jerárquica facilita el proceso de entrenamiento del ASR.

Figura 3. Arquitectura jerárquica para el reconocimiento de voz continuo basado en procesamiento top-down.

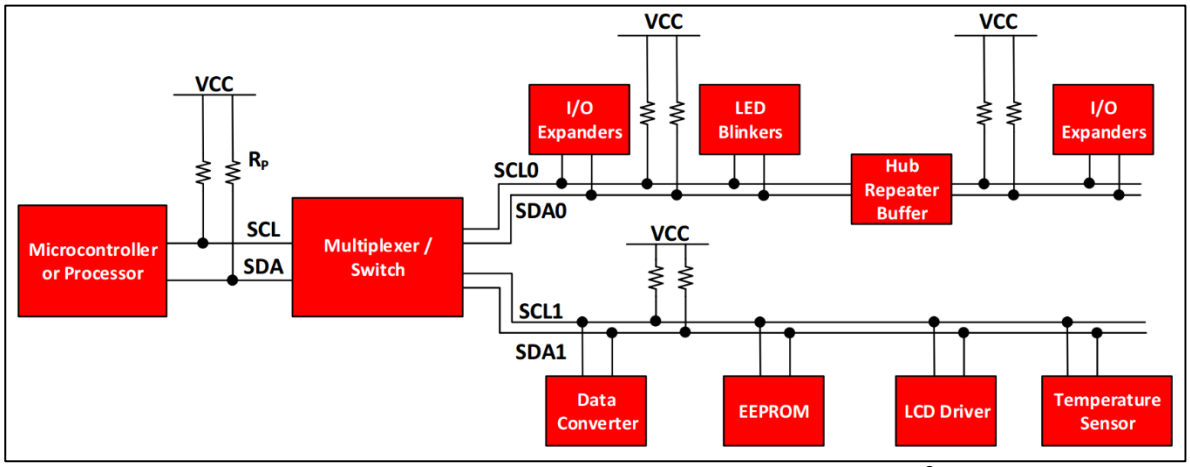



Fuente: PICONE, Joseph. Continuous Speech Recognition Using Hidden Markov Models. En: IEEE ASSP Magazine. Julio, 1990. p. 28

## 5.4 PROTOCOLO I2C

El circuito inter-integrado, más conocido como protocolo $I^2C$ (por sus siglas en inglés), es uno de los buses más populares y eficientes utilizados a nivel industrial para la comunicación entre un maestro (o múltiples maestros) y uno o varios dispositivos esclavos. En la *Figura 4* se observa como varios dispositivos diferentes comparten un bus $I^2C$ conectado a un procesador (maestro). El protocolo requiere únicamente de 2 cables para su funcionamiento, lo cual es uno de los principales beneficios que ofrece la comunicación $I^2C$ en comparación con otras interfaces similares. El siguiente análisis eléctrico sobre el protocolo $I^2C$, se encuentra basado en el documento de Jonathan Valdez y Jared Becker.[26]

Figura 4. Protocolo I2C bus para un sistema embebido, con soporte para múltiples dispositivos esclavos. El microcontrolador representa el maestro quien controla la comunicación usando 2 conectores.




Fuente: VALDEZ, Jonathan y BECKER, Jared. Understanding the $I^2C$ Bus. En: Texas Instruments Application Report. Junio, 2015. p. 1.

El protocolo $I^2C$ consta de 2 puntos de comunicación entre el maestro y los múltiples dispositivos conectados a él. Estos 2 puntos se conocen como las líneas SDA y SCL.

El protocolo $I^2C$ requiere un circuito de colector abierto (BJT), drenador abierto (CMOS) junto con un buffer de entrada, lo cual permite que la línea de datos opere con un flujo bidireccional.

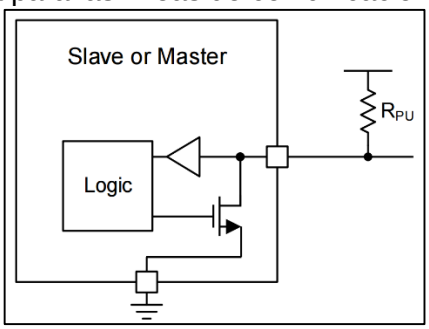
### 5.4.1 Circuito de colector abierto para la comunicación bidireccional

Colector abierto se refiere a un tipo de salida que permite forzar el bus a un voltaje mínimo (tierra en la mayoría de los casos), o liberar el bus y dejarlo ser forzado por

[26] VALDEZ, Jonathan y BECKER, Jared. Understanding the $I^2C$ Bus. En: Texas Instruments Application Report. Junio, 2015. p. 2

una resistencia de pull-up. En el momento en que el bus es liberado, ya sea por el maestro o alguno de los dispositivos esclavos, la resistencia de pull-up ($R_{PU}$) es la responsable de forzar el bus a un voltaje alto (ver *Figura 5*). Dado que no hay ningún dispositivo capaz de forzar la línea de datos en alto, la resistencia de pull-up garantiza que el bus de comunicación no tendrá inconveniente alguno en el momento en que algún dispositivo quiera transmitir un dato alto en cualquier dirección. Por otra parte previene que se pueda generar un corto cuando un dispositivo trata de enviar un valor alto mientras otro envía un valor bajo.

Figura 5. Estructura interna básica para las líneas de comunicación SDA/SCL.

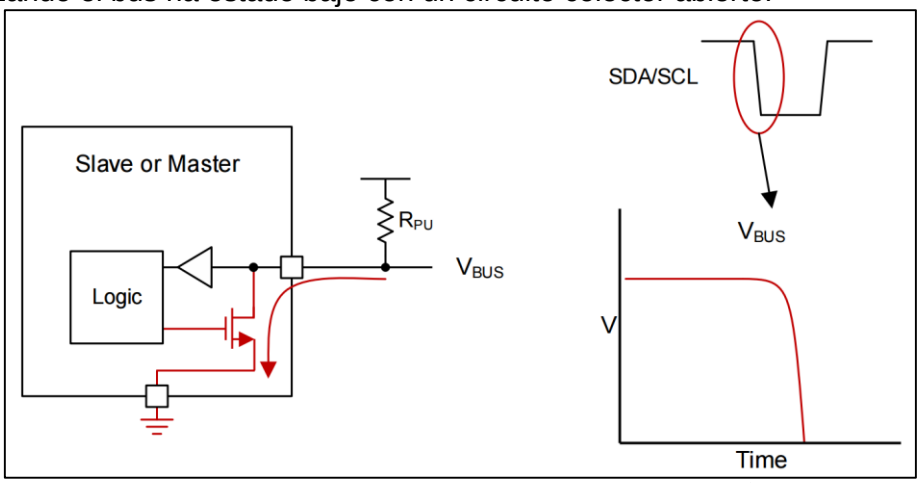



Fuente: VALDEZ, Jonathan y BECKER, Jared. Understanding the I²C Bus. En: Texas Instruments Application Report. Junio, 2015. p. 2.

*5.4.1.1 Colector abierto forzando el estado bajo en el bus*

La configuración del circuito colector abierto, solo le permite forzar el bus a un estado bajo. Para que éste pueda pasar a estado alto, el bus es liberado y finalmente permitir a la resistencia de pull-up llevarlo a estado alto. La *Figura 6* muestra el flujo de corriente que permite forzar el bus ha estado bajo, como se aprecia en la imagen, la lógica que desee poner la línea en bajo, estimula el FET de pull-down, el cual provee un corto a tierra y garantiza el bus en estado bajo.

Figura 6. Forzando el bus ha estado bajo con un circuito colector abierto.

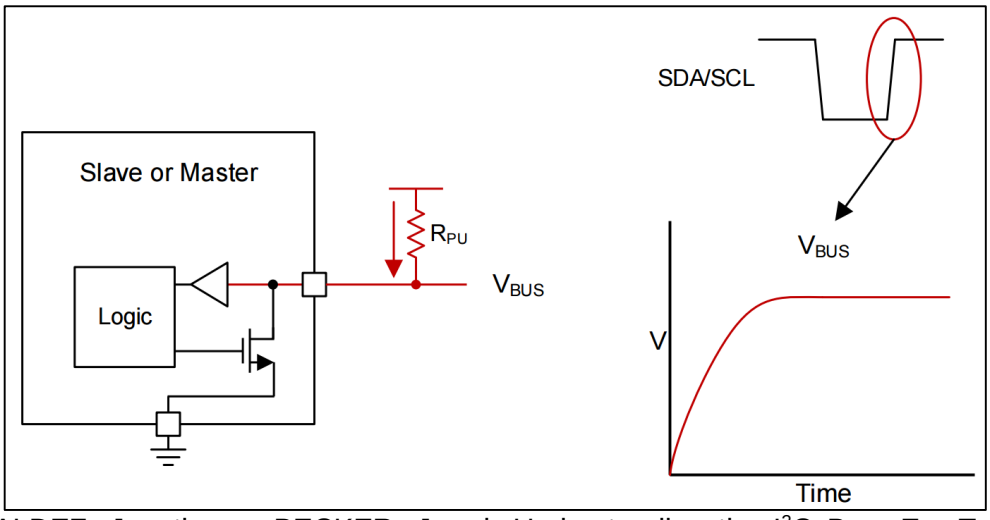



Fuente: VALDEZ, Jonathan y BECKER, Jared. Understanding the I²C Bus. En: Texas Instruments Application Report. Junio, 2015. p. 2.

*5.4.1.2 Colector abierto liberando el bus*

Cuando algún dispositivo conectado al bus $I^2C$ desea transmitir un estado lógico alto éste solo se debe encargar de apagar el FET pull-down, lo cual se conoce como liberar el bus. Al interrumpir el corto a tierra, la resistencia de pull-up ($R_{PU}$) se encargará de elevar la tensión hasta un valor lógico alto. La *Figura 7* muestra el flujo de corriente a través de la resistencia de pull-up.

Figura 7. Liberando el bus con un circuito colector abierto.




Fuente: VALDEZ, Jonathan y BECKER, Jared. Understanding the $I^2C$ Bus. En: Texas Instruments Application Report. Junio, 2015. p. 3.

## 5.5 TECNOLOGÍA FPGA

La tecnología de arreglos de compuertas programables de campo ha estado en constante crecimiento durante la última década. Se espera que el mercado de FPGAs en todo el mundo aumente de $1.900 millones en el 2005 a $2.750 millones en el 2010 y alcance los $4.550 millones para el año 2015.[27] Xilinx propuso esta plataforma en el año 1984, y desde entonces han pasado de ser sencillos chips de lógica de acoplamiento a reemplazar los circuitos integrados de aplicación específica (ASIC).

De forma amplia, una FPGA es un chip de silicio reprogramable. Utiliza bloques de lógica pre-construidos y recursos para ruteo inteligentes. Ofrecen la posibilidad de diseñar y personalizar hardware sin necesidad de utilizar una protoboard o un cautín. Permite desarrollar un circuito diferente cada vez que se compila y se lanza un programa, dándole la versatilidad de la que no puede presumir ningún otro controlador en el mercado actual. Anteriormente sólo los ingenieros con un profundo

[27] The Field-Programmable Gate Array (FPGA): Expanding its Boundaries. En: InStat Market Research. Abril, 2006.

entendimiento de diseño en hardware digital podían trabajar con esta tecnología, sin embargo con el aumento de fondos en investigación y desarrollo, hoy existen poderosas herramientas de diseño para programación en alto nivel que facilitan las cosas al momento de crear nuevos proyectos. La expansión acelerada que han tenido las FPGA se deben en gran medida a los importantes beneficios listados a continuación:

- *Rendimiento:*

El pilar más representativo de las FPGAs es su descomunal rendimiento frente a otros sistemas micro controlados. Aprovechando el paralelismo del hardware, se excede la potencia de cómputo de los DSP, dejando de lado el paradigma de la ejecución secuencial y logrando más en cada ciclo de reloj. BDTI, una destacada firma analista que se encarga de la revisión de evaluaciones de referencia, lanzó un informe que evidencia como las FPGA pueden entregar más potencia de procesamiento por dólar que una solución de DSP.[28] Permite controlar entradas y salidas a nivel de hardware, ofreciendo así tiempos de respuesta mucho más pequeños.

- *Costo:*

Su costo es superior con respecto al de los MPU alternativos para tareas sencillas. Sin embargo el precio de la ingeniería no recurrente de un diseño personalizado ASIC excede considerablemente al de las soluciones de hardware basadas en FPGA.[29] La fuerte inversión inicial de los ASICs es fácilmente justificable para los fabricantes de equipos originales que embarcan miles de chips por año, pero muchos usuarios finales necesitan la funcionalidad de un hardware personalizado para decenas o incluso cientos de sistemas que aún se encuentran en desarrollo.
La capacidad programable del silicio implica que no hay un costo extra de fabricación o largos plazos de ensamblado.

- *Fiabilidad:*

Mientras que las herramientas de software ofrecen un entorno de programación, los circuitos de una FPGA son una implementación segura de la ejecución de un programa. Los sistemas basados en procesadores frecuentemente implican varios niveles de abstracción para programar las tareas y compartir los recursos entre procesos múltiples. El software a nivel de driver se encarga de administrar los recursos de hardware y el sistema operativo administra la memoria y el ancho de banda del procesador. El núcleo de un procesador solo puede ejecutar una instrucción a la vez, y los sistemas basados en procesadores están siempre en riesgo de que sus tareas se obstruyan entre sí. Las FPGAs minimizan los retos de

[28] BDTI Focus Report: FPGAs for DSP. En: BDTI Benchmarking, Second Edition. 2006.
[29] THOMPSON, M. FPGAs accelerate time to market for industrial designs. En: EE Times. 2004. [en Linea] <http://www.us.design-reuse.com/articles/8190/fpgas-accelerate-time-to-market-for-industrial-designs.html> [Citado en: Septiembre, 2016].

fiabilidad con ejecución paralela de procesos y hardware preciso dedicado a cada tarea.

- *Soporte y mantenimiento:*

Los proyectos se encuentran en constante cambio, la necesidad de rediseñar está siempre presente y en este aspecto las FPGA destacan por su facilidad a la hora de adaptarse a nuevas situaciones. Los protocolos de comunicación digital, por ejemplo, tienen especificaciones que pueden cambiar con el tiempo y las interfaces basadas en ASIC podrían causar retos en el momento de la actualización. Mientras el producto se va desarrollando, se pueden implementar mejoras tanto a nivel software y hardware con tan solo programar y quemar el dispositivo.

### 5.5.1 Altera Cyclone v

Cyclone V[30] es una familia de embebidos que hacen parte del catálogo de Altera. Según la página del desarrollador, la familia Cyclone V proporciona el sistema con mayor relación costo/beneficio de la industria, permitiendo bajos consumos de energía y altos niveles de rendimiento que hacen a los dispositivos ideales para dar un plus a las aplicaciones de gran volumen.

Cyclone V ofrece un consumo hasta 40% menor al de la generación previa, con habilidades de integración lógica eficientes, y como atractivo principal, la inclusión de un sistema ARM basado en un sistema de procesamiento avanzado (HPS).

El procesador embebido en el chip de altera, es un dual core ARM Cortex-A9. El chip contiene transistores fabricados a 28nm, que se encuentran en la misma placa del ecosistema FPGA. La integración de campo programable junto con un procesador de alto rendimiento en un espacio tan reducido, eleva enormemente las posibilidades de desarrollo y el potencial que puede alcanzar el producto final. Para lograr esto, el modelo más económico de la familia Cyclone V cuenta con al menos:

- Procesador dual-core ARM Cortex-A9 800MHz (HPS).
- 85k elementos lógicos programables.
- 4.450 Kbits de memoria embebida.
- 6 PLLs Fraccionales.
- Controladores de memoria robustos para SDRAM (FPGA) y DDR3 (HPS).

[30] ALTERA. CYCLONE V FPGA & SOCS. [en Linea] <https://www.altera.com/products/fpga/cyclone-series/cyclone-v/overview.html> [Citado en: Septiembre, 2016].

### 5.5.2 Terasic boards

Terasic Technologies[31] es la empresa encargada de distribuir los chips de Altera. Para ello, acopla un SoC de trabajo al chip brindándole una gran cantidad de periféricos que aumenten las posibilidades de desarrollo. Tales periféricos pueden ser jacks para manejo de Audio, memoria de tipo SDRAM o DDR3, EPCS128, puertos USB, puerto PS/2, pulsadores, switches, leds, displays, entre otros.

El uso de un board permite desarrollar proyectos con mayor facilidad, y además con mayor fiabilidad, puesto que son dispositivos soldados de manera superficial que no solo buscan la universalidad en espacio reducido, sino además cuentan con el respaldo de un montaje con diseño de calidad y diversas protecciones contra posibles fallos o corto circuitos.

## 5.6 SENSORES DE PROXIMIDAD

Un sensor es, según lo define NESTEL[32], un convertidor técnico, que se encarga de traducir una variable física (temperatura, distancia, presión, humedad, etc.) en otra de diferente naturaleza más fácil de evaluar, que por lo general se trata de una señal eléctrica.

Los sensores son dispositivos que pueden funcionar tano por medio de un contacto físico (finales de carrera o sensores de fuerza) como sin contacto físico (barreras fotoeléctricas, detectores infrarrojos, sensores de reflexión ultrasónicos, sensores magnéticos, etc.).

Dentro de un proceso controlado, los sensores representan los perceptores que supervisan dicho proceso, indicando fallos, recogiendo estados y transmitiendo esta información a los demás componentes del proceso.

Los sensores de proximidad son aquellos que reaccionan a una variación física entre un objeto en particular con respecto a la posición actual del sensor. Estos se clasifican según el mecanismo que emplean para lograr la transducción de la variable posición.

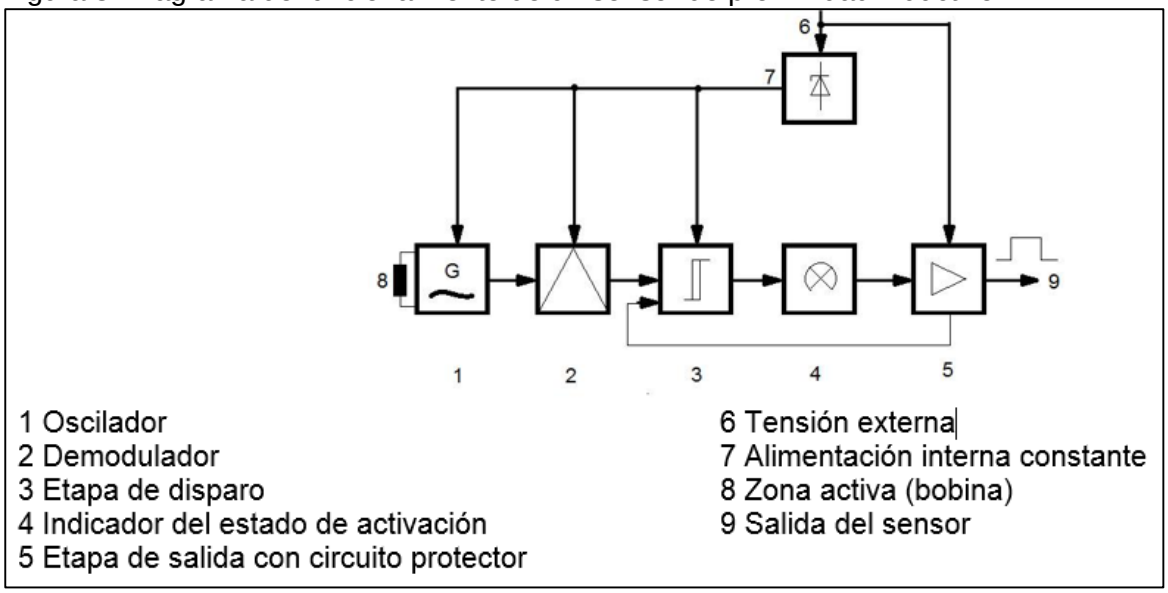
- Sensores de proximidad magnéticos[33]: son capaces de reaccionar ante los campos magnéticos de imanes permanentes y de electroimanes. Generalmente contienen láminas de contacto hechas a partir de material

[31] Terasic Technologies es un proveedor exclusivo del programa de desarrollo de FPGA de la Universidad de Altera, así como fabricante exclusivo de los kits de desarrollo de Altera.
[32] NESTEL, S. Sensores para la técnica de procesos y manipulación: Sensores de proximidad. Festo Didactic, 1993. 336 h. ISBN 381-27-3047-2. p. 11
[33] Ibid., p. 44

ferromagnético (Fe-Ni) y están selladas junto con un gas inerte como el nitrógeno. Si se acerca un campo magnético al sensor de proximidad, las láminas se unen por magnetismo y se produce un contacto eléctrico.

- Sensores de proximidad inductivos[34]: los componentes más importantes de un sensor de proximidad inductivo son un oscilador (circuito resonante LC), un rectificador demodulador, un amplificador biestable y una etapa de salida. El campo magnético que es dirigido hacia el exterior, es generado por medio del núcleo de ferrita semiabierto de una bobina osciladora y de un apantallado adicional. Esto crea un área limitada a lo largo de la superficie activa del sensor de proximidad inductivo, la cual se conoce como zona activa de conmutación. Cuando se aplica una tensión al sensor, el oscilador se activa y fluye una corriente de reposo definida. Si un objeto conductor de electricidad se introduce en la zona activa de conmutación, se crean unas corrientes parasitas que restan energía al oscilador. La oscilación se atenúa y esto produce un cambio en el consumo de corriente del sensor de proximidad. Los dos estados (oscilación atenuada y sin atenuación) se evalúan electrónicamente. Esto conlleva a la limitante de que un sensor de proximidad inductivo solo puede detectar materiales conductores de electricidad. En la *Figura 8* se aprecian las etapas que componen el funcionamiento de un sensor de proximidad inductivo.

Figura 8. Diagrama de funcionamiento de un sensor de proximidad inductivo.




Fuente: NESTEL, S. Sensores para la técnica de procesos y manipulación: Sensores de proximidad. Festo Didactic, 1993. 336 h. ISBN 381-27-3047-2. p. 64.

[34] Ibid., p. 64

- Sensores de proximidad capacitivos[35]: el principio de funcionamiento de un sensor de proximidad capacitivo, está basado en la medición de los cambios de capacitancia eléctrica de un condensador en u circuito resonante RC, ante la aproximación de cualquier material. En un sensor de proximidad capacitivo, entre un electrodo activo y uno puesto a tierra, se crea un campo electrostático disperso, generalmente también se halla presente un tercer electrodo para compensación de las influencias que pueda ocasionar la humedad en el sensor de proximidad. Si un objeto o un medio (metal, plástico, vidrio, madera, agua) irrumpe en la zona activa de conmutación, la capacitancia del circuito resonante se altera, permitiendo interpretar esta variación como un reflejo de la aproximación del mismo.

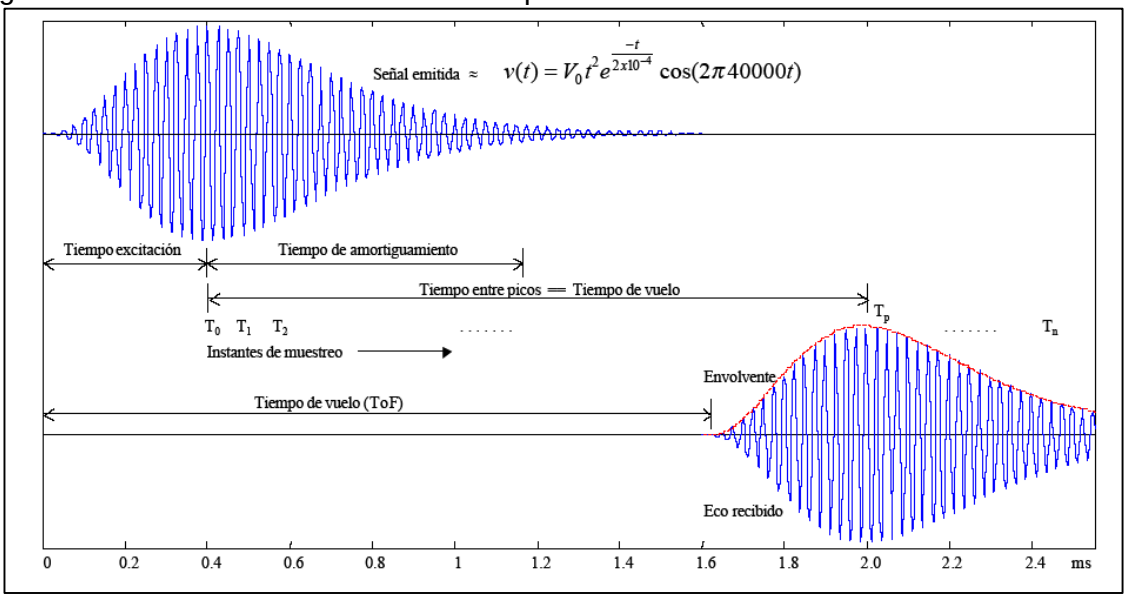
- Sensores de proximidad ópticos[36]: estos sensores utilizan medios ópticos y electrónicos para la detección de objetos. Para ello se utiliza una señal de luz roja (visible) o infrarroja (invisible). Los diodos semiconductores emisores de luz (LED) son una fuente particularmente fiable de luz roja e infrarroja. Son pequeños y robustos, tienen larga vida útil y pueden modularse fácilmente. Los fotodiodos y los fototransistores se utilizan como elemento receptor. La luz roja tiene ventaja a la hora de ajustar un sensor de proximidad, ya que es visible. La luz infrarroja se utiliza en ocasiones en las que se requiere cubrir mayor distancia y es casi imperceptible a las perturbaciones o interferencias de la luz ambiental. Con ambos tipos de sensores de proximidad ópticos, la supresión adicional de las influencias de luz externas se alcanza por medio de la modulación de la señal óptica. El receptor (con excepción de los sensores de barrera) se sintoniza con los pulsos del emisor. Con sensores de barrera se utiliza un pasa banda eléctrico en el receptor.

- Sensores de proximidad ultrasónicos[37]: el principio de funcionamiento de un sensor de proximidad ultrasónico está basado en la emisión y reflexión de ondas acústicas entre un emisor, un objeto y un receptor. Normalmente, el portador de estas ondas sónicas es el aire. Se mide y se evalúa el tiempo que tarda en desplazarse el sonido. El sensor de proximidad ultrasónico puede dividirse en tres módulos principales. El transductor ultrasónico, la unidad de evaluación y la etapa de salida. Un pulso corto dispara brevemente el transmisor ultrasónico, este es generalmente un módulo piezoeléctrico, es decir, basado en piezo-óxidos (materiales cerámicos que reaccionan según el piezoeléctrico, de forma similar al cuarzo). El transmisor ultrasónico emite ondas sónicas en el rango inaudible a cualquier frecuencia (generalmente entre 30 y 300 KHz). En muchos casos el transmisor ultrasónico cambia de emisor a receptor, es decir, operando como un

[35] Ibid., p. 80-81
[36] Ibid., p. 92
[37] Ibid., p. 138-139

micrófono. Los filtros dentro del sensor de proximidad ultrasónico, comprueban si el sonido recibido es realmente el eco de las ondas sónicas emitidas, analizando la envolvente de la señal recibida como se aprecia en la *Figura 9*.

Figura 9. Forma de onda de un sensor de proximidad ultrasónico.




Fuente: Wiki de Robótica. Sensor de Ultra sonidos [en línea]. <http://wiki.robotica.webs.upv.es/wiki-de-robotica/sensores/sensores-proximidad/sensor-de-ultrasonidos/> [citado en 7 de septiembre de 2016]

- Sensores de proximidad neumáticos[38]: con los sensores de este tipo pueden detectarse la presencia o ausencia de un objeto por medio de chorros de aire que detectan sin contacto. Cuando se presenta un objeto se produce un cambio en la presión de la señal, que se procesa posteriormente. Este tipo de sensores se usa cuando el medio se encuentra en un estado de suciedad elevada, o con temperatura excesiva, y es inmune a perturbaciones de tipos magnético o sónico.

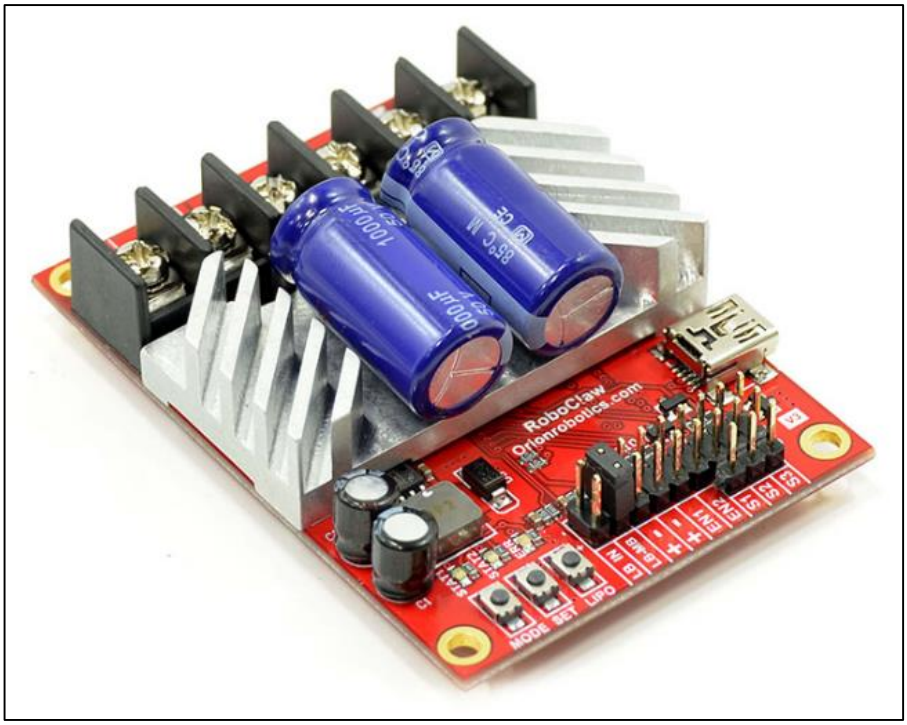
## 5.7 POLOLU ROBOCLAW

Roboclaw[39] es una tarjeta inteligente desarrollada por Ion Motion Control (ver *Figura 10),* diseñada para controlar un par de motores de corriente continua utilizando señales provenientes de un puerto USB serial, TTL serial, RC o análogas. Los

[38] Ibid., p. 150
[39] RoboClaw. ION MOTION CONTROL. Tarjeta para el control de motores de corriente continúa. [en Linea] <http://www.ionmc.com/34VDC_c_19.html> [Citado en: Septiembre, 2016].

decodificadores duales de cuadratura integrados hacen más fácil crear un sistema de control de velocidad en lazo cerrado.

Figura 10. RoboClaw 2X30A versión 4.




Fuente: RoboClaw. Pololu Electronics [en línea]. < https://www.pololu.com/product/2393> [citado en 7 de septiembre de 2016]

Roboclaw viene en diferentes versiones que satisfacen diversas necesidades. Su versión más básica ofrece la posibilidad de manejar dos canales síncronos y regenerativos, donde cada canal soporta corrientes nominales de hasta 7 A (14 A corriente pico), mientras que la versión más completa soporta hasta 60 A (120 A corriente pico). Esto convierte a Roboclaw en una tarjeta eficiente y versátil para el control de velocidad y sentido de giro en motores de corriente continua.

El driver requiere de configuración previa al uso, para definir la manera de recepción de información, y la activación de diversas protecciones como del uso o no del frenado regenerativo para la carga de baterías en el frenado o la inversión de giro.

Existen 4 modos principales de funcionamiento, con pequeñas variaciones internas, para un total de 14 modos diferentes de operación. Cada modo habilidad al Roboclaw para ser controlado en un modo específico.

El puerto USB puede ser conectado en cualquier modo de operación. Cuando el Roboclaw no se encuentra configurado para la comunicación serial a través del

puerto USB, este se usa para leer comandos como el estado actual del dispositivo o los parámetros con los que ha sido programado.

En la siguiente lista se explica brevemente cada modo principal de funcionamiento del Roboclaw y la aplicación ideal para cada uno:

- Modo RC (1-2): con el modo RC el Roboclaw puede ser controlado desde cualquier sistema de radio control. El modo RC también permite alimentar microcontroladores de baja potencia desde el mismo Roboclaw. Durante este modo de configuración, se espera modulación PWM para controlar la dirección y velocidad de los motores. Es el modo más similar al funcionamiento de un servo convencional.

- Modo Análogo (3-4): el modo análogo usa señales desde 0 V hasta 2 V para controlar la velocidad y dirección de cada motor. Roboclaw puede ser controlado utilizando un potenciómetro o una señal filtrada desde un microcontrolador. El modo análogo es ideal para sistemas de posicionamiento utilizando joysticks.

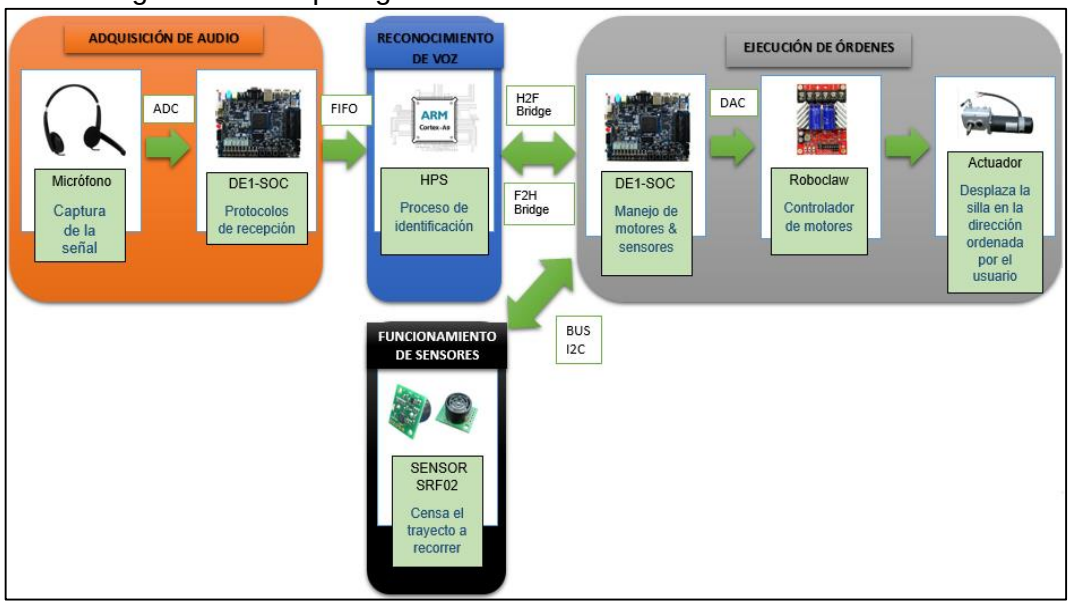
- Modo Serial Estándar (5-6): el modo serial estándar espera señales en nivel TTL a través de estándar RS-232 para controlar dirección y velocidad en cada motor. Es el estándar más utilizado para controlar el Roboclaw desde un microcontrolador o un pc. El modo estándar serial incluye un modo de selección de esclavo que permite conectar múltiples Roboclaw para que se comuniquen entre sí. Los encoders no están soportados en este modo de funcionamiento.

- Modo de Paquetes Seriales (7-14): el modo de paquetes seriales espera señales en nivel TTL a través de estándar RS-232 para controlar dirección y velocidad en cada motor. Es el estándar más utilizado para controlar el Roboclaw desde un microcontrolador o un pc. En este modo cada Roboclaw conectado al puerto RS-232 posee una dirección única, donde hasta 8 Roboclaw pueden comunicarse simultáneamente con el microcontrolador. Los enconders están soportados en este modo de operación.

# 6 DIAGRAMA GENERAL DEL SISTEMA

El procedimiento está distribuido en cuatro bloques principales, indicando la ejecución y desarrollo del proceso como se muestra en la *Figura 11*, desde el momento, pasando por el procesamiento de la señal hasta llegar a la repuesta representada en los actuadores.

Figura 11. Diagrama de bloques general del sistema.




Fuente: Elaboración propia.

Para la captura de las muestras de audio se utiliza una diadema LifeCHat LX-1000 de Microsoft la cual se conecta a un códec de audio Wolfson WM8731 donde se realiza el proceso de conversión de la señal análoga a digital. Este códec se comunica mediante protocolo I2C y se encarga de enviar los datos muestreados a 48 KHz a un módulo AUDIO_DAC. Este módulo se encarga de separar los dos canales de audio y almacenarlos en un FIFO diseñado en Hardware. El HPS recibe muestras de Audio a 8 KHz y se completa la etapa de captación de la señal audible.

Todo el proceso de reconocimiento de voz ocurre en el HPS. En él se realiza la etapa de preprocesamiento aplicando MFCC y la identificación utilizando HMM. Cuando se obtiene una hipótesis sobre la frase pronunciada, se envía a la etapa de manejo.

Finalmente, el módulo Motor_Controller recibe una acción concreta a realizar y se encarga de su ejecución. Para ello envía al Roboclaw la información del movimiento deseado sobre los actuadores. Ocurre en paralelo durante todo el proceso, una etapa de detección de obstáculos mediante sensores de ultrasonido SRF02. Este módulo es el encargado de impedir el desplazamiento para evitar colisiones.

# 7 SELECCIÓN DE FPGA

Para poder realizar el sistema de reconocimiento digital de voz en un sistema programable de campo, se realizó un proceso de selección del modelo comercial más viable para tal fin. Desarrollar un algoritmo robusto de reconocimiento de voz con vocabulario extendible requiere el uso de hardware y software especializado en técnicas de identificación de patrones mediante la ejecución de cálculos matemáticos de gran envergadura. Altera y Xilinx son los principales desarrolladores dedicados a la tecnología FPGA. Ambas soluciones ofrecen procesadores embebidos como son NIOS II y MicroBlazer, respectivamente. En cantidad de elementos lógicos y cores IP, ambas desarrolladoras ofrecen modelos viables que se ajustan a diferentes rangos de precios. Para encontrar el modelo ideal que se adapte a los requerimientos que demanda el proyecto, se deben tener en cuenta algunos factores de selección como el costo, las herramientas de desarrollo disponibles o los periféricos que vienen incluidos en el board.

## 7.1 FACTORES DE CONSIDERACIÓN

### 7.1.1 Herramientas de desarrollo

A la hora de desarrollar un sistema embebido en FPGA, las herramientas de desarrollo que el fabricante ofrece toman un papel protagónico pues tienen la capacidad de facilitar y hacer más intuitiva la labor del programador. Cabe aclarar que los kits de desarrollo contienen, por lo general, muchas mega-funciones que requieren de suscripción para su uso final. Cuando se programa sobre una FPGA, se busca tener al alcance herramientas de desarrollo poderosas al menor costo posible. Para el caso específico, la Universidad Pontificia Bolivariana no posee licencia para desarrollo con ningún fabricante de FPGA.

### 7.1.2 Periféricos incluidos en el board

El objetivo de una FPGA reside en su capacidad para lograr un procesamiento universal. Para lograrlo, los board donde vienen soldadas las FPGA, contienen un gran número de periféricos con diversas funciones y habilidades. El tema central de este proyecto recae en el tratamiento de señales de audio, por ello, un requerimiento es la inclusión de jacks de 3,5mm para uso exclusivo de audio, así como conversor ADC y DAC incluidos. Sin embargo se requieren muchos otros periféricos como puerto USB para el uso de memorias flash, puerto Ethernet, SDRAM, EPCS, a parte de los más básicos y frecuentes como pulsadores, leds y switches.

### 7.1.3 Conversores ADC y DAC

Existen muchos modelos que excluyen los conversores de señales digitales y análogas, así como otros que incluyen más de un canal de conversión a 20 bits de resolución. En este proyecto se requiere de conversión de señales tanto ADC como DAC, sin embargo no es representativa la inclusión de módulos de 2 canales o más, o de resoluciones superiores a los 12 bits, dado que las señales de audio van a ser muestreadas a 48 KHz.

### 7.1.4 Unidad central de procesamiento

Es quizá la característica más importante en el desarrollo de este proyecto. Para el desarrollo de un sistema que realice reconocimiento de voz por deletreo, debido al grado de complejidad del algoritmo a emplear, se requiere una unidad central de procesamiento con la potencia suficiente para garantizar una respuesta eficiente en el sistema, con tiempos de ejecución de tareas cortos y cómodos para el usuario final. Aquí cabe resaltar que no es lo mismo interpretar comandos de habla que realizar un motor de reconocimiento de voz. Si el objetivo fuera reconocer comandos pre grabados, muchas de las soluciones ASIC podrían desarrollar esta labor por un costo menor. Sin embargo realizar procesamiento de voz utilizando MFCC y HMM exige sistemas eficientes de alto rendimiento. Quizá lo más viable en este caso será una FPGA que cuente con un procesador de alto rendimiento embebido, que permita integración de software y hardware en la misma capa de procesamiento. Esta unidad central de procesamiento deberá ser capaz de aplicar matemáticamente transformada de Fourier, transformada de coseno, bancos de filtros de Mel, e identificación de fonemas mediante hipótesis formuladas usando procesos estocásticos no observables. NIOS y Microblaze requieren de licencia para el uso de sus funciones de alto rendimiento, por lo que se descartan.

### 7.1.5 Costo

Otro de los factores de gran peso es el costo que demande la compra de la FPGA. En el mercado, existen gran variedad de SoCs de desarrollo con una gran cantidad de periféricos para tareas exclusivas. Muchos de estos pueden llegar a ser sub utilizados en el proyecto por lo cual no se contempla la adquisición de un módulo con periféricos más allá de los requeridos. A medida que el número de elementos lógicos que contiene una FPGA aumenta, también lo hace el precio. Lo ideal es adquirir un modelo que se adapte a las necesidades y ofrezca la capacidad de desarrollar un sistema de bajo costo pero utilizando tecnología de vanguardia.

## 7.2 COMPARACIÓN DE FPGAS COMERCIALES

Teniendo en cuenta las características mencionadas anteriormente, se realizó una tabla donde se comparan algunos de los modelos disponibles que satisfacen las necesidades. Cabe resaltar que la Universidad Pontificia Bolivariana cuenta actualmente con diferentes modelos de FPGA, de los cuales ninguno cumple con los requerimientos mínimos para el desarrollo de este proyecto especialmente por la ausencia de conectores dedicados para audio así como la no inclusión de un sistema central de procesamiento de alto rendimiento con garantías suficientes para el procesamiento digital de voz en tiempo real. De esta manera se hace pertinente la adquisición de un módulo de desarrollo que permita seguir con la siguiente fase de desarrollo del proyecto. La *Figura 12* muestra la tabla comparativa de los SoCs.

Figura 12. Tabla comparativa de FPGAs comerciales.

| |  Terasic DE1-SoC |  Arrow SoCKit ( from Terasic ) |  Xilinx Zedboard |  Embest Lark Board |  Altera Cyclone V SoC Development Board |
|---|---|---|---|---|---|
| FPGA Device | Cyclone V SoC 5CSEMA5F31C6N | Cyclone V SoC 5CSXFC6D631C6N | "Xilinx® XC7Z0201CLG484C Zynq-7000 AP SoC" | Cyclone V SoC 5CSXFC6D6F31 | Cyclone V SoC 5CSXFC6D6F31C6N |
| logic Elements | 85K | 110K | 85K | 110K | 110K |
| HPS SDRAM | 1GB DDR3 | 1GB DDR3 | 512MB DDR3 | 1GB DDR3 | 1GB DDR3 |
| FPGA SDRAM | 64MB SDRAM | 1GB DDR3 | | 1GB DDR3 | 1GB DDR3 |
| Flash | | 128MB QSPI Flash | 256MB QSPI Flash | | 1Gb QSPI Flash |
| USB | Host x2 | OTG x1 | OTG x1 | Host x4 | OTG x 1 |
| Ethernet | 1 Gigabit | 1 Gigabit | 1 Gigabit | 1 Gigabit | 1 Gigabit + |
| UART | UART to USB | UART to USB | UART to USB | UART to USB | UART to USB |
| PCIe | | PCIe x4 | | PCIe x4 | PCIe x4 |
| Video in | CVBS | SDI | | SDI | |
| Video Out | 24-bit VGA DAC | 24-bit VGA DAC | VGA/ HDMI | VGA / HDMI / SDI | SDI |
| Audio | 24-bit CODEC | 24-bit CODEC | Audio COEDC | | |
| FPGA | 40-pin GPIO x2 | HSMC x1 | FMC x1 / Pmod x5 | 40-pin GPIO x1 | HSMC x1 |
| HPS Expansion | LTC Connector (I2C/SPI) | LTC Connector (I2C/SPI) | | 40-pin GPIO x1 | LTC Connector (I2C/SPI) |
| User LED/Botton/Switch/7-SEG | LEDS x 11 / 10 Switches / 4 Buttons / 6 7-SEGs | 8 LEDS/ 8 Switches/ 8 Buttons | 9 LEDS/ 8 Switches/ 7 Buttons | 8 LEDS / 4 Switches/ 4 Buttons | 8 LEDS / 6 Switches/ 8 Buttons |
| LCD | | 128x64 dots LCD Module | 128x64 OLED | | 16x2 character LCD |
| ADC | 8CH/12-bit | 8CH/12-bit | | 2CH/12-bit | |
| Sensor | G-Sensor | G-Sensor/ Temperature sensor | | | |
| Other Interface | PS/2 & IR Emitter / Receiver | | | LCD and FPC Connector | CAN |
| Price | $249 | $350 | $495 | $1.699 | $1.795 |

Fuente: Elaboración propia.

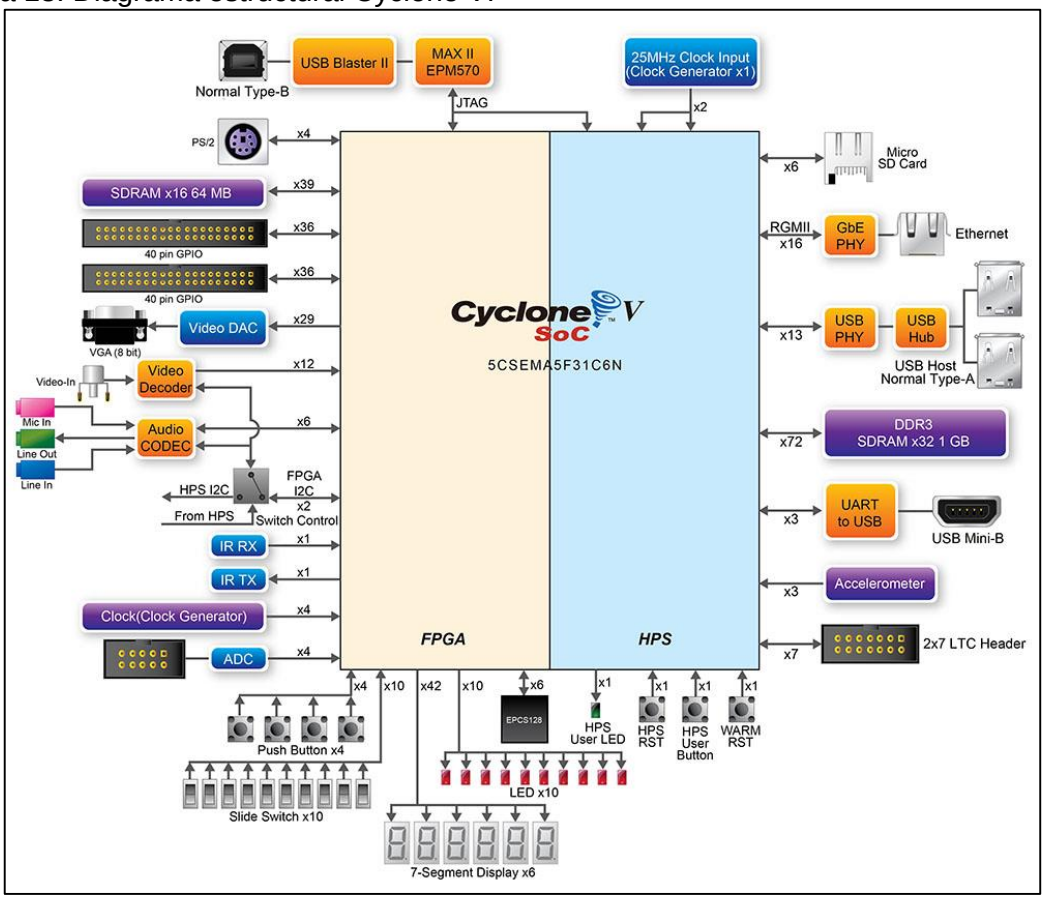
## 7.3 FPGA SELECCIONADA

Luego de realizar un análisis en detalle sobre los distintos modelos existentes en el mercado, se decidió que la FPGA que se ajusta de manera favorable para el desarrollo óptimo del proyecto es una Terasic DE1-SoC que incluye un Cyclone V de Altera. La Universidad Pontificia Bolivariana no contaba con este modelo anteriormente, por tal razón se realizó el trámite para el proceso de adquisición del mismo.

La DE1-SoC es una board que incluye una FPGA Cyclone V desarrollada por Altera y distribuida por Terasic. Este modelo cuenta con la FPGA y el HPS embebidos en la misma placa. Contiene todas las prestaciones que ofrece la programación en campo con los beneficios de un procesador embebido de alto rendimiento para la ejecución de programas sobre un sistema operativo.

La *Figura 13* muestra el diagrama de bloques donde se evidencian los diferentes periféricos que posee el board de desarrollo DE1-SoC.

Figura 13. Diagrama estructural Cyclone V.




Fuente: TERASIC TECHNOLOGIES. DE1-SoC: User Manual. Taiwan: Terasic Technologies Inc. All Rights reserved: 2015. p. 9.

### 7.3.1 Especificaciones técnicas DE1-SoC

Las siguientes son las especificaciones técnicas del board de desarrollo DE1-Soc.

**Dispositivo FPGA:**

- Cyclone V SoC 5CSEMA5F31C6
- Procesador dual-core ARM Cortex-A9 (HPS)
- 85k elementos lógicos programables
- 4450 Kbits de memoria embebida
- 6 PLLs Fraccionales
- Controladores de memoria robustos para SDRAM (FPGA) y DDR3 (HPS).

**Configuración y Debug:**

- Dispositivo de configuración serial – EPCS128
- USB Blaster II (conector USB tipo B)

**Dispositivos de Memoria:**

- SDRAM 64 MB (32x16)
- DDR3 1 GB (2x256MBx16)
- Socket para tarjetas microSD

**Dispositivos de Comunicación:**

- Dos puertos USB 2.0
- USB a UART (micro USB conector tipo B)
- 10/100/1000 Ethernet
- PS/2 para mouse/teclado
- IR transmisor/receptor

**Conectores:**

- Dos conectores de 40 pines (voltaje: 3.3v)
- Conector de 10 pines para entrada de ADC
- Conector LTC

**Video:**

- 24-bit VGA DAC
- Decodificador TV (NTSC/PAL/SECAM) y conector entrada de TV

**Audio:**

- 24-bit CODEC, con jacks para línea de entrada, línea de salida y micrófono

**ADC:**

- Muestreo: 500 KSPS
- Número de canales: 8
- Resolución: 12 bits
- Rango de entrada análoga: 0 - 4.096 V

**Switches, botones e indicadores:**

- 4 botones
- 10 Switches
- 11 LEDs (10 para FPGA, 1 para HPS)
- 2 botones de reset para HPS
- Seis 7-segmentos

**Sensores:**

- G-Sensor para el HPS

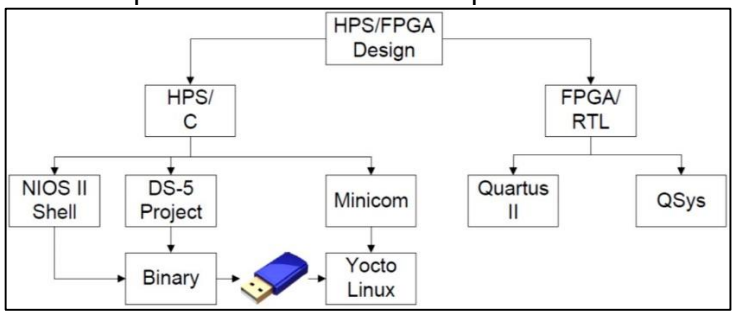
**Alimentación:**

- 12 V DC de alimentación

# 8 DISEÑO HPS

Lo primero en el desarrollo de un proyecto en FPGA es el diseño interno de los módulos que componen la estructura principal del sistema. Para el DE1-SoC, este proceso se centra en el diseño del HPS, pues este requiere de varios pasos de configuración inicial para definir su funcionamiento posterior. En la *Figura 14* se observa una estructura general de las herramientas que están implícitas en el desarrollo de software y hardware de un diseño HPS/FPGA.

Figura 14. Diagrama conceptual de las herramientas para el diseño del HPS.




Fuente: TERASIC TECHNOLOGIES. DE1-SoC: User Manual. Taiwan: Terasic Technologies Inc. All Rights reserved: 2015.

El HPS es un módulo hardware que se debe preparar para utilizar el procesador ARM cortex-A9 incluido en el chip Cyclone V, sobre el cual se va a correr una variante del sistema operativo Linux, conocida como Yocto. El diseño del hardware del HPS debe definir los periféricos a los que el ARM va a acceder y asignar recursos de memoria para conectarlos virtualmente, así como la administración de tiempos de ejecución. Cabe aclarar que las conexiones en este apartado ocurren de manera similar a la creación de un sistema embebido NIOS II[40], por lo que se requiere del sistema QSYS[41] para su elaboración.

En el proceso de diseño, es importante decirle al campo de la FPGA como se va a comportar el HPS. Para lograr esto, Altera ofrece un archivo de tipo QSF donde se encuentran implícitas las asignaciones internas de configuración para un HPS.
Para añadir el archivo de asignaciones al proyecto se siguen los siguientes pasos:

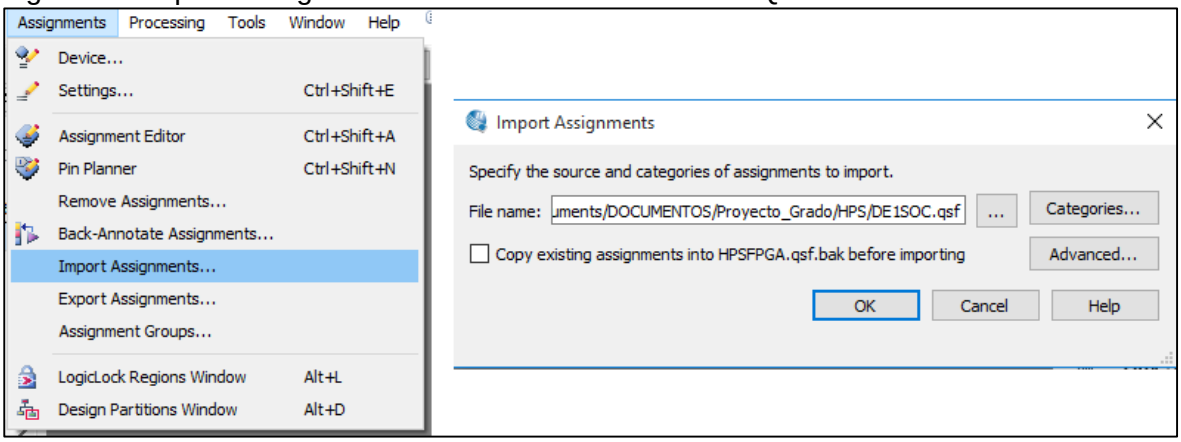
- En el proyecto, ingresar al apartado de Assignments
- En la ventana Assignments, seleccionar la opción de importación
- Ubicar la dirección del archivo DE1SOC.qsf
- Seleccionar el archivo DE1SOC.qsf

[40] Altera NIOS II. Procesador ASIC optimizado para desarrollo en tiempo real. Soporta los SoC y Familia de FPGA de Altera.
[41] Altera QSYS. Herramienta de integración para el diseño y generación automático de interconexiones lógicas para funciones y subsistemas de propiedad intelectual IP.

- Deshabilitar la opción de copiar asignaciones existentes en HPSFPGA.qsf
- Finalizar con la opción OK

En la *Figura 15* se describe el procedimiento para la importación de las asignaciones de pines internos en el HPS.

Figura 15. Importar asignaciones del archivo DE1SOC en Quartus II.




Fuente: Elaboración propia en Quartus II, versión 15.0.

La herramienta Qsys de Quartus II incluye diferentes bloques IP (propiedad intelectual), los cuales son subcircuitos prediseñados para el uso en diversos diseños. Estos bloques son elaborados, en su gran mayoría, por el instituto de desarrollo de propiedad intelectual de la Universidad de Altera, sin embargo, muchos de ellos provienen de empresas asociadas, razón por la cual se encuentran bloques IP que requieren licencia para su inclusión en proyectos y diseños. Entre el portafolio de bloques IP que Altera ofrece se encuentran funciones básicas, funciones para DSP, diversos protocolos de interfaz, funciones de bajo consumo para tareas generales, controladores de memoria, procesadores, sistemas embebidos, periféricos, entre muchos otros.

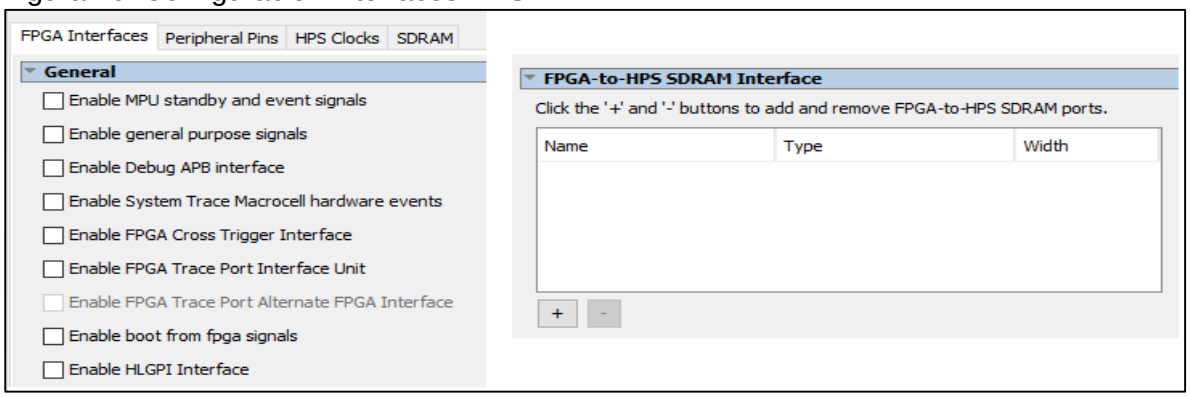
Lo primero es añadir el IP core relacionado con el diseño del HPS. Para ello se busca en el catálogo de las mega-funciones disponibles en la sección de procesadores embebidos. Una vez agregado este módulo, se procede a la configuración más relevante para la configuración de conexiones y comportamiento.

Cuando se diseña en FPGA es muy importante usar los recursos de la manera más eficiente posible, dado que muchos de ellos son limitados. Uno de los recursos más valiosos en este proyecto es la memoria SDRAM, pues en ella se realizan los puentes de información que permiten la comunicación entre HPS/FPGA y distintos periféricos. En el desarrollo del sistema de reconocimiento de voz no se requiere que la FPGA acceda a periféricos conectados al HPS, pues la mayoría de procesamiento ocurrirá en el ARM. Se consideró pertinente desactivar la interfaz de comunicación H2F que permite el acceso de recursos de Hardware a periféricos del

HPS. Seleccionando Arria/Cyclone V Hard Processor Systems se realizan las respectivas configuraciones.

En la pestaña FPGA Interfaces, se puede referir a la configuración general donde se deshabilita la espera MPU y señales de evento relacionadas a la comunicación HPS/FPGA. Posteriormente en la ventana de interfaz para la comunicación específica FPGA-to-HPS se desactiva pulsando el botón – para suprimir esta opción. En la *Figura 16* se visualiza gráficamente el proceso anteriormente descrito.

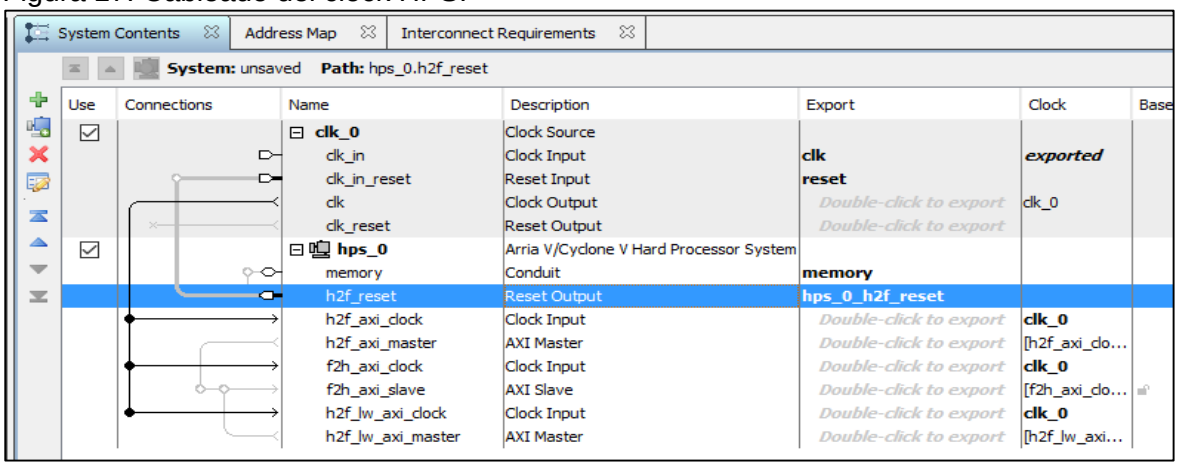
Figura 16. Configuración interfaces HPS.




Fuente: Elaboración propia en Quartus II, versión 15.0.

En la ventana de System Contents aparecerán todos los IP cores que se vayan añadiendo al proyecto. En la *Figura 17* se observa el módulo HPS_0, así como los tres principales puertos de comunicación existentes en el ecosistema HPS/FPGA, los cuales son el h2f, el f2h y el lwh2f. Es necesario conectar todos los clock de los puertos de comunicación con el clock maestro (clk_0) para sincronizar el sistema.

Figura 17. Cableado del clock HPS.

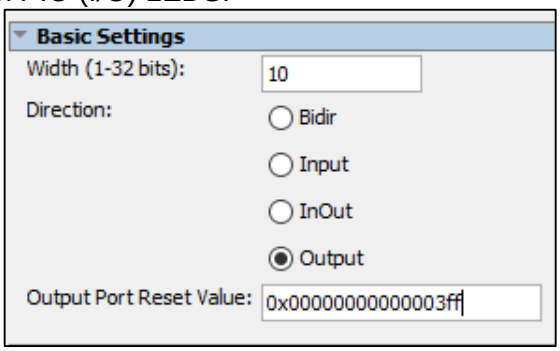



Fuente: Elaboración propia en Quartus II, versión 15.0.

Para que el HPS pueda acceder a periféricos conectados directamente a la FPGA, se deben agregar puertos paralelos de entradas y salidas PIO's en la herramienta QSYS. Estos puertos permiten al HPS leer o escribir en periféricos como LEDs, Switches y pulsadores. Al proyecto se agregaron PIO's para el control de los LEDS que pueden ser una ayuda para observa o confirmar los procesos ejecutados, así como swiches y los pulsadores para controlar acciones específicas. Estos puertos se utilizan principalmente para realizar pruebas en la etapa de desarrollo y elaboración del proyecto.

La configuración de un PIO es bastante sencilla. Solo se debe configurar el puerto como entrada, salida o bidireccional y asignarle un ancho en número de bits. Para el caso de los LEDS se requiere un PIO con dirección de salida, pues es el HPS quien busca escribir en los registros de valores de los LED, y se le da un ancho de 10 bits (debido a que se desea controlar 10 LEDs) además de la configuración Output para salida. En la casilla Port Reset Value se selecciona el valor en hexadecimal 0x3ff (en binario 1111111111) que enciende por defecto los LEDs. En la *Figura 18* se puede apreciar la configuración anteriormente descrita.

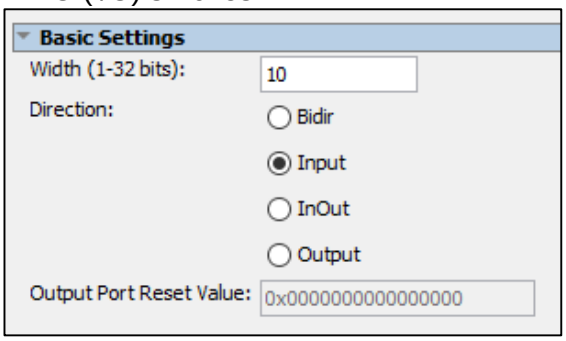
Figura 18. Configuración PIO (I/O) LEDS.




Fuente: Elaboración propia en Quartus II, versión 15.0.

Se agregan otros PIO para swiches y pulsadores. Se configura para que sean entradas de un ancho de 10 y 4 bits respectivamente como se aprecia en la *Figura 19*.

Figura 19. Configuración PIO (I/O) swiches.

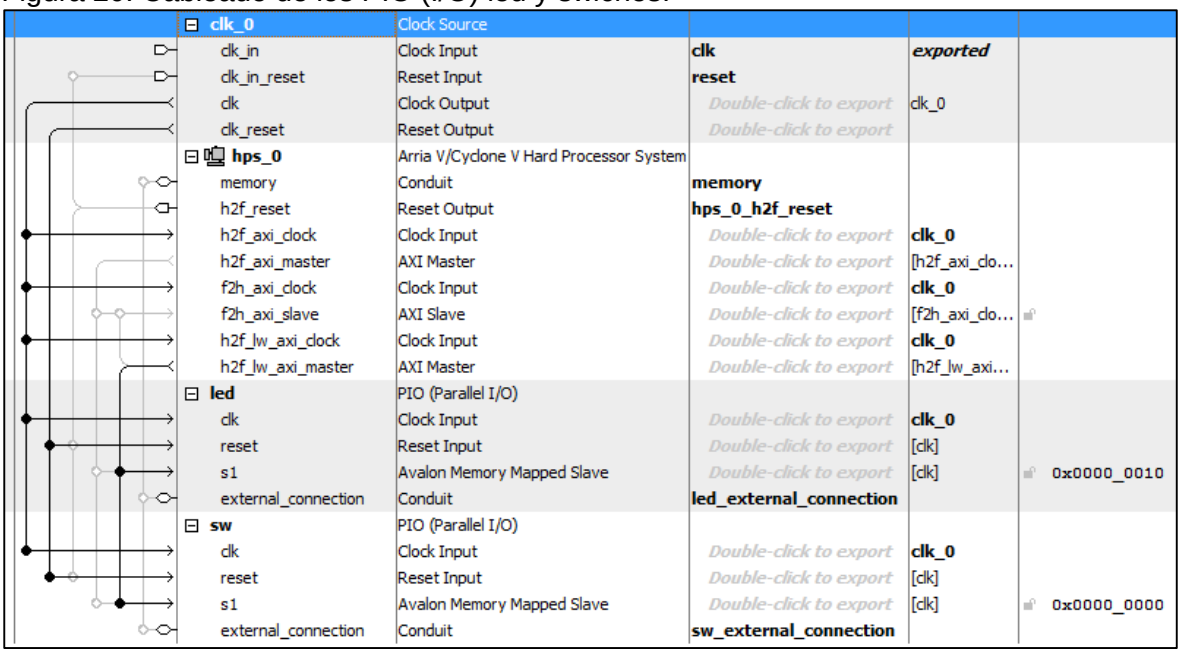



Fuente: Elaboración propia en Quartus II, versión 15.0.

Una vez más, todos los bloques IP deben estar referenciados según el clock global. Los PIO utilizados para comunicarse con periféricos en la FPGA, contienen una conexión de tipo Avalon para la memoria mapeada donde serán apuntados los periféricos. Para el caso de dispositivos como led, switches y pulsadores se utiliza el puente de comunicación h2f_lw pues el flujo de datos es bajo.

Es muy importante generar las conexiones externas de cada uno de los elementos incluidos en el sistema QSYS, pues a partir de estas se podrá instanciar sobre la plantilla base del proyecto elaborada en Verilog. En la *Figura 20* se observan las conexiones del módulo HPS con los PIO que han sido renombrados según la función que cumpla cada uno.

Figura 20. Cableado de los PIO (I/O) led y swiches.




Fuente: Elaboración propia en Quartus II, versión 15.0.

En total, se requirieron 11 direcciones de memoria mapeada para el desarrollo de este proyecto. En la Tabla *1* se aprecian los diferentes puertos añadidos en el QSYS, junto con algunas características como la dirección de la información, el tamaño de cada puerto y una breve descripción de la función que cumple la información enviada o recibida en el módulo asociado a cada puerto.

No todos los puertos están dirigidos exclusivamente a un dispositivo en particular, algunos están dedicados exclusivamente a intercambiar información relevante entre el HPS y la FPGA, como señales de alerta por la detección de un objeto por parte de un sensor o señales de finalización de alguna tarea.

Tabla 1. Configuraciones para los PIO's.

| PIO (nombre) | I/O | Tamaño en No bits | Función |
|---|---|---|---|
| Div_freq | output | 32 | Enviar un valor para el contador de un FreqGen a 8 KHz. |
| Setup_index | output | 24 | Contiene los valores calibrados para los actuadores. |
| Setup_motor | output | 4 | Contiene la orden del HPS para enviar a motor_controller. |
| Tempo_motor | output | 8 | Tiene el valor de un tiempo de espera para motor_controller. |
| Finish_motor | Input | 1 | Indica al HPS que termino la acción ordenada. |
| Div_motor | output | 32 | Envía un valor para el contador de un FreqGen a 25 Hz. |
| Warning_motor | input | 1 | Indica al HPS que un sensor bloquea el desplazamiento. |
| Ready_motor | output | 1 | Indica a motor_controller que el HPS está listo. |
| led | output | 10 | Permite al HPS controlar los LED conectados a la FPGA. |
| sw | input | 10 | Permite al HPS controlar los SW conectados a la FPGA. |
| key | input | 4 | Permite al HPS controlar los key conectados a la FPGA. |

Fuente: Elaboración propia.

Para cada uno de ellos se realizan las mismas conexiones. Estos bloques funcionaran de manera simultánea para cada ciclo de reloj, por ello, todos estos deben ir referenciados por el clock global. Todos los puertos de conexión de memoria deben ir conectados al puente de comunicación h2f_lw_master.

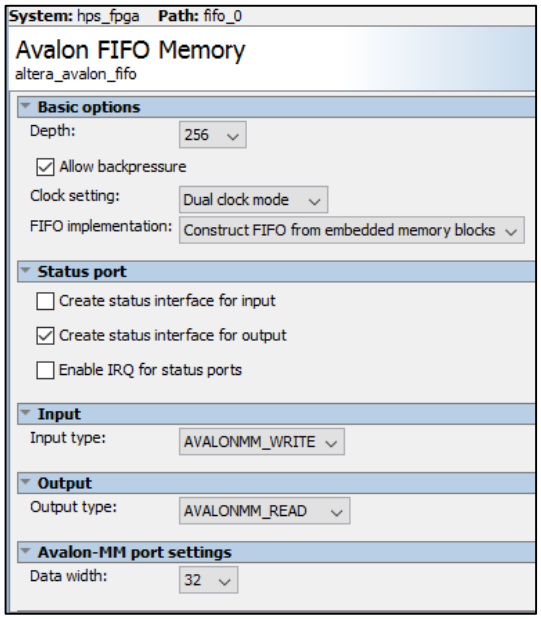
Se exportan los external_connection de los PIO's con los nombres mencionados en la Tabla *1*. Con esta configuración se concluye la comunicación HPS-to-FPGA, la

cual será explicada con mayor detenimiento más adelante. Por otra parte, la recepción de audio ocurre en la FPGA, y estos datos deben ir al HPS para poder ser procesados. Como se trata de audio, no es recomendable enviar datos a través de un puerto de comunicación convencional pues la FPGA y el HPS funcionan a una velocidad diferente y no se encuentran sincronizados.

Para que la transmisión de audio de la FPGA to HPS se realice correctamente, garantizando que los datos muestreados no se corrompan o varíe la frecuencia de muestreo, se necesita un FIFO que controle de manera sincronizada el flujo de datos. Xilinx[42] define el FIFO como una memoria intermedia de datos secuencial, donde la escritura y la lectura se pueden solapar pero no ocurre direccionamiento explícito. Generalmente se utilizan direcciones de memoria duales mapeadas (RAM), una dedicada a la escritura de un puerto, dirigida por el contador de escritura interna, y la otra dedicada a la lectura a través del otro puerto, dirigida por el contador de lectura interna. Esto permite el uso de dos relojes totalmente independientes para escritura y lectura, permitiendo a la FPGA ingresar datos al buffer del FIFO a una velocidad diferente a la que usa el HPS para leerlos. Esto se llama operación asincrónica, aunque la escritura y la lectura son operaciones síncronas en su respectivo dominio de reloj.

Para configurar el FIFO se toma del catálogo IP el módulo llamado Avalon FIFO Memory y se seleccionan los parámetros como se encuentra en la *Figura 21*.

Figura 21. Configuración del FIFO.




Fuente: Elaboración propia en Quartus II, versión 15.0.

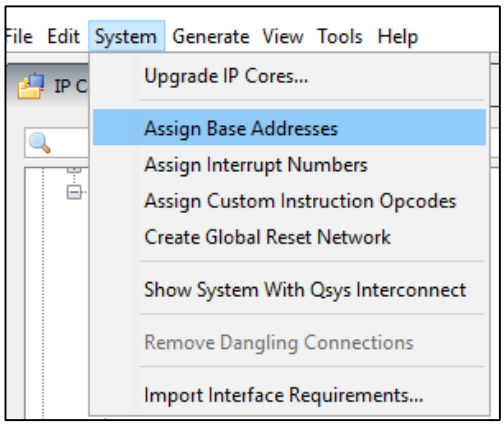
[42] PETER, A. FIFOs: A Tutorial Description. En: Xilinx PLD Blog, Community Forums. [en Linea] <https://forums.xilinx.com/t5/PLD-Blog/FIFOs-A-Tutorial-Description/ba-p/16959>. [Citado en: Septiembre, 2016].

El flujo de datos proveniente del muestreador del Codec Audio es superior al de los demás periféricos y señales a las que el HPS tiene acceso. Por esta razón, el FIFO no se comunica con la FPGA a través del puente lwh2f, usa el puerto de mayor densidad de datos h2f. Para el proyecto, este puente se encuentra exclusivamente para el funcionamiento del FIFO que mueve los datos de la señal de audio. Se cablea el FIFO al HPS y se conecta al puerto de h2f_axi_master y se exporta el external_connection nombrada fifo_0_in.

Como todos los dispositivos requieren un espacio de memoria donde se logra la comunicación, es importante que cada dispositivo o señal ya sea que se use para lectura o escritura, tenga una dirección de registro única. Esta dirección le permite a cada dispositivo usar un espacio determinado de memoria en el cual ningún otro dispositivo podrá acceder.

Estas direcciones base se definen en el QSYS ingresando a System > Assign Base Addresses como lo indica la *Figura 22*.

Figura 22. Asignación de dirección de bases del HPS y sus periféricos.



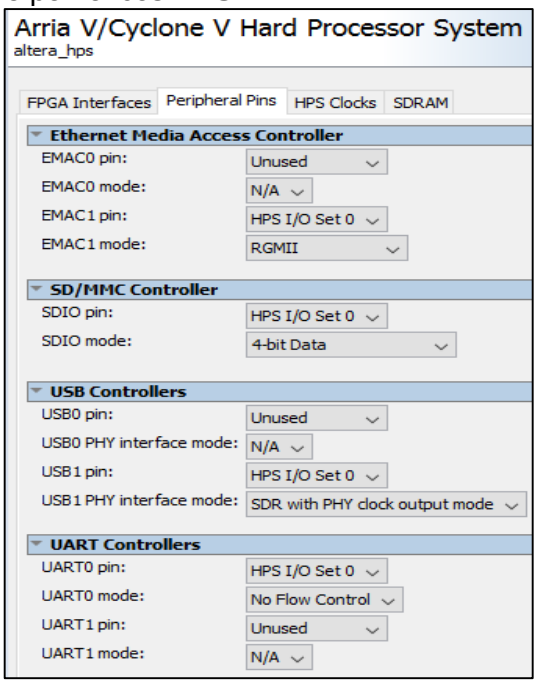

Fuente: Elaboración propia en Quartus II, versión 15.0.

Ahora se configuran los periféricos conectados directamente al HPS. Para esto, desde la sección de parámetros, se garantiza que el HPS pueda acceder al puerto Ethernet, a los puertos USB y UART configurando sus respectivos controladores.

- En el apartado de Ethernet Media Access Controller se modifica EMAC1 pin para seleccionar la opción HPS I/O Set 0, además el parámetro EMAC1 mode se configura en modo RGMII.
- En los parámetros de SD/MMC Controller se escoge para SDIO pin: HPS I/O Set 0 y para SDIO mode: 4-Bit Data.
- Para USB Controllers USB1 pin: HPS I/O Set 0 y USB1 PHY interface mode: SDR with PHY clock output mode.
- Para UART Controllers UART0 pin: HPS I/O Set 0 y en UART0 mode: No Flow Control.

En la *Figura 23* se visualiza el proceso de activación de los periféricos conectados directamente al HPS.

Figura 23. Configuración de periféricos HPS.



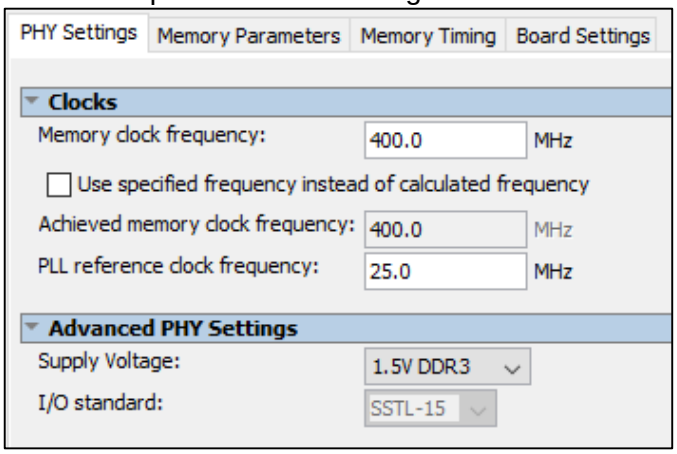

Fuente: Elaboración propia en Quartus II, versión 15.0.

Modificar los parámetros del reloj interno del HPS solo es recomendado para el usuario avanzado, con amplios dominios en arquitectura de computadores. No hace falta ninguna modificación adicional en este apartado para el desarrollo del proyecto. Se opta por conservar la configuración de tiempos internos de ciclos de máquina para el hps sugerida por Altera.

Configurar la sincronización de la memoria SDRAM si es un apartado destacado en la elaboración de cualquier proyecto. Los tiempos a elegir en el manejo de la memoria SDRAM no solo afectaran la velocidad de respuesta de los periféricos conectados a ella para la comunicación HPS/FPGA sino que determinaran la calidad de la información recibida y suministrada. Velocidades superiores a 10 veces la frecuencia del reloj global son soportadas pero no recomendadas. El clock global funciona a 50 MHz, por tanto la frecuencia de memoria elegida para el funcionamiento del proyecto fue de 400 MHz, garantizando un margen entre rendimiento y fiabilidad.

En la *Figura 24* se aprecia la ventana de configuración de la frecuencia establecida para el funcionamiento de la memoria SDRAM.

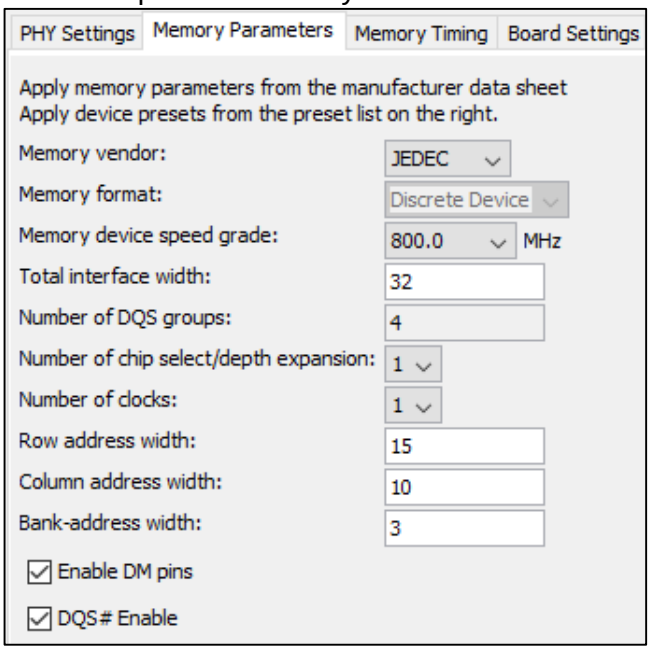
Figura 24. Configuración de la pestaña PHY Settings.




Fuente: Elaboración propia en Quartus II, versión 15.0.

Se debe configurar también el funcionamiento de la memoria RAM DDR3 de 1 GB conectada al HPS. Esta debe funcionar (por lo general) a la misma frecuencia que lo hace el HPS, es decir, 800 MHz. En la *Figura 25* se visualiza la configuración recomendada para la memoria DDR3.

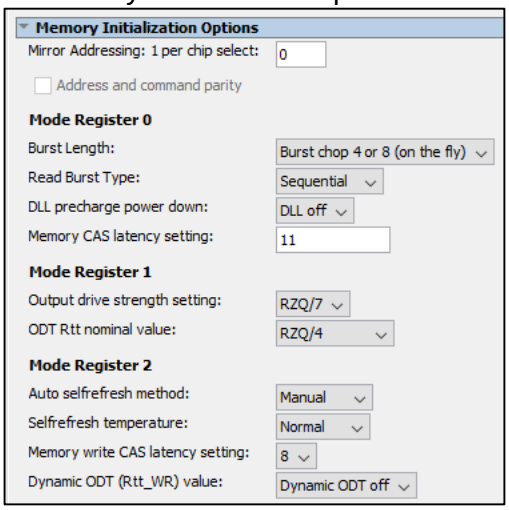
Figura 25. Configuración de la pestaña Memory Parameters del HPS.




Fuente: Elaboración propia en Quartus II, versión 15.0.

Configurar la memoria con una rutina de arranque garantiza que esta opere siempre que ocurra un evento de apagado o reinicio del sistema. Cuando se cargue el sistema operativo en el HPS, este funcionará utilizando procesos a través de memoria RAM. La configuración para la inicialización de memoria se aprecia en la *Figura 26*.

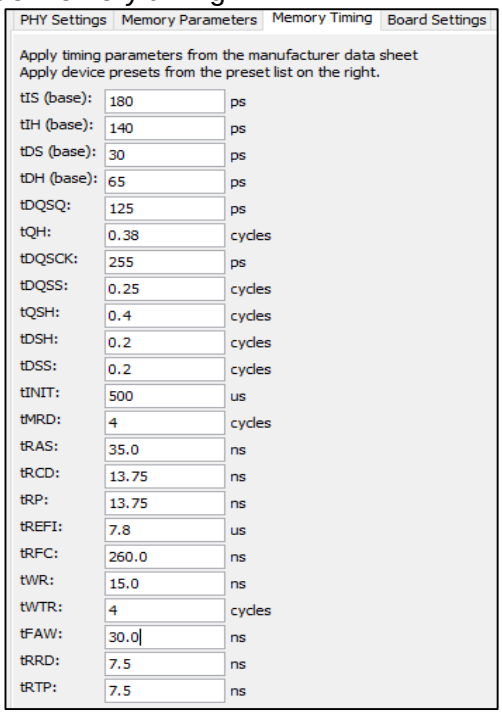
Figura 26. Configuración de Memory Initialization Options.




Fuente: Elaboración propia en Quartus II, versión 15.0.

Altera recomienda que los tiempos internos del uso de memoria se configuren según se aprecia en la *Figura 27*.

Figura 27. Configuración de Memory timing

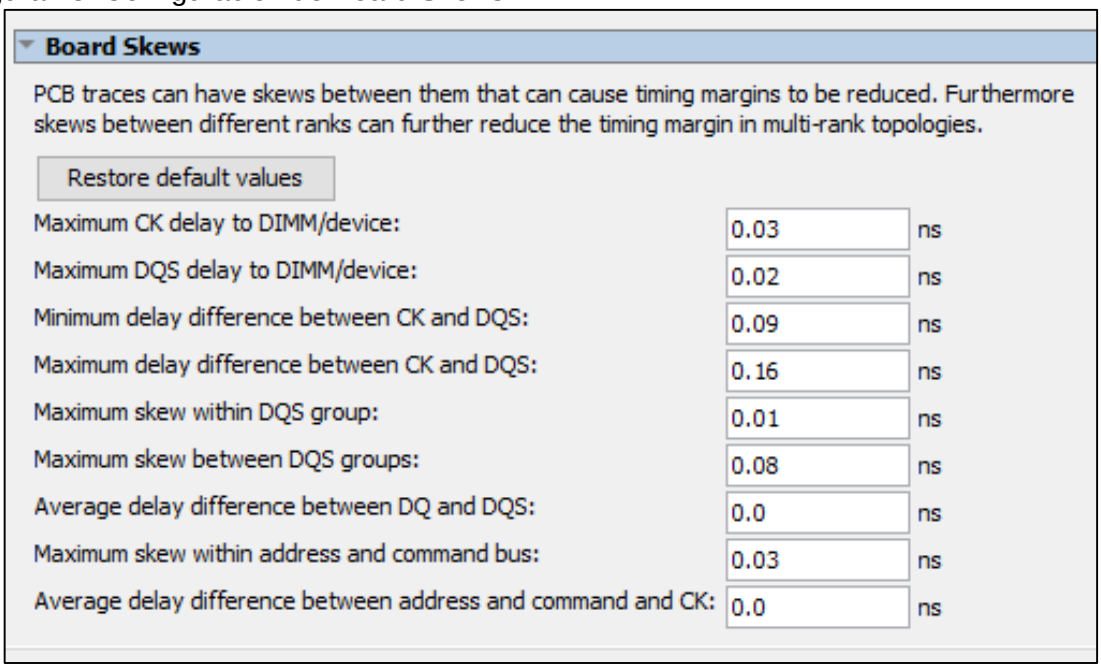



Fuente: Elaboración propia en Quartus II, versión 15.0.

El board al igual que cualquier circuito electrónico físico, tiene tiempos de retraso o tiempos muertos, que en la mayoría de ocasiones son despreciados. Altera propone un sistema de ajuste de márgenes para aumentar la fiabilidad de las aplicaciones y

la sincronización de sus módulos funcionando en paralelo para cada ciclo de reloj. En la *Figura 28* se aprecian los valores que sugiere altera para disminuir el riesgo de fallos o inestabilidad en el sistema.

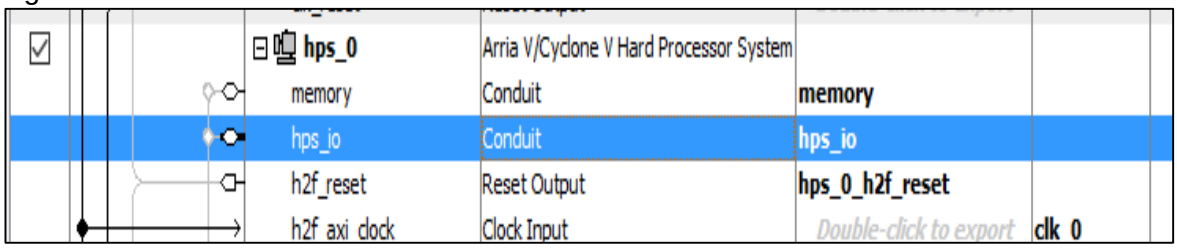
Figura 28. Configuración de Board Skews.




Fuente: Elaboración propia en Quartus II, versión 15.0.

Con esto se concluye la configuración del módulo HPS en el QSYS. Resta añadir una conexión externa para las señales hps_io y h2f_reset. Estas conexiones externas se utilizan para posteriormente instanciar el HPS a la plantilla general del proyecto. En la *Figura 29* se observan los puertos externos de conexión para el HPS.

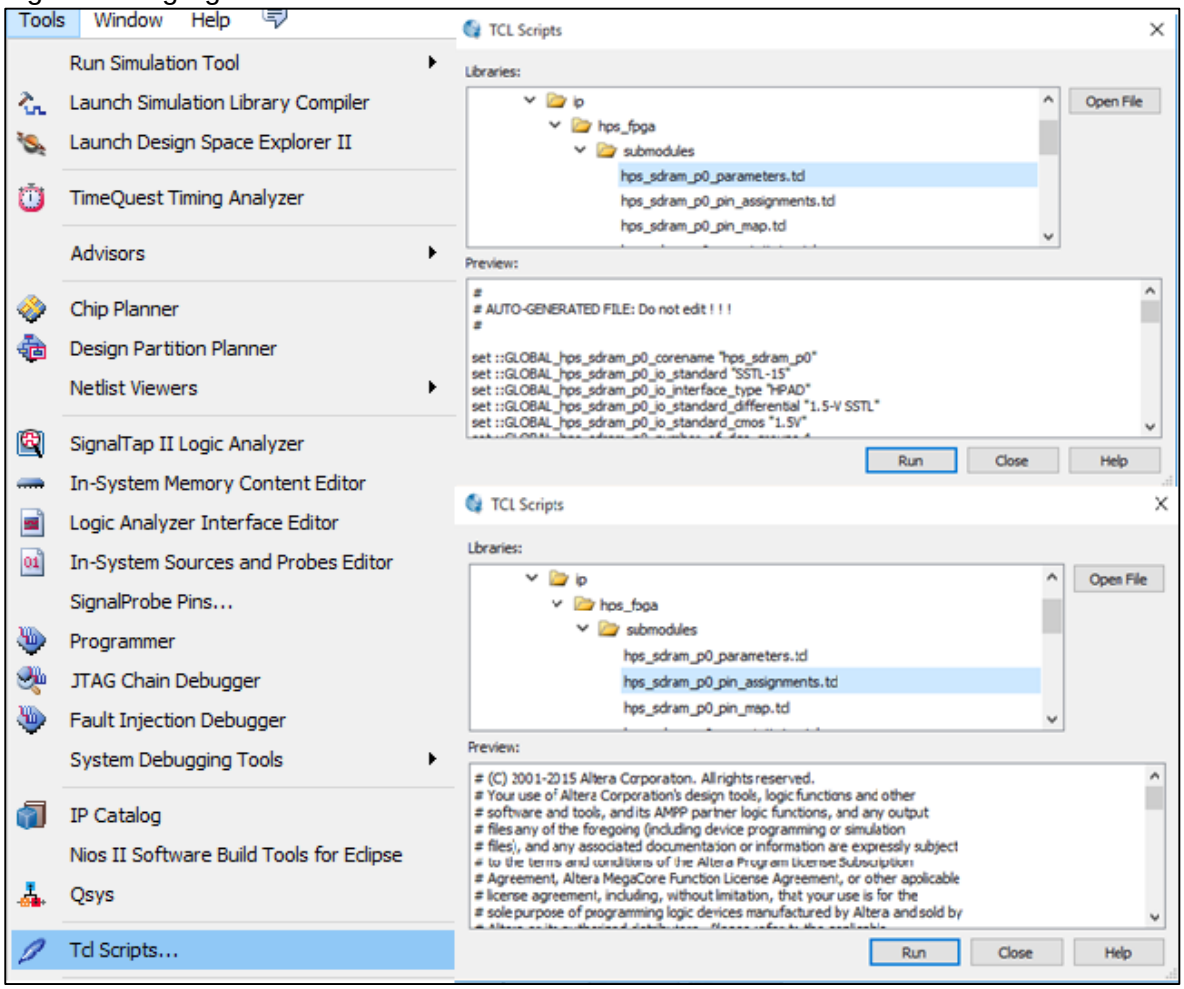
Figura 29. Creación de conexiones externas del HPS.




Fuente: Elaboración propia en Quartus II, versión 15.0.

Antes de compilar el QSYS, se deben agregar unos archivos extra para la configuración de parámetros internos de la memoria SDRAM. Este archivo viene en formato .tcl y son suministrados por Altera para el manejo óptimo de periféricos a través de puentes en memoria. Se deben ejecutar los tcl para los parámetros y la asignación de pines. En la *Figura 30* se visualiza el procedimiento para ejecutar los archivos tcl para el uso de la SDRAM.

Figura 30. Agregando Tcl del HPS.




Fuente: Elaboración propia en Quartus II, versión 15.0.

Con esto se procede a compilar y generar el QSYS del sistema para el entorno del HPS. Una vez compilado se añade a la plantilla del proyecto base por medio de instanciación para dejar lista la configuración de Hardware del HPS.

Cuando se culmina la etapa del QSYS, se genera un archivo hps_0.h que se ubica en la carpeta del proyecto en donde se contiene información clave relacionada con los tiempos y la configuración establecida. A partir de este archivo se deben generar una pequeña librería para que el software conozca características importantes como el espacio en memoria definido para comunicarse con cada dispositivo.

El archivo generate.sh contiene un pequeño código como se observa en la *Figura 31*, que crea un encabezado de las direcciones en memoria de cada dispositivo o puertos I/O utilizados dentro del HPS. El archivo hps_0.h es utilizado para la construcción del software reconocimiento de voz.

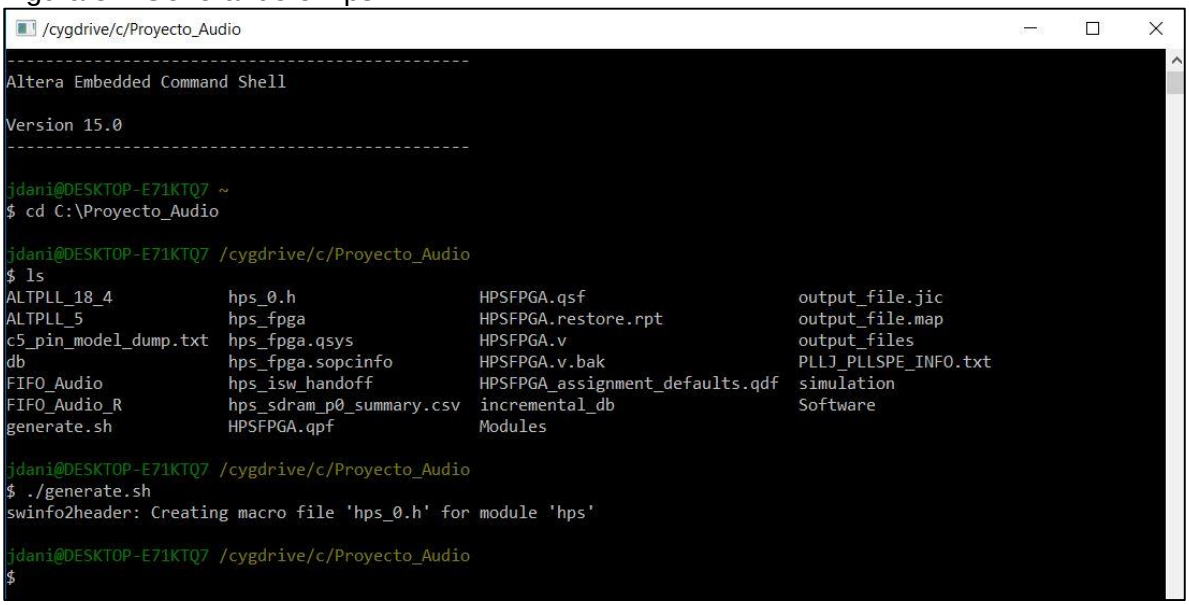
Figura 31. Código generate.sh.

```
#!/bin/sh
sopc-create-header-files \
"./hps_fpga.sopcinfo" \
--single hps_0.h \
--module hps
```

Fuente: Elaboración propia en Notepad.

Se introduce el script generate.sh en la carpeta del proyecto, y luego se ejecuta el Embedded Command Shell de Altera. En él se selecciona la ruta del proyecto, y se ejecuta el script para crear el macro de la librería en C. En la *Figura 32* se visualiza la creación de la librería .h para programación en C.

Figura 32. Generando el hps.h.



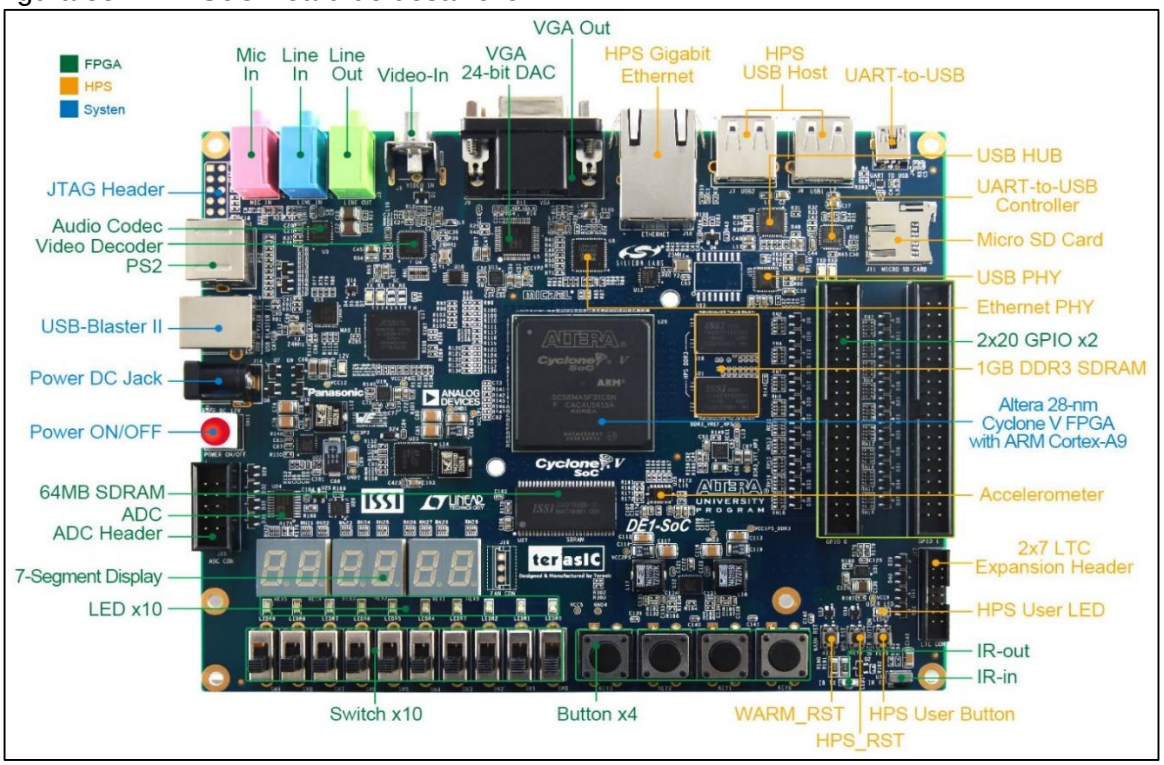

Fuente: Elaboración propia en SoC EDS 15.0 Command Shell

Una vez terminada la construcción del módulo HPS se añade LinuxYocto al procesador para lanzar el SO que va a ser el host del proyecto. Altera ofrece en su página web la imagen de Linux para el procesador ARM. Esta imagen se monta en una memoria microSD que ha sido configurada en el HPS con una rutina de arranque, para bootear el SO. Con esto se concluye el diseño del sistema embebido en la FPGA para realizar el reconocimiento de voz.

## 8.1 COMUNICACIÓN HPS/FPGA

En la DE1-SoC board, la FPGA y el HPS actúan como maestros del sistema. Como se aprecia en la *Figura 33*, los diferentes periféricos de los cuales dispone la board DE1-SoC, se encuentran conectados directamente a la FPGA o al HPS. Existen tres puentes principales que se usan para la comunicación: el puente FPGA-to-HPS (f2h), el puente HPS-to-FPGA (h2f) y el puente ligero HPS-to-FPGA (lwh2f "Lightweight)

Figura 33. DE1-SoC Board de desarrollo.




Fuente: TERASIC TECHNOLOGIES. DE1-SoC: User Manual. Taiwan: Terasic Technologies Inc. All Rights reserved: 2015. p. 8.

Los puentes de comunicación se recrean a partir de regiones en memoria RAM distribuida para el acceso a los periféricos del HPS o de la FPGA. En la Tabla *2*, se aprecia la asignación de bloques de memoria correspondientes para cada labor.

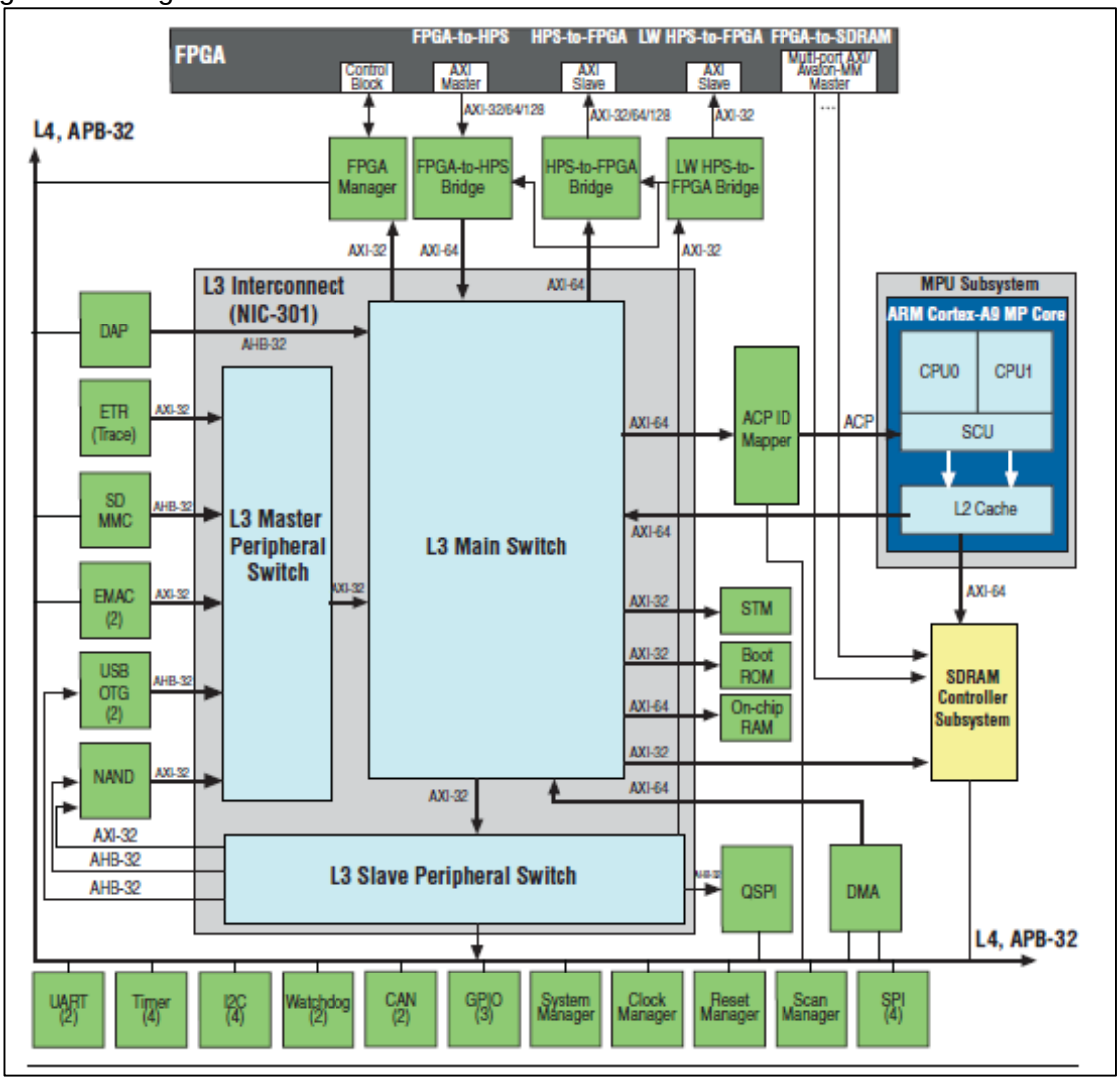
Tabla 2. Distribución de memoria para los puentes de comunicación entre FPGA y HPS.

| Region Name | Description | Base Address | Size |
|---|---|---|---|
| FPGA Slaves | For accessing FPGA slaves connected to the h2f bridge | 0xC0000000 | 960MB |
| HPS Peripherals | Accessing slaves directly connected to the HPS | 0xFC000000 | 64MB |
| Lightweight FPGA Slaves | Accessing slaves connected to the lwh2f bridge | 0xFF200000 | 2MB |

Fuente: ALTERA. Cyclone V device handbook: Hard Processor System Technical Reference Manual. San José: junio 2012. p. 465.

Hay dos posibilidades de comunicar el procesador ARM con los periféricos conectados directamente a la FPGA; a través del puente h2f con una capacidad de 960 MB y conectividad que soporta 64bits. Por otra parte, para adquisición o envío de datos de menor densidad, se puede usar una conectividad lwh2f que posee 2 MB, la cual soporta 32bits y es ideal para periféricos con bajo flujo de datos. Ambos puentes permiten al HPS comunicarse con la FPGA y sus periféricos a través de conexión L3, la cual está conectada al controlador de la SDRAM. En la *Figura 34* se aprecia el diagrama de bloques usado por el Cyclone V para la conexión HPS-to-FPGA y FPGA-to-HPS. En este se visualiza de manera detallada como el HPS, la FPGA y los periféricos asociados a cada uno se comunican entre sí a través de una serie de puentes y buses AXI.

Figura 34. Diagrama de conexión interna del HPS.




Fuente: KITTYDERK-HARBOR ELECTRÓNICA. Altera FPGA SoC. [en línea]. <http://www.eefocus.com/kittyderk/blog/13-10/299435_99408.html> [citado en 7 de septiembre de 2016]

Una vez que todo el hardware ha sido correctamente asignado, la comunicación entre el HPS y la FPGA es programada a través de una aplicación designada para el mapeo de memoria en C. Este mapeo permite al CPU ver y acceder a las direcciones de la FPGA, ya sea para leer o escribir información según se requiera, controlando de esta manera el hardware desde el software.

La plataforma empleada para la elaboración del software que se ejecuta sobre el HPS es el Eclipse ARM DS-5 Development Studio.

El HPS puede acceder a periféricos conectados a la FPGA a través de los puentes h2f y lwh2f de manera simultánea. Solo se deben tener en cuenta las direcciones de memoria donde inicia cada partición y la cantidad de memoria (SPAN) a la que se requiere acceder. En la *Figura 35*, se aprecia la definición del registro base que hace referencia a la dirección de inicio de memoria (REG_BASE) y un registro que indica la cantidad de memoria solicitada (REG_SPAN), para cada uno de los puentes de información.

Figura 35. Código implementado para definir las direcciones de los puentes de comunicación.

```
/* Voice Controller Parameters */
#define AXI_LW_REG_BASE         0xFF200000              //  Puerto h2f_lw_axi_master
#define AXI_LW_REG_SPAN         0x00200000
#define AXI_LW_REG_MASK         (AXI_LW_REG_SPAN-1)

#define AXI_REG_BASE            0xC0000000              //  Puerto h2f_axi_master
#define AXI_REG_SPAN            512
#define AXI_REG_MASK            (AXI_REG_SPAN-1)
```

Fuente: Elaboración propia en Eclipse ARM DS-5 Development Studio.

Para cada periférico se crea un puntero. Este va a ser el encargado de direccionar el programa a cada dirección de memoria específica asociada a un dispositivo conectado al puente de comunicación h2f. Recordar que en el proyecto, el puente h2f se encuentra habilitado para el uso exclusivo del envío de datos muestreados desde el Audio Codec hasta el HPS. Todas las demás señales se envían usando el puente LW. En la *Figura 36* se muestra la manera de garantizar el acceso a los puentes de memoria mapeada, así como la declaración de los apuntadores que permiten controlar los dispositivos desde el HPS.

Figura 36. Código para determinar las variables para el acceso de memoria.

```
/* Grant Memory Access */
void * axi_lw_virtual_base=NULL;        //  Puerto h2f_lw_axi_master
void * axi_virtual_base=NULL;           //  Puerto h2f_axi_master
volatile unsigned long * divfreq_addr=NULL;
volatile unsigned long * right_pwm_addr=NULL;
volatile unsigned long * left_pwm_addr=NULL;
volatile unsigned long * div_motor_addr=NULL;
volatile unsigned long * setup_index_addr=NULL;
volatile unsigned long * setup_motor_addr=NULL;
volatile unsigned long * tempo_motor_addr=NULL;
volatile unsigned long * finish_motor_addr=NULL;
volatile unsigned long * warning_motor_addr=NULL;
volatile unsigned long * ready_motor_addr=NULL;
```

Fuente: Elaboración propia en Eclipse ARM DS-5 Development Studio.

Se utiliza la función open para otorgar acceso a un determinado bloque de memoria RAM al procesador HPS, brindándole permisos de escritura y lectura, forzando la sincronización de acceso a memoria con los tiempos de trabajo del procesador ARM. En la *Figura 37* se visualiza la apertura de memoria en código C.

Figura 37. Abrir  el acceso de memoria.

```
fd=open("/dev/mem",(O_RDWR|O_SYNC))
```

Fuente: Elaboración propia en Eclipse ARM DS-5 Development Studio.

La función open, puede retornar en algunos casos un valor de -1, esto ocurre cuando la localidad de memoria a la que se intentó acceder no se encontró o tiene acceso restringido. Por ello se usa la pequeña subrutina para identificar si el mapeo se realizó de manera satisfactoria. En la *Figura 38* se aprecia la rutina que determina el estado de la localidad de memoria.

La función mmap permite mapear archivos o dispositivos en memoria. Para ello se indica que el mapeo ocurrirá desde una dirección de memoria específica, y para un ancho de memoria determinado. Además será de tipo compartido y con permiso de lectura y escritura. Una vez mapeado, se obtiene una dirección de memoria utilizable, sobre la cual se puede trabajar. Como se observa en la *Figura 39*, los espacios de memoria virtuales hacen referencia a los puentes de comunicación con la FPGA. No es necesario mapear cada dispositivo al cual se quiera acceder, solo mapear un bloque de memoria suficiente para albergarlos a todos, y luego direccionar esa memoria virtual mediante el apuntador asociado a cada uno de los periféricos seleccionados.

Ahora que la memoria ha sido mapeada virtualmente y se encuentra disponible para el acceso desde el HPS, se le dice al programa donde se encuentra cada uno de los componentes a través de los registros base de cada periférico. En la *Figura 40*, se visualiza la asignación de una dirección única para cada dispositivo conectado al puente de comunicación.

Figura 38. Código de la función de iniciación del sistema.

```
void init_system()
{
    if((fd=open("/dev/mem",(O_RDWR|O_SYNC)))==-1)
    {
        printf("ERROR: could not open \"/dev/mem\"...\n");
        exit (1);
    }

    axi_virtual_base = mmap(NULL,AXI_REG_SPAN,(PROT_READ|PROT_WRITE),MAP_SHARED,fd,AXI_REG_BASE);
    if(axi_virtual_base==MAP_FAILED)
    {
        printf( "ERROR: mmap lw_h2f() failed...\n" );
        close(fd);
        exit (2);
    }

    axi_lw_virtual_base = mmap(NULL,AXI_LW_REG_SPAN,(PROT_READ|PROT_WRITE),MAP_SHARED,fd,AXI_LW_REG_BASE);
    if(axi_lw_virtual_base==MAP_FAILED)
    {
        printf( "ERROR: mmap lw_h2f() failed...\n" );
        close(fd);
        exit (3);
    }
```

Fuente: Elaboración propia en Eclipse ARM DS-5 Development Studio.

Figura 39. Código para el mapeo de los puentes de comunicación.

```
axi_virtual_base = mmap(NULL,AXI_REG_SPAN,(PROT_READ|PROT_WRITE),MAP_SHARED,fd,AXI_REG_BASE);
axi_lw_virtual_base = mmap(NULL,AXI_LW_REG_SPAN,(PROT_READ|PROT_WRITE),MAP_SHARED,fd,AXI_LW_REG_BASE);
```

Fuente: Elaboración propia en Eclipse ARM DS-5 Development Studio.

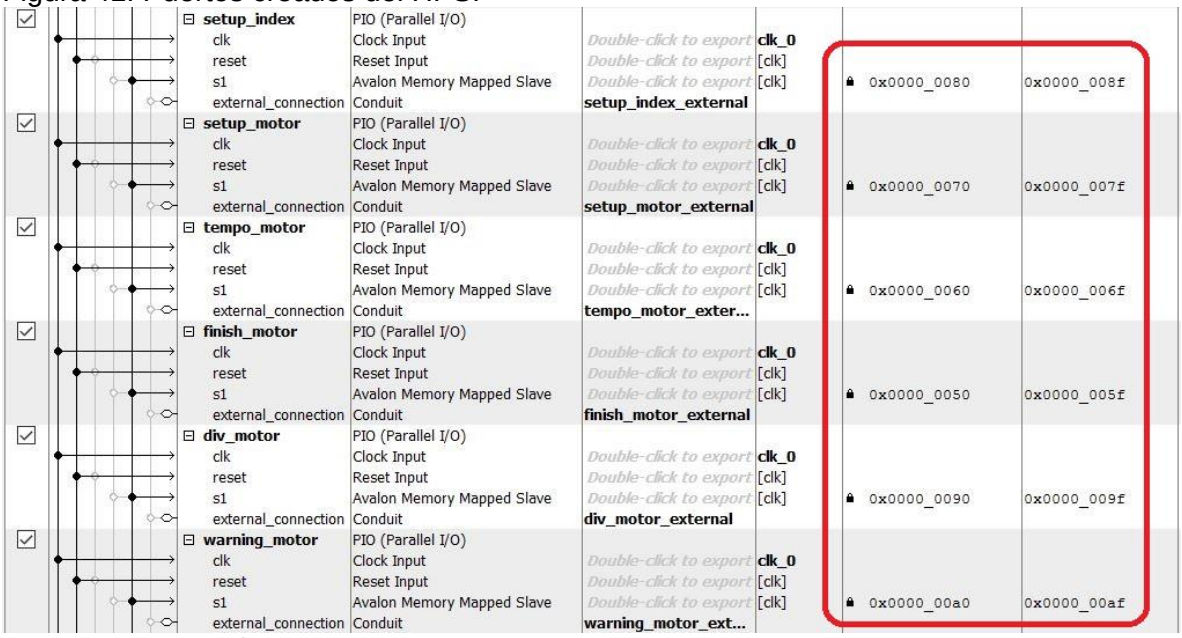
Figura 40. Código de acceso de cada dirección de los puertos del HPS.

```
divfreq_addr        = axi_lw_virtual_base + ((unsigned long)(AXI_LW_REG_BASE+DIV_FREQ_BASE)&(unsigned long)(AXI_LW_REG_MASK));
right_pwm_addr      = axi_lw_virtual_base + ((unsigned long)(AXI_LW_REG_BASE+RIGHT_PWM_BASE)&(unsigned long)(AXI_LW_REG_MASK));
left_pwm_addr       = axi_lw_virtual_base + ((unsigned long)(AXI_LW_REG_BASE+LEFT_PWM_BASE)&(unsigned long)(AXI_LW_REG_MASK));
div_motor_addr      = axi_lw_virtual_base + ((unsigned long)(AXI_LW_REG_BASE+DIV_MOTOR_BASE)&(unsigned long)(AXI_LW_REG_MASK));
setup_index_addr    = axi_lw_virtual_base + ((unsigned long)(AXI_LW_REG_BASE+SETUP_INDEX_BASE)&(unsigned long)(AXI_LW_REG_MASK));
setup_motor_addr    = axi_lw_virtual_base + ((unsigned long)(AXI_LW_REG_BASE+SETUP_MOTOR_BASE)&(unsigned long)(AXI_LW_REG_MASK));
tempo_motor_addr    = axi_lw_virtual_base + ((unsigned long)(AXI_LW_REG_BASE+TEMPO_MOTOR_BASE)&(unsigned long)(AXI_LW_REG_MASK));
finish_motor_addr   = axi_lw_virtual_base + ((unsigned long)(AXI_LW_REG_BASE+FINISH_MOTOR_BASE)&(unsigned long)(AXI_LW_REG_MASK));
warning_motor_addr  = axi_lw_virtual_base + ((unsigned long)(AXI_LW_REG_BASE+WARNING_MOTOR_BASE)&(unsigned long)(AXI_LW_REG_MASK));
ready_motor_addr    = axi_lw_virtual_base + ((unsigned long)(AXI_LW_REG_BASE+READY_MOTOR_BASE)&(unsigned long)(AXI_LW_REG_MASK));
```

Fuente: Elaboración propia en Eclipse ARM DS-5 Development Studio.

Estas direcciones de cada periférico han sido definidas previamente desde el diseño del HPS y la compilación en el Qsys. Dependiendo de la densidad de la información que maneja cada periférico, el HPS asignará una cantidad de bytes asociada a cada dispositivo. En la *Figura 41* se muestran algunos ejemplos de los periféricos añadidos al HPS utilizando los puertos PIO.

Figura 41. Puertos creados del HPS.




Fuente: Elaboración propia en Quartus II.

Cada vez que se quiera realizar una nueva identificación de voz, es necesario reasignar la memoria en donde se guardan las hipótesis realizadas por el HMM. Para ello, se deben reabrir los puertos de comunicación con la SDRAM, así como liberar los recursos usados previamente para evitar tener problemas de overflow en memoria RAM.

Finalmente, cuando el equipo va a ser desconectado o apagado para su uso posterior, este se debe encargar de liberar la memoria mapeada para garantizar que no ocurran inconvenientes la próxima vez que el dispositivo vaya a ser utilizado. Para ello se utilizan las funciones munmap y close antes de dar por terminada la sesión. En la *Figura 42* se visualiza su aplicación desde código C.

Figura 42. Código para vaciar los espacios de memoria utilizados.

```
// desmontar memoria virtual
munmap(virtual_base_lw,REG_SPAN_LW);
munmap(virtual_base,REG_SPAN);
// cerrar archivo, dispositivo dev/mem
close(fd);
printf("Cerrando el programa \n");
return 0;
```

Fuente: Elaboración propia en Eclipse ARM DS-5 Development Studio.

# 9 ADQUISICIÓN DE AUDIO

La adquisición de audio es uno de los procesos más importantes en el proyecto de procesamiento de voz, pues todo parte de la captura de una muestra apta para el procedimiento de detección e identificación de los fonemas estructurales de cada oración.

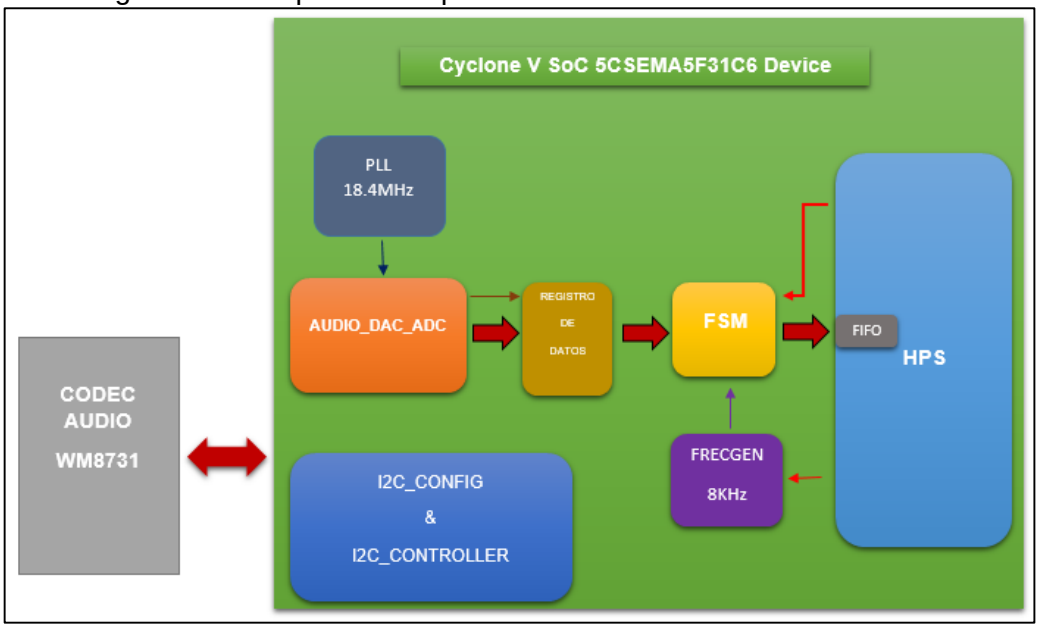
En este bloque de adquisición se implementaron en la FPGA módulos para el manejo de las señales de audio proveniente del micrófono, utilizando protocolos de comunicación I2C para el control del chip WM8731 CODEC AUDIO y la decodificación para determinar el canal derecho e izquierdo.

Se utilizaron los siguientes módulos para esta etapa del sistema:

- I2C_Config
- I2C_Controller
- PLL 18.4 MHz
- Audio_DAC_ADC
- FSM
- FrecGen 8Khz

La función de estos seis módulos nos permite obtener la muestra de audio lista para su procesamiento. En la *Figura 43* se observa un diagrama de bloques que describe la relación entre los módulos implícitos en el procedimiento de adquisición de datos.

Figura 43. Diagrama de bloques de adquisición de datos.




Fuente: Elaboración propia.

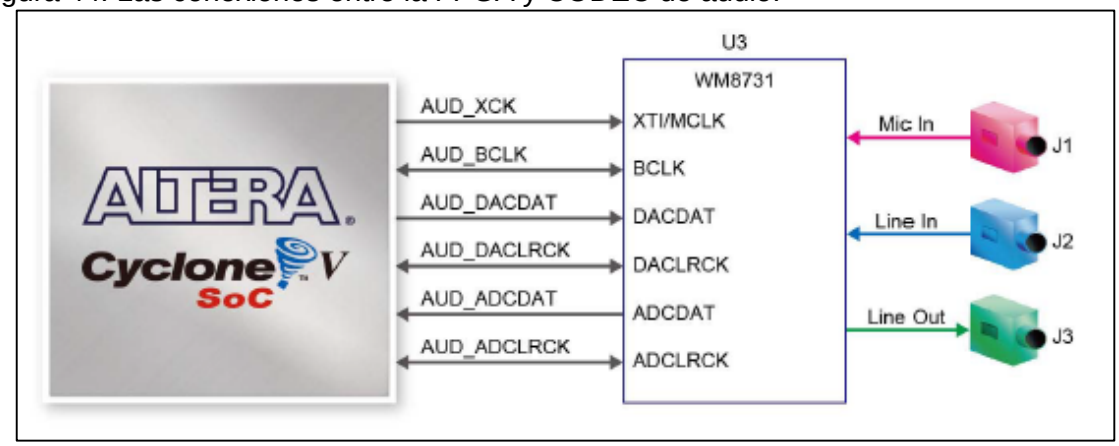
## 9.1 CODEC AUDIO

En la DE1-SoC Board, para utilizar los jacks de audio ya sea para entradas o salidas, se debe configurar previamente el códec de audio, definiendo las características de su modo de operación. Este códec de audio está compuesto por un integrado diseñado por Wolfson, y un módulo de hardware digital que se encarga de hacer la función de controlador.

### 9.1.1 Codec Wolfson WM8731

Los jacks de 3.5 mm de entrada/salida de audio y micrófono de la DE1-SoC están conectados a un Codec de audio Wolfson QM8731 (codificador y decodificador), que ofrece una alta calidad en audio de 24 bits. Este chip es compatible con entrada de micrófono, línea de entrada, línea de salida y frecuencia de muestreo ajustable desde 8 KHz a 96 KHz. El chip WM8731 se configura a través del bus I2C, que es conectado con la FPGA Cyclone V. la asignación de pines asociada a la FPGA aparece en la *Figura 44*. Más información sobre el códec WM8731 está disponible en la hoja de anexos.

Figura 44. Las conexiones entre la FPGA y CODEC de audio.



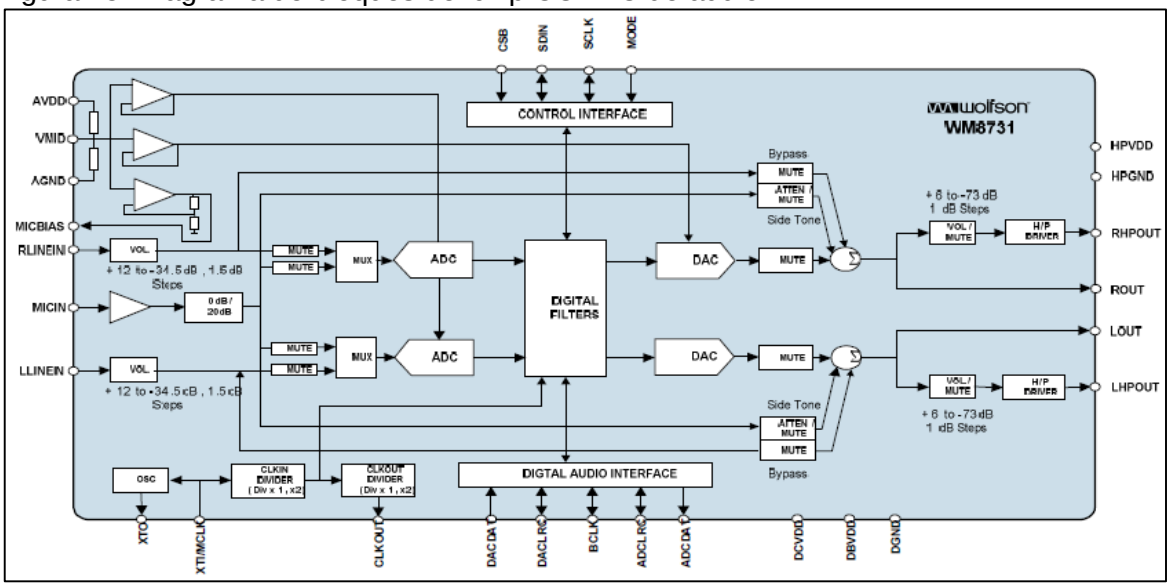

Fuente: TERASIC TECHNOLOGIES. DE1-SoC: User Manual. Taiwan: Terasic Technologies Inc. All Rights reserved: 2015. p.31.

Las señales analógicas de entradas como la del micrófono y línea de audio son convertidas a señales digitales mediante un ADC interno de 12 bits de resolución, para luego pasar a un filtro FIR pasa banda y posteriormente entregar estas muestras a través de la interfaz de control para señales digitales conectada directamente a la FPGA. Estas señales son manejadas mediante la interfaz de control y la interfaz de audio que soporta el chip CODEC. En la *Figura 45* se muestra un diagrama interno del Audio Codec WM8731.

Para el manejo optimo del códec de audio, generalmente se utiliza un módulo programado en lenguaje Verilog conocido como AUDIO_DAC_ADC que terasic

ofrece como una solución al control del chip interno. En el proyecto de reconocimiento de voz se utilizó una adaptación a este código realizada por Bruce Land[43] para la universidad de Cornell, donde suprime los manejos de memoria interna, y además simplifica la interfaz de usuario para hacer del programa una herramienta más intuitiva a la hora de utilizar la codificación de audio en FPGA.

Figura 45. Diagrama de bloques del chip CODEC de audio.




Fuente: Wolfson Microelectronics, datasheet Codec WM8731 Production Data, Abril 2009, Rev 4.8.p.1.

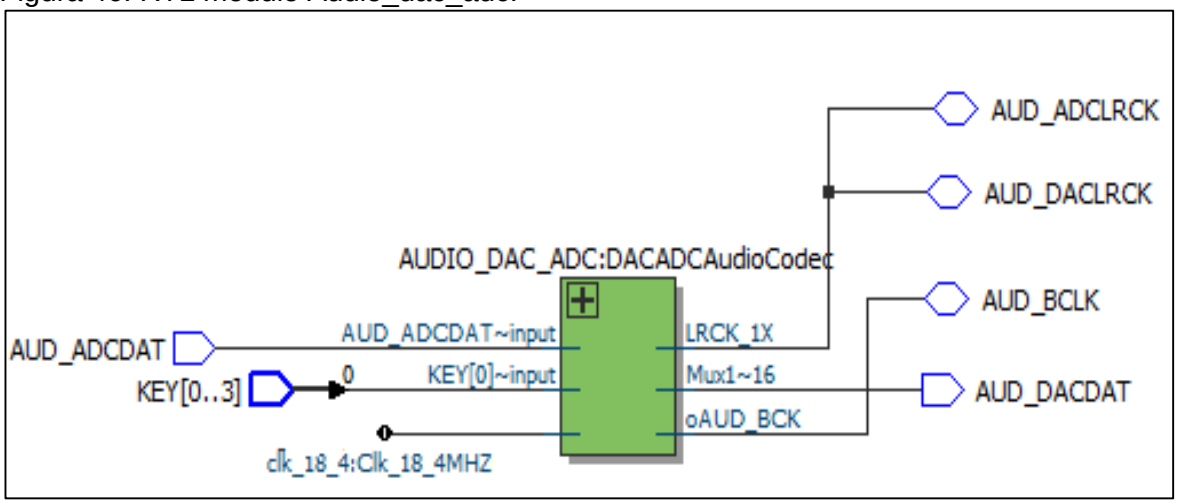
### 9.1.2 Módulo AUDIO_DAC_ADC

El módulo AUDIO_DAC_ADC soporta conversión ADC y DAC para el audio de entrada/salida en tiempo real. Este código trabaja en conjunto con el módulo I2C_Config y son los que permiten el correcto funcionamiento del chip de audio interno del DE1-SoC. Por esta razón, este módulo no es otra cosa que el controlador del códec de audio, y por ello su conexión se realiza directamente sobre la interfaz de entrada que tiene referenciada la FPGA para los pines de este códec. En la Figura *46* se aprecia la conexión interna del módulo AUDIO_DAC_ADC con los pines del codificador de audio.

Este módulo decodifica las señales provenientes del chip WM8731 ubicándolas en puertos de 16 bits de salida para cada canal de manera sincronizada, para ser enviadas al exterior. El controlador de audio tiene la opción de enviar señales de audio internas o generadas desde la FPGA hacia el exterior por medio de los pines de entrada de 16 bits para cada canal que soporta el módulo.

[43] LAND, Bruce. AUDIO_DAC_ADC module. En: Universidad de Cornell. Nueva York, 2007.

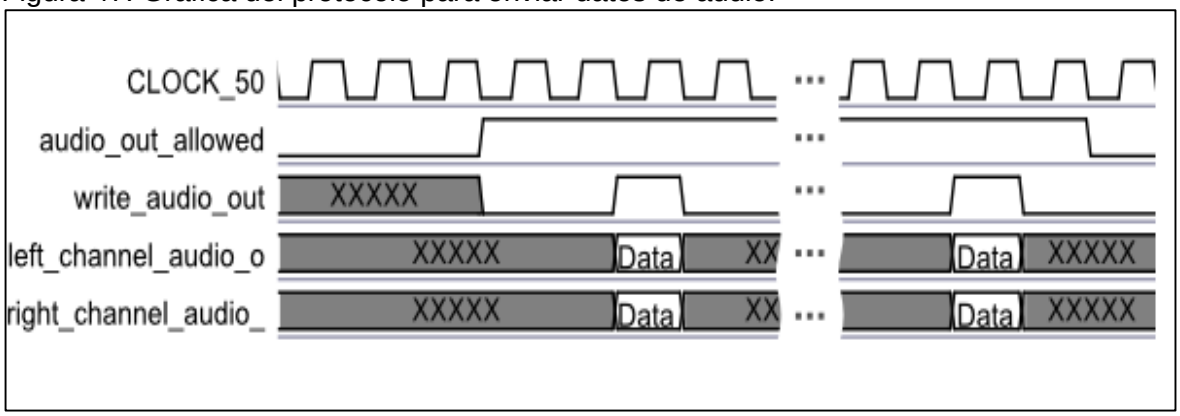
Figura 46. RTL módulo Audio_dac_adc.




Fuente: Elaboración propia en Map Viewer Quartus II

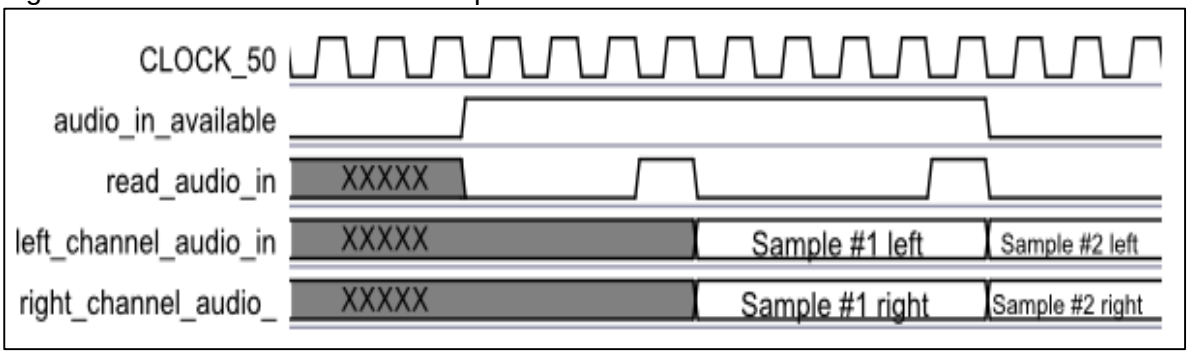
El módulo AUDIO_DAC_ADC tiene la capacidad de mantener una comunicación bidireccional de audio (full-dúplex). Los puertos de datos son de 32 bits de ancho por defecto y se conectan a las memorias intermedias de datos. Esta información se representa como una variable de tipo entero con signo que representa una muestra de audio. Todas las señales se sincronizan con el mismo reloj. El protocolo de interfaz para enviar datos se ilustra en la Figura *47*, y la Figura *48* muestra el protocolo para la recepción de datos de audio.

Figura 47. Gráfica del protocolo para enviar datos de audio.




Fuente: LAND, Bruce. Módulo Audio_DAC_ADC. [en línea]. <http://www.eecg.toronto.edu/~jayar/ece241_08F/AudioVideoCores/audio/audio.htm> [citado en 7 de agosto de 2016]

Figura 48. Gráfica Protocolo de recepción de datos de audio.




Fuente: LAND, Bruce. Módulo Audio_DAC_ADC. [en línea]. <http://www.eecg.toronto.edu/~jayar/ece241_08F/AudioVideoCores/audio/audio.htm> [citado en 7 de agosto de 2016]

El controlador de audio utiliza los flujos de datos PCM, tanto para la entrada como para la salida. El flujo de datos PCM es básicamente una secuencia de números que representan la intensidad de la señal. En cada una de estas secuencias se toma una muestra. El sonido puede ser representado por una secuencia de las muestras. Esta secuencia tiene una frecuencia asociada con él, que cuenta la velocidad a la que se tomaron muestras de la señal original. Esta tasa de muestreo es necesario para la reconstrucción de la señal de forma correcta.

Para el controlador de audio la frecuencia de muestreo predeterminada es de 48 kHz con un tamaño de muestra por defecto de 32 bits. Además, hay 2 canales tanto para la entrada como para la salida. La frecuencia de muestreo y el tamaño de la muestra se pueden cambiar si es necesario. Las muestras de audio se presentan como los números enteros de complemento a 2 con signo.

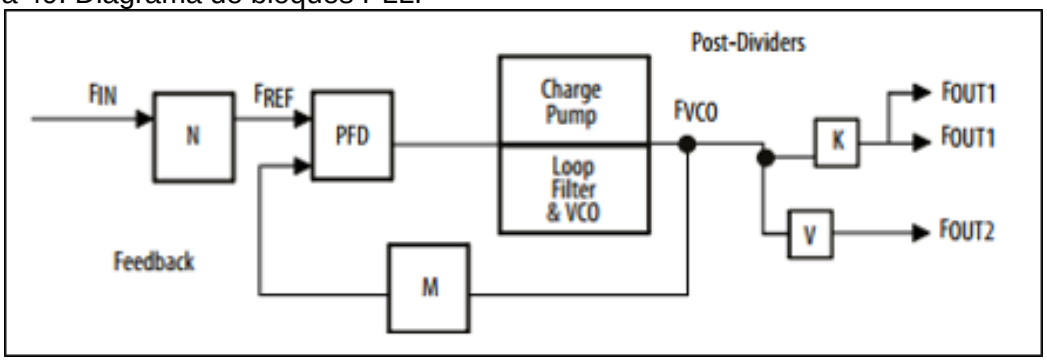
El modulo funciona a partir un reloj de 18.4MHz que controla el generador del reloj base del módulo. Un módulo PLL es creado para generar el pulso de 18.4MHz mediante la función de los IP Cores que ofrece Altera.

Un PLL es un sistema de control de frecuencia de bucle cerrado basado en la diferencia de fase entre la señal de reloj de entrada y la señal de reloj de realimentación de un oscilador controlado.

El PLL consiste en un contador de pre-divisor (N counter), un PFD del circuito, filtro de bucle, un VCO, un contador multiplicador de retroalimentación (contador de M), y post-divisor de contadores (K y V contadores).

Se utilizó un PLL para generar una frecuencia de 18.4Mhz a partir de una frecuencia de 50Mhz, con el fin de cablear este reloj creado hacia el módulo Audio_DAC_ADC (ver Figura *49*).

Figura 49. Diagrama de bloques PLL.




Fuente: © 2014 Altera Corporation.ALTPLL (Phase-Locked Loop) IP Core User Guide: Building Blocks of a PLL, U.S: Altera Corporation, 2008. p.2 (ISO 9001).

## 9.2 CONFIGURACIÓN I2C PARA AUDIO

Para configurar el códec de audio anteriormente descrito, es necesaria una comunicación entre la FPGA y el códec mediante el bus I2C. Por medio de este protocolo se deben enviar los ajustes pertinentes para definir el funcionamiento de algunas características de audio como la habilitación de los canales izquierdo y derecho, el uso del micrófono, el volumen, la atenuación o amplificación de las líneas de entrada, el filtrado de la señal mediante filtros pasa bajo, etc.

El datasheet del códec audio especifica en detalle los parámetros que definen el comportamiento del tratamiento de audio. La programación se realiza mediante la escritura de datos en los registros internos del chip. En el protocolo I2C se envían paquetes de datos de 1 byte (8 bits). Para la configuración de cada uno de los registros es necesario el envío de 2 bytes de información. En total se deben modificar los primeros 10 registros del chip para lograr obtener audio. En la Tabla *3* y la Tabla *4*, los valores de configuración para los canales de audio izquierdo y derecho, así como el control análogo para el control de boost y mute.

Tabla 3. Configuración del I2C para Audio CODEC.

| REGISTER ADDRESS | BIT | LABEL | DEFAULT | DESCRIPTION |
|---|---|---|---|---|
| 0000000<br>Left Line In | 4:0 | LINVOL[4:0] | 10111<br>( 0dB ) | Left Channel Line Input Volume Control<br>11111 = +12dB . . 1.5dB steps down to 00000 = -34.5dB |
| | 7 | LINMUTE | 1 | Left Channel Line Input Mute to ADC<br>1 = Enable Mute<br>0 = Disable Mute |
| | 8 | LRINBOTH | 0 | Left to Right Channel Line Input Volume and Mute Data Load Control<br>1 = Enable Simultaneous Load of LINVOL[4:0] and LINMUTE to RINVOL[4:0] and RINMUTE<br>0 = Disable Simultaneous Load |
| 0000001<br>Right Line In | 4:0 | RINVOL[4:0] | 10111<br>( 0dB ) | Right Channel Line Input Volume Control<br>11111 = +12dB . .1.5dB steps down to 00000 = -34.5dB |
| | 7 | RINMUTE | 1 | Right Channel Line Input Mute to ADC<br>1 = Enable Mute<br>0 = Disable Mute |
| | 8 | RLINBOTH | 0 | Right to Left Channel Line Input Volume and Mute Data Load Control<br>1 = Enable Simultaneous Load of RINVOL[4:0] and RINMUTE to LINVOL[4:0] and LINMUTE<br>0 = Disable Simultaneous Load |

Fuente: Wolfson Microelectronics, datasheet Codec WM8731 Production Data, Abril 2009, Rev 4.8. p.50.

Tabla 4. Configuración del I2C para micrófono.

| REGISTER ADDRESS | BIT | LABEL | DEFAULT | DESCRIPTION |
|---|---|---|---|---|
| 0000100<br>Analogue Audio Path Control | 0 | MICBOOST | 0 | Microphone Input Level Boost<br>1 = Enable Boost<br>0 = Disable Boost |
| | 1 | MUTEMIC | 1 | Line Input Mute to ADC<br>1 = Enable Mute<br>0 = Disable Mute |

Fuente: Wolfson Microelectronics, datasheet Codec WM8731 Production Data, Abril 2009, Rev 4.8. p.51.

### 9.2.1 Módulo I2C_controller

Para desarrollar la comunicación I2C que configura el audio, se utilizó un módulo I2C_controller desarrollado por Terasic para control de I2C en la FPGA. Aquí cabe aclarar que éste módulo permite el envío de datos desde un maestro a un esclavo a través del puerto I2C, pero no permite la lectura de un dispositivo, razón por la cual su aplicación se limita a la caracterización y definición de características, mas no a la comunicación bidireccional con los dispositivos. Este módulo es quien escribe la información directamente sobre el códec de audio.

Este módulo I2C_controller envía como primera medida la dirección única del códec de audio dentro del protocolo I2C. Para la DE1-SoC, esta dirección es la **0x34** en hexadecimal. Una vez enviada dicha dirección, se envía la dirección interna del registro al que se desea escribir. Los registros a modificar van desde el **0x00** hasta el **0x12**. Posterior al envío del registro interno, se envían los datos que se desean escribir para cada uno de ellos.

Para la configuración de los dos canales de la línea de salida (izquierdo y derecho) se requería deshabilitar el mute, deshabilitar la carga de datos simultanea entre canales, y definir un volumen con una ganancia de +4.5 dB. Tomando como referencia la figura, esto se logra mediante el envío del dato **0x01A (000000011010)**, garantizando los bits 7 y 8 en estado bajo (0), y los bits del 0 al 4 en 11010 que garantiza 3 pasos por encima del default (0dB), siendo cada paso 1.5dB.

### 9.2.2 Módulo I2C_config

Para almacenar y administrar el envío de la información, se diseñó un módulo llamado I2C_config, el cual usa al módulo I2C_controller para enviar la información deseada, en el orden determinado por el datasheet del códec audio. Contiene una lógica que garantiza sincronización con los tiempos de reloj del módulo I2C_controller. Ambos módulos para la configuración del códec de audio trabajan a una velocidad de 20 KHz, la cual es relativamente lenta comparada con el resto del proceso. Sin embargo, estos módulos son de un único uso, puesto que su labor es la de configurar el audio para funcionar de determinada manera. Esto quiere decir que una vez que la configuración se concluye, estos módulos se apagan hasta la próxima vez que se encienda la DE1-SoC y se inicie una nueva etapa de configuración. El tiempo que tarda en configurar el chip de audio es de aproximadamente 20ms.

I2C_config. Es el encargado de cargar la información en el puerto del controlador una vez llegue el ciclo de reloj destinado para el envío de información. Primero garantiza la dirección única del códec de audio en el puerto I2C, tal como se ve en la Figura *50*. Esta se encuentra concatenada con la información contenida en un banco de datos con los parámetros en hexadecimal para enviar por el puerto I2C.

Figura 50. Código para la configuración del BUS I2C.


```
//////////////////////////////////////////////////////////////////////
//////////////////////   Config Data LUT    //////////////////////////
//////////////////////////////////////////////////////////////////////
always
begin
    case(LUT_INDEX)
    // Audio Config Data
    SET_LIN_L    :  LUT_DATA <= 16'h001A;
    SET_LIN_R    :  LUT_DATA <= 16'h021A;
    SET_HEAD_L   :  LUT_DATA <= 16'h047B;
    SET_HEAD_R   :  LUT_DATA <= 16'h067B;
//  A_PATH_CTRL  :  LUT_DATA <= 16'h0810;
    A_PATH_CTRL  :  LUT_DATA <= 16'h0815;
//  A_PATH_CTRL  :  LUT_DATA <= 16'h08F8;
    D_PATH_CTRL  :  LUT_DATA <= 16'h0A06;
    POWER_ON     :  LUT_DATA <= 16'h0C00;
    SET_FORMAT   :  LUT_DATA <= 16'h0E01;
    SAMPLE_CTRL  :  LUT_DATA <= 16'h1002;
    SET_ACTIVE   :  LUT_DATA <= 16'h1201;
    // Video Config Data
    default:        LUT_DATA <= 16'hxxxx;
    endcase
end
//////////////////////////////////////////////////////////////////////
```

/// VOL MUTE
/// VOL SOFT
/// VOL ESTANDAR

Fuente: Elaboración propia en Notepad.

Figura 51. Código de asignación de un parámetro I2C.

```
mI2C_DATA    <= {8'h34,LUT_DATA};
```

Fuente: Elaboración propia en Notepad.

Se puede entender este módulo de configuración de audio como una FSM compuesta por 10 estados como se aprecia en la Figura *52*. Cada uno de ellos es el detonante que envía al controlador la información de dirección, registro y dato a través del puerto de conexión interno. Este módulo de configuración envía la información al módulo del controlador, por tanto se puede entender el I2C_controller como un puente entre el módulo que contiene la lógica para definir el funcionamiento del audio, y el chip físico del audio.

Figura 52. Código de los parámetros del módulo I2C_Config.

```
// Clock Setting
parameter   CLK_Freq    =  50000000;   // 50 MHz
parameter   I2C_Freq    =  20000;      // 20 KHz
// LUT Data Number
parameter   LUT_SIZE    =  50;
// Audio Data Index
parameter   SET_LIN_L   =  0;
parameter   SET_LIN_R   =  1;
parameter   SET_HEAD_L  =  2;
parameter   SET_HEAD_R  =  3;
parameter   A_PATH_CTRL =  4;
parameter   D_PATH_CTRL =  5;
parameter   POWER_ON    =  6;
parameter   SET_FORMAT  =  7;
parameter   SAMPLE_CTRL =  8;
parameter   SET_ACTIVE  =  9;
// Video Data Index
parameter   SET_VIDEO   =  10;
```

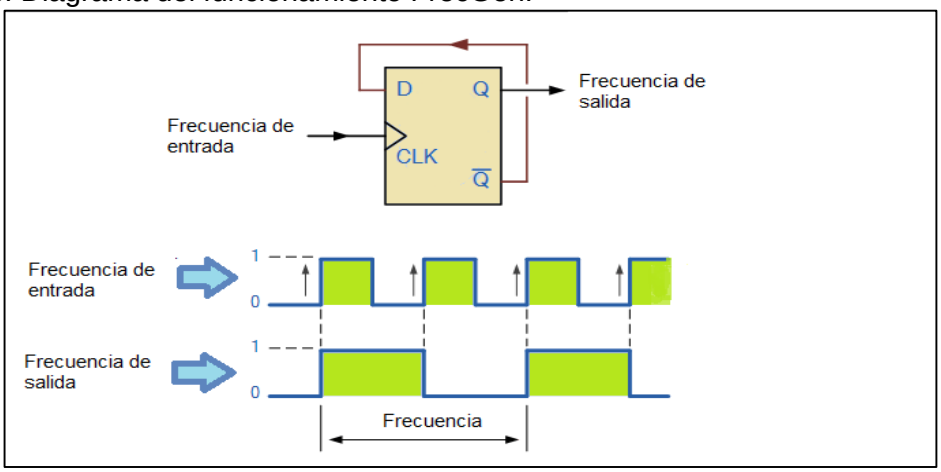
Fuente: Elaboración propia en Notepad.

## 9.3 TRATAMIENTO Y ENVIO DEL AUDIO

Las muestras de audio capturadas en el códec aún deben ser remuestreadas para que el HPS pueda hacer uso de ellas. Generalmente en los sistemas de reconocimiento de voz, las señales de audio se muestrean a una velocidad de 8 KHz, pues el uso de bancos de filtros de la escala de Mel junto con los modelos de Markov demandan un elevado cálculo matemático. Si la frecuencia aumenta, la cantidad de valores de información también se eleva y los tiempos de respuesta del proceso se disparan. Si a esto le añadimos la disponibilidad de un vocabulario extenso, el análisis no es favorable.

Se requiere un sincronizador de 8 KHz, que obligue a los dispositivos involucrados en la comunicación de la FIFO esperar una señal síncrona para operar y permita realizar un submuestreo de la señal de audio proveniente. Para lograr la frecuencia de 8 KHz, se utilizó un módulo FreqGen que permite generar diferentes frecuencias a partir de una señal de reloj. El diseño de un divisor de frecuencia (Frequency Generator) consta de un registro que evita la oscilación en la señal de salida hasta que se cumpla una condición, generalmente un conteo, y permite obtener una frecuencia menor a la frecuencia de referencia como se observa en la Figura *53*.

Figura 53. Diagrama del funcionamiento FrecGen.




Fuente: Elaboración propia.

La ecuación que rige el funcionamiento de un divisor de frecuencia por conteo se observa en la ecuación 14.

$$Dato = \frac{freciencia\ entrada}{2\ x\ frecuencia\ salida} - 1 \qquad (14)$$

Ahora que las muestras de audio tienen una frecuencia de muestreo de 8 KHz, se debe garantizar que el audio no se verá afectado por la velocidad de lectura/escritura de los dispositivos involucrados en la transferencia. El HPS trabaja con un reloj 16 veces más veloz que el de la FPGA, y este desfase entre lectura y escritura puede generar pérdidas en la información muestreada.

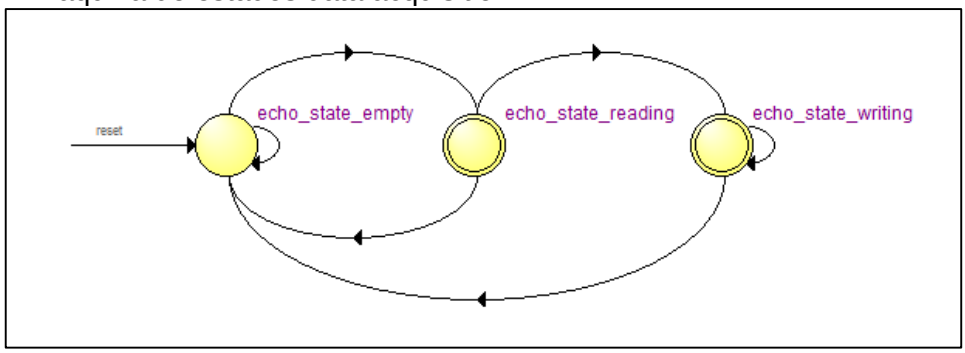
Para solucionar este problema, se plantea el uso de un registro FIFO, donde garantice que el primer dato en entrar sea el primero en salir, a pesar de la diferencia de velocidad en escritura y lectura. Este FIFO es controlado por una FSM que trabaja con los flancos de subida del reloj global de 50 MHz.

Esta FSM se construyó para el control de transferencia en la adquisición de datos. El módulo AUDIO_DAC_ADC entrega las muestras a una memoria de registro de escritura del FIFO. El HPS podrá leer todos los datos de las señales muestreadas desde la segunda memoria del FIFO, la cual estará disponible para lectura siempre que exista algún dato almacenado. Los datos llegarán al FIFO a una velocidad constante, pues lo provee hardware dedicado. Por otra parte, el HPS leerá del FIFO a una velocidad superior pero no constante, puesto que en él corre un sistema operativo con diferentes procesos ejecutándose de manera simultánea.

La FSM por su parte, garantizará que los bloques de memoria operen de manera satisfactoria, mediante los estados de memoria vacía, lectura y escritura. El diagrama de la FSM puede apreciarse en la Figura *54*. En el primer estado se garantiza que el FIFO se encuentre listo para recibir un nuevo dato, y que el

sincronizador entregue el pulso de activación. Mientras esto no suceda, el estado se mantiene. Una vez la condición se cumpla, en el siguiente estado la FSM permite que el paquete de datos correspondiente a una muestra proveniente del módulo AUDIO_DAC_ADC se cargue en la memoria interna del FIFO. En el último estado de la FSM, se permite que el paquete de datos sea leído por algún dispositivo externo (el HPS en este caso). Se mantendrá en ese estado siempre y cuando queden datos almacenados en la memoria interna del FIFO.

Figura 54. Máquina de estados data acquisition.




Fuente: Elaboración propia en State Machine Viewer Quartus II.

Este proceso se repite indefinidamente, cada vez que se requiera enviar un dato muestreado de audio al HPS. Una vez en el HPS, la información digitalizada se encuentra lista para pasar a la etapa de procesamiento digital para identificación de fonemas y palabras. En la Figura *55* se puede observar una simulación en ModelSIM[44] sobre el funcionamiento de la FSM para la adquisición de datos desde el códec de audio.

Figura 55. Simulación del funcionamiento de la FSM  data acquisition.

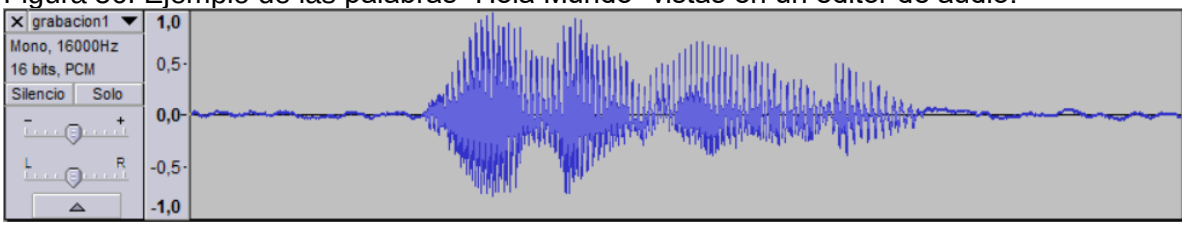

Fuente: Elaboración propia en ModelSim 10.3d.

[44] Altera ModelSIM. Software para desarrolladores proporcionado por altera para la simulación de los procedimientos diseñados para los boards y Familias de FPGA compatibles.

# 10 RECONOCIMIENTO DE VOZ

El habla es un fenómeno complejo. Las personas rara vez se entienden cómo se produce y percibe. Ingenuamente se piensa en la constitución de la lingüística como la construcción de palabras para formar oraciones, y a su vez estas palabras compuestas de sílabas o fonemas. La realidad es, por desgracia, muy diferente. En un contexto más específico, el habla se comprende como un proceso dinámico que carece de partes o estructuras que se puedan ser claramente distinguidas entre sí. Siempre es útil identificar la forma de onda del habla en un editor de sonidos, como en el caso de la Figura *56*, donde se aprecia una muestra de audio en el dominio del tiempo en un editor de audio.

Figura 56. Ejemplo de las palabras “Hola Mundo” vistas en un editor de audio.




Fuente: Elaboración propia en Audacity 2.1.2.

Todas las descripciones modernas del habla son de carácter probabilístico. Esto significa que no hay límites certeros entre las palabras que componen el habla. La transcripción de habla a texto, y demás aplicaciones del reconocimiento de voz, no han llegado a ser nunca 100% correctas e infalibles.

El habla es una corriente continua de audio donde los estados estables se mezclan con los estados dinámicamente cambiantes. En estas secuencias conocidas de estados, se puede definir algunas clases de sonidos similares (fonos). Se entiende una palabra como la unidad construida a partir de varios fonos, pero esto no es del todo cierto. Las propiedades acústicas de la forma de onda correspondiente a un fono puede variar en gran medida dependiendo de varios factores como: el contexto del fono, la persona en cuestión, el estado de ánimo, entre otros. El efecto producido por la coarticulación de los fonos, hace que estos suenen muy diferente a su representación canónica.

Las transiciones entre las palabras tienen un gran valor oculto, pues incluso son más informativas que las regiones estables de las muestras de audio. Se conocen como fonemas, y son las partes de los fonos entre dos fonos consecutivos. A veces algunos desarrolladores de software especializado para reconocimiento de voz hablan acerca de unidades subfonéticas (diferentes sub-estados de un fono). Al menos tres o más regiones de diferente naturaleza pueden ser fácilmente encontradas.

Por ejemplo, el número tres en el idioma inglés “three” es fácil de explicar. La primera parte del fono depende del fono anterior, la parte media es estable y la parte siguiente depende del fono subsiguiente. Este es el por qué existen al menos tres estados en un fono seleccionado para el reconocimiento de voz.

En el idioma inglés, algunas veces los fonos deben ser considerados según el contexto. Este tipo de fonos dentro de un contexto se conocen trifonos. Por ejemplo el fono “u” con un fono a su izquierda “b” y un fono a su derecha “d” componen la palabra “bad”, y aun así este fono “u” suena un poco diferente con un fono “b a la izquierda y un fono “n” a la derecha en la palabra “ban”. Esto muestra que el uso de trifonos está dedicado a describir las pequeñas diferencias sonoras para los sonidos de algunos fonos. La pronunciación de las vocales en ingles varía dependiendo de la palabra en concreto, es por ello que se encuentran más de 42 fonos diferentes para la estructura del idioma.

Tabla 5. Fonemas que componen la articulación lingüística en el lenguaje español.

| Fonema | Definición | Ejemplo |
|---|---|---|
| /a/ | Fonema vocálico de apertura máxima | Alófonos: [a], [ɑ]. |
| /b/ | Fonema obstruyente bilabial sonoro | Grafías: b, v y w, alófonos: [b], [β]. |
| /tʃ/ | Fonema africado palatal | Grafía ch. |
| /d/ | Fonema obstruyente coronal-alveolar sonoro | Alófonos: [d], [ð]. |
| /e/ | Fonema vocálico palatal de apertura media | Alófonos: [e], [ɛ]. |
| /f/ | Fonema labial, fricativo, sordo, oral, en muchas zonas se realiza fricativo bilabial | [ɸ] |
| /g/ | Fonema obstruyente velar sonoro | Grafías g y gu, alófonos: [g], [ɣ]. |
| /i/ | Fonema vocálico palatal y apertura mínima | Alófono usual: [i], en diptongos: [j]. |
| /x/ | Fonema fricativo velar | Grafías g y j, alófonos: [h], [x], [χ], [ħ] en algunas zonas |
| /k/ | Fonema oclusivo velar sordo | Grafías c, qu y k. |
| /l/ | Fonema lateral | Coronal- |
| /m/ | Fonema nasal labial | Alófono usual: [m], ante a una f: [ɱ]. |
| /n/ | Fonema nasal (coronal-)alveola | Alófonos: [n], [ŋ], [ɴ]. |
| /ñ/ | Fonema nasal palatal | [ɲ] |
| /o/ | Fonema vocálico velar de apertura media | Alófonos: [o], [ɔ]. |
| /p/ | Fonema oclusivo | (bi)labial sordo |
| /r/ | Fonema vibrante simple | Grafía -r-, -r |
| /r/(rr) | Fonema vibrante múltiple | Grafía -rr-, r- |
| /s/ | Fonema fricativo (coronal-)alveolar | Grafía s, en algunas variedades z y c |
| /t/ | Fonema oclusivo alveolar sordo | Coronal- |
| /u/ | Fonema vocálico velar de apertura mínima | Alófono usual: [u], en diptongos: [w]. |
| /ʝ/ | Fonema sonorante palatal | Grafía y, ll |

Fuente: MARTÍNEZ, Eugenio. El sonido en la comunicación humana. 2 ed. Barcelona : Octaedro, 2003.

Por otra parte, el idioma español o castellano está compuesto por 5 vocales y 22 consonantes para un total de 27 letras que permiten hasta 22 pronunciaciones fonéticas considerando la estructura general del idioma e ignorando acentos (ver

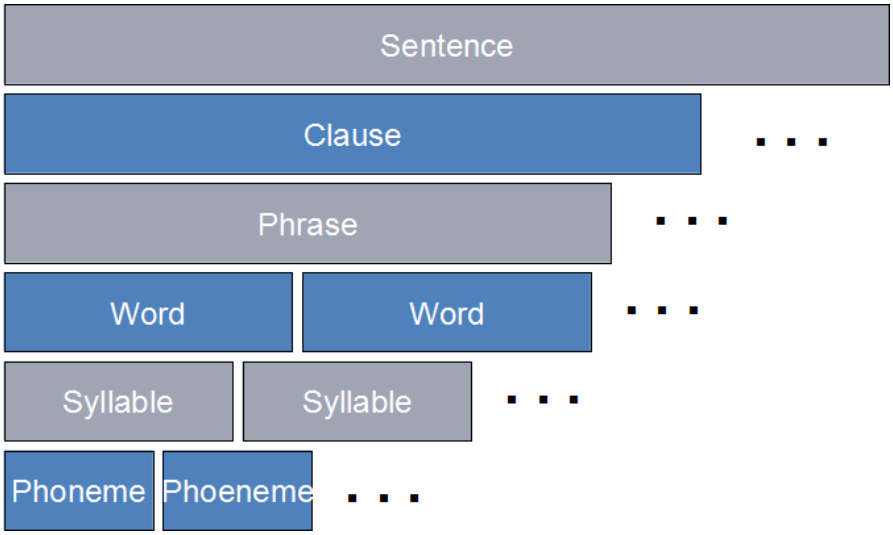
Tabla *5*). Todas las letras que componen el alfabeto del idioma español, están asociadas directamente a un fono, y esto es una característica muy importante que define la manera en cómo se realiza reconocimiento de voz español. Ofrece la ventaja que no requiere, en la mayoría de situaciones, la inclusión de análisis dentro de un contexto específico para la detección de los fonemas. Sin embargo, la ausencia de un reconocimiento contextual aumenta ligeramente la probabilidad de WER.

En cualquier idioma, los fonos permiten construir unidades de sub-palabras como las sílabas, las cuales se definen como “entidades de reducción estable”. Para ilustrar la anterior definición, cuando el habla se vuelve rápida, los fonos pueden cambiar, pero las sílabas siguen siendo las mismas. Así mismo, las sílabas están relacionadas con la entonación. Las unidades sub-palabras se utilizan a menudo en reconocimiento de voz, especialmente cuando se trata de un interpretador de habla con vocabulario abierto.

Las unidades de sub-palabras componen palabras. Estas son muy importantes en el reconocimiento de voz porque se encargan de restringir las posibles combinaciones de los fonos de manera significativa.

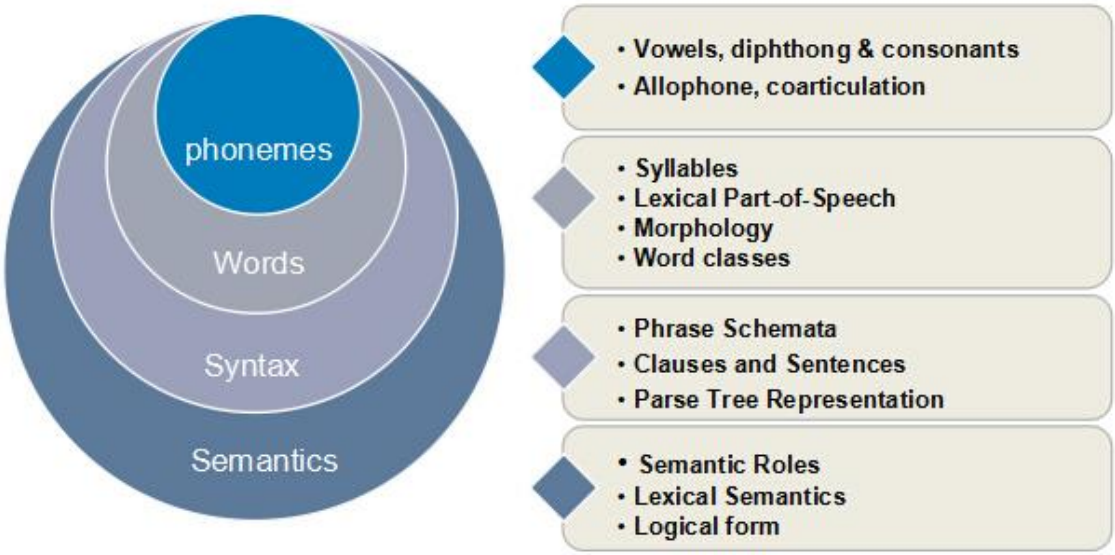
Finalmente las palabras y otros sonidos no lingüísticos como los rellenos (respiración, pausas, muletillas), forman los pequeños enunciados. Estos son fragmentos separados de audio entre las pausas del locutor que pueden conllevar a la formación de oraciones siempre y cuando se considere la existencia de una semántica (ver Figura *58*). En la Figura *57* se observa la estructura de una oración a partir de las unidades subgramaticales que la componen.

Figura 57. Estructura de una oración.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/speech/composition-of-speech> [citado en 17 de septiembre de 2016]

Figura 58. Composición del lenguaje.

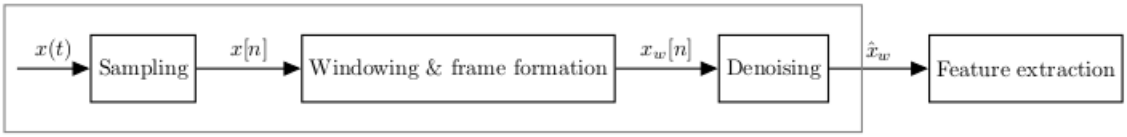



Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/speech/composition-of-speech> [citado en 17 de septiembre de 2016]

## 10.1 PREPROCESAMIENTO

La etapa de preprocesamiento en los sistemas de reconocimiento de voz se utiliza con el fin de aumentar la eficiencia de las etapas de extracción de características y clasificación posteriores y por lo tanto para mejorar el rendimiento global del reconocimiento. Normalmente, el preprocesamiento incluye un primer paso de muestreo, un segundo paso de aplicación de ventanas y una etapa de eliminación de ruido. Al final de la etapa de preprocesamiento, las tramas comprimidas y filtradas del habla son enviadas a la etapa de extracción de características. En la Figura *59* se aprecia un diagrama de los pasos incluidos en la etapa de preprocesamiento del audio.

Figura 59. Estructura general de la etapa de preprocesamiento del audio.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/preprocessing> [citado en 17 de septiembre de 2016]

### 10.1.1 Muestreo

Para que un MCU/MPU pueda procesar una señal de voz, esta primero debe ser digitalizada. Por tanto la señal audible en tiempo continuo debe ser muestreada y cuantificada. El resultado de este procedimiento es una señal discreta en el tiempo. De acuerdo con el teorema de muestreo de Nyquist-Shannon una señal de tiempo continuo $x(t)$ que está limitada a cierta frecuencia finita máxima $F_{max}$ necesita ser muestreada con una $F_s$ de al menos dos veces la frecuencia máxima. De esta manera, la señal puede ser reconstruida a partir de su señal de tiempo discreto $x(s)$. El estudio realizado por SSNDERSON *et al*.[45] Ha demostrado que la frecuencia de muestreo en combinación con el tamaño del vector tiene un efecto directo en la precisión del reconocimiento de voz.

Dado que la voz humana tiene un ancho de banda relativamente bajo (en su gran mayoría entre 100 Hz y 5 KHz), una frecuencia de muestreo de 16 KHz es suficiente para tareas relacionadas con el reconocimiento de voz.

Para el propósito de tener una señal discreta los valores muestreados son cuantificados, lo cual conduce a una reducción significativa de datos. Por lo general los sistemas de reconocimiento de voz codifican las muestras con 8 o 16 bits por muestra en función de la potencia de procesamiento disponible. 8 bits por muestra significaría $2^8 = 256$ niveles de cuantificación, por ende 16 bits por muestra conlleva a un total de $2^{16} = 65536$ niveles de cuantificación. Siempre es preferible trabajar con muestras con una resolución de bits más alta, pero siempre y cuando el hardware a utilizar lo permita.

En la etapa de adquisición de datos, la señal de audio proveniente es muestreada a 16 KHz, con una tasa de 16 bits por muestra. En el reconocimiento de voz se suele utilizar un solo canal de audio.

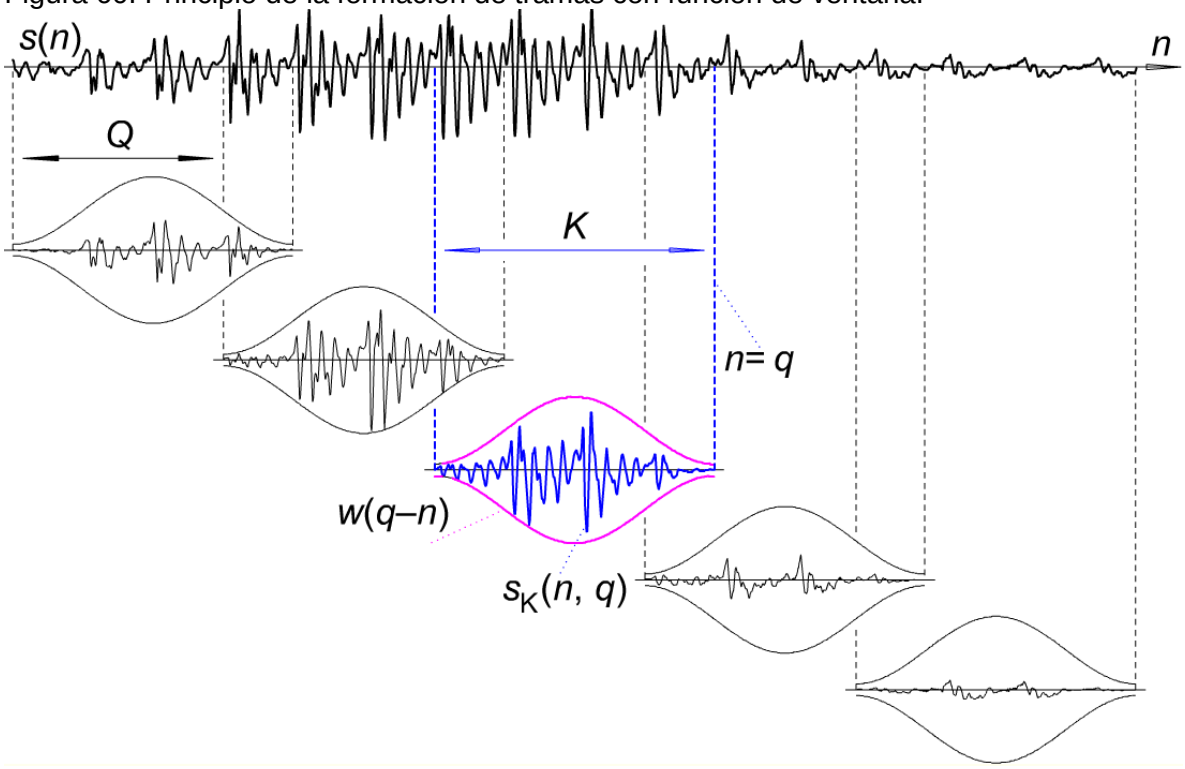
### 10.1.2 Ventana y formación de tramas

El habla es una señal variable en el tiempo no estacionaria. Una señal se considera estacionaria si sus componentes espectrales o de frecuencia no cambian con el tiempo. Se asume que la voz humana se construye a partir de un diccionario de fonemas. La ventaja es que la mayoría de los fonemas mantienen sus propiedades invariantes en el tiempo para periodos cortos (entre 5 ms y 100 ms). Por tanto se asume y se espera que la señal se comporte estacionaria para determinadas tramas dentro de los marcos de tiempo. Con el fin de obtener las tramas se multiplica la señal de voz con una función de ventana. Esta función de ventana sopesa la señal en el dominio del tiempo y la divide en una secuencia de señales parciales. Al dividir

[45] SSNDERSON, C, PALIWAL, K. Effect of different sampling rates and feature vector sizes on speech recognition perfomance. En: IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications. 1997. vol. 1.

la señal se gana información en cada partición, además de facilitar la tarea de extracción de características, la cual se realiza para cada una de las tramas de la señal original. En la Figura *60* se observa la fragmentación de la señal en tramas por medio de una función de ventana, donde $S(n)$ denota la señal de audio muestreada, $Q$ el largo de la trama, $K$ el largo de la ventana, $q$ es el punto de muestra donde la ventana es aplicada y $S_K(n,q)$ es el resultado contenido en el fragmento de la señal.

Figura 60. Principio de la formación de tramas con función de ventana.



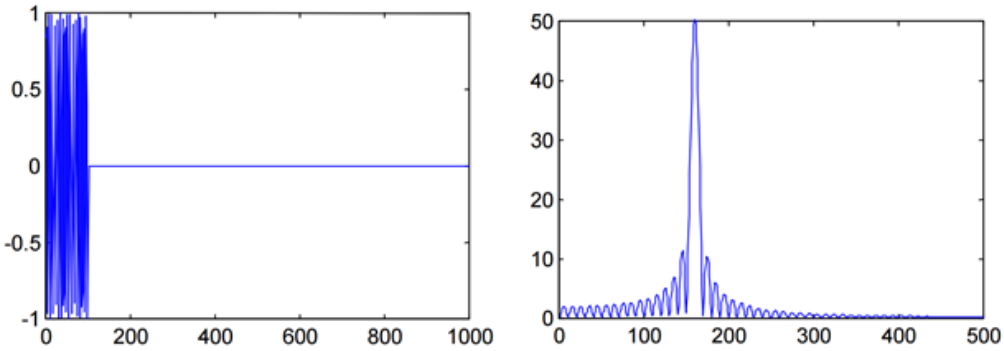

Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/preprocessing> [citado en 17 de septiembre de 2016]

Cuando la señal de divide en tramas cada 20 ms o 25 ms, se genera una pequeña superposición de tiempo $K - Q$. Una superposición pequeña significa mayores cambios en la señal, asi como una potencia de procesamiento menos representativa, pero la diferencia de los parámetros en las tramas vecinas puede ser mucho mayor. Considerando una superposición mayor se generan cambios más suaves en los parámetros de las tramas, aunque requiere de una mayor potencia de procesamiento en la MCU/MPU.

La función de ventana a utilizar debe ser acorde a la capacidad de procesamiento. Es evidente que una ventana rectangular sería la opción más sencilla y de menor demanda para su implementación, pero su uso trae consigo algunos inconvenientes en el dominio de la frecuencia. En la Figura *61* se observa una señal sinusoidal a la que se le ha aplicado una ventana de tipo rectangular, y posteriormente se ha

transformado a frecuencia mediante FFT. En el análisis frecuencial se puede apreciar tanto la frecuencia de la señal sinusoidal como sus armónicos.

Figura 61. Señal sinusoidal a través de una ventana rectangular, transformada en frecuencia mediante FFT.



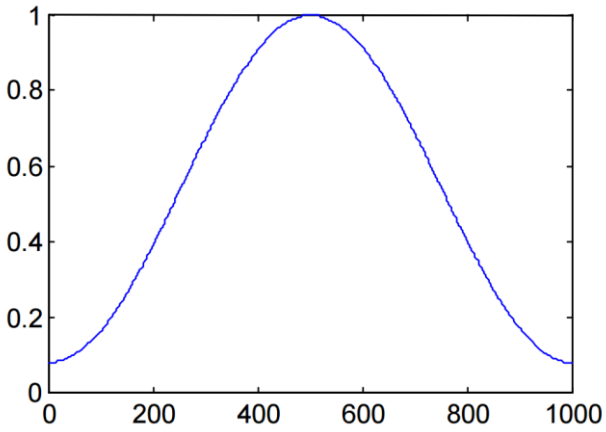

Fuente: elaboración propia en MATLAB R2015a.

El espectro de una señal sinusoidal es un impulso a la frecuencia correspondiente, se observa la deformación de dicho espectro debido al truncamiento temporal y la discontinuidad de la señal en los extremos.

Existen varias funciones de ventanas, cada una con diferentes características, considerando la señal original de diferentes maneras, y por lo tanto, produciendo diferentes señales resultantes. En el procesamiento de voz, sin embargo, la forma de la función de ventana no es algo crucial, y por lo general se busca únicamente reducir las discontinuidades de la señal de audio en los bordes de cada trama. Para tal fin una ventana Von-Hann o Hamming basta. La ventana Hamming está descrita por la ecuación 15, donde $n = 0 \ldots K - 1$, y se puede apreciar en la Figura *62*.

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi(n-1)}{K-1}\right) \quad (15)$$

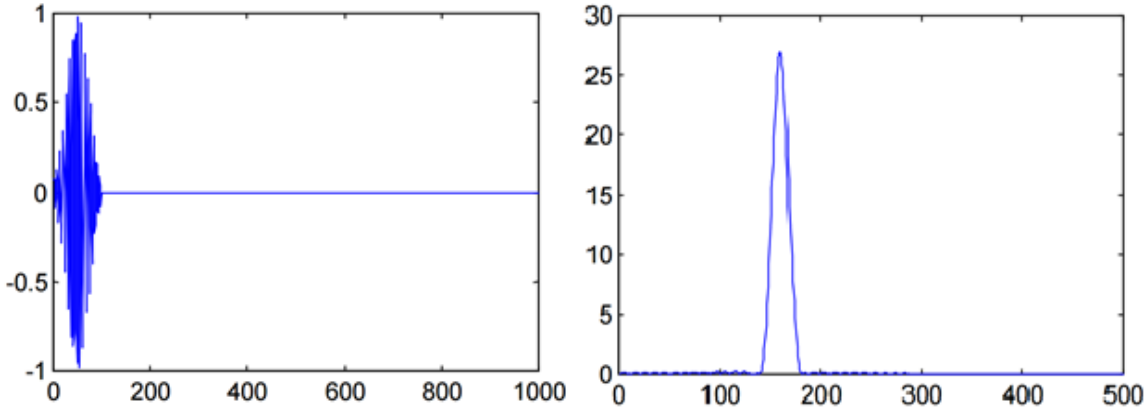
Figura 62. La ventana Hamming.




Fuente: elaboración propia en MATLAB R2015a.

Aplicando la ventana Hamming a la señal sinusoidal del ejemplo anterior, se logra reducir la discontinuidad de los bordes (ver Figura *63*), lo cual es crucial una vez se divida la señal original para evitar duplicar la información en las tramas vecinas cuando ocurra la superposición de las mismas. Por otra parte, favorece el análisis en frecuencia de cada una de las tramas, permitiendo obtener componentes de frecuencia con mayor precisión.

Figura 63. Señal sinusoidal a través de una ventana rectangular, transformada en frecuencia mediante FFT.




Fuente: elaboración propia en MATLAB R2015a.

En el desarrollo del proyecto se realizó la división en tramas cada 45 ms, con una superposición de 20 ms. Es necesario identificar cuales tramas contienen información audible y cuales corresponden a espacios de silencio, esto con el fin de identificar el momento en el que inicia y termina una palabra y además acelerar el proceso de reconocimiento al evitar analizar muestras indeseadas.

El algoritmo utilizado para la detección de las regiones de silencio se encuentra basado en el cálculo de las variaciones de las muestras de señal en cada trama de voz con respecto a su media. Si la variación es representativa, la trama se considera una muestra de voz. El proceso de reconocimiento de regiones de silencio se representa en las ecuaciones 16 y 17, donde $S_k(n)$ son las muestras de la señal de la trama $k$, $\mu$ es la media y $\Omega$ es el resultado de la sumatoria de variaciones.

$$\mu = \frac{1}{N}\sum_{n=1}^{N} S_k(n) \quad (16)$$

$$\Omega = \sum_{n=1}^{N} |S_k(n) - \mu| \quad (17)$$

Finalmente $\Omega$ debe ser comparada con un umbral, que permita determinar si la trama analizada corresponde o no a a una muestra de silencio. Todo valor de $\Omega$ inferior a 7.5 en la escada de SAITO[46], corresponde a una ausencia de sonido.

[46] SAITO, Shuko y TANAKA, Kazuo. Fundamentals of Speech Signal Processing. Academic Press, 1986. 266 p. ISBN: 012-61-4880-5

### 10.1.3 Eliminación de ruido y mejora del habla

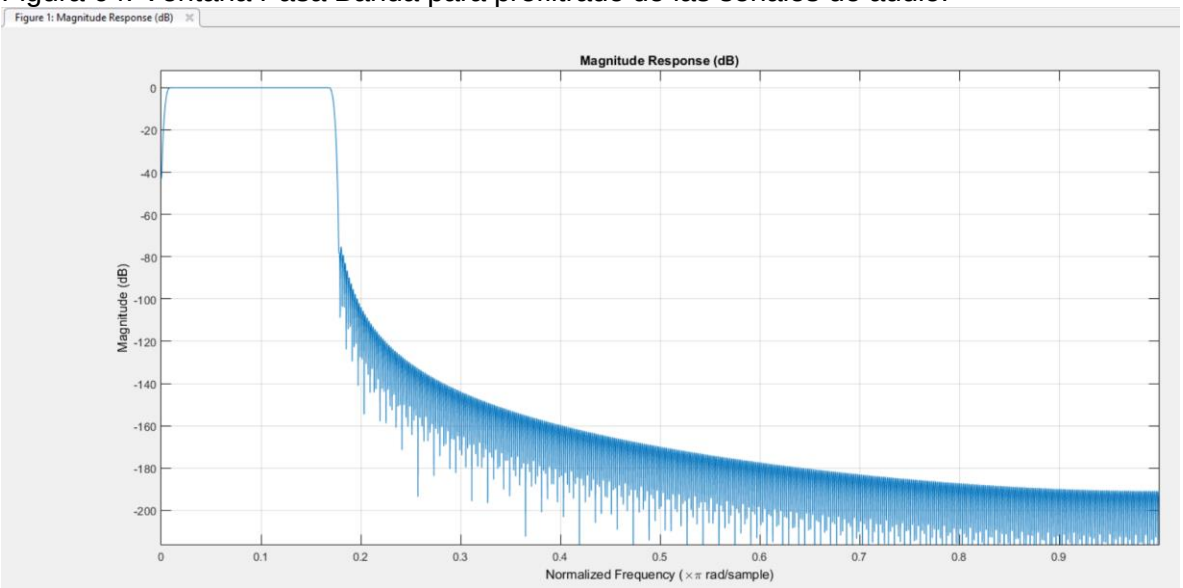
En la etapa de eliminación de ruido se busca mejorar la calidad de las señales de habla provenientes del locutor a través del muestreador, mediante filtrado que permita eliminar perturbaciones comúnmente asociadas a ruido adquirido por el micrófono o de tipo ambiental. Compensar una eliminación de ruido ambiental es un proceso extenso y no elemental, debido a la alta variabilidad de sonidos existentes. Por otra parte, la eliminación del ruido adquirido puede ser implementada sin comprometer la muestra de audio mediante el filtrado de frecuencias no deseadas.

El proceso de filtrado está comprendido por un filtro FIR pasa banda, con frecuencias de corte $\omega_{c1} = 100\ Hz$ y $\omega_{c2} = 3800\ Hz$, con un orden $n = 200$.

En la construcción del filtrado para las tramas, se utilizó una ventana Blackman definida por la ecuación 18, que permite definir la ventana de paso para las frecuencias deseadas. En la Figura *64* se aprecia la respuesta en magnitud para la ventana final de filtrado obtenida por las condiciones del filtro FIR, para una ventana tipo Blackman.

$$0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right) \quad (18)$$

Figura 64. Ventana Pasa Banda para prefiltrado de las señales de audio.

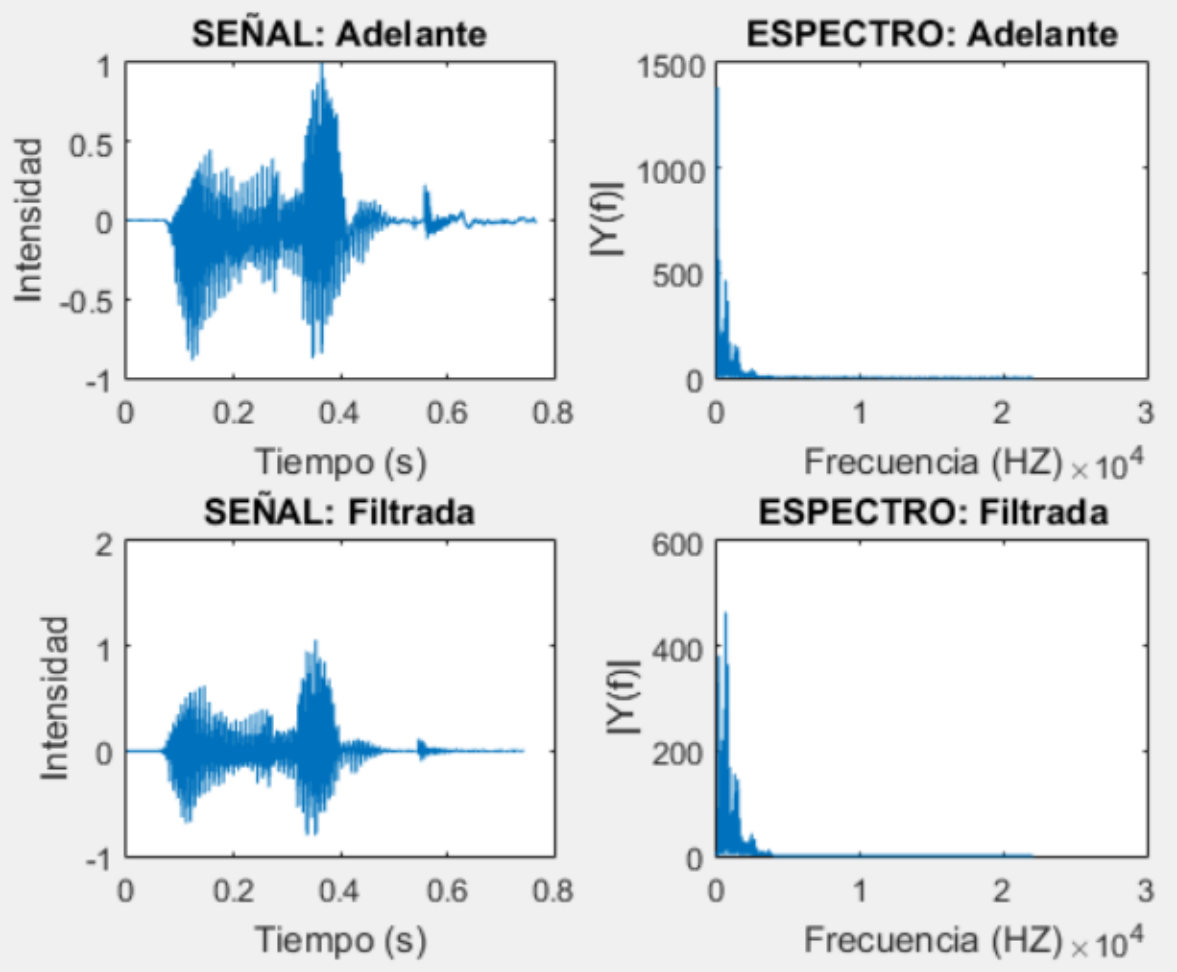



Fuente: elaboración propia en MATLAB R2015a.

Como se aprecia en la *Figura 64*, la atenuación para las frecuencias inferiores a 100 Hz es sensiblemente menos a la atenuación para las frecuencias superiores a 3.8 KHz. Esto se mejora aumentando el orden del filtro, pero requerirá mayor potencia de procesamiento.

En la *Figura 65* se visualiza la representación de la palabra adelante en el tiempo y en frecuencia, antes y después de realizar el proceso de filtrado.

Figura 65. Ejemplo de filtrado para la palabra: adelante.




Fuente: elaboración propia en MATLAB R2015a.

## 10.2 EXTRACCIÓN DE CARACTERISTICAS

Después de la etapa de preprocesamiento, la extracción de características constituye el segundo componente de los sistemas ASR. En este componente se debe derivar características descriptivas de las tramas en las que se ha descompuesto la señal original con el fin de obtener una clasificación de los sonidos involucrados. El objetivo principal de esta etapa según BOURLARD[47] es la extracción de la información del mensaje lingüístico de alta dimensionalidad que se

[47] BOURLARD, H, *et al*. Towards increasing speech recognition error rates. En: Speech Communications. vol. 18. p. 205-231. 1995.

encuentra contenido en los datos en bruto de la señal de audio, mediante algoritmos para la extracción y formación de características de dimensionalidad más baja que permita ser clasificada.

Un vector de características debe enfatizar la información importante relacionada con la tarea específica y suprimir el resto de datos. Dado que el objetivo de un sistema de reconocimiento automático es transcribir el mensaje lingüístico, toda la información característica proveniente de este mensaje debe ser destacada. Las características dependientes del locutor, las características del medio externo y las características del medio utilizado para la grabación de las muestras deben ser suprimidas debido a que estas características no contienen ninguna información relacionada con el mensaje lingüístico. La inclusión de esta información introduciría una variabilidad adicional, que podría traer un impacto negativo sobre la separabilidad de las clases de fonos del modelo de lenguaje. Además, HERMANSKY[48] sugiere que la extracción de características añada también una reducción en la dimensionalidad de los datos muestreados para reducir el tiempo de cálculo y el número de muestras de entrenamiento.

Hasta ahora se han propuesto muchas características diferentes que resaltan diferentes aspectos de la señal de voz. Estas características en su mayoría pueden dividir rasgos en dos grandes grupos: lingüísticos y acústicos.

Las características acústicas solo son relevantes para la clasificación de los sonidos vocales no verbales tales como la risa o los suspiros. Las características lingüísticas son mucho más relevantes para los sistemas ASR porque aquí es donde se intenta realizar la transcripción del mensaje proveniente del locutor. Entre los extractores de características lingüísticas más representativos en la literatura se encuentran:

- Codificación predictiva lineal (LPC)
- Coeficientes de percepción lineal predictiva (PLP)
- Coeficientes cepstrales de la escala de MEL (MFCC)
- Coeficientes cepstrales de predicción lineal (LPCC)

### 10.2.1 MEL-Frequency Cepstral Coefficients

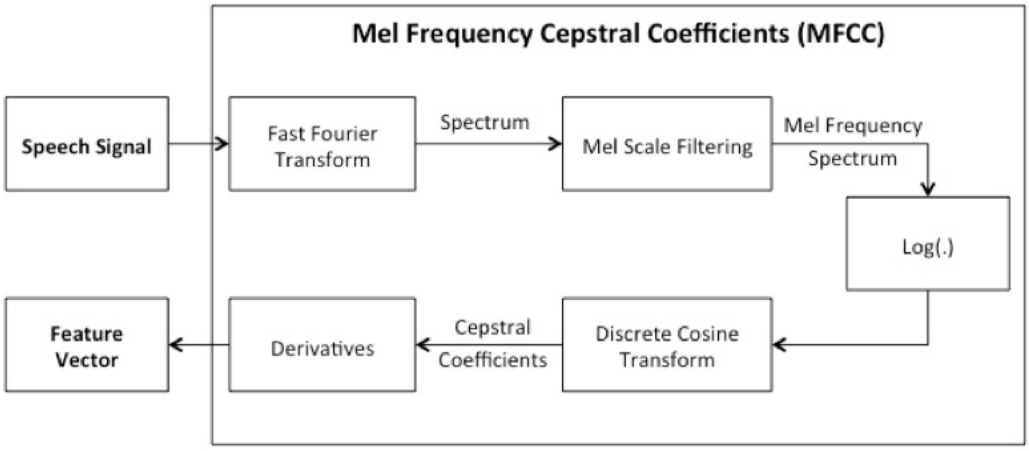
El método de extracción de características más comúnmente utilizado en los sistemas de reconocimiento de voz automático es el MFCC.[49]

[48] HERMANSKY, B, *et al*. Perceptually based linear predictive analysis of speech. En: Acoustics, Speech, and Signal Processing. vol. 10. p. 509-512. 1985

[49] O'SHAUGHNESSSY, D. Automatic speech recognition: History, methods and challenges. En: Pattern Recognition. vol. 41, no. 10. p. 2965-2976. 2008

La implementación estándar para el cálculo de los coeficientes cepstrales de la escala de MEL se muestra en la *Figura 66.*

Figura 66. Diagrama de bloques general de un algoritmo MFCC.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/feature-extraction> [citado en 17 de septiembre de 2016]

### *10.2.1.1 Transformada Rápida de Fourier*

El primer paso del procedimiento para él la obtención de los MFCC consiste en el cálculo de la representación, en el dominio de la frecuencia, de la señal de entrada. Esto se consigue mediante la transformada rápida de Fourier, descrita por la ecuación 19, donde $N$ es el número de puntos de muestreo dentro de una trama de voz en el marco de tiempo $\tau$.

$$C_{\tau,k} = \left|\frac{1}{N}\sum_{j=0}^{N-1} f_j \exp\left[-i\,2\pi\frac{jk}{N}\right]\right| \quad \text{para:} \quad k = 0,1,\dots,\left(\frac{N}{2}-1\right) \tag{19}$$

### *10.2.1.2 Espectro de frecuencia MEL*

En el segundo paso del procesamiento se realiza el cálculo del espectro de frecuencias en la escala de MEL. Por lo tanto, este espectro es filtrado con $N_d$ filtros pasabanda diferentes en donde la potencia de cada banda de frecuencia es recalculada. Este filtrado se basa en la escala de frecuencias de MEL, la cual imita el oído humano, buscando utilizar una potencia sobre una banda de frecuencia como señal de procesamiento. Esta etapa de procesamiento esta descrita por la ecuación 20, donde $d$ s la amplitud del filtro pasa banda con exponente $j$ en la frecuencia $k$.

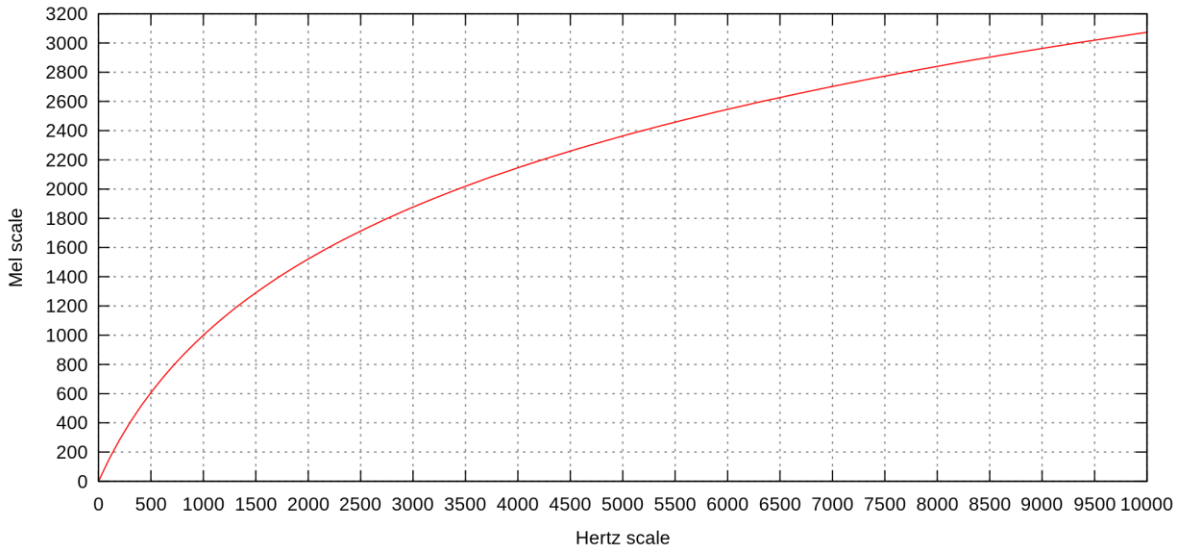
$$C_{\tau,j} = \sum_{k=0}^{\frac{N}{2}-1} d_{j,k}\,.\,C_{\tau,k} \qquad \text{Para:} \quad j = 0,1,\dots,N_d \tag{20}$$

Para realizar el banco de filtros, se considera la escala de frecuencia de MEL, definida por la ecuación 21.

$$m = \frac{1000}{\ln\left(1+\frac{1000}{700}\right)} \ln\left(1+\frac{f}{700}\right) = 2595 \log\left(1+\frac{f}{700}\right) \quad (21)$$

El banco de filtros con los filtros pasa banda no puede igualar al oído, debido a que este puede usar cualquier frecuencia como frecuencia central. Para un sistema ASR se utilizan filtros pasa banda $N_d$ equidistantes según la escala de MEL (ver Figura *67*), la cual es una escala no lineal que se adapta a la percepción del tono no lineal del sistema auditivo humano. Cabe aclarar que tanto el número, la forma y la frecuencia central de los filtros pasa banda que conforman el banco de filtros de MEL pueden variar para adaptarse a cada aplicación.

Figura 67. Gráfico de la escala de MEL.



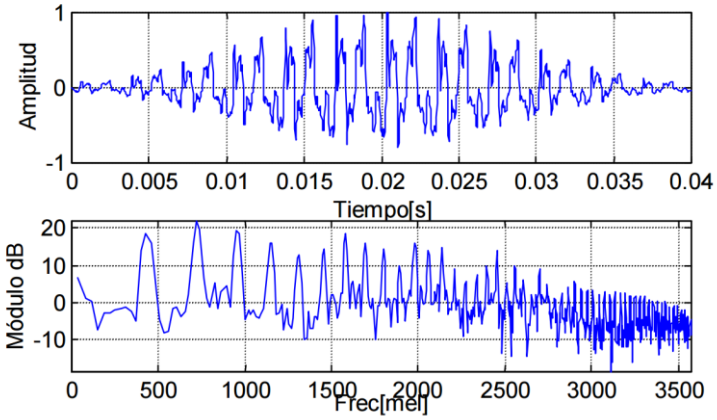

Fuente: DROZDOV, Gilad y ZELIANSKY, Daniel. Media Segmentation for Audio Features Extraction. Israel, 2011. Technion Electrical Engineering Department, Signal & Image Processing LAB.

Cuando se usa la escala de MEL se obtiene el mejor compromiso entre la resolución, frecuencia y tiempo ya que usa un ancho de banda pequeño en baja frecuencia lo que permite resolver armónicos, y en alta frecuencia un ancho de banda más grande que permita buena resolución de ráfagas temporales. En la *Figura 68*, se aprecia el espectro de potencia de una señal $F(t)$ en dB transportado a la escala de Mel.

Una caracterización de la señal vocal que da actualmente grandes resultados es la extracción de los coeficientes cepstrum obtenidos a partir de la escala de Mel, dichos coeficientes se denominan, "Mel-Cepstrum". Para obtenerlos una vez hallado el espectro de la señal vocal, se filtra mediante un banco de filtros en el

dominio de Mel a partir de los cuales se obtienen las correspondientes bandas energía que luego debidamente tratadas formarán parte del cepstrum. A dichos coeficientes se pueden agregar para incrementar su eficiencia, la potencia o logaritmo de la potencia, la primera y segunda derivadas del cepstrum llamadas respectivamente diferencia y aceleración, que aportan información dinámica del proceso.

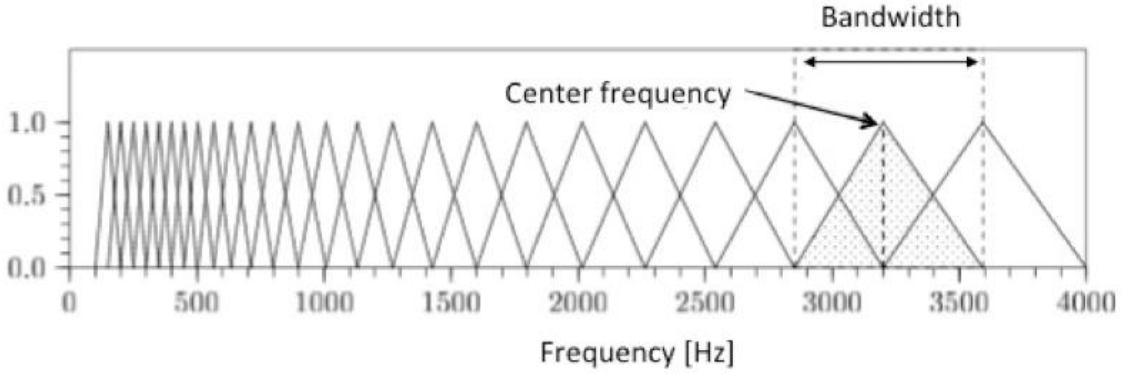
Figura 68. Espectro de una señal F(t) representado en la escala de MEL.




Fuente: elaboración propia en MATLAB R2015a.

En la Figura *69* se aprecia un banco de filtros MEL típico de 25 filtros pasa banda triangulares, donde se aprecia la frecuencia central que involucra cada banco de filtros, así como el ancho de banda asociado a cada uno según la equidistancia de la escala de MEL.

Figura 69. Banco de filtros con un total de 25 filtros pasa banda triangulares para calcular el espectro de la frecuencia de MEL.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/feature-extraction> [citado en 17 de septiembre de 2016]

El banco de filtros desarrollado consta de 12 filtros pasa banda triangulares equiespaciados en la escala de MEL como se muestra en la Figura *70*, puesto que

ZHENG[50] sugiere en su estudio que muy pocos y demasiados filtros pasa banda tienen un impacto negativo en el rendimiento de la clasificación de características, sugiriendo un valor alrededor de los 15 filtros para sistemas ASR.

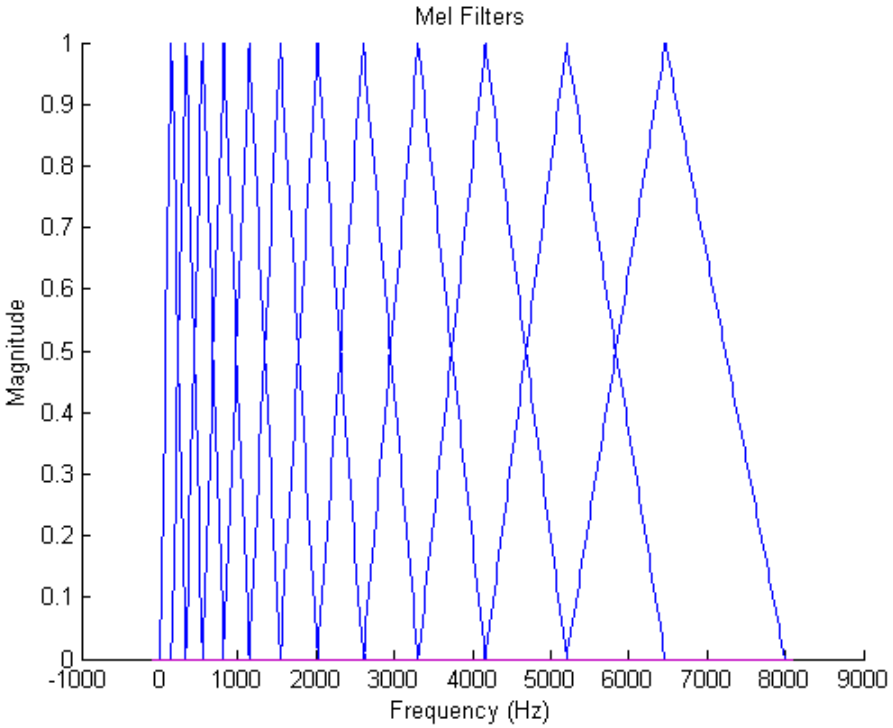
Para el diseño del banco de filtros, se utiliza como rango de frecuencias 300 Hz a 8000 Hz, que son las frecuencias de voz humana que pueden ocurrir en un muestreo a 16 KHz. Utilizando la ecuación 21, se puede obtener que 300 Hz corresponden a 401.25 Mel, así como 8 KHz corresponde a 2834.99 Mel. El banco de filtros debe ser equiespaciado en la escala de MEL, por tanto, los puntos equidistantes que se obtienen son los siguientes:

m(i) = 401.25, 588.46, 775.67, 962.88, 1150.09, 1337.3, 1524.51, 1711.72, 1898.93, 2086.14, 2273.35, 2460.56, 2647.77, 2834.98

Utilizando la ecuación 21, se retornan los valores de la escala de MEL a su valor correspondiente en frecuencia:

h(i) = 300, 479.95, 693.18, 944.94, 1242.19, 1593.16, 2007.5, 2496.8, 3074.53, 3756.61, 4561.96, 5512.84, 6634.9, 8000

Figura 70. Banco de 10 filtros pasa banda triangulares equiespaciados en la escala de MEL.




Fuente: elaboración propia en MATLAB R2015a.

[50] ZHENG, F, ZHANG, G y SONG, Z. Comparison of Different Implementatios of MFCC. En: Journal of Computer Science & Technology. vol. 16, p. 582-589. 2001

De esta manera se logra no sólo aproximar la escala Mel sino reducir de una manera significativa la cantidad de información a procesar en posteriores etapas. Por último, suele usarse en el campo del tratamiento de voz, ya sea para codificación o reconocimiento, la transformada del coseno, usada por su propiedad de compresión de energía, lo que otorga un mejor modelo hacia las frecuencias bajas que en definitiva es la información correspondiente al tracto vocal.

#### *10.2.1.3 Logaritmo*

La tercera etapa de procesamiento calcula el logaritmo de la señal, para imitar la percepción de la sonoridad humana, y además simplificar el cálculo matemático implícito en el procedimiento, ya que permite pasar de una convolución de difícil solución a una suma de funciones logarítmicas. La función logarítmica se aplica al resultado actual del banco de filtros, como lo describe la ecuación 22.

$$C_{\tau,j} = \log\left(C_{\tau,j}\right) \qquad \text{Para: } j = 0,1,\dots,N_d \tag{22}$$

#### *10.2.1.4 Coeficientes Cepstrales*

La cuarta etapa de procesamiento busca eliminar las características dependientes del locutor mediante el cálculo de los coeficientes cepstrales. A partir del modelo Source-Filter se sabe que una señal de voz $S$ se puede descomponer en una excitación $E$ y un filtro lineal caracterizado por su respuesta en frecuencia $H(f)$, lo cual en el dominio del tiempo se puede expresar como una convolución descrita en la ecuación 23, donde $H$ es la respuesya al impulso del filtro.

$$S(t) = H(t) * E(t) \tag{23}$$

Esta expresión se puede expresar como una multiplicación en el dominio de la frecuencia, como se describe en la ecuación 24.

$$S(f) = H(f)\,E(f) \qquad \text{Siendo: } S(f) = |S(f)|e^{j\varphi} \tag{24}$$

El cepstrum se calcula con el fin de suprimir la señal de la fuente. Este puede ser interpretado como el espectro de un espectro, por tanto, los armónicos de la frecuencia central dependiente del locutor son transformados a un coeficiente Cepstral de mayor orden en condiciones ideales (ver Figura *71-b*). La transformada inversa de los coeficientes cepstrales de menor orden muestran la respuesta en frecuencia del tracto vocal (ver Figura *71-c*), y la transformada inversa de los coeficientes cepstrales de mayor orden el espectro de frecuencia de la señal original. Por lo tanto, los armónicos dependientes son suprimidos mediante la adopción de los coeficientes cepstrales de orden inferior para su posterior procesamiento. El cepstrum de una señal se calcula mediante la ecuación 25, donde $f$ es la señal de entrada y $F$ es la transformada de Fourier.

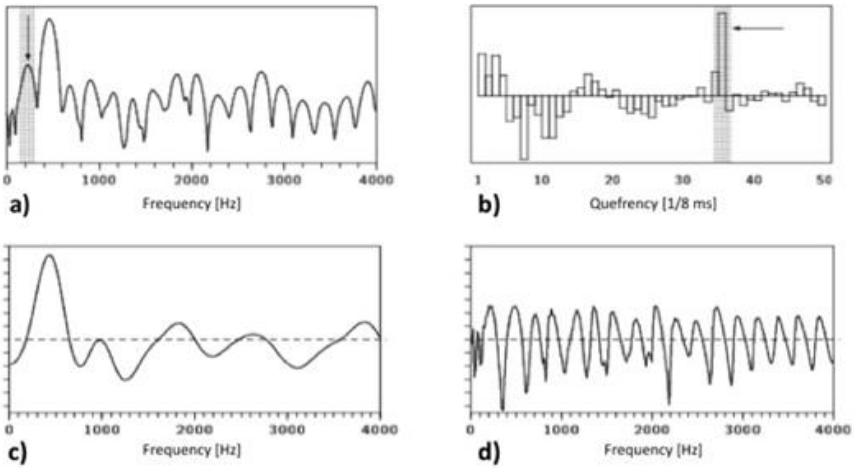
$$F^{-1}\{\log(F\{f_n\})\} \tag{25}$$

Finalmente los coeficientes cepstrales se calculan mediante la ecuación 26, donde $N_{mc}$ es el número de coeficientes cepstrales escogidos para el procesamiento. Por lo general, $N_{mc}$ suele ser un valor entre 13 y 20.

$$C_{\tau,j} = \sum_{j=1}^{N_d} C_{\tau,j} \cos\left[\frac{k(2j-1)\pi}{2N_d}\right] \qquad \text{Para: } k = 0,1,\dots,N_{mc} < N_d \tag{26}$$

Figura *71:* a) Espectro de densidad de potencia logarítmica de una señal de voz. La frecuencia resaltada corresponde a la frecuencia fundamental del locutor. b) Coeficientes cepstrales de una señal de voz. La "quefrency" resaltada es la transformada de la frecuencia fundamental y sus correspondientes armónicos. c) Espectro de densidad de potencia logarítmica de la transformada inversa de los coeficientes cepstrales, aplicado un filtro pasa bajo. "quefrency" de corte $q = 20\ ms$. d) Espectro de densidad de potencia logarítmica de la transformada inversa de los coeficientes cepstrales, aplicado un filtro pasa alto. "quefrency" de corte $q = 20\ ms$.

Figura 71. Figuras ejemplo para el cálculo de los coeficientes cepstrales.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/feature-extraction> [citado en 17 de septiembre de 2016]

### *10.2.1.5 Derivados*

Todas las etapas de procesamiento anteriores incluyen información acerca de la trama actual de la señal. Para representar la naturaleza dinámica del habla, las derivadas de primer y segundo orden de los coeficientes cepstrales (ver ecuaciones 27 y 28) amplían el vector de características. Finalmente el vector de características se representa mediante la ecuación 29.

$$\Delta C_{\tau,j} = C_{\tau+1,j} - C_{\tau-1,j} \tag{27}$$

$$\Delta\Delta C_{\tau,j} = \Delta C_{\tau+1,j} - \Delta C_{\tau-1,j} \tag{28}$$

$$C_\tau = \left[C_{\tau,j}\ , \Delta C_{\tau,j}\ , \Delta\Delta C_{\tau,j}\ \right] \tag{29}$$

El vector de características MFCC se calculó a partir de una ventana de 512 puntos de muestra y un total de 10 coeficientes cepstrales. Según la ecuación 29, en total se tienen 30 coeficientes cepstrales (correspondientes a los 10 coeficientes y sus derivadas de primer y segundo orden).

### 10.2.2 LIMITACIONES

A pesar de que los vectores de características MFCC se utilizan comúnmente en los sistemas ASR estos poseen algunas limitaciones, la mayoría asociadas a la computación de los coeficientes cepstrales.

Una suposición critica de los coeficientes cepstrales es que la frecuencia fundamental es mucho menor que las componentes de frecuencia que componen el mensaje lingüístico. Se requiere de esta hipótesis porque de lo contrario la exclusión de la frecuencia fundamental y sus armónicos no sería posible. Sin embargo, muchas de las voces femeninas no satisfacen esta suposición. Por lo tanto, se desconoce si las características dependientes del locutor pueden ser suprimidas para todas las personas.

Otra limitación de los MFCC es su falta de interpretación. Sólo los dos primeros coeficientes $C_0$ y $C_1$ tienen una interpretación significativa. $C_0$ es la potencia sobre todas las bandas de frecuencia y $C_1$ es el balance entre las componentes de frecuencia altas y bajas dentro de las tramas de la señal. Los otros coeficientes cepstrales no tienen una interpretación clara.

### 10.2.3 CUANTIFICACIÓN VECTORIAL

Una parte importante en cualquier tipo de procesamiento de voz es la optimización de los algoritmos en cuanto a velocidad y almacenamiento, entonces, la cuantificación de vectores trae consigo la idea de clasificar un conjunto de vectores, entre los cuales se buscarán los mejores representantes para reducir el tamaño de la información a manejar. La forma de medir la fidelidad de un cuantificador es determinar el error que éste produce al reemplazar los datos de entrada que recibe por los vectores representantes o codewords, dicho parámetro es llamado error por distorsión. La finalidad de un cuantificador es obtener un conjunto de vectores representativos llamado codebook, que presente el menor error por distorsión, por ejemplo para cuantificar los vectores de observación.

Las técnicas de parametrización de la señal de voz se realizan tomando una secuencia de ventanas temporales, cada una de las cuales se representa por un número de $D$ parámetros. La información de cada ventana se representaría por un vector de observación de $D$ posiciones. Cuando se almacenan estos parámetros se cuantifica cada uno de ellos con un determinado número de bits, este proceso se denomina cuantificación escalar y no es la manera más eficiente para almacenar la

información, además, implica la ocurrencia uniforme de las ventanas de información. Una forma más conveniente es realizar una cuantificación vectorial.

Comparando la información del vector representante con respecto a la forma de onda original de la señal de voz, se concluye que el análisis espectral contiene significativamente menos información. Por ejemplo, una señal de voz se muestrea a 10Khz y la cuantificación es de 16 bits, se necesita una velocidad de 160000 bps para almacenar las muestras de la señal de voz en el formato original. Si se realiza el análisis en el espectro, se tienen en consideración vectores de dimensión n=10 usando 100 vectores de observación por segundo. Si se representa cada parámetro en 16 bits, se requiere aproximadamente 100x10x16 bps o 16000 bps con una reducción de diez veces sobre la señal original.

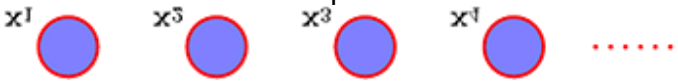
Las compresiones en ancho de banda y almacenamiento son imprevisibles, se basan en el concepto de la necesidad de la representación única para cada fonema (sonido diferenciable de una lengua, generalmente representado por una letra), esto puede ser posible para reducir la representación espectral original de la señal de voz sacando a los vectores de observación desde un pequeño, finito número de vectores espectrales únicos, donde cada uno corresponde a las unidades básicas de la voz o fonemas. La representación ideal es impracticable porque hay mucha variabilidad en las propiedades espectrales de cada uno de los fonemas. De cualquier forma, el concepto de construir un codebook de vectores de análisis, "distintos" y "únicos", aunque con más palabras de código que el grupo o set básico de fonemas, sigue siendo una idea atractiva y es el fundamento de un conjunto de técnicas denominadas métodos de cuantificación de vectores. Basándose en este razonamiento, se necesita un codebook con aproximadamente 1024 vectores espectrales únicos (24 variantes para cada uno de los 22 fonemas básicos). Si para representar un vector espectral arbitrario se tiene un número de 10 bits, tomando una velocidad de 100 vectores por segundo, se obtiene una velocidad de 1000 bps para representar los vectores espectrales de una señal de voz. Esta velocidad es aproximadamente 1/16 de la velocidad necesaria para procesar vectores espectrales continuos. Por lo tanto la representación cuantificada es eficiente para representar información espectral de la señal de voz.

## 10.3 MODELO ESTADISTICO DE MARKOV

### 10.3.1 Proceso estocástico

Un proceso estocástico se define como una secuencia de variables aleatorias. La variable aleatoria se llama estado del proceso en el tiempo. Se asume un proceso de tiempo discreto, es decir, el estado sólo se observa en determinados instantes de tiempo. El ejemplo más simple de un proceso de este tipo es una secuencia de observaciones independientes. (ver Figura *72*)

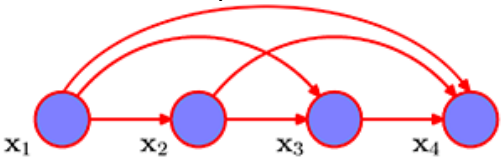
Figura 72. Secuencia de observaciones independientes.



Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/introduction-to-hidden-markov-models> [citado en 17 de septiembre de 2016]

Asumiendo todas las observaciones independientes entre sí, se obtendría una perdida de toda la información dependiente del tiempo. Para superar esto, se considera un proceso en el cual el estado actual depende exclusivamente de los estados anteriores (ver Figura *73*). Para describir completamente este proceso, es necesario definir la distribución de probabilidad para el estado inicial $p(x1)$ así como para todos los siguientes estados $n = 2,3, \ldots$ la distribución condicional $p(x_n|x_1, \ldots, x_{n-1})$.

Figura 73. Secuencia de observaciones dependientes.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/introduction-to-hidden-markov-models> [citado en 17 de septiembre de 2016]

Como se aprecia en la Figura *73*, con un número creciente de estados rápidamente se genera confusión por la gran cantidad de dependencias. Por ello, un modelo donde cada estado depende de todos los estados anteriores no es factible en un sistema práctico.

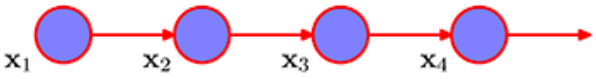
### 10.3.2 Modelo de Markov

Un tipo especial de proceso estocástico es el llamado modelo de Markov. Posee una propiedad conveniente: la probabilidad condicional del estado actual $n$ depende únicamente del estado anterior, regido por la ecuación 30.

$$p(x_n|n_1, \ldots, x_{n-1}) = p(x_n|x_{n-1}) \quad (30)$$

Como se aprecia en la *Figura 74*, el modelo permanece simple aun cuando el número de estados incrementa.

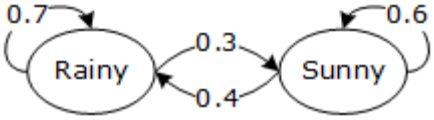
Figura 74. Diagrama de un modelo de Markov.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/introduction-to-hidden-markov-models> [citado en 17 de septiembre de 2016]

Ahora se supone que se quiere predecir el clima con un modelo de Markov. Este modelo debería tener dos estados: lluvioso y soleado. Se sabe que después de un día de lluvia, la probabilidad de que el día siguiente sea soleado será del 30%. Después de un día de sol, la probabilidad de que el día siguiente sea lluvioso dera del 40%. El diagrama de estado para este ejemplo se observa en la Figura *75*.

Figura 75. Diagrama de estados para el ejemplo de un modelo de Markov.



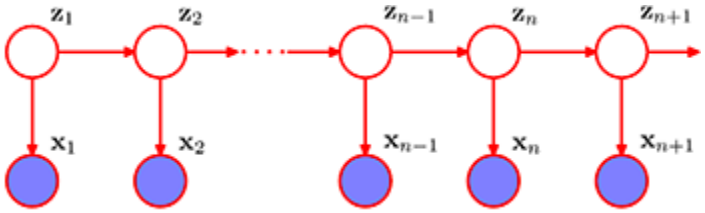

Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/introduction-to-hidden-markov-models> [citado en 17 de septiembre de 2016]

Basado en el clima del día actual, ahora es posible hacer una predicción del clima para el día siguiente utilizando un modelo de Markov.

### 10.3.3 Modelo oculto de Markov

El modelo de Markov está bastante bien para el ejemplo del clima anterior. Sin embargo, este no es el caso para muchas otras aplicaciones. Por lo tanto, es necesario extender el modelo hacia un HMM. Este modelo oculto consiste en un proceso de Markov con estados $z_1, z_2, \ldots$ que no pueden ser observados directamente. Lo que se puede apreciar son las emisiones $x_1, x_2$. La emisión $x_n$ es una función probabilística del estado $z_n$. Una suposición importante inherente en los modelos ocultos de Markov es que el estado actual contiene toda la informacion acerca de las observaciones anteriores (ver Figura *76*), sin necesidad de depender de ellas.

Figura 76. Representación modelo oculto de Markov.

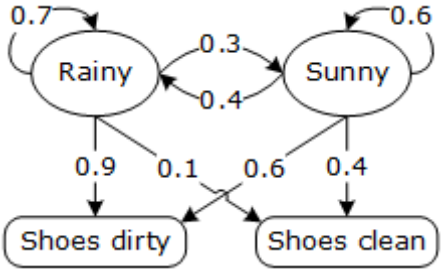



Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/introduction-to-hidden-markov-models> [citado en 17 de septiembre de 2016]

Retomando el ejemplo del clima, ahora se busca ampliarlo para un modelo oculto de Markov. Ahora hay un prisionero encerrado sin ventanas. Este observador no puede ver si el día se encuentra lluvioso o soleado. Sin embargo, puede observar si los zapatos de los guardias están sucios o limpios. Él sabe que si esta lloviendo, los zapatos de los guardias estarán sucios en el 90% de los casos, pero cuando

hace sol estarán sucios solamente con una probabilidad del 60%. El diagrama de espacio de los estados se aprecia en la Figura *77*.

Figura 77. Diagrama de estados para el ejemplo de un modelo oculto de Markov.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/introduction-to-hidden-markov-models> [citado en 17 de septiembre de 2016]

Tal y como se aprecia, el prisionero puede explotar este conocimiento y hacer predicciones acerca del clima exterior, con sólo mirar los zapatos de los guardias. Existen muchas aplicaciones para los HMM. A menudo se utilizan como modelo para las aplicaciones de reconocimiento de patrones, como los sistemas ASR.

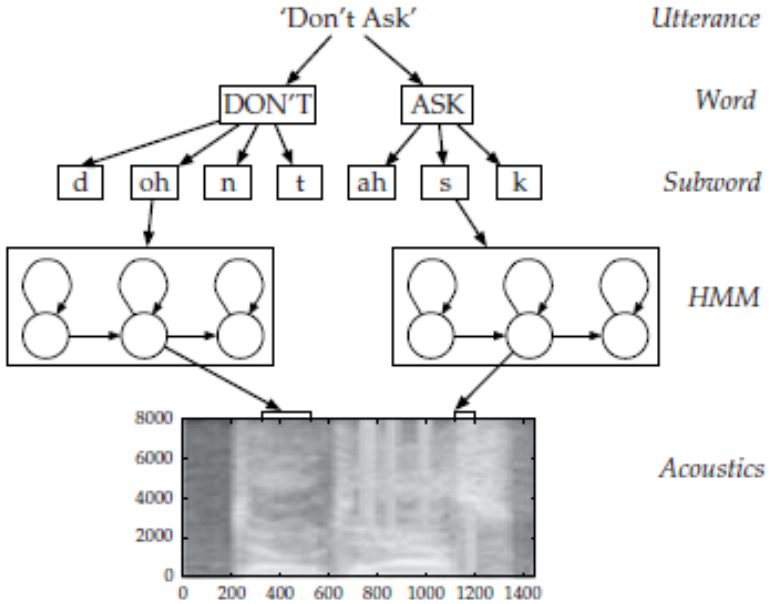
## 10.4 APLICACIÓN DE LOS HMM EN LOS SISTEMAS ASR

### 10.4.1 Modelo Probabilístico

Para la aplicación de un HMM en un sistema de reconocimiento de voz se debe resolver a que va a apuntar cada uno de los estados presentes en el modelo estadístico. Intuitivamente se podría decir, en el campo del reconocimiento de voz, que los estados del modelo representan a cada una de las palabras del idioma. Sin embargo, el idioma español está compuesto por unas 100.000 palabras según la Real Academia de la lengua Española, lo cual es una cantidad de estados que no hace factible al sistema. Por tal razón, para detección de palabras, estas se dividen en formas de representación de sonidos básicos conocidos como fonemas. Para el español, existen alrededor de 22-26. Una palabra se divide en fonemas utilizando un diccionario de pronunciación. Estos fonemas si pueden ser asociados a estados de un modelo oculto de Markov (ver Figura *78*). Cada fonema se asume como una representación de un HMM de tres estados.

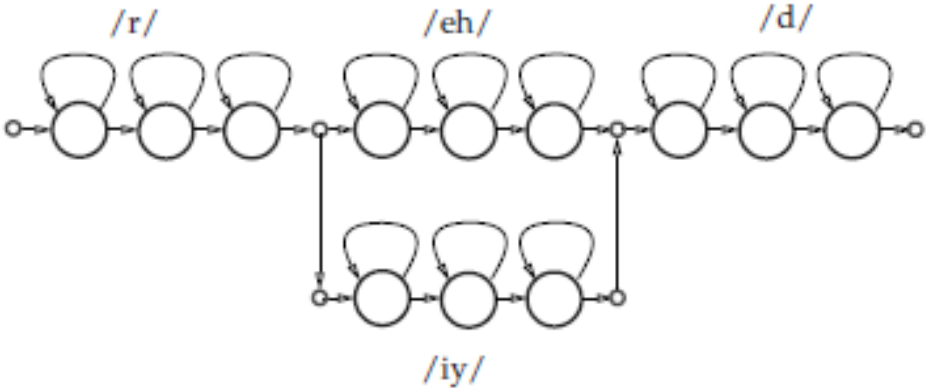
El siguiente ejemplo (ver Figura *79*), muestra el modelado en HMM para la palabra en ingles “read”. Cada uno de los fonemas es representado por un modelo de tres estados. Los modelos de fonemas están concatenados para formar la palabra. Una ventaja de utilizar estos modelos de inteligencia artificial es el uso de vías paralelas, para la inclusión de varias palabras con pronunciación similar.

Figura 78. Modelado jerárquico del habla.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/acoustic-model > [citado en 17 de septiembre de 2016]

Figura 79. Concatenación de fonemas: la ramificación para diferentes pronunciaciones.

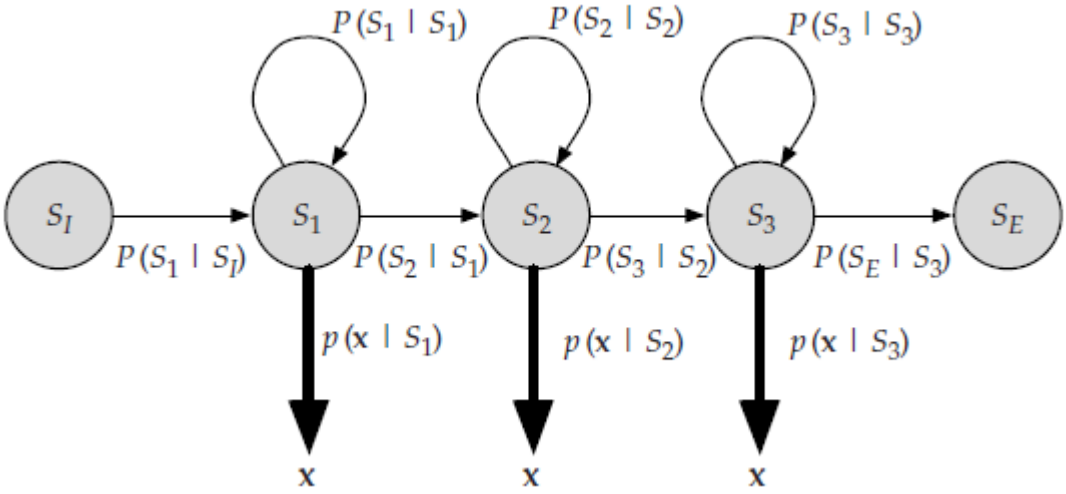



Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/acoustic-model> [citado en 17 de septiembre de 2016]

El HMM puede ser representado como un autómata de estados finitos. La secuencia de estados 2-3 corresponden a un fonema distinto. Los estados I y E representan el inicio y el fin. La transición de estados esta descrita por la probabilidad de transición $Pi|i-1$. La emisión de cada estado (fonema) es un vector de características acústicas $x$ que es generado por determinado fonema. La probabilidad de emisión para cierto estado esta descrita por su distribución de salida $Px|s_i$ de la Figura *80*.

Con el fin de parametrizar completamente un modelo acústico basado en HMM, todas las distribuciones de transición y de salida tienen que ser estimadas a partir de unos datos de entrenamiento.

Figura 80. Transición y emisión de probabilidades.

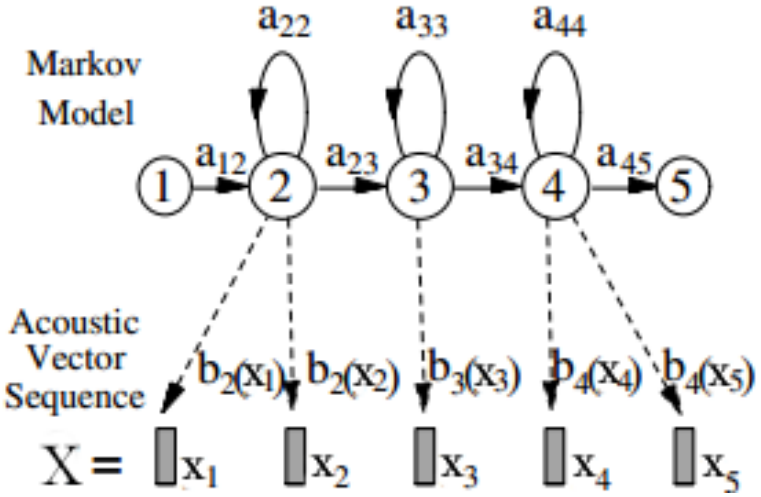



Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/acoustic-model> [citado en 17 de septiembre de 2016]

### 10.4.2 Estructura del modelo estadístico

La estructura subyacente del HMM puede ser visualizado en la *Figura 81*.

Figura 81. Estructura modelo oculto de Markov.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/acoustic-model> [citado en 17 de septiembre de 2016]

Las probabilidades de transición entre estados ocultos $q_i$ y $q_j$ están dadas por $\{a\}_{i,j} \coloneqq P(q_j|q_i)$. Las densidades de emisión condicionales están dadas por $b_i(xt) \coloneqq p(x_t|q_{i,t})$. Para las emisiones discretas estacionarias $\{x_j\}$ la distribución de salida se puede representar como una matriz $\{b\}_{i,j}$ para salida continua (como las emisiones Gaussianas).

### 10.4.3 Entrenamiento de HMM

El proceso de entrenamiento de un modelo HMM, utiliza los algoritmos necesarios para realizar el cálculo o estimación de los parámetros que definen al modelo.

Se trata básicamente de un proceso iterativo que maximiza en forma local la probabilidad de que una secuencia de observación haya sido generada por un modelo particular $[P(O|\lambda)]$ y que garantiza en cierta forma la convergencia del proceso a partir de un modelo inicial aleatorio.

Uno de los métodos más conocidos para realizar esta tarea es la técnica de maximización de la estimación, y como una especialización de la misma, el algoritmo Baum-Welch. Este último define los parámetros de un HMM como se muestra a continuación:

- Probabilidades de transición de estados (Matriz A)

$$a_{ij} = \frac{\text{número esperado de transiciones de i a j}}{\text{número esperado de transiciones desde i}}$$

- Probabilidades de emisión (Matriz B)

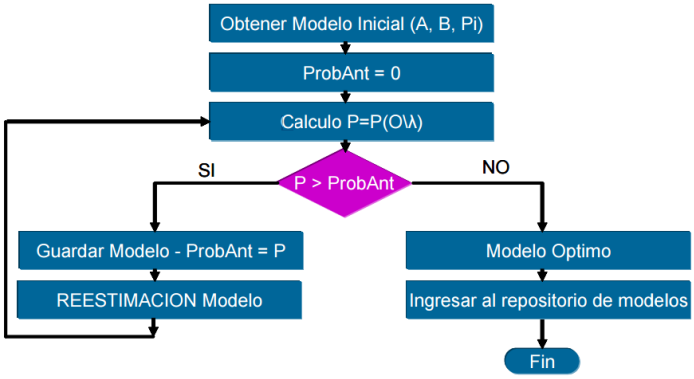
$$b_{j(k)} = \frac{\text{número esperado de veces en el estado } j \text{ con el simbolo } v_k \text{ observado}}{\text{número de veces en el estado } j}$$

- Vector de probabilidad inicial

$$\prod_i = \text{probabilidad de iniciar en estado } i$$

De este modo, dada una secuencia muestral, se busca mediante el método de maximización de la estimación, obtener el modelo HMM que tenga más probabilidad de generar la secuencia indicada utilizando las fórmulas expuestas y el algoritmo general que se indica en la Figura *82*.

Figura 82. Algoritmo general para entrenamiento de un HMM.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/acoustic-model> [citado en 17 de septiembre de 2016]

- **Obtención del modelo inicial**: se obtiene en forma totalmente aleatoria, sujeto como es de suponer a las restricciones de probabilidades comunes. Existe la posibilidad de implementar mejoras que ayuden a obtener parámetros iniciales más exactos. Cabe aclarar que cuanto más exacto o cercano al máximo global se encuentre el modelo inicial, más exacto será el modelo final obtenido.

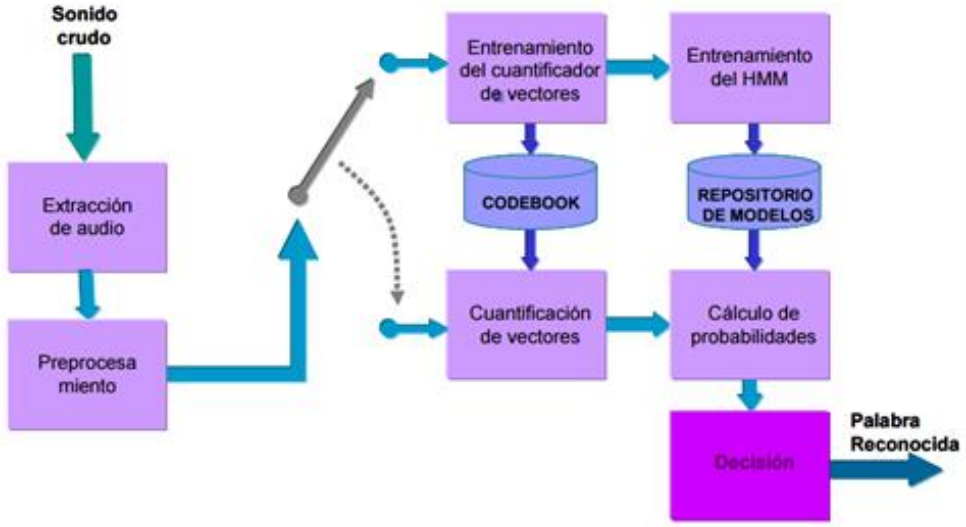
- **Cálculo de Probabilidad**: la probabilidad de que la observación haya sido generada por el modelo obtenido es calculada con la ayuda de algoritmos intermedios auxiliares que permiten reducir la complejidad computacional del proceso (algoritmos forward y backward). Esta probabilidad es calculada por cada uno de los modelos obtenidos hasta verificar que la misma es máxima.

- **Condicional P > ProbAnt**: define el ciclo iterativo del algoritmo, esto es, se recalcularán los parámetros del modelo hasta que la probabilidad obtenida para uno de ellos sea menor que la del modelo anterior. Esto indica que se ha alcanzado un máximo local para el modelo inmediatamente anterior al actual.

- **Reestimación del modelo**: se trata de recalcular los parámetros del modelo utilizando las fórmulas anteriores, basándose para ello en el modelo obtenido en la iteración anterior.

- **Modelo óptimo**: una vez alcanzada la máxima probabilidad, se está en presencia del modelo óptimo el cual debe ser guardado para su utilización posterior en lo que se da a llamar el repositorio de modelos.

Figura 83. Visión global del proceso de entrenamiento/reconocimiento utilizando HMM.




Fuente: SONG, Yang. Composition of language: an introduction to linguistics. [en línea]. <http://recognize-speech.com/acoustic-model> [citado en 17 de septiembre de 2016]

## 10.5 MODELO ACÚSTICO

En el proyecto, la etapa de identificación de patrones a partir de los vectores de observación que contienen la informacion acústica y lingüística del audio original, ocurre mediante la implementación de la librería de reconocimiento robusto de audio SPHINX, desarrollada por la Universidad de Carnage Mellon (CMU).

Sin embargo, el modelo de acústico que utiliza la librería de identificación de patrones SPHINX, lo provee una organización llamada VoxForge, la cual tiene soporte para más de 16 diferentes idiomas, entre ellos, el español. La problemática con el uso de los modelos acústicos es que no son de tipo Open Source. VoxForge es un proyecto que tiene como objetivo recoger transcripciones de textos mediante voz, para ser usados en algun sistema ASR.

Este esta compuesto por dos tipos de ficheros: el Corpus (voces) y la gramática o modelo de lenguaje.

- Corpus: se crea mediante la extracción de datos estadísticos de ficheros con voces. Esta información estadística es una representación del sonido que forma cada palabra. Entre mas información de palabras se tenga, mas exacto serpa el modelo.

- Modelo de lenguaje: la gramática es un fichero relativamente pequeño que contiene conjuntos de combinaciones de palabras. El modelo de lenguaje es un fichero más grande que contiene la probabilidad de que aparezcan ciertas palabras en un determinado orden. Dependiendo de la aplicación se usará o bien una gramática reducida, o un modelo de lenguaje.

Para el correcto funcionamiento del reconocimiento de voz, se limitó el modelo de lenguaje disponible, con algunas palabras asociadas a las tareas de movilidad en la silla de ruedas. A continuación se listan las palabras incluidas en el modelo de lenguaje, y su respectiva distribución fonética del modelo acústico asociado en la Tabla *6*.

Tabla 6. Modelos de lenguaje asociados a determinados fonemas del modelo acústico.

| Modelo Lenguaje | Modelo Acústico |
|---|---|
| a | a |
| abajo | a b a j o |
| adelante | a d e l a n t e |
| arriba | a rr i b a |
| atrás | a t r a s |
| avance | a b a n s e |

| | |
|---|---|
| avanzar | a b a n z a r |
| como | c o m o |
| corrija | c o rr i j a |
| corregir | c o rr e j i r |
| derecha | d e r e ch a |
| detenerse | d e t e n e r s e |
| detener | d e t e n e r |
| deténgase | d e t e n g a s e |
| doblar | d o b l a r |
| doble | d o b l e |
| dos | d o s |
| escucha | e s k u ch a |
| esta | e s t a |
| favor | f a b o r |
| girar | j i r a r |
| gire | j i r e |
| grados | g r a d o s |
| hacia | a z i a |
| hola | o l a |
| ir | i r |
| izquierda | i z k i e r d a |
| jota | j o t a |
| la | l a |
| me | m e |
| metro | m e t r o |
| pare | p a r e |
| por | p o r |
| quieto | k i e t o |
| quiero | k i e r o |
| rectificar | rr e k t i f i k a r |
| rectifique | rr e k t i f i k e |
| retroceda | rr e t r o s e d a |
| retroceder | rr e t r o s e d e r |
| revesa | rr e b e r s a |
| tres | t r e s |
| un | u n |
| voltear | b o l t e a r |
| voltee | b o l t e e |

Fuente: elaboración propia.

## 10.6 ANALISIS DE LAS PALABRAS IDENTIFICADAS POR EL HMM

La ejecución de la librería SPHINX se lleva a cabo en el procesador ARM. Se diseño un algoritmo que trabaje en sintonía con la identificación de las palabras para generar las instrucciones de movimiento que serán enviadas al módulo de control de los motores. Este algoritmo ha sido diseñado para ejecutarse en Linux, y responde a una serie de etapas de funcionamiento.

La primera etapa corresponde a una función de despertar (wake up). Debido a la demanda de procesamiento que implica la detección de patrones de audio en tiempo real, el procesamiento de una muestra audible solo se realizará una vez se realice una rutina de wake up. Esta rutina esta revisando constantemente los datos provenientes a través del registro FIFO, y espera a que provenga un dato que supere cierto umbral de intensidad para desencadenar la etapa de creación de instrucciones.

En la Figura *84* se aprecia la rutina diseñada para el levantamiento del sistema.

Figura 84. Rutina Wake Up.

```
bool in_mic(short sample)
{
    bool flag = false;
    static unsigned long    sum = 0;
    static unsigned short   cnt = 0;
    unsigned short          power = 0;
    sum += (sample >= 0)?sample:-sample;
    cnt++;
    if (cnt == 32)
    {
        power = sum >> 13;
        int j;
        unsigned long Mask = 0;
        for(j=0;j<power;j++)
        {
            Mask <<= 1;
            Mask |= 0x01;
        }
        if(Mask>0x40)
        {
            flag = true;
        }
        sum = 0;
        cnt = 0;
    }
    return flag;
}
```

Fuente: elaboración propia en Eclipse ARM DS-5 Development Studio.

Una vez se supera el umbral de activación, establecido para un valor de intensidad 0x40, se procede a grabar un archivo de audio WAV en la memoria SDRAM del board DE1-SoC. Este audio WAV se graba con las muestras provenientes de los siguientes 2.4 segundos próximos al instante de activación. En la Figura *85* se visualizan los parámetros de configuración maestros para la grabación de audio.

Figura 85. Parametros globales de configuración para la grabación de audio.

```
#define FS                      (16000)
#define RECORD_TIME             2.4
#define BUFFER_TIME             10
#define RECORD_LENGTH           (FS * RECORD_TIME)
#define WAIT_MIC_SPAN           (FS * BUFFER_TIME)
```

Fuente: elaboración propia en Eclipse ARM DS-5 Development Studio.

Con la muestra de audio almacenada en un archivo WAV, se ejecutan las etapas de extracción de características de la muestra, así como la identificación de las palabras utilizando HMM. Una vez el proceso ha culminado con la identificación, el algoritmo busca entre las palabras detectadas la palabra clave de activación de comandos.

Para el proyecto se ha elegido la palabra "**jota**" como contraseña para permitir que se ejecuten los comandos de movimiento en la silla de ruedas. En la Figura *86* se aprecia la lógica para la detección de la palabra clave en la tira de datos proveniente del reconocimiento de voz.

Figura 86. Rutina para la detección de la palabra clave.

```
case IS_KEY_WORD:

    /* IS_JOTA? */

    if (strstr(text, "jota") != NULL)
    {
        state = CONTROL;
    }
    else
    {
        printf("Waiting for KeyWord\n");
        usleep(4);
        state = IDLE;
    }
break;
```

Fuente: elaboración propia en Eclipse ARM DS-5 Development Studio.

Si la palabra clave se encuentra dentro de las palabras identificadas por el motor de reconocimiento de voz, el algoritmo busca además alguna de las palabras asociadas al movimiento de la silla de ruedas, como adelante, atrás, derecha, izquierda, entre otras. Solo se ejecutará una tarea mientras no se haya solicitado más de una acción de movimiento en la misma oración. Es decir, si el motor de reconocimiento detecta peticiones para ir en dos o más direcciones opuestas, el algoritmo desechara la solicitud y pedirá al usuario una nueva orden de movimiento.

Finalmente, si la orden se realiza de manera satisfactoria, el algoritmo envía al módulo de control de los motores, las instrucciones para indicar un movimiento en determinada direccion.

# 11 EJECUCIÓN DE ÓRDENES

En la etapa de procesamiento no solo se realiza el reconocimiento de voz sino que además el procesador se encarga de la creación de subrutinas de operación para el manejo de la silla de ruedas, mediante una lógica que finalmente envía órdenes de desplazamiento para que en la última etapa esta información permita generar el movimiento en cuestión.

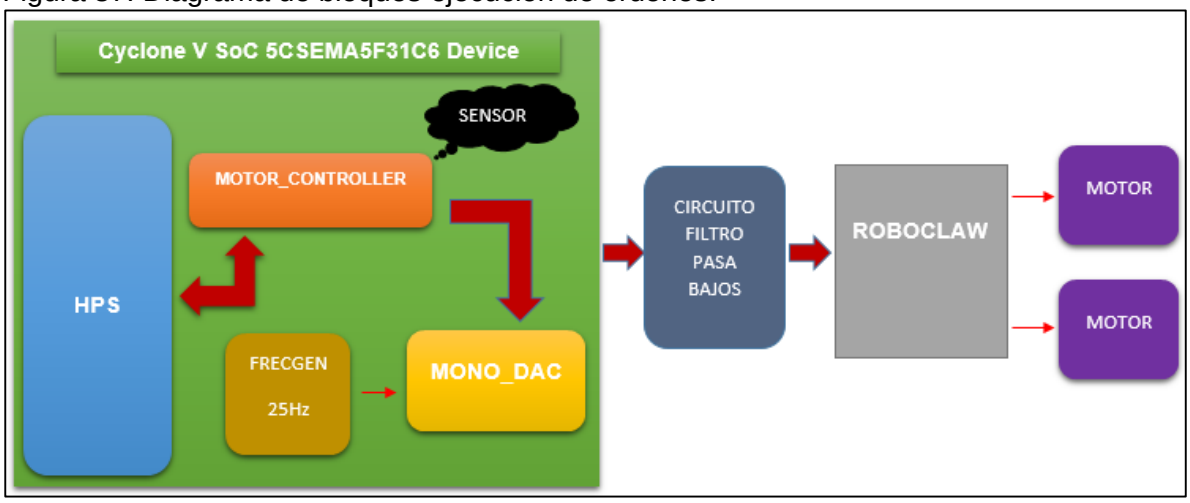
La etapa de ejecución es la encargada de recibir las órdenes provenientes del HPS para transformarlas en movimiento que se percibe sobre la silla de ruedas. Se utilizan principalmente módulos hardware para el procesamiento de las instrucciones recibidas, la construcción de rampas para aceleración y frenado, la administración de los tiempos de ejecución y la supervisión del desplazamiento.

Para la implementación de esta etapa de ejecución, se diseñaron tres módulos hardware, listados a continuación:

- FrecGen (Divisor de frecuencia)
- Motor_Controller
- Mono_DAC

La silla de ruedas debe desplazarse a una velocidad baja, pues su prioridad es la seguridad del usuario. El generador de la señal de salida es el módulo motor_controller, sin embargo la comunicación con el RoboClaw la realiza el módulo mono_dac junto con un filtro pasa bajos. En la Figura *87* se observa el diagrama general de la etapa de ejecución de tareas.

Figura 87. Diagrama de bloques ejecución de órdenes.




Fuente: Elaboración propia.

## 11.1 MÓDULO MOTOR_CONTROLLER

Es un driver, desarrollado en hardware usando lenguaje Verilog, encargado de crear y manipular las señales lineales necesarias para el manejo óptimo de los motores que permiten el desplazamiento de la silla de ruedas. Motor_controller se encarga de decirle a Roboclaw en qué sentido y a qué velocidad debe acelerar los motores, durante cuánto tiempo permanecer a velocidad nominal y como realizar los tipos de frenado (suavizado o de emergencia).

El núcleo central del driver es una FSM. El objetivo es lograr un desplazamiento lento que dé prioridad a la seguridad del usuario por encima de la velocidad de movimiento. Lograr este procedimiento en hardware requiere una de dos adaptaciones: el uso de amplios registros (32 bits) para la creación de subrutinas como aceleración y frenado, o emplear un reloj de baja frecuencia como referencia para la operación del módulo. Se decidió desarrollar la segunda opción ya que es más eficiente con el uso de elementos lógicos disponibles en la FPGA. Por esta razón, el módulo motor_controller trabaja a una frecuencia de 25 Hz, que se logra a través de un módulo divisor de frecuencia.

Motor_controller recibe la información de los sensores de ultrasonido conectados a la silla de ruedas para determinar si es conveniente detener el movimiento en algún momento del trayecto. 25 Hz no es una velocidad recomendable para leer sensores y decidir detener los motores, ya que la velocidad de respuesta será excesiva. Por ello se emplea una red de asignaciones de alta prioridad, utilizando hardware dedicado, que interrumpe el desplazamiento en el instante que se requiere, sin necesidad de esperar los flancos del reloj de referencia.

La interfaz del módulo consta de 16 señales de entrada y 5 de salida; dedicadas al reloj de referencia, reset para la FSM, ocho buses de señales de 16 bits provenientes de los sensores de ultrasonido, cuatro buses de 8 bits para la configuración de los rangos de operación del controlador, un puerto setup de 4 bits a través del cual llegan las ordenes de movimiento, así como una señal ready proveniente del HPS, dedicada a la sincronización en la comunicación HPS-FPGA. Los parámetros de salida están compuestos por dos puertos de 1 bit llamados warning y finish que son enviados al procesador para avisar de la interrupción en el movimiento por obstáculos, y 2 puertos de 8 bits llamados L_motor y R_motor a través de las cuales se envía la información de manejo. En la Figura *88* se puede observar las variables de entradas y salidas del módulo.

Originalmente motor_controller utilizaba como salida modulación PWM para controlar los motores. Sin embargo, el roboclaw actual trabaja correctamente únicamente con señales análogas o comunicación serial. Por ello actualmente Motor_controller soporta comunicación con Roboclaw mediante el envío de información a través de variación en tensión o en ancho de pulso.

Figura 88. Puertos del módulo Motor_Controller.

```
input           CLK;                    // CLOCK
input           RESET;                  // RESET
output [7:0]    L_motor;                // MOTOR LEFT
output [7:0]    R_motor;                // MOTOR RIGHT
input  [7:0]    index_min;              // PARAMETER VEL MAX GO
input  [7:0]    index_neutral;          // PARAMETER VEL STOP
input  [7:0]    index_max;              // PARAMETER VEL MAX REVERSE
input  [3:0]    setup;                  // COMMANDS
output          finish;                 // INDICATOR FINISH COMMANDS
output          warning;                // INDICATOR SENSOR
input           ready;                  // INDICATOR HPS
input  [7:0]    temp;                   // PARAMETER COUNTING MAX
input  [15:0]   DATA_SRF_E0;            // SENSOR FRONT     1
input  [15:0]   DATA_SRF_E2;            // SENSOR FRONT     2
input  [15:0]   DATA_SRF_E4;            // SENSOR RIGHT
input  [15:0]   DATA_SRF_E6;            // SENSOR BOT       1
input  [15:0]   DATA_SRF_E8;            // SENSOR BOT       2
input  [15:0]   DATA_SRF_EA;            // SENSOR LEFT
input  [15:0]   DATA_SRF_EC;            // SENSOR UNDER FRONT
input  [15:0]   DATA_SRF_EE;            // SENSOR UNDER BOT
```

Fuente: Elaboración propia en Quartus II, versión 15.0.

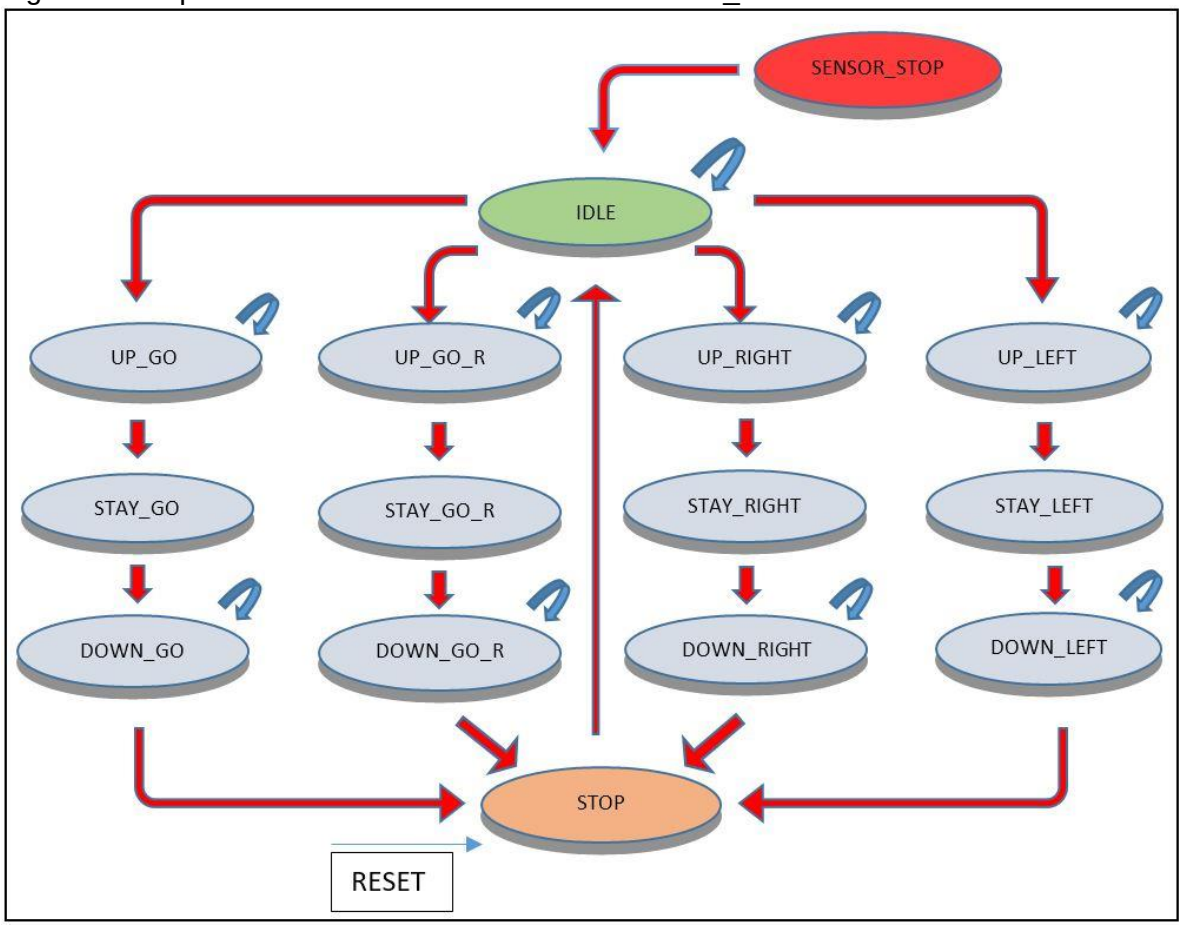
Una de las funciones de motor_controller consiste en evaluar las distintas distancias entregadas por cada sensor para ser comparadas entre los rangos de operación y minimizar el riesgo de colisión contra un obstáculo durante la trayectoria. Esto se realiza a partir de una red de asignaciones en hardware dedicado para aumentar la velocidad de respuesta ante una eventualidad de este tipo.

La FSM se encuentra a la espera de recibir una instrucción proveniente del HPS para ejecutarla. Una vez se recibe una orden, se desencadena un recorrido a lo largo de las diferentes rutas posibles para la ejecución de las tareas, tal y como se observa en la Figura *89*. Estas rutas incluyen momentos dedicados a la aceleración suavizada, la administración del tiempo que perduraran los motores en velocidad nominal y el frenado mediante desaceleración. La aceleración y el frenado se desarrollan mediante rampas de tensión diseñadas a partir de un contador de baja frecuencia. El tiempo de duración de la aceleración y el frenado los controla directamente el HPS modificando el valor hasta el cual se debe contar para ir aumentando o reduciendo la escala de tensión.

A continuación se describen los estados principales de los cuales se componen las posibles rutas de operación:

- **IDLE:** estado de espera para recibir la orden proveniente del procesador, mediante el bus de entrada llamado setup.
- **UP & DOWN:** son los estados encargados de generar las rampas de forma creciente o decreciente. Las rampas están delimitadas por unos valores máximos y mínimos que define el HPS.
- **STAY:** estado encargado de mantener los motores a velocidad nominal.
- **STOP:** estado de culminación. Detiene el movimiento de los motores.

Figura 89. Representación de la FSM del módulo Motor_Controller.




Fuente: Elaboración propia.

Para la composición de las rampas ascendente y descendente se tienen en cuenta unos límites máximo y mínimo que define el HPS. Además, existe un valor neutral que permite detener el movimiento de los motores. Actualmente se controla el movimiento a partir se señales análogas de tensión, que varían entre 0-2 Voltios. En la Tabla *7* se aprecian los valores de la señal índex que permiten girar los motores a máxima velocidad en cada sentido, así como el valor que detiene los motores.

Tabla 7. Calibraciones del parámetro índex en el módulo Motor_Controller.

| Nombre del índex | Valor del índex | Tensión | Acción del giro |
|---|---|---|---|
| Index_minimo | 128 | 0 V | Adelante: velocidad máxima |
| Index_neutral | 200 | 1 V | Parada |
| Index_maximo | 255 | 2 V | Atrás: velocidad máxima |

Fuente: Elaboración propia.

Las órdenes enviadas desde el HPS al puerto setup de 4 bits activan una de las posibles rutas de operación de la máquina de estados. Las instrucciones se aprecian en la Tabla *8*.

Tabla 8. Puerto setup que traduce la acción en el módulo Motor_Controller.

| Orden de setup | Caso a iniciar | Acción |
|---|---|---|
| 4'd0 | STOP | Parar |
| 4'd1 | UP_GO | Avanzar |
| 4'd2 | UP_GO_R | Retroceder |
| 4'd3 | UP_RIGHT | Girar: derecha |
| 4'd4 | UP_LEFT | Girar: izquierda |

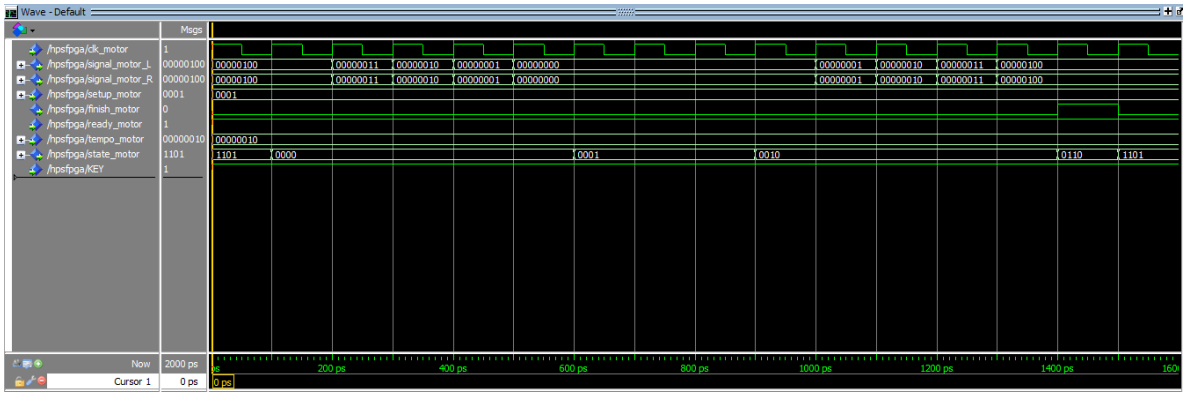
Fuente: Elaboración propia.

La variable temp se utiliza para determinar el tiempo de ejecución de las tareas que ejecuta motor_controller. Esta variable la controla el HPS y le permite determinar la duración del desplazamiento o de los giros. Se aclara que en la silla de ruedas es manejada, mas no controlada. Esto quiere decir que el HPS envía un tiempo determinado para realizar los giros buscando que estos sean aproximadamente de 90°, más no se garantiza que este sea el valor recorrido. Lo mismo ocurre con las distancias para avanzar o retroceder.

En el controlador de los motores existen también dos estados dedicados a la interrupción del movimiento, uno por petición del usuario y otro por obstáculos detectados. El controlador tiene una lógica inteligente que detecta obstáculos en determinados sensores solo si estos perjudican el desplazamiento actual de la silla. En caso tal, el controlador indicará al HPS que no es posible avanzar en la dirección solicitada mediante una señal de aviso (warning). Esto permite que el usuario pueda utilizar un comando de desplazamiento en otra dirección para poder zanjar el obstáculo que impide el movimiento.

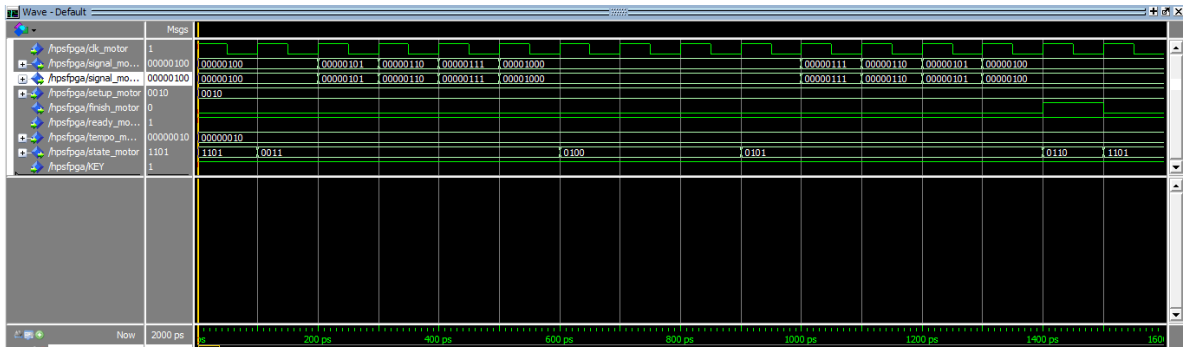
En las Figura *90*, Figura *91*, Figura *92* y Figura *93* se muestran las simulaciones del módulo motor_controller para las 4 rutas de operación disponibles.

Figura 90. Representación del funcionamiento del módulo Motor_Controller con la orden "avanzar".



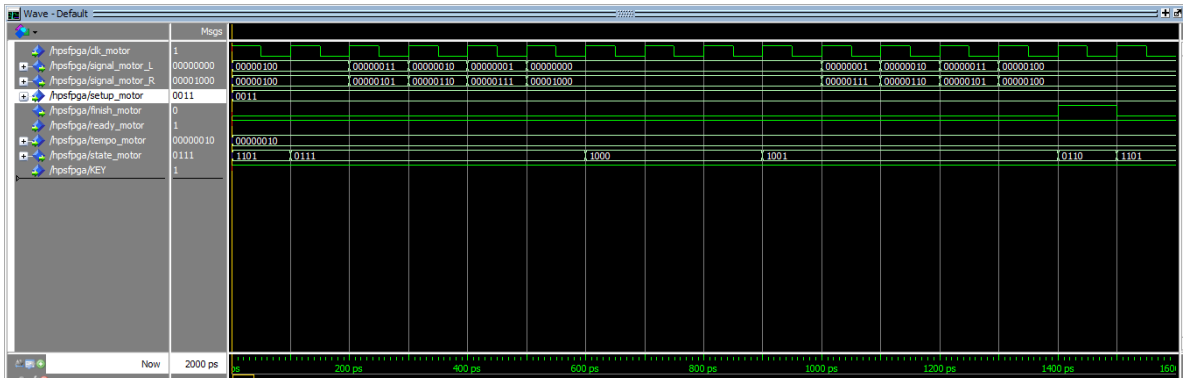

Fuente: Elaboración propia en ModelSim 10.3d.

Figura 91. Representación del funcionamiento del módulo Motor_Controller con la orden “retroceder”.



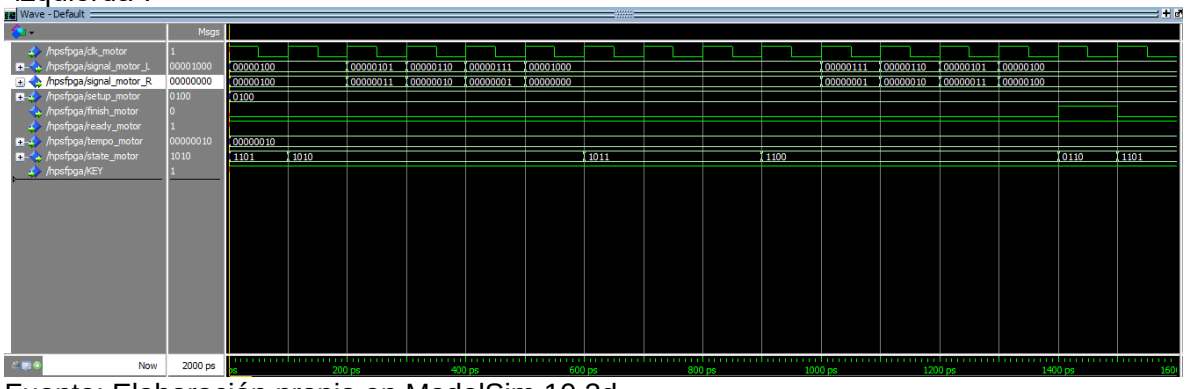

Fuente: Elaboración propia en ModelSim 10.3d.

Figura 92. Representación del funcionamiento del módulo Motor_Controller con la orden “derecha”.




Fuente: Elaboración propia en ModelSim 10.3d.

Figura 93. Representación del funcionamiento del módulo Motor_Controller con la orden “izquierda”.

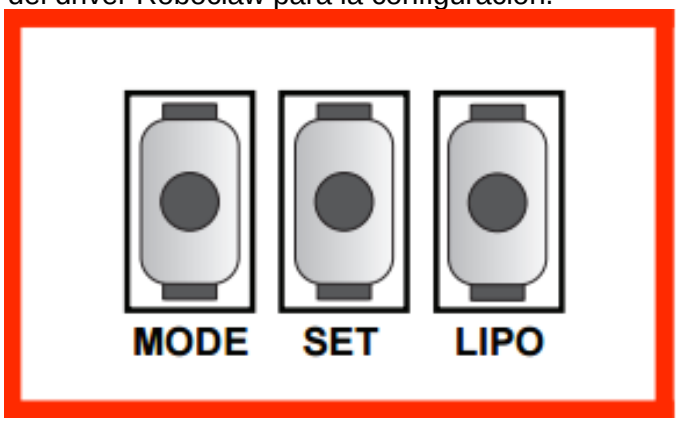



Fuente: Elaboración propia en ModelSim 10.3d.

## 11.2 CONFIGURACIÓN DEL ROBOCLAW

Roboclaw debe estar configurado para recibir señales análogas para el control de los motores, ya que este fue el modo elegido para el desarrollo y funcionamiento del proyecto. Para configurar el Roboclaw 2x30A V4, Ion Motion Control ha dispuesto en su diseño 3 botones dedicados exclusivamente a tal función, los cuales se pueden observar en la Figura *94*.

Figura 94. Botones del driver Roboclaw para la configuración.




Fuente: ©Ion Motion Control. RoboClaw Series Brushed DC Motor Controllers: User Manual. Revision 5.1: Ion Motion Control, 2014, 2015. p 15.

El botón MODE permite abrir el menú de configuración principal en donde se selecciona entre uno de los 4 modos de operación principales: RC, análogo, serial estándar o por paquetes seriales.

El botón SET permite abrir el menú de configuración secundario, donde se activan algunas características adicionales como estados lógicos en TTL o RC, y modos de funcionamiento como exponencial o MCU.

El botón LIPO permite abrir el menú de configuración de los parámetros de protección del Roboclaw para niveles de alimentación. Si Roboclaw detecta que el nivel de tensión en el circuito de alimentación es inferior al seleccionado, se apagará para evitar fallos en los dispositivos conectados.

Para establecer el modo deseado se deben seguir los siguientes pasos:

- Pulsar y soltar el botón asociado al modo de configuración que se desea modificar. El LED STAT2 comenzará a parpadear indicando la posición actual dentro del menú seleccionado.
- Dentro del menú elegido, pulsar SET para incrementar al siguiente parámetro, y pulsar MODE para regresar al parámetro anterior.
- Pulsar y soltar el botón LIPO para guardar en la memoria el modo de configuración seleccionado.

Para el proyecto se utilizó el modo de configuración análogo del menú principal, tal y como se muestra en la Tabla *9*.

Tabla 9. Modos de configuración para el Roboclaw.

| Mode | Description |
|---|---|
| 1 | RC mode |
| 2 | RC mode with mixing |
| 3 | Analog mode |
| 4 | Analog mode with mixing |
| 5 | Standard Serial |
| 6 | Standard Serial with slave pin |
| 7 | Packet Serial Mode - Address 0x80 |
| 8 | Packet Serial Mode - Address 0x81 |
| 9 | Packet Serial Mode - Address 0x82 |
| 10 | Packet Serial Mode - Address 0x83 |
| 11 | Packet Serial Mode - Address 0x84 |
| 12 | Packet Serial Mode - Address 0x85 |
| 13 | Packet Serial Mode - Address 0x86 |
| 14 | Packet Serial Mode - Address 0x87 |

Fuente: ©Ion Motion Control. RoboClaw Series Brushed DC Motor Controllers: User Manual. Revisión 5.1: Ion Motion Control, 2014, 2015. p 15.

En el menú de configuración adicional, se seleccionó la opción TTL Flip Switch, la cual utiliza nivel de estado lógico TTL, y mantiene la acción de movimiento en los motores únicamente mientras exista un pulso en el canal de entrada. Este menú se aprecia en la Tabla *10*.

Tabla 10. Opciones del modo análogo y RC del Roboclaw.

| Option | Description |
|---|---|
| 1 | TTL Flip Switch |
| 2 | TTL Flip and Exponential Enabled |
| 3 | TTL Flip and MCU Enabled |
| 4 | TTL Flip and Exp and MCU Enabled |
| 5 | RC Flip Switch |
| 6 | RC Flip and Exponential Enabled |
| 7 | RC Flip and MCU Enabled |
| 8 | RC Flip and Exponential and MCU Enabled |

Fuente: ©Ion Motion Control. RoboClaw Series Brushed DC Motor Controllers: User Manual. Revisión 5.1: Ion Motion Control, 2014, 2015. p 16.

Para la configuración de protección de funcionamiento, se seleccionó el límite inferior de carga en 9 V, lo cual indica que si la batería que alimenta el circuito de potencia suministra una tensión inferior a este valor, el Roboclaw se deshabilita y los motores se desenergizan. Esta configuración corresponde a la posición 3 del menú de protecciones como se aprecia en la Tabla *11*.

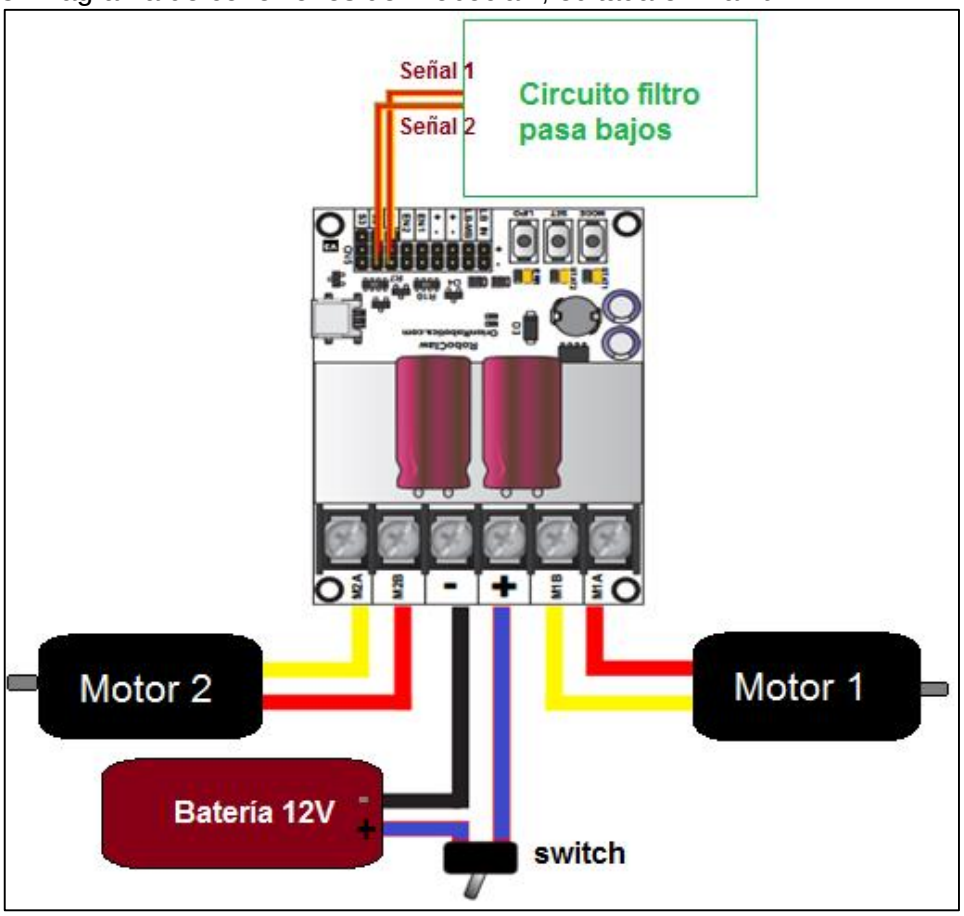
Tabla 11. Opciones para ajuste de corte del nivel de tensión, imagen tomada de User Manual Roboclaw.

| Option | Description |
|---|---|
| 1 | Disabled (6V Cuttoff) |
| 2 | Auto Detect |
| 3 | 3 Cell(9V Cutoff) |
| 4 | 4 Cell(12V Cutoff) |
| 5 | 5 Cell(15V Cutoff) |
| 6 | 6 Cell(18V Cutoff) |
| 7 | 7 Cell(21V Cutoff) |
| 8 | 8 Cell(24V Cutoff) |

Fuente: ©Ion Motion Control. RoboClaw Series Brushed DC Motor Controllers: User Manual. Revisión 5.1: Ion Motion Control, 2014, 2015. p 17.

Una vez terminado estos pasos de configuración el sistema es cableado al circuito filtro pasa bajos para recibir las señales analógicas en cada canal que ofrece el driver Roboclaw como se muestra en la Figura *95*.

Figura 95. Diagrama de conexiones del Roboclaw, editada en Paint.

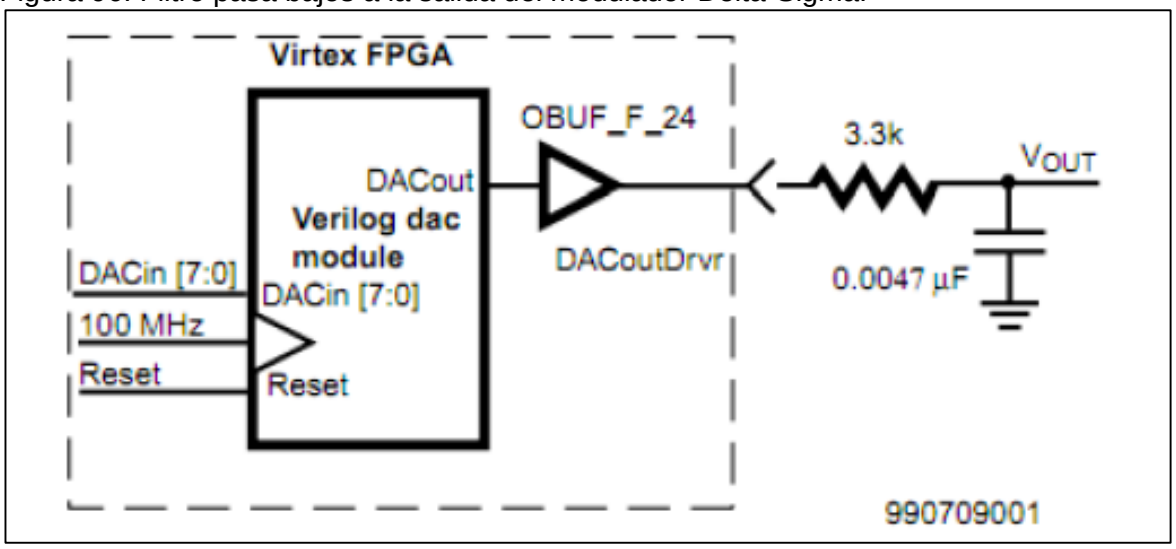



Fuente: Elaboración propia.

## 11.3 MÓDULO MONO_DAC

La dirección y velocidad de movimiento de los motores DC se controla mediante señales análogas de tensión. Para este fin, se requirió el uso de un convertidor digital análogo (DAC) controlado mediante un módulo diseñado en hardware llamado MONO_DAC, el cual convierte una única señal con valores binarios en un voltaje directamente proporcional según una escala usada como referencia. Este tipo de módulos es ampliamente utilizado en aplicaciones de generadores de forma de onda o fuentes de tensión. En la Figura *96* se aprecia el módulo diseñado en verilog para controlar la conversión, y el circuito externo que es utilizado como un filtro pasa bajo.

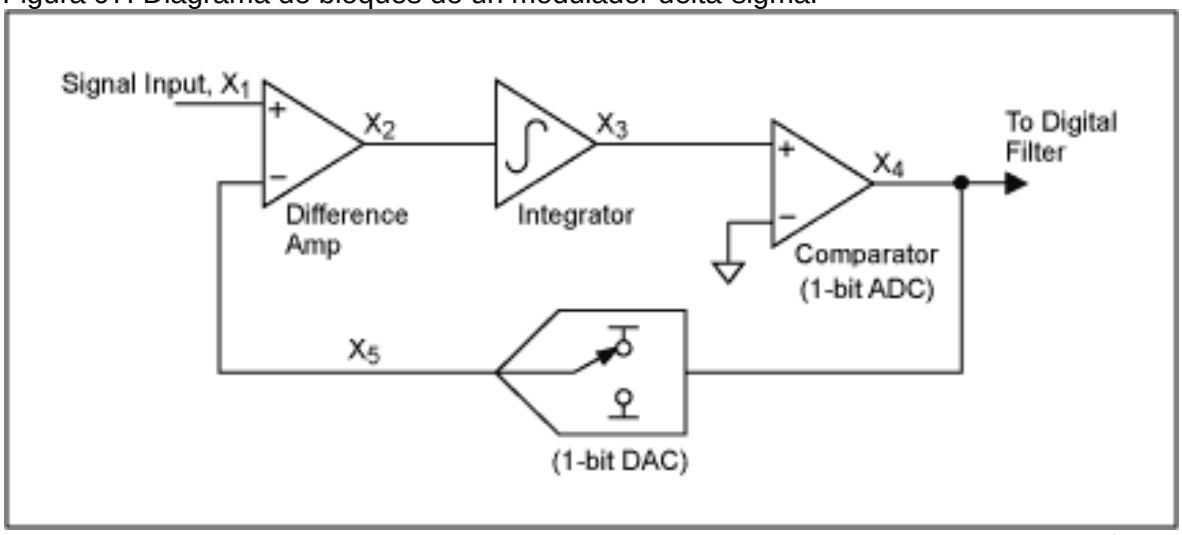
Figura 96. Filtro pasa bajos a la salida del modulador Delta-Sigma.




Fuente: Xilinx. Virtex Synthesizable Delta-Sigma DAC: XAPP154. Versión 1.1: John Logue, Septiembre 23 de 1999. p 1.

El modulo funciona utilizando lógica digital para generar frecuencias altas que semejan una señal analógica convencional. El modulo descrito en hardware tiene 3 etapas importantes, la primera es la selección de volumen la cual permite seleccionar la amplitud de salida que el mono_dac debe generar, la segunda etapa es el multiplicador que escala la señal de entrada de acuerdo a la condición de volumen seleccionada y la última etapa es el integrador retroalimentado que hace que se genere un amplitud en frecuencia de acuerdo a la entrada del sistema con retroalimentación delta-sigma. El circuito descrito en hardware para la modulación delta-sigma se aprecia mediante el diagrama de bloques de la Figura *97*.

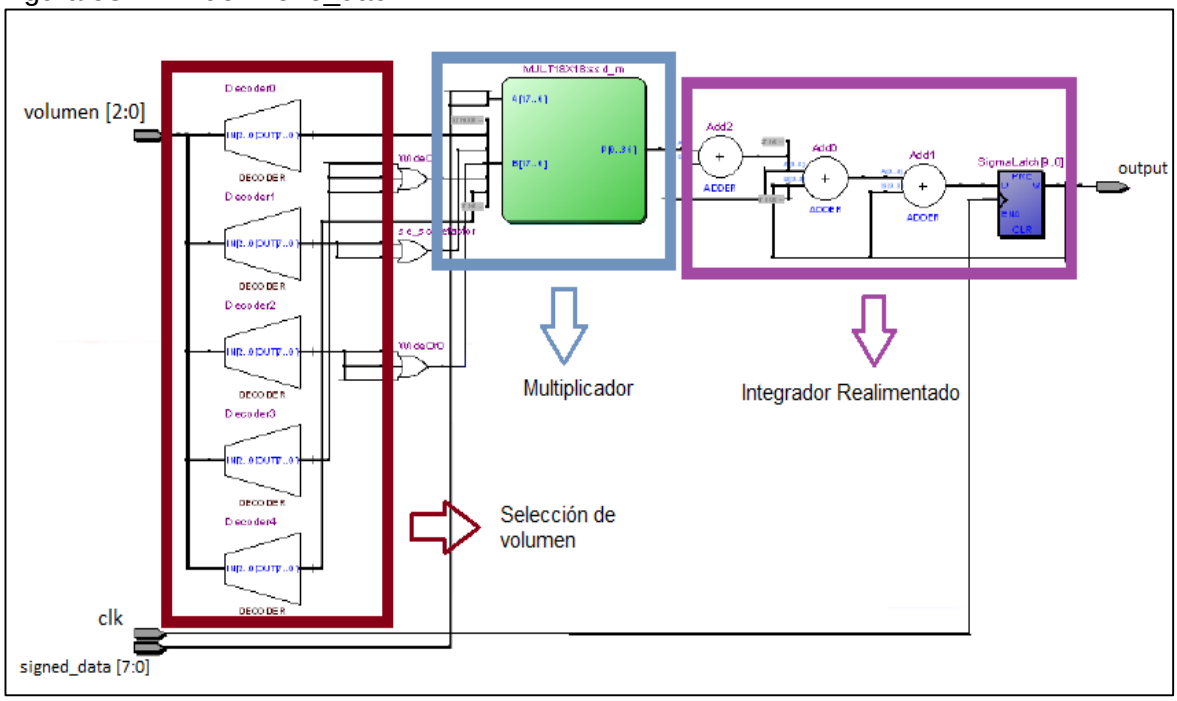
Figura 97. Diagrama de bloques de un modulador delta-sigma.




Fuente: Maxim integrad. Demystifying Delta-Sigma ADCs. [en línea]. <https://www.maximintegrated.com/en/app-notes/index.mvp/id/1870> [citado en 7 de septiembre de 2016]

En la Figura *98* se aprecia la descomposición de las etapas principales del módulo MONO_DAC para la conversión ADC mediante modulación delta-sigma, vista desde el visualizador RTL de Quartus II.

Figura 98. RTL del mono_dac.




Fuente: Elaboración propia en Quartus II.

Este módulo MONO_DAC es el encargado de tomar las señales del driver motor_controller para convertirlas en señales análogas que se envían al Roboclaw para controlar los motores. Son necesarios dos módulos MONO_DAC para las los entradas independientes del RoboClaw 2x30A. En la Tabla *12* se puede observar los valores de la resistencia y del capacitor del circuito pasa bajos requerido para el filtrado de la señal de salida.

Tabla 12. Implementación del DAC para el filtro pasa bajo.

| No. bits | No. slices | Max. Freq | Typical R | Typical C |
|---|---|---|---|---|
| 6 | 4 | 223 MHZ | 3.3 KΩ | 0.0047µF |
| 8 | 5 | 219 MHZ | 3.3 KΩ | 0.0047µF |
| 10 | 6 | 215 MHZ | 6.8KΩ | 0.0047µF |

Fuente: Elaboración propia.

## 11.4 ACTUADORES Y BATERIAS

Los actuadores del sistema son los motores eléctricos 24 VDC encargados de permitir el desplazamiento del chasis (silla de ruedas). Los motores están instalados en los ejes de las llantas traseras como se muestra en la Figura *99*, cada motor cuenta con una caja de reducción con relación 32:1, brindando la capacidad de movilizar a una persona de hasta 90 Kg de peso en terreno plano sin mayor dificultad.

Figura 99. Fotografía de los motores en silla de ruedas.


Fuente: Elaboración propia.

Las baterías tiene una tensión 12V a una corriente de 12 AH (Amperios Hora) diseñadas con la tecnología AGM (Absorbent Glass Mat), de alto rendimiento con placas y electrolito para ganar potencia, utilizadas para la alimentación de la FPGA y el Roboclaw (ver Figura *100*). Se implementaron dos baterías para separar el circuito de control y de potencia, evitando que los picos de corriente provocados por el arranque de los motores puedan afectar a los componentes de control del sistema.

La batería del circuito de control del sistema alimenta la FPGA que esta a su vez suministra tensión a los sensores y a otros dispositivos. La tensión de entrada de board DE1-SOC para su funcionamiento es de 12VDC y consume entre 1 y 2 Amperios, según la cantidad de periféricos encendidos.

La batería del circuito de potencia alimenta al Roboclaw que distribuye internamente la tensión para los motores.

Figura 100. Fotografía de las baterías en silla de ruedas.

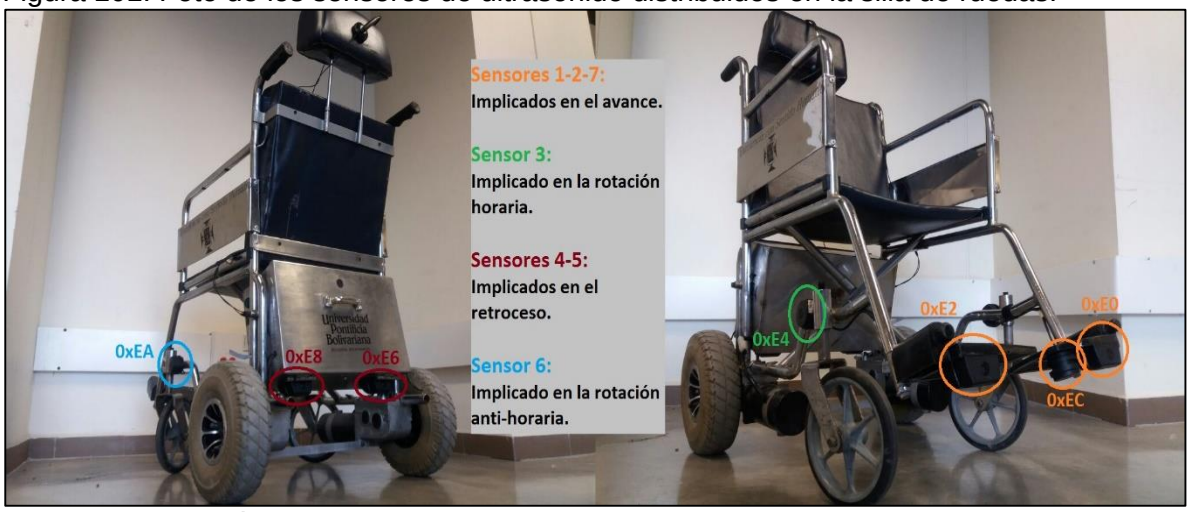

Fuente: Elaboración propia.

# 12 SENSORES DE ULTRASONIDO: SEGURIDAD

En el desarrollo de este proyecto se consideró de vital importancia elaborar un sistema que evite colisiones a lo largo de los desplazamientos para minimizar la posibilidad de accidentes relacionados con el movimiento realizado por la silla de ruedas y garantizar la seguridad del usuario. Este modelo de seguridad se basa en la detección de objetos que se consideren de alto riesgo para el desplazamiento de la silla de ruedas, como obstáculos frontales de gran tamaño, o incluso ausencia de superficie por la cual transitar, como el comienzo de una escalera.

Se utilizan sensores de ultrasonido estratégicamente distribuidos para la medición de distancia entre la silla de ruedas y objetos cercanos alrededor de la misma (ver Figura *101*). Se utilizaron 8 sensores para identificar obstáculos en las cuatro posibles direcciones de desplazamiento de la silla de ruedas. Además, se utilizan 2 sensores de ultrasonido para sensar el vacío por donde transita la silla de ruedas y de esta manera evitar caer en superficies no deseadas.

Figura 101. Foto de los sensores de ultrasonido distribuidos en la silla de ruedas.




Fuente: Elaboración propia.

El sensor utilizado es un SRF02, el cual es un medidor ultrasónico de distancia, es decir un sonar de pequeño tamaño que opera a través de un solo transductor. Entre las características más destacadas de este sensor se encuentra sin duda el hecho de utilizar un único transductor tanto para transmitir la ráfaga ultrasónica como para recibir el eco de la misma y poder así medir la distancia. Esta característica hace que la distancia mínima medida sea mayor que la de otros medidores del mercado que utilizan para ello dos transductores diferentes, siendo esta aproximadamente 15cm (6"). Como otros sensores de distancia, el SRF02 permite arrojar resultados en micro segundos, centímetros o pulgadas.

Este medidor permite comunicaciones serie e I2C, con un total de 16 dispositivos diferentes en ambos modos. Ya que los niveles de tensión son TTL se pueden conectar directamente con la UART de cualquier microcontrolador.

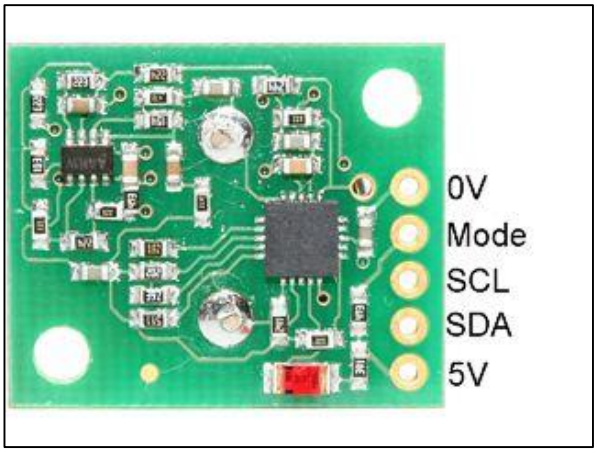
Para la conectividad y comunicación con la FPGA, se eligió el estándar I2C, pues anteriormente se ha elaborado un controlador para este tipo de comunicación en el proyecto que permite envío de información. En este caso se requiere re diseñar este módulo para que soporte tanto envío como recepción de información y así permitir que el maestro pueda escribir y leer en un dispositivo esclavo (sensor).

El maestro en la comunicación con los sensores de ultrasonido será un hardware diseñado dedicado a la exclusiva función de indicarle a los SRF02 el tiempo adecuado para realizar el sondeo y almacenar los resultados para posteriormente enviarlos al módulo de control de los motores. Este hardware recibe el nombre de I2C_controller_SRF02.

## 12.1 SRF02: MODO I2C

Para seleccionar el modo I2C se debe dejar sin conectar el pin modo del SRF02. El pin SDA corresponde a la señal de datos seriales del bus I2C y el pin SCL a la señal de reloj. Ambas señales se deben polarizar a 5 V a través de dos resistencias de polarización positiva (resistencias de Pull Up) que normalmente se encuentran en el circuito maestro del bus I2C que controla los dispositivos I2C esclavos. En la Figura *102* se visualizan los pines de conexión para un sensor SRF02.

Figura 102. Pines de conexión de un sensor SRF02 observados desde la vista inferior.




Fuente: SRF02: Sensor de distancias por ultrasonido. [en línea]. <http://www.superrobotica.com/s320122.htm> [citado en 13 de agosto de 2016]

La dirección única I2C del medidor SRF02 por defecto es 0xE0 pero se puede elegir cualquiera de las otras 16 siguientes para conectar otros sensores al mismo bus:

0xE0, 0xE2, 0xE4, 0xE6, 0xE8, 0xEA, 0xEC, 0xEE, 0xF0, 0xF4, 0xF6, 0xF8, 0xFA, 0xFC, 0xFE.

Todo dispositivo con soporte para conexión I2C contiene registros internos en los cuales se almacena información relevante a la que el maestro puede acceder para realizar lectura de datos o escritura.

### 12.1.1 Registros internos

El sensor SRF02 tiene un juego de seis registros para su operación. Algunos de ellos son asequibles para el maestro y otros tienen restricciones para limitar solo su uso de manera interna. En la Tabla *13* se analizan los registros disponibles y su operación para escritura o lectura.

Tabla 13. Registros internos del SRF02.

| Registros N° | Modo de lectura | Modo de escritura |
|---|---|---|
| 0 | Revisión de software interno | Registros de comandos |
| 1 | No utiliza (se lee 0x80) | No disponible |
| 2 | MSB de la medida realizada | No disponible |
| 3 | LSB de la medida realizada | No disponible |
| 4 | MSB valor mínimo de distancia | No disponible |
| 5 | LSB valor mínimo de distancia | No disponible |

Fuente: SRF02: Sensor de distancias por ultrasonido. [en línea]. <http://www.superrobotica.com/s320122.htm> [citado en 13 de agosto de 2016]

El único registro que permite escritura es el 0, y se utiliza para indicar al sensor que debe realizar un nuevo cálculo. Las medidas tardan alrededor de 65 ms en llevarse a cabo y mientras se realizan el radar no responde a ninguna otra operación que se realice mediante el bus I2C. Si se intenta leer este registro se obtiene la versión del firmware interno del sonar SRF02. El registro 1 está reservado para el procedimiento de auto calibración que soporta el sensor.

En los registros 2 y 3 se almacena el resultado de la última medida realizada, la cual se encuentra en un tamaño de 16 bits ya sea que la medición se haya realizado en pulgadas, centímetros o microsegundos. Un valor de 0 indica que el sensor no ha detectado ningún objeto. Si el maestro ordena realizar una medición antes de los 65 ms, es posible que el sensor no reaccione a dicha orden pues aún se encuentra esperando a que la ráfaga de ultrasonido regrese.

Los registros 4 y 5 almacenan el valor mínimo que el sonar puede medir para las condiciones de operación en que se encuentre. El sensor se auto calibra todo el tiempo, y la distancia mínima medible depende de condiciones como la temperatura del ambiente, la exposición de luz u otras señales que puedan generar algún tipo de interferencia en el desplazamiento de la onda sonora. Este valor se almacena en una variable de 16 bits.

### 12.1.2 Comandos de operación

El sonar SRF02 dispone de varios comandos programados previamente para ejecutar las mediciones según el usuario lo requiera. Permite configurar la unidad de distancia para realizar la medición, así como las condiciones de la muestra e incluso modificar la dirección única del radar en el puerto I2C. Los comandos disponibles para el sensor SRF02 se aprecian en la Tabla 14.

Tabla 14. Comandos disponibles para el sensor SRF02.

| COMANDOS | | DESCRIPCIÓN |
|---|---|---|
| **Decimal** | **Hexadecimal** | |
| 80 | 0x50 | Iniciar nueva medición. Resultado en pulgadas |
| 81 | 0x51 | Iniciar nueva medición. Resultado en centímetros |
| 82 | 0x52 | Iniciar nueva medición. Resultado en microsegundos |
| | | |
| 86 | 0x56 | Medición ráfagas externas. Resultado en pulgadas |
| 87 | 0x57 | Medición ráfagas externas. Resultado en centímetros |
| 88 | 0x58 | Medición ráfagas externas. Resultado en microsegundos |
| | | |
| 92 | 0x5C | Transmite ráfaga de 8 ciclos de 40 KHz. No realiza medición |
| | | |
| 96 | 0x60 | Fuerza un reinicio del sonar SRF02 realizando un autoajuste |
| | | |
| 160 | 0xA0 | 1° comando: secuencia para cambio de dirección I2C |
| 165 | 0xA5 | 3° comando: secuencia para cambio de dirección I2C |
| 170 | 0xAA | 2° comando: secuencia para cambio de dirección I2C |

Fuente: SRF02: Sensor de distancias por ultrasonido. [en línea]. <http://www.superrobotica.com/s320122.htm> [citado en 13 de agosto de 2016]

Para realizar un cálculo de medición se debe escribir uno de los comandos descritos en la Tabla 14 en el registro interno cero (0) y esperar un tiempo aproximado de 66 milisegundos para leer el resultado directamente de los registros 2 y 3, siendo el MSB almacenado en el registro 2 y el LSB almacenado en el registro 3. Juntos conforman el resultado completo de una medición realizada.

## 12.2 MÓDULO I2C_CONTROLLER_SRF02

El módulo I2C_controller_SRF es una variante del módulo I2C_controller descrito anteriormente. Este ha sido rediseñado para permitir al maestro no solo la escritura sino también la lectura desde un dispositivo esclavo, lo cual será necesario para poder establecer la comunicación bidireccional con los sensores SRF02.

El bus $I^2C$ es una interfaz bidireccional estándar que usa un controlador, conocido como el maestro, para comunicarse con dispositivos esclavos. Por regla general, un esclavo no tiene permitido comunicarse o transmitir datos a menos que haya sido direccionado por el maestro. Cada dispositivo conectado al bus $I^2C$ tiene una dirección específica propia que lo diferencia de los otros dispositivos que están conectados en el mismo bus. Una gran cantidad de dispositivos esclavos requerirán de algunos parámetros iniciales para configurar su comportamiento.

La comunicación se inicia por orden del maestro (I2C_controller_SRF02). Éste será el encargado de enviar a través de la línea de datos (SDA) la dirección de registro única del sensor al que desea acceder. Cada SRF02 contiene 6 registros internos donde los datos son almacenados, leídos o escritos.

La interfaz física del protocolo $I^2C$ consta de 2 líneas. Una dedicada al reloj de referencia y otra que permite el flujo de datos seriales. Ambas líneas deben estar conectadas a una tensión de alimentación ($V_{CC}$) a través de una resistencia de pull-up ($R_{PU}$). El tamaño de la resistencia pull-up está determinado por la capacitancia en cada una de las líneas del bus.

La transferencia de debe ser iniciada solo cuando el bus de comunicación se encuentre disponible. El bus es considerado disponible si ambas líneas SDA y SCL se encuentran en estado alto después de una condición de PARE.

El procedimiento general que debe realizar el módulo I2C_controller_SRF02 para enviar datos a un dispositivo esclavo (SRF02) es el siguiente:

- El maestro-transmisor inicia la transferencia con una condición de INICIO.
- El maestro-transmisor envía la dirección del dispositivo al que desea acceder.
- El maestro-transmisor envía la dirección interna en la que desea escribir.
- El maestro-transmisor envía los datos al dispositivo esclavo-receptor.
- El maestro-transmisor finaliza la transferencia con una condición de PARE.

Por otra parte, el procedimiento general que el módulo debe realizar para leer datos almacenados en un dispositivo esclavo (SRF02) tiene algunos pasos adicionales listados a continuación:

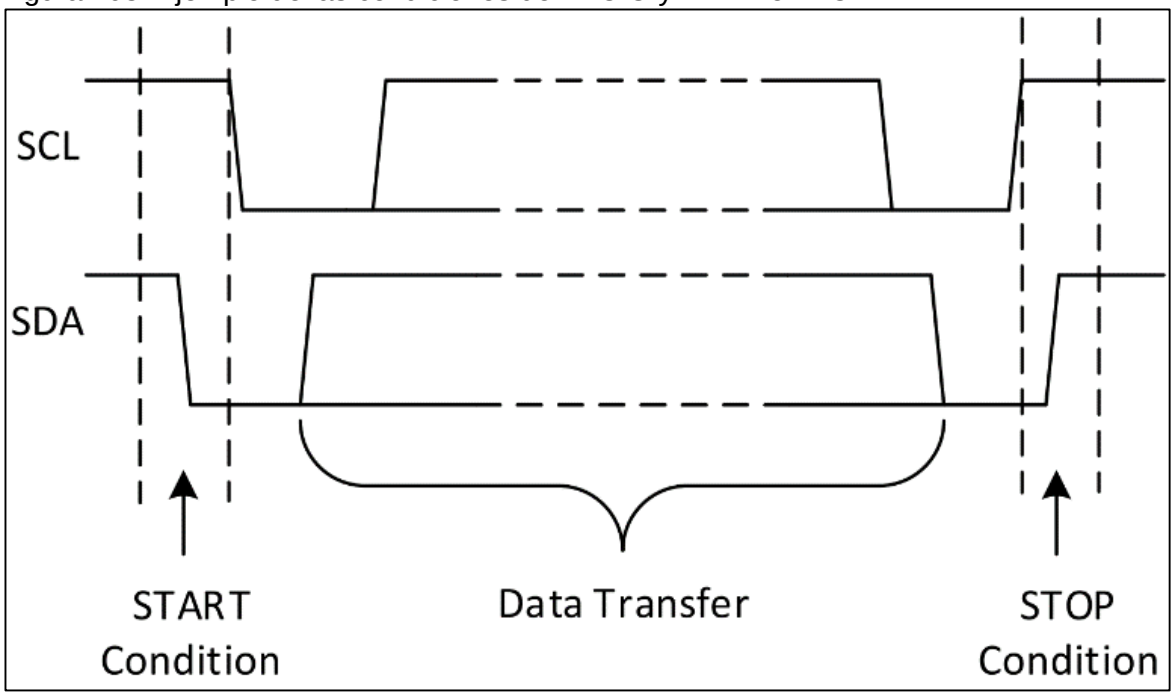
- El maestro-receptor inicia la transferencia con una condición de INICIO.
- El maestro-receptor envía la dirección del dispositivo al que desea acceder.
- El maestro-receptor envía la dirección interna en la que desea leer.
- El maestro-receptor finaliza la transferencia con una condición de PARE.
- El maestro-receptor reinicia la transferencia con una condición de REINICIO.
- El maestro-receptor envía la dirección del dispositivo al que desea acceder.
- El maestro-receptor almacena los datos que el esclavo-transmisor envía.
- El maestro- receptor finaliza la transferencia con una condición de PARE.

### 12.2.1 Condición de INICIO y PARE

La comunicación $I^2C$ con un dispositivo es iniciada por el maestro a través de una condición de INICIO, y se da por finalizada cuando se envía una condición de PARE (ver Figura *103*).

La condición de INICIO está definida como la transición de estado alto a bajo en la línea de datos SDA mientras la línea de reloj SCL se encuentra en estado alto. De este modo, la transición de estado bajo a alto en la línea de datos SDA mientras la línea de reloj SCL se encuentra en alto, define la condición de PARE.

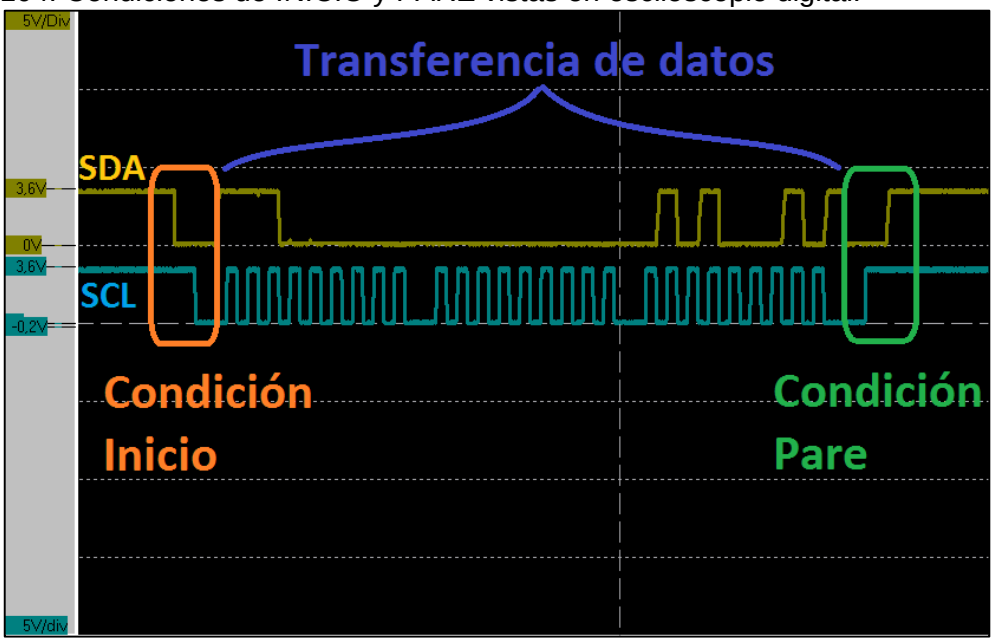
Figura 103. Ejemplo de las condiciones de INICIO y PARE en $I^2C$.




Fuente: SRF02: Sensor de distancias por ultrasonido. [en línea]. <http://www.superrobotica.com/s320122.htm> [citado en 13 de agosto de 2016]

El controlador del puerto $I^2C$ diseñado para comunicarse con los sensores debe garantizar que se respeten los tiempos de uso de la línea SDA, y esto se logra con una buena sincronización en el diseño del algoritmo que envía los pulsos de reloj al bus SCL. En la Figura *104* se aprecian las condiciones de INICIO y PARE que el controlador genera para la comunicación $I^2C$.

Figura 104. Condiciones de INICIO y PARE vistas en osciloscopio digital.




Fuente: Elaboración propia.

Una condición repetida de INICIO, también llamada condición de REINICIO, es una rutina de INICIO que se da justo después de una condición de PARE, es decir, no permite que el bus de comunicación quede en estado disponible, sino que lo ocupa inmediatamente para iniciar una nueva transferencia. Es muy útil cuando un maestro desea iniciar una nueva transferencia, pero no desea dejar el puerto en estado disponible, donde otro maestro podría arrebatarle el control del puerto, o cuando se desea acelerar el proceso de comunicación para evitar tiempos de respuesta más largos. En la Figura *105* se observa la condición de reinicio generada por el módulo I2C_controller_SRF02 vista desde un osciloscopio digital.

Figura 105. Condición de reinicio generada por el módulo I2C_controller_SRF02.


Fuente: Elaboración propia.

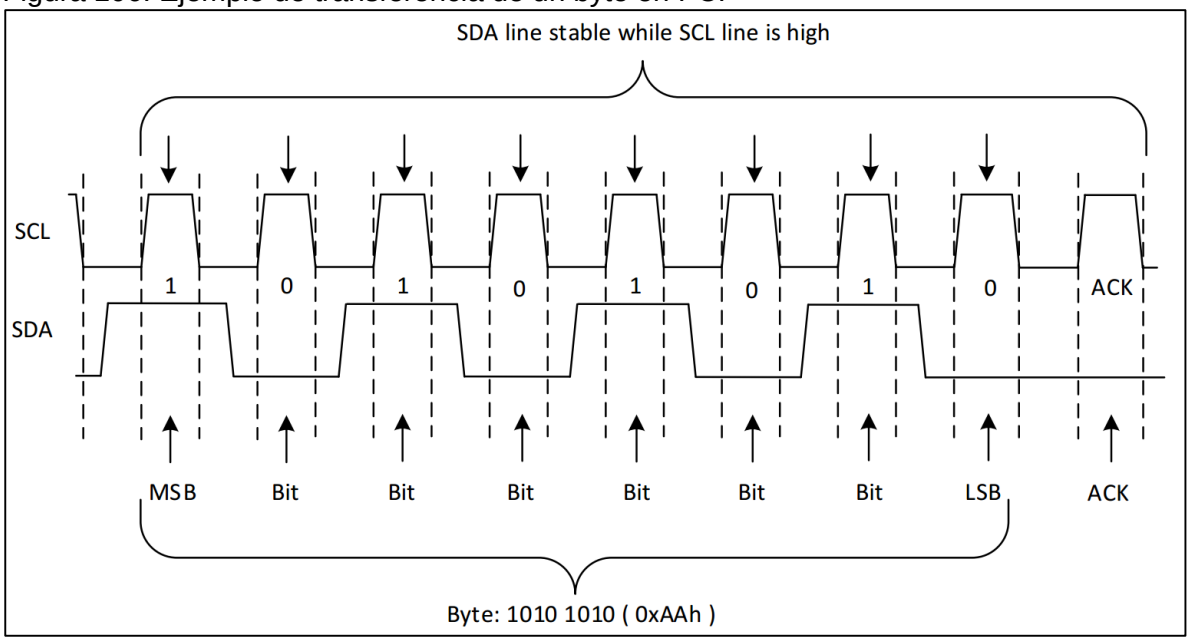
### 12.2.2 Validación de los datos y formato de transferencia

Los datos se envían bit a bit, donde cada uno de ellos es transferido durante cada flanco de subida del reloj SCL. Un byte está compuesto por 8 bits. Cuando el controlador inicia una transferencia con alguno de los sensores, estos esperan recibir al menos un byte de datos a través de la línea SDA. Estos bytes corresponden a la dirección de registro, la dirección de registro interno, o los datos que el controlador desee leer o escribir en el sensor. Los datos se transfieren partiendo del bit más significativo (MSB). No existe un límite de bytes que puedan ser transferidos entre las condiciones de INICIO y PARE para una misma transferencia.

Tanto el procesador (maestro) como los periféricos conectados al bus (esclavos) deben garantizar que los datos en la línea SDA se mantengan estables mientras el reloj se encuentra en la fase alta, puesto que cualquier variación de estado en la línea SDA mientras el reloj se encuentra en estado alto será interpretado como un comando de control de INICIO o PARE.

En la Figura *106* se aprecia el envío de 1 byte en el protocolo $I^2C$, así como el comportamiento de las líneas SCL y SDA en el tiempo.

Figura 106. Ejemplo de transferencia de un byte en $I^2C$.




Fuente: VALDEZ, Jonathan y BECKER, Jared. Understanding the $I^2C$ Bus. En: Texas Instruments Application Report. Junio, 2015. p. 5

#### *12.2.2.1 Reconocimiento (ACK) y no-reconocimiento (NACK)*

Cada byte de datos, incluyendo los bytes de dirección, están seguidos por un bit de reconocimiento el cual envía el receptor y se conoce como ACK. Este bit de

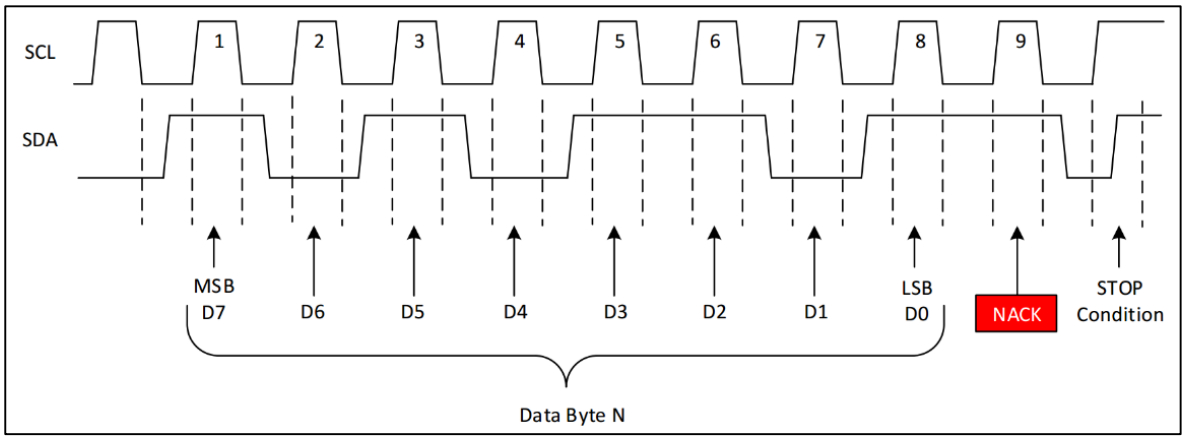
reconocimiento le permite al dispositivo receptor comunicarle al transmisor que el byte que acaba de enviar fue recibido satisfactoriamente y que se encuentra disponible para recibir otro byte.

Antes que el receptor pueda enviar el bit de reconocimiento ACK, el transmisor debe liberar la línea SDA. Para que la comunicación se considere satisfactoria, el dispositivo receptor debe forzar la línea SDA hacia el estado bajo durante el noveno periodo del reloj en la transferencia de cada byte. Este noveno periodo de reloj es el espacio que está destinado en la comunicación para enviar/recibir el bit de reconocimiento.

Cuando la línea SDA se mantiene en estado alto durante la fase positiva del noveno periodo del reloj, se interpreta como un reconocimiento negativo por parte del receptor (NACK) (ver *Figura 107*). Existen varias razones por las cuales se puede obtener un bit NACK las cuales se listan a continuación:

- El receptor es incapaz de recibir o transmitir porque se encuentra ocupado ejecutando alguna función en tiempo real, y no se encuentra listo para continuar la comunicación con el maestro.
- Durante la transferencia, el receptor recibió datos o comandos que no pudo interpretar, o que no son coherentes, como una dirección de registro interna que el dispositivo no contiene.
- Durante la transferencia, el receptor no puede recibir ningún otro dato.
- Cuando el maestro se comporta como receptor, y ya no desea leer más datos de algún esclavo, éste mantiene la línea SDA en alto durante noveno periodo del reloj SCL, indicándole al esclavo que es el último byte al que intenta acceder durante la transferencia actual y que la comunicación va a culminar.

Figura 107. Ejemplo de transferencia interrumpida por un NACK seguido de una condición de PARE.




Fuente: VALDEZ, Jonathan y BECKER, Jared. Understanding the $I^2C$ Bus. En: Texas Instruments Application Report. Junio, 2015. p. 6

### 12.2.3 Escritura y lectura de datos en $I^2C$

En $I^2C$ los datos deben ser enviados y recibidos hacia o desde los dispositivos esclavos. Para que esto se logre, los dispositivos esclavos cuentan con registros internos para almacenar la información. El maestro debe conocer estos registros internos de cada dispositivo al que desee acceder.

Los registros son localizaciones dentro de la memoria de cada esclavo que contienen información, ya sea de configuración o de datos muestreados para ser enviados de regreso al maestro. El maestro debe escribir información en determinados registros (diferentes para cada dispositivo esclavo) con el fin de instruir al dispositivo esclavo para ejecutar una tarea.

Es muy común que los dispositivos esclavos que soportan comunicación $I^2C$ contengan registros, pero es posible que otros carezcan de ellos. Algunos dispositivos están simplificados, y usan solo un registro para recibir y enviar todo a través de él. La mayoría de dispositivos tienen registros dedicados a una única función. Los sensores SRF02 tienen 6 registros internos para configuración como se describen en la Tabla *13*.

#### *12.2.3.1 Escribiendo a un esclavo en el bus $I^2C$*

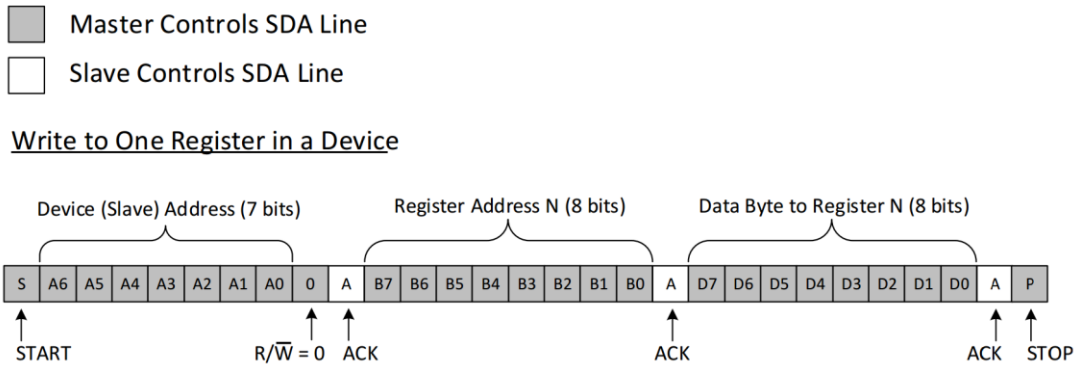
Para escribir en el bus $I^2C$, el maestro debe enviar una condición de INICIO seguida de la dirección del esclavo con quien se desea comunicar. En el estándar $I^2C$, cuando se quiere escribir en un dispositivo esclavo, se debe primero direccionar al dispositivo a través de su dirección única. Después de esto, el maestro debe decidir en cuál de los registros internos del esclavo desea escribir y finalmente, enviar la información que desea escribir. Para cada uno de los pasos a realizar para la escritura sobre un esclavo, el maestro dispone de un byte diferente.

Una vez el maestro inicia una transferencia mediante la condición de INICIO, el byte a enviar corresponde a la dirección del esclavo con quien desea establecer la comunicación. Esta dirección única para cada dispositivo está compuesta por 7 bits. El octavo bit lo tiene a su disposición el maestro-transmisor, y le sirve para indicarle al receptor si la intención de la transferencia es una lectura o una escritura, siendo bajo para escritura (0) y alto para lectura (1). Debido a lo anterior, el bus $I^2C$ convencional soporta hasta 128 dispositivos esclavos simultáneos.

Cuando el maestro inicia una transferencia mediante la condición INICIO, todos los dispositivos esclavos conectados al bus prestan atención y leen la dirección que el maestro envía, pero solo aquel dispositivo cuya dirección ha sido enviada continuará comunicándose con el maestro a lo largo de esa transferencia. Dicho esclavo enviará en el noveno periodo del reloj el bit de reconocimiento ACK, con el cual el maestro entiende que el dispositivo se encuentra listo para la comunicación. Acto seguido, el maestro debe enviar la dirección de registro interna del esclavo a la cual desea escribir, y esta dirección debe estar compuesta por 8 bits. El esclavo debe

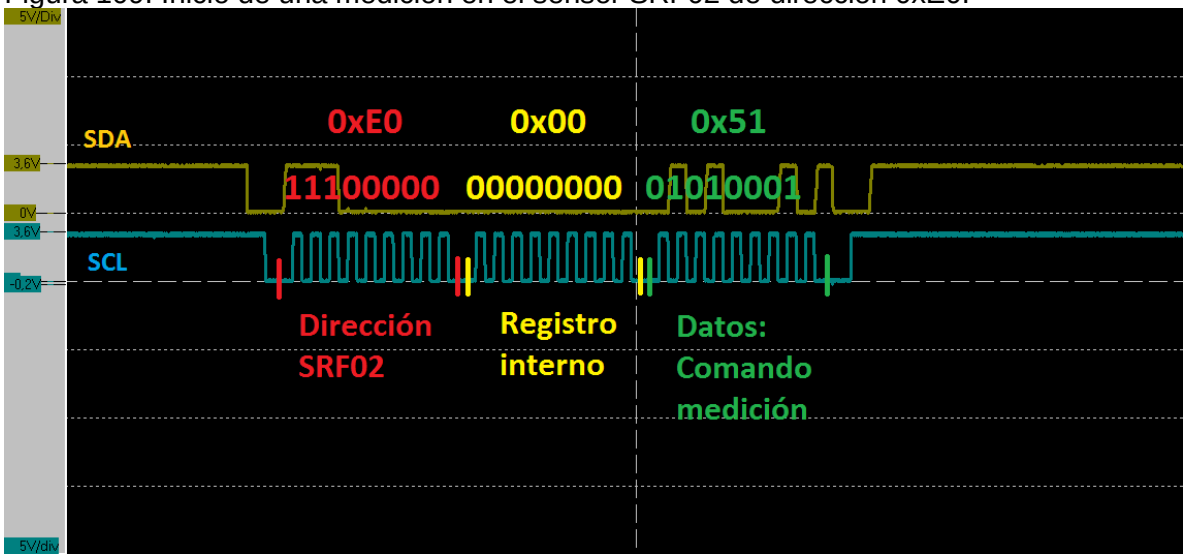
enviar un reconocimiento nuevamente indicando si la comunicación es satisfactoria. El siguiente byte que envía el maestro-transmisor contiene los datos que desea escribir en dicho registro mencionado. Los datos que a enviar pueden contener uno o más bytes, pero el esclavo debe enviar un bit de reconocimiento ACK por cada byte que reciba por parte del transmisor. El maestro da por finalizada la transferencia mediante una condición de PARE. El esquema de escritura en $I^2C$ se muestra en la Figura *108*.

Figura 108. Protocolo I2C para escritura en un dispositivo esclavo.




Fuente: VALDEZ, Jonathan y BECKER, Jared. Understanding the $I^2C$ Bus. En: Texas Instruments Application Report. Junio, 2015. p. 7

Figura 109. Inicio de una medición en el sensor SRF02 de dirección 0xE0.




Fuente: Elaboración propia.

En la Figura *109* se observa un fragmento de la comunicación entre el controlador y el sensor de dirección 0xE0. El módulo direcciona el primer sensor SRF02 enviando en el primer byte de la comunicación el valor 0xE0 por la línea de datos SDA. Acto seguido, envía la dirección de registro interno a la que desea escribir (los sensores SRF02 solo admiten escritura en el registro interno 0x00), y finalmente se envía el

dato 0x51 que se desea escribir en dicho registro interno. Esta rutina se realiza para indicarle al sensor que debe realizar una medición en centímetros, tal como se aprecia en la Tabla 14.

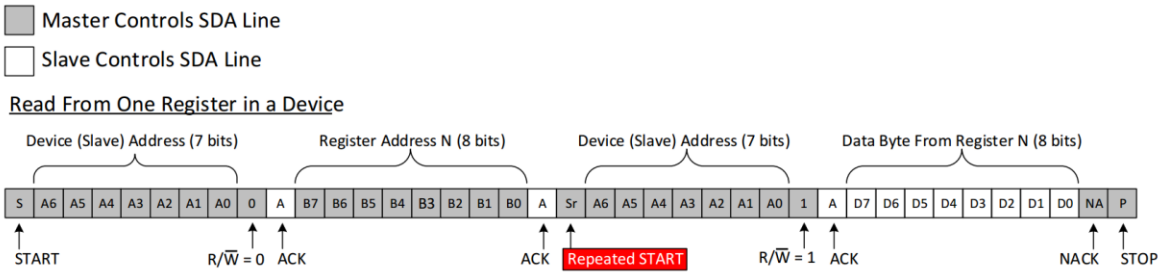
### *12.2.3.2 Leyendo de un esclavo en el bus I²C*

Para leer de un esclavo en el puerto I²C el procedimiento es muy similar al de escritura pero con algunos pasos adicionales. El proceso inicia de la misma manera, el maestro debe enviar una condición de INICIO, y direccionar al dispositivo esclavo al que desea acceder. Una vez hecho esto, el maestro le indica al dispositivo esclavo la dirección del registro interno a partir de la cual desea leer. Hasta este momento, el proceso es el mismo realizado para la escritura.

Ahora es momento de indicarle al dispositivo que se desea leer. Para ello, el maestro debe realizar una condición de RE-INICIO, la cual consiste en realizar las condiciones de PARE seguida de una nueva condición de INICIO. El dispositivo esclavo entiende esto como un intento del maestro por restaurar la comunicación con el fin de comenzar el proceso de extracción de datos. Para ello, el maestro envía nuevamente la dirección única del dispositivo esclavo (7bits), pero en esta ocasión, el 8vo bit de dicha transferencia debe encontrarse en estado alto (1).

Cuando el dispositivo es re direccionado con el bit de lectura activo, se entiende que el maestro liberará la línea SDA y ahora el dispositivo esclavo será quien la controle para enviar datos. Para ello, el maestro debe seguir manteniendo el reloj en la línea SCL, pero esta vez leyendo los datos que llegan a través de la línea de datos seriales. El proceso es ahora inverso, y el dispositivo esclavo-transmisor esperará que el maestro-receptor envíe un bit de reconocimiento (ACK) al final de cada byte transmitido. Si el maestro envía un 0 en el ACK, indica al esclavo que debe enviar otro byte de datos. Cuando el maestro quiera terminar la comunicación, debe enviar un bit de reconocimiento en estado alto (NACK) y seguido a esto, concluir con una condición de PARE. En la Figura *110* se aprecia el esquema para lectura en I²C.

Figura 110. Protocolo I2C para lectura de un dispositivo esclavo.

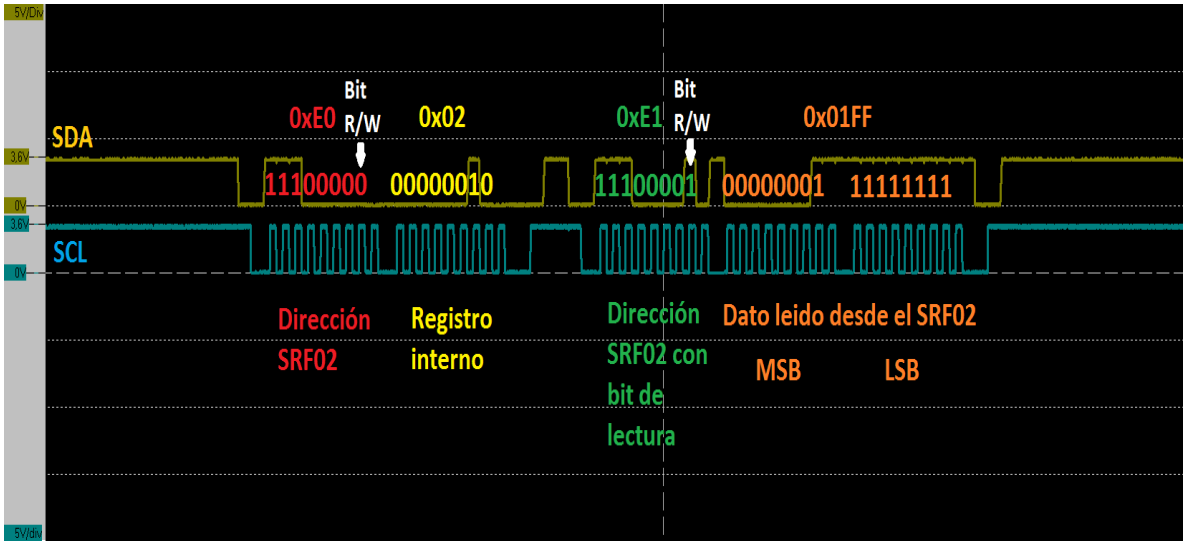



Fuente: VALDEZ, Jonathan y BECKER, Jared. Understanding the I²C Bus. En: Texas Instruments Application Report. Junio, 2015. p. 7

En la Figura *111* se observa un fragmento de la comunicación entre el controlador y el sensor de dirección 0xE0. El módulo direcciona el primer sensor SRF02 enviando

en el primer byte de la comunicación el valor 0xE0 por la línea de datos SDA. Acto seguido, envía la dirección de registro interno a la que desea leer (para leer el resultado de la última medición realizada se apunta al registro interno 0x02). Después de una condición de reinicio, se direcciona nuevamente al sensor pero esta vez indicando en el bit 8 de lectura/escritura que la acción requerida es de lectura. Finalmente el controlador libera la línea SDA y mantiene los pulsos de reloj en la línea SCL para que el sensor envíe los datos de la última medición realizada, y el controlador almacene esta información para ser enviada al módulo controlador de los motores y decidir en qué momentos se debe realizar una pausa en el desplazamiento de la silla de ruedas.

Figura 111. Lectura de los resultados de una medición en el sensor SRF02 de dirección 0xE0.




Fuente: Elaboración propia.

## 12.3 CAPACIDAD DE MEDICIÓN DEL SRF02

El sensor SRF02 tiene un rango de operación de aproximadamente 60° para su haz ultrasónico, según lo presentan Vargas y Sánchez[51] en su trabajo de grado; realizaron mediciones con ayuda de un transportador para conocer el rango de detección de los sensores SRF02.

En las pruebas inicialmente se utilizó un objeto metálico a una distancia de 350cm. Se rotaba el sensor cada 5° para observar la distancia máxima de detección en cada punto de observación. Los resultados de estas mediciones se visualizan en la *Tabla 15*.

[51] VARGAS, Oskar y SÁNCHEZ, Ruth. Manejo de los motores de una silla de ruedas eléctrica mediante comandos de voz. Floridablanca, Colombia, 2014, 61 h. Trabajo de grado (ingeniero electrónico). Universidad Pontificia Bolivariana. Facultad de Ingeniería Electrónica. p. 28

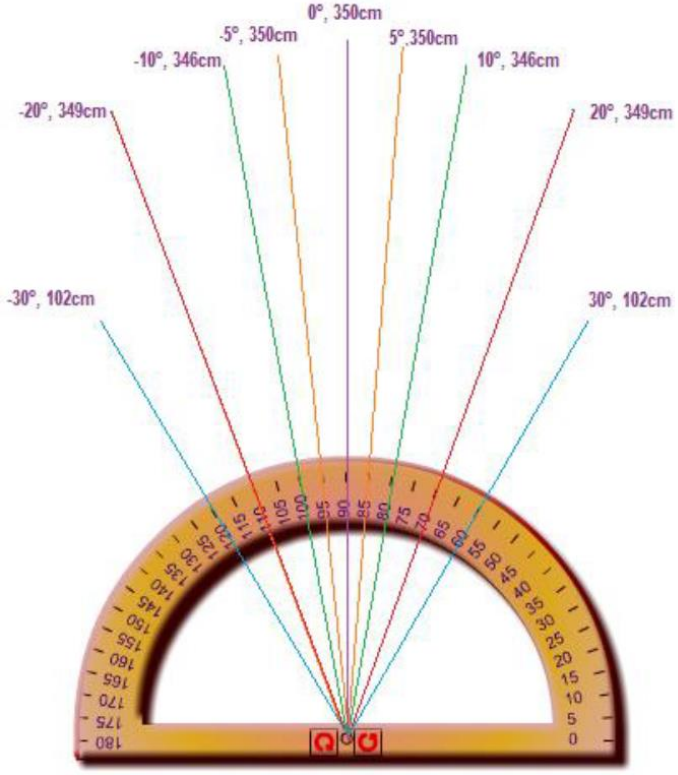
Tabla 15. Distancia máxima medible del sensor SRF02 para diferentes ángulos de vista.

| Grados | Distancia máxima |
|---|---|
| 0° | 350 cm |
| 5° | 350 cm |
| 10° | 346 cm |
| 15° | 348 cm |
| 20° | 349 cm |
| 30° | 102 cm |

Fuente: VARGAS, Oskar y SÁNCHEZ, Ruth. Manejo de los motores de una silla de ruedas eléctrica mediante comandos de voz. Floridablanca, Colombia, 2014, 61 h. Trabajo de grado (ingeniero electrónico). Universidad Pontificia Bolivariana. Facultad de Ingeniería Electrónica. p. 31

Basado en los anteriores datos, en la Figura 112 se visualiza el rango de operación medible (ancho del haz ultrasónico) para el sensor SRF02.

Figura 112. Ancho del haz ultrasónico de un sensor SRF02.




Fuente: VARGAS, Oskar y SÁNCHEZ, Ruth. Manejo de los motores de una silla de ruedas eléctrica mediante comandos de voz. Floridablanca, Colombia, 2014, 61 h. Trabajo de grado (ingeniero electrónico). Universidad Pontificia Bolivariana. Facultad de Ingeniería Electrónica. p. 32

# 13 EVALUACIÓN Y CONFIGURACIÓN FINAL DEL SISTEMA

La evaluación del sistema se realiza mediante pruebas para determinar la eficiencia de software implementado de reconociendo de voz en la ARM y además de comprobar con el rendimiento se finaliza con la construcción de archivos para dejar en funcionamiento el software y hardware de manera predeterminada y fija (no volátil).

## 13.1 EVALUACIÓN

Una de las formas para determinar o calificar el funcionamiento del reconocimiento de voz es evaluar distintas frases. Con un 20 de personas voluntarias accedieron a estas pruebas. Es importante resalta la búsqueda del error y la precisión del software.

La prueba consiste en que la persona debe decir 10 frases, cada una con una repetición de 5 veces para luego tomar los resultado y realizar los cálculos. En la Tabla *16* se puede observar las frases y demás datos e indicaciones sobre el registro de la prueba.

Tabla 16. Evaluación del sistema.

| ***Registro de Evaluación*** | | | | **Reconocimiento de voz** | | | |
|---|---|---|---|---|---|---|---|
| **Cant. de Personas** | **Sexo** | | | **Fecha** | **Universidad** | **Frases** | |
| 20 | M | ó | F | 12-9-2016 | Pontificia Bolivariana | 1. "Jota como está" | 6. "Quiero retroceder" |
| **Testigo 1** | | | | **Relación** | | 2. "Jota adelante" | 7. "Por favor avanzar" |
| Juan Daniel Prieto A. | | | | Desarrollador del Proyecto | | 3. "Doblar a la derecha" | 8. "Escucha por favor" |
| **Testigo 2** | | | | **Relación** | | 4. "Un dos tres metro" | 9. "Detenerse quieto" |
| Jefferson Manuel Garcia P. | | | | Desarrollador del Proyecto | | 5. "Ir hacia la izquierda" | 10. "Grados escucha" |

Fuente: elaboración propia.

El porcentaje de error se determina mediante la siguiente ecuación 31:

$$WER = \frac{I+D+S}{N} \quad (31)$$

Donde:
$N =$ número de palabras reales
$I =$ número de inserciónes en la frase
$D =$ número de palabras eliminadas de la frase
$S =$ número de palabras sustituidas de la frase
$WER =$ tasa de error de la frase y se suele medir en porcentaje.

Precisión determinada por la ecuación 32:

$$Precisión = \frac{N-D-S}{N} \quad (32)$$

Tabla 17. Evaluación femenina 1-3.

| Género femenino | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Usuario 1 | | | Usuario 2 | | | Usuario 3 | | |
| **Frase** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** |
| 1 | jota como esta | 0 | 100 | jota como esta | 0 | 100 | pare como esta | 33.3 | 66.6 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | la la corrija | 100 | 0 |
| | jota corrija | 66.6 | 33.3 | jota como esta | 0 | 100 | pare corrija pare | 100 | 0 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | a la corrija | 100 | 0 |
| | cuatro corrija pare | 100 | 0 | a la corrija | 100 | 0 | pare por la esta | 75 | 25 |
| 2 | jota adelante | 0 | 100 | jota adelante | 0 | 100 | la adelante | 50 | 50 |
| | jota adelante | 0 | 100 | a adelante | 50 | 50 | la adelante | 50 | 50 |
| | jota adelante | 0 | 100 | pare adelante | 50 | 50 | jota adelante | 0 | 100 |
| | cuatro adelante | 50 | 50 | jota adelante | 0 | 100 | cuatro adelante un | 66.6 | 33.3 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| 3 | doblar a la derecha | 0 | 100 | la la la derecha | 50 | 50 | doblar a la derecha | 0 | 100 |
| | doblar derecha | 50 | 50 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | la la la derecha | 50 | 50 | la la la derecha | 50 | 50 | la a la a metro | 75 | 50 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| 4 | un dos tres metro | 0 | 100 | dos tres metro | 25 | 75 | hola esta dos metro | 50 | 50 |
| | un a pare metro | 50 | 50 | un dos tres metro | 0 | 100 | un a la dos metro | 50 | 75 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | dos tres metro | 25 | 75 | un a metro | 50 | 50 |
| | un detener | 75 | 25 | un dos tres me | 25 | 75 | un dos tres metro | 0 | 100 |
| 5 | la la izquierda | 50 | 50 | ir a la izquierda | 25 | 75 | ir a la la izquierda | 0 | 100 |
| | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 | ir a la la izquierda | 0 | 100 |
| | ir a la izquierda | 25 | 75 | ir a la izquierda | 25 | 75 | ir a la la izquierda | 0 | 100 |
| | quieto izquierda | 75 | 25 | ir a quiero izquierda | 50 | 50 | ir a la la izquierda | 0 | 100 |
| | la izquierda | 50 | 50 | ir a me izquierda | 50 | 50 | ir a la la izquierda | 0 | 100 |
| 6 | quiero retroceder | 0 | 100 | la detenerse | 100 | 0 | la retroceder | 50 | 50 |
| | quiero tres | 50 | 50 | girar retroceder | 50 | 50 | la tres | 100 | 0 |
| | quiero retroceder | 0 | 100 | girar retroceder | 50 | 50 | la adelante | 100 | 0 |
| | la retroceder | 50 | 50 | quiero retroceder | 50 | 50 | la retroceder | 50 | 50 |
| | la retroceder | 50 | 50 | ciento la adelante | 100 | 0 | la retroceder | 50 | 50 |
| 7 | por favor avanzar | 0 | 100 | corrija avanzar | 66.6 | 33.3 | pare favor avanzar | 33.3 | 66.6 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | pare avanzar | 66.6 | 33.3 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor detener | 33.3 | 66.6 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor atrás | 33.3 | 66.6 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| 8 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | metro por favor | 33.3 | 66.6 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | la cuatro por favor | 66.6 | 66.6 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | cuatro por favor | 33.3 | 66.6 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| 9 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detener quieto | 50 | 50 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detener a quieto | 66.6 | 33.3 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detener quieto | 50 | 50 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detener a quieto | 50 | 50 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detener a la quieto | 75 | 33.3 |
| 10 | grados escucha | 0 | 100 | grados escucha | 0 | 100 | la escucha | 50 | 50 |
| | grados escucha | 0 | 100 | grados detener | 50 | 50 | la dos corrija | 100 | 0 |
| | grados escucha | 0 | 100 | la dos cuatro | 100 | 0 | la escucha | 50 | 50 |
| | la escucha | 50 | 50 | la dos tres por | 100 | 0 | la la la cuatro | 100 | 0 |
| | la jota | 100 | 0 | grados escucha | 0 | 100 | la cuatro | 100 | 0 |
| | **Total promedio %** | **17,9** | **82,1** | **Total promedio %** | **23,0** | **77,0** | **Total promedio %** | **41,9** | **60,3** |

Fuente: elaboración propia.

Tabla 18. Evaluación femenina 4-6.

| Género femenino | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Usuario 4 | | | | Usuario 5 | | | Usuario 6 | | |
| **Frase** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** |
| 1 | jota como esta | 0 | 100 | jota corrija | 66.6 | 33.3 | jota como esta | 0 | 100 |
| | jota como esta | 0 | 100 | jota corrija | 66.6 | 33.3 | jota como esta | 0 | 100 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| 2 | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| 3 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la ir | 25 | 75 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | a a la derecha | 25 | 75 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| 4 | un dos detener me | 50 | 50 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | dos tres metro | 25 | 75 | un dos tres metro | 0 | 100 | un dos tres la por | 50 | 75 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| 5 | me hacia la izquierda | 25 | 75 | ir hacia la izquierda | 0 | 100 | ir a quiero quiero | 50 | 50 |
| | me hacia la izquierda | 25 | 75 | la la la izquierda | 50 | 50 | ir hacia la izquierda | 0 | 100 |
| | diez a la izquierda | 50 | 50 | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 |
| | me a la la izquierda | 75 | 50 | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 |
| | me corrija la izquierda | 50 | 50 | la la la izquierda | 50 | 50 | ir hacia la izquierda | 0 | 100 |
| 6 | quiero retroceder | 0 | 100 | quiero la retroceder | 33.3 | 66.6 | quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | quiero la retroceder | 33.3 | 66.6 | quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | quiero detenerse | 50 | 50 | quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | la la retroceder | 66.6 | 33.3 | quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | la detenerse | 100 | 0 | quiero retroceder | 0 | 100 |
| 7 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| 8 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | tres cuatro por favor | 66.6 | 66.6 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| 9 | detener un quieto | 66.6 | 50 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detener dos quieto | 66.6 | 50 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detener ciento | 100 | 0 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detener ciento | 100 | 0 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detener quieto | 50 | 50 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| 10 | grados escucha | 0 | 100 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | grados escucha | 0 | 100 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | la dos escucha | 66.6 | 50 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | la dos escucha | 66.6 | 50 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | la dos escucha | 66.6 | 50 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | **Total promedio %** | **13,3** | **83,5** | **Total promedio %** | **5,6** | **94,4** | **Total promedio %** | **2,0** | **98,5** |

Fuente: elaboración propia.

Tabla 19. Evaluación femenina 7-9.

| Género femenino | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Usuario 7 | | | | Usuario 8 | | | Usuario 9 | | |
| **Frase** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** |
| 1 | jota como esta | 0 | 100 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | jota corrija | 66.6 | 33.3 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | jota como corrija | 66.6 | 33.3 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | corrija corrija | 100 | 0 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| 2 | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | ochenta adelante | 50 | 50 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota la detener | 66.6 | 66.6 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | cuatro adelante | 50 | 50 |
| 3 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| 4 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos detener metro | 25 | 75 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos detener | 50 | 50 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos pare detener | 50 | 50 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres pare | 25 | 75 |
| 5 | quiero la izquierda | 50 | 50 | ir a la la izquierda | 50 | 75 | giro a la la quiero | 83 | 16 |
| | la la la izquierda | 50 | 50 | ir a la izquierda | 25 | 75 | ir a la la izquierda | 40 | 80 |
| | ir a la la izquierda | 50 | 75 | ir a la izquierda | 25 | 75 | ir a la adelante | 50 | 50 |
| | gire a la la izquierda | 75 | 50 | ir a la la izquierda | 50 | 75 | ciento a la izquierda | 50 | 50 |
| | ir a la la izquierda | 50 | 75 | ir a la la izquierda | 50 | 75 | girar a la izquierda | 50 | 50 |
| 6 | quiero retroceder | 0 | 100 | la retroceder | 50 | 50 | ciento retroceder | 50 | 50 |
| | quiero retroceder | 0 | 100 | quiero a retroceder | 50 | 75 | la detenerse | 100 | 0 |
| | quiero retroceder | 0 | 100 | la retroceder | 50 | 50 | la retroceder | 50 | 50 |
| | quiero retroceder | 0 | 100 | la retroceder | 50 | 50 | ciento retroceder | 50 | 50 |
| | quiero retroceder | 0 | 100 | a retroceder | 50 | 50 | quiero detenerse | 50 | 50 |
| 7 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor a corrija | 40 | 75 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor a esta | 40 | 75 |
| | cuatro a la doblar | 100 | 0 | por favor avanzar | 0 | 100 | por favor a | 33.3 | 66.6 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| 8 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | cuatro por favor | 33.3 | 66.6 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| 9 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse ciento | 0 | 50 | detener pare quieto | 66.6 | 33.3 |
| | detenerse quieto | 0 | 100 | detenerse ciento | 0 | 50 | detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detener a quieto | 33.3 | 66.6 |
| 10 | la dos cuatro | 0 | 100 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | la dos escucha | 0 | 100 | la dos escucha | 66.6 | 66.6 | la la escucha | 66.6 | 66.6 |
| | grados escucha | 0 | 100 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | grados escucha | 0 | 100 | grados escucha | 0 | 100 | grados por escuchar | 33.3 | 66.6 |
| | grados escucha | 0 | 100 | la dos escucha | 66.6 | 66.6 | la la por | 100 | 0 |
| | **Total promedio %** | **7,7** | **93,9** | **Total promedio %** | **9,4** | **90,6** | **Total promedio %** | **26,3** | **74,9** |

Fuente: elaboración propia.

Tabla 20. Evaluación masculina 1-3.

| Género masculino | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Usuario 1 | | | | Usuario 2 | | | Usuario 3 | | |
| **Frase** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** |
| 1 | jota corrija | 66.6 | 33.3 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| | jota como atrás | 33.3 | 66.6 | jota como esta | 0 | 100 | jota como derecha | 33.3 | 66.6 |
| | jota como tres atrás | 66.6 | 75 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| | jota como atrás | 33.3 | 66.6 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| | jota como atrás | 33.3 | 66.6 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| 2 | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| 3 | doblar la derecha | 25 | 75 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar pare derecha | 50 | 50 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| 4 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| 5 | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 | ir pare izquierda | 50 | 50 |
| | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 | r a la hacia la izquierda | 33.3 | 66.6 |
| | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 |
| | ir corrija izquierda | 50 | 50 | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 |
| | ir hacia izquierda | 25 | 75 | ir hacia la izquierda | 0 | 100 | ir pare izquierda | 0 | 100 |
| 6 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 |
| 7 | corrija favor avanzar | 33.3 | 66.6 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| | corrija corrija | 100 | 0 | por favor avanzar | 0 | 100 | la por favor avanzar | 33.3 | 66.6 |
| | corrija corrija | 100 | 0 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| 8 | escucha por favor | 0 | 100 | a escucha por favor | 25 | 75 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| 9 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detener a quieto | 0 | 100 | detener la quieto | 33.3 | 66.6 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detener pare quieto | 33.3 | 66.6 |
| 10 | grados escucha | 0 | 100 | grados escucha | 0 | 100 | la escucha | 50 | 50 |
| | grados escucha | 0 | 100 | dos escucha | 50 | 50 | me la dos escucha | 75 | 50 |
| | grados escucha | 0 | 100 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | grados escucha | 0 | 100 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | grados escucha | 0 | 100 | dos escucha | 50 | 50 | grados escucha | 0 | 100 |
| | **Total promedio %** | **6,8** | **92,8** | **Total promedio %** | **3,5** | **96,5** | **Total promedio %** | **3,9** | **96,7** |

Fuente: elaboración propia.

Tabla 21. Evaluación masculina 4-6.

| Género masculino | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Usuario 4 | | | Usuario 5 | | | Usuario 6 | | |
| **Frase** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** |
| 1 | jota como esta | 0 | 100 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| | jota como atrás | 33.3 | 100 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| | jota como esta | 0 | 100 | jota corrija | 66.6 | 33.3 | jota como esta | 0 | 100 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 | jota corrija | 66.6 | 33.3 |
| 2 | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota un adelante | 33.3 | 66.6 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | jota adelante | 0 | 100 |
| 3 | doblar pare derecha | 66.6 | 33.3 | doblar a la derecha | 0 | 100 | doblar la derecha | 25 | 75 |
| | doblar a la derecha | 0 | 100 | hola la derecha | 50 | 50 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar la derecha | 25 | 75 | doblar a la derecha | 0 | 100 |
| | doblar corrija | 75 | 25 | doblar derecha | 50 | 50 | doblar a la derecha | 0 | 100 |
| | la a la derecha | 50 | 50 | doblar la derecha | 25 | 75 | doblar a la derecha | 0 | 100 |
| 4 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos detener | 50 | 50 | un dos tres cuatro | 25 | 75 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | un dos tres cuatro | 25 | 75 |
| 5 | ir hacia la izquierda | 0 | 100 | girar izquierda | 75 | 25 | ir hacia la izquierda | 0 | 100 |
| | ir adelante | 50 | 50 | ir pare la izquierda | 25 | 75 | la derecha quiero | 75 | 25 |
| | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 |
| | diez la izquierda | 50 | 50 | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 |
| | giro hacia la izquierda | 25 | 75 | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 |
| 6 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 |
| | diez un retroceda | 100 | 0 | quiero retroceda | 50 | 50 | quiero retroceda | 50 | 50 |
| | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | quiero retroceda | 50 | 50 |
| 7 | por favor avanzar | 0 | 100 | un favor avanzar | 33.3 | 66.6 | por favor avanzar | 0 | 100 |
| | por favor avanzar | 0 | 100 | corrija un avanzar | 66.6 | 33.3 | por favor avanzar | 0 | 100 |
| | un favor avance | 33.3 | 66.6 | corrija un avanzar | 66.6 | 33.3 | por favor avanzar | 0 | 100 |
| | corrija corrija esta | 100 | 0 | corrija un avanzar | 66.6 | 33.3 | por favor avanzar | 0 | 100 |
| | por favor metro | 33.3 | 66.6 | corrija un avanzar | 66.6 | 33.3 | por favor avanzar | 0 | 100 |
| 8 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha un por favor | 33.3 | 66.6 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha un por favo | 33.3 | 66.6 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| 9 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detener pare quieto | 33.3 | 66.6 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detener | 100 | 0 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 |
| 10 | grados escucha | 0 | 100 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | grados escucha | 0 | 100 | la dos escucha | 50 | 50 | grados escucha | 0 | 100 |
| | grados escucha | 0 | 100 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | grados escucha | 0 | 100 | grados escucha | 0 | 100 | grados escucha | 0 | 100 |
| | dos escucha | 50 | 50 | la dos escucha | 50 | 50 | grados escucha | 0 | 100 |
| | **Total promedio %** | **13,6** | **86,7** | **Total promedio %** | **10,47** | **89,5** | **Total promedio %** | **5,2** | **94,8** |

Fuente: elaboración propia.

Tabla 22. Evaluación masculina 7-9.

| Género masculino | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Usuario 7 | | | | Usuario 8 | | | Usuario 9 | | |
| **Frase** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** |
| 1 | jota como girar | 33.3 | 66.6 | jota como esta | 0 | 100 | Jota como esta | 0 | 100 |
| | jota corrija | 66.6 | 33.3 | jota como esta | 0 | 100 | Jota como esta | 0 | 100 |
| | jota corrija | 66.6 | 33.3 | jota corrrija | 66.6 | 33.3 | Jota como esta | 0 | 100 |
| | jota como doblar | 33.3 | 66.6 | jota como esta | 0 | 100 | Jota como esta | 0 | 100 |
| | jota corrija | 66.6 | 33.3 | jota como esta | 0 | 100 | Un jota como esta | 33.3 | 75 |
| 2 | jota adelante | 0 | 100 | jota adelante | 0 | 100 | Jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | Jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | Jota pare la | 66.6 | 33.3 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | Jota adelante | 0 | 100 |
| | jota adelante | 0 | 100 | jota adelante | 0 | 100 | Jota adelante | 0 | 100 |
| 3 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | Doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | Doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | Doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | Doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 | Doblar a la derecha | 0 | 100 |
| 4 | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | Un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | Un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | Un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | Un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 | Un dos tres metro | 0 | 100 |
| 5 | ir hacia la izquierda | 0 | 100 | girar izquierda | 75 | 25 | Ir hacia la izquierda | 0 | 100 |
| | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 | Ir hacia la izquierda | 0 | 100 |
| | ir hacia la izquierda | 0 | 100 | girar la izquierda | 50 | 50 | Ir hacia la izquierda | 0 | 100 |
| | diez atrás la izquierda | 50 | 50 | sta hacia la izquierd | 50 | 50 | Ir hacia la izquierda | 0 | 100 |
| | diez hacia la izquierda | 50 | 50 | ir hacia la la quiero | 25 | 75 | Ir hacia la izquierda | 0 | 100 |
| 6 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | Quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | Quiero retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 | Quieto retroceder | 0 | 100 |
| | quiero retroceder | 0 | 100 | la retroceder | 50 | 50 | Quiero retroceder | 0 | 100 |
| | quiero retroceda | 50 | 50 | quiero retroceder | 0 | 100 | Quiero retroceder | 0 | 100 |
| 7 | corrija favor avanzar | 33.3 | 66.6 | por favor avanzar | 0 | 100 | La por favor avanzar | 25 | 75 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | La por favor avanzar | 25 | 75 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | Por favor avanzar | 0 | 100 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | Por favor avanzar | 0 | 100 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 | Por favor avanzar | 0 | 100 |
| 8 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | Escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | Escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | La escucha por favor | 25 | 75 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | Escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 | Escucha por favor | 0 | 100 |
| 9 | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | Detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | Detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | Detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | Detenerse quieto | 0 | 100 |
| | detenerse quieto | 0 | 100 | detenerse quieto | 0 | 100 | Detener quieto | 50 | 50 |
| 10 | grados escucha | 0 | 100 | grados escucha | 0 | 100 | Grados escucha | 0 | 100 |
| | grados escucha | 0 | 100 | la avance escucha | 66.6 | 33.3 | Grados escucha | 0 | 100 |
| | grados escucha | 0 | 100 | grados escucha | 0 | 100 | Grados escucha | 0 | 100 |
| | grados escucha | 0 | 100 | grados escucha | 0 | 100 | Grados | 50 | 50 |
| | grados escucha | 0 | 100 | la dos escucha | 66.6 | 33.3 | Grados escucha | 0 | 100 |
| | **Total promedio %** | **3,4** | **96,6** | **Total promedio %** | **5,319** | **94,7** | **Total promedio %** | **3,6** | **95,9** |

Fuente: elaboración propia.

Tabla 23. Evaluación usuarios 10 - femenino y masculino.

| | Usuario 10 femenino | | | Usuario 10 masculino | | |
|---|---|---|---|---|---|---|
| **Frase** | **Resultado Texto** | **WER %** | **Precisión %** | **Resultado Texto** | **WER %** | **Precisión %** |
| 1 | cuatro como esta | 33.3 | 66.6 | cuatro corrija | 100 | 0 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| | cuatro como izquierda | 66.6 | 33.3 | corrija corrija | 100 | 0 |
| | jota como esta | 0 | 100 | jota corrija pare | 66.6 | 33.3 |
| | jota como esta | 0 | 100 | jota como esta | 0 | 100 |
| 2 | cuatro adelante | 50 | 50 | jota delante | 0 | 100 |
| | jota adelante | 0 | 100 | jota delante | 0 | 100 |
| | jota adelante | 0 | 100 | jota delante | 0 | 100 |
| | jota adelante | 0 | 100 | jota delante | 0 | 100 |
| | cuatro adelante | 50 | 50 | jota delante | 0 | 100 |
| 3 | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar la derecha | 25 | 75 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| | doblar a la | 25 | 75 | doblar a la derecha | 0 | 100 |
| | doblar a la derecha | 0 | 100 | doblar a la derecha | 0 | 100 |
| 4 | un la detener me | 75 | 25 | un dos tres metro | 0 | 100 |
| | un dos pare me | 50 | 50 | un dos tres metro | 0 | 100 |
| | un atrás tres me | 50 | 50 | un dos tres metro | 0 | 100 |
| | un dos tres metro | 0 | 100 | un dos tres metro | 0 | 100 |
| | un dos tres me | 25 | 75 | un dos tres metro | 0 | 100 |
| 5 | ir a la izquierda | 25 | 75 | ir a la la izquierda | 50 | 75 |
| | ir a la la izquierda | 50 | 75 | ir hacia la izquierda | 0 | 100 |
| | ir la izquierda | 25 | 75 | ir hacia la izquierda | 0 | 100 |
| | ir a la la la quiero | 75 | 75 | ir hacia la izquierda | 0 | 100 |
| | ir hacia la izquierda | 0 | 100 | ir hacia la izquierda | 0 | 100 |
| 6 | quiero retroceder | 0 | 100 | quiero retroceder | 0 | 100 |
| | la a la atrás me | 100 | 0 | quiero retroceder | 0 | 100 |
| | hacia a la detenerse | 100 | 0 | quiero retroceder | 0 | 100 |
| | la retroceder | 50 | 50 | quiero retroceder | 0 | 100 |
| | la a la detenerse | 100 | 0 | quiero retroceder | 0 | 100 |
| 7 | pare favor avanzar | 33.3 | 66.6 | por favor avanzar | 0 | 100 |
| | pare favor avanzar | 33.3 | 66.6 | corrija por avanzar | 33.3 | 66.6 |
| | por favor avanzar | 0 | 100 | por favor avanzar | 0 | 100 |
| | corrija dos pare | 100 | 0 | por favor avanzar | 0 | 100 |
| | pare favor | 66.6 | 33.3 | por favor avanzar | 0 | 100 |
| 8 | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | cuatro por favor | 33.3 | 66.6 | escucha por favor | 0 | 100 |
| | la cuatro por favor | 66.6 | 66.6 | escucha por favor | 0 | 100 |
| | escucha por favor | 0 | 100 | escucha por favor | 0 | 100 |
| | cuatro por favor | 33.3 | 66.6 | escucha por favor | 0 | 100 |
| 9 | detener a quieto | 33.3 | 66.6 | detener pare quieto | 33.3 | 75 |
| | detener a quieto | 33.3 | 66.6 | detener pare quieto | 33.3 | 75 |
| | detener a quieto | 33.3 | 66.6 | detener pare quieto | 33.3 | 75 |
| | detener a quieto | 33.3 | 66.6 | detener pare quieto | 33.3 | 75 |
| | detener a esta | 66.6 | 33.3 | detener pare quieto | 33.3 | 75 |
| 10 | la cuatro | 100 | 0 | la dos tres escucha | 75 | 25 |
| | la cuatro | 100 | 0 | la dos tres cuatro | 100 | 0 |
| | la cuatro | 100 | 0 | la do tres cuatro | 100 | 0 |
| | la cuatro | 100 | 0 | la dos tres jota | 100 | 0 |
| | la dos cuatro | 100 | 0 | grados la jota | 66.6 | 33.3 |
| | **Total promedio %** | **39,2** | **62,8** | **Total promedio %** | **15,48** | **84,0** |

Fuente: elaboración propia.

Una vez obtenido los datos se realizan los cálculos para establecer el porcentaje de error y la precisión por género. Además en la Tabla *24* también se puede observar para ambos géneros.

Tabla 24. Resultados WER.

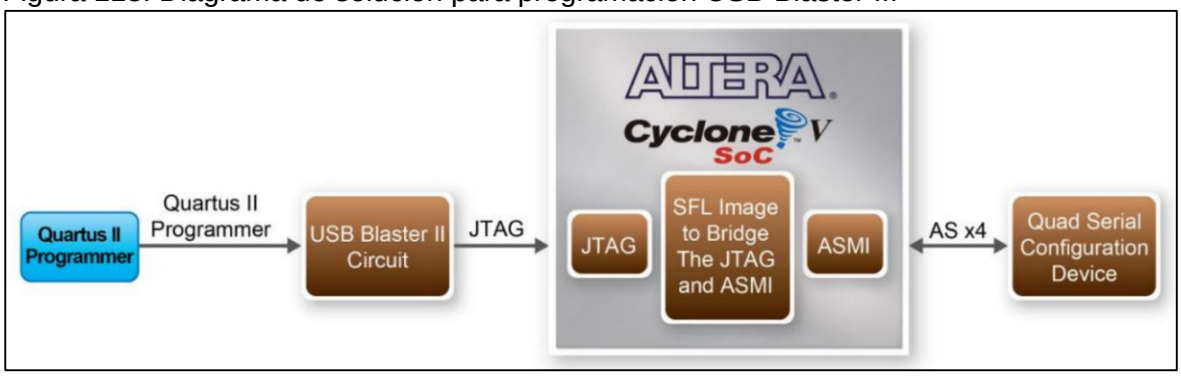
| Genero | WER % | Precisión % |
|---|---|---|
| Femenino | 18,6 | 81,8 |
| Masculino | 7,1 | 92,8 |
| Ambos | 12,9 | 87,3 |

Fuente: elaboración propia.

## 13.2 CONFIGURACIÓN PARA EJECUTAR SOFTWARE Y HARDWARE DE MANERA NO VOLÁTIL

En la FPGA una vez terminado la construcción del hardware programado es necesario grabar en la EPCS un archivo que permita la acción de iniciar la FPGA con el código programado y evitar reprogramar cada vez que se utilice. Esta configuración se logra con la función de Serial Flash Loader (SFL) a través de la interfaz JTAG (ver Figura *113*).

Figura 113. Diagrama de solución para programación USB Blaster II.

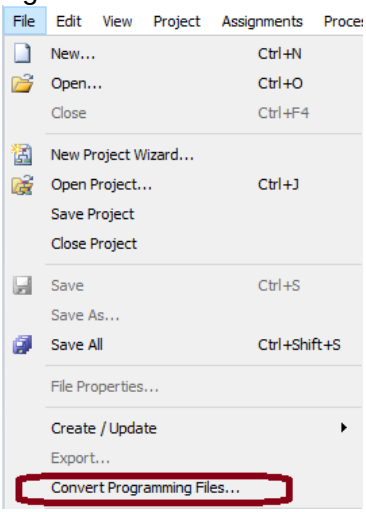



Fuente: DE1-SoC User Manual Agosto 5 del 2015.

Se puede programar los dispositivos CPE con un archivo de configuración JTAG indirecta (.jic), que se convierte a partir de un archivo de objeto especificado por el diseñador (.sof) en Quartus II. El archivo .sof se genera después de la compilación proyecto tiene éxito. Las etapas de convertir a .sof a .jic en Quartus II se describen a continuación:

- Seleccionar Convert Programming Files ubicado en la pestaña File de Quartus II como se muestra en la Figura *114*.

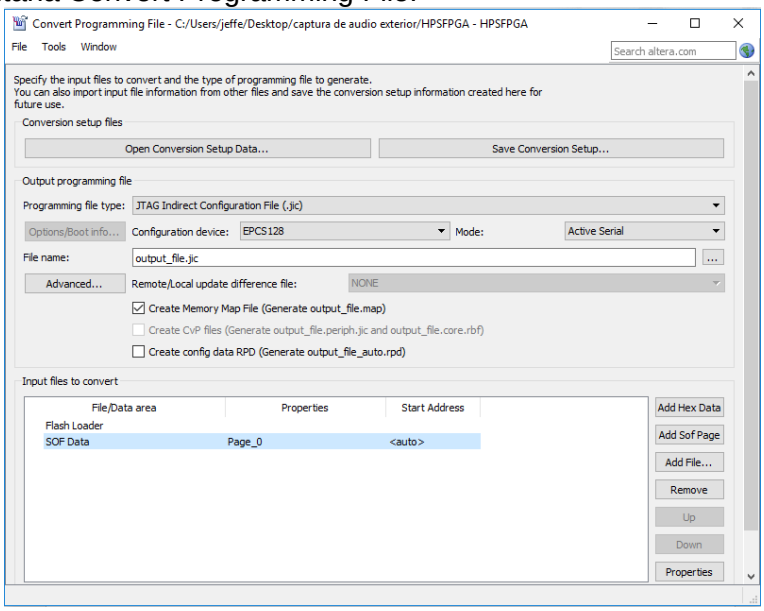
Figura 114. Convertir archivo programado.




Fuente: DE1-SoC User Manual Agosto 5 del 2015.

- Seleccionar el archivo de configuración JTAG indirecta (.jic) desde la pestaña de tipo de archivo de programación en el cuadro de diálogo de convertir archivos de programación.
- Elegir EPCS128 desde el campo de los dispositivos de configuración.
- Elegir serie activa desde el modo presentado.
- Examinar el directorio de destino en la pestaña Nombre de archivo y especificar el nombre del archivo de salida.
- Hacer clic en los datos SOF en la sección de archivos de entrada para convertir, como se muestra en la Figura *115*.

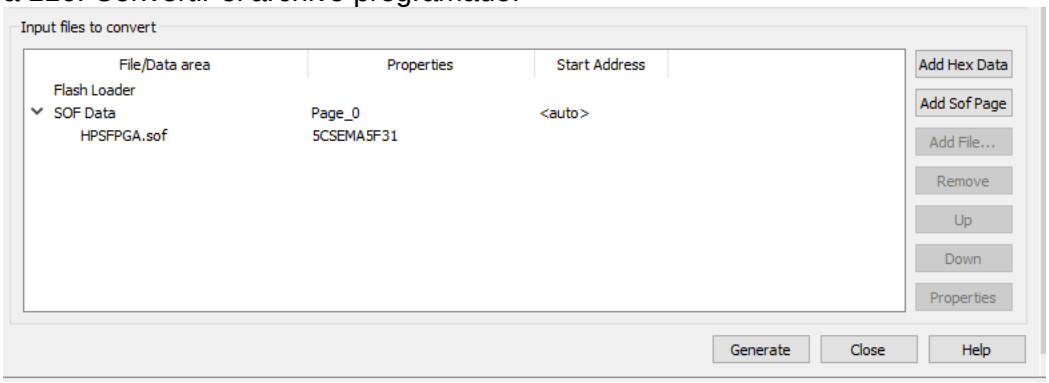
Figura 115. Ventana Convert Programming File.




Fuente: DE1-SoC User Manual Agosto 5 del 2015.

- Hacer clic en Agregar archivo.
- Seleccionar .sof que ser convertido a un archivo .jic desde el diálogo Abrir archivo.
- Hacer clic en Abrir.
- Hacer clic en el cargador en Flash y haga clic en Agregar dispositivo, como se muestra en la Figura *116*.

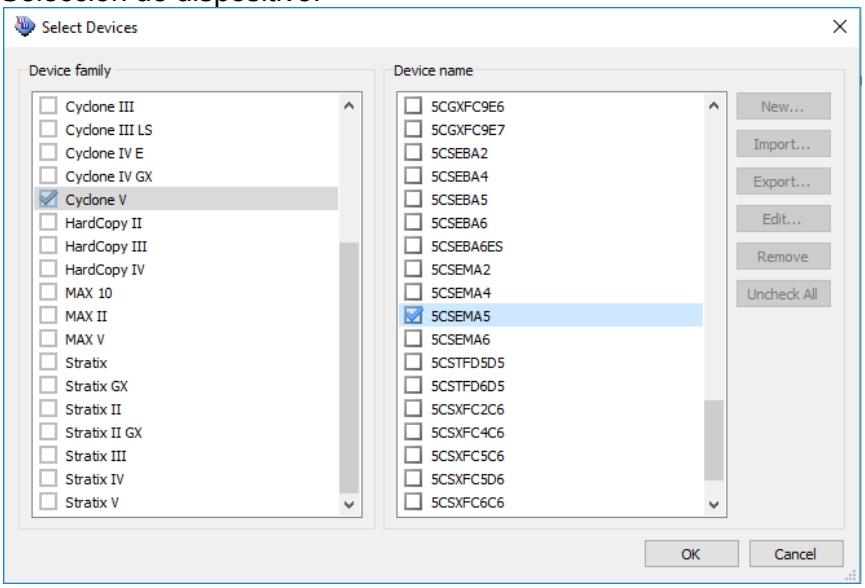
Figura 116. Convertir el archivo programado.




Fuente: DE1-SoC User Manual Agosto 5 del 2015.

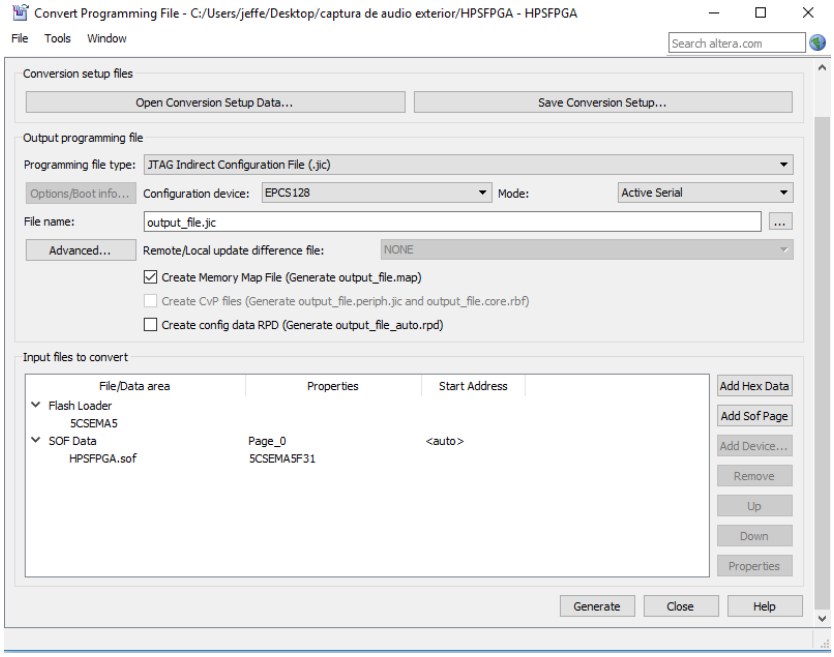
- Hacer clic en OK y aparecerá la página seleccionar dispositivos.
- Seleccionar la FPGA dirigido a ser programado en el EPCS, como se muestra en la Figura *117*.
- Hacer clic en OK y el Convert Programación página Archivos aparecerá, como se muestra en la Figura *118*.
- Hacer clic en Generar.

Figura 117. Selección de dispositivo.




Fuente: DE1-SoC User Manual Agosto 5 del 2015.

Figura 118. Ventana Convert Programming Files.




Fuente: DE1-SoC User Manual Agosto 5 del 2015.

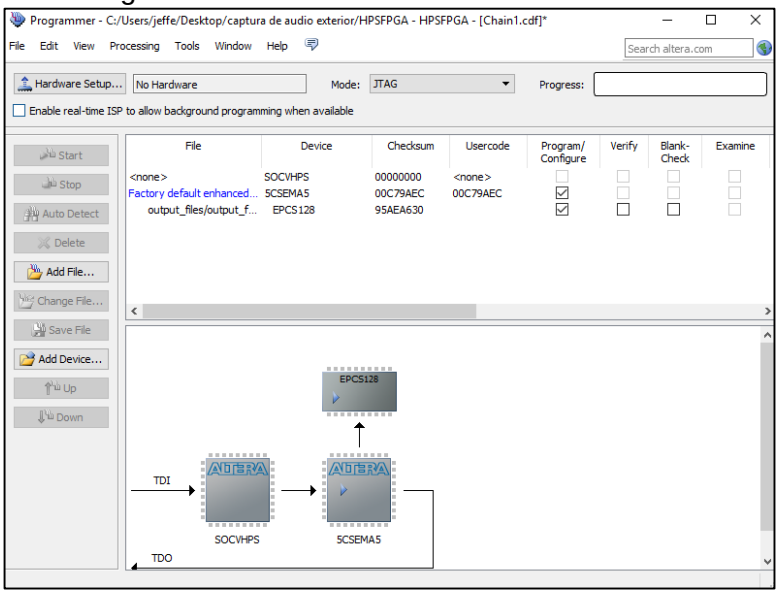
Cuando la conversión de archivos a SOF- JIC es completada, se siguen los siguientes pasos para programar el dispositivo EPCS con el archivo .jic creada en Quartus II programador.

- Elegir Programador en el menú Herramientas y la ventana Chain.cdf que aparece.
- Hacer clic en Detectar automáticamente y luego seleccione el dispositivo correcto. Tanto dispositivo FPGA y HPS deben detectarse.
- Seleccionar nueva programación del archivo. Seleccione el archivo .jic a programar.
- Programar el dispositivo EPCS haciendo clic en el cuadro correspondiente Programa / Configurar. Una imagen SFL por defecto de fábrica se carga, como se muestra en la Figura *119*.
- Hacer clic en Inicio para programar el dispositivo EPCS.

Una vez terminado esta programación de la FPGA tenemos la ventaja que podemos apagar y encenderla sin temor a que el programa se pierda obligando a volver a montarlo en la FPGA.

El Software construido en lenguaje C debe ser ejecutado como una tarea del sistema operativo sin tener que llamarlo y además esté en funcionamiento en todo momento, unos de los método más conocidos y utilizados es la creación de un Daemon que en definición es un proceso programado que se ejecuta de manera persistente y normal mente se inician con el arranque del sistema operativo.

Figura 119. Ventana Programmer.



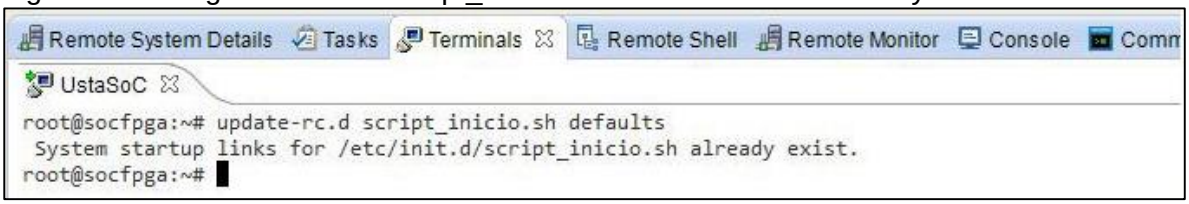

Fuente: DE1-SoC User Manual Agosto 5 del 2015.

El Daemon común mente es creado mediante un script que es cargado en la memoria del sistema operativo. Para este caso el daemon tiene asociado un shell script situado en la carpeta /etc/init.d/ que nos permite iniciarlo.

El script_inicio.sh contiene las asignaciones de las variables necesarias para el software de reconocimiento de voz, además indica la ejecución del el archivo mencionado con sus rutas de ubicación.

La forma de cargar de script editado se hace escribiendo en el terminar de comunicación que puede ser Putty o una de las opciones del software Eclipse para DS-5 llamado Remote System Explorer como se muestra en la Figura *120*.

Figura 120. Carga del archivo script_inicio.sh en el terminal Remote System.




Fuente: elaboración propia en Eclipse ARM DS-5 Development Studio.

Luego se debe generar los permisos de escritura y lectura para este Daemon donde se debe escribir estos comandos en la ventana de interfaz con la computadora y el HPS (Putty). Utilizando el comando: chmod 777 -R /home/root/...proyecto/

El comando chmod genera el permiso dependiendo del número que tenga, cada número tiene su significado: Lectura = 4, Escritura = 2, Ejecución = 3.

Para ofrecer una combinación de estos permisos al usuario, el conjunto de la suma los números relacionados para el caso de leer, escribir y ejecutar es 7. El número en la primera posición es para el permiso del propietario, el segundo es para el permiso de grupo y el tercero se asigna al usuario público.

Por lo tanto, el código 777 significa que los tres usuarios (propietario, grupo, y las autoridades públicas) pueden llevar a cabo las tres operaciones para determinado archivo, es decir, lectura, escritura y ejecución.

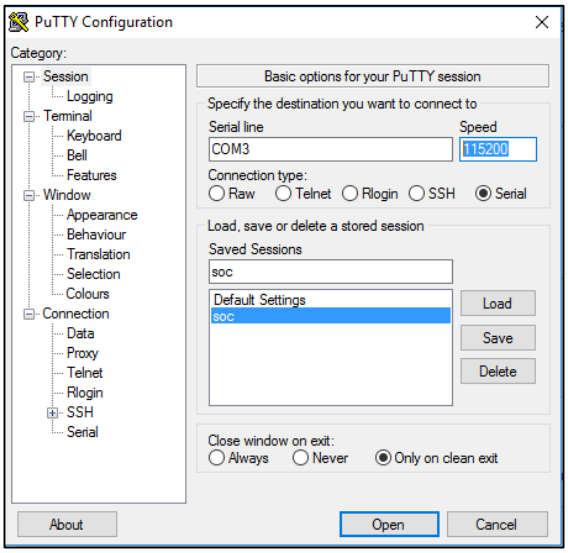
Finalmente se obtiene la construcción de una especie de computadora se puede reiniciar o apagar sin temor a perder el .sof y el .c del sistema.

## 13.3 COMUNICACIÓN SSH: PUTTY

Putty software definido para trabajar como un cliente SSH (Secure SHell), Telnet (Telecommunication Network), Rlogin (Remote Login), Raw y Serial que son protocolos de red y comunicación de datos secuenciales permitiendo el enlace de una maquina a otra de forma remota mediante comandos. El ejemplo más claro es cuando empleamos PuTTY para ejecutar comandos en un servidor VPS (servidor virtual privado) y así poder instalar algún programa o configurar alguna parte del servidor. Con PuTTY conseguimos abrir una sesión de línea de comandos en el servidor remoto para administrarlo.

Para la configuración del Putty se selecciona la línea Serial a una velocidad de 115200 baudios, con un tipo de conexión serial. Además se puede guardar la configuración como se un encuentra en la Figura *121*.

Figura 121. Ventana inicialización PuTTY.




Fuente: elaboración propia en PuTTY.

# 14 CONCLUSIONES

En este proyecto se realizó la revisión bibliográfica de los diferentes métodos y procesos que permiten la construcción un motor de reconocimiento e identificación de voz. Partiendo desde la adquisición de datos luego pasando por el preprocesamiento y la extracción de características hasta el proceso de identificación de modelos acústicos y del lenguaje mediante HMM. De los métodos de extracción de características que se encuentran en la literatura, la implementación de los MFCC garantiza el mejor rendimiento para la formación de los vectores de observación, pues contiene la mayor cantidad de informacion posible para el mensaje lingüístico del locutor.

Se logró programar un algoritmo en lenguaje C encargado de reconocer las palabras detectadas por el modelo HMM para la creación de las rutinas de desplazamiento, mediante el envío de instrucciones al módulo controlador de los motores. Este algoritmo garantiza el requerimiento de superación de un umbral de intensidad sonora para desencadenar el procedimiento del sistema de reconocimiento de voz. A su vez, analiza los resultados del mensaje lingüístico para determinar si la acción a realizar es favorable, o por el contrario carece de sentido.

El motor de reconocimiento de voz SPHINX fue implementado satisfactoriamente en un sistema embebido HPS diseñado en la FPGA Cyclone V de altera, para su correcto funcionamiento con tiempos de procesamiento bajos. Varios módulos diseñados en hardware permiten la aceleración del proceso de detección mediante la distribución inteligente de cargas en procesamiento paralelo.

Se utilizó el sistema de motores eléctricos acoplados a la silla y los sensores SRF02 fueron implementados además mejorando su apariencia y optimizando funcionamiento mediante un driver diseñado desde cero para la aceleración del bus i2C en hardware.

Se realizaron pruebas al sistema de reconocimiento de voz con el fin de observar el rendimiento en la identificación de las palabras y frases que se utilizaran en el manejo de la silla. Estos resultados arrojan un WER de 7.1% en el caso de un locutor masculino, contra un 18.6% para un locutor femenino. En términos generales, el sistema de reconocimiento diseñado e implementado obtuvo un WER de 12.9%, lo cual demuestra que es un gran comienzo para el desarrollo de esta nueva tecnología de transcripción de mensajes lingüísticos mediante un sistema ASR. Aun así, lo ideal es lograr un WER del 10% o menor, por lo que el estudio y diseño de esta tecnología sigue estando en fase de desarrollo.

# 15 RECOMEDACIONES Y TRABAJOS FUTUROS

Para trabajos futuros con el proyecto realizado se propone implementaciones y ajustes que permitirían un funcionamiento cómodo, innovador y seguro para la silla de ruedas, a continuación se presentan las siguientes recomendaciones:

- Confirmación de tareas realizadas: Utilizar mecanismos visuales como pantallas con interfaces agradables o mecanismos auditivos como respuestas del sistema por voz pregrabadas, con el fin de permitir una comunicación más entendible entre la máquina y el usuario.
- Sensores específicos: Mediante la implementación de sensores de vacio dedicados a situaciones como escalones o desniveles del terreno, además de sensores giroscopios que ayuden medir la orientación de la silla en el espacio mejorando la movilidad en los distintos terrenos.
- Estructura de la silla: diseñar o cambiar el chasis de la silla para darle una apariencia fisca más agradable, además mejorar el sistema de las ruedas y los motores.
- Controlar los motores: Diseñar un controlador PID que garantice los recorridos interpretados por el usuario.

Tratamiento de imágenes: diseñar por medio de una cámara la opción de evitar colisiones mediante de visualizaciones del área que selecciona los posibles choques durante la trayectoria.

# 16 BIBLIOGRAFÍA


ALTERA CORPORATION, FPGA, SoC and CPLD from Altera. Nios® II Boot from EPCQ or EPCS in Quartus® II 13.1 [en línea]. <https://www.altera.com/support/support-resources/knowledge-base/solutions/rd11192013_118.html>

JAIN, Aakash. TEJA, Krishna and PALIWAL, Nitish. Real time speech recognition engine [en línea]. < http://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/f2010/np276_ksp55_aj355/np276_ksp55_aj355/>

EMIL MIKULIC. Windowing and FFTs [en línea]. < http://dmr.ath.cx/sound/window/> [citado en 4 de septiembre de 2016]

JORDAN Crittenden and PARKER Evans. SPEAKER RECOGNITION [en línea]. < https://courses.cit.cornell.edu/ece576/FinalProjects/f2008/pae26_jsc59/pae26_jsc59>

MARTINEZ, Fernando, et al. Reconocimiento de voz, apuntes de cátedra para Introducción a la Inteligencia Artificial [en línea]. <http://www.secyt.frba.utn.edu.ar/gia/IA1_IntroReconocimientoVoz.pdf> [citado en 4 de septiembre de 2016]

University of Maryland, Baltimore County . Lecture 18, FFT Fast Fourier Transform [en línea]. <http://userpages.umbc.edu/~squire/cs455_l18.html>[citado en 4 de septiembre de 2016]

DAVIS, S. MERMELSTEIN, P. (1980) Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. In IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 28 No. 4.

X. Huang, A. Acero, and H. Hon. Spoken language processing: a guide to theory, algorithm, and system development. Prentice hall, 2001.

VALDEZ, Jonathan y BECKER, Jared. Understanding the $I^2C$ Bus. En: Texas Instruments Application Report. Junio, 2015, SLVA704

ALCUBIERRE, *et al.* Silla de ruedas inteligente controlada por voz. En: Primer congreso internacional de domótica, robótica y teleasistencia para todos. Madrid. 2005.

HIGUERA, Oscar. Espitia, Helberth. MENDEZ, Diego. prototipo de silla de ruedas comandada por voz empleando HMM en un ambiente controlado / Revista Ingeniería, Investigación y Desarrollo. Vol. 16 Nº 1, 2016.
PAREDES, Montoto, *et al*. Silla de ruedas controlada por voz. En: 6to congreso nacional de mecatrónica. Soledad de Draciano Sánchez, México. 2007.

WATTS, et al. From HMMS to dnns: where do the improvements come from?. En:The Centre for Speech Technology Research, University of Edinburgh, United Kingdom, 2016.

ACERO, Alejandro. Acoustical and Environmental Robustness in Automatic Speech Recognition. Pittsburgh, Pennsylvania September 13, 1990. Trabajo de grado de Doctor en Filosofía en Ingeniería Eléctrica. Carnegie Mellon University. Department of Electrical and Computer Engineering.

PICONE, Joseph. Continuous Speech Recognition Using Hidden Markov Models. En: IEEE ASSP MAGAZINE JULY 1990, 0740-7467/90/0700-0026.

RABINER, L. JUANG, B. An Introduction to Hidden Markov Models. En: IEEE ASSP MAGAZINE JANUARY 1986, Stanford University, 0740-7467/86/0100-0004.

SIMPSON, Richard. Smart wheelchairs: A literature review. En:Journal of Rehabilitation Research & Development. Department of Rehabilitation Science and Technology, University of Pittburgh, Pittburgh, PA. Volumen 42, number 5, page 423. July/August 2005.

Some statistical issues in the comparison of speech recognition algorithms. En: Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on. ISSN 1520-6149.

RABINER, L. A tutorial on hidden Markov models and selected applications in speech recognition. En: Proceedings of the IEEE (Volume: 77, Issue: 2, Feb 1989). ISSN: 0018-9219

MOVELLAN, Javier. Visual Speech Recognition with Stochastic Networks. University of California San Diego, University of California San Diego. La Jolla, Ca 92093-0515.

HUANG, X. ARIKI, Y. and JACK, M. Briefly Noted: Hidden Markov Models for Speech Recognition. (Edinburgh Information Technology Series 7, edited by S. Michaelson and M. Steedman), 1990, x + 276 pp. Hardbound, ISBN 0-7486-0162-7.

PICONE, Joseph. Signal modeling techniques in speech recognition. En: Proceedings of the IEEE (Volume: 81, Issue: 9, Sep 1993). ISSN: 0018-9219.

ROCKETBOARDS.ORG, Datamover Design Example [en línea]. <https://rocketboards.org/foswiki/view/Projects/Datamover> [citado en 4 de septiembre de 2016]

PALIWAL, K. SSNDERSON, C. Effect of different sampling rates and feature vector sizes on speech recognition performance. En: TENCON '97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE. ISBN: 0-7803-4365-4

SSNDERSON, C, PALIWAL, K. Effect of different sampling rates and feature vector sizes on speech recognition perfomance. En: IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications. 1997. vol. 1.

SAITO, Shuko y TANAKA, Kazuo. Fundamentals of Speech Signal Processing. Academic Press, 1986. 266 p. ISBN: 012-61-4880-5.

BOURLARD, H, et al. Towards increasing speech recognition error rates. En: Speech Communications. vol. 18. p. 205-231. 1995.

HERMANSKY, B, et al. Perceptually based linear predictive analysis of speech. En: Acoustics, Speech, and Signal Processing. vol. 10. p. 509-512. 1985

O'SHAUGHNESSSY, D. Automatic speech recognition: History, methods and challenges. En: Pattern Recognition. vol. 41, no. 10. p. 2965-2976. 2008.

ZHENG, F, ZHANG, G y SONG, Z. Comparison of Different Implementatios of MFCC. En: Journal of Computer Science & Technology. vol. 16, p. 582-589. 2001.

Gernot A. Fink. Markov Models for Pattern Recognition: From Theory to Applications. Second Edition. Dortmund, Germany : Springer-Verlag London 2008, 2014. ISBN 978-1-4471-6307-7.

YU Dong. DENG Li. Automatic Speech Recognition: A Deep Learning Approach. One Edition. USA : Springer-Verlag London 2015. ISBN 978-1-4471-5778-6.

KOLOSSA, Dorothea. Haeb-Umbach, Reinhold. Robust Speech Recognition of Uncertain or Missing Data: Theory and Applications. Springer-Verlag Berlin Heidelberg 2011. ISBN 978-3-642-21316-8.

# ANEXO

## GUÍA RÁPIDA DE USUARIO

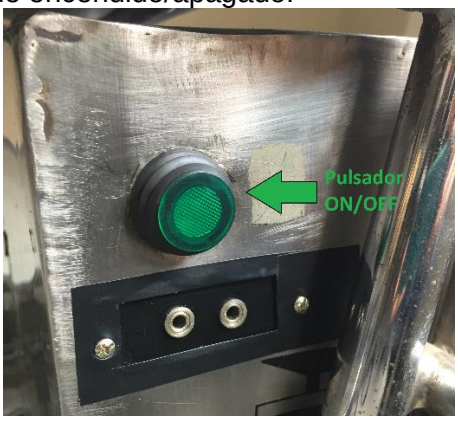
### INTRODUCCIÓN

Este manual le permitirá aprender a utilizar todas las funcionalidades básicas de operación para el manejo de la silla de ruedas. Tener en cuenta que el único requerimiento para la utilización del dispositivo es la voz.

### 1. ON/OFF

El dispositivo se enciende y apaga desde un pulsador ubicado en el lateral izquierdo de la caja de control trasera de la silla de ruedas. Este pulsador indica mediante un dispositivo LED si el sistema se encuentra encendido o apagado. En la Figura Anexo: *1* se observa la ubicación del pulsador de encendido/apagado de la silla de ruedas.

Figura Anexo: 1. Pulsador de encendido/apagado.




En el instante en que el dispositivo es encendido, los motores reaccionan realizando un ligero movimiento en direccion de retroceso para verificar el estado actual de las baterias que alimentan el sistema. En caso de no observar ningun movimiento en las llantas traseras, refierase al capitulo 3 relacionado con la carga de las baterías.

### 2. AURICULAR

Para la comunicación entre el usuario y el sistema, se requiere una diadema convencional con soporte para micrófono y conectores estándar de 3.5mm para audio. Los conectores de audio del sistema se encuentran ubicados en el lateral izquierdo justo debajo del pulsador de encendido/apagado. El conector ubicado a la izquierda corresponde a la entrada de micrófono (rosado), mientras que el conector ubicado a la derecha (verde) permite la salida de audio a través de los auriculares.

En la Figura Anexo: *2* se aprecian los jacks dispuestos para la conexión de los auriculares con micrófono para la comunicación con el sitema de reconocimiento automático de voz.

Figura Anexo: 2. Jacks de 3.5mm para manejo del audio.


El sistema incluye unos auriculares Microsoft LifeCHat LX-1000, los cuales han sido probados en reiteradas ocasiones y para diferentes ambientes, garantizando su correcto funcionamiento en la captación de muestras de audio aptas para el reconocimiento de voz. Sin embargo, el dispositivo soporta el uso de cualquier auricular que utilice el estándar 3.5mm y tenga los canales de micrófono y audio en puertos indepentientes.

## 3. BATERIAS

Las baterías que se encargan de la alimentación del sistema son de tipo VRLA (Ácido-Plomo regulada por válvula). El montaje esta compuesto por dos baterías de 12 V – 12 AH que alimentan los circuitos de potencia y control (ver Figura Anexo: *3*).

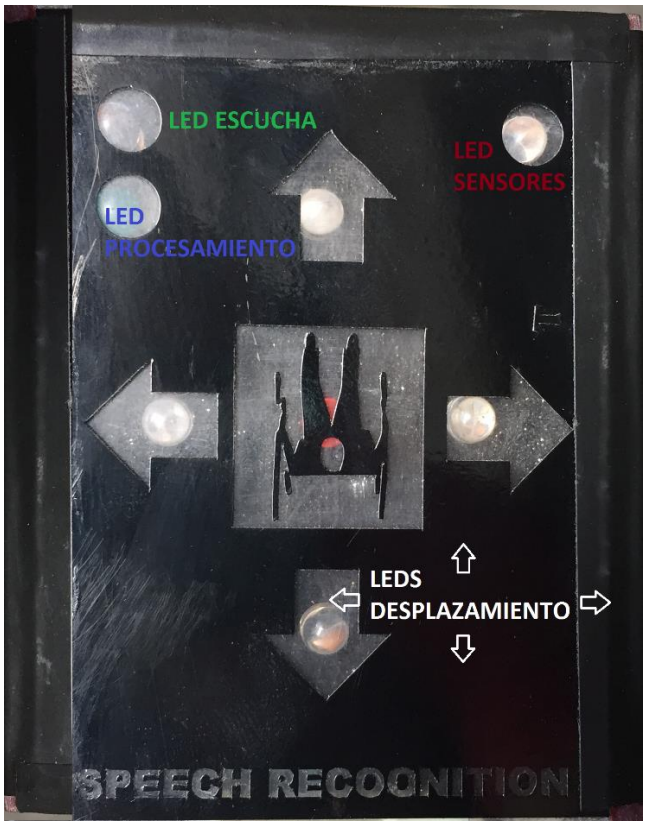
Figura Anexo: 3. Baterias de alimentación del sistema


En el circuito de control, se utiliza un fusible de 10 A, para la protección del controlador RoboClaw. Si se requiere reemplazar alguna de estas baterías, es indispensable que la tensión de salida sea equivalente a 12 V.

Para cargar las baterías, se requiere una alimentación entre 13.5 V – 13.8 V, a una corriente no superior a los 3.6 A. El proceso de carga puede extenderse por varias horas, sin embargo, se sugieren tiempos no inferiores a 4 horas ni superiores a 24 horas.

## 4. PANEL INDICADOR

Figura Anexo: 4. Panel indicador.




Se dispusto este panel (ver Figura Anexo: *4*) para indicarle al usuario sobre la etapa que se lleva a acabo, permitiendo un mejor manejo de la silla. El panel está compuesto por diferentes indicadores que se verán a continuación.

## 5. INDICADOR DE ESCUCHA

Este LED le indica al usuario el tiempo que tiene disponible para almacenar sus muestras de audio en la memoria del sistema. Este tiempo es de tres (3) segundos. Para poder comunicarse con el sistema ASR se debe superar determinado umbral de intensidad. Basta con hablar en un tono moderado para que el sistema comience a almacenar los datos de interés. Una vez transcurra el tiempo disponible para almacenamiento de datos, se pasa a la etapa de procesamiento donde el sistema intenta interpretar el mensaje lingüístico contenido en la muestra.

## 6. INDICADOR DE PROCESAMIENTO

El LED de procesamiento le indica al usuario el tiempo que debe esperar mientras el sistema procesa los datos y predice el mensaje transmitido. Es un tiempo muerto de aproximadamente 1.2 segundos.

## 7. PALABRA CLAVE

Para las ejecuciones de moviento de la silla se requiere que el usuario pronuncie la palabra "**JOTA**" sin importa en que parte de la oración se encuentre. Mediante esta palabra clave la silla se desplazara hacia su dirección deseada.

## 8. RUTINAS

El moviento de la silla esta determinado en cuatro direcciones y 13 rutinas. Estas direcciones se necuentra ubicadas en el panel como se puede observar en la *Figura Anexo: 4*.

En el panel se visualiza la dirección de desplazamiento descrita por el usuario determinadas por adelante, atrás, derecha e izquierda.

Las 13 rutinas consisten en:

Dirección de "adelante", "avance" ó "avanzar"

- "Jota adelante" : desplazamiento aproximado de 10m.
- "Jota adelante uno" : desplazamiento aproximado de 1m.
- "Jota adelante dos" : desplazamiento aproximado de 2m.

Dirección de "retroceder" ó "retroceda"

- "Jota retroceda" : desplazamiento aproximado de 50cm.
- "Jota retroceda uno" : desplazamiento aproximado de 1m.
- "Jota retorceda dos" : desplazamiento aproximado de 2m.

Dirección de "derecha"

- "Jota derecha" : giro aproximado de 90 grados.
- "Jota derecha uno" : giro aproximado de 45 grados.
- "Jota derecha dos" : giro aproximado de 135 grados.

Dirección de “izquierda”

- “Jota izquierda” : giro aproximado de 90 grados.
- “Jota izquerda uno” : giro aproximado de 45 grados.
- “Jota izquierda dos” : giro aproximado de 135 grados.

Dirección extra “giro 180”

- “Jota giro 180” : giro aproximado de 180 grados

## 9. ¿COMANDO ES CORRECTO?

El usuario debe indicarle a la silla solo un comando de dirección para su correcto funcionamiento. Por ejemplo:

- “Jota por favor ir hacia la derecha” comando correcto.
- “Jota por favor ir hacia la derecha iquierda” comando incorrecto.
- “Jota adelante” comando correcto.
- “Jota adelante retroceda” comando incorrecto.

Si el comando es correcto el panel de visualización indicara la dirección y la silla ejecutara la acción de moviento deseada.

## 10. INDICADOR DE SENSORES

El LED de sensores indica al usuario que el desplazamiento ha sido interumpido por su seguridad, mediante el aviso de los sensores. Cuando una acción de movimiento de la silla es ordenada por el usuario puede ser anulada por sensores que indica un obstáculo en la dirección deseada.