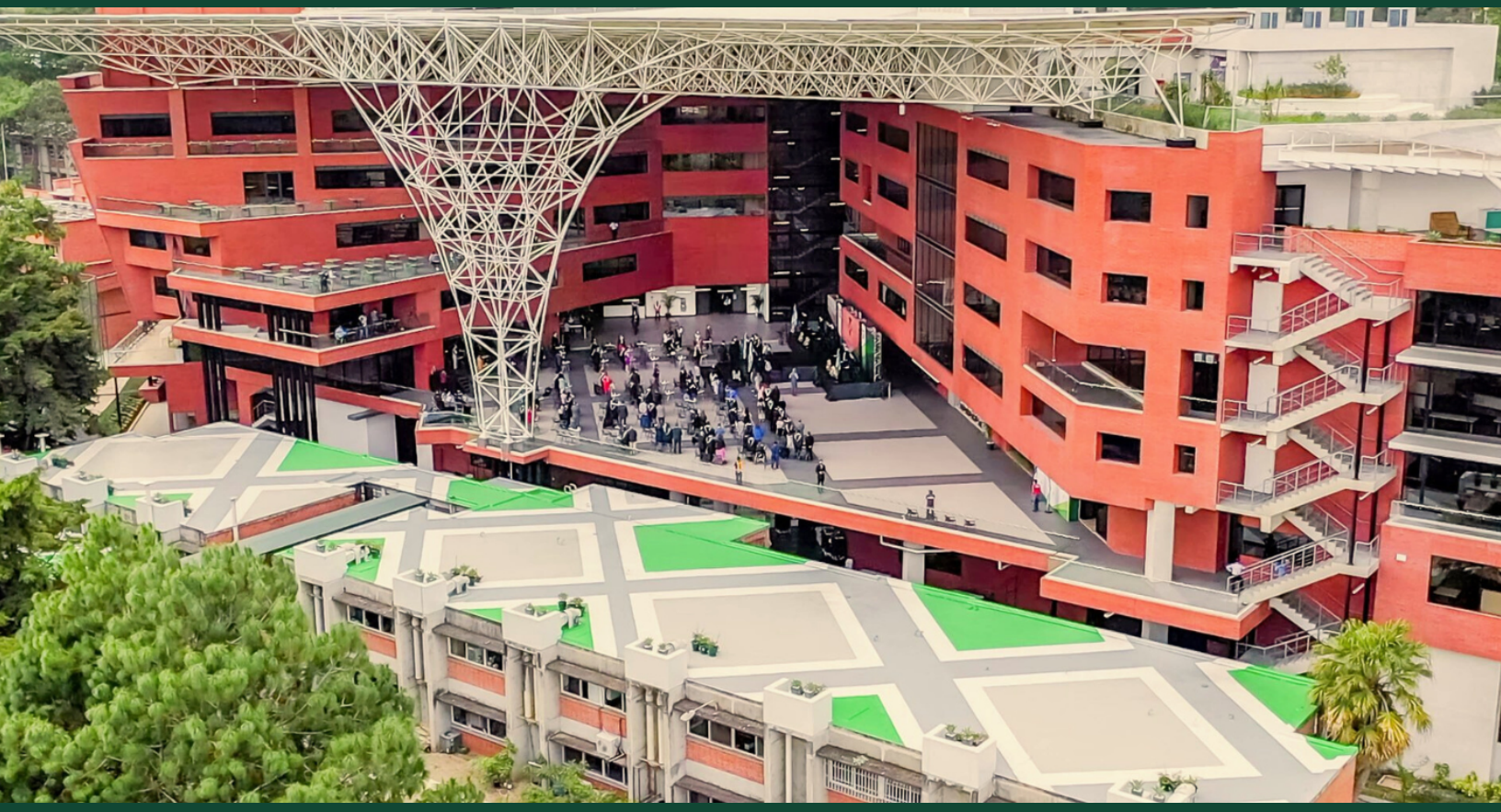
Diseño e implementación de un módulo de reconocimiento de voz para el control de sistemas robóticos

Oscar Fernando López Godínez



UVG | UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

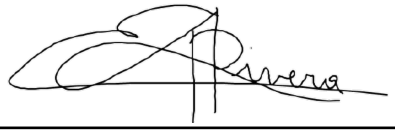
Diseño e implementación de un módulo de reconocimiento de voz para el control de sistemas robóticos

Trabajo de graduación presentado por Oscar Fernando López Godínez para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2026

Vo.Bo.:



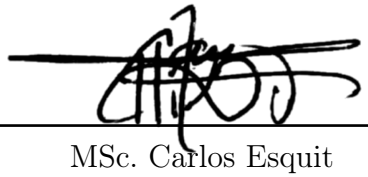
(f) _____

PhD. Luis Alberto Rivera

Tribunal Examinador:

(f) _____

PhD. Luis Alberto Rivera



(f) _____

MSc. Carlos Esquit

(f) _____

MSc. Miguel Zea

Fecha de aprobación: Guatemala, 17 de Junio de 2026.

Prefacio

Agradezco a Dios, quien me ha brindado la vida y la fortaleza necesarias durante este proceso. Sin Él, nada en mi vida hubiera sido posible. Doy gracias a mis padres, por su apoyo y amor incondicional, y por brindarme su tiempo y cariño en los momentos difíciles. Agradezco a Santiago Sánchez, ingeniero que estuvo presente durante el proceso de investigación y me brindó su compañía y conocimiento técnico para el desarrollo gráfico del proyecto; a Karla López, ingeniera que me brindó su apoyo emocional para superar los estancamientos creativos; a Luz Galindo, ingeniera que me brindó apoyo emocional y creativo para encontrar nuevas ideas en la investigación; y a Rony Carranza, quien fue parte fundamental en las decisiones creativas para conseguir el mejor resultado posible. Agradezco a mi asesor, Luis Rivera, por su constante guía y retroalimentación a lo largo del trabajo.

Por último, me gustaría agradecer a cada compañero, amigo y persona que hizo posible esta tesis, cuya culminación no habría sido posible sin su compañía y colaboración.

Índice general

Índice de figuras

Índice de cuadros

Resumen

Este trabajo presenta el diseño, la implementación y la validación de un módulo de reconocimiento de voz en español para el control de sistemas robóticos. El objetivo principal es reducir la carga operativa relacionada al uso de las manos. El sistema se estructura de cuatro componentes. El primero transcribe en tiempo real la voz del usuario a texto legible mediante un servicio *speech-to-text* en modalidad de *streaming*. El segundo, un ánalizador sintáctico-semántico, toma las expresiones transcritas y las traduce a intenciones entendibles para el sistema robótico. El tercero es un controlador capaz de ejecutar órdenes sobre el robot que considere los límites articulares y los comandos con instrucciones preestablecidas. El cuarto es una interfaz gráfica que permite visualizar las transcripciones, las acciones interpretadas, el estado y los ángulos del robot.

Para la validación experimental de este proyecto, se utilizó un myCobot280 M5 y un micrófono JBL Quantum, con 8 sujetos de prueba. El sistema alcanzó un desempeño de comandos interpretados y ejecutados de 96.25 % en modo absoluto y 95 % en modo relativo, lo que demuestra su viabilidad operativa.

Palabras clave: reconocimiento de voz, conversión de voz a texto, análisis sintáctico-semántico, interacción humano-robot, control robótico, interfaz gráfica de usuario, MyCobot280.

Abstract

This work presents the design, implementation, and validation of a Spanish voice recognition module for the control of robotic systems. The main objective is to reduce the operational burden associated with the use of hands when manipulating robotic systems. The system is structured into four components. The first component transcribes the user's speech into readable text in real time through a streaming speech-to-text service. The second component is a syntacticsemantic analyzer that processes the transcribed expressions and translates them into system-understandable robotic commands. The third component is a controller capable of executing robot commands while enforcing joint limits and predefined instruction constraints. The fourth component is a graphical user interface that displays live transcriptions, interpreted actions, system status, and robot joint angles.

For experimental validation, a myCobot280 M5 manipulator and a JBL Quantum microphone were used, involving eight test subjects. The system achieved a correctly interpreted and executed command rate of 96.25 % in absolute mode and 95 % in relative mode, demonstrating operational feasibility.

Keywords: voice recognition, speech-to-text, syntactic-semantic analysis, human-robot interaction, robotic control, graphical user interface, myCobot 280.

CAPÍTULO 1

Introducción

La robótica se ha consolidado como una solución en la automatización de procesos industriales, la investigación académica, la asistencia médica y los servicios de consumo diario. No obstante, el máximo potencial de un sistema robótico reside no solo en su arquitectura mecánica o sofisticación cinemática, sino en la calidad de la interacción con el usuario. Las interfaces comúnmente utilizadas, como paneles HMI (*human-machine interface*), controles físicos o consolas virtuales, imponen una carga operativa y cognitiva que, en ciertas situaciones, limita la fluidez del trabajo, lo que exige atención visual constante y entorpece la coordinación simultánea de tareas. Ante tal problemática, el reconocimiento de voz surge como un canal de comunicación hombre a máquina que permite expresar instrucciones en lenguaje natural, lo que libera la atención física y cognitiva del operador.

Este trabajo aborda el diseño y la implementación de un módulo de reconocimiento de voz orientado al control de sistemas robóticos. El sistema convierte en tiempo real enunciados verbales en instrucciones de movimiento ejecutables por el robot, priorizando la intención, la trazabilidad técnica de cada transcripción y la preservación de la integridad del entorno de operación.

Para la reducción de la atención física y cognitiva de los usuarios en la manipulación de un sistema robótico, se adoptó una arquitectura compuesta por tres subsistemas definidos y acoplados. El primero consistió en la adquisición y la transcripción del habla en español con soporte para un flujo continuo. El segundo consistió en la interpretación sintáctico-semántica de comandos, con manejo explícito de equivalencias léxicas, signos y modalidades. El tercero abordó la generación de órdenes a nivel digital compatibles con las restricciones cinemáticas y los límites de seguridad del sistema robótico.

A nivel metodológico, el desarrollo se organizó en cuatro etapas. En la primera, se estudiaron los requisitos y la selección tecnológica, con alternativas de transcripción en español. La segunda etapa sirvió para implementar el flujo de trabajo de procesamiento de voz, que incluyó la captura, la limpieza y la transcripción continua de audio con transmisión en vivo. Luego, en la tercera, se implementó el núcleo lingüístico mediante un analizador que combinó reglas, expresiones regulares y un diccionario de equivalencias para mapear en

forma computacional un comando que el sistema robótico fuera capaz de interpretar. Por último, en la cuarta etapa, se integró el módulo de control y la interfaz gráfica, encapsulando las órdenes en un protocolo de comunicación con el robot. El funcionamiento completo se validó en sesiones controladas con escenarios de prueba que incluyeron variaciones en las condiciones acústicas del entorno y en las características vocales de los usuarios.

CAPÍTULO 2

Antecedentes

Los sistemas de reconocimiento de voz son tecnologías en constante desarrollo, que forman parte de una tendencia hacia el desarrollo de interfaces naturales, las cuales tienen el propósito de mejorar la interacción entre humanos y máquinas. Aunque tradicionalmente se ha recurrido a métodos visuales o basados en imágenes, la interacción por voz ha ganado importancia debido a su capacidad para ofrecer interacciones rápidas y flexibles. Sus aplicaciones empiezan desde asistentes virtuales, asistencia en cirugías y se pueden ver en entornos de automatización industrial. Los avances buscan incrementar la precisión en la conversión de voz a texto y asimismo, se busca implementar sistemas que se adapten a distintos hablantes sin importar idioma o pronunciación.

Anteriormente, en la Universidad del Valle de Guatemala se inició una búsqueda por encontrar una interfaz natural que permita el control de sistemas robóticos. De esta forma surge la primera línea de investigación, la cual tuvo como objetivo la interpretación de información visual del sistemaăBrainlab en movimientos robóticos. [1]

En el proyecto desarrollado [1] se logró implementar una herramienta de procesamiento digital de imágenes y reconocimiento óptico de caracteres donde se consiguió la extracción automática de datos visuales necesarios para configurar la posición de un brazo robótico físico. Sin embargo, el trabajo presentó ciertas limitaciones, especialmente en la adaptabilidad del sistema ante variaciones en la calidad visual, como condiciones de iluminación o ángulos no óptimos de captura de pantalla. En consecuencia, se destaca la importancia de desarrollar métodos alternativos al procesamiento visual, capaces de reducir las restricciones operativas.

Por consiguiente, la presente investigación abre una segunda línea de trabajo centrada en el reconocimiento de voz.

2.1. Diseño de un sistema de reconocimiento de voz para un brazo robótico para cirugía laparoscópica

En el trabajo realizado por Ricardo Pastor [2] sobre un sistema de reconocimiento de voz para un brazo robótico para asistencia en cirugías, el autor trabajó en un hardware embebido el cual consiste en una RaspberryăPi, una tarjeta de sonido USB Manhattan, un micrófono, una microSD y un botón físico. Asimismo, Python fue el lenguaje de programación principal ya que posee una gran variedad de librerías oportunas para el manejo de audio.

El principal objetivo de este trabajo era crear un sistema que reconociera comandos de voz y poder transcribirlos internamente e interpretarlos como acciones que se deben ejecutar por parte del robot asistente. Por lo tanto, se propuso un módulo de 9 pasos que pueda pasar de un estímulo auditivo a una acción física.

La primera etapa corresponde a la eliminación de silencio y ruido, cuyo objetivo es detectar la porción activa de la señal de voz y descartar los fragmentos que contienen ruido de fondo o silencio, lo cual reduce el procesamiento innecesario y mejora la precisión del reconocimiento. Posteriormente, se aplica una técnica conocida como preénfasis, que consiste en reforzar las frecuencias altas de la señal, ya que estas contienen información fonética crítica que podría perderse si se tratara la señal de forma plana.

En la tercera etapa, una vez la señal esté preprocesada, es dividida en pequeñas ventanas; en cada una de estas ventanas se aplica la ventana de Hamming, una función que suaviza los bordes y evita discontinuidades al aplicar la transformada de Fourier. Luego, se utiliza la Transformada de Fourier de Tiempo Reducido (STFT), la cual permite observar cómo varían las frecuencias a lo largo del tiempo, generando un espectrograma de la señal.

Posteriormente, en la penúltima etapa se extraen los coeficientes cepstrales en la frecuencia de Mel (MFCC). Esta técnica convierte la información espectral en un conjunto compacto de coeficientes que representan cada segmento del habla. Finalmente, la última etapa para comparar los comandos hablados con las muestras almacenadas en el sistema se aplica el método de alineamiento temporal dinámico (DTW), el cual permite medir la similitud entre secuencias que pueden diferir en duración o velocidad de pronunciación. [2]

2.2. Aprendizaje por refuerzo de un *parser* semántico óptimo en DRT

El trabajo realizado por Jessenia Piza [3] tuvo un énfasis en el aprendizaje por refuerzo de un *parser* semántico. Su trabajo pretendía sustituir los *parsers* tradicionales, ya que estos requieren de reglas manuales complejas, difíciles de mantener y poco escalables. Por lo que decidió incluir el aprendizaje por refuerzo profundo para que un agente aprenda a generar estas estructuras semánticas de forma autónoma, partiendo de recompensas por cada estructura lógica acertada.

La solución desarrollada por Piza empezaba desde un entorno simulado donde un agente recibe una oración simple y debe ser capaz de convertirla en una estructura DRS (*Deep*

Reinforcement Learning). El entorno permite realizar acciones como moverse a lo largo de las palabras y darles una interpretación como sujeto, verbo o sustantivo, y también permite agregar referentes y condiciones a la oración. Es así como se iba recompensando al agente únicamente si respondía una pregunta correctamente donde se demostrara la comprensión lógica de la oración.

Para lograr esto se usó una red Deep Q-Network en donde la entrada es un vector que representa el estado del agente, es decir, las palabras procesadas y la estructura DRS actual, y su salida es la selección entre una de las cinco acciones disponibles.

Dentro del entrenamiento para el modelo, la autora notó mucha inestabilidad por lo que decidió ajustar dos hiperparámetros, los cuales eran Gamma (γ) y Epsilon (ϵ): Gamma decide cuánta importancia se otorga a las recompensas futuras, y Epsilon es responsable de equilibrar la elección entre acciones conocidas y acciones nuevas. Estos ajustes se realizaron con el propósito de obtener un mejor desempeño, lo cual fue efectivo y lo demostró en una tabla comparando los rendimientos. [3]

CAPÍTULO 3

Justificación

La integración de la robótica en diversos sectores ha sido una herramienta fundamental para reducir la carga operativa y facilitar tareas que antes eran complejas o incluso inviables para el ser humano. Sin embargo, uno de los principales desafíos en la interacción del humano al robot es el control intuitivo y seguro que permitan una comunicación eficiente sin generar distracciones durante la operación del sistema.

En ese sentido, los sistemas de reconocimiento de voz se presentan como una solución viable y altamente prometedora, ya que permite al usuario emitir comandos de forma verbal sin desviar su concentración y sin intervenir físicamente. Por lo tanto, es capaz de llevar a cabo cambios de posición, de orientación o de fuerza que le sean de utilidad para la aplicación pertinente.

En este trabajo se desarrolló un sistema que interpreta comandos de voz emitidos por el usuario para controlar una articulación robótica. Por lo tanto, se desarrolló una solución universal que puede adaptarse al control de sistemas robóticos de distintas áreas, tales como entornos médicos, industriales, académicos o de asistencia personal, contribuyendo así a mejorar la interacción entre humanos y robots.

Anteriormente, en la primera línea de investigación la cual se centra en el procesamiento de imágenes y reconocimiento óptico de caracteres, se determinaron ciertas limitaciones y desafíos que se buscan remover al implementar el reconocimiento de comandos por voz. Dentro de las limitaciones más significativas se encuentra la incapacidad que tiene el modelo de hacer la interpretación de una imagen si esta no está centrada de la forma correcta, ángulo correcto e inclusive si no posee el brillo adecuado. Estos aspectos hacen que haya una gran dependencia de la configuración de la pantalla y no es capaz de hacer ninguna interpretación si alguno de los aspectos mencionados no estén en el punto deseado.

Además, también existen limitaciones con el módulo Tesseract el cual extrae el texto de las imágenes ya que depende mucho de la calidad de la imagen, ya que si hay mucho ruido visual entonces su precisión baja considerablemente.

De igual manera, se reconocieron limitaciones para esta segunda línea de investigación ya que aspectos como calidad del micrófono y cantidad de ruido ambiental afectan la interpretación de comandos. También se debe considerar que la pronunciación de palabras es única, al igual que la velocidad de habla, lo que se convirtió en un gran reto para la construcción de un modelo que sepa adaptarse a las características del habla de cada usuario.

Por lo tanto, se desarrolló un sistema de reconocimiento de comandos por voz que es capaz de funcionar correctamente ante situaciones críticas donde demuestre su adaptabilidad, precisión y seguridad.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Diseñar y desarrollar un sistema de reconocimiento de voz capaz de interpretar comandos para el control de sistemas robóticos.

4.2. Objetivos específicos

- Investigar, evaluar y seleccionar las bibliotecas de software más adecuadas para la transcripción de voz en tiempo real.
- Desarrollar un analizador sintáctico y un analizador semántico para la interpretación de comandos de voz.
- Validar la efectividad del sistema de interpretación de comandos de voz mediante pruebas con sistemas robóticos disponibles en el Departamento de Ingeniería Electrónica y Mecatrónica de la Universidad del Valle de Guatemala.
- Implementar un módulo de activación por voz que detecte un comando o frase especial para iniciar la comunicación con el sistema robótico y habilitar la recepción de instrucciones.
- Diseñar una interfaz gráfica intuitiva que permita visualizar en tiempo real la transcripción y el análisis de los comandos de voz, así como el estado de ejecución del sistema robótico.

CAPÍTULO 5

Definición del problema

La interacción humano a máquina mediante dispositivos convencionales como teclados, ratones, mandos de control, interfaces genéricas o HMIs introduce gran carga cognitiva y riesgos de uso indebido en tareas que requieren órdenes breves y repetitivas a nivel articular. En la Universidad del Valle de Guatemala no se dispone de un módulo en español, integrado y validado, que convierta comandos de voz en acciones seguras sobre sistemas robóticos en tiempo real, con tolerancia a acentos, ruido ambiente y variaciones en la formulación del comando.

Se requiere un módulo de software que a partir de audio, transcriba con baja latencia, luego interprete la intención en el dominio del control articular y posteriormente aplique la programación defensiva donde se validan los rangos y estados, límites articulares y bloqueos ante ambigüedades de la instrucción para finalmente entregar retroalimentación al usuario en forma de transcripción, comando interpretado y estado de ejecución del manipulador utilizado.

Para que el módulo de transcripción por voz se considere funcional, debe de ser capaz de operar en modo *streaming* y debe activarse únicamente cuando detecta un comando hacia el sistema robótico. El módulo también debe ser capaz de analizar el texto obtenido por la transcripción y extraer parámetros como junta, ángulo y sentido de rotación. Con los parámetros obtenidos del análisis el módulo debe ser capaz de enviar órdenes al sistema robótico, incorporando la confirmación de comandos. Además, debe mostrar en tiempo real la transcripción, la interpretación y el estado del robot mediante una interfaz gráfica.

Dentro del alcance para el proyecto se define la integración de un motor de transcripción en español en tiempo real, también un analizador semántico basado en expresiones regulares. Igualmente, se contempla la incorporación de una capa de seguridad con validación de límites angulares para prevenir colisiones y una interfaz gráfica para mantener trazabilidad del comportamiento del módulo.

Fuera de alcance consideramos el entrenamiento de modelos ASR (*Automatic Speech Recognition*). Igualmente, no se contempla que el módulo de transcripción por voz comprenda

el lenguaje natural de forma libre, asimismo no se contempla la planificación de trayectorias de movimiento para el sistemas robótico. Con ello, el trabajo queda acotado a diseñar e implementar un módulo de reconocimiento e interpretación de comandos de voz en español, seguro y monitoreable, orientado al control articular y validado mediante métricas objetivas de precisión y seguridad.

CAPÍTULO 6

Marco teórico

6.1. Reconocimiento automático del habla

El reconocimiento automático del lenguaje natural es una tecnología que permite transformar las señales de voz a texto utilizando diferentes procesos. Estos procesos requieren la captura de audio, su transformación en representaciones digitales y posteriormente, la interpretación del contenido lingüístico. [4]

Esta tecnología ha evolucionado como consecuencia de los grandes avances en el procesamiento digital de señales y también gracias a los avances en el desarrollo de modelos estadísticos y neuronales.

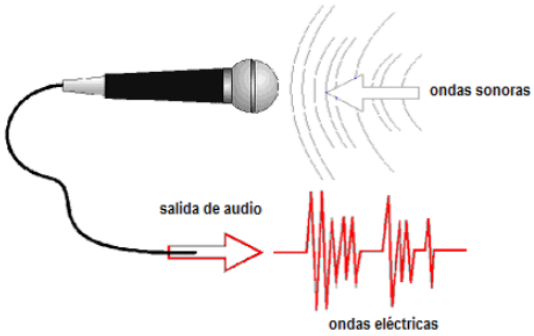
6.2. Captura de audio

La captura de audio constituye la primera etapa de un sistema de reconocimiento automático del habla. Este proceso inicia transformando ondas sonoras, que son variaciones de presión en el aire, en una representación digital manipulable por el ordenador. La calidad de esta captura impacta directamente a toda la serie de procesos siguientes, con pérdida de información o distorsión dentro del audio. Por lo tanto, el tipo de micrófono y las condiciones de captura son aspectos cruciales para garantizar la representación fiel de las señales de voz. [5]

6.2.1. Funcionamiento del micrófono

Un micrófono recibe el nombre de transductor acústico. Este convierte las variaciones de presión del aire generadas por ondas sonoras en señales eléctricas. [6]

Figura 1. Esquema de funcionamiento de un micrófono



Nota. Esta imagen fue obtenida de [7].

Tipos de micrófono y respuesta en frecuencia

En aplicaciones de reconocimiento automático del habla se utilizan dos principios de transducción, los micrófonos dinámicos (bobina móvil) destacan por su robustez y menor susceptibilidad a la sobrecarga, mientras que los de condensador ofrecen mayor sensibilidad, menor masa móvil y una respuesta más uniforme en la banda vocal, cualidades que favorecen la preservación de formantes y transiciones espectrales finas, críticas para el *ASR*. [5]

Además del principio de transducción, el patrón polar determina la direccionalidad de la captación. Existen micrófonos omnidireccionales los cuales captan el sonido de manera uniforme de las direcciones, lo que los hace apropiados en lugares donde el ruido está controlado. [8]

La respuesta en frecuencia describe la variación de sensibilidad del micrófono en función de la frecuencia. Para reconocimiento de voz, es preferible una respuesta plana como sea posible en la banda pertinente, usualmente de 8 a 16 kHz, lo cual reduce coloraciones espectrales que podrían distorsionar los formantes. En conjunto, la elección de transductor y patrón, junto con una respuesta en frecuencia adecuada, condiciona la integridad de las características acústicas que alimentan el modelo de reconocimiento y tasa de error del sistema. [5]

6.2.2. Pre-procesamiento de señales de audio

El pre-procesamiento de señales de audio es una etapa crucial en los sistemas de reconocimiento de voz, ya que se encarga de preparar las señales para que el modelo de reconocimiento las interprete de manera correcta. Este proceso tiene como objetivo mejorar la calidad de la señal, reducir el ruido y hacer que las grabaciones sean más consistentes, sin importar las condiciones de grabación.

La primera etapa consiste en la normalización del nivel, que ajusta el volumen para que las señales de voz tengan un nivel adecuado. Además, se utiliza el pre-énfasis, un

filtro pasa altas el cual resalta las frecuencias que contienen información importante para diferenciar sonidos. Luego se segmenta la señal en pequeños tramos de usualmente 20 a 25 milisegundos para poder analizarlas de manera efectiva, ya que se necesitan trozos de tiempo suficientemente pequeños donde la señal se pueda considerar constante. Estos tramos se procesan con una ventana para reducir los errores que puedan ocurrir en los bordes de cada fragmento. [9]

En ambientes ruidosos se utilizan técnicas como la detección de actividad de voz *VAD*, que permite identificar las partes del audio que realmente contienen voz y las que no. Con esto se reduce el procesamiento innecesario y es posible enfocarse en las partes relevantes de la grabación. También existen técnicas como la sustracción espectral, que estima el ruido de fondo durante largos silencios y lo elimina de los tramos donde existe voz sobrepuesta a dicho ruido. [10]

6.2.3. Extracción de características

Finalizado la etapa de preprocesamiento, el siguiente paso consiste en extraer las características acústicas que representen de forma compacta la información más importante de la señal. El objetivo es reducir la cantidad de datos a procesar, conservando la información relevante para el reconocimiento de voz.

Para ello, se utilizan los denominados coeficientes cepstrales en escala Mel *MFCC*, que simulan la forma en que los oídos humanos perciben las frecuencias del sonido. Estos coeficientes se obtienen a partir de una transformación matemática de la señal, donde se aplican filtros de Mel, una versión adaptada de la escala de frecuencias que toma en cuenta la forma en que percibimos los sonidos de distintas frecuencias.

Una alternativa de los *MFCC* son los bancos de filtros Mel en log-energía, que son útiles en redes neuronales profundas ya que conservan mejor la información detallada de la señal sin perder el contexto. [10]

6.3. Interpretación mediante modelos acústicos y de lenguaje

En un sistema ASR (*automatic speech recognition*) los modelos acústicos constituyen el núcleo del funcionamiento, ya que establecen la relación probabilística entre las unidades de sonido y las unidades lingüísticas que posteriormente se utilizan para los modelos de lenguaje y los analizadores sintáctico-semánticos. El propósito es estimar para cada segmento de audio, la probabilidad de que dicho segmento corresponda a cierta unidad lingüística. [11]

Estos modelos trabajan con un procedimiento fundamental que les permite entender una señal de voz. Primero en cualquier sistema de reconocimiento de voz el paso principal es transformar el sonido en un conjunto de datos que una computadora pueda entender. Dicho proceso se llama extracción de características y este es un traductor intermedio que transforma del mundo físico al digital. [12]

Para lograr que una máquina comprenda un segmento de voz, dicha señal de voz que

es continua y varía en el tiempo, se divide en pequeños fragmentos de aproximadamente 20 a 25 mili segundos, que se superponen entre si, ya que si no se divide entonces estaría tratando de comprender una conversación a partir de sonido continuo. [13]

Posteriormente, a cada fragmento se le aplica una ventana de Hamming que sirve para suavizar los bordes de la señal para evitar futuros errores cuando se hace el análisis matemático. Luego se aplica una transformación llamada transformada rápida de Fourier que permite visualizar sus componentes de frecuencia, las cuales se tienen que escalar debido a que el oído humano no percibe las frecuencias de forma lineal, estas se tienen que convertir a escala Mel y se logra dividiendo la energía del sonido en distintos filtros que simulan cómo el oído agrupa los sonidos. [14]

Finalmente, se calcula el logaritmo de la energía en cada filtro lo que elimina diferencias pequeñas y reduce efectos de variaciones fuertes. Asimismo, se aplica una transformación matemática adicional que sintetiza esa información en una fracción mas pequeña de números conocidos como coeficientes cepstrales en las frecuencias de Mel o MFCCs. Dichos números contienen qué tipo de sonido es, qué tono posee, cómo termina o como empieza. Usualmente, se suelen obtener 12 o 13 de estos coeficientes por fragmento, los cuales posteriormente se le suman las diferencias con los fragmentos anteriores para poder resultar en una matriz numérica que cada fila representa un fragmento de la voz en forma de varios valores los cuales serán interpretadas por un modelo. [15]

6.3.1. Modelos acústicos

Los modelos acústicos son la continuación de un proceso generalizado por el cual se transforman ondas de voz a números. Una vez se obtiene la matriz de números, el siguiente paso es determinar qué sonidos corresponden a esos patrones numéricos. Este trabajo lo realiza el modelo acústico, haciendo la asociación de cada fragmento de la señal con una unidad del lenguaje, como un fonema, una letra o palabra. [16]

Existen varios modelos los cuales han sido utilizados a lo largo de los años, los principales más famosos fueron los llamados HMM-GMM (*hidden markov model, gaussian Mixture Model*), que combinan dos ideas. Primero, los modelos HMM dividen el sonido en tres partes: Inicio, centro y final. Los modelos GMM calculan en cada parte dividida que tan probable es que un fragmento corresponda a ese sonido específico. Esta combinación se utilizó durante muchos años hasta que demostró muchas limitaciones en presencia de variaciones neutrales del habla, como velocidad, entonación o acento. [16]

La siguiente generación de modelos empieza con las redes neuronales, ya que se sustituyó el uso de gaussianas y se empezaron a usar redes neuronales profundas, las cuales son capaces de aprender patrones complejos a partir de una gran enseñanza a base de ejemplos. El funcionamiento de estas redes empieza en recibir varios fragmentos de voz al mismo tiempo y devuelven como salida cuál es el sonido más probable en ese instante. [16]

Consecuentemente a estos avances surgieron arquitecturas basadas en modelos famosos, como el caso de las redes TDNN (*time-delay neural network*) que son redes neuronales con retraso en el tiempo las cuales provocarón el surgimiento de las versiones con convoluciones CNN-TDNN (*convolutional time-delay neural network*) que aprovechan mejor la

informacion sin necesidad de redes extremadamente grandes. [16]

6.3.2. Análisis sintáctico

Cuando el sistema de reconocimiento de voz ha convertido la señal en un texto plano, ese texto aún no posee un significado que sea entendible para cualquier sistema robótico, por lo que se utiliza una herramienta llamada analizador para poder darle un significado a dicho texto. Un analizador sintáctico es un programa que se encarga de analizar la estructura gramatical de una oración. Este funciona a través de *tokens* o elementos léxicos, los cuales son utilizados para validar si la estructura satisface con lo esperado por la estructura de referencia programada. El principal objetivo es identificar qué función cumple cada palabra o grupo de palabras. Así pues, es posible identificar en un analizador tres etapas que gestionan el proceso de análisis para alcanzar dicha meta. La primer etapa recibe el nombre de análisis léxico, y es donde el analizador toma el texto del preprocesador y lo descompone en fragmentos pequeños. Estos fragmentos de texto son agrupados en secuencias de caracteres llamados lexemas, los cuales pertenecen a un *token*, los cuales son las unidades gramaticales del lenguaje de programación que el compilador comprende. En esta etapa el analizador se hace cargo de eliminar errores y espacios en blanco para poder seguir a la segunda etapa. [17]

En el Cuadro 1 se muestra un ejemplo de este proceso de tokenización aplicado a una frase. La frase corresponde a una instrucción "junta 2 a más 5 grados". Cada palabra o número de la frase se convierte en un *token*, al que se le asocia una categoría gramatical (columna POS) y una función sintáctica dentro de la oración.

Cuadro 1. Tokenización del analizador sintáctico.

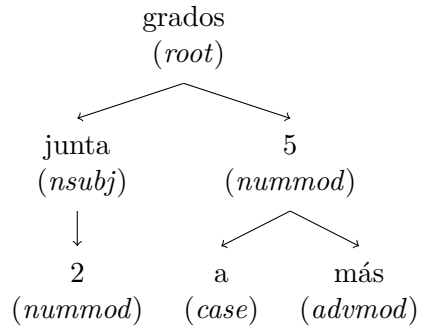
Token	POS (Tipo)	Función sintáctica
junta	Sustantivo	nsubj
2	Número	nummod
a	Preposición	case
más	Adverbio	advmod
5	Número	nummod
grados	Sustantivo	root

Nota. Elaboración propia a partir del análisis sintáctico.

En esta oración el analizador identifica a *junta* como el sujeto de la oración y le asigna internamente una función sintáctica (nsubj). Por otro lado, a los números *2* y *5* los identifica como modificadores numéricos (nummod) asociados a los sustantivos, a *a* como una preposición (case) y a *más* como un adverbio modificador (advmod). En este caso el término *grados* se marca como la raíz de la estructura sintáctica (root), ya que representa el núcleo de la acción que se está describiendo.

A continuación, la etapa siguiente denominada análisis sintáctico corresponde a la construcción de un árbol de dependencias el cual es creado a través de la combinación de la gramática predefinida con los *tokens* obtenidos de la cadena de entrada. Por consiguiente, si existe algún error en la estructura creada, entonces el analizador regresa un error de sintaxis. [17]

Figura 2. Árbol de dependencias



Nota. Árbol de dependencias elaborado a partir de la frase "junta 2 a más 5 grados". Elaboración propia.

El árbol de dependencias en la Figura 2 muestra cómo se organizan las palabras de la frase "junta 2 a más 5 grados" según sus relaciones sintácticas. En este árbol, *grados* actúa como la raíz de la estructura, esto fue definido durante el proceso de tokenización. Las siguientes palabras dependen de ella de acuerdo a su función en la oración. *Junta* se conecta como el sujeto de la acción, *2* y *5* son modificadores numéricos de *junta* y *grados* respectivamente, *a* es una preposición que enlaza las palabras y *más* es un adverbio que modifica el número *5*.

6.3.3. Análisis semántico

Cuando el analizador sintáctico termina de validar la estructura gramatical de una oración entonces se le debe asignar un significado comprensible. A diferencia del análisis sintáctico, que valida la estructura gramatical.

En sistemas de control por lenguaje natural, el análisis semántico transforma expresiones como "gira la segunda junta cinco grados a la izquierda" en representaciones parametrizadas aptas para los módulos de control. De manera general, la representación abstrae los elementos lingüísticos relevantes y los codifica como una tupla (j, a), donde j apunta hacia el índice de la articulación y a indica el valor angular requerido. [18]

Existen diferentes métodos para llevar a cabo el análisis semántico. Además de los enfoques que dependen de árboles de dependencias, se emplean estrategias de análisis basadas en patrones y recursos léxicos, tales como expresiones con significado regular y diccionarios de equivalencias. [18]

Cuadro 2. Mapa funcional de términos a comandos.

Palabra reconocida	**Tipo**	**Significado funcional**
gira, mueve, rota	Verbo	Ejecutar rotación de junta
coloca, posiciona	Verbo	Establecer posición
junta, eslabón, articulación	Objeto	Referencia a articulación
primera, segunda, tercera	Ordinal textual	Índice numérico de la junta
+10, -15, 40 grados	Ángulo/valor	Magnitud e intensidad de rotación

Nota. Correspondencia ilustrativa entre formas de superficie y roles semánticos en comandos de control. Elaboración propia.

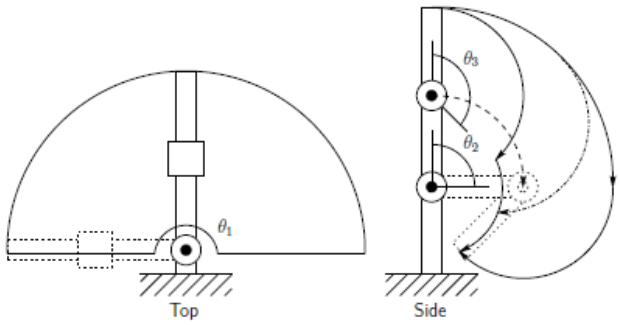
El Cuadro 2 muestra un esquema típico en sistemas orientados a intención. Se define un vocabulario operativo y un conjunto de reglas de mapeo entre formas de superficie y roles semánticos que incluyen la intención verbal, el destinatario de la acción, el índice ordinal de la articulación y los parámetros cuantitativos. Esta normalización léxica permite resolver sinonimia y variación morfológica que facilita la extracción de parámetros discretos y continuos requeridos por el sistema de control, con tolerancia al ruido y a la variación lingüística en los datos de entrada.

6.4. Fundamentos de control de manipuladores

6.4.1. Espacio articular y cartesiano

En la ciencia de manipuladores robóticos el concepto de espacio articular representa el conjunto de configuraciones definido por las variables de junta, que se agrupan en un vector q y determinan por completo la postura del manipulador. El espacio cartesiano describe posición y orientación del efector con respecto a un marco de referencia fijo, y su volumen es descrito como espacio de trabajo.

Figura 3. Espacio de trabajo generado por una articulación de un manipulador robótico.

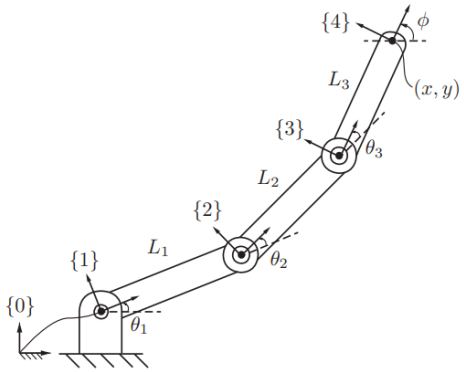


Nota. La figura muestra en vista lateral la región alcanzable por la articulación al variar el ángulo de giro. Esta imagen fue obtenida de [19].

6.4.2. Cinemática directa e inversa

El control geométrico de manipuladores se apoya en dos estrategias fundamentales. La cinemática directa establece una aplicación desde el espacio articular hacia la pose del efector final, conocida como posición y orientación respecto de un marco de referencia fijo. Este mapeo se construye mediante el producto de transformaciones homogéneas asociadas a cada eslabón y se parametriza de forma sistemática con la convención de Denavit-Hartenberg. [20]

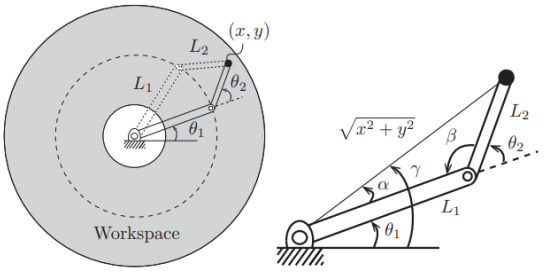
Figura 4. Cinemática directa de una cadena abierta plana con tres articulaciones tipo rotación.



Nota. Se ilustran los marcos locales y los ejes cartesianos por eslabón. La pose del efector final resulta del encadenamiento de transformaciones homogéneas a partir de las longitudes L1, L2 y L3 y de los ángulos articulares. Esta imagen fue obtenida de [20].

La cinemática inversa plantea el problema contrario, determinar las variables articulares que generan una pose objetivo. Se trata de un problema no lineal que puede carecer de solución o presentar indeterminación en configuraciones singulares. La existencia de límites articulares y la acotación del espacio de trabajo condicionan las soluciones admisibles. Su resolución puede lograrse mediante métodos numéricos basados en el Jacobiano y en técnicas de optimización, incorporando criterios como evitar colisiones y favorecer configuraciones bien condicionadas. [19]

Figura 5. Cinemática inversa de un manipulador plano de dos eslabones.



Nota. A la izquierda se muestra el espacio de trabajo y dos configuraciones equivalentes del codo. A la derecha se ilustra la solución geométrica que entrega los ángulos articulares a partir de la posición objetivo. Esta imagen fue obtenida de [20].

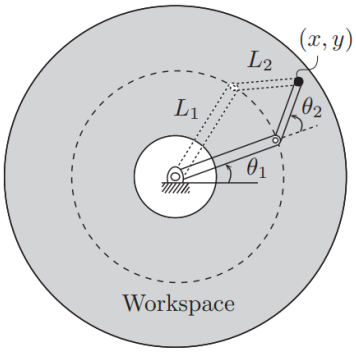
6.4.3. Límites articulares y singularidades

Los manipuladores independientemente de su estrategia de control, operan con coordenadas articulares acotadas. Cada articulación dispone de un intervalo permitido de movimiento, velocidad y aceleración. Dicho intervalo define una restricción geométrica y dinámica que reduce el espacio de configuraciones que el sistema es capaz de alcanzar. [20]

Límites articulares

Los límites articulares establecen barreras al movimiento. Al aproximarse a un límite, por pequeña que sea la variación en la pose deseada se puede exigir grandes variaciones de la variable articular correspondiente. Esto condiciona el problema inverso y puede forzar velocidades articulares elevadas durante el seguimiento.

Figura 6. Espacio de trabajo y frontera resultante de límites articulares en un manipulador 2R.



Nota. El anillo sombreado indica la región alcanzable al respetar los rangos de las juntas. La línea de puntos ilustra configuraciones que llevan al mismo punto objetivo. Esta imagen fue obtenida de [20].

Figura 7. Efecto de los límites de par articular sobre el conjunto de fuerzas del efector.

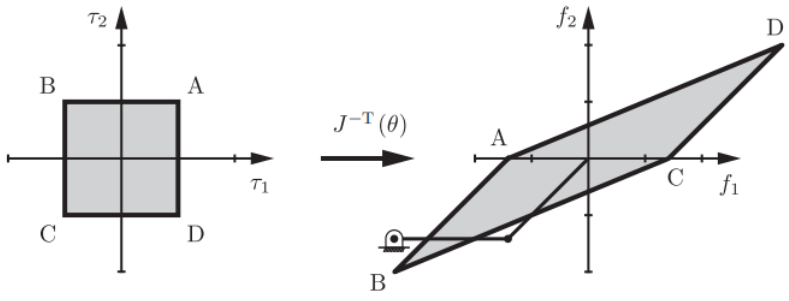


Figure 5.4: Mapping joint torque bounds to tip force bounds.

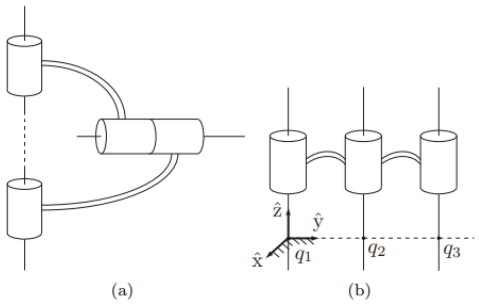
Nota. Las cotas de par por junta se proyectan mediante el jacobiano traspuesto inverso y generan un polígono de fuerzas admisibles en el efector. Esta imagen fue obtenida de [20].

Singularidades

Una singularidad es una configuración en donde el Jacobiano pierde rango, es decir, el efector pierde alguna dirección de movimiento o se requiere de velocidades articulares no acotadas para generar determinadas velocidades del efector. [20]

La destreza de un manipulador establece la capacidad de generar movimientos precisos y controlados en distintas direcciones. Cuando esta capacidad disminuye, el manipulador se vuelve menos eficiente y más sensible a errores. Por consecuencia, resulta fundamental evitar operar cerca de configuraciones singulares, ya que comprometen la estabilidad y la seguridad de la tarea encomendada. [20]

Figura 8. Casos de singularidades



Nota. (a) Singularidad cinemática en la que dos ejes de articulación de rotación son colineales. (b) Singularidad cinemática en la que tres ejes de articulación de rotación son paralelos y coplanarios. Esta imagen fue obtenida de [20].

6.5. Interacción humano-robot mediada por voz

La interacción humano-robot mediada por voz puede entenderse como un canal de comunicación en el que el usuario expresa intenciones operativas mediante lenguaje hablado y el sistema responde con retroalimentación sobre estado, interpretación y resultado. Dentro del campo de la interacción humano-robot, el valor de este canal no radica únicamente en sustituir botones, menús o interfaces tradicionales, sino en redirigir la atención del operador durante la tarea. En este sentido, el funcionamiento del sistema no depende solamente de la capacidad mecánica del sistema, sino también de la claridad de la comunicación, la previsibilidad de la interfaz y la carga mental asociada a su uso [21]. En aplicaciones donde el usuario debe observar el espacio de trabajo, supervisar la postura del robot y mantener conciencia del entorno, la voz representa una alternativa especialmente útil, ya que permite emitir instrucciones sin ocupar de manera constante las manos.

6.5.1. Principios de usabilidad y carga cognitiva en sistemas por voz

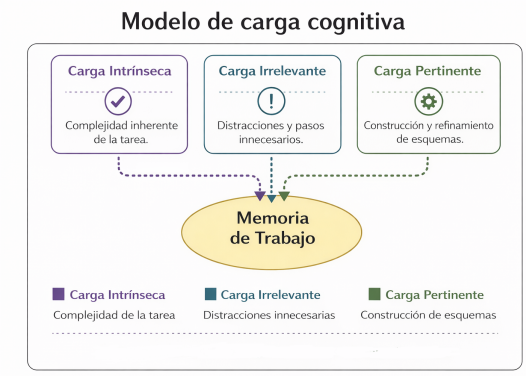
La carga cognitiva puede comprenderse como la porción de recursos mentales que una persona debe dedicar para procesar información y tomar decisiones durante una tarea. Cuando la interfaz exige recordar demasiadas opciones, interpretar mensajes extensos o corregir ambigüedades de forma repetitiva, parte de esa capacidad deja de estar disponible para la actividad principal [22]. En un sistema robótico, la demanda cognitiva no proviene únicamente del movimiento del manipulador, también intervienen la selección de la junta adecuada, la verificación del sentido de giro, la comprobación del valor angular y la validación de que la orden interpretada coincide con la intención real del usuario [21], [23].

Desde la teoría de recursos múltiples, la conveniencia del canal por voz depende de cómo se combina con el resto de la tarea. Cuando el operador ya se encuentra comprometido con observación visual del robot y supervisión manual del entorno, trasladar parte de la interacción al canal auditivo-verbal puede disminuir la competencia directa por los mismos recursos perceptivos y motores [24]. No obstante, la voz no elimina por sí sola la carga cognitiva; más bien la redistribuye. Si los comandos son demasiado largos, si existen muchas variantes sintácticas o si el sistema responde con mensajes poco previsibles, el usuario debe mantener más información en memoria de trabajo y aumenta la probabilidad de error, repetición o corrección [22], [25].

Por esta razón, la usabilidad de una interfaz por voz en robótica depende de principios de diseño concretos. La literatura sobre interfaces de voz recomienda proporcionar retroalimentación inmediata sobre el estado del reconocimiento, utilizar confirmaciones o aclaraciones cuando una acción crítica pueda fallar y mantener los mensajes del sistema breves, con el fin de no incrementar innecesariamente la demanda cognitiva del usuario [25], [26].

Aplicado al control por voz, lo anterior implica que un sistema usable no solo debe reconocer palabras, sino también disminuir el esfuerzo mental asociado a formular, verificar y corregir órdenes.

Figura 9. Modelo de carga cognitiva



Nota. Modelo conceptual basado en la teoría de la carga cognitiva, que ilustra la distribución de recursos mentales del usuario durante la ejecución de tareas. Elaboración propia.

CAPÍTULO 7

Metodología de diseño y desarrollo del sistema

En este capítulo se describe la metodología de investigación y desarrollo adoptada para la construcción dé un módulo de reconocimiento de voz aplicado al control robótico. La naturaleza híbrida del proyecto, que combina *hardware* y *software*, requirió de un enfoque iterativo que permita validar avances parciales e introducir mejoras en cada ciclo de trabajo. Por lo tanto, se seleccionó un modelo de desarrollo en espiral, el cual organizó el trabajo en planeamiento, modelado, construcción y despliegue. Esta metodología resulta pertinente ya que cada iteración ofrece retroalimentación inmediata para refinar tanto el diseño lingüístico como la integración con el robot.

7.1. Selección de la metodología de trabajo

7.1.1. Revisión de enfoques

Durante la selección de metodologías, se consideraron modelos tradicionales como cascada o *V-model*. Sin embargo, presentan limitaciones en proyectos de interacción humano a robot, ya que restringen la validación temprana y el rediseño ágil. En consecuencia, los métodos ágiles facilitan la evaluación continua de prototipos y la adaptación a condiciones reales de uso.

El modelo en espiral resulta útil en entornos con riesgos técnicos y de seguridad, ya que permite iterar sobre los requisitos y obtener distintas versiones operativas en cada ciclo.

7.1.2. Adaptación al proyecto

Cada fase del proyecto se ajusta al desarrollo del módulo de voz de la siguiente manera:

- Planificación: levantamiento de requisitos, donde se incluye la definición de la lista de comandos soportados, la respuesta ante ruido y tolerancia ante diferentes expresiones. En esta etapa se definieron además los recursos de hardware y software necesarios, entre ellos el micrófono, el entorno de desarrollo, las bibliotecas de software necesarias para el correcto funcionamiento del entorno de desarrollo y las bibliotecas de software para el robot MyCobot280.
- Modelado: diseño de la arquitectura donde se incluye el módulo de captura de audio, el módulo de transcripción con Google Speech-To-Text, el módulo encargado del análisis sintáctico y semántico con expresiones regulares y diccionarios y finalmente el módulo de generación de comandos articulares.
- Construcción: implementación del analizador semántico, la interfaz gráfica y el controlador del robot. Se configuró el entorno para ser capaz de trabajar con las bibliotecas de software importantes como *sounddevice*, *Google cloud speech*, *spacy* y *unidecode*.
- Despliegue: integración del sistema con validaciones de ejecución de comandos sobre el robot, revisando coherencia entre transcripción, interpretación y acción física.

El flujo metodológico del desarrollo del sistema, ilustrado en la Figura 10, presenta de manera visual las etapas del proyecto, desde la investigación y selección de las bibliotecas de software necesarias hasta la integración completa de los módulos correspondientes. Estos módulos incluyen el analizador semántico, el control del sistema robótico y la transcripción en tiempo real.

Figura 10. Flujo metodológico del desarrollo del sistema.

Nota. La figura ilustra el flujo metodológico seguido para el desarrollo del sistema. Se muestran las etapas prácticas desde la investigación de librerías hasta la integración completa de los módulos. Elaboración propia.

7.2. Evaluación y gestión de riesgos

Para el desarrollo del módulo de reconocimiento por voz es importante considerar las posibles situaciones de riesgo para evitar daños. Por lo tanto, se deben identificar los riesgos técnicos y los riesgos que comprometen la seguridad del usuario o del sistema. De este modo, los principales riesgos identificados son los siguientes:

- Técnicos
  - Fallos de conexión de red que interrumpen la transcripción de audio.
  - Errores en la comunicación entre el módulo de voz y el sistema robótico.
  - Latencia excesiva que impide una respuesta en tiempo real del sistema.
- Seguridad
  - Ejecución involuntaria de órdenes.
  - Sobrepaso de límites articulares.
  - Colisiones del brazo robótico con objetos cercanos o articulaciones propias.

Estos riesgos se gestionaron siguiendo lineamientos de seguridad donde se adoptaron estrategias específicas:

- Validación de comandos antes de su ejecución.
- Pruebas iniciales en entornos simulados.
- Programación defensiva dentro del controlador, el cual bloquea acciones fuera de límites permitidos.

Este enfoque permitió que el sistema evolucionara con retroalimentación constante y bajo criterios de seguridad.

CAPÍTULO 8

Configuración del entorno de trabajo y sistema robótico

Este capítulo aborda la preparación digital y física del módulo de reconocimiento de voz. Desde la selección de hardware hasta la preparación del software con las bibliotecas de software necesarias para el funcionamiento del módulo de reconocimiento de voz. La correcta integración de estos elementos fue fundamental para garantizar la comunicación fluida entre los comandos emitidos por el usuario y la ejecución de movimientos en el manipulador robótico.

8.1. Descripción del sistema robótico utilizado

El sistema robótico empleado es un brazo robótico MyCobot 280 M5, presentado en la Figura 11, diseñado por Elephant Robotics [27]. Este manipulador posee seis grados de libertad, está equipado con servomotores de precisión y es controlado mediante comunicación serial. Este robot resalta por su capacidad de ejecutar movimientos tanto relativos como absolutos dentro de sus límites articulares. Además, su arquitectura permite al usuario implementar distintos métodos de comunicación y control, especialmente el control mediante Pyhton, con bibliotecas de software dedicadas para le ejecución de comandos.

Figura 11. Manipulador robótico MyCobot280 M5.

Nota. Representación física del manipulador manufacturado por Elepthant Robotics. Esta imagen fue obtenida de [27].

8.1.1. Conexión y configuración

El manipulador MyCobot280 M5 permite establecer comunicación a través de puertos USB-C, mediante interfaces seriales tradicionales. Esta arquitectura de hardware facilita su uso en distintos entornos. El robot puede ser alimentado con una fuente externa o con baterías recargables, lo que permite su uso sin necesidad de estar conectado permanentemente a la red eléctrica. Para la configuración del manipulador robótico se admiten distintos métodos de comunicación y control, como los protocolos UART, CAN e I2C, así como entornos de programación alternativos como C++, C, inclusive ROS.

8.2. Micrófono utilizado

En este proyecto se utilizó el micrófono JBL Quantum Stream Talk, presentado en la Figura 12. Este es un dispositivo alimentado por bus USB, posee un control físico que permite ajustar la ganancia, posee un silenciador activado por pulso y posee una salida de monitoreo para audífonos. Su utilización es directa en sistemas Windows o macOS mediante cable USB de audio, por lo que no requiere una interfaz adicional.

Este micrófono JBL Quantum Stream Talk fue seleccionado ya que su patrón polar supercardioide contribuye a reducir la captación de ruidos ambientales que provienen de los laterales y la zona posterior del micrófono. Asimismo, la conexión USB-C simplifica el cableado y la instalación ya que no se necesita de interfaces de audio externas o protocolos de comunicación complejos. Además, permite el monitoreo directo mediante la salida de

Figura 12. Micrófono utilizado para la captura de audio.

Nota. Ilustración física del micrófono utilizado. Esta imagen fue obtenida de [28].

audífonos y permite verificar la calidad de la señal antes de iniciar una sesión de reconocimiento. En el Cuadro 3 se presenta un resumen de todas las características extraídas de la ficha técnica del fabricante.

Cuadro 3. Especificaciones técnicas del micrófono JBL Quantum Stream Talk.

Propiedad	**Valor**
Tipo de micrófono	Condensador *electret*
Frecuencia de respuesta	50 Hz – 12 kHz
Tasa de muestreo soportada	44.1 / 48 / 96 kHz
Profundidad de bit	16 / 24 bits
Máx. SPL	110 dB
Sensibilidad	Aproximadamente 47 dB
Conectividad	USB-C
Monitoreo	Salida 3.5 mm para audífonos
Funciones físicas	Mute instantáneo, anillo LED de estado
Software de apoyo	JBL QuantumENGINE

Nota. Datos obtenidos de la hoja técnica oficial de JBL Quantum Stream Talk [28]. Elaboración propia.

8.3. Configuración del software y entorno de programación

8.3.1. Resumen del ecosistema de desarrollo

El proyecto se desarrolló con Python y el editor Visual Studio Code, empleando GitHub para control de versiones. La interfaz gráfica se implementó con Qt/PySide6 y la integración con el manipulador se realizó mediante bibliotecas específicas del fabricante y utilidades en Python. El Cuadro 4 resume los componentes y sus versiones en el entorno de validación.

Cuadro 4. Componentes del software y versiones.

Componente	Versión
Python	3.11.7
VS Code	2022
Github	2025
PySide6	6.9.0
sounddevice	0.5.1
google-cloud-speech	2.32.0
spacy	3.8.7
pymycobot	4.0.0

Nota. Versiones indispensables para el correcto funcionamiento del software.
Elaboración propia.

8.3.2. Creación del entorno virtual

A lo largo del trabajo se adoptó una política de manejo de versiones donde la prioridad es la reproducibilidad del entorno. El proyecto establece el intérprete en Python 3.11.7 y aísla las dependencias del proyecto mediante un entorno virtual, evitando discrepancias entre equipos y previene conflictos con bibliotecas del sistema operativo. El proceso de creación y activación de un entorno virtual se expone en el documento *Guía de instalación de librerías* que se encuentra en el repositorio oficial del proyecto incluido en el Anexo 16.

Cuando se crea un entorno virtual en Python, el sistema genera una carpeta independiente usualmente llamada **.venv** que contiene su propio intérprete y un espacio exclusivo para la instalación de dependencias. Esto significa que todas las librerías que se descargan e instalan quedan restringidas a ese entorno, sin mezclarse con las del sistema operativo ni con la de otros proyectos. Gracias a este aislamiento, se evitan conflictos de versiones entre paquetes, también se garantiza la portabilidad del software y se mantiene un control claro sobre las librerías específicas que requiere el proyecto.

8.3.3. Gestión de dependencias

Una vez creado el entorno virtual, se deben administrar correctamente dependencias. Todas las bibliotecas de software necesarias se gestionaron mediante el proceso de instalación expuesto en el documento *Guía de instalación de librerias* que se encuentra en el repositorio oficial del proyecto incluido en el Anexo 16, donde se expone como instalar cada paquete con su versión correspondiene. Asimismo, en el Cuadro 5 se resumen las dependencias de cada biblioteca de software utilizadas.

Cuadro 5. Dependencias detectadas en el código fuente y su función.

Paquete / Módulo	**Tipo**	**Rol principal en el proyecto**
PySide6	Externa	GUI Qt
sounddevice	Externa	Captura/reproducción de audio en tiempo real
google (cloud speech)	Externa	Cliente ASR en la nube (transcripción de voz)
unidecode	Externa	Normalización de texto (acentos)
six	Externa	Utilidades de compatibilidad Py2/3 usadas por dependencias
pymycobot	Externa	Control del brazo MyCobot 280
queue	Estándar	Colas para comunicación entre hilos
threading	Estándar	Concurrencia
time	Estándar	Temporización y esperas
re	Estándar	Expresiones regulares
functools	Estándar	Utilidades funcionales (decoradores, parciales)
typing	Estándar	Tipado estático
sys	Estándar	Acceso a intérprete/argv/salida
unicodedata	Estándar	Normalización Unicode

Nota. Paquetes indispensables para el funcionamiento general del módulo de reconocimiento de voz. Elaboración propia.

8.3.4. Configuración de la API de reconocimiento de voz

Una API (interfaz de programación de aplicaciones) es un conjunto de reglas y contratos que permite a un programa solicitar servicios a otro software por medio de llamadas definidas. En este proyecto, la API de *speech-to-text* (STT) de Google Cloud recibe audio, lo procesa en la nube con modelos de reconocimiento del habla y devuelve las transcripciones en texto para que el sistema de *parsing* semántico interprete los comandos del usuario y el controlador robótico los ejecute bajo condiciones de seguridad.

Para usar STT de Google Cloud es necesario contar con una cuenta de Google Cloud y habilitar la facturación del proyecto. El proceso es el siguiente:

- Crear o seleccionar un proyecto en la consola de Google Cloud.
- Habilitar la API “Cloud Speech-to-Text” para ese proyecto.
- Vincular un método de pago (facturación) y revisar las cuotas vigentes.
- Crear una cuenta de servicio con rol mínimo recomendado y, si es necesario, *Storage Object Viewer* para logs relacionados.

- Generar una clave JSON de la cuenta de servicio y guardarla en un directorio seguro fuera del repositorio.
- Definir la variable de entorno *KEYFILE* apuntando al archivo JSON.

La autenticación se hace con la cuenta de servicio. El archivo JSON contiene claves privadas, no debe subirse a Git ni compartirse. Se debe limitar los permisos al mínimo necesario y rotar las claves si cambian las políticas de seguridad.

CAPÍTULO 9

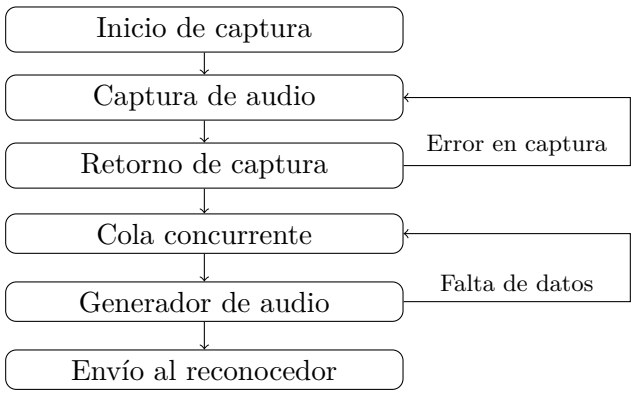
Diseño del módulo de reconocimiento de voz

El módulo de reconocimiento de voz se estructura como un proceso que inicia desde la captura de audio y prosigue con la transcripción en tiempo real. Sobre esa salida se aplica un análisis sintáctico-semántico que resulta en la generación de acciones de control articular. Estas acciones son finalmente validadas por el sistema y se ejecutan al cumplir con las restricciones definidas. Finalmente, el sistema retroalimenta al usuario mediante la interfaz gráfica.

9.1. Arquitectura del módulo de reconocimiento de voz

El módulo se implementa como una cadena de procesamiento asíncrona y no bloqueante cuya solución separa responsabilidades por capas, solución la cual se representa en un diagrama de bloques (Figura 13). La primera capa es la adquisición de audio, luego el reconocimiento en *streaming* para posteriormente el análisis sintáctico-semántico y así finalmente la generación de acciones y ejecución sobre el manipulador. La comunicación entre capas se logra mediante colas, *callbacks* y señales Qt, lo que habilita baja latencia y una interfaz receptiva. Si en algún punto del proceso, como en la generación de audio, no se recibe suficiente información o hay un retraso el flujo regresa a la capa anterior. Asimismo sucede con la captura de audio, para asegurar que los datos estén listos antes de continuar.

Figura 13. Proceso de captura de audio.

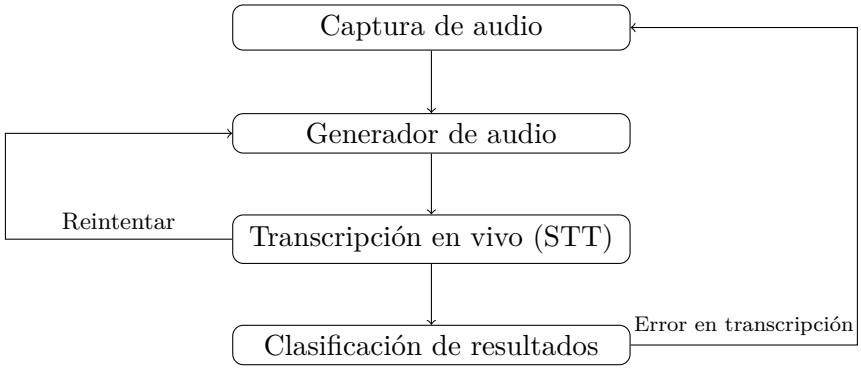


Nota. Elaboración propia.

9.2. Configuración y uso de STT en tiempo real

La transcripción en tiempo real se configura para audio PCM lineal a 16 kHz, idioma español y puntuación automática. Se habilitan resultados interinos con el fin de proveer retroalimentación inmediata, mientras que sólo las hipótesis marcadas como finales activan el análisis semántico. Las hipótesis son la frase final que el *STT* estableció. Posteriormente, el cliente de reconocimiento consume el generador de audio y produce un flujo de respuestas que el bucle clasifica de forma explícita en parciales y definitivas. Este proceso se ilustra en la Figura 14, siendo el resultado final una frase completa que continúa hacia el analizador sintáctico-semántico.

Figura 14. Diagrama de bloques del flujo de transcripción de voz en tiempo real.



Nota. Elaboración propia.

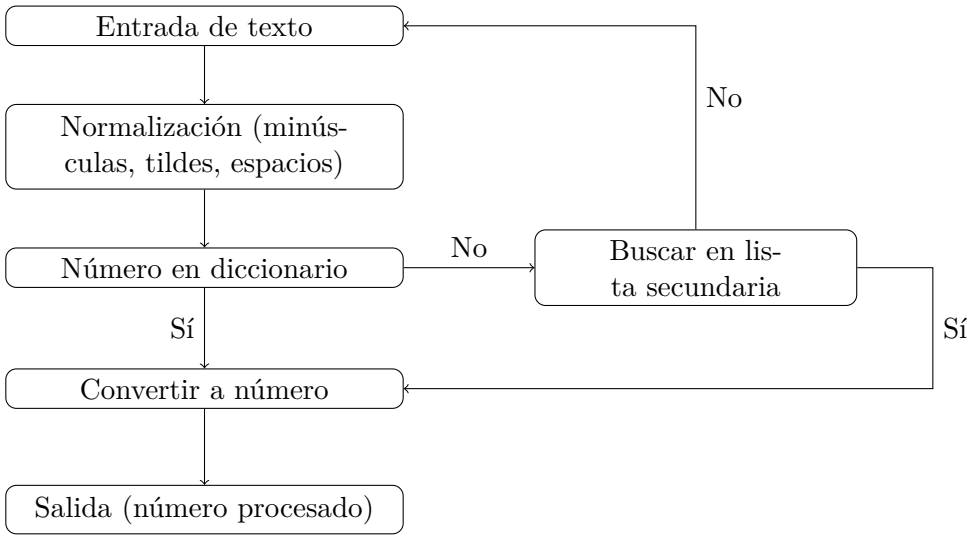
9.3. Análisis sintáctico-semántico de comandos

Esta etapa convirtió la transcripción en una lista de acciones de control sin ambigüedades. La estrategia de solución se estructuró en tres pasos. El primero consistió en la normalización lingüística, orientada a eliminar tildes, corregir variaciones menores y reducir diferencias entre expresiones equivalentes. El segundo correspondió a la extracción estructurada de la información mediante expresiones regulares y diccionarios de equivalencias. Finalmente, se construyeron tuplas de control en modo relativo o absoluto, además de identificar intenciones globales.

1) Normalización y números en español

La normalización se aplicó como estrategia para reducir la variabilidad presente en las frases transcritas. Este proceso incluyó la conversión a minúsculas, la eliminación de tildes y la compactación de espacios. De manera paralela, se interpretaron números cardinales y ordinales frecuentes en los comandos hablados. Esta transformación también permitió reducir inconsistencias generadas por la *API*, debido a que una misma palabra podía producir transcripciones diferentes en ejecuciones distintas.

Figura 15. Diagrama de bloques para la normalización y manejo de números en español.



Nota. Elaboración propia.

En la Figura 15 se ilustra el proceso aplicado a una frase para identificar y convertir los números contenidos en ella. Por ejemplo, la expresión “gira la junta tercera quince grados” se transformó en una estructura estable, en la que “tercera” se convirtió en 3 y “quince” en 15. Asimismo, mediante un diccionario de equivalencias, se estableció un procedimiento para convertir números ordinales en valores numéricos. De esta manera, en la frase “rota el primer eslabón menos treinta grados”, la palabra “primer” se sustituyó por el valor 1, con el propósito de asociarla numéricamente con el sujeto correspondiente, en este caso, el “eslabón”.

2) Verbos con signo, modalidad relativa y absoluta

Los verbos presentes en el enunciado definieron el sentido del movimiento. Términos como “sube” y “aumenta” se interpretaron como incrementos, representados con signo positivo, debido a que fueron incluidos en un diccionario de equivalencias lingüísticas. Por otro lado, expresiones como “baja”, “reduce”, “resta” o “gira en contra” se interpretaron como decrementos, representados con signo negativo. A partir de esta clasificación, se aplicaron patrones de expresiones regulares para distinguir dos tipos de órdenes: las relativas, que modificaron el valor actual en una cantidad determinada, y las absolutas, que establecieron un valor objetivo sin depender del ángulo previo de la junta.

La separación entre las modalidades absoluta y relativa permitió diferenciar la instrucción “mueve la junta 2 diez grados” de “junta 2 a cuarenta grados”. La primera se interpretó como un movimiento incremental, mientras que la segunda estableció una posición absoluta. De manera similar, las expresiones “gira la junta 6 a más 17.84 grados” y “coloca la junta 6 en la posición 17.84 grados” produjeron comportamientos distintos. En la primera, se sumaron 17.84 grados al valor previo de la junta 6. En la segunda, la junta se posicionó directamente en 17.84 grados, sin considerar su valor anterior.

3) Intenciones globales

El sistema incorporó un módulo de intenciones globales que identificó órdenes sin parámetros y las tradujo en etiquetas internas de control. Este módulo operó sobre el texto normalizado y aplicó mapeos directos. Por ejemplo, el comando “regresa a la posición inicial” se interpretó como la intención `reinicio`; de igual manera, “confirmar” se asoció con la intención `continuar` y “cancelar” con la intención `parar`. Estas intenciones no requirieron argumentos adicionales, sino que se propagaron como señales internas y produjeron cambios de estado sin necesidad de especificar juntas o magnitudes.

Dentro del módulo se implementó una función encargada de devolver una lista de tuplas *(j, a)*, donde *j* representó la junta y *a* el valor angular asociado. Estas tuplas se almacenaron para posteriormente asignarlas a una modalidad de control. En el Cuadro 6 se presenta el resultado final obtenido después de extraer las tuplas y clasificarlas en modalidad relativa o absoluta.

Cuadro 6. Ejemplos de mapeo de transcripción a acciones.

Transcripción	**Salida del analizador**
Gira la junta 2 diez grados	(’REL’, 2, +10)
Baja j3 cinco grados	(’REL’, 3, -5)
Junta 1 a 40 grados	(’ABS’, 1, +40)
Volver a home	(’HOME’)

Nota. Resultados esperados luego del análisis semántico. Elaboración propia.

En conjunto, estas piezas constituyen una solución legible y mantenible, donde la normalización reduce la ambigüedad, estandarizando la comunicación con el controlador.

CAPÍTULO 10

Diseño del módulo de control para sistema robótico

El módulo de control es el responsable de convertir intenciones de movimiento en trayectorias seguras a nivel articular. Esta etapa puede ser completada únicamente si el analizador finaliza de enviar un resultado definido.

10.1. Objetivos de diseño

El diseño del módulo se centra en preservar la seguridad mediante límites y validaciones previas a cada movimiento. Asimismo, expone una interfaz clara y desacoplada para que otras capas puedan accionar el robot sin conocer detalles de bajo nivel y garantizar trazabilidad de cada orden por medio de *logs* y estados.

10.2. Requisitos y supuestos

10.2.1. Requisitos funcionales

Los requisitos funcionales describen qué hace el sistema. Son las funciones, servicios y comportamientos observables que debe cumplir. Por ejemplo, "el controlador debe mover la articulación uno a un ángulo específico" o el sistema debe permitir enviar un comando "HOME". Así los requisitos funcionales pertinentes para el software son:

- Aplicar movimientos relativos y absolutos por junta ($J1 \dots J6$).
- Ejecutar "HOME" y reportar ángulos actuales tras cada acción.

- Publicar estados y *logs* para trazabilidad.

10.2.2. Requisitos no funcionales

Definen cómo debe comportarse el sistema mientras ejecuta esas funciones. No añaden nuevas capacidades, sino condiciones de calidad o restricciones bajo las que esas funciones se deben realizar.

- Seguridad: verificación estricta contra límites articulares y una ventana de tolerancia.
- Desacoplamiento: interfaz por señales y cola de acciones.

10.3. Controlador y API pública

El controlador se implementa como un objeto reactivo con un hilo dedicado a la ejecución. La comunicación hacia el exterior se realiza por señales (estado, ángulos, *logs*, errores) y la recepción de órdenes mediante una cola de acciones. La API pública ofrece como mínimo: start(), stop(), apply_actions(pares_REL) y apply_absolute(pares_ABS). Cada operación es registrada y la última lectura de ángulos se conserva para validaciones posteriores.

Cuadro 7. Señales y contratos de la API del controlador.

Señal	**Tipo de dato**	**Semántica**
sig_status	str	Estados de alto nivel: *DISCONNECTED*, *CONNECTED*, *EXECUTING* y *ERROR*.
sig_angles	list[float]	Ángulos actuales $[J1 \dots J6]$ obtenidos después de las lecturas o movimientos.
sig_log	str	Trazas de validación y ejecución, como bloqueos, movimientos y retorno a la posición *HOME*.
sig_error	str	Errores de entrada y salida o del actuador.

Nota. Las señales descritas constituyen la interfaz pública del controlador. Permiten desacoplar la lógica de control de los módulos externos y facilitan la integración con la interfaz gráfica, el analizador semántico y las rutinas de validación en tiempo real. Elaboración propia.

10.4. Modelo de comandos y política de seguridad

Internamente el controlador acepta dos tipos de acciones:

- Relativas (REL): $(J, \Delta\theta)$. Se calcula un candidato $\theta^* = \theta_{\text{actual}} + \Delta\theta$ y se valida contra límites del usuario.
- Absolutas (ABS): $(J, \theta_{\text{dest}})$. Se valida directamente θ_{dest} contra límites.

10.4.1. Límites articulares y ventana de tolerancia

El cuadro 8 resume los límites operativos configurados por junta representado en grados y la ventana de tolerancia ($\pm 5°$) que define la "ventana de comando". Si un destino cae fuera de los límites de usuario, la orden se bloquea.

Cuadro 8. Límites por junta.

	J1	**J2**	**J3**	**J4**	**J5**	**J6**
MínMáx	0150	−1200	0150	−135135	−120120	−160160
Ventana	−5155	−1255	−5155	−140140	−125125	−165165

Nota. Restricciones establecidas por el usuario. Elaboración propia.

10.4.2. Reglas de validación

- Relativo: Se arma un candidato por junta y se verifica que cada θ^* quede dentro de los límites. Si no, se bloquea y no ejecuta el comando.
- Absoluto: Se rechazan destinos fuera de los límites antes de enviar al actuador.
- Lectura y realimentación: Tras cada envío, se vuelve a leer el vector de ángulos y se emite por medio de una variable para poder mostrarlo textualmente en una interfaz gráfica.

10.5. Gestión de errores y trazabilidad

Los errores de comunicación, lecturas inválidas o puertos ocupados se reportan por medio de variables dentro del sistema, manteniendo el estado interno coherente. Las órdenes se procesan en una cola FIFO con descarte explícito si el controlador está deteniéndose, evitando condiciones de carrera.

CAPÍTULO 11

Implementación e integración del sistema

Este capítulo describe la unión de los módulos que constituyen el sistema de reconocimiento de voz en conjunto. Unifica la interfaz gráfica (GUI), servicio de transcripción en tiempo real (STT), analizador sintáctico- semántico de comandos y controlador del manipulador. Primero la interfaz gráfica lanza y monitorea el reconocimiento, luego la transcripción entrega la frase final y el analizador convierte texto en intenciones y parámetros (*junta*, *ángulo*, *modo*); y el controlador ejecuta movimientos con validaciones de seguridad (*límites*, *ventanas de comando* y *confirmaciones*).

11.1. Arquitectura de los módulos

La integración se organizó en torno a cuatro bloques principales:

- **Interfaz gráfica (GUI)**: permite iniciar o detener el reconocimiento, visualizar transcripciones en vivo y finales, y mostrar los estados y ángulos actuales del robot.
- **Servicio de transcripción (STT)**: procesa audio en bloques de 100 ms a 16 kHz, devolviendo hipótesis interinas para retroalimentación inmediata y finales para la ejecución.
- **Analizador sintáctico–semántico**: normaliza texto, identifica verbos de dirección, junta afectada y tipo de movimiento (relativo o absoluto), además de intenciones globales como `HOME`, `CONFIRM` y `CANCEL`.
- **Controlador**: recibe comandos estructurados, valida que no excedan límites de seguridad y ejecuta los movimientos sobre el manipulador, enviando retroalimentación de estado a la GUI.

11.2. Interfaz entre componentes

La comunicación se implementó con un esquema de colas:

- El audio capturado se encola hacia el servicio STT.
- Las hipótesis se transmite a la GUI y al analizador.
- El analizador entrega tuplas normalizadas $(modo, junta, valor)$ al controlador.
- El controlador emite señales de estado, ángulos y errores de vuelta a la GUI.

11.3. Ciclo de arranque y apagado

La secuencia de inicialización garantiza que cada módulo esté listo antes de aceptar comandos:

- El usuario activa la opción *Encender* en la GUI.
- Se inicia el hilo de captura de audio y se crea la sesión de reconocimiento en *streaming*.
- El controlador establece conexión con el manipulador, enciende LEDs y verifica los ángulos iniciales.
- Una vez validado el estado, el sistema queda en modo listo.

Para el apagado, la GUI detiene la captura de audio, cierra hilos, termina la sesión STT y devuelve al robot a un estado estable.

11.4. Manejo de errores

Durante la integración se definieron mecanismos para tolerancia a fallas:

- **STT**: reintentos automáticos ante pérdida de conexión de red.
- **Controlador**: bloqueo de movimientos fuera de rango con notificación explícita.
- **Colas**: descarta mensajes cuando el sistema se encuentra en estado de apagado.
- **Lecturas**: reintentos escalonados si se reciben vectores de ángulos incompletos.

CAPÍTULO 12

Validación experimental, análisis semántico y limitaciones

12.1. Metodología de validación

12.1.1. Entorno y condiciones controladas

Las pruebas se realizaron sobre el manipulador MyCobot 280 M5, conectado a una computadora portátil Predator G9-593. La captura de audio se efectuó con el micrófono JBL Quantum Stream Talk, manteniendo una distancia de locución no mayor a 1 m entre el emisor y el transductor. Los ensayos tuvieron lugar en un espacio interior con condiciones controladas pero con ruido ambiental moderado propio de personas conversando.

12.1.2. Escenarios y protocolos

El protocolo seguido fue deliberadamente iterativo y centrado en el uso real del sistema. Cada sesión se comenzaba preparando las condiciones de ensayo, es decir, la conexión del micrófono y verificación de niveles, encendido del manipulador, puesta en la posición inicial y comprobación de red para la transcripción en tiempo real. Posteriormente se iniciaba la captura de audio y la transcripción, entonces el operador pronunciaba los comandos previstos y observaba dos evidencias en paralelo, lo que la interfaz gráfica mostraba como hipótesis final de la transcripción y la reacción efectiva del robot, es decir, si se movía y cuánto se movía. A partir de esa observación inmediata se determinaba si el resultado era aceptable o si había ocurrido un error; cuando aparecía un desajuste, se aplicaba una micro iteración de mejora (ajuste de dicción o pausa entre palabras, revisión del modo absoluto/relativo, verificación de límites y, cuando era necesario, modificación del patrón del comando). El objetivo era consolidar una secuencia natural de hablarvercorregir.

Este mismo procedimiento se repitió con distintas variantes del contexto como nivel de ruido típico del laboratorio, diferentes distancias al micrófono dentro del rango operativo y cambios controlados de modo de control, pero siempre manteniendo la misma lógica.

12.1.3. Métricas de evaluación

La evaluación se enfocó en medir "qué tan bien se entiende y ejecuta" un comando hablado, sin exigir perfección textual cuando esta no es necesaria para la acción correcta. Por ello se distinguieron tres niveles de resultado para cada intento. Primero, la transcripción puede ser exacta o contener pequeñas desviaciones; lo relevante es si esas desviaciones afectan o no la interpretación. Segundo, el análisis semántico puede extraer correctamente la intención y los argumentos (junta, valor y modo) aun cuando la cadena de texto no sea idéntica a la esperada. Tercero, la ejecución puede completar el movimiento solicitado dentro de límites y tolerancias.

Con esta lógica, un comando se considera correctamente interpretado y ejecutado cuando, a pesar de ligeras imperfecciones en la transcripción, el sistema infiere la intención adecuada y aplica al robot la acción solicitada con la magnitud correcta, signo y valor dentro de la tolerancia definida. Si la desviación textual no afecta la intención ni los argumentos, el intento cuenta como acierto. En cambio, cuando la transcripción o el análisis semántico llevan a una intención equivocada, a argumentos erróneos, o a la imposibilidad de ejecutar la acción, el intento se considera fallo.

Sobre esta base se definieron las métricas principales. La métrica de referencia es el porcentaje de comandos correctamente interpretados y ejecutados (CCE), que cuantifica la utilidad efectiva del sistema:

$$\mathrm{CCE} \;=\; \frac{\#\,\text{comandos que terminan en la acción correcta}}{\#\,\text{comandos emitidos}}.$$

12.2. Iteraciones y resultados

12.2.1. V0: evaluación preliminar con Whisper

Durante la primera iteración se intentó integrar *Whisper* como motor ASR. El entorno de pruebas incluía una GPU NVIDIA GeForce GTX 1060 (6 GB GDDR5), cuya memoria de video disponible condiciona el tamaño de modelo utilizable. La documentación oficial del repositorio de Whisper reporta requerimientos aproximados de VRAM por tamaño de modelo: *medium* alrededor de 5 GB y *large* alrededor de 10 GB, además de enfatizar que la velocidad real dependen del hardware y del idioma. Esto sitúa a la GTX 1060 en el límite para *medium* y la deja por debajo del umbral práctico para *large*.

El intento de cargar *large* en la GTX 1060 resultó inviable por memoria, al pasar a *medium*, el sistema llegó a inicializar pero la latencia fue elevada y la calidad de la transcripción en español resultó inestable en el flujo de comandos: repeticiones espurias, fragmentos no relacionados y degradación bajo habla continua. Este patrón de líneas repetidas o *looping* ha

sido observado por terceros en implementaciones afines, donde ciertos segmentos se repiten o se traducen indebidamente.

A continuación en la Figura 16 se muestran extractos del registro de consola obtenidos en la prueba, donde en el motor de transcripción se forzó el lenguaje español para evitar problemas de detección. La frase utilizada probar el motor de transcripción fue "Hola, mueve la junta uno treinta grados".

Figura 16. Registro de consola: ensayo Whisper (*medium*).

```
Cargando modelo medium para comando ...
> Consumidor activo. Esperando la frase clave...
(kws) hola. hola.
(kws) hola. hola.
> Frase clave detectada -> grabando comando...
> Comando detectado -> hola hola
(No se entendio ningun comando valido)
(kws) no, no, no, no, ... (repeticion)
(kws) junta, uno, treinta junta, uno, treinta
(kws) junta 130 grados
```

Nota. Este registro fue recuperado de [29].

El comportamiento demostró que el motor era débil. Este transcribió "hola" en varias ocasiones, cuando debió hacerlo una sola vez. Asimismo, no fue capaz de estructurar la oración en el orden emitido y transcribió repetidamente las palabras "junta" y "treinta".

Se decidió descartar Whisper para su uso en tiempo real. La GPU GTX 1060 de 6 GB resultó insuficiente para ejecutar modelos grandes y, aun con el modelo *medium*, la latencia fue elevada y la transcripción en español presentó repeticiones y desalineaciones.

Este descarte no invalida el uso de Whisper. Con hardware que disponga de mayor VRAM o con variantes optimizadas, su implementación podría reconsiderarse. Sin embargo, para los casos de uso y los recursos disponibles, no cumplió con los criterios operativos establecidos.

12.2.2. V1: integración de STT en *streaming* y analizador sintáctico

Esta versión sustituyó Whisper por un servicio de *speech-to-text* en *streaming* y añadió un analizador sintáctico pequeño. Las principales novedades fueron la implementación de audio a 16 kHz, así como la creación de las hipótesis `[LIVE]` y `[FIN]`, que permitieron observar el texto durante el habla y al finalizar la frase. Por último, se agregó un analizador que reconoció el patrón básico "junta N a ±M grados" y publicó una salida simulada con el formato "[SERIAL_SIM] J<n>:<ángulo>".

Se configuró Google Cloud STT con idioma español, puntuación automática y resultados interinos. El analizador normalizó el texto, convirtió expresiones como "a más X" en valores con signo positivo y "a menos X" en valores con signo negativo, admitió la coma decimal y extrajo la tupla *(junta, ángulo)* mediante una sola expresión regular.

A continuación, en la Figura 17, se muestran extractos del registro de consola obtenidos durante la segunda prueba. Las frases utilizadas para probar la transcripción de Google *speech-to-text* en *streaming* fueron “mueve la junta 3 40 grados”, “mueve la junta 4 60 grados”, “necesito que muevas la junta 3 60 grados” y “necesito que muevas la junta 3 a 60 grados”.

Figura 17. Registro de consola: primera iteración para el uso en vivo.

```
Transcribiendo...
[FIN] mueve la Junta 340 grados
[SERIAL_SIM] J34:0.0
[FIN] 9 la Junta 4 60 grados
[SERIAL_SIM] J4:60.0
[FIN] necesito que muevas la Junta 360 grados.
[SERIAL_SIM] J36:0.0
[FIN] Necesito que muevas la Junta 3 a 60 grados.
[SERIAL_SIM] J3:60.0

Terminando...
```

Nota. Este registro fue recuperado de [29].

El comportamiento demostró que el motor era débil. Este transcribió “hola” en varias ocasiones, cuando debió hacerlo una sola vez. Asimismo, no fue capaz de estructurar la oración en el orden emitido y transcribió repetidamente las palabras “junta” y “treinta”.

Se decidió descartar Whisper para su uso en tiempo real. La GPU GTX 1060 de 6 GB resultó insuficiente para ejecutar modelos grandes y, aun con el modelo *medium*, la latencia fue elevada y la transcripción en español presentó repeticiones y desalineaciones.

Este descarte no invalida el uso de Whisper. Con hardware que disponga de mayor VRAM o con variantes optimizadas, su implementación podría reconsiderarse. Sin embargo, para los casos de uso y los recursos disponibles, no cumplió con los criterios operativos establecidos.

12.2.3. V1. Integración de STT en *streaming* y analizador sintáctico

Esta versión sustituyó Whisper por un servicio de *Speech-to-Text* en *streaming* y añadió un analizador sintáctico pequeño. Las principales novedades fueron la implementación de audio a 16 kHz, así como la creación de las hipótesis `[LIVE]` y `[FIN]`, que permitieron observar el texto durante el habla y al finalizar la frase. Por último, se agregó un analizador que reconoció el patrón básico “junta N a ±M grados” y publicó una salida simulada con el formato “[SERIAL_SIM] J<n>:<ángulo>”.

Se configuró Google Cloud STT con idioma español, puntuación automática y resultados interinos. El analizador normalizó el texto, convirtió expresiones como “a más X” en valores con signo positivo y “a menos X” en valores con signo negativo, admitió la coma decimal y extrajo la tupla *(junta, ángulo)* mediante una sola expresión regular.

A continuación, en la Figura 17, se muestran extractos del registro de consola obtenidos

durante la segunda prueba. Las frases utilizadas para probar la transcripción de Google *Speech-to-Text* en *streaming* fueron “mueve la junta 3 40 grados”, “mueve la junta 4 60 grados”, “necesito que muevas la junta 3 60 grados” y “necesito que muevas la junta 3 a 60 grados”.

Figura 18. Registro de consola: prueba 2 con STT y analizador semántico.

```
Transcribiendo
[FIN] Buenas hola hola hola hola.
Sin ordenes
[FIN] Quiero que me muevas la Junta 3 30 grados a la derecha.
[Acciones] -> [(3, 30.0)]
[SERIAL_SIM] J3:30.0
[FIN] Quiero que mueva la punta 2 30 grados a la izquierda y la Junta 330.
[Acciones] -> [(3, 30)]
[SERIAL_SIM] J3:30
[FIN] Quiero que muevas la Junta 230 grados a la izquierda y la Junta 460
    grados.
[Acciones] -> [(2, 30), (4, 60)]
[SERIAL_SIM] J2:30
[SERIAL_SIM] J4:60

Terminando...
```

Nota. Este registro fue recuperado de [29].

Se decidió continuar con el desarrollo del analizador semántico. El servicio de STT en *streaming* se mantuvo estable y el analizador logró extraer pares *(junta, ángulo)*, incluso cuando se incluyeron varias órdenes en una sola frase. Sin embargo, se identificaron algunas limitaciones. El sinónimo “punta” no fue reconocido, la expresión “a la izquierda” no se asoció con un valor de signo negativo y la concatenación “junta 330” activó una corrección parcial del tipo “37 0”, lo que produjo interpretaciones incompletas en algunos casos. En esta iteración se obtuvo un CEE del 60 %, debido a que se interpretaron correctamente 3 de los 5 comandos evaluados.

Esta situación no invalidó el enfoque semántico, dado que la ampliación del léxico de sinónimos, la incorporación de reglas para identificar direcciones y la desambiguación de concatenaciones numéricas permitirían mejorar el desempeño general.

12.2.4. V3: mejora del analizador sintáctico

Esta versión fortaleció el enfoque basado en reglas. Se mantuvo el flujo de transcripción mediante *speech-to-text* en *streaming* y se reescribió el analizador para tolerar variaciones lingüísticas sin abandonar la simplicidad de un analizador sintáctico. Las principales mejoras incluyeron la segmentación de cláusulas mediante conectores, como comas, “y”, “luego” y “después”; la ampliación del conjunto de sinónimos asociados con “junta”; y el mapeo de números ordinales a índices numéricos.

La estrategia adoptada para esta versión del analizador se estructuró en dos funciones. La primera dividió el enunciado en cláusulas manejables a partir de conectores como comas, “y” o “luego”. La segunda aplicó patrones alternativos para extraer todos los pares *(junta,*

ángulo), respetando el orden en que aparecieron en el discurso.

A continuación, en la Figura 19, se muestran extractos del registro de consola obtenidos durante la cuarta prueba. Las frases utilizadas para evaluar la transcripción de Google *speech-to-text* en *streaming* fueron "quiero que primero muevas la junta 4 65.3 grados y después de eso logres mover la junta 5 40 grados y coloques la junta 1 a 30 grados" y "primero quiero que muevas la junta 5 a 65.3 grados y después de eso muevas la junta 3 a 40 grados y, por último, muevas la junta 3 otra vez, pero menos 30 grados".

Figura 19. Registro de consola: prueba 3 con STT y analizador versión 3.

```
Transcribiendo
[FIN] Quiero que primero muevas la Junta 465.3 grados y despues de eso logres
     mover la Junta 5, 40 grados y coloques la Junta 1 a 30 grados.
[Entendi] -> [(5, 40.0), (1, 30.0)]
[SERIAL_SIM] J5:40.0
[SERIAL_SIM] J1:30.0
[FIN] Primero quiero que muevas la Junta 5 a 65.3 grados y despues de eso
    muevas la Junta 3 a 40 grados y por ultimo muevas la Junta 3 otra vez,
    pero menos 30 grados.
[Entendi] -> [(5, 65.3), (3, 40.0)]
[SERIAL_SIM] J5:65.3
[SERIAL_SIM] J3:40.0

Terminando...
```

Nota. Este registro fue recuperado de [29].

Como se aprecia en la Figura 19, el sistema separó y ejecutó órdenes encadenadas dentro de una misma frase. En el primer enunciado, extrajo correctamente los pares *(5, 40)* y *(1, 30)*, mientras que, en el segundo, identificó los pares *(5, 65.3)* y *(3, 40)*. No obstante, persistieron dos limitaciones. En primer lugar, la concatenación "junta 465.3" se interpretó como una secuencia unitaria, por lo que se perdió la intención correspondiente a "junta 4" con "65.3 grados". En segundo lugar, la continuación "otra vez, pero menos 30 grados" no se asoció con el último objetivo, debido a que quedó contenida en una cláusula independiente. Estas observaciones motivaron los ajustes posteriores.

12.2.5. Comparación del desempeño

Se realizó una primera comparación en la que se calculó un *CCE* simple, a partir del conteo de las órdenes intencionales que produjeron la acción correcta en los registros mostrados. La versión 0 se excluyó del análisis, debido a que no resultó operativa en tiempo real. En la versión 1 se observó un único acierto. En la versión 2, el desempeño mejoró, aunque todavía se presentaron errores en la interpretación del signo asociado con la dirección. En la versión 3, la segmentación por cláusulas y la ampliación de los patrones incrementaron el número de aciertos.

Nota. *CCE* simple calculado sobre los *logs*. Elaboración propia.

Según lo observado en el Cuadro 9, la versión 3 presentó el mejor equilibrio entre simplicidad y desempeño. Por esta razón, se adoptó como base para su adaptación e integración

Cuadro 9. CCE simple por versión.

Versión	Órdenes	Aciertos	CCE
Versión 1 (sintáctico simple)	4	1	25 %
Versión 2 (semántico simple)	5	3	60 %
Versión 3 (semántico mejorado)	6	4	66 %

en el módulo de control, así como para la ejecución de las pruebas físicas con los modos de trabajo ampliados que se presentan más adelante.

12.2.6. Pruebas del módulo completamente integrado

Configuración inicial

El sistema se evaluó con el micrófono ubicado a una distancia considerable del manipulador. Para establecer la comunicación serial, el manipulador se conectó al ordenador mediante un cable USB-C. Asimismo, fue necesario conectarlo a la red eléctrica del laboratorio, debido a que la alimentación suministrada por el ordenador no resultó suficiente. En la Figura 20 se muestra la disposición inicial del sistema antes de establecer la comunicación serial.

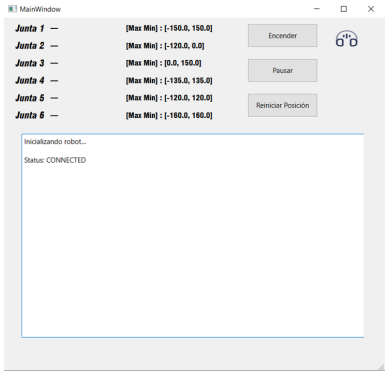
Figura 20. Configuración inicial del espacio de trabajo.

Nota. La configuración inicial no varía a lo largo de las pruebas. Elaboración propia.

Funcionamiento de interfaz gráfica

La interfaz gráfica empieza con los límites por junta visibles y el panel de control. En la Figura 21 se observa el proceso de inicialización y el enlace con el robot.

Figura 21. Inicialización de comunicación con el MyCobot280.

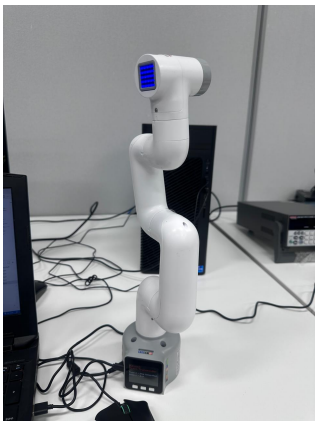


Nota. Inicio de comunicación entre el módulo de reconocimiento y el manipulador MyCobot280. Elaboración propia.

Conexión establecida

El sistema confirma *Status: CONNECTED* y aparecen lecturas angulares por junta, evidenciando adquisición correcta de estado. Asimismo, el robot en físico se coloca en posición inicial independientemente del valor de los ángulos iniciales (Figura 22). Posteriormente, la Figura 23 muestra el proceso de confirmación de la comunicacion serial entre la computadora con el robot y la captura de ángulos iniciales.

Figura 22. Posición *HOME* del manipulador MyCobot280.

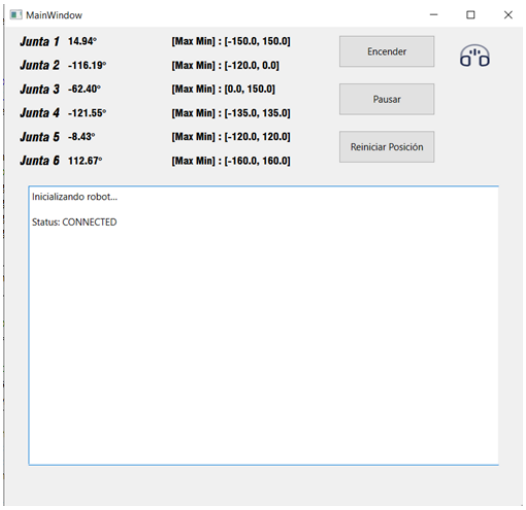


Nota. Posición *HOME* luego de iniciar la comunicación exitosamente. Elaboración propia.

Funcionamiento del sistema con el analizador semántico mejorado

Se dictan órdenes y la consola muestra su obtención como pares (junta, angulo). Para la iteración observada en la Figura 24 las órdenes se reconocen de forma consistente. Tras leer ángulos, el sistema lleva el brazo á la posición deseada luego de la verificación de límites

Figura 23. Conexión establecida y lectura de estado.



Nota. Se muestra la conexión realizada correctamente junto con los ángulos iniciales. Elaboración propia.

programados. Las órdenes emitidas para la Figura 24 fueron observadas físicamente en MyCobot280 y la posición final se observa en la Figura 25.

Es importante establecer que durante estas pruebas se fueron evaluando las capacidades del sistema robótico para responder ante las situaciones inseguras, como ángulos fuera de los límites internos o posiciones extremadamente comprometedoras.

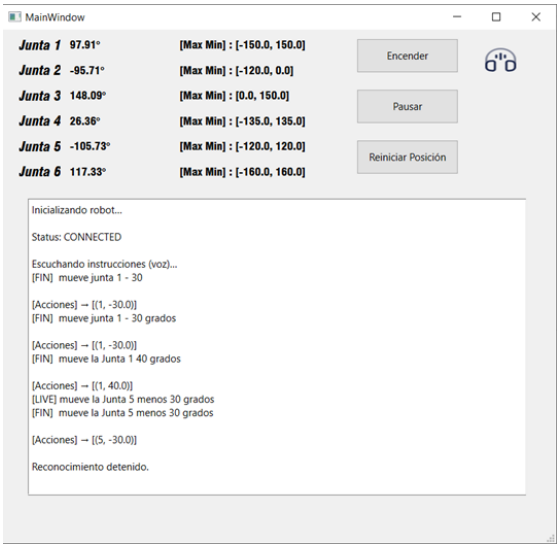
Segunda versión de interfaz, modo absoluto

Posteriormente se implementaron las modalidades de trabajo, con el fin de adaptarse a las necesidades de los usuarios finales. En la segunda interfaz se integraron dos modos, el modo absoluto y el modo relativo. El modo absoluto confirma movimientos hacia posiciones específicas dentro de los límites de cada junta. El funcionamiento de este modo se ilustra en la Figura 26.

Tercera versión de interfaz

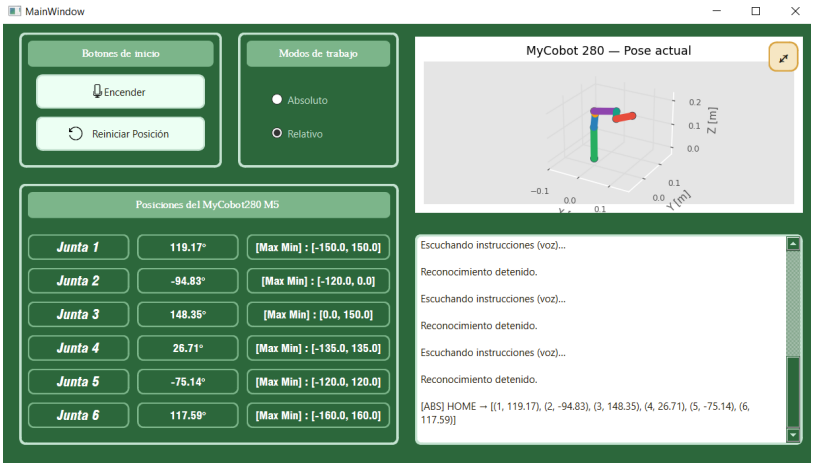
Para la tercera versión se cambiaron las posiciones de todos los componentes para que fuesen más intuitivas de ver y controlar. Como se observa en la Figura 27 se colocaron los botones de control del lado lateral izquierdo superior. El botón “Encender” es el encargado de empezar la transcripción en tiempo real y el botón “Reiniciar posición” es el encargado de colocar los ángulos iniciales al myCobot280. Por otro lado, vemos que se colocaron los valores de juntas en la zona inferior izquierda y lado de ellas la transcripción en tiempo real. Finalmente se muestra la posición real del robot en la gráfica que se encuentra en la zona derecha superior.

Figura 24. Reconocimiento por voz y trazabilidad de acciones.



Nota. Interpretación de comandos utilizando la versión más robusta del analizador sintáctico. Elaboración propia.

Figura 27. Tercera versión de la interfaz gráfica.



Nota. Diseño final de la interfaz gráfica programada. Elaboración propia.

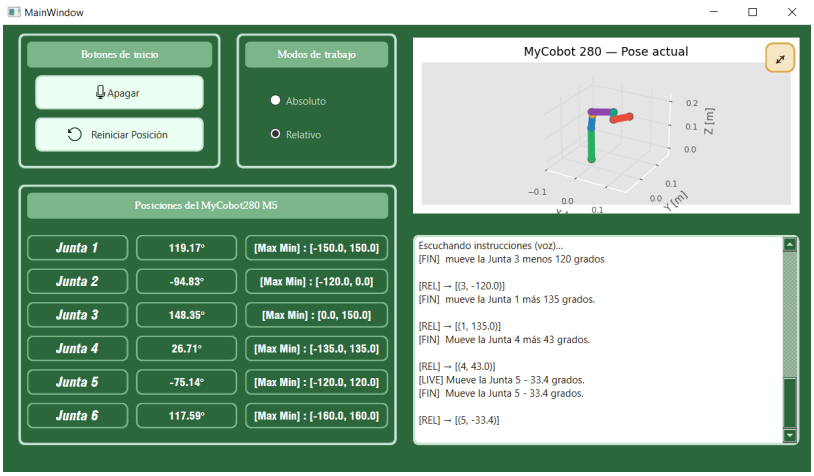
Segunda versión de interfaz, modo relativo

El modo relativo toma el ángulo inicial que tenía en ese instante la junta y lo resta o suma, dependiendo de la dirección del comando. Posteriormente envía la posición final que calculó hacia el robot. En la Figura 28 se ilustra la interpretación de los comandos emitidos para posteriormente comparar la efectividad del analizador ante el modo absoluto.

Figura 25. Posición establecida por la interpretación del analizador.

Nota. Posición final luego de emitir el comando. Elaboración propia.

Figura 28. Tercera versión de la interfaz gráfica, modo relativo.

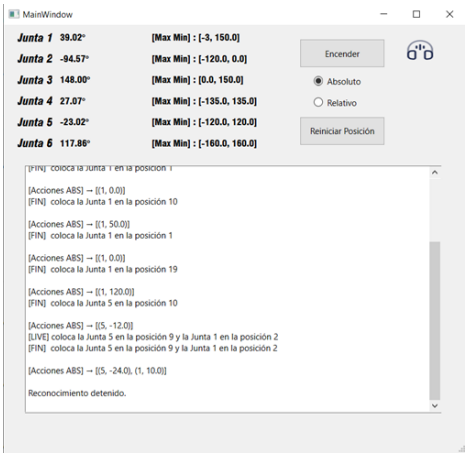


Nota. Diseño final de la interfaz gráfica programada funcionando en modo relativo. Elaboración propia.

12.2.7. Comparación de aciertos del mismo analizador por etapa

Se evaluó el mismo analizador en tres momentos específicos: antes de incorporar los modos de trabajo y, posteriormente, en los modos absoluto y relativo. En cada caso, se revisaron los registros de ejecución y se verificó si la acción interpretada correspondía con la frase pronunciada. Se consideró como acierto la coincidencia entre la junta objetivo y la magnitud solicitada, incluido el signo cuando este resultó relevante.

Figura 26. Interfaz gráfica en modo absoluto.



Nota. Interpretación de comandos para el movimiento de las juntas en modo absoluto, excluyendo los comandos establecidos para el modo relativo. Elaboración propia.

Desempeño en modo absoluto

En la Figura 26 se presenta la evaluación del modo absoluto. Las frases del tipo "coloca la junta N en la posición K" se reflejaron correctamente en las acciones absolutas y en las confirmaciones por voz. Se observaron cuatro órdenes, todas ejecutadas de manera consistente. En cada caso, el analizador respetó la junta objetivo y la magnitud solicitada.

Desempeño en modo relativo

En la Figura 28 se confirmó el cambio al modo relativo y el funcionamiento del esquema de confirmación por voz. Durante esta prueba, todos los comandos emitidos generaron una respuesta del sistema. Las frases del tipo "mueve la junta N $\pm$ M grados" se reflejaron correctamente en las acciones relativas del robot y en las confirmaciones mostradas en la caja de texto de la interfaz gráfica. Se observaron cuatro órdenes, todas ejecutadas de manera consistente. El analizador identificó correctamente la junta objetivo, la magnitud y el sentido del movimiento.

Desempeño del módulo de reconocimiento de voz

Finalmente, para evaluar la efectividad del analizador desarrollado, se realizaron 80 pruebas para cada modo de trabajo. Participaron ocho sujetos distintos, quienes emitieron diez comandos por cada modalidad. Los resultados se registraron de forma progresiva con el propósito de identificar errores y realizar ajustes durante las distintas iteraciones. Los resultados correspondientes al modo absoluto se resumen en el Cuadro 10, donde se presenta la cantidad de aciertos respecto de las órdenes emitidas, así como las observaciones utilizadas para las iteraciones posteriores.

Cuadro 10. Aciertos del analizador sintáctico-semántico en modalidad absoluta.

Usuario	Órdenes	Aciertos	*CCE*	Comentarios
Usuario 1	10	10	100 %	Comandos reconocidos correctamente
Usuario 2	10	10	100 %	Comandos reconocidos correctamente
Usuario 3	10	10	100 %	Comandos reconocidos correctamente
Usuario 4	10	9	90 %	Falló al reconocer “Un grado”
Usuario 5	10	10	100 %	Comandos reconocidos correctamente
Usuario 6	10	8	80 %	No se respetó la estructura
Usuario 7	10	10	100 %	Comandos reconocidos correctamente
Usuario 8	10	10	100 %	Comandos reconocidos correctamente

Nota. Resultados obtenidos durante las pruebas realizadas con distintos usuarios para el modo absoluto. Elaboración propia.

En el Cuadro 10 se observa un desempeño favorable del último analizador sintáctico-semántico desarrollado. Las principales fallas identificadas durante estas pruebas estuvieron relacionadas con la velocidad del habla de los usuarios y con un error de programación, debido a que la palabra “un” no se encontraba registrada en el diccionario de equivalencias numéricas. Este error se corrigió para la siguiente iteración dentro del mismo conjunto de pruebas.

Cuadro 11. Aciertos del analizador sintáctico-semántico en modalidad relativa.

Usuario	Órdenes	Aciertos	*CCE*	Comentarios
Usuario 1	10	10	100 %	Comandos reconocidos correctamente
Usuario 2	10	9	90 %	No se mencionó la palabra “Junta”
Usuario 3	10	9	90 %	No se reconoció el número de Junta
Usuario 4	10	10	100 %	Comandos reconocidos correctamente
Usuario 5	10	10	100 %	Comandos reconocidos correctamente
Usuario 6	10	8	90 %	No se separó el número de junta con el ángulo
Usuario 7	10	10	100 %	Comandos reconocidos correctamente
Usuario 8	10	10	100 %	Comandos reconocidos correctamente

Nota. Resultados obtenidos durante las pruebas realizadas con distintos usuarios para el modo relativo. Elaboración propia.

En el Cuadro 11 se presentan los resultados obtenidos con el analizador sintáctico-semántico. La primera falla se debió a que el usuario no mencionó el número de junta, por lo que no fue necesario realizar una nueva iteración del analizador. La siguiente falla ocurrió porque el sistema no reconoció correctamente el número de junta, aunque este comportamiento estuvo asociado con el proceso de captura y reconocimiento de voz. Posteriormente, se identificó que el analizador no separó adecuadamente el número de junta del valor angular solicitado. Por esta razón, se desarrolló una iteración adicional orientada a mejorar la separación de ambos valores en oraciones emitidas a una velocidad elevada.

Cuadro 12. Desempeño global del analizador sintáctico-semántico.

Etapa	Ordenes	Aciertos	*CCE*
Pre modos (Figura 24)	4	4	100 %
Modo absoluto (Cuadro 10)	80	77	96.25 %
Modo relativo (Cuadro 11)	80	76	95 %

Nota. Resultados obtenidos durante las pruebas realizadas con distintos usuarios. Elaboración propia.

Como se resume en el Cuadro 12, el analizador alcanzó un desempeño consistente en todas las etapas evaluadas. En la fase preliminar obtuvo un CCE del 100 %, mientras que en los modos absoluto y relativo alcanzó valores de 96.25 % y 95 %, respectivamente. La ligera disminución del desempeño al pasar de pruebas cortas a 80 órdenes por modo se atribuyó a variaciones en la velocidad del habla, omisiones de palabras clave y errores puntuales de reconocimiento, más que a fallas sistemáticas del analizador. En conjunto, estos resultados evidenciaron que el módulo mantuvo una tasa elevada de interpretación y ejecución correcta en escenarios más exigentes, lo que respaldó su uso como una solución viable para el control articular por voz en condiciones de operación realistas.

CAPÍTULO 13

Conclusiones

El estudio integró y evaluó un módulo de reconocimiento de voz en español para control articular de sistemas robóticos bajo un esquema de operación de transcripción en vivo, con registro de decisiones y mecanismos básicos de seguridad. A partir de los resultados, se llegaron a las siguientes conclusiones:

- El sistema desarrollado demostró ser capaz de convertir en tiempo real comandos de voz en movimientos articulares sobre el manipulador MyCobot280 M5. La integración de captura de audio, la transcripción en tiempo real, el análisis sintáctico-semántico, validación de límites articulares y ejecución sobre el robot demostraron un funcionamiento correcto bajo las condiciones de ensayo definidas.
- Las pruebas realizadas con ocho usuarios y 80 órdenes por cada modo de operación mostraron porcentajes de comandos correctamente ejecutados del 96.25 % en modo absoluto y del 95 % en modo relativo. Estos resultados permitieron concluir que el módulo es viable para el control articular por voz, siempre que se respeten las restricciones establecidas para la formulación de los comandos.
- Se determinó que el uso de la biblioteca de software Google *speech-to-text* permitió obtener una alta calidad de transcripción continua y un comportamiento estable ante variaciones moderadas de ruido y distancia al micrófono.
- La implementación de un analizador sintáctico-semántico, basado en una biblioteca de expresiones regulares, es capaz de interpretar de forma consistente los comandos de voz en modos relativo y absoluto, identificando la junta objetivo, la magnitud del movimiento y su sentido.
- Se estableció como módulo de activación por voz la estructura completa del comando. Esto activa el procesamiento de la oración cuando se emite un comando completo, de lo contrario, no se realizan movimientos articulares.
- Se diseñó e implementó una interfaz gráfica en PySide6 que integra en una sola vista la transcripción en tiempo real, los comandos interpretados, el estado de conexión

con el robot, las posiciones articulares actuales y el modo de operación (absoluto o relativo).

- El análisis cualitativo de los registros de reconocimiento mostró que, en los casos de error, el sistema tendió a bloquear movimientos cuando el destino estaba fuera de los rangos configurados o cuando la estructura no cumplía con las reglas mínimas. Este comportamiento es coherente con la programación defensiva y permite concluir que la validación en el robot fue exitosa, ya que el módulo prioriza la seguridad frente a órdenes ambiguas o mal formuladas.

CAPÍTULO 14

Recomendaciones

A partir de los hallazgos y alcance establecido, se proponen las siguientes recomendaciones para fortalecer el sistema.

- Aumentar la robustez frente a condiciones acústicas especificas. Se deben estandarizar ensayos con niveles de ruido representativos, diferentes distancias y posiciones del micrófono, así como variación de hablantes para reflejar pronunciación y velocidad de habla. Incluir un conjunto de pruebas con ruido de laboratorio y voces múltiples para caracterizar sensibilidad y sesgos del sistema.
- Ampliar la cobertura lingüística y las pruebas del analizador. Extender el diccionario de verbos y variantes del español, especialmente sinónimos de dirección, ordinales y cardinales, expresiones con y sin "grado", coma decimal y crear un corpus de regresión con frases ambiguas.
- Incorporar confirmaciones auditivas y estados explícitos. Incorporar confirmaciones por voz o sonidos breves para "LIVE" y "FIN", y mostrar bandas de seguridad por junta. Mantener consistencia con la interfaz existente y su rol de trazabilidad en tiempo real.
- Utilizar un software de diseño para interfaces gráficas que permita la inclusión de modelos renderizados en 3D para mostrar el movimiento del sistema robótico en tiempo real.
- Se recomienda empaquetar el sistema en un ejecutable independiente. De modo que no sea necesario depender del entorno de desarrollo para su ejecución. Esto facilitaría la instalación en otros equipos del laboratorio, reduciría la complejidad para el usuario final y favorecería la adopción del sistema en futuros proyectos.

CAPÍTULO 15

Referencias

[1] S. Boch, «Optimización de la herramienta de procesamiento de imágenes para el sistema Brainlab de Humana, Fase IV,» Trabajo de graduación de licenciatura, Universidad del Valle de Guatemala, 2024.

[2] R. Hernández, «Facultad de ciencias de la computación,» Trabajo de graduación de licenciatura, Benémerita Universidad Autónoma de Puebla, ago. de 2018. Dirección: https://hdl.handle.net/20.500.12371/7914.

[3] J. Piza, «Parser semántico basado en redes neuronales de tipo Deep Q-Network,» ago. de 2024. DOI: https://doi.org/10.48713/10336_43268.

[4] E. Verbit, «Reconocimiento automático de voz (ASR),» sep. de 2023. Dirección: https://verbit-ai.translate.goog/transcription/automatic-speech-recognition-asr/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc.

[5] J. Banesty, «Springer Handbook of Speech,» 2008. DOI: https://doi.org/10.1007/978-3-540-49127-9.

[6] R. Maher, *A Tutorial on Acoustical Transducers: Microphones and Loudspeakers*. Dirección: https://www.montana.edu/rmaher/ee417/transducer_tutorial.pdf.

[7] *II.5. Micrófonos: tipos y manejo*, es, II.5, Circular técnica, Serie CIRCULARES. Documento descargable sin metadatos de fecha ni autor individual., Taller de Alabanza, Área Técnica, n.d.

[8] M. Brandstein y D. Ward, eds., *Microphone Arrays: Signal Processing Techniques and Applications*. Springer, 2001. DOI: 10.1007/978-3-662-04619-7.

[9] J. Sohn, N. S. Kim y W. Sung, «A Statistical Model-Based Voice Activity Detection,» *IEEE Signal Processing Letters*, vol. 6, n.º 1, págs. 1-3, 1999. DOI: 10.1109/97.736233. Dirección: https://www.researchgate.net/publication/3342424_A_statistical_model_based_voice_activity_detector.

[10] S. B. Davis y P. Mermelstein, «Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences,» *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-28, n.º 4, págs. 357-366, 1980. DOI: 10.1109/TASSP.1980.1163420. Dirección: https://doi.org/10.1109/TASSP.1980.1163420.

[11] S. Singh, «Impact of Dataset on Acoustic Models for Automatic Speech Recognition,» *arXiv*, 2022. Dirección: https://arxiv.org/abs/2203.13590.

[12] M. Fabien, «Introduction to Automatic Speech Recognition (ASR),» *Medium*, 2019. Dirección: https://maelfabien.github.io/machinelearning/speech_reco/.

[13] A. University, «8.3. Speech Recognition,» *Aalto University*, 2020. Dirección: https://speechprocessingbook.aalto.fi/Recognition/Speech_Recognition.html.

[14] J. Hui, «Speech Recognition Feature Extraction MFCC & PLP,» *Medium*, 2019. Dirección: https://jonathan-hui.medium.com/speech-recognition-feature-extraction-mfcc-plp-5455f5a69dd9.

[15] J. Hui, «Speech Recognition Feature Extraction MFCC & PLP,» *Medium*, 2019. Dirección: https://jonathan-hui.medium.com/speech-recognition-feature-extraction-mfcc-plp-5455f5a69dd9.

[16] G. Hinton, L. Deng y D. Yu, «Deep Neural Networks for Acoustic Modeling in Speech Recognition,»

[17] B. Lutkevich, «Analizador sintáctico,» jul. de 2022. Dirección: https://www-techtarget-com.translate.goog/searchapparchitecture/definition/parser?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc.

[18] D. Jurafsky y J. H. Martin, *Speech and Language Processing*, 3rd ed. (draft). Prentice Hall, 2023, Chs. on semantics, SLU, and NER.

[19] M. W. Spong, S. Hutchinson y M. Vidyasagar, *Robot Modeling and Control*. John Wiley & Sons, 2006.

[20] K. M. Lynch y F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[21] C. Carissoli, L. Negri, M. Bassi, F. A. Storm y A. Delle Fave, «Mental Workload and Human-Robot Interaction in Collaborative Tasks: A Scoping Review,» *International Journal of Human-Computer Interaction*, vol. 40, n.º 20, págs. 6458-6477, 2024. DOI: 10.1080/10447318.2023.2254639.

[22] J. Sweller, «Cognitive Load During Problem Solving: Effects on Learning,» *Cognitive Science*, vol. 12, n.º 2, págs. 257-285, 1988. DOI: 10.1207/s15516709cog1202_4.

[23] M. A. Goodrich y A. C. Schultz, «Human-Robot Interaction: A Survey,» *Foundations and Trends in Human-Computer Interaction*, vol. 1, n.º 3, págs. 203-275, 2007. DOI: 10.1561/1100000005.

[24] C. D. Wickens, «Multiple Resources and Mental Workload,» *Human Factors*, vol. 50, n.º 3, págs. 449-455, 2008. DOI: 10.1518/001872008X288394.

[25] C. Murad, H. Candello y C. Munteanu, «What's The Talk on VUI Guidelines? A Meta-Analysis of Guidelines for Voice User Interface Design,» en *Proceedings of the ACM Conference on Conversational User Interfaces*, New York, NY, USA: ACM, 2023, págs. 1-15. DOI: 10.1145/3571884.3597129.

[26] M. H. Cohen, J. P. Giangola y J. Balogh, *Voice User Interface Design*. Boston, MA: Addison-Wesley Professional, 2004, ISBN: 9780321185761.

[27] Elephant Robotics. «MyCobot 280 M5.» Recuperado el 28 de septiembre de 2025. Dirección: https://www.elephantrobotics.com/en/mycobot-en.

[28] Harman International Industries, Inc. «Quantum Stream Talk Especificaciones técnicas,» Harman International Industries, Inc. Dirección: https://www.jbl.com/QUANTUM-STREAM-TALK.html.

[29] O. F. López Godínez, *Proyecto_Modulo_Reconocimiento_de_voz*, Repositorio en GitHub, 2025. Dirección: https://github.com/Olopez12/Proyecto_Modulo_Reconocimiento_de_voz/blob/main/speech_parser.py.

CAPÍTULO 16

Anexos

Anexo A. Código fuente del proyecto

El código fuente desarrollado para el módulo de reconocimiento de voz se encuentra disponible en el siguiente repositorio de GitHub:

https://github.com/Olopez12/Trabajo-de-graduacion-modulo-de-reconocimiento-de-voz

Este contiene los archivos correspondientes a la implementación del sistema, incluyendo el módulo de reconocimiento de voz, el analizador sintáctico-semántico, la interfaz gráfica y el controlador utilizado para la integración con el sistema robótico.