UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS AUTOMÁTICOS Y ELECTRÓNICA INDUSTRIAL

TRABAJO DE FIN DE MÁSTER

PROYECTO DE DISEÑO DE UNA APLICACIÓN PARA EL CONTROL DE MOVIMIENTOS DE UN ROBOT ABB MEDIANTE RECONOCIMIENTO DE VOZ

Autor:

Diego Esteban Mosquera Araujo

Director:

Rita Maria Planas Dangla

Convocatoria:

Julio 2019

RESUMEN

El propósito del presente proyecto, es el desarrollo de una aplicación que permita el control de movimiento del robot ABB IRB140 mediante comandos generados por voz. El proyecto presenta tres etapas que son, el desarrollo de software que permite reconocer voz y generar comandos que el robot pueda reconocer para realizar una tarea, el establecimiento de la comunicación de la aplicación desarrollada y envió de información a la controladora del robot ABB IRC5, y el desarrollo de la programación del robot para interpretar la información que recibe la controladora y ejecutar determinadas acciones. Adicionalmente, se explica la integración de las tres etapas de trabajo, y se analiza los resultados obtenidos.

La primera etapa, está enfocada al desarrollo de la aplicación de reconocimiento de voz. Aquí, se trata puntos como el análisis del funcionamiento de esta tecnología, el proceso de digitalización de la señal de audio, y las técnicas utilizadas actualmente para este fin. Además, se explica que son las APIs (Interfaz de programación de aplicaciones) y su funcionalidad para el desarrollo de aplicaciones de reconocimiento de voz. Y finalmente, se explica el desarrollo del algoritmo que es capaz de reconocer comandos por voz, y generar una secuencia de comandos por texto que pueda ser enviada y posteriormente interpretada por la controladora del robot.

En una segunda parte, se explica los procesos necesarios para establecer la comunicación entre el software de reconocimiento de voz desarrollado en Python, y la controladora del robot. Para lograr este objetivo se usa un servidor OPC, que será el encargado de establecer el vínculo entre los comandos generados por medio de la aplicación y el robot ABB IRB140. Además, se realiza la configuración del servidor, y se explica que tipo de información se comparte y en qué momento para así entender cómo funciona la lógica de la comunicación.

En la última etapa, se desarrolla la programación del robot en lenguaje Rapid. El programa proporciona la capacidad de interpretar la información recibida desde el aplicativo de reconocimiento de voz por medio del servidor OPC y ejecutar acciones con el manipulador robótico. Se analizan los parámetros de funcionamiento del aplicativo, y la lógica utilizada en la programación. Además, se detalla la guía de operación del aplicativo, se detalla los comandos programados y como generar más comandos, y se analiza las bondades del sistema.

PALABRAS CLAVE

Speech recognition, ABB IRB 140, ABB IRC5, ABB OPC server, Api Google, PyAudio, Python, Rapid, Raspberry, Reaspeaker, Reconocimiento de voz, Robot

AGRADECIMIENTOS

En primer lugar, quiero agradecer al gobierno de Ecuador y en especial a la Secretaria de Educación Superior, Ciencia, Tecnología e Innovación que es la institución que ha financiado mis estudios de Máster Universitario en Ingeniería de Sistemas Automáticos y Electrónica Industrial, estudios que son de cuarto nivel y los realice en la Universidad Politécnica de Cataluña en España.

Al alcanzar una meta es importante ver atrás y recordar a quienes estuvieron ahí empujándonos en cada tramo del camino. Por esto agradezco a toda mi familia, que siempre me brindo apoyo y estuvieron conmigo, guiándome y buscando la manera de motivarme en cada etapa. A mi mamá, que siempre que ha sido pilar fundamental para cada logro de mi vida, y para quien este logro en particular será motivo de una inmensa alegría. A mi novia Karen que ha estado a mi lado apoyándome incondicionalmente en todo momento. A mis hermanos y amigos con quienes comparto mi felicidad. A mi papá en quien siempre puedo confiar. Por esto y mucho más, mis más profundos agradecimientos a todos.

Mi trabajo final de máster lo dedico con mucho cariño y amor a quienes han estado conmigo incondicionalmente y han sido en quienes me he apoyado para seguir siempre adelante. A mi mamá, mi papá, mi novia, mis hermanos, mis amigos, mis profesores y todas las personas que formaron parte de mi vida en esta etapa.

Haz de tu vida un sueño, y de tu sueño una realidad.

Antoine de Saint-Exupéry

ÍNDICE DE LA MEMORIA

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Este capítulo muestra un análisis general del proyecto desarrollado, detallando los puntos iniciales de manera descriptiva. Aquí, se tratará la propuesta del trabajo, su repercusión, y una descripción de como el proyecto ha sido estructurado para su desarrollo

1.1. Justificación

El Presente proyecto consta de tres áreas principales que están en constante crecimiento en la actualidad y que son tecnologías clave para el desarrollo de sistemas automáticos aplicados a industrias y procesos productivos. Estas tres tecnologías son la robótica, la inteligencia artificial y las comunicaciones industriales. Debido a que su uso se ha extendido en una medida considerable, son campos de estudio en los cuales se invierte muchos recursos y continúan están en desarrollo.

La robótica ha experimentado desarrollo en dos diferentes campos, tanto en su estructura mecánica como en la manera en la que operan. Las mejoras en los robots industriales se han enfocado en el control del posicionamiento, y en la solución de los problemas dinámicos y estáticos, lo que implica que la estación robótica realice movimientos precisos, y tengan una repetibilidad aceptable con las variaciones del entorno. La inteligencia artificial en la robótica se ha enfocado en como el manipulador robótico debe manejar eventos no predecibles en un mundo no estructurado. El diseño de un sistema de inteligencia artificial para un robot debe considerar como este va a representar el conocimiento que tiene del entorno, entender lenguaje natural, aprender tareas, que tipo de planificación y resolución de problemas tendrá que hacer, cuanta interferencia espera, que tan rápido puede procesar las respuestas esperadas, y que tipos de mecanismos utiliza para interactuar con el medio. (Perez, Deligianni, Ravi, & Yang, 2016)

La robótica ha presentado un vertiginoso crecimiento desde sus inicios, y esto es debido a la infinidad de aplicaciones que se puede dar a estos manipuladores tanto dentro de la industria como fuera de ella. El desarrollo de este campo ha sido ampliamente estudiado, y cada vez más se busca incorporar nuevas funcionalidades para poder incrementar la capacidad de trabajo de los robots, no solo en tareas repetitivas sino además en acciones donde son requeridas toma de decisiones sin la supervisión de un operador.

Tecnologías como la inteligencia artificial han sido desarrolladas e incorporadas a diversos sistemas con la finalidad de mejorar el rendimiento y la interacción entre usuario máquina, este es el caso de los manipuladores robóticos industriales los cuales en la actualidad están dotados de capacidades

como la visión artificial para poder usar la información del medio donde están desarrollando su trabajo para la toma de decisiones. El incremento de la capacidad para manejar información ha apoyado al mejoramiento del desempeño de múltiples sistemas como es el caso de los robots, ya que la incorporación de inteligencia otorga un gran rango de flexibilidad en su operación que antes no se disponía.

En la actualidad existen muchas aplicaciones donde se ha comprobado que la integración de sistemas de reconocimiento de voz ha mejorado la experiencia del usuario al ser intuitivas y fáciles de usar. Es aquí donde se enmarca el presente proyecto, ya que se desea integrar la capacidad de reconocimiento de voz al control del movimiento de un robot y así dotar al sistema de nuevas características que lo harán más útil y de uso accesible para usuarios que no necesariamente sean expertos en el tema de la robótica.

1.2. Objeto

El presente proyecto tiene como objetivo el desarrollo y diseño de un aplicativo que permita el control de movimiento de un robot ABB IRB140 por medio de comandos reconocidos por voz, así como la integración de los elementos del sistema, la implementación de una interfaz de pruebas para comprobar las bondades del sistema, y el análisis de los resultados obtenidos.

Los objetivos específicos del proyecto son los siguientes:

- Estudiar la tecnología de reconocimiento de voz, los principios con los que funciona, y entender su uso en la actualidad.
- Diseñar y desarrollar un aplicativo por medio de software que sea capaz de reconocer comandos por voz, y dar como respuesta una serie de órdenes que puedan ser interpretadas por el robot para realizar un movimiento determinado.
- Establecer la comunicación de la controladora del robot con el dispositivo donde se ejecuta el aplicativo de voz, y generar las variables de conexión necesarias para establecer el intercambio de datos entre ambos sistemas.
- Realizar la programación del robot ABB IRB140 para que sea capaz de leer datos externos a su sistema por medio de una conexión, y en base a esta información realizar una serie de movimientos que serán determinados por la interfaz de pruebas de funcionamiento.

1.3. Alcance

El presente trabajo de fin de máster, comprenderá todos los puntos involucrados con el análisis del funcionamiento del sistema de reconocimiento de voz, la programación del aplicativo, la programación de la interfaz de pruebas de funcionamiento, la interconexión de todos los elementos, y la puesta en marcha del sistema de control de movimiento del robot ABB IRB140.

El desarrollo del aplicativo que reconocerá ordenes por voz será desarrollado en lenguaje de programación Python, ya que este es uno de los lenguajes más utilizados en la actualidad para el desarrollo de aplicaciones de inteligencia artificial, y dispone de soporte para ejecutarse en varios entornos. El aplicativo será ejecutado en un mini computador con el sistema operativo Windows 10, y la voz será adquirida por medio de un micrófono externo, el archivo de audio adquirido por medio del micrófono será el que posteriormente se procese para reconocer la voz.

La programación del manipulador robótico se llevará a cabo en el propio lenguaje de la marca ABB denominado RAPID, y para la controladora IRC5. Se realizarán pruebas de funcionamiento mediante el software RobotStudio. El código desarrollado tendrá el propósito de leer información externa a la controladora y poder traducir está en una respuesta de movimiento para el manipulador robótico. Se ha elegido el robot ABB IRC140 ya que este es el que se dispone en el laboratorio de robótica, y ahí es donde se desarrollara el presente trabajo de fin de máster.

La comunicación entre el aplicativo de reconocimiento de voz y las variables del programa desarrollado en Rapid para ejecutar movimientos del robot ABB IRB140 será establecida por medio del servidor OPC propio de ABB y un cliente OPC creado para correr en Python.

1.4. Antecedentes

El campo de la electrónica industrial cubre una gran cantidad de problemas que deben resolverse en la práctica industrial. Los sistemas electrónicos controlan muchos procesos que comienzan con el control de dispositivos relativamente simples como motores eléctricos, a través de dispositivos más complicados, como robots, para controlar procesos de fabricación completos. Un ingeniero de electrónica industrial se ocupa de muchos fenómenos físicos, así como de los sensores que se utilizan para medirlos. (Wilamowski & Irwin, 2011)

Dado que la mayoría de los procesos de automatización son relativamente complejos, existe un requisito inherente para el uso de sistemas de comunicación que no solo vinculan los diversos elementos del proceso industrial, sino que también están hechos a medida para el entorno industrial

específico. Además, el control y supervisión eficientes de las fábricas requiere la aplicación de sistemas inteligentes en una estructura jerárquica para atender las necesidades de todos los componentes empleados en los procesos. Esta necesidad se logra mediante el uso de sistemas inteligentes como redes neuronales, sistemas difusos y métodos evolutivos en la gestión del control de los sistemas automáticos.

Fuera del campo industrial, el hecho de hablar con máquinas es algo comúnmente más asociado a la ciencia ficción que a los actuales sistemas industriales de manufactura. De hecho, la mayoría de investigaciones y proyectos relacionados a la temática empiezan con algo relacionado a la inteligencia artificial como películas de ciencia ficción, o robots usados en películas, donde las maquinas hablaban como humanos y eran capaces de entender conversaciones complejas con humanos sin ningún problema. Sin embargo, los sistemas de manufactura industriales se han visto muy beneficiados del reconocimiento de voz en interfaces humano maquina (IHM) aun si esta tecnología no se encuentra actualmente tan desarrollada. Un sistema que implemente este tipo de tecnología, evidentemente se beneficiara en términos de autonomía, eficiencia, y agilidad. El mundo moderno requiere mejores productos a menores precios, lo que requiere tener plantas de manufactura aún más eficientes y que estén enfocadas a obtener productos de la más alta calidad, usando procesos rápidos y económicos. Esto significa autonomía, el tener sistemas que requieren poca intervención por parte de los operadores para operar de manera normal, mejorar las interfaces humano-maquina, y la interacción entre humanos y maquinas que comparten el mismo espacio de trabajo como compañeros.

En algunos casos el objetivo final a lograrse es obtener sistemas semi autónomos, sistemas altamente automatizados que requieran mínima intervención del operario. En muchas industrias, la producción es controlada de cerca en cada parte del ciclo de manufactura, el cual está compuesto de varios sistemas de manufactura en línea que realizan las operaciones necesarias para transformar la materia en bruto en un producto terminado. En muchos casos, si el sistema está bien diseñado, cada una de las celdas de manufactura que lo componen requerirán simplemente de parametrizar los valores de funcionamiento para ejecutar las tareas que fueron diseñados a realizar. Si la parametrización puede ser realizada remotamente, por medios automáticos, entonces el sistema se ha vuelto casi automático ya que la intervención del operador ha sido reducida al mínimo y requiere mínimos ajustes para operar tanto en situaciones de fallos o mantenimiento. En otros casos, una cooperación más cercana entre humano maquina es deseable pese a la dificultad para lograr esta meta por las limitaciones de la actual robótica, los sistemas automáticos, y los sistemas de seguridad industrial.

Lo descrito anteriormente, hace referencia a un escenario donde el enfoque esta puesto sobre la interfaz hombre maquina (IHM), donde la interfaz de reconocimiento de voz juega un papel muy

importante ya que la eficiencia del sistema de manufactura va a verse incrementada si la interfaz es natural, o similar a la interacción con humanos en la forma en que se ordena las cosas. Sin embargo, el reconocimiento de voz no es algo común en aplicaciones industriales por las siguientes razones.

- Las tecnologías de reconocimiento de voz y texto a voz son relativamente nuevas, a pesar que ya son tecnologías suficientemente robustas para ser usadas en aplicaciones industriales.
- El ambiente industrial es muy ruidoso lo cual pone una gran dificultad a los sistemas de reconocimiento automático de voz.
- Los sistemas industriales no fueron diseñados para incorporar este tipo de características, ya que generalmente no incluyen computadoras con gran capacidad de procesamiento especialmente dedicados al interfaz humano máquina.

El reconocimiento de voz automático (automatic speech recognition ASR) es comúnmente descrito como convertir la voz a texto. El proceso inverso, en el cual el texto es convertido a voz se conoce como síntesis de voz (Speech Synthesis text to speech TTS). La síntesis de voz generalmente genera un resultado que no es muy cercano a un sonido natural. La síntesis de voz es diferente que el procesamiento de voz, el cual involucra digitalización, comprensión (no siempre), grabación, y la posterior reproducción de la voz. Los resultados del procesamiento de voz generan sonido muy natural, pero la tecnología es limitada en flexibilidad y almacenamiento intensivo de espacio comparado con la síntesis de voz. (Pires, 2010)

Los desarrolladores de tecnologías de reconocimiento de voz, aún están trabajando en encontrar un interfaz humano maquina perfecta, un sistema de reconocimiento de voz que pueda entender a cualquier interlocutor, interpretar patrones naturales de voz, mantenerse estable a pesar del ruido del entorno, y que cuente con un vocabulario ilimitado con entendimiento del contexto. Sin embargo, los diseñadores de productos actuales pueden hacer uso de motores de reconocimiento de voz para generar mejoras en los sistemas de nuevos mercados y aplicaciones. Seleccionar un motor de procesamiento de voz, requiere el entendimiento de como estas tecnologías afectaran el tanto rendimiento como el coste del sistema, y si estos factores encajan dentro de la aplicación deseada.

Un sistema automático de reconocimiento de voz es aplicado con el propósito de comandar una celda de manufactura industrial genérica. Este tipo de sistemas pueden ser aplicados con gran éxito en aplicaciones industriales. El ruido del entorno de trabajo aun es un problema, pero al usar estructuras de comandos cortos con palabras específicas como una cadena de caracteres con un pre comando es posible reducir en gran parte los efectos del ruido.

La gramática es la que define la manera en la que un sistema de reconocimiento de voz trabaja con comandos de un usuario. Cuando una secuencia de palabras incluida en la gramática es reconocida, el sistema genera un evento que puede ser manejado por la aplicación para efectuar una tarea planeada. Las APIs de varios proveedores de servicios, proveen los métodos y las estructuras de datos necesarias para extraer la información relevante para generar un evento y para que se pueda generar una identificación apropiada del comando y obtener información de esta con detalle.

El uso de una interfaz de voz es una gran mejora para los sistemas HMI, debido a las siguientes razones:

1. El reconocimiento de voz es una interfaz natural, similar a la "interfaz" que compartimos con otros humanos, que es lo suficientemente robusta como para ser utilizada con aplicaciones exigentes. Eso cambiará drásticamente la forma en que los humanos interactúan con las máquinas;
2. El reconocimiento de voz reduce la cantidad y la complejidad de diferentes interfaces HMI, generalmente desarrolladas para cada aplicación. Dado que se utiliza una plataforma de PC, que en la actualidad tiene una gran capacidad informática, los sistemas ASR se vuelven asequibles y muy fáciles de usar.

2. DESCRIPCIÓN

2.1. Descripción general

En esta sección se describe todas las herramientas utilizadas para la elaboración del proyecto, en primera instancia se analiza el hardware o herramientas físicas como computadores, robot, controladores, sensores. En la siguiente sección se analiza las herramientas de software o programas que son usados para el desarrollo de la aplicación y para la conexión con cada uno de los elementos.

2.2. Descripción del hardware utilizado

Esta sección detalla las herramientas utilizadas en el presente proyecto, y muestra una breve descripción de cada elemento y como son utilizados para crear el aplicativo. En primer lugar, todo el proyecto es implementado en una placa Lattepanda que tiene en su interior el sistema operativo Windows 10. La obtención de la señal exterior de audio es realizada por medio de un micrófono, y la comunicación con el manipulador robótico IRB140 es creada por medio de la controladora IRC5.

2.2.1. Placa Lattepanda

El proyecto esta implementado en una placa Lattepanda que se muestra en la Figura 1, es la primera placa de desarrollo que puede ejecutar una versión completa de Windows 10. El sistema operativo Windows es necesario debido a que el servidor OPC de ABB funciona solo sobre este sistema operativo. La placa cuenta con un procesador Intel Quad Core y tiene una conectividad excelente, con tres puertos USB y WiFi y Bluetooth 4.0 integrados. También incluye un coprocesador Arduino que le permite interactuar con el mundo físico al disponer de entradas y salidas digitales o analógicas.

Figura 1. Placa de desarrollo Lattepanda (LattePanda, 2019)

2.2.1.1. Características

Las principales características de la placa Lattepanda se encuentran descritas a continuación:

- Sistema operativo Windows 10 completo
- Lattepanda es diferente de la Raspberry Pi y otras placas de desarrollo, ya que admite un sistema completo de Windows 10. Con abundantes recursos de software y un ecosistema de Windows maduro a su disposición.
- Conectividad con el exterior a través de puertos de entrada y salida digitales o analógicos.
- Computadora de placa única, con gran capacidad de potencia y rendimiento. Procesador turboalimentado Intel Quad Core de 1.8GHz, 2-4GB de RAM y 32-64GB de memoria flash incorporada, Lattepanda puede realizar fácilmente el reconocimiento de imágenes, control CNC en tiempo real y otras tareas.

2.2.1.2. Especificaciones

Las especificaciones del hardware de la placa Lattepanda se detalla a continuación.

- Procesador: Intel Cherry Trail Z8350 Quad Core 1.8GHz
- Sistema operativo: Edición completa preinstalada de Windows 10
- RAM: 4GB DDR3L
- Capacidad de almacenamiento: 64 GB
- GPU: Intel HD Graphics, 12 EUs @ 200-500 Mhz, memoria de un solo canal
- Un puerto USB 3.0 y dos puertos USB 2.0
- WiFi y Bluetooth 4.0
- Co-procesador Arduino incorporado: ATmega32u4
- Salida de video: HDMI y MIPI-DSI
- Conector de superposición de panel táctil integrado
- Compatible con 100Mbps Ethernet
- GPIO:
- 6 GPIOs del procesador Cherry Trail
- 20 GPIOs de Arduino Leonardo
- 6 conectores de sensor de gravedad plug and play
- Potencia: 5v / 2A
- Dimensión de la tabla: 88 * 70 mm / 3.46 * 2.76 pulgadas
- Peso: 55g

La Figura 2, muestra el detalle de cada uno de los componentes de la placa Lattepanda.

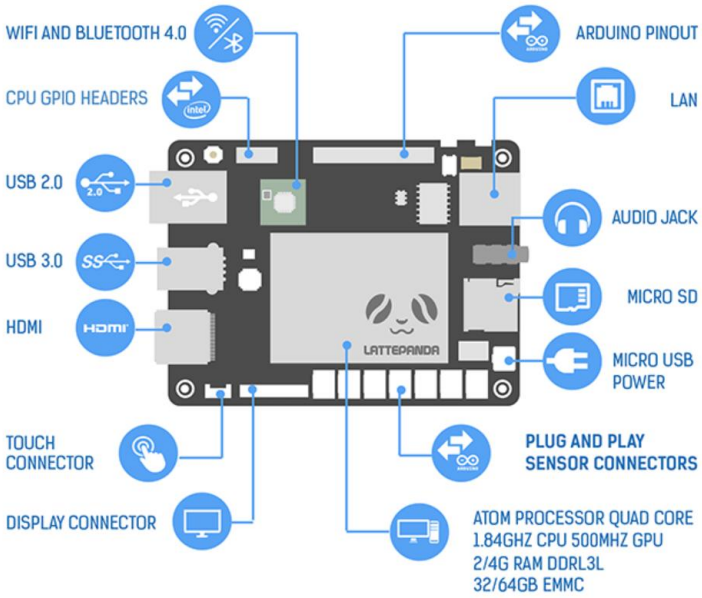

Figura 2. Características de la placa Lattepanda (LattePanda, 2019)

Lattepanda presenta todas las características de un computador de tamaño normal, pero con un coste y tamaño muy deducido. Las características de esta placa de desarrollo la hacen ideal para el presente proyecto ya que cuenta con los puertos de conexión necesarios y el sistema operativo Windows 10 que es el usado para el servidor OPC de ABB.

2.2.2. Micrófono

Una parte fundamental del proyecto es el poder ingresar información al sistema por medio de comandos de voz, para esto lo primero que se debe hacer es procesar la señal de entrada obtenida por el sensor que en este caso será un micrófono. Para que las aplicaciones de reconocimiento de voz sean robustas y confiables, el micrófono debe poder otorgar la mayor calidad de sonido posible, así se disminuye el ruido que puede introducirse debido al entorno de trabajo o a la tarjeta de procesamiento de la señal de entrada.

El micrófono seleccionado es un sistema de grabación de alta fidelidad mostrado en la Figura 3, es utilizado generalmente para aplicaciones como MSN, Skype, YouTube, karaoke, y chat por internet entre otras. Está diseñado para trabajar con Windows en un sistema plug and play, por lo que en general solo es necesario conectarlo a un puerto USB para poder usarlo.

2.2.2.1. Características

- Compatible con la mayoría de computadores con puertos USB
- Micrófono de alta calidad de sonido y un sistema de grabación de alta fidelidad (High-fidelity)
- Soporta tasas de muestreo de: 8kHz, 11.0592kHz, 22.05kHz, 44.1kHz, 48kHz. Sistema estéreo de 16-bits

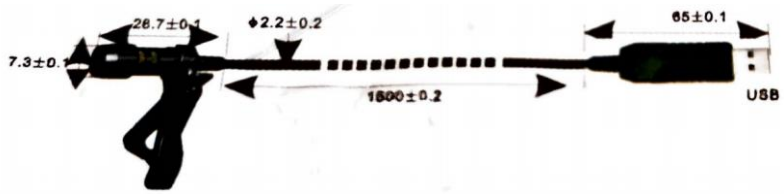

Figura 3. Características físicas del micrófono

2.2.2.2. Especificaciones

El micrófono tiene las siguientes especificaciones obtenidas de la documentación dada por la empresa fabricante.

- Sensibilidad: -30dB±3dB
- Patrón polar: omnidireccional
- Impedancia: ≥2.2kΩ
- Rango de sensibilidad: -3dB a 1V
- Voltaje de funcionamiento: 5V
- Frecuencia de respuesta: 20Hz - 16kHz
- Radio de señal: 84dB
- Longitud del cable: 1.5m
- Plug: USB2

2.2.2.3. Rango de frecuencia

El micrófono trabaja en el rango de frecuencia de 20Hz a 20kHz como muestra la información del fabricante en la Figura 4.

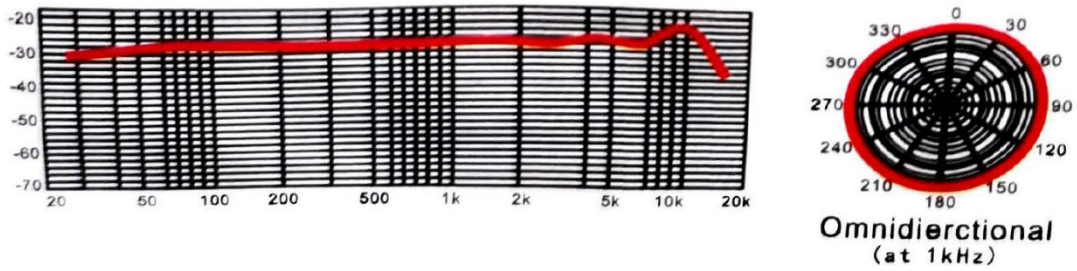

Figura 4. Rango de frecuencias del micrófono

Tras analizar las características de este micrófono se determinó que es el ideal para trabajar en el proyecto, ya que ha sido diseñado justamente para el uso con la voz, y es de bajo coste.

2.2.3. Controlador IRC5

El control del funcionamiento de un robot generalmente no se realiza en su interior, sino en un elemento externo llamado controladora, que es capaz de procesar toda la información necesaria para planificar las acciones del robot. La controladora maneja las etapas de control, cinemática, dinámica, manejo de energía, comunicaciones, conexión de periféricos, entre otras. En el caso del robot ABB IRB140 la controladora es del modelo ABB IRC5. La Figura 5 muestra la controladora.

Figura 5. Controladora ABB IRC5 (ABB Robotics, 2013)

2.2.3.1. Módulos de la controladora IRC5

El IRC5 se compone de los módulos siguientes:

- Módulo de accionamiento, que contiene el sistema de accionamiento
- Control Module, que contiene el ordenador principal (incluidas cuatro ranuras PCI para tarjetas de extensión), el panel del operador, el interruptor principal, las interfaces de comunicación, la conexión para FlexPendant, los puertos de servicio y cierto espacio para los equipos del cliente, por ejemplo, tarjetas de E/S de ABB.
- El controlador también contiene el software de sistema, es decir RobotWare-OS, que incluye todas las funciones básicas de manejo y programación descritas más detalladamente en este capítulo. Sobre RobotWare-OS es posible instalar un número de opciones con funcionalidad adicional.

2.2.3.2. Descripción del panel principal de la controladora IRC5

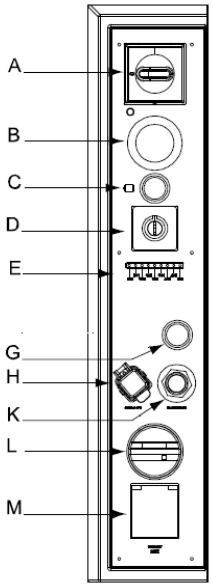

Figura 6. Panel de control de IRC5 con gabinete simple (ABB Robotics, 2013)

En su parte exterior la controladora dispone de botones físicos, estos están indicados en la Figura 6 y sirven para tareas básicas del funcionamiento como activar y desactivar los motores, arrancar el robot, o gestionar paros de emergencia entre otras funcionas más. La Tabla 1 describe el funcionamiento de cada uno de los botones de la controladora IRC5.

A	Interruptor principal y control remoto de la alimentación de los módulos de accionamiento
B	Paro de emergencia. Si está introducido, tire para liberarlo.
C	MOTORS ON
D	Selector de modo de funcionamiento
E	LEDs de la cadena de seguridad (opción)
G	Pulsador de hot plug de FlexPendant (opción)

H	Conexión de PC de servicio
K	Conexión de FlexPendant
L	Contador de tiempo de funcionamiento (opcional)
M	Toma de servicio a 115/230 V, 200 W (opcional)

Tabla 1. Descripción del panel de control del IRC5 (ABB Robotics, 2013)

2.2.4. Manipulador robótico ABB IRB140

La empresa ABB es líder mundial en ingeniería eléctrica y automatización. Esta compañía es resultado de la unión de las empresas Asea y BBC en 1988. En la actualidad, ABB tiene su sede central en Zúrich (Suiza). La oferta industrial de ABB es grande, y cuenta con 5 divisiones Power Products, Power Systems, Discrete Automation and Motion y Low Voltage Products, mismas que ofrecen productos desde interruptores para iluminación hasta sistemas de control de potencia eléctrica.

En este proyecto se propone el desarrollo de control por voz de un manipulador robótico, para esta tarea se utiliza el robot ABB IRB140 que se dispone en el laboratorio. El IRB 140 es un robot industrial de 6 ejes diseñado específicamente para industrias de fabricación que utilizan una automatización flexible basada en robots. El robot tiene una estructura abierta especialmente adaptada para un uso flexible y presenta unas grandes posibilidades de comunicación con sistemas externos. En cuanto a seguridad, el robot se ha diseñado para ofrecer una seguridad total. Cuenta con un sistema de seguridad dedicado, que se basa en un circuito de doble canal que se controla continuamente. Si cualquiera de los componentes falla, se interrumpe la alimentación eléctrica de los motores y se aplican los frenos.

2.2.4.1. Características físicas del manipulador ABB IRB140

Este manipulador tiene un peso de 98 kg sin tomar en cuenta los cables de alimentación, y se desplaza a una velocidad máxima de 1000 mm/s con un consumo de potencia de 0.44 KW, la velocidad se encuentra limitada al momento de controlar el robot manualmente. La repetitividad que estos robots pueden llegar a ofrecer es de 0.01 mm. Las dimensiones del manipulador se encuentran especificadas en la Figura 7.

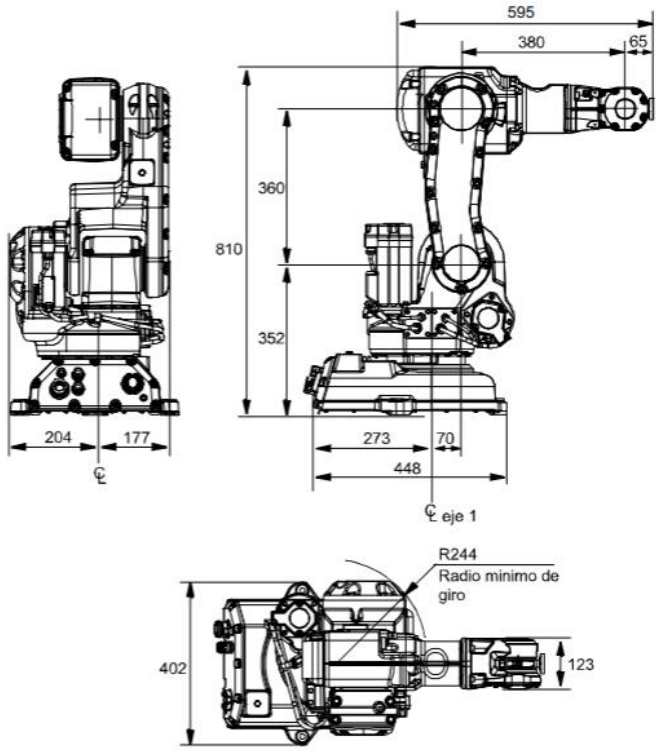

Figura 7. Vistas posterior, lateral y superior del manipulador (dimensiones en mm). (ABB Robotics, 2004)

2.2.4.2. Alcance del manipulador robótico ABB IRB140

El manipulador puede realizar movimientos con un alcance horizontal máximo de 810 mm y con una velocidad lineal de movimiento de la punta de la herramienta del orden de 1000 mm/s. En la Figura 8 se detalla el área de trabajo del manipulador donde las posiciones extremas del brazo de manipulador se especifican respecto al centro de la muñeca.

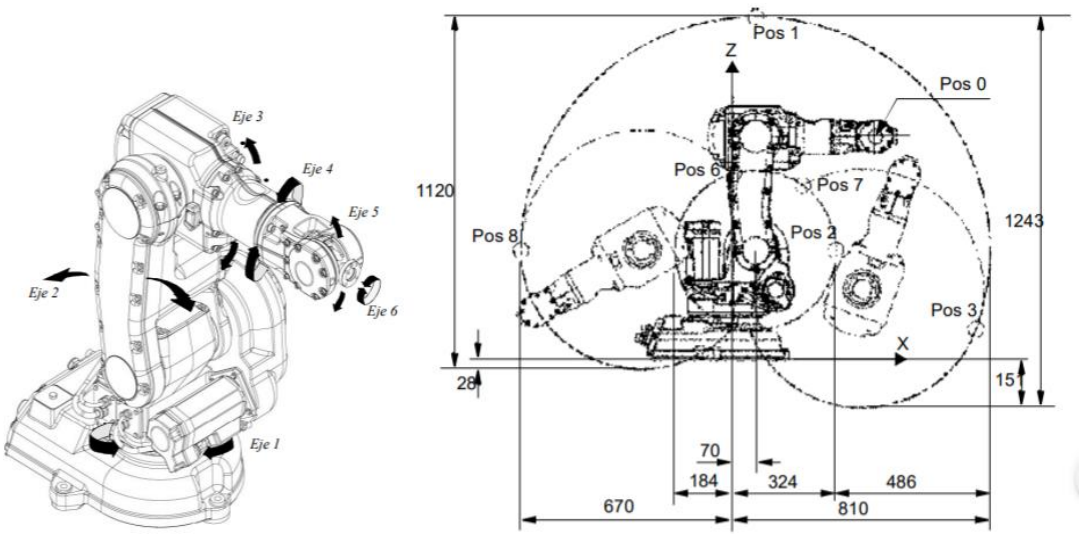

Figura 8. Ejes de rotación y límites de desplazamiento del manipulador IRB 140 (ABB Robotics, 2004)

2.3. Descripción del software utilizado

2.3.1. Python

El lenguaje de programación usado para este proyecto será Python ya que este programa es actualmente muy utilizado en el campo académico dada su relativa facilidad, su gran soporte para varias tareas como aplicaciones de inteligencia artificial, y su entorno multiplataforma. Además, el uso de Python es gratuito y puede correr en software libre como Linux por lo que no se requiere ningún tipo de licencias de uso. La Figura 9 muestra los usos principales de Python en la actualidad.

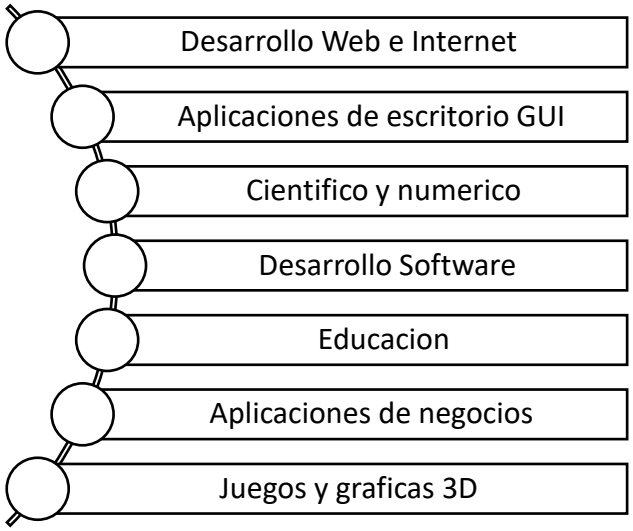

Figura 9. Aplicaciones y usos de Python en el mundo (Lutz, 2010)

Python es un lenguaje de programación interpretado de tipado dinámico cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma y disponible en varias plataformas. A continuación, se describen las bondades de Python. (Lutz, 2010)

- Multiparadigma: Soporta programación funcional, programación imperativa y programación orientada a objetos.
- Al hacer uso de una sintaxis legible, la curva de aprendizaje es muy rápida, siendo de este modo, uno de los mejores lenguajes para iniciarse en la programación en modo texto.
- Contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas comunes sin necesidad de tener que programarlas desde cero.
- Utiliza una sintaxis elegante, que facilita la lectura de los programas que escribe.

- Es un lenguaje fácil de usar que simplifica el funcionamiento de su programa. Esto hace que Python sea ideal para el desarrollo de prototipos y otras tareas de programación ad-hoc, sin comprometer la capacidad de mantenimiento.
- Viene con una gran biblioteca estándar que admite muchas tareas de programación comunes, como conectarse a servidores web, buscar texto con expresiones regulares, leer y modificar archivos.
- El modo interactivo de Python facilita la prueba de fragmentos cortos de código. También hay un entorno de desarrollo incluido llamado IDLE.
- Se extiende fácilmente agregando nuevos módulos implementados en un lenguaje compilado como C o C ++.
- Funciona en cualquier lugar, incluyendo Mac OS X, Windows, Linux y Unix, con versiones no oficiales también disponibles para Android y iOS.
- Es software libre en dos sentidos. No cuesta nada descargar o usar Python, o incluirlo en su aplicación. Python también se puede modificar y redistribuir libremente, porque mientras el idioma tiene derechos de autor, está disponible bajo una licencia de código abierto.
- Hay disponibles una variedad de tipos de datos básicos: números (coma flotante, enteros complejos y enteros largos de longitud ilimitada), cadenas (tanto ASCII como Unicode), listas y diccionarios.
- El código se puede agrupar en módulos y paquetes.
- El lenguaje es compatible con generar y capturar excepciones, lo que resulta en un manejo de errores más limpio.

Todas las características descritas anteriormente hacen de Python el lenguaje ideal para desarrollar aplicativos de inteligencia artificial, y en este caso de reconocimiento de voz. Además, que permite realizar pruebas en diferentes dispositivos como PC o tarjetas como la raspberry sin mayor dificultad.

2.3.2. Interfaz de programación de aplicaciones (API)

Una herramienta muy utilizada en la actualidad para la creación de aplicaciones es el procesamiento en la nube y el uso de APIs, una API es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software. Las siglas API vienen del inglés Application Programming Interface. En español significa Interfaz de Programación de Aplicaciones.

Una API nos permite implementar las funciones y procedimientos que engloba en nuestro proyecto sin la necesidad de programarlas de nuevo. En términos de programación, es una capa de abstracción.

el término API ha evolucionado con la revolución de los servicios web que dieron lugar a lo que hoy se conoce como API REST, o como muchos suelen llamarla, solo API.

Una API REST es una biblioteca apoyada totalmente en el estándar HTTP. Visto de una forma más sencilla, una API REST es un servicio que nos provee de funciones que nos dan la capacidad de hacer uso de un servicio web que no es nuestro, dentro de una aplicación propia, de manera segura.

Existe una gran oferta de servicios de aplicaciones de servicios, empresas como Google, Microsoft, Amazon, brindan un amplio soporte para la gestión de estos servicios que pueden incluir tareas como análisis de imágenes, audio, bigdata, e integración de servicios como Facebook, o pagos con tarjeta de crédito. Esta integración de servicios hace que el crear aplicaciones que contengan múltiples funcionalidades sea relativamente fácil y el tiempo de desarrollo se vea disminuido.

Las API de Google son un conjunto de API desarrollada por la empresa Google, que permiten la comunicación e integración de los Servicios de Google con otros servicios. Algunos ejemplos de estos servicios incluyen las API de Búsqueda, Gmail, Traductor o Maps. Las aplicaciones de terceros pueden usar esas API para extender la funcionalidad de sus servicios.

Figura 10. Speech to text API de Google

Las API proveen funcionalidades como análisis, aprendizaje automático (machine learning), o acceso a los datos de usuario (donde estén establecidos los permisos de lectura). Otro ejemplo es incrustar algún mapa en un sitio web, lo que se puede lograr usando las API de Static Maps,1 Places2 o Google Earth.3

En el caso del presente proyecto, la API de Google que se utiliza es la de Speech to text mostrada en la Figura 10, es un motor que funciona a base de machine learning y poderosas redes neuronales de múltiples capas o más conocidas como técnicas de deep learning para distinguir y reconocer características del sonido realizado al hablar e interpretar por medio de texto lo que se ha dicho. Esta herramienta transcribe el discurso que escucha por medio del micrófono y retorna varias piezas de información sobre el análisis de ese audio, entre los que se encuentra la transcripción que posteriormente puede ser utilizada de la forma más conveniente en el aplicativo desarrollado. La

Figura 11 muestra la lógica del funcionamiento de la API de reconocimiento de voz de la empresa Google.

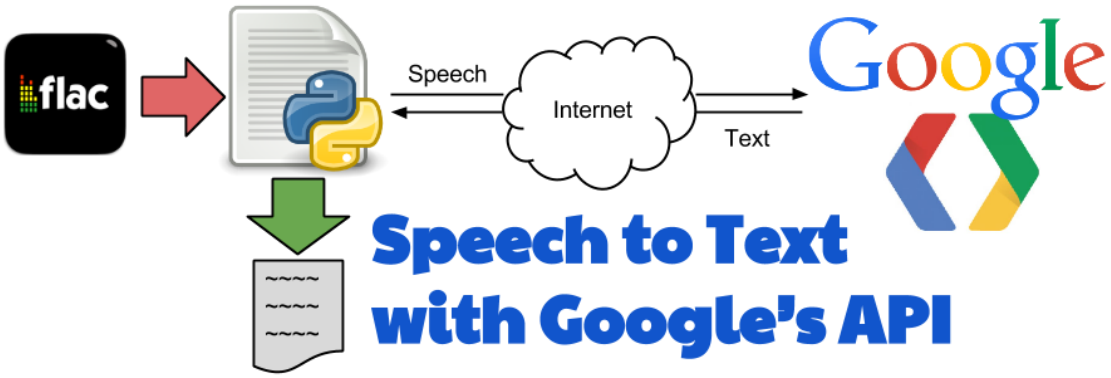

Figura 11. Funcionamiento de la API speech to text

En una primera etapa se obtiene un archivo de audio por medio de un micrófono (archivo flag), luego este se procesa con el software que en este caso será Python, el cual se encarga de subir el archivo de audio a procesar en el servidor de Google, donde se procesa por medio de algoritmos desarrollados por esta empresa y da por respuesta la transcripción de lo escuchado. La salida del texto la gestiona directamente el software Python que puede usar esta información para realizar diferentes tareas.

2.3.3. Lenguaje Rapid

Diseñado por la compañía ABB, el lenguaje de programación RAPID es un lenguaje de alto nivel para el control de los robots industriales de esta marca. Este lenguaje, proporciona una gran variedad de funciones y rutinas las cuales incluyen dinámica el manipulador, aritmética, expresiones lógicas, gestión de errores, multitarea, y muchas otras, razón por la cual es utilizado para manejar operaciones industriales complejas en el ámbito robótico.

Una aplicación RAPID, al igual que en varios programas de desarrollo de programación consta de un programa principal, y una serie de módulos del sistema que soportan o brindan características adicionales al programa principal.

Un programa de RAPID contiene un conjunto de instrucciones las cuales describen el funcionamiento del manipulador robótico. En su gran mayoría, los comandos tienen una serie de argumentos que se utilizan para definir y describir los movimientos del brazo robot con parámetros como precisión y velocidad. Se puede decir, que un programa en Rapid es una secuencia de instrucciones para controlar el robot y en general este consta de tres partes: rutina principal, subrutinas y los datos del programa.

En la Figura 12 se observa la estructura básica de un programa desarrollado en lenguaje RAPID en el que localizamos cada una de las partes que acabamos de mencionar anteriormente.

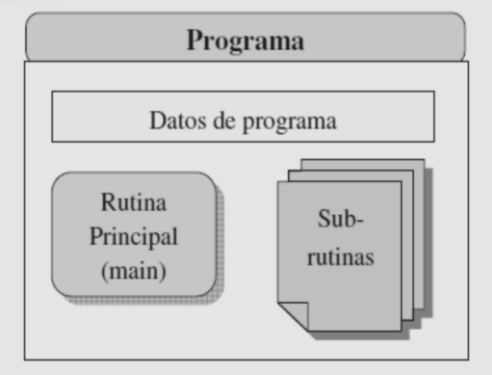

Figura 12. Estructura básica de un programa en Rapid

Como se muestra en la Figura 12, un programa en Rapid consta de los bloques rutina principal, subrutinas, y datos de programa los cuales se describen en la Tabla 2.

Rutina principal	Rutina donde se inicia la ejecución
Subrutinas	Fragmentos de código que sirven para dividir el programa en partes más pequeñas y dividirlo en forma modular
Datos de programa	Definen posiciones, valores numéricos, sistemas de coordenadas, etc.
Procedimiento	Es un subprograma. Al llamar a un procedimiento desde cualquier parte del programa principal, se pueden ejecutar tareas adicionales para el cálculo de los datos necesarios, mediante comprobaciones del estado actual del robot o de una parte concreta del mismo.
Función	Es un fragmento de código con un propósito, este puede ser alimentado de información y puede dar como respuesta un dato o una acción.
Rutinas	Las rutinas TRAP son rutinas que se ejecutan debido a la aparición de las interrupciones.

Tabla 2. Bloques de un programa en Rapid (ABB Robotics, 2010)

El lenguaje RAPID constituye una combinación equilibrada de simplicidad, flexibilidad y potencia. Contiene los conceptos siguientes:

- Estructura de programa jerárquica y modular, para admitir la programación estructurada y la reutilización de códigos
- Las rutinas pueden ser funciones o procedimientos

- Datos y rutinas locales o globales
- Asignación de tipos a datos, incluidos tipos de datos estructurados y de matriz
- Nombres definidos por el usuario a variables, rutinas y E/S
- Amplio control del flujo del programa
- Expresiones aritméticas y lógicas
- Gestión de interrupciones
- Gestión de errores (en cuanto a la gestión de excepciones en general, consulte Gestión de excepciones en la página 43)
- Instrucciones definidas por el usuario (que aparecen como parte inherente del sistema)
- Número elevado de potentes funciones incorporadas, como funciones matemáticas y específicas de robots
- Sin límite impuesto por el lenguaje (no hay ningún número máximo de variables, etc., sino que el único límite es la memoria disponible). La compatibilidad con RAPID incorporada en las interfaces de usuario, como por ejemplo en las listas de selección definidas por el usuario, facilitan el trabajo en general con RAPID.

2.3.4. Robot Studio

RobotStudio es el software de simulación y programación de robots offline de ABB, esta herramienta permite crear y simular estaciones industriales robotizadas con el diseño 3D del entorno en un ordenador. Dadas sus múltiples características, este programa cuenta con la capacidad para realizar una gran variedad de tareas programadas en los sistemas robóticos.

El software RobotStudio aporta las herramientas que ayudan a mejorar la productividad y rentabilidad de sistemas automatizados mediante manipuladores robóticos, pues permite realizar tareas tal como programación y optimización de programas sin necesidad de estar conectado en línea al sistema por lo que no afecta la producción. Adicional a estos incluye varias ventajas detalladas a continuación:

- Reducción de riesgos
- Arranques más rápidos
- Menor tiempo para modificaciones
- Aumento de la productividad

RobotStudio está basado en el controlador virtual de ABB, que es una copia simulada del software real que utilizan los robots de ABB. Gracias a este software, se puede ejecutar en nuestro ordenador un

sistema robótico que previamente haya sido diseñado antes de ser ejecutado en el robot real, con lo que se evitan costes innecesarios.

RobotStudio es una aplicación para PC que permite trabajar eficientemente con datos del IRC5. RobotStudio puede considerarse como el compañero ideal del FlexPendant, usándolos de forma que se complementan y cada uno está optimizado para sus tareas concretas. Al explotar todas las ventajas de esta potente combinación, es posible disfrutar de una nueva forma de trabajar con más eficiencia.

El FlexPendant tiene como fines principales el movimiento del robot con el joystick, la programación de posiciones, el manejo y el ajuste de movimientos, mientras que RobotStudio resulta ideal para el manejo de datos de configuración, gestión de programas, documentación en línea y acceso remoto.

RobotStudio actúa directamente sobre los datos activos del controlador. La conexión al controlador puede hacerse localmente a través de la conexión para PC de servicio y, si el controlador cuenta con la opción de RobotWare PC Interface, a través de una conexión de red.

Un sistema de control maestro seguro garantiza que RobotStudio sólo pueda tomar el control de un robot si tal operación se autoriza desde el FlexPendant.

La entrada principal a la funcionalidad de RobotStudio es un explorador de vistas de robot. Desde él se selecciona el robot con el que se desea trabajar, si hay varios robots instalados, y las partes del sistema que se desea utilizar.

El paquete básico de RobotStudio contiene:

- System Builder para crear, instalar y mantener sistemas
- Un Editor de configuraciones, para editar los parámetros de sistema del
- sistema que se está ejecutando.
- Un Editor de programas para programación en línea.
- Una grabadora de eventos, para grabar y monitorizar los eventos del robot.
- Herramientas para realizar copias de seguridad y restauraciones de sistemas
- Una herramienta de administración para autorización de usuarios
- Otras herramientas para visualización y manejo de propiedades del
- controlador y del sistema

El acceso a todas las posibilidades de RobotStudio como una potente herramienta de programación y simulación son servicios adicionales que se deben adquirir con la empresa, pero en el proyecto se

pueden usar ya que se cuenta con las licencias. En la Figura 13 se puede observar el entorno de trabajo del programa RobotStudio.

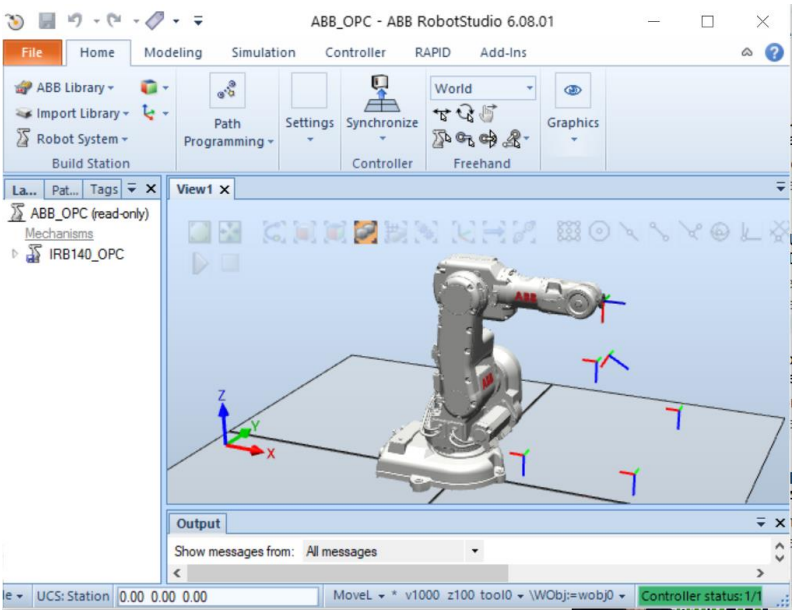

Figura 13. Pantalla principal de RobotStudio

2.3.5. ABB IRC5 OPC Server

Para establecer la conexión entre la aplicación desarrollada en Python, y el robot es necesario un medio de comunicación y para el presente proyecto se usará la herramienta OPC IRC5 de ABB que ha sido diseñada para este propósito. Con este aplicativo, se puede leer o escribir datos en ciertas variables una vez que se realiza la configuración del servidor OPC.

La aplicación de configuración del servidor OPC de ABB IRC5 mostrada en la Figura 14 se utiliza para crear y administrar Alias para los controladores de robot ABB IRC5. Un Alias es un descriptor fácil de usar que representa una interfaz de comunicaciones para un controlador de robot ABB IRC5. Se necesita crear un alias para cada controlador de robot al que ABB IRC5 accederá Servidor OPC.

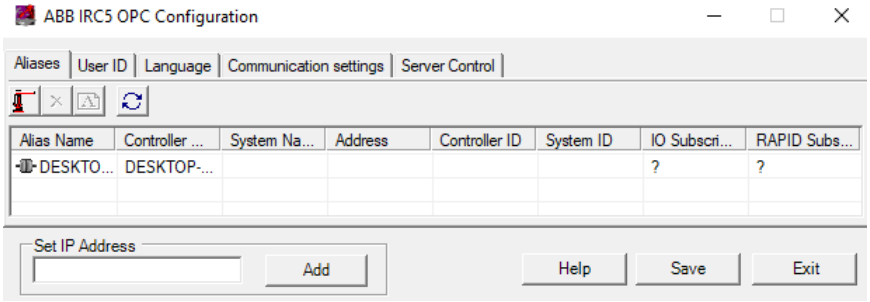

Figura 14. Pantalla principal del programa ABB IRC5 OPC Server

El servidor de ABB OPC IRC5 muestra los elementos OPC de robots con alias configurados en la red. Se debe utilizar la herramienta de configuración de ABB IRC5 OPC para crear alias para los robots que se desea monitorear. Si no se crea ningún alias antes de iniciar el navegador, no se verá ningún elemento en el espacio de nombres de acceso a datos. Existen ciertos requerimientos que se deben cumplir para establecer la conexión por medio del servidor OPC.

- El robot debe estar conectado a la misma red de computadoras que la PC ejecutando el servidor OPC.
- El servidor OPC IRC5 de ABB solo muestra datos para los robots con controladoras IRC5 de ABB con alias configurados (un descriptor que identifica un robot en particular). Las entradas de configuración se almacenan en el ID (63786) El OPC Alias Configuration file_en.xml.
- El sistema que se ejecuta en el controlador al que se refiere el alias debe tener La opción PC Interface RobotWare instalada. De lo contrario, el alias seguirá siendo creado, pero no será posible la comunicación con el controlador.
- Para acceder al servidor OPC de IRC5 de forma remota, debe permitir que Configuración de la instalación para configurar sus ajustes DCOM.

2.3.6. Kepserver y OPC Quick Client

En el campo de las comunicaciones industriales existen varios protocolos utilizados para el intercambio de información entre una gran variedad de dispositivos industriales como por ejemplo PLCs, sensores, robots, y sistemas de supervisión. Las empresas han creado protocolos propios con los que se pueden comunicar sus dispositivos, pero además de esto se ha buscado crear medios de comunicación para dispositivos de diferentes marcas comerciales.

Uno de los protocolos de comunicación más importantes y extendidos es el estándar OPC. En la actualidad, existe una gran cantidad de programas que facilitan la interconexión de dispositivos utilizando este estándar, y actúan como vínculo entre los múltiples elementos que pueden estar presentes en entornos industriales y de esta manera compartir información dentro de todos los niveles de la gestión de la automatización. Uno de estos programas es Kepserver.

La comunicación por medio de Kepserver, se establece con tres parámetros que son el canal, el dispositivo, y los ítems. Para asegurar que la conexión OPC es segura, hay que crear usuarios y grupos de usuarios que serán exclusivos para este uso. Se pueden añadir manualmente desde un usuario que tenga las credenciales necesarias.

Channel: El servidor OPC admite el uso de varios controladores de comunicaciones simultáneos. Cada protocolo o controlador utilizado en un proyecto del servidor se denomina canal. Un proyecto de servidor puede constar de muchos canales con el mismo controlador de comunicaciones o con controladores de comunicaciones únicos. Un canal actúa como el componente básico de un enlace OPC. Este grupo se utiliza para especificar las propiedades generales del canal, como los atributos de identificación y el modo de operación.

Device: Un dispositivo la representación que se da en el servidor a cada elemento presente en un canal, así un canal configurado para un protocolo puede tener múltiples elementos. Por ejemplo, si se configura un canal de comunicación con robots ABB, pueden existir varios robots (dispositivos) en esta red y cada uno debe tener una identificación propia.

Item: Un ítem es la representación que se da a las variables presentes en un dispositivo, cada dispositivo puede manejar varias variables para su funcionamiento. Por ejemplo, una variable puede ser la lectura de estado de un sensor, que es de interés conocer en un sistema de supervisión.

Para establecer la comunicación con el servidor se necesita conocer el nombre completo de acceso de los datos, la estructura es la siguiente.

$$< channel >.< device >.< item >$$

2.3.6.1. Kepserver

KEPSeverEX es una plataforma líder en conectividad industrial, que proporciona acceso al entorno industrial o a la infraestructura a través de sus más de 150 protocolos incorporados y sirviendo dicha información en OPC DA y OPC UA. KEPServerEX permite al usuario conectar, gestionar, monitorizar y controlar cualquier tipo de dispositivo a través de una única e intuitiva interface. El entorno de trabajo principal del programa Kepserver se muestra en la Figura 15.

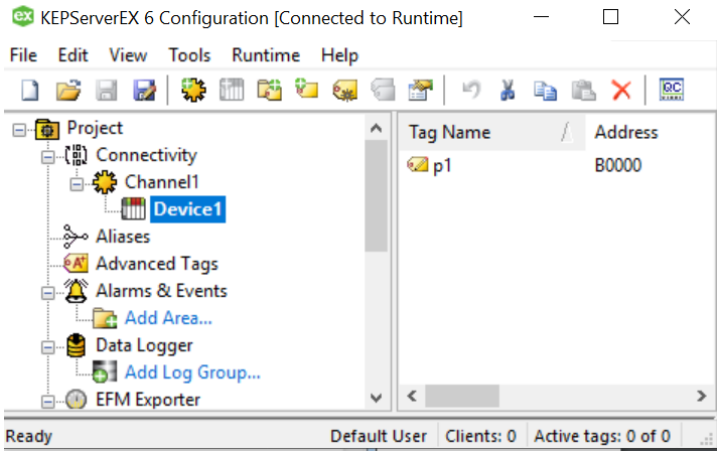

Figura 15. Interfaz KepserverEX

Además de cumplir con las funcionalidades de un OPC Sever, también es capaz de servir los datos adquiridos en otros formatos según las necesidades, ya sea por SNMP, ODBC, REST o MQTT, proporcionando al usuario una única fuente de datos sea cual sea el entorno en el que debe trabajar.

2.3.6.2. OPC Quick Client

OPC Quick Client es uno de los programas que acompaña a Kepserver, y tiene el objetivo de crear clientes que puedan tener acceso a servidores OPC e interactuar con ellos. Este programa asiste en las pruebas y el desarrollo de los servidores OPC Data Access 1.0 y 2.0. Además, admite conexiones de servidor OPC local y remoto. Las conexiones remotas se manejan a través de la interfaz DCOM del sistema operativo.

En la Figura 16 se muestra la pantalla principal del programa Quick Client, con un ejemplo de lectura de variables de un servidor OPC local.

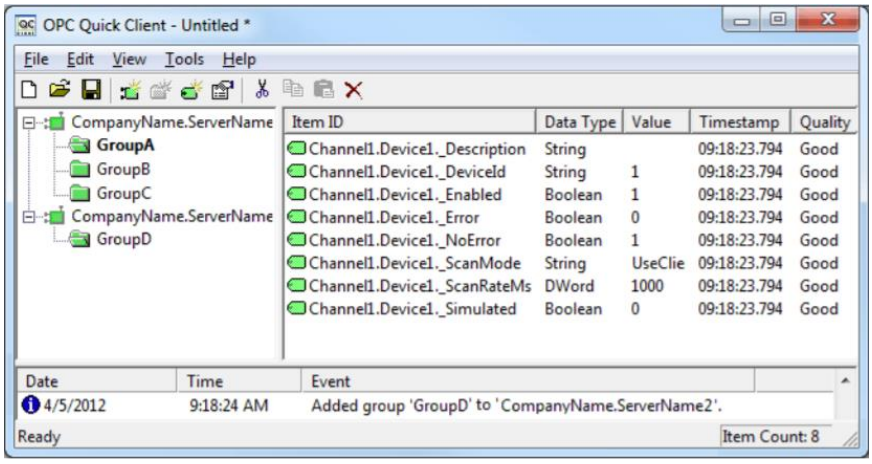

Figura 16. Interfaz OPC Quick Client

3. NÚCLEO DE LA MEMORIA

En este capítulo se detalla el desarrollo del aplicativo de control de movimiento de un robot ABB IRB140 por medio de comandos de voz, analizando los tres campos que abarca este proyecto. En primera instancia se explica la configuración del computador, y el funcionamiento del aplicativo de reconocimiento de voz con las funciones y características que este presenta. En segundo lugar, se detalla el proceso de configuración del servidor OPC, la conexión entre la controladora del robot y el computador, y la configuración del cliente OPC en Python. La última sección de este capítulo está dedicada a la programación del robot ABB IRB140 para la realización de diferentes movimientos ordenados desde la aplicación realizada en Python. Cada punto cuenta con una introducción sobre el trabajo realizado y la explicación detallada del trabajo técnico necesario para la puesta en marcha del aplicativo.

3.1. Reconocimiento de voz

Para lograr una comprensión de la producción del habla humana, se debe estudiar la anatomía del sistema vocal (la maquinaria de producción de señales del habla). Es justo decir que se debe comprender el proceso de producción del habla, antes de intentar modelar un sistema que lo entienda. Una vez que se comprende mejor este mecanismo, podemos intentar crear sistemas que reconozcan sus características y matices distintivos, reconociendo así al hablante individual. Además, dado que la evolución tiene una forma de afinar nuestra anatomía, los sistemas vocales y auditivos han evolucionado para funcionar al unísono. Una comprensión del sistema auditivo puede ayudar a hacer un mejor trabajo para captar las características de nuestra voz. Además, la evolución ya ha incorporado una capacidad de reconocimiento en nuestra percepción auditiva que debe estudiarse antes de intentar crear nuestro propio sistema de reconocimiento automático de altavoces. Este sistema de reconocimiento natural ha sido diseñado para permitirnos reconocer las voces de nuestros padres en la infancia y generalmente se desarrolla incluso antes de que empecemos a dar sentido a su contenido de habla. Por ejemplo, hay estudios que muestran que los bebés comienzan a reconocer las voces humanas de otros sonidos entre las edades de 4 y 7 meses, mucho antes de que empiecen a entender el habla. (Beigi, 2011).

En este apartado se estudia el funcionamiento del reconocimiento de voz, y su uso en el desarrollo del aplicativo para controlar el movimiento del robot ABB IRB140.

3.1.1. Funcionamiento del reconocimiento de voz

Tradicionalmente, el trabajo de procesamiento de lenguaje natural ha sido visto como un proceso de análisis que se descompone en varias etapas, lo que refleja las distinciones lingüísticas teóricas dibujadas entre sintaxis, semántica y pragmática. Puesto de una forma más simple, las oraciones de un texto se analizan primero en términos de su sintaxis; esto proporciona un orden y una estructura que son más susceptibles de análisis en términos de semántica o significado literal; y esto es seguido por una etapa de análisis pragmático por el cual se determina el significado de la expresión o el texto en contexto. Esta última etapa a menudo se considera que está relacionada con el discurso, mientras que las dos anteriores se refieren generalmente a asuntos de la oración. Este intento de correlación entre diferentes estrategias (sintaxis, semántica y pragmática) y una distinción en términos de granularidad (oración versus discurso) a veces causa cierta confusión al pensar sobre los problemas involucrados en el procesamiento del lenguaje natural; y se reconoce ampliamente que en términos reales no es tan fácil separar el procesamiento del lenguaje de forma ordenada en recuadros correspondientes a cada uno de los estratos. Sin embargo, tal separación sirve como una ayuda pedagógica útil, y también constituye la base para modelos arquitectónicos que hacen que la tarea de análisis del lenguaje natural sea más manejable desde el punto de vista de la ingeniería de software. (Indurkhya & Damerau, 2010)

No obstante, la distinción tripartita en sintaxis, semántica y pragmática solo sirve como punto de partida cuando consideramos el procesamiento de textos en lenguaje natural real. Una descomposición más precisa del proceso es útil cuando tenemos en cuenta el estado actual del arte en combinación con la necesidad de tratar con datos de lenguaje reales. En la Figura 17 se muestra las etapas de análisis del procesamiento del lenguaje natural.

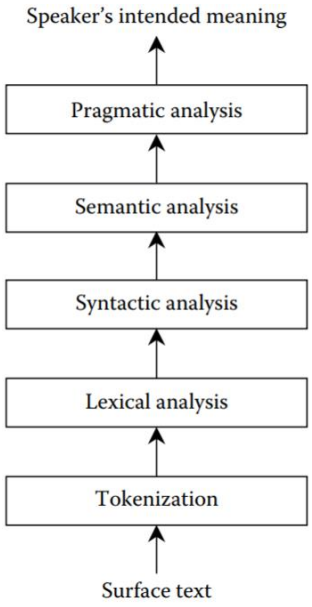

Figura 17. Etapas de análisis en el procesamiento natural de lenguaje (Indurkhya & Damerau, 2010)

Aquí se identifica la etapa de tokenización y segmentación de oraciones como un primer paso crucial. El texto en lenguaje natural generalmente no se compone de oraciones cortas, limpias, bien formadas y bien delimitadas que encontramos en los libros de texto; y para idiomas como el chino, el japonés o el tailandés, que no comparten la tokenización delimitada por espacios aparentemente fácil que podríamos creer que es una propiedad de idiomas como el inglés, la capacidad de abordar problemas de tokenización es esencial para iniciar el proceso. También tratamos el análisis léxico como un paso separado en el proceso. Hasta cierto punto, esta descomposición de grano más fino refleja nuestro estado actual de conocimiento sobre el procesamiento del lenguaje: sabemos bastante sobre técnicas generales de tokenización, análisis léxico y análisis sintáctico, y mucho menos sobre semántica y procesamiento a nivel de discurso. Pero también refleja el hecho de que lo conocido es el texto de superficie, y cualquier cosa más profunda es una abstracción representativa que es más difícil de precisar; por lo que no es tan sorprendente que tengamos técnicas mejor desarrolladas en el extremo más concreto del espectro de procesamiento.

3.1.1.1. Representación de la señal de voz

Una señal es una medida observable de un fenómeno físico. En general, describe una observación de un fenómeno físico de nivel superior en correlación con conceptos de medición de nivel inferior, como el tiempo o el espacio. Por lo tanto, una señal es un mapeo de un punto en las bases de bajo nivel, como el tiempo o el espacio, en la medición de nivel superior. Dado que el espacio y el tiempo son continuos, todos los fenómenos físicos observados en su presencia deben estar relacionados con esas

bases de manera continua. Por lo tanto, es justo decir que la mayoría de las interacciones naturales ocurren en un dominio analógico (continuo).

La señal de habla es una medida observada, realizada con respecto al paso del tiempo. Puede verse como el mapeo de la fuerza de las ondas de sonido de la voz en un determinado momento del tiempo. Es importante tener en cuenta que este valor no se relaciona únicamente con alguna onda de frecuencia específica.

Para simplificar el procesamiento de señales continuas, el conjunto infinito de valores posibles que el sistema continuo puede tomar en un intervalo debe reducirse a un conjunto finito de intervalo [a, b], a través de otro proceso de mapeo llamado muestreo. Esta acción se llama discretización y la señal recién definida, capaz de asignar este conjunto finito de puntos a una medición de nivel superior, se llama señal discreta. Por supuesto, en general, también se puede imponer una restricción similar al rango del mapeo y reducirlo a un conjunto finito, pero eso no es necesario.

Una señal de voz cambia de forma a medida que cambia el estado del tracto vocal. El sistema vocal humano es bastante dinámico y, de hecho, está diseñado para cambiar la forma de la señal en función del tiempo. Incluso dentro de un muestreo de 80 ms, sabemos que hay varias transiciones que suceden en las ondas de sonido que producen la voz, cambiando las características de la señal en el camino. Esto clasifica el sistema de producción de voz en un sistema no lineal con parámetros que cambian constantemente.

3.1.1.2. Muestreo de sonido

Dado que el reconocimiento del hablante es básicamente un proceso pasivo y solo observa la señal de audio para tomar una decisión, se considera que utiliza técnicas de procesamiento de señal en contraste con los sistemas activos como los sistemas de control que contribuyen a la dinámica del sistema en el que están involucrados. Por lo tanto, solo nos interesa un proceso de muestreo al principio y una vez que la señal está en estado muestreado, los algoritmos son independientes del mundo analógico. Por supuesto, en otras disciplinas relacionadas con el habla, este no es el caso.

El punto de partida natural es muestrear la señal de audio analógica para procesarla más tarde. Hay varias formas posibles de muestrear una señal, a saber, muestreo modulado periódico, cicle rate, multi rate, aleatorio y de ancho de pulso. En el procesamiento del habla usualmente usamos un muestreo periódico en el cual la frecuencia de muestreo (tasa de muestreo) es fija. Aunque es posible que las aplicaciones relacionadas con el habla traten con señales de baja actividad y que puedan usar técnicas de muestreo variable. Muchas técnicas de compresión con pérdida utilizan muestreo variable. Un

ejemplo es la señal codificada en MP3. Para simplificar la operación, por lo general los sistemas de reconocimiento de altavoces convierten estas representaciones en aquellas que utilizan una frecuencia de muestreo periódica como la Modulación de Código de Pulso (PCM).

Al reducir las posibilidades al muestreo periódico de la señal, se debe considerar qué período de muestreo (o frecuencia de muestreo) se debe utilizar, y para esto se debe examinar un teorema fundamental en la teoría de la información y el procesamiento de señales denominado Teorema de muestreo.

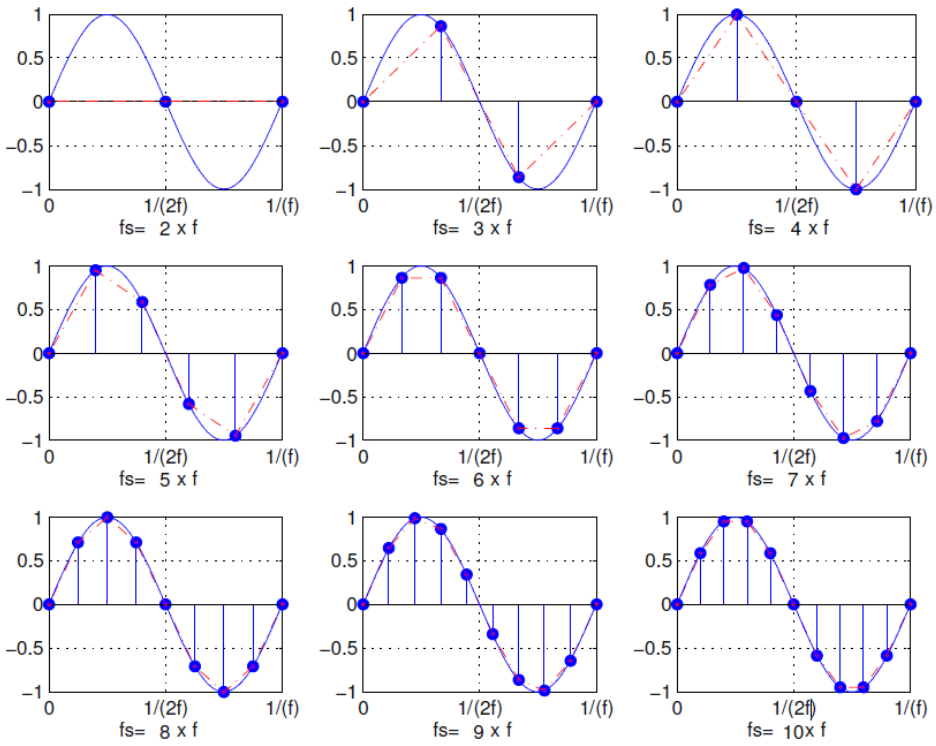

Figura 18. Muestreo de una señal sinusoidal a diferentes tasas de muestreo (Beigi, 2011)

La Figura 18 muestra el resultado del muestreo a una frecuencia fs (variable) de una señal sinusoidal con frecuencia f (fija), se puede observar que en rangos de frecuencia de muestreo bajos como por ejemplo $fs = 2f$ el resultado no describe el comportamiento de la señal sinusoidal, en este caso el muestreo mostraría una señal constante. Al ir aumentando la frecuencia de muestreo, como por ejemplo $fs = 3f$, la señal empieza a parecerse a una señal diente de sierra. Para una frecuencia $fs = 4f$, el muestreo indica una señal triangular, y para una frecuencia $fs = 6f$ la señal se asemeja a una señal cuadrada. El comportamiento sinusoidal se empieza a notar a medida que la frecuencia de muestreo aumenta, es por esto que es necesario obtener una frecuencia de muestreo adecuada para lograr representar todas las características de la señal que se desea leer.

El teorema de muestro dice que si una función $h(t)$ no contiene frecuencias superiores a fc ciclos por segundo, la función está completamente determinada al dar en su ordenada una serie de puntos separados a $\frac{1}{2fc}$.

La afirmación de que la función no tiene frecuencias superiores a $fc\ Hz$ equivale a decir que la función está limitada a la banda fc (ancho de banda) desde la parte superior, o en términos matemáticos, $H\ (f)\ =\ 0\ \forall\ |\ f\ |\ \geq\ fc$, donde $H\ (f)$ es la representación espectral de $h\ (t)$. fc se conoce como la Frecuencia Crítica de Nyquist y establece los límites de lo que se conoce más ampliamente como el ancho de banda de la señal. Por lo tanto, la frecuencia de muestreo de la señal debe ser $fs\ \geq\ 2\ fc$.

3.1.1.3. Forma de onda de voz

Dados las diferentes características del teorema de muestreo, se puede muestrear la salida de un micrófono analógico para producir la llamada señal de onda de la voz. Usando la cuantización de amplitud junto con el muestreo en el dominio de la frecuencia, tendremos una representación de la señal de voz llamada forma de onda de voz. La Figura 19 es un gráfico de dicha forma de onda muestreada a 22050 Hz.

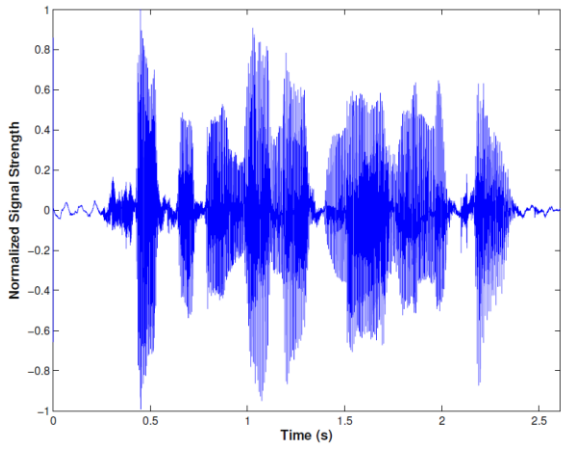

Figura 19. Forma de onda de voz muestreada a fs = 22050 Hz

En el procesamiento de audio se utiliza diagramas como los de la forma de onda de voz para encontrar características únicas que ayuden a distinguir entre diferentes sonidos. Para encontrar patrones en estas señales se utilizan métodos como las redes neuronales que se explican más adelante.

3.1.1.4. El espectrograma

El análisis de señales de tiempo en el dominio de la frecuencia nos permite descubrir aspectos de la señal que serían muy difíciles o imposibles de observar en su representación temporal.

Una señal que parece ser sinusoidal es mostrada en la Figura 20 (a), sin embargo, en el dominio de la frecuencia se puede observar que existen tres componentes de esta señal como se muestra en la Figura 20 (b). Estas componentes quedan enmascaradas en el dominio del tiempo por una señal de gran amplitud.

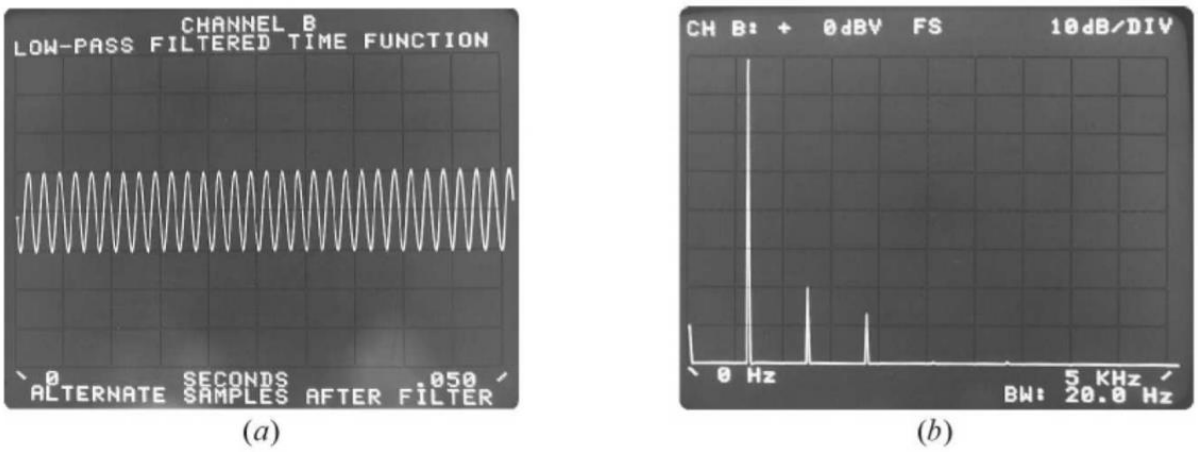

Figura 20. Señal sinusoidal en a) Dominio del tiempo. b) Dominio de la frecuencia

El teorema de Fourier establece que toda señal en el dominio del tiempo puede ser llevada al dominio de la frecuencia utilizando la transformada de Fourier. Por ejemplo, una señal que parece ser sinusoidal como la que se muestra en la Figura 21, en el dominio de la frecuencia presenta tres componentes de esta señal. Estas componentes quedan enmascaradas en el dominio del tiempo por una señal de gran amplitud.

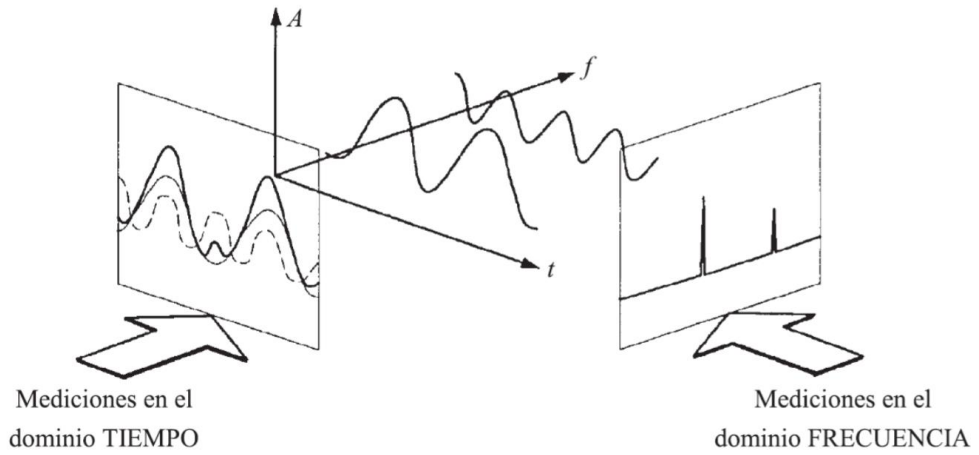

Figura 21. Tiempo versus frecuencia

Un espectrograma es una representación tridimensional del contenido espectral de la señal de voz. Representa el poder de los diferentes componentes espectrales de cada instancia del habla. Es el

resultado de calcular el espectro de cada trama de una señal. El resultado es una gráfica tridimensional que representa la energía del contenido frecuencial de la señal según va variando a lo largo del tiempo.

Se usa, por ejemplo, para identificar sonidos fonéticos y procesado del habla. El instrumento que genera espectrogramas es llamado espectrómetro. También existe software que puede crear espectrogramas. El espectrograma es una herramienta básica de representación que se utiliza para el análisis de las señales eléctricas, de comunicaciones, y cualquier señal audiovisual en su contenido frecuencial.

Es una representación en tres dimensiones, temporal, frecuencial y amplitud de la distribución de energía de una señal. La representación del espectro de una señal en el dominio frecuencial puede ayudar a entender mejor su contenido, que con una representación en el dominio temporal. El espectrograma se puede interpretar como una proyección en dos dimensiones de una sucesión de Transformadas de Fourier de tramas consecutivas, donde la energía y el contenido frecuencial de la señal va variando a lo largo del tiempo.

Por ejemplo, en la Figura 22 se muestra la forma de una señal de audio de voz en el dominio del tiempo en la parte superior, y en el dominio de la frecuencia en la parte inferior.

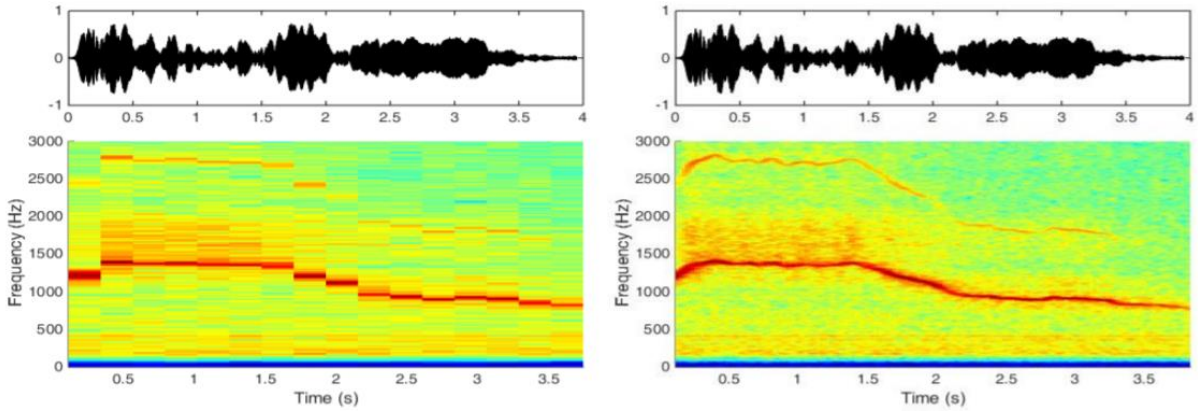

Figura 22. Forma de onda y espectrograma de un audio de voz (Beigi, 2011)

En la parte izquierda de la figura, se observa el resultado de una grabación con un micrófono sin usar ningún tipo de filtro y con mucho ruido ambiental. Y en la parte derecha se muestra el mismo audio reduciendo el ruido de la señal por medio de filtros. Como se puede apreciar, en el dominio del tiempo las señales son muy similares, y es imposible identificar diferencia entre ellas pese a ser diferentes, pero en el dominio de la frecuencia estos cambios son evidentes. El espectrograma es importante porque muestra características de la señal de audio que en un análisis del tiempo serian imposibles

de observar, al usar esta herramienta se puede obtener más información sobre la señal y mejorar su posterior procesamiento por ejemplo en el reconocimiento de voz.

3.1.1.5. Procesamiento de señales

Existen técnicas que mejoran el muestreo de la señal de voz, y si bien hay cierta libertad para elegir el orden en el que se generan las muestras a partir de la señal analógica, existe evidencia de que un cierto orden daría mejores resultados. Las siguientes tres figuras (Figura 23, Figura 24, Figura 25) muestran tres de estas combinaciones posibles. Cuando diseñamos nuestro proceso de muestreo.

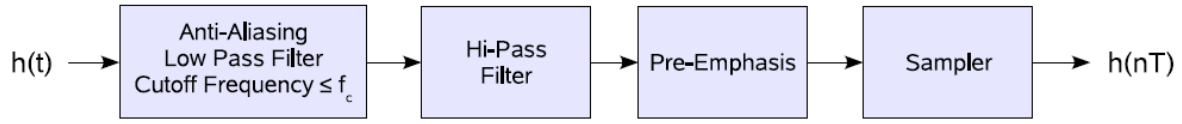

Figura 23. Diagrama de bloques de un proceso típico de muestreo de voz - Mejor Alternativa (Beigi, 2011)

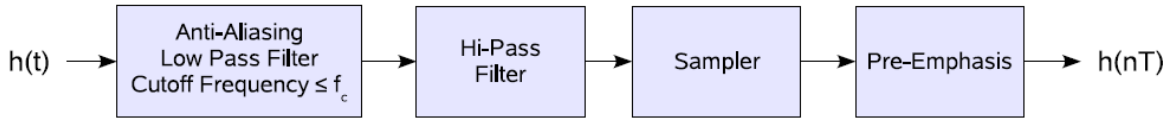

Figura 24. Diagrama de bloques de un proceso típico de muestreo de voz - Opción alternativa (Beigi, 2011)

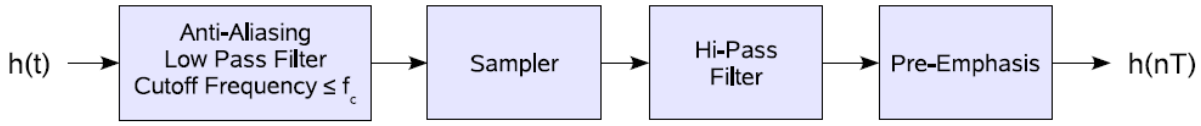

Figura 25. Diagrama de bloques de un proceso típico de muestreo de voz – Opción alternativa (Beigi, 2011)

En la Figura 22. Forma de onda y espectrograma de un audio de voz se puede apreciar el resultado del uso de filtros en el procesamiento de una señal. El uso de una grabación de audio sin ningún tipo de filtro para realizar reconocimiento de voz presenta grandes inconvenientes ya que existe mucho ruido que distorsiona la información que se pueda extraer. A usar filtros, se busca el dejar únicamente las características propias de la voz, esta es la importancia del uso de los filtros expuestos a continuación.

Anti- aliasing

El aliasing son componentes de alta frecuencia que alteran el espectro de sonido en una grabación realizada con el micrófono. Para evitar el aliasing, la señal analógica (salida del micrófono) debe pasar a través de un filtro anti-aliasing que es realmente un filtro de paso bajo con una frecuencia de corte

que es menor que la frecuencia de plegado. Este es uno de esos bloques que tiene una ubicación obligatoria. Por supuesto, tenga en cuenta que la mayor parte del proceso de muestreo generalmente se realiza mediante un software independiente contenido en el controlador del sistema de sonido del sistema de micrófono que se está utilizando. Esto es cierto tanto para interfaces basadas en computadora como en telefonía.

Hi-Pass Filtering

En el mejor escenario, hay un filtro de paso alto inmediatamente después del filtro anti-alias de paso bajo. La razón para hacer este filtrado de paso alto es que todos los micrófonos no son creados iguales. Tendrán diferentes componentes de corriente directa (DC) que no están necesariamente vinculados al contenido de voz que proporcionan sus señales. Un filtro de paso alto con una frecuencia de corte baja eliminará este offset de CC y permitirá una menor variabilidad y dependencia del micrófono en diferentes plataformas y configuraciones. Si el software de muestreo que se está utilizando no proporciona este filtro de paso alto, puede colocarse justo después de la muestra. Sin embargo, es mejor mantenerlo al comienzo del proceso y actuar sobre la señal analógica antes de ingresar al muestreador.

Pre-Emphasis

La densidad espectral de potencia de la forma de onda del habla presenta una fuerte caída de la potencia de la señal en las frecuencias altas. Aproximadamente el 80% de la potencia está contenida dentro de componentes de frecuencia por debajo de 1000 Hz. De 1 kHz a 8 kHz, la potencia cae a una velocidad de aproximadamente −12 dB / Octave2 y es casi despreciable en frecuencias superiores a 8 kHz.

Se observa que el oído humano puede reconocer fácilmente estas regiones de baja energía. Dado que desea diseñar un sistema de reconocimiento automático, se debemos hacer algo similar, para poder utilizar las funciones importantes integradas en frecuencias más altas. Esto se puede lograr a través del pre-énfasis. Esta técnica logra que la potencia absoluta para cada rango de frecuencia se reduzca, pero la potencia relativa este mejor distribuida a lo largo de las diferentes frecuencias. Con esta etapa se logra mejorar la señal en las zonas de energía bajas para poder utilizarlas en el análisis.

Las tres técnicas antes mencionadas mejoran la calidad de la señal de audio que será luego muestreada para generar el espectro de sonido de la voz, con esto se obtiene más información para poder procesarla, pero se descarta la información que causa distorsión y no es necesaria.

3.1.1.6. Modelos ocultos de Markov

Un modelo oculto de Márkov o HMM (por sus siglas del inglés, Hidden Markov Model) es un modelo estadístico en el que se asume que el sistema a modelar es un proceso de Márkov de parámetros desconocidos. El objetivo es determinar los parámetros desconocidos (u ocultos, de ahí el nombre) de dicha cadena a partir de los parámetros observables. Los parámetros extraídos se pueden emplear para llevar a cabo sucesivos análisis.

Los modelos ocultos de densidad continua de Markov son un componente esencial de los sistemas modernos para el reconocimiento automático de voz (ASR). Estos modelos asignan probabilidades a las secuencias de vectores de características acústicas extraídas por el procesamiento de la señal de las formas de onda del habla.

En un modelo de Márkov normal, el estado es visible directamente para el observador, por lo que las probabilidades de transición entre estados son los únicos parámetros. En un modelo oculto de Márkov, el estado no es visible directamente, sino que sólo lo son las variables influidas por el estado. Cada estado tiene una distribución de probabilidad sobre los posibles símbolos de salida. Consecuentemente, la secuencia de símbolos generada por un HMM proporciona cierta información acerca de la secuencia de estados.

Los modelos ocultos de Márkov son especialmente aplicados a reconocimiento de formas temporales, como reconocimiento del habla, de escritura manual, de gestos, etiquetado gramatical o en bioinformática. En el reconocimiento de voz se emplea para modelar una frase completa, una palabra, un fonema o trifonema en el modelo acústico. Por ejemplo, la palabra "gato" puede estar formada por dos modelos acústicos de Markov para los dos trifonemas que la componen /gat/ y /ato. Y lo que se busca es encontrar todas las posibles combinaciones que estos sonidos pueden crear.

El estudio de los modelos ocultos de Markov representa un tema de gran amplitud que no será analizado en el presente trabajo, pero es necesario mencionarlo debido al gran aporte que brinda en todos los sistemas de reconocimiento de voz actuales.

3.1.1.7. Redes neuronales

Los modelos de redes neuronales, se han estudiado durante muchos años con la esperanza de que las máquinas hechas por el hombre puedan emular la capacidad superior de aprendizaje y reconocimiento del cerebro humano. Similares redes en el cerebro humano, hacen posible reconocer el complejo patrón del habla en humanos. En contraste con las computadoras, que procesan

información secuencialmente, las redes neuronales emplean enormes redes paralelas de muchos elementos interconectados llamados neuronas. Las redes neuronales se han utilizado en muchas aplicaciones diferentes, como el control de aprendizaje y adaptación, el reconocimiento de patrones, el procesamiento de imágenes, el reconocimiento de firmas, el procesamiento de señales y el reconocimiento de voz, etc.

Una neurona es la unidad computacional más elemental en una red neuronal que suma un número de entradas ponderadas y pasa el resultado a través de una función de activación no lineal. Las redes neuronales multicapa consisten en un gran número de neuronas. Antes de poder utilizar una red neuronal para cualquier propósito, los pesos que conectan las entradas a las neuronas y los parámetros de las funciones de activación de las neuronas deben ajustarse de modo que las salidas de la red coincidan con los valores deseados para conjuntos específicos de entradas. Los métodos utilizados para ajustar estos pesos y parámetros generalmente se conocen como algoritmos de aprendizaje. En la Figura 26 se muestra la estructura de un perceptrón y una red neuronal simple.

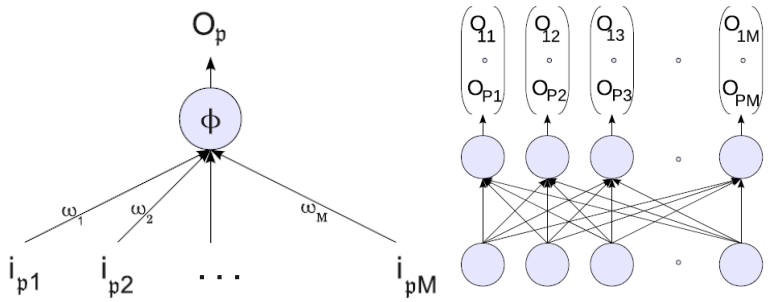

Figura 26. Perceptrón y una red neuronal

Las arquitecturas de diferentes redes neuronales que se utilizan para resolver problemas varían enormemente. El diseño de la arquitectura es bastante importante y depende de la naturaleza del problema. Varios tipos diferentes de redes neuronales han sido probados para realizar reconocimiento de voz. Se han utilizado diferentes arquitecturas y tipos de neuronas, como la arquitectura de red neuronal con retardo de tiempo (TDNN, por sus siglas en inglés) que captura la dinámica inherente del habla como una señal variable en el tiempo. Los TDNN se han combinado con éxito con los modelos ocultos de Markov (HMM) en el reconocimiento de voz.

Al igual que con los modelos ocultos de Markov, el estudio de las redes neuronales y su funcionamiento son temas de gran extensión en los que no se profundizara en este trabajo, pero es

importante mencionar que son las tecnologías más usadas en los actuales sistemas de reconocimiento de voz.

3.1.1.8. Modelo del reconocimiento de voz

Los sistemas modernos de reconocimiento de voz se han construido invariablemente sobre la base de los principios estadísticos, como fue pionero en el trabajo de Baker (1975) y Jelinek (1976) y se expuso en detalle en Huang et al. (2001); Deng y O'Shaughnessey (2003). Un modelo matemático de canal de fuente o un tipo de modelo estadístico generativo se usa a menudo para formular problemas de reconocimiento de voz. Como se ilustra en la Figura 27, la mente del hablante decide la secuencia de palabras de origen W que se entrega a través de su generador de texto. La fuente pasa a través de un canal de comunicación ruidoso que consiste en el aparato vocal del hablante para producir la forma de onda del habla y el componente de procesamiento de la señal de voz del reconocedor de voz. Finalmente, el decodificador de voz apunta a decodificar la señal acústica X en una secuencia de palabras W', que en casos ideales está cerca de la secuencia de palabras original W.

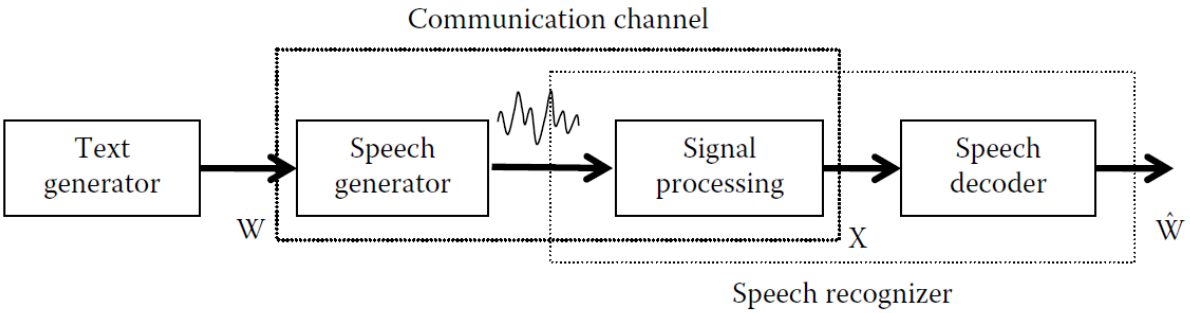

Figura 27. Modelo típico de reconocimiento de voz (Clark, Fox, & Lappin, 2010)

Un sistema de reconocimiento de voz típico y práctico consiste en componentes básicos que se muestran en la Figura 28. Las aplicaciones interactúan con el decodificador para obtener resultados de reconocimiento que pueden utilizarse para adaptar otros componentes del sistema. Los modelos acústicos incluyen la representación del conocimiento sobre acústica, fonética, micrófono y variabilidad del entorno, diferencias de género y dialecto entre los oradores, etc. Los modelos de lenguaje se refieren al conocimiento de un sistema de lo que constituye una palabra posible, qué palabras pueden coexistir, y en que secuencia La semántica y las funciones relacionadas con una operación que un usuario puede desear realizar también pueden ser necesarias para el modelo lingüístico. Existen muchas incertidumbres en estas áreas, asociadas con las características del hablante, el estilo y la velocidad del habla, el reconocimiento de los segmentos básicos del habla, las palabras posibles, las palabras probables, las palabras desconocidas, la variación gramatical, la

interferencia de ruido, los acentos no nativos y la puntuación de confianza de los resultados. Un sistema exitoso de reconocimiento de voz debe enfrentar todas estas incertidumbres. La incertidumbre acústica de los diferentes acentos y estilos de habla de los hablantes individuales se ve agravada por la complejidad léxica y gramatical y las variaciones del lenguaje hablado, todas representadas en el modelo del lenguaje.

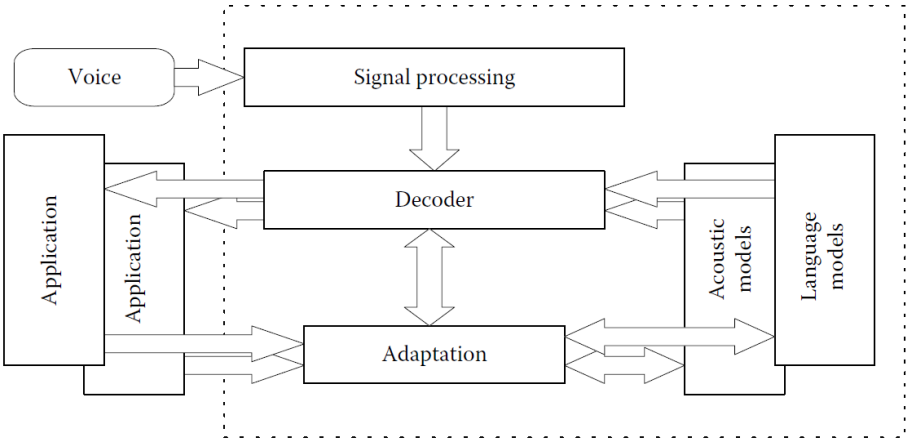

Figura 28. Arquitectura básica de un sistema de reconocimiento de voz (Clark, Fox, & Lappin, 2010)

Como se muestra en la Figura 28, la señal de voz se procesa en el módulo de procesamiento de señales que extrae los vectores de características salientes para el decodificador. El decodificador utiliza modelos acústicos y de lenguaje para generar la secuencia de palabras que tiene la probabilidad posterior máxima para los vectores de entidades de entrada. También puede proporcionar la información necesaria para que el componente de adaptación modifique los modelos acústicos o de lenguaje para obtener un mejor rendimiento.

3.1.1.9. Aplicaciones del reconocimiento de voz

El impacto final del reconocimiento de voz depende de si se puede integrar completamente las tecnologías habilitantes con las aplicaciones. La forma de integrar efectivamente el habla en las aplicaciones a menudo depende de la naturaleza de la interfaz de usuario y la aplicación. Al analizar algunos principios y pautas generales para desarrollar aplicaciones de lenguaje hablado, se debe observar detenidamente el diseño de la interfaz de usuario. Una interfaz de usuario bien diseñada implica considerar cuidadosamente el grupo particular de usuarios de la aplicación y entregar una aplicación que funcione de manera efectiva y eficiente. Como norma general, uno debe asegurarse de que la interfaz coincida con la forma en que los usuarios desean realizar una tarea. También es necesario utilizar la modalidad más adecuada en el momento adecuado para ayudar a los usuarios a lograr sus objetivos. Un desafío único en las aplicaciones de reconocimiento de voz es que el

reconocimiento de voz (así como la comprensión) es imperfecto. Además, el comando hablado puede ser ambiguo, por lo que es necesaria una estrategia de diálogo para aclarar el objetivo del hablante. Siempre hay errores con los que uno tiene que lidiar. Es fundamental que las aplicaciones empleen las técnicas de manejo interactivo de errores necesarias para minimizar el impacto de estos errores. Por lo tanto, los desarrolladores de aplicaciones deben comprender completamente las fortalezas y debilidades de las tecnologías de voz subyacentes e identificar el lugar adecuado para usar el reconocimiento de voz y comprender la tecnología de manera efectiva. (Clark, Fox, & Lappin, 2010)

3.1.2. Configuración y requisitos Windows 10

El aplicativo de reconocimiento de voz será realizado en un mini computador que cuenta con una arquitectura de 32 bits, por lo que todos los programas, librerías, y demás archivos necesarios para el funcionamiento deberán ser compatibles con esta arquitectura. La versión utilizada de Python será la más actual que es Python 3.7.3, por lo que igual que antes se debe ver que todos los programas complementarios necesarios sean compatibles con esta versión.

3.1.2.1. Python 3.7.3

Como se explicó en capítulos anteriores, el lenguaje de programación elegido para este proyecto es Python por que presenta una gran cantidad de ventajas entre las que figuran ser gratuito y multiplataforma. Lo primero que se debe hacer es descargar la versión más actualizada de este programa que se puede encontrar en la página oficial. El link utilizado se muestra a continuación.

$$https://www.python.org/downloads/$$

En esta dirección web se debe buscar la versión más actual del software y seleccionar la opción descargar.

Figura 29. Descarga de Python 3.7.3

Una vez descargado el archivo que tiene un peso de aproximadamente 24.4 Mb se debe presionar el botón derecho del ratón sobre el para desplegar el menú de opciones, y en este se debe seleccionar ejecutar como administrador para que se concedan todos los permisos de instalación.

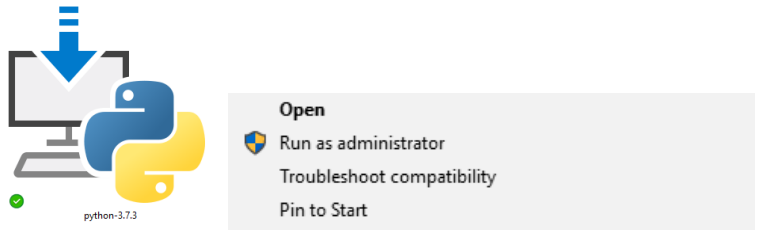

Figura 30. Instalador de Python

Una vez ejecutado el archivo descargado se realiza la instalación con todos los parámetros por defecto. Al finalizar, es necesario reiniciar el sistema operativo y configurar las rutas de ejecución del programa para poder ejecutar comandos desde la consola de Windows. Para poder configurar las rutas se debe verificar la dirección en la que se encuentra instalado y para esto se debe presionar el clic derecho sobre el programa y seleccionar la opción abrir ubicación del programa.

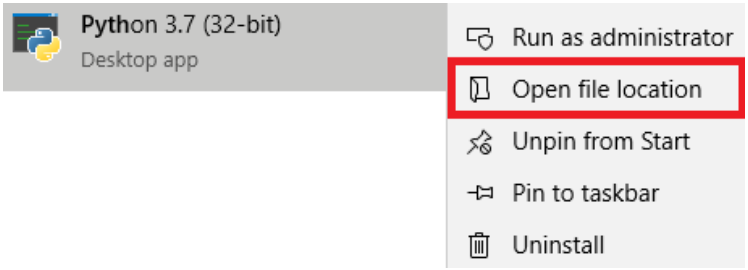

Figura 31. Dirección de instalación de Python

Una vez se encuentra el directorio donde está instalado Python, es necesario copiar esa ruta y guardarla para el siguiente proceso. En el explorador de archivos de Windows, se debe presionar el clic derecho del ratón sobre “MI Equipo” y seleccionar propiedades. Una vez abiertas las opciones del sistema se debe seleccionar los ajustes avanzados del sistema como se observa en la Figura 32.

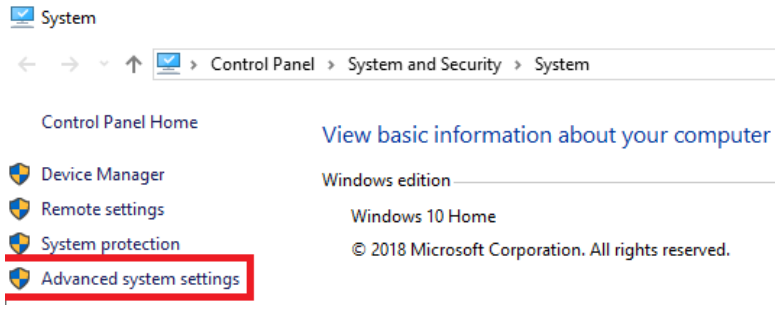

Figura 32. Opciones del sistema Windows 10

Con esto se abre la ventana de propiedades del sistema en la cual hay que seleccionar la pestaña avanzado, y seleccionar la opción variable de ambiente como en la Figura 33.

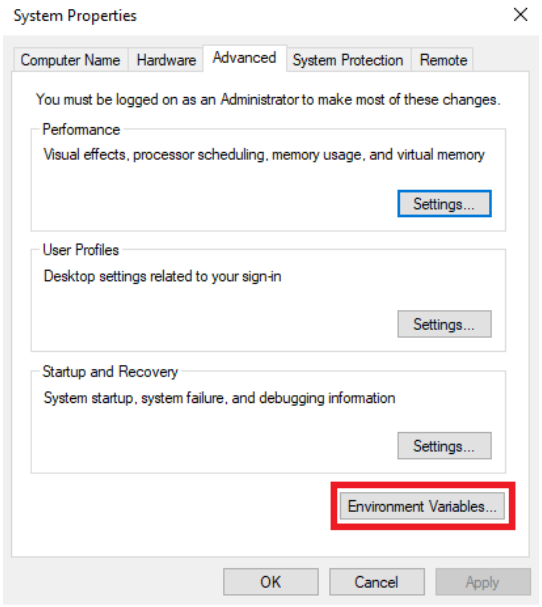

Figura 33. Propiedades del sistema Windows 10

En la ventana de variables de ambiente, se debe seleccionar la variable Path y presionar en el botón editar lo cual da acceso a un menú como esta en la Figura 34.

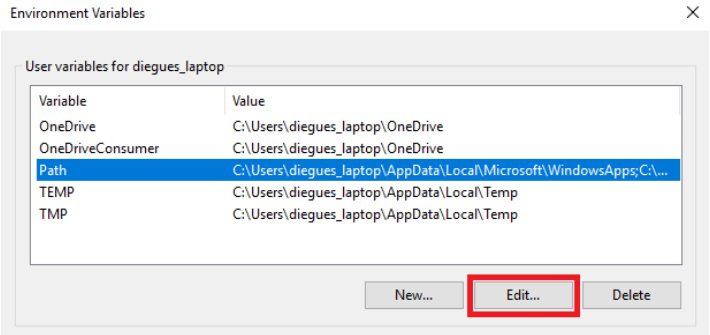

Figura 34. Variables de ambiente

En la opción de edición de variables de ambiente se debe crear una nueva y colocar la dirección donde se encuentra instalado el programa Python como se muestra en la Figura 35, con eso ahora se puede ejecutar programas de Python desde la consola de Windows

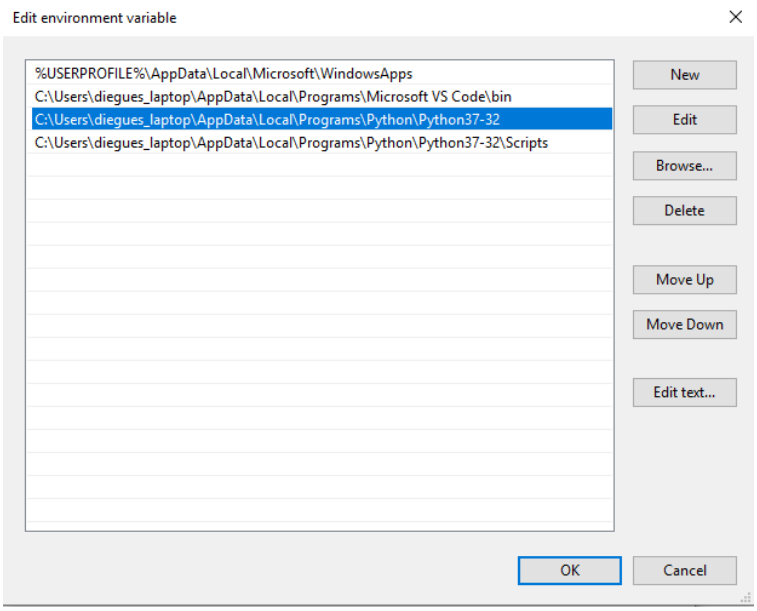

Figura 35. Editar una variable de ambiente

Una vez completado el proceso anterior se dispone de una instalación lista de Python para ejecutar programas escritos en este lenguaje de programación, y se continúa preparando el sistema para poder realizar reconocimiento de voz.

3.1.2.2. Librería pip

El módulo Pip de Python es el instalador de paquetes de este programa. Se puede usar pip para instalar paquetes desde el Índice de Paquetes de Python y otros índices. Es necesario ejecutar el comando desde la consola de Windows. En muchas instalaciones de Python la librería pip viene instalada por defecto, pero en ciertos casos esta no funciona correctamente por lo que es necesario instalarla manualmente. Para realizar la instalación se puede usar el siguiente comando.

$$>> python\ get - pip.py$$

Luego de instalar la librería pip, es necesario actualizarla para que se encuentre en su versión más actual.

$$\gg python - m\ pip\ install - U\ pip$$

Una vez instalada y actualizada la librería pip, esta permite instalar nuevos módulos en Python tan solo ejecutando comandos desde la consola de Windows. Los módulos que se instalen deben estar indexados a los paquetes de Python o en la siguiente página web.

$$https://pypi.org$$

3.1.2.3. Librería PyAudio

La librería PyAudio es necesaria para la realización del proyecto ya que es la que permite a Python el interactuar con entradas y salidas de audio. PyAudio proporciona enlaces Python para PortAudio, la biblioteca de entradas y salidas de audio multiplataforma. Con PyAudio, se puede usar Python fácilmente para reproducir y grabar audio en una variedad de plataformas. PyAudio está inspirado en:

- pyPortAudio / fastaudio: enlaces de Python para la API de PortAudio v18.
- tkSnack: kit de herramientas de sonido multiplataforma para Tcl / Tk y Python

Para instalar PyAudio se debe ejecutar el comando mostrado en la Figura 36 en la consola de Windows.

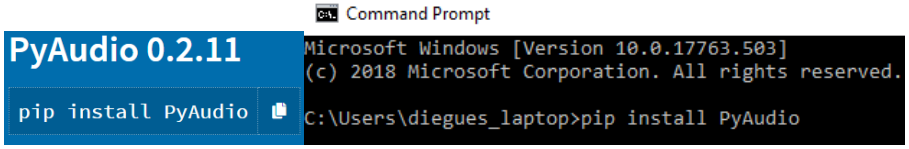

Figura 36. Instalación PyAudio desde internet

En ciertos casos la función de instalación de PyAudio no se puede ejecutar desde la consola de Windows por lo que es necesario instalar los archivos Wheel manualmente, para esto se debe descargar el archivo correspondiente a la versión de Python y Windows que se encuentra en el siguiente link.

$$https://pypi.org/project/PyAudio/\#files$$

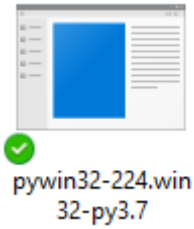

Figura 37. Versión 32 bits de PyAudio

En la Figura 37 se puede observar el nombre del archivo descargado para la versión de Windows utilizado en el presente proyecto. Este archivo debe ubicarse en un directorio con una dirección conocida y ejecutar el comando mostrado en la Figura 38.

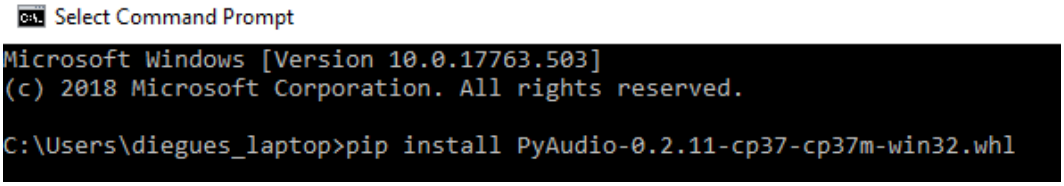

Figura 38. Instalación PyAudio desde un archivo Wheel

El funcionamiento de esta librería se explica a continuación con algunas funciones, las cuales sirven para establecer las entradas y salidas de audio del computador en Python.

- Para usar PyAudio, primero se crea una instancia de PyAudio usando pyaudio. PyAudio (), con esto se configura el sistema portaudio.
- Para grabar o reproducir audio, de debe abrir una transmisión en el dispositivo deseado con los parámetros de audio deseados utilizando pyaudio.PyAudio.open (). Esto configura un pyaudio.Stream para reproducir o grabar audio.
- Se reproduce audio escribiendo datos de audio en la transmisión usando pyaudio.Stream.write (), o lea datos de audio de la transmisión usando pyaudio.Stream.read ().
- Se usa pyaudio.Stream.stop_stream () para pausar la reproducción / grabación, y pyaudio.Stream.close () para terminar la transmisión.
- Finalmente, finalizar la sesión de portaudio utilizando pyaudio.PyAudio.terminate ()

Una vez instalado este módulo, un programa creado en Python es capaz de tener acceso al micrófono del sistema y con esto poder realizar grabaciones de sonido. Estas grabaciones son los comandos de voz que luego van a ser procesador por algoritmos de reconocimiento de voz.

3.1.2.4. Librería SpechRecognition

Uno de los puntos principales del proyecto es el reconocimiento de voz, ya que con este se obtienen los comandos para el movimiento del robot. Existen múltiples métodos y algoritmos de reconocimiento de voz que usan varias técnicas de inteligencia artificial como son el aprendizaje, las redes neuronales, las redes neuronales multicapa llamadas deep learning, entre otras.

La librería SpeechRecognition es una biblioteca para realizar el reconocimiento de voz, con soporte para varios motores y API, en línea y fuera de línea. Alos motores que soporta esta librería de reconocimiento de voz con soporte de API se detallan a continuación:

- CMU Sphinx (trabaja sin conexión)
- Reconocimiento de voz de Google
- API de Google Cloud Speech
- Wit.ai
- Reconocimiento de voz de Microsoft Bing
- API de Houndify
- IBM Speech to Text
- Snowboy Hotword Detection (funciona sin conexión)

De todos estos motores de reconocimiento de voz, se ha decidido usar el de Google ya que es uno de los que más integración tiene con otros servicios y está en constante desarrollo y mejora. Además, tiene soporte en varios idiomas y múltiples funciones para mejorar la calidad del audio obtenido por medio del micrófono por filtros.

A continuación, se detallan los requerimientos que debe cumplir el sistema para que la librería SpeechRecognition pueda funcionar:

Para utilizar toda la funcionalidad de la biblioteca, debe tener:

- Python 2.6, 2.7 o 3.3+ (requerido).
- PyAudio 0.2.11+ (requerido si se necesita usar una entrada de micrófono).
- Google API Client Library para Python (solo se requiere si necesita usar la API de Google Cloud Speech, recognizer_instance.recognize_google_cloud).
- Codificador FLAC (requerido solo si el sistema no está basado en x86 Windows / Linux / OS X).

PyAudio es obligatorio si desea utilizar la entrada de micrófono. PyAudio versión 0.2.11+ es necesaria, ya que las versiones anteriores tienen errores conocidos de administración de memoria al grabar desde micrófonos en ciertas situaciones. Si no está instalado, todo en la biblioteca seguirá funcionando, excepto que intentar crear una instancia de un objeto Micrófono generará un AttributeError.

Para instalar la librería SpeechRecognition, se puede ejecutar el comando mostrado en la Figura 39 en la consola de Windows, con esto ya se puede empezar a utilizar las funciones.

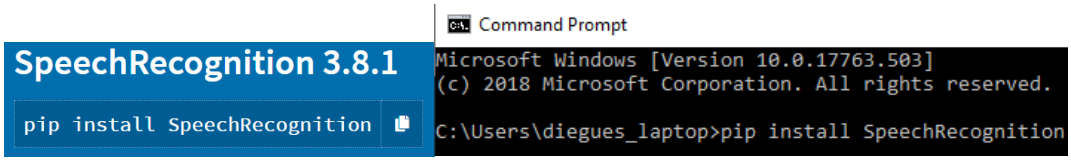

Figura 39. Instalación SpeechRecognition desde internet

Los detalles del funcionamiento de la librería SpeechRecognition se explican en el desarrollo de las funciones del aplicativo de reconocimiento de voz que se encuentra en la siguiente sección.

3.1.3. Configuración del uso de la API de Google en Python

En el funcionamiento de una aplicación de reconocimiento de voz, la primera etapa es el micrófono que es el que digitaliza el sonido de la voz para luego procesarlo y generar texto, el cual va a ser interpretado por el algoritmo de procesamiento de lenguaje natural, y generar alguna acción previamente especificada. Si es necesario el sistema dará una respuesta que será procesada por software de texto a voz y reproducida por un altavoz. La Figura 40 presenta un diagrama del funcionamiento típico de una aplicación de reconocimiento de voz.

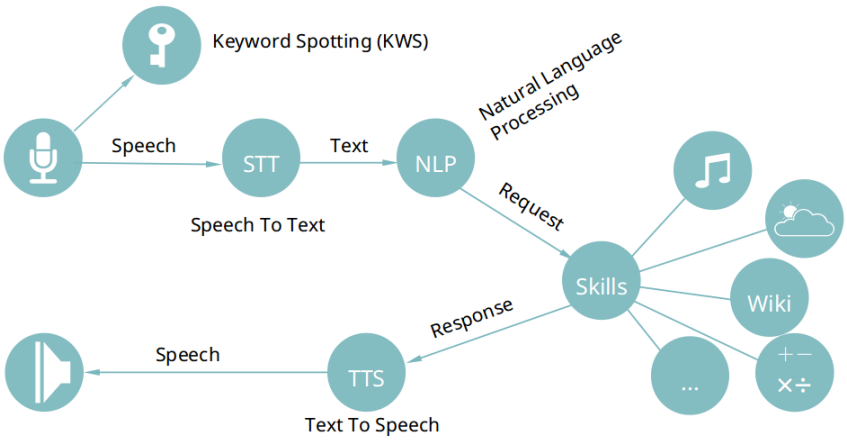

Figura 40. Funcionamiento de un aplicativo de reconocimiento de voz

En el proyecto se utiliza la API de reconocimiento de voz de Google, por lo que el archivo de audio que es obtenido con el micrófono y procesado en él computador se sube a los servidores de Google Cloud por medio de internet, y es el servidor que envía de vuelta una respuesta al aplicativo. Para poder configurar esta comunicación e interpretar la información enviada y recibida es necesario la configuración de ciertos parámetros descritos a continuación.

Para importar la librería de reconocimiento de voz a Python se usa el siguiente comando.

```
import speech_recognition as sr
```

Se crea una clase que es la encargada de grabar audio por medio del micrófono.

```
def recognize_speech_from_mic(recognizer, microphone):
```

Esta clase graba el sonido obtenido desde el micrófono y retorna un diccionario con tres keys:

- "success": es un valor booleano que indica si la respuesta de la API fue o no exitosa
- "error": `None` si no hay error, o un string que contiene un mensaje de error si no se pudo acceder a la API o si la voz no fue reconocida
- "transcription": `None` si la voz no se puede transcribir, o un string que contiene el texto transcrito

Se crea una función de lectura, donde se ajusta la sensibilidad del micrófono y se determina la variable audio como una función de reconocimiento.

```
    with microphone as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)
```

Se crea las instancias recognizer y microphone, encargadas del uso del micrófono y del llamado de la función de reconocimiento de voz.

```
recognizer = sr.Recognizer()
microphone = sr.Microphone()
```

Se configura el uso de la API de Google para el reconocimiento de voz.

```
    try:
        response["transcription"] = recognizer.recognize_google(audio)
    except sr.RequestError:
        # API was unreachable or unresponsive
        response["success"] = False
        response["error"] = "API unavailable"
    except sr.UnknownValueError:
        # speech was unintelligible
        response["error"] = "Unable to recognize speech"

    return response
```

Se establece la función reconocer_voz, encargada de grabar el audio, y recibir la transcripción de la API de Google

```
def reconocer_voz():
    guess = recognize_speech_from_mic(recognizer, microphone)
    print("---------------------")
    print("Usted ha dicho:")
    respuesta=str(guess["transcription"])
```

```
    print(respuesta)
    print("--------------------")
    return respuesta
```

Al usar la función reconocer voz, esta hace que el micrófono empiece a grabar el sonido hasta que se presente una pausa, y si se identifica las palabras que el usuario ha hablado, devuelve una transcripción de lo grabado. Generalmente el tiempo de respuesta del servidor es de 5 segundos.

3.1.4. Entrenamiento del reconocimiento de voz

La función de reconocimiento de voz activa el micrófono y empieza a grabar la voz que está escuchando hasta que se presenta una pausa, o un tiempo en el que no escucha ningún sonido. Luego carga este archivo de audio al servidor y devuelve un grupo de información en el que se presenta el audio transcrito. Dado que el idioma que se está usando es el inglés, y que la voz difiere de persona a persona, y que además existe ruido ambiental que distorsiona la grabación es necesario el observar la respuesta de la función de reconocimiento de voz para observar que el texto dicho concuerde con la transcripción.

Al realizar pruebas y verificar la respuesta del aplicativo se nota que existen comandos que, al ser pronunciados la respuesta dada por la API de Google, difiere de lo dicho. Este es el caso de la palabra clave de inicio de la aplicación “Ok Robot” la cual da muchas respuestas diferentes entre las que se incluye por ejemplo “Ok Robert”. Es por esto que se ha visto la necesidad de entrenar cada uno de los comandos para que pese a que el reconocimiento de voz no de la respuesta dicha, todas las opciones cercanas también sean aceptadas, con esto el programa podrá entender por ejemplo que sí reconoció la orden “Ok Robert” esta corresponde a la palabra clave “Ok Robot”.

Otro objetivo del reconocimiento de voz es el lograr que el lenguaje de comunicación entre persona y maquina sea natural, con esto se quiere decir que no es necesario el tener una única forma de dar una orden como se hace en la comunicación entre humanos. Por ejemplo, si se pide a una persona realizar una actividad como levantar un objeto del piso se pueden decir cosas como “Levanta eso”, “por favor levanta eso”, “levántalo”, “recoge eso”, “recógelo”, y la persona será capaz de entender cada uno de estos comandos y ejecutar la orden.

El entrenamiento del aplicativo de reconocimiento de voz busca incluir lenguaje natural para su funcionamiento, es por esto que para cada comando existen varias opciones que también serán reconocidas. Para ilustrar esto, si se requiere que el robot realice un movimiento hacia arriba y se da el comando “up”, adicionalmente a esto se puede decir los comandos “go up” y “move up”. Esto hace

que la aplicación sea más intuitiva y reconozca el lenguaje de un operario pese a que este no esté familiarizado con el funcionamiento.

El proceso de entrenamiento se realiza con la metodología que se describe a continuación:

1. Determinar la acción que se desea que realice el robot
2. Buscar una expresión que describa de la manera más simple la acción deseada para el robot.
3. Buscar expresiones similares a la encontrada en el punto anterior, que pueden o no contener la orden principal encontrada en el punto anterior.
4. Ejecutar la función de reconocimiento de voz, y empezar a pronunciar todas las expresiones halladas en el punto anterior. En este punto se debe anotar cada una de las respuestas que de la función ya que aquí se encontraran opciones no tomadas en cuenta anteriormente.
5. Cada uno de los comandos deben ser dichos en la aplicación varias veces procurando cambiar la velocidad y la pronunciación, si se desea mejorar la exactitud del reconocimiento se puede hacer que varias personas pronuncien los comandos con las variaciones que ellos consideren.
6. Guardar los resultados obtenidos con cada una de las variaciones realizadas en una lista, la cual debe ir en el código del programa.

Para mostrar el proceso de entrenamiento del reconocimiento de voz, a continuación, se muestra como se ha entrenado el comando “Up” que tiene el propósito de generar un movimiento en el robot de 20 mm hacia arriba en el eje z.

1. La acción deseada es un movimiento hacia arriba del robot.
2. El comando más básico para indicar ese movimiento es “up”.
3. Comandos similares pueden ser “go up”, “move up”.
4. La respuesta de la función de reconocimiento de voz son las variaciones “App”, “Go app”, “morph app”, “move app”
5. Al buscar más variaciones no se encuentra ninguna con presencia significativa para ser tomada en cuenta.
6. Se juntan todas las opciones obtenidas en la siguiente lista

$$\gg go\_up = ["up", "app", "Go\ app", "morph\ app", "move\ app", "go\ up", "move\ up"]$$

Con esto se logra que la aplicación reconozca las variaciones más frecuentes del comando “up” y así obtener mejores resultados al momento del funcionamiento del control de movimiento del robot.

El tiempo aproximado de lectura de cada uno de los comandos es de 30 segundos, en este caso se encontraron tres posibles variaciones las que se pronunciaron hasta que dejaron de aparecer nuevas

variaciones lo cual fue aproximadamente 15 iteraciones. El tiempo aproximado de entrenamiento se detalla con la formula a continuación.

t: $tiempo\ por\ comando\ [s]$

n: $numero\ de\ variaciones + 1$

e: $cantidad\ de\ veces\ que\ se\ entrena\ cada\ variacion$

t_e: $tiempo\ total\ de\ entrenamiento\ por\ comando$

$$t_e = t * n * e\ [s]$$

Para el caso del comando "up" el tiempo aproximado de entrenamiento se calcula a continuación

$$t = 30\ [s]$$

$$n = 2 + 1$$

$$e = 15$$

$$t_e = t * n * e\ [s]$$

$$t_e = (30) * (3) * (15)\ [s]$$

$$t_e = 1350\ [s]$$

Con este cálculo de tiempo se observa que en el caso del comando "Up" el tiempo aproximado de entrenamiento para una persona es de 1350 segundos lo cual es equivalente aproximadamente a 23 minutos. Si el entrenamiento es realizado por varias personas, cada una de ellas demoraría aproximadamente el mismo tiempo, y el tiempo total deberá multiplicarse por el número de personas.

A continuación, la Tabla 3 detalla las funciones programadas para el movimiento del robot.

Comando	Función
ok robot	Palabra clave de inicio de la aplicación
home	Ir a la posición inicial del robot
Up	Moverse hacia arriba (eje z positivo)
Down	Moverse hacia abajo (eje z negativo)
Right	Moverse a la derecha (eje y positivo)
Left	Moverse a la izquierda (eje y negativo)
Front	Moverse hacia el frente (eje x positivo)
Back	Moverse hacia atrás (eje x negativo)

Point a	Moverse al punto a
Point b	Moverse al punto b
Point c	Moverse al punto c
Recognize	Reconocer el área de trabajo

Tabla 3. Comandos de movimiento del robot

Cada uno de estos comandos ha sido entrenado procurando lograr lenguaje natural y con las variaciones más frecuentes presentadas, y el resultado se muestra a continuación en la Tabla 4. Adicional a las mostradas en la tabla hay más opciones programadas, pero solo se muestran las principales.

Comando	**Variaciones Principales**					
ok robot	okay Robert	okay robot	okay rabbit	hey Robert	okay	okay remote
home	home	come	crumb	Chrome		
up	up	app	go app	morph up	move app	move up
down	down	go down	move down	more down	bow down	morph down
right	right	go right	move right	bright	gold right	
left	left	go left	move left	more left	bow left	
front	front	go front	move front	how front	move to front	go to front
back	back	go back	move back	move to back		
point a	point a	go to point a	go to point a	move to the point a	go to the point a	go point a
point b	point b	go to point b	go to point b	move to the point b	go to the point b	go point b
point c	point c	go to point c	go to point c	move to the point c	go to the point c	go point c
recognize	recognize					

Tabla 4. Comandos de movimiento y variaciones

3.1.5. Funcionamiento del aplicativo en Python

El aplicativo de control de movimiento de un robot ABB por medio de comandos de voz, ha sido creado con base en varias funcionalidades que se integran para obtener un programa funcional y capaz de cumplir con el objetivo de su diseño. Las funcionalidades mencionadas son:

- Comunicación con Google Cloud
- Reconocimiento de voz
- Entrenamiento del reconocimiento de voz
- Identificación de los comandos
- Creación de un cliente OPC

- Comunicación con el servidor OPC

La secuencia lógica del algoritmo se describe a continuación

1. Arranca el aplicativo
2. Se establece la comunicación con Google Cloud, y se comienza a reconocer la voz del operario.
3. Se solicita una palabra clave de inicio del sistema. La palabra clave es “OK ROBOT”. Si se identifica la palabra clave se inicia el control del robot, caso contrario sigue preguntando la palabra clave. Si no se conoce la palabra clave es posible decir el comando “EXIT” para salir del sistema y terminar el aplicativo.
4. Al identificar la palabra clave, se establece comunicación con el servidor OPC y se configura las variables “start” y “mode” en cero, lo que indica que el robot no debe realizar ningún tipo de movimiento.
5. El aplicativo solicita decir un comando y comienza a grabar el sonido con el micrófono. Crea un archivo de audio con lo que el usuario habla. Sube el audio al servidor de Google Cloud y obtiene una transcripción de lo hablado por el usuario.
6. Si la transcripción coincide con alguna de las opciones de los comandos preestablecidos (entrenamiento del reconocimiento de voz), identifica el valor numérico que está relacionado al comando, y escribe en el servidor OPC los valores start=1 y mode=<<valor numérico comando>> este proceso hace que el robot realice el movimiento que la variable mode indique.
7. Una vez el robot ha realizado el movimiento, se escribe en el servidor los valores start=0 y mode=0, lo que indica que el robot no debe realizar ningún tipo de movimiento.
8. El aplicativo vuelve a pedir comandos para movimiento de voz, hasta que el operario dice el comando “EXIT” que termina el aplicativo y envía el robot a la posición home.

Este proceso iterativo que realiza el aplicativo se describe en el diagrama de flujo mostrado en la Figura 41.

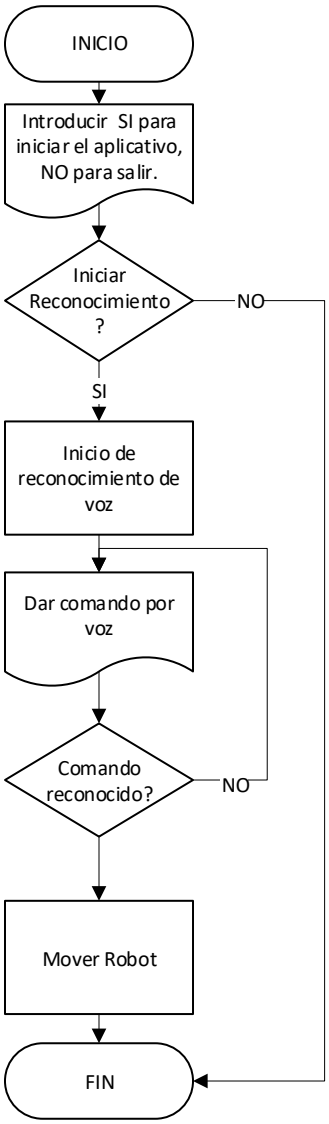

Figura 41. Diagrama de flujo del aplicativo de reconocimiento de voz

3.2. Comunicación OPC

En este proyecto se utiliza el programa desarrollado por la empresa ABB llamado IRC5 OPC Server que está diseñado para establecer una red de comunicación entre el robot y los computadores que los controlaran. En este caso se conecta un robot ABB IRB140 con computador donde se ejecuta el aplicativo de reconocimiento de voz desarrollado en el lenguaje de programación Python.

3.2.1. Estándar de comunicación OPC

3.2.1.1. Tecnología OPC

OPC es una tecnología de comunicación con una arquitectura de cliente y servidor. Una aplicación actúa de servidor proporcionando datos y otra actúa como cliente leyéndolos o manipulándolos. OPC

es, con mucha diferencia, la tecnología de comunicación industrial estándar. Ello permite el intercambio de información entre múltiples dispositivos y aplicaciones de control sin restricciones o límites impuestos por los fabricantes. Un servidor OPC puede estar comunicándose continuamente con los PLCs de campo, RTUs, estaciones HMI u otras aplicaciones. Aunque el hardware y el software provengan de diferentes marcas comerciales, el cumplimiento del estándar OPC posibilita la comunicación continua en tiempo real.

Por ello, OPC ha permitido una mejor cooperación entre proveedores y usuarios, ayudando a construir soluciones completamente transversales, dando a los consumidores más poder de elección entre diferentes aplicaciones industriales. La interoperabilidad, las soluciones modulares y la libertad de elección han sido los grandes motivadores para que los usuarios de todo el mundo – y por tanto los proveedores – hayan incorporado OPC a sus entornos industriales.

Si OPC es un estándar abierto es porque su interoperabilidad viene dada por la creación, mantenimiento y mejora de unas especificaciones estándar realizadas por un grupo de trabajo multidisciplinar y que no se decanta por ninguna marca en particular. La primera especificación fue resultado de la colaboración de un conjunto de fabricantes en colaboración con Microsoft.

Los softwares que tienen la capacidad de adquirir datos de los dispositivos de campo y servirlos en OPC son los llamados Servidores OPC o OPC Servers. El gran beneficio que aportan es poder desvincular los sistemas de explotación de datos superiores de la casuística concreta de campo. Con ello, se consigue toda la información del sistema, aunque se tengan controladores, HMIs, o dispositivos de diferentes fabricantes que utilicen diferentes protocolos.

3.2.1.2. OPC UA

La comunicación es un requisito previo para los sistemas distribuidos. Dichos sistemas pueden definirse libremente como un grupo de sistemas informáticos individuales que aparecen ante el usuario como un único sistema coherente. La naturaleza espacialmente dispersa de los procesos industriales, en la escala de un piso de una fábrica o de una red eléctrica, en realidad se usa a menudo como una guía para el diseño y la distribución de los sistemas de automatización. Esto se puede observar, por ejemplo, en el control basado en la red, donde el bucle de control puede distribuirse entre diferentes procesadores en una red. En este entorno, se aplican los principios básicos de los sistemas distribuidos. Sin embargo, si bien la teoría clásica de los sistemas distribuidos se ha desarrollado teniendo en cuenta principalmente los sistemas informáticos de uso general, la automatización industrial se centra en sistemas dedicados con hardware y software altamente especializados. (Wilamowski & Irwin, 2011)

Uno de los estándares de comunicación más utilizados en la actualidad es el OPC (OLE for Process Control), el cual es un estándar de comunicación en el campo del control y supervisión de procesos industriales, basado en una tecnología Microsoft, que ofrece una interfaz común para comunicación que permite que componentes de software individuales interactúen y compartan datos. La comunicación OPC se realiza a través de una arquitectura Cliente-servidor. El servidor OPC es la fuente de datos (como un dispositivo hardware a nivel de planta) y cualquier aplicación basada en OPC puede acceder a dicho servidor para leer/escribir cualquier variable que ofrezca el servidor. Es una solución abierta y flexible al clásico problema de los drivers propietarios. Prácticamente todos los mayores fabricantes de sistemas de control, instrumentación y de procesos han incluido OPC en sus productos.

OPC Unified Architecture (OPC UA) es el nuevo estándar de la Fundación OPC que proporciona interoperabilidad (por ejemplo, la tecnología Microsoft DCOM ya no se usa) en la automatización de procesos y más allá. Al definir servicios abstractos, OPC UA proporciona una arquitectura orientada a servicios (SOA) para aplicaciones industriales, desde dispositivos de planta a aplicaciones empresariales. Desde la primera introducción de OPC en 1996, OPC UA representa un gran logro, que comenzó a fines de 2003, que propone integrar las diferentes características de las especificaciones OPC anteriores en un espacio de direcciones unificado accesible con un único conjunto de servicios.

El uso típico de OPC UA se refiere a los servicios de middleware que ofrecen una capa de abstracción entre dispositivos (por ejemplo, controlador lógico programable [PLC], sensores, actuadores, ...) y aplicaciones de planta (SCADA, MES, ...), como se muestra en la Figura 42.

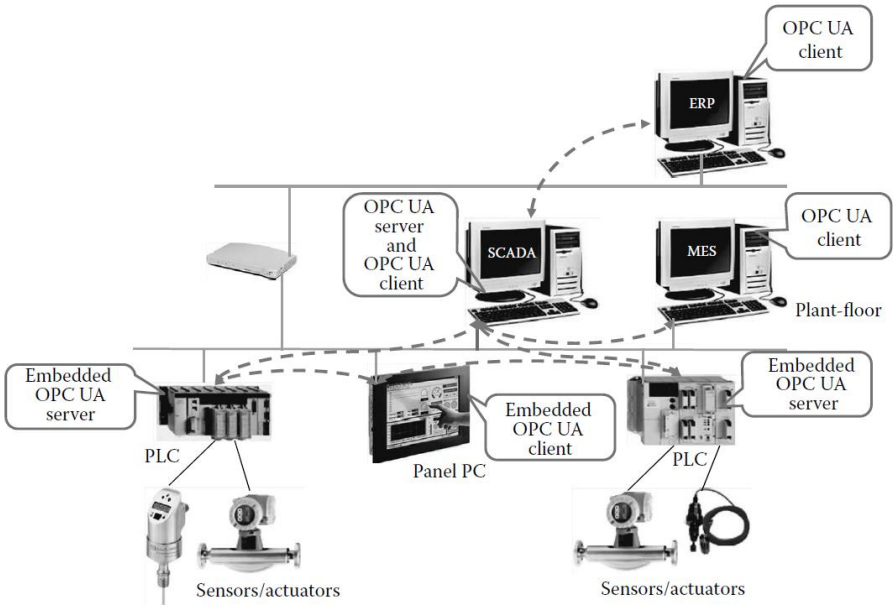

Figura 42. Ejemplo de una aplicación de OPC UA (Wilamowski & Irwin, 2011)

Como se observa en la Figura 42, el servidor OPC sirve como vínculo entre los diferentes dispositivos en un sistema de automatización, así se conectan elementos como sensores con elementos de supervisión y control, e incluso varios sistemas independientes pueden compartir información entre ellos. La comunicación se establece con el esquema de cliente servidor.

3.2.1.3. Arquitectura OPC UA

La arquitectura del software OPC UA está diseñada utilizando el modelo de cinco capas, como se mostrado en la Figura 43. Los modelos de información de negocios (por ejemplo, EDDL, MIMOSA, ISA ...) y el modelo de información de proveedores generalmente se construyen sobre los servicios básicos de OPC UA, que también proporcionan la base para la reconstrucción de los modelos de información existentes de OPC.

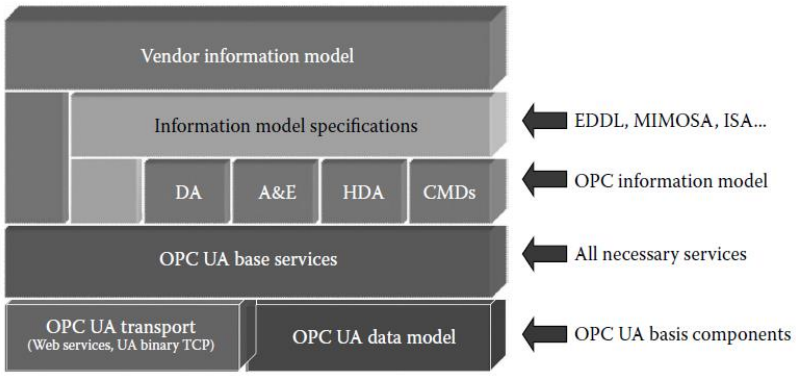

Figura 43. Arquitectura de OPC UA (Wilamowski & Irwin, 2011)

3.2.1.4. Aplicaciones OPC UA

OPC UA aborda una amplia gama de aplicaciones de destino:

- Control de supervisión y adquisición de datos (SCADA) e interfaz hombre-máquina (HMI) o, en general, interfaz de sistema humano (HSI).
- Control avanzado en el sistema de control distribuido (DCS): los comandos pueden enviarse a dispositivos y controlador lógico programable (PLC).
- El sistema de ejecución de fabricación (MES) que generalmente se implementa en la supervisión, simulación y programación de procesos por lotes (por ejemplo, proceso farmacéutico).
- Middleware que sirve aplicaciones intra-planta y aplicaciones extra-planta (a nivel empresarial)

Middleware es una pieza de software que sirve para "pegar entre sí" o mediar entre dos programas separados y generalmente ya existentes. El middleware a veces se llama plomería porque conecta dos lados de una aplicación y pasa datos entre ellos. El middleware desempeña un papel crucial en las aplicaciones empresariales al integrar los componentes del sistema, lo que les permite interactuar de manera correcta y confiable, y facilitar la administración y la evolución del sistema. Este es un dominio muy importante de las aplicaciones a las que se dirige OPC UA, uniendo los sistemas de información industrial con los de la empresa. (Wilamowski & Irwin, 2011)

La arquitectura del sistema de OPC UA se basa en el concepto bien conocido de interacción cliente-servidor. Cada sistema puede contener múltiples clientes y servidores. Cada cliente puede interactuar simultáneamente con uno o más servidores, y cada servidor puede interactuar simultáneamente con uno o más clientes. Una aplicación puede combinar componentes de servidor y cliente para permitir la interacción con otros servidores y clientes, como se muestra en la Figura 44.

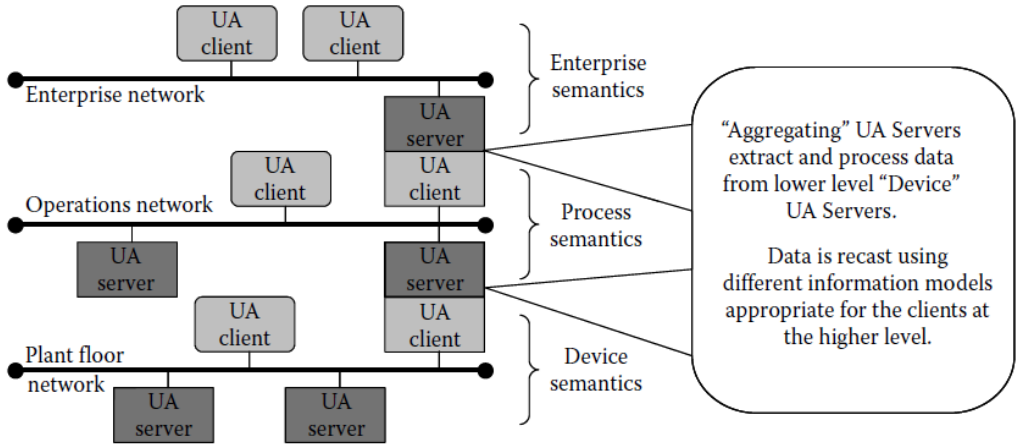

Figura 44. Cadena de servidores y clientes OPC UA (Wilamowski & Irwin, 2011)

3.2.1.5. Arquitectura de aplicaciones cliente UA

Para escribir una aplicación cliente OPC UA, generalmente se necesita usar la interfaz de programación de aplicaciones (API) OPC UA, que expone los servicios estándar que deben usarse para enviar y recibir solicitudes de servicio OPC UA y respuestas al servidor OPC UA, como se muestra en la Figura 45. La API aísla el código de la aplicación cliente de una pila de comunicación OPC UA que se puede elegir entre diferentes tipos: UA Binary sobre TCP y SOAP XML / Text sobre HTTP. La pila de comunicación OPC UA convierte las llamadas de la API del cliente OPC UA en mensajes y las envía a través de la entidad de comunicaciones subyacente al servidor a solicitud de la aplicación cliente. La pila de comunicaciones OPC UA también recibe respuestas y mensajes de notificación de la entidad de comunicaciones subyacente.

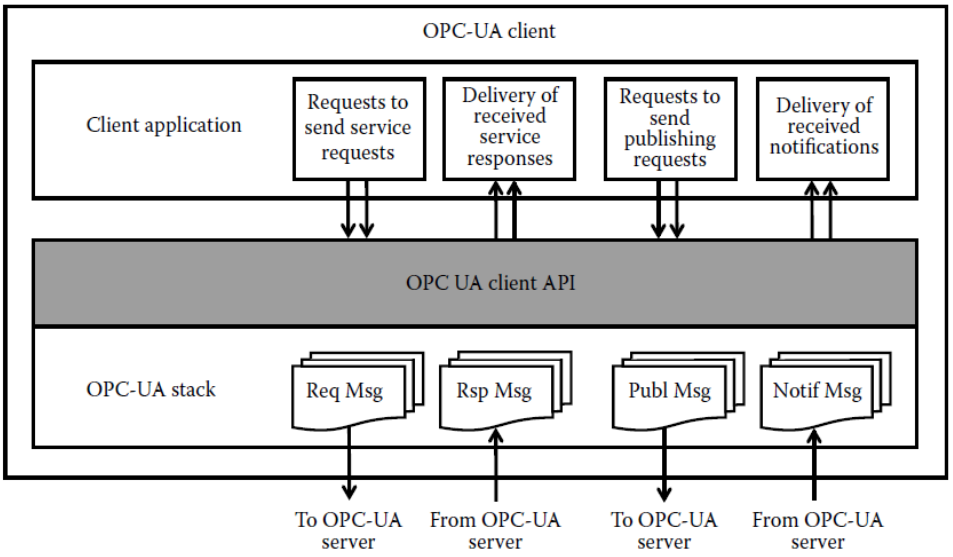

Figura 45. Arquitectura de un cliente OPC (Wilamowski & Irwin, 2011)

En el caso de la aplicación de reconocimiento de voz, esta debe ser un cliente OPC que se comunique con la controladora del robot y sea capaz de enviar información.

3.2.2. Configuración del sistema

Esta sección presenta una descripción detallada de como instalar y configurar el software ABB IRC5 OPC Server y como establecer un cliente OPC desde Python que sea capaz de escribir información de manera remota en la controladora del robot para de esta manera controlar sus movimientos.

3.2.2.1. Configuración del sistema para el Servidor OPC IRC5

Primero, es necesario Instalar el software ABB IRC5 OPC Server, este programa está disponible en la página oficial de ABB, y funciona únicamente con controladoras de robots IRC5 que tengan instalada esta funcionalidad y en un sistema operativo Windows. El link de descarga se encuentra a continuación.

$$http://developercenter.robotstudio.com/downloads\_opcserver$$

Se debe seleccionar la versión adecuada ya que este funciona con los programas RobotStudio que es el software de simulación y RobotWare que es el sistema que corre internamente en la controladora del robot. En este caso se trabaja con la versión IRC5OPCServer.6.08.01 mostrado en la Figura 46 que es la más actual al momento del desarrollo del proyecto.

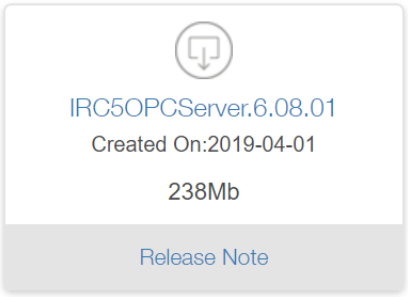

Figura 46. Descarga del servidor ABB OPC IRC5

Tras la descarga, es necesario ejecutar el archivo e instalar el programa con las opciones que vienen por defecto. Al ejecutar la aplicación se muestra una ventana mostrada en la Figura 47 que brinda varias opciones para la creación y configuración del servidor OPC. La pantalla principal de la aplicación de configuración del servidor OPC de ABB IRC5 muestra una lista de Alias que se han creado, esto representa tanto los robots físicos como en simulación que están conectados al servidor. La pantalla principal muestra importantes información sobre los alias creados, el nombre asignado, el nombre del controlador, Nombre del sistema, dirección, etc.

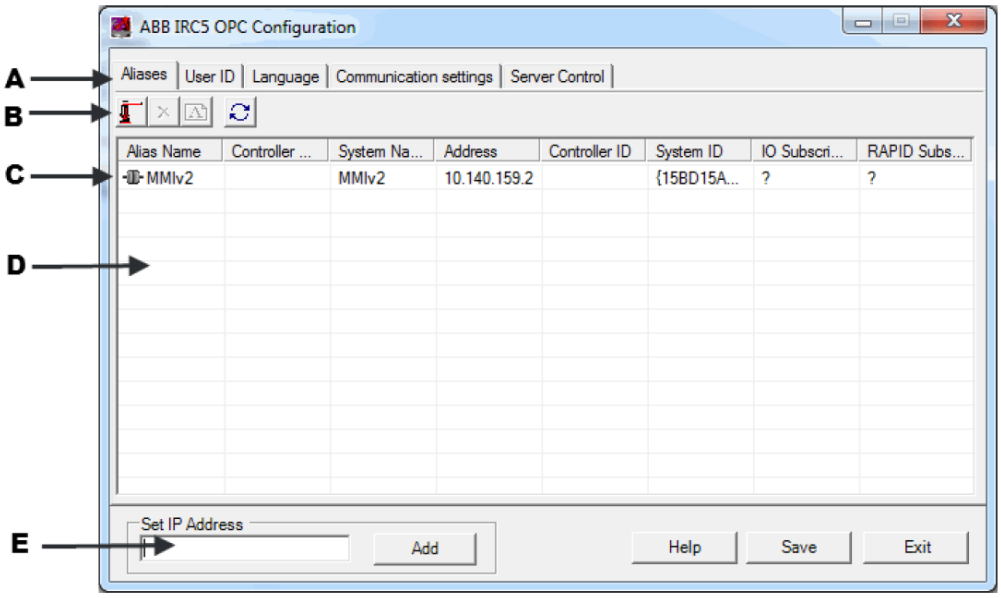

Figura 47. Pantalla principal de configuración de la aplicación ABB IRC5 OPC Server

La Tabla 5 muestra los elementos principales presentes en la pantalla principal de configuración mostrados en la Figura 47 del servidor OPC IRC5, el programa presenta las opciones necesarias para establecer la conexión con todos los robots presentes en la red y poder gestionarlos.

A	Function tab
B	Toolbar buttons

C	Device status icons
D	Device panel
E	Add remote controller

Tabla 5. Descripción de los elementos de la pantalla principal de ABB IRC OPC Server (ABB Robotics, 2015)

En las pestañas de funciones (Toolbar buttons) se presentan varias opciones detalladas en la Tabla 6:

Componente	Función
Alias Tab	Configuraciones principales del servidor OPC
User ID Tab	Ingreso del nombre de usuario y contraseña para que el OPC pueda acceder al robot
Language	Selección del lenguaje utilizado en el servidor OPC

Tabla 6. ABB OPC Server pestañas de funciones (ABB Robotics, 2015)

La función que cumple cada uno de los botones en la barra Toolbar buttons se describe en la Tabla 7:

Botón	Función
Add Alias 	Abre la ventana de dialogo para añadir un alias de robot IRC5
Delete Alias 	Remueve el alias seleccionado del panel de dispositivos
Edit Alias 	Abre el cuadro de dialogo editar alias
Refresh Main Screen 	Actualiza el estatus de los alias de robots en el panel de dispositivos

Tabla 7. ABB OPC Server Toolbar buttons (ABB Robotics, 2015)

El panel de dispositivos (device panel) muestra información de todos los dispositivos disponibles en la red del servidor OPC, y la información detallada en este panel se detalla en la Tabla 8.

Componente	Función
Alias Name	Descripción amigable al usuario para identificar la comunicación con la controladora del robot
Controller System	El nombre de la controladora IRC5
System Name	El nombre del sistema RobotWare corriendo en la controladora IRC5
Address	La dirección IP de la controladora IRC5

Controller ID	La ID de la controladora IRC5
System ID	La ID del sistema RobotWare corriendo en la controladora IRC5
IO Subscriptions	El número de señales de entradas/salidas subscritas a la controladora IRC5
RAPID Subscriptions	El número de variables de Rapid subscritas a la controladora IRC5

Tabla 8. ABB OPC Server panel de dispositivos (Device panel) (ABB Robotics, 2015)

3.2.2.2. Configuración del sistema para el cliente OPC en Python

A continuación, se detallan todos los requerimientos del sistema para poder establecer un cliente OPC desde Python con lenguaje de programación. Estos elementos se describen a continuación.

- Python 3.7 32 bits
- PYWIN 32 bits
- OPC DA Auto Wrapper
- OpenOPC para Python

3.2.2.3. Python

En el presente proyecto se utilizará Python como cliente OPC para poder acceder a las variables presentes en el código Rapid que corre dentro de la memoria de la controladora del robot, pero para usar Python como cliente OPC es necesario instalar varios drivers y programas que se describen a continuación.

El primer paso en la configuración es descargar e instalar Python 3.7 de 32 bits desde la web oficial, este es un programa gratuito. El siguiente es el link de descarga oficial de Python.

$$https://www.python.org/downloads/$$

3.2.2.4. PYWIN 32 bits

Una vez se ha descargado e instalado la última versión de Python, se debe descargar e instalar la extensión de Python para Windows 32 bits llamada “PYWIN 32 bits”, la cual provee acceso a muchas de las APIs de Windows desde Python. A continuación, se muestra un enlace de donde se puede descargar la extensión “PYWIN 32 bits”, es importante tener en cuenta que se debe descargar la versión correspondiente a la versión de Python disponible en el sistema.

$$https://github.com/mhammond/pywin32/releases$$

Al iniciar la instalación se mostrará una ventana como la que aparece en la Figura 48. Se debe seleccionar siguiente en todas las opciones hasta que al final aparece una notificación que indica que se ha instalado la extensión correctamente.

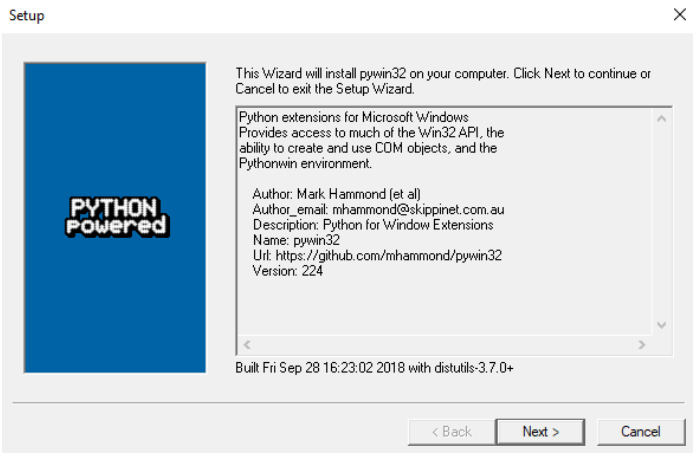

Figura 48. Instalación de PyWin32

Además de la opción anterior para instalar la extensión "PYWIN 32 bits" existe otra que utiliza la función PIP (Python Package Index) desde la consola de Windows. Para hacer esto se debe ejecutar el comando mostrado en la Figura 49.

$$pip\ install\ pywin32$$

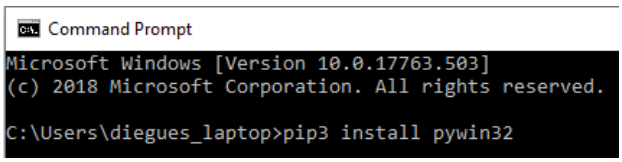

Figura 49. Comando de instalación de PyWin32 desde la consola de Windows

Si esta opción no presenta ningún problema al final de la instalación saldrá una nota de confirmación indicando que se ha instalado correctamente la extensión.

3.2.2.5. OPC DA Auto Wrapper

El siguiente paso es descargar e instalar el módulo dll llamado Graybox OPC auto graped, en el que se implementan todos los objetos OLE necesarios para establecer un cliente OPC. Después de registrar este módulo, podrá usar cualquier servidor de acceso a datos OPC con casi cualquier lenguaje de programación habilitado para OLE (Visual Basic, VBA, etc.).

Se puede descargar Graybox OPC DA Auto Wrapper de forma gratuita. Además, este puede ser usado sin costo en cualquier proyecto aun si es para distribuirlo o comercializarlo. No hay limitación de licencia para este producto. El link de descarga se muestra a continuación.

$$http://www.gray-box.net/download\_daawrapper.php$$

Una vez descargado el archivo, se debe proceder a instalarlo con comando regsvr32 gbda_aut.dll desde la consola de Windows como se muestra en la Figura 50, y desde el directorio donde este se encuentre. Al finalizar aparece una ventana que indica que se ha instalado el archivo dll exitosamente.

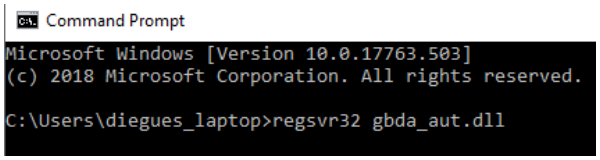

Figura 50. Instalación de Graybox OPC auto graped

3.2.2.6. OPENOPC

OpenOPC para Python es un kit de herramientas OPC (OLE for Process Control) gratuito, de código abierto, diseñado para usar con el popular lenguaje de programación Python. Las características únicas que lo distinguen de los muchos kits de herramientas OPC disponibles comercialmente incluyen:

- Es de fácil uso debido a que la biblioteca OpenOPC implementa un número mínimo de funciones de Python que pueden estar encadenadas de varias maneras, la biblioteca es fácil de aprender y de recordar. En su forma más simple, puede leer y escribir elementos OPC tan fácilmente como cualquier variable en un programa de Python
- Soporte para varias plataformas, OpenOPC funciona con plataformas Windows y no Windows. Se ha probado con Windows, Linux y Mac OS X.
- Estilo de programación funcional, OpenOPC permite que las llamadas OPC se encadenen en un estilo de programación elegante y funcional. Por ejemplo, puede leer los valores de todos los elementos que coincidan con un patrón de comodín utilizando una sola línea de código Python.
- Diseñado para lenguajes dinámicos, la mayoría de los kits de herramientas OPC de hoy en día están diseñados para usarse con lenguajes de sistema estáticos (como C ++ o C #), lo que proporciona un mapeo cercano a los métodos COM de Win32 subyacentes. OpenOPC descarta este engorroso modelo y, en su lugar, intenta aprovechar las características dinámicas de lenguaje proporcionadas por Python.

Para instalar este software para Python se debe ejecutar el comando mostrado en la Figura 51 desde la consola de Windows con la función PIP.

$$Pip3\ install\ OpenOPC - Python3x$$

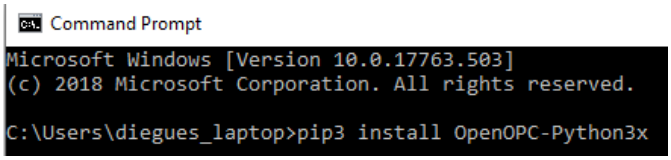

Figura 51. Instalación de OpenOPC

Tras ejecutarse la instrucción, si no se ha presentado ningún error aparece una nota de confirmación indicando que la instalación ha sido exitosa. Una vez instalados los 4 requerimientos que se resumen a continuación, se puede establecer una conexión como cliente OPC desde lenguaje de programación Python. Esto permitirá establecer la comunican entre la aplicación de reconocimiento de voz y el robot.

3.2.3. Configuraciones de parámetros del servidor OPC

3.2.3.1. Configuración de parámetros del servidor ABB OPC IRC5

Una vez se tiene identificadas las opciones de configuración del programa ABB IRC5 OPC Server, y habiendo establecido la conexión física de la controladora del robot con el computador donde se ejecute la aplicación de reconocimiento de voz, se procede a configurar un nuevo alias.

Presionar el botón new alias que se encuentra en la Figura 52.

Figura 52. Botón New Alias

Tras presionar el botón se debe abrir la ventana de dialogo para añadir un nuevo alias mostrada en la Figura 53.

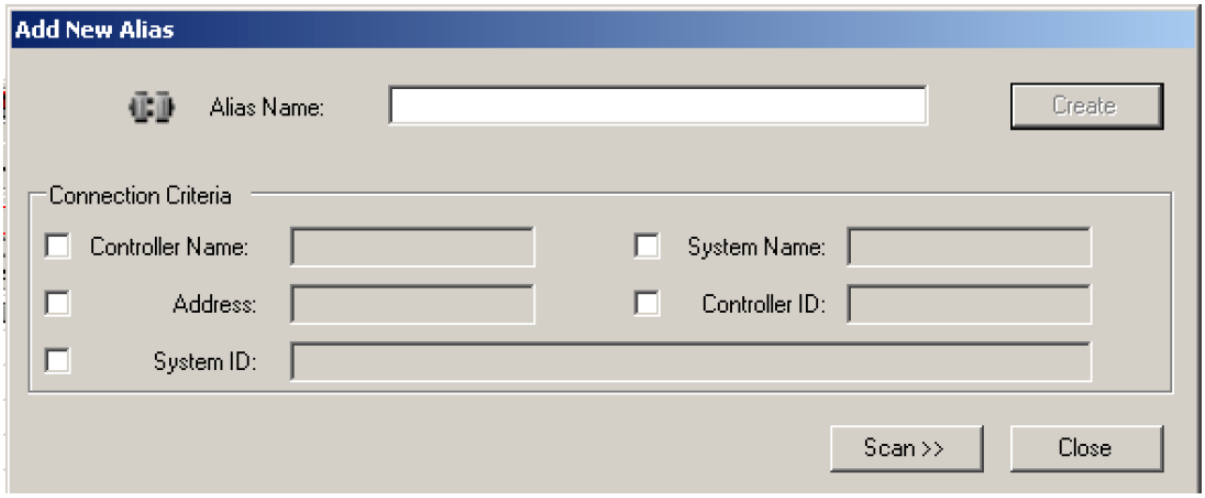

Figura 53. Cuadro de dialogo de nuevo alias

La descripción de cada uno de los elementos del cuadro de dialogo de la Figura 53 para añadir un nuevo alias se detallan en la Tabla 9 a continuación.

Componente	Función
Alias Name field	Este campo permite introducir un nombre para el alias
Create button	Al presionar se crea un nuevo alias, este cuadro se activa tras completar toda la información
Connection Criteria check boxes	Controller name: Es el nombre asignado a la controladora del robot System Name: Es el nombre asignado al sistema BaseWare activo en la controladora Address: Es la IP del robot. Si la controladora es virtual muestra el directorio Controller ID: Es el ID único de la controladora del robot System ID: Es la ID asignada al sistema BaseWare activo en la controladora del robot
Scan button	Muestra los resultados al escanear posibles conexiones

Tabla 9. ABB OPC Server opciones para crear un nuevo alias (ABB Robotics, 2015)

Si se ha establecido correctamente la conexión física por medio de un cable, y se conoce todos los parámetros de configuración de la controladora del robot, como direcciones IP, nombres de controladoras, ID de controladoras y de robots se pueden introducir manualmente y establecer la comunicación como se muestra en la Figura 54.

Controller Name	System Name	Address	Controller ID	System ID
	rel5_xx	192.168.8.59	3HAC14431-1	{19048F68-0D4A-...

Figura 54. Configuración manual de un alias para una controladora IRC5 (ABB Robotics, 2015)

Si, por el contrario, no se saben conocen todos los detalles de configuración de la conexión es mejor utilizar la herramienta para escanear todas las controladoras conectadas a la red donde se encuentra el computador. El servidor puede establecer comunicación con múltiples dispositivos simultáneamente por lo que esta herramienta sirve para trabajar con varios robots en una misma red.

Se selecciona el botón $scan >>$ y se obtiene como resultado una lista con todos los dispositivos conectados al servidor y su información de configuración como se muestra en la Figura 55.

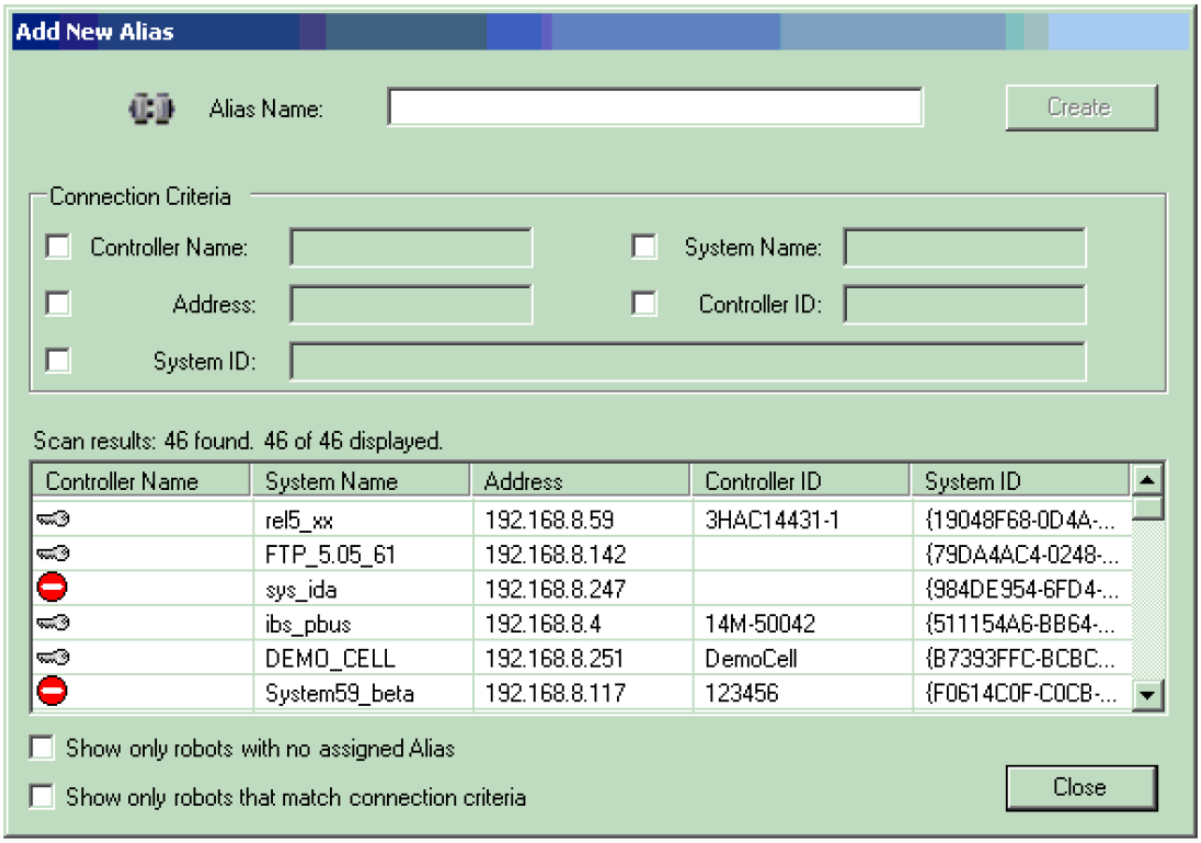

Figura 55. Scan new alias

Luego de haber escaneado la red, si se muestra la controladora con la que se quiere establecer conexión es necesario seleccionarla para poder crear un nuevo alias y completar ciertos datos que se detallan en la siguiente imagen, este proceso esta mostrado en la Figura 56.

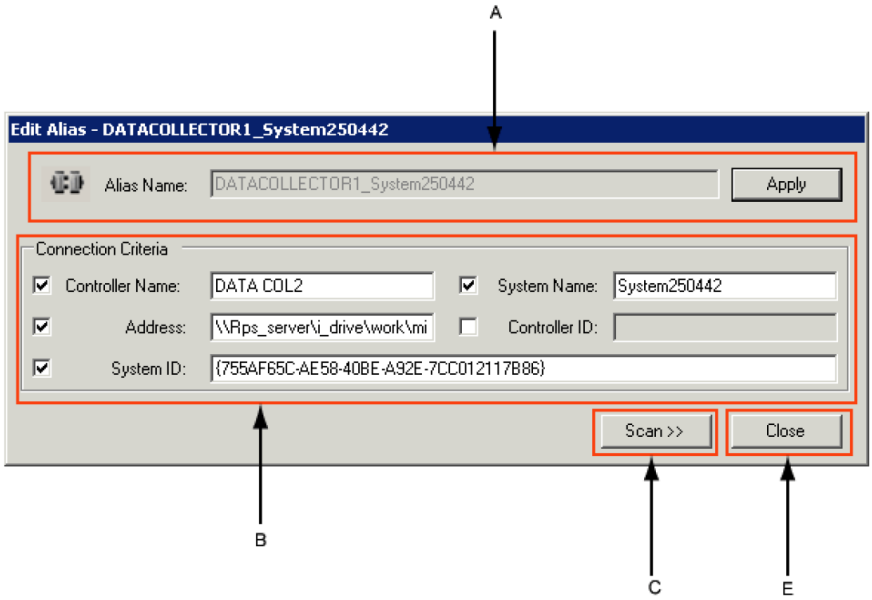

Figura 56. Configuración de un nuevo alias

Las funcionalidades de esta pantalla mostrada en la Figura 56 se describen en la Tabla 10.

A	Nombre de alias
B	Criterio de conexión
C	Botón de escaneo
D	Botón de cierre

Tabla 10. ABB OPC Server campos de la ventana de configuración de nuevo alias (ABB Robotics, 2015)

3.2.3.2. Configuración de parámetros del cliente OPC en Python

Como se indicó anteriormente, la configuración del cliente OPC desde Python se realizará con ayuda del programa OpenOPC, el cual presenta funciones concretas para establecer la conexión al servidor. La configuración de la comunicación por medio de las funciones de OpenOPC en la consola de ejecución de Python se detallan a continuación.

3.2.3.2.1. Importar el módulo OpenOPC

El primer paso es hacer que la biblioteca OpenOPC esté disponible para la aplicación. Este comando importa el archivo del módulo OpenOPC.py ubicado en el directorio lib / site-packages /.

$$>> \ import\ OpenOPC$$

3.2.3.2.2. Crear una instancia OpenOPC (modo DCOM)

El modo DCOM se utiliza para comunicarse directamente con los servidores OPC sin la necesidad del servicio de puerta de enlace OpenOPC. Este modo solo está disponible para clientes Windows.

$$>> \ opc \ = \ OpenOPC.client()$$

3.2.3.2.3. Obtener una lista de servidores OPC disponibles

El siguiente comando muestra todos los servidores OPC conectados a la red tanto de manera local como remota.

$$>> \ opc.servers()$$

3.2.3.2.4. Conectar al servidor OPC seleccionado

Esta función permite conectarse al servidor OPC especificado. La función devuelve True si la conexión fue exitosa, False en caso de falla de conexión.

$$>> \ opc.connect(ABB.IRC5.OPC.Server.DA)$$

3.2.3.2.5. Obtener una lista de tags disponibles en el servidor OPC

Retorna una lista con todos los tags encontrados en una dirección especificada

$$>> \ opc.list()$$

3.2.3.2.6. Leer un tag en el servidor OPC

Esta función lee el tag del servidor OPC especificado. La función devuelve los parámetros (valor, calidad, marca de tiempo). Si la llamada falla, la calidad se establecerá en 'Error'.

$$>> \ opc.read('< Nombre\ del\ Tag >')$$

3.2.3.2.7. Escribir en un tag del servidor OPC

La escritura de un solo elemento se puede realizar al enviar los datos de nombre de tag y el valor que se desea escribir (nombre, valor) a la función de escritura. Si la escritura fue exitosa, se devuelve True, o False en caso de error.

$$>> \ opc.write(('< Nombre\ del\ Tag >', < Valor >))$$

La Figura 57 muestra como se ha establecido comunicación por medio de Python con el servidor ABB OPC IRC5. El aplicativo de Python se configura como un cliente OPC capaz de leer y escribir sobre variables del programa de ejecución cargado en la controladora del robot.

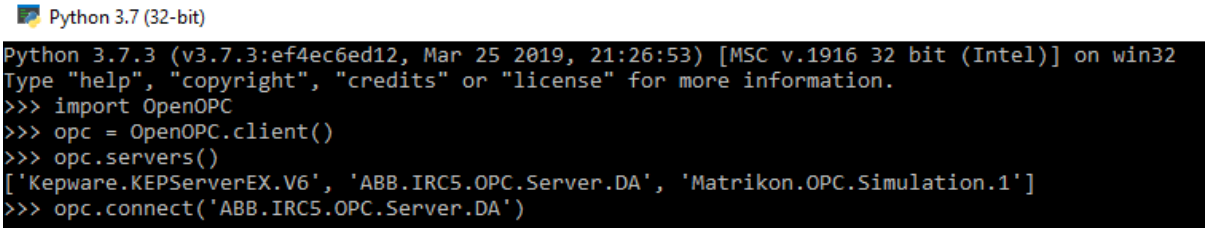

Figura 57. Configuración de un cliente OPC desde Python

Una vez configurado el cliente OPC se puede verificar desde este que la conexión con el servidor OPC del robot se ha establecido correctamente y se puede proceder a configurar los nombres de los tags de las variables de comunicación.

3.2.4. Configuración de las variables de comunicación

Para poder establecer la comunicación se utiliza la función de acceso a datos del servidor OPC ABC IRC5, con la cual se puede leer y escribir datos gestionados por el controlador de robot ABB IRC5. Los datos en el servidor ABB IRC5 OPC son referidos por sus nombres de etiqueta (tag names).

El servidor OPC IRC5 de ABB presenta varias etiquetas predefinidas que proporcionan información sobre el estado actual del controlador del robot. Además de estas etiquetas predefinidas, el servidor OPC presenta hasta 1000 etiquetas adicionales que contienen los valores de las señales de E / S del IRC5, así como hasta 200 etiquetas que contienen el Los valores de los datos de RAPID del IRC5.

La Figura 58 muestra la dirección de los accesos de memoria de los tags en una controladora IRC5, donde se pueden identificar tres tipos de variables

- System: estas indican los estados de todos los procesos internos del robot como por ejemplo detección de colisiones, hora del sistema, velocidad de movimiento, estado de los motores, nombre del sistema, etc.
- I/O System: Este tipo de variables están relacionadas con las entradas y salidas digitales físicas que se encuentran en la controladora del robot.
- Rapid: Estas variables son las definidas directamente en el código de Rapid y deben ser de tipo persistente.

Para poder controlar el movimiento del robot ABB IRB140 se han establecido dos variables de comunicación que se deben crear tanto en el cliente OPC que será el aplicativo desarrollado en Python como en el programa que se ejecutara en la controladora IRC5 y que es a la que accederá el programa ABB IRC5 OPC Server

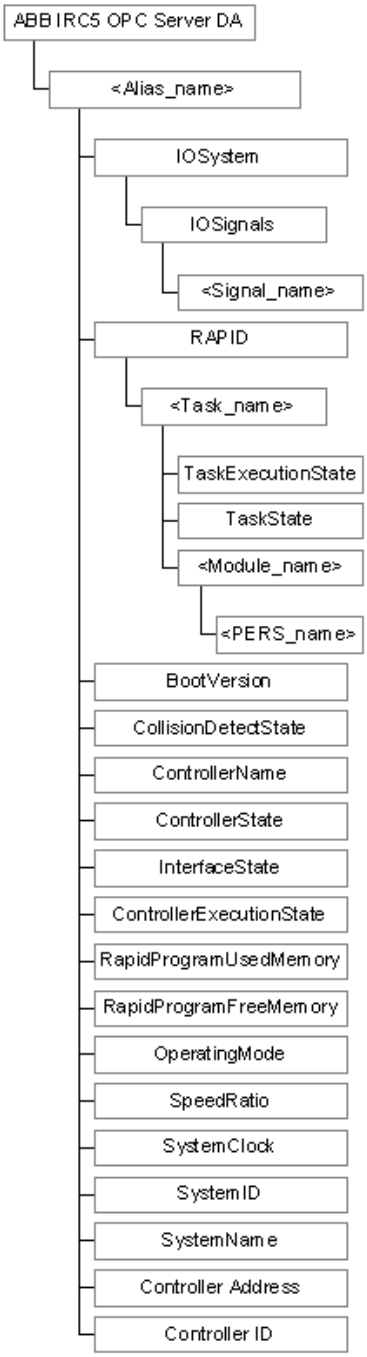

Figura 58. Direcciones de acceso de memoria en la controladora IRC5

A continuación, en la Tabla 11 se describen las variables de comunicación utilizadas:

Comando	Función
START	Esta variable es del tipo binario e indica al robot si debe o no empezar el movimiento. Si esta variable se encuentra en un valor FALSE, el robot permanecerá en el lugar donde se encuentra sin realizar ningún movimiento hasta que esta variable cambie a un valor TRUE
MODE	Esta variable es del tipo entero, e indica que tipo de movimiento debe realizar el robot. El aplicativo de reconocimiento de voz escribirá un valor dependiendo de el comando que se ordene y lo escribirá en la controladora del robot. En conjunto con la variable START indican al robot que este se debe empezar a mover y que tipo de movimiento debe realizar

Tabla 11. Variables de comunicación

Al integrar estas dos variables al sistema es posible establecer una comunicación completa para el movimiento del robot en base a ordenes generadas desde el aplicativo. La configuración de las variables se debe realizar tanto en el programa que ejecutara el Rapid, como en el programa que se ejecutara en Python por lo que a continuación se describe como se debe realizar este proceso.

Con estas las variables se ha logrado una comunicación cuyo funcionamiento se detalla a continuación, y que ha servido para interactuar entre el programa Python y el código en Rapid ejecutado en la controladora del robot ABB IRB140.

- El sistema se inicia al decir la palabra clave “OK ROBOT” y esto causa que arranque la conexión y comunicación con el servidor ABB OPC IRC5
- Al iniciar el sistema, la aplicación escribe en la variable START el valor de TRUE. Este cambio en el valor de la variable START causa que el robot empiece a leer la variable MODE esperando que este cambie su valor de “0” que se encuentra por defecto.
- El aplicativo reconoce un comando de voz y compara con una de las opciones programadas
- Si el comando de voz reconocido coincide con una de las opciones programadas, escribe el valor asignado para esta opción en la variable MODE.
- El estado de la variable START en “TRUE” y de la variable MODE a un valor diferente de “0” causa que el programa del robot ejecute una de las trayectorias programadas que coincide con el comando de voz reconocido.
- Una vez realizado el movimiento, la variable MODE vuelve al valor de “0” dejando al robot en estado de espera hasta que se escriba otro comando.
- Si se escribe otro comando desde la aplicación de voz, el robot ejecuta el movimiento y se repite la secuencia descrita anteriormente
- Si no se escribe ningún comando el robot queda en estado de espera

- Si el software de reconocimiento de voz cambia el valor de la variable STOP a “TRUE”, el robot detiene su movimiento en el punto que se encuentre y será necesario reiniciar el sistema colocando la variable STOP en “FALSE “para ejecutar nuevamente movimientos con el robot.
- Si cuando el robot se encuentra en espera de un comando de la variable MODE se usa el comando por voz “EXIT” el robot se desplaza hacia la posición home y se termina la comunicación con el servidor y el aplicativo de Python

Las descripciones de los estados del sistema se muestran en la Tabla 12.

Variable	Estado	Función
START	FALSE	Inicio de la comunicación por el OPC
START	TRUE	Fin de la comunicación por el OPC
MODE	0	Robot sin movimiento
MODE	1	Inicio del movimiento del robot
MODE	2	Ir a la posición HOME
MODE	3	Ir a la posición ARRIBA
MODE	4	Ir a la posición ABAJO
MODE	5	Ir a la posición DERECHA
MODE	6	Ir a la posición IZQUIERDA
MODE	7	Ir a la posición ADELANTE
MODE	8	Ir a la posición ATRAS
MODE	9	Ir a la posición PUNTO A
MODE	10	Ir a la posición PUNTO B
MODE	11	Ir a la posición PUNTO C

Tabla 12. Descripción estado de variables de comunicación

Al utilizar una única variable como lo es MODE para determinar el tipo de movimiento que el robot debe ejecutar se simplifica el proceso para extender las funciones de movimiento, ya que si se quiere programar otro comando solamente se debería asignar un nuevo número a MODE para el nuevo comando y generar la trayectoria en Rapid

3.2.4.1. Configuración de las variables de estado en Rapid

Existen dos tipos de variables que pueden ser utilizadas por el servidor OPC, las unas son entradas y salidas digitales que se encuentran físicamente en la controladora y la otra son variables de Rapid que se definen directamente en el código de programación del robot.

Para el presente proyecto se utilizarán variables internas definidas en el código de Rapid ya que no se requiere manipular ninguna salida o entrada digital de la controladora, aunque esto podría ser interesante en futuros proyectos ya que abre la posibilidad de generar comunicación con entradas de

información como sensores o estados del proceso y salidas como indicadores luminosos o alarmas entre otros. Además, al tener una comunicación directa con el estado del robot y de medios externos se podría generar interfaz humano maquina en código Python, capaces de monitorizar todo el proceso productivo de una estación robótica con varios elementos.

Al definir una variable para la comunicación con el servidor ABB OPC IRC5, esta variable debe ser del tipo "PERSISTENT", esto es debido a que los estados del robot no pueden cambiar cada vez que se ejecuta el código sino deben mantenerse en sus valores a menos que algún ente externo o explícitamente en el código se cambie su valor.

El servidor OPC IRC5 puede leer o escribir cualquier dato de RAPID siempre y cuando este sea del tipo persistente (PERS). Los controladores IRC5 están limitados a 200 suscripciones en datos de RAPID por robot. Este límite se aplica a todos los procesos que usen datos en un robot. Esto significa que, si otro proceso se ha suscrito a 100 elementos de un robot, el servidor OPC no puede suscribirse a más de 100 elementos de datos de RAPID adicionales en el mismo controlador.

En el caso del código del robot se han definido las tres variables de comunicación como se muestra en la Figura 59.

```
4    PERS num start;
5    PERS num mode;
6    PERS num stop;
```

Figura 59. Definición de las variables tipo PERS

Es importante en este punto comprobar que las variables creadas se encuentran en el servidor OPC, por lo que se utiliza una herramienta de cliente OPC como Quick Client de KepServer como se muestra en la Figura 60, esto ayuda a verificar que se ha configurado bien la comunicación.

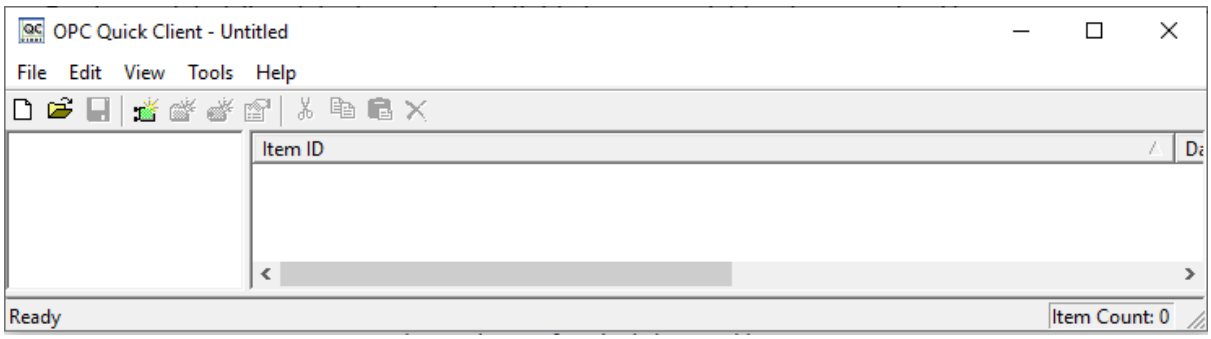

Figura 60. Quick Client de Kepserver

Para establecer la comunicación con un servidor OPC se deben buscar los disponibles al presionar el botón “New Server” mostrado en la Figura 61.

Figura 61. Botón para crear una nueva comunicación con un servidor OPC "New Server"

Este botón new server abrirá un cuadro de diálogo mostrado en la Figura 62 donde se pueden explorar todos los servidores OPC disponibles en el sistema tanto de manera local como remota. Se debe seleccionar el servidor correspondiente a ABB IRC5 OPC Server, lo que crea una nueva conexión que se muestra en la pantalla de inicio del programa.

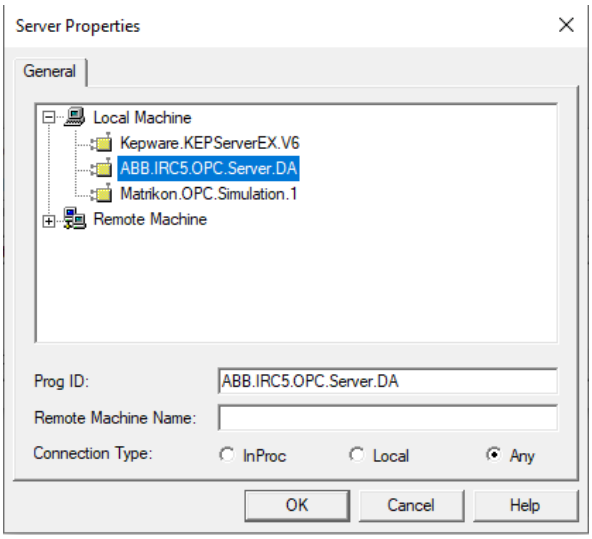

Figura 62. Servidores OPC disponibles en la red

En la conexión creada se debe seleccionar la opción “Add Items”, lo que abrirá la ventana de dialogo de la Figura 63 y permitirá explorar las direcciones de donde se encuentran las variables definidas en Rapid. Si estas se quieren monitorizar por medio del cliente OPC se deben agregar a la conexión con el botón “Add Leaves”

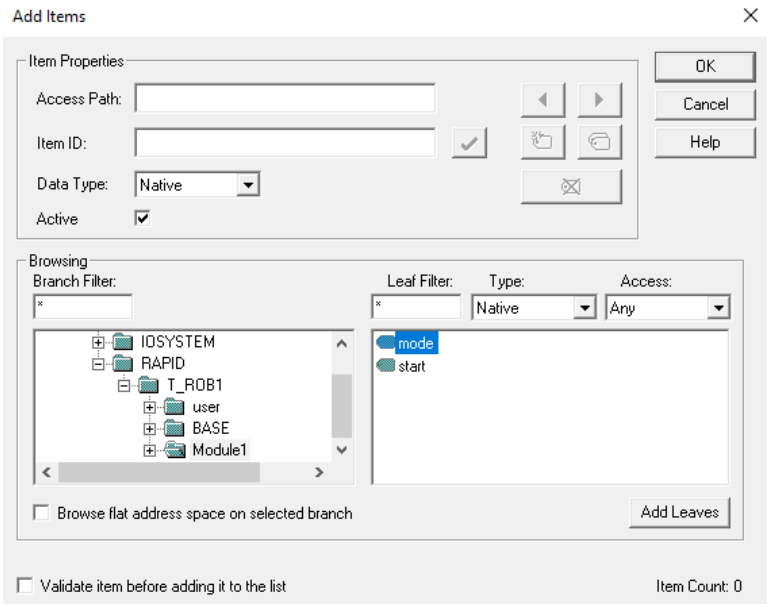

Figura 63. Búsqueda de variables de Rapid en el servidor OPC

El resultado del procedimiento anterior es obtener las variables y sus direcciones como se muestran en la Figura 64, además estas variables pueden ser modificadas por medio del cliente OPC con la finalidad de realizar pruebas de funcionamiento del sistema.

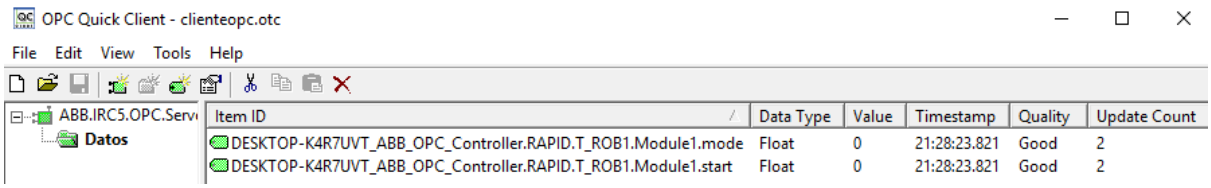

Figura 64. Variables de comunicación en el cliente OPC

Es aconsejable realizar este procedimiento ya que, si no se conoce el nombre y ubicación de los tags, no se podría acceder a ellos, pero aquí se muestran los nombres de cada uno de ellos en el apartado "Item ID" además de información como el tipo de dato, el valor, la fecha que se ha modificado, y la calidad de la señal. El valor de "Item ID" es con el que el cliente OPC de Python va a poder acceder, escribir, y leer en las variables dentro de la memoria del controlador.

Al lograr establecer la comunicación con el servidor ABB OPC IRC5 por medio de un cliente OPC se puede verificar que se ha configurado de manera exitosa la conexión del robot con el servidor OPC. La siguiente etapa de este proceso es configurar el cliente OPC de Python para que la aplicación de reconocimiento de voz pueda leer y escribir directamente en el controlador del robot.

3.2.4.2. Configuración de variables de estado en Python

Una vez establecido Python como un cliente OPC, se puede acceder a las variables creadas en el lenguaje de programación Rapid y que se ejecutan en la controladora del robot desde el aplicativo de reconocimiento de voz. Esto se hace en base a los nombres obtenidos desde el programa Quick Client de Kepserver, ya que los nombres que se muestran ahí son las direcciones de memoria del servidor OPC.

Los nombres de los tags usados para la comunicación se muestran en la Tabla 13.

Variable	Dirección
MODE	DESKTOP-K4R7UVT_ABB_OPC_Controller.RAPID.T_ROB1.Module1.mode
START	DESKTOP-K4R7UVT_ABB_OPC_Controller.RAPID.T_ROB1.Module1.start

Tabla 13. Direcciones de las variables de comunicación en el servidor OPC

Con estas direcciones desde Python se puede utilizar las funciones de escritura y lectura de OpenOPC para enviar información desde el aplicativo de reconocimiento de voz hasta la controladora del robot. Para lograr esta comunicación se usa los comandos descritos a continuación.

Lectura de variables

$$>>> \ opc.read('DESKTOP - K4R7UVT\_ABB\_OPC\_Controller.RAPID.T\_ROB1.Module1.mode')$$

$$>>> \ opc.read('DESKTOP - K4R7UVT\_ABB\_OPC\_Controller.RAPID.T\_ROB1.Module1.mode')$$

$$>>> \ opc.read('DESKTOP - K4R7UVT\_ABB\_OPC\_Controller.RAPID.T\_ROB1.Module1.mode')$$

Escritura de variables

$$>>> \ opc.write(('DESKTOP - K4R7UVT\_ABB\_OPC\_Controller.RAPID.T\_ROB1.Module1.mode', 0))$$

$$>>> \ opc.write(('DESKTOP - K4R7UVT\_ABB\_OPC\_Controller.RAPID.T\_ROB1.Module1.start', 0))$$

$$>>> \ opc.write(('DESKTOP - K4R7UVT\_ABB\_OPC\_Controller.RAPID.T\_ROB1.Module1.stop', 0))$$

Con estos comandos, el aplicativo es capaz de escribir y leer en las variables definidas en el código de programación en Rapid que se ejecutan en la controladora y se competa la configuración de la comunicación.

3.3. Programación robot ABB IRC 140

Para el funcionamiento del proyecto, es necesario desarrollar un programa en Rapid que es el lenguaje de programación de la marca ABB para sus robots. Este programa debe ser capaz de leer instrucciones externas a la controladora IRC5 e interpretarlas para poder dar una respuesta que en este caso será generar algún tipo de movimiento.

3.3.1. Funcionamiento RobotStudio

El programa está compuesto por un conjunto de instrucciones que describen la actividad del robot. Por tanto, existen instrucciones específicas para los distintos comandos, por ejemplo, una para mover el robot, otra para seleccionar una salida.

Por lo general, las instrucciones llevan asociado un conjunto de argumentos que definen qué debe ocurrir con una instrucción concreta Estos argumentos pueden especificarse mediante uno de los métodos siguientes:

- Un valor numérico, por ejemplo 5 ó 4.6
- Una referencia a un dato, por ejemplo, reg1
- Una expresión, por ejemplo 5+reg1*2
- Una llamada a una función, por ejemplo, Abs(reg1)
- Un valor de cadena de caracteres, por ejemplo "Producción de la pieza A"

3.3.1.1. Tipos de rutinas

En un programa en Rapid, se pueden usar tres tipos de rutinas.

- Los procedimientos: Estos son utilizados como subprogramas.
- Las funciones: Estas devuelven un valor de un tipo concreto y se utilizan como argumento de una instrucción.
- Las rutinas TRAP: Estas proporcionan una forma de responder a las interrupciones. Las rutinas TRAP pueden asociarse con una interrupción determinada. Por ejemplo, al establecer una entrada, se ejecuta automáticamente si se produce dicha interrupción en concreto.

3.3.1.2. Tipos de variables

También existen tres tipos de variables que son las siguientes.

- Constantes: Estas variables representan valores fijos y la única forma de asignarles un nuevo valor es manualmente.
- Variables: La asignación de nuevos valores a los datos variables puede realizarse durante la ejecución del programa.
- Persistente: Un valor persistente puede describirse como una variable “persistente”. Cuando se guarda un programa, el valor de inicialización corresponde al valor actual del valor persistente.

3.3.1.3. Características Rapid

Por último, el lenguaje de programación Rapid cuenta con otras características que se muestran a continuación.

- Parámetros de rutinas
- Expresiones aritméticas y lógicas
- Gestión automática de errores
- Programas modulares
- Multitasking

3.3.1.4. Flujo del programa

Al estructurar un programa en Rapid se debe tener en cuenta cómo funciona el flujo con el cual se ejecutan las instrucciones. El programa se ejecuta secuencialmente como una regla, es decir, una instrucción tras otra. En ocasiones, se requieren instrucciones que interrumpen esta ejecución secuencial y que llaman a otra instrucción, para enfrentarse a las distintas situaciones que pueden darse durante la ejecución

El flujo del programa puede controlarse acorde con cinco principios diferentes:

- Llamar a otra rutina (procedimiento) y, una vez ejecutada dicha rutina, continuar la ejecución con la instrucción que sigue a la llamada a la rutina.
- Ejecutar instrucciones diferentes en función de si se cumple o no una condición determinada.
- Repetir una secuencia de instrucciones un número determinado de veces o hasta que se cumple una condición determinada.
- Ir a una etiqueta dentro de la misma rutina.
- Detener la ejecución del programa.

Una vez entendidos estos principios básicos de programación del lenguaje Rapid, y con ayuda de los comandos proporcionados por el manual de referencia de instrucciones se procede con el desarrollo del programa.

3.3.1.5. Variable Robtarget

Lo primero que se realiza en el desarrollo de un programa en Rapid, es detallar los puntos a los cuales se desea que el robot se mueva. Estos puntos serán constantes y variables ya que algunos no necesitan modificarse a lo largo de la ejecución del programa, pero otros sí.

Los datos de posición se utilizan en las instrucciones de movimiento para indicar la posición hacia la que deben desplazarse los ejes del robot y los ejes externos. Un punto de trabajo del robot se define como un dato del tipo RobTarget.

Debido a que el robot puede alcanzar una misma posición con métodos diferentes, también se especifica la configuración de los ejes. De esta forma, se definen los valores de los ejes si por algún motivo resultan ambiguas, por ejemplo, en los casos siguientes:

- Si el robot se encuentra en una posición avanzada o retrasada
- Si el eje 4 está orientado hacia abajo o hacia arriba
- Si el eje 6 se encuentra en una revolución negativa o positiva

Una variable RobTarget cuenta con toda la información necesaria para que el robot se posicione en un lugar, con cierto ángulo y una configuración. A continuación, se detalla cómo se estructura.

Traslación: [trans]

Tipo de dato: pos

La posición (x, y, z) del punto central de la herramienta, expresado en mm.

La posición se especifica respecto del sistema de coordenadas del objeto actual, incluido el desplazamiento de programa. Si no se ha especificado ningún objeto de trabajo, se utiliza el sistema de coordenadas mundo.

Rotación: [rot]

Tipo de dato: orient

La orientación de la herramienta, expresada en forma de un cuaterniones (q1, q2, q3 y q4).

La orientación se especifica respecto del sistema de coordenadas del objeto actual, incluido el desplazamiento de programa. Si no se ha especificado ningún objeto de trabajo, se utiliza el sistema de coordenadas mundo.

Configuración del robot: [robconf]

Tipo de dato: confdata

La configuración de ejes del robot (cf1, cf4, cf6 y cfx). Esto se define en forma del cuarto de revolución actual de los ejes 1, 4 y 6. El primer cuarto de revolución positivo, de 0 a 90° se define como 0. El significado del componente cfx depende del tipo de robot.

Ejes externos: [extax]

Tipo de dato: extjoint

La posición de los ejes externos.

La posición se define de la forma siguiente para cada eje independiente (eax_a, eax_b...eax_f):

Para los ejes de rotación, la posición se define como la rotación en grados de la posición de calibración. Para los ejes lineales, la posición se define como la distancia en mm existente respecto de la posición de calibración.

3.3.2. Programación del robot

3.3.2.1. Lógica de programación

El propósito del programa que ejecuta el robot, es el poder leer información externa desde el aplicativo de reconocimiento de voz, la cual indica que tipo de movimiento debe realizar y cuando debe empezar estos movimientos. Para lograr coordinar el movimiento del robot con las órdenes dadas por el aplicativo se utiliza las variables de comunicación definidas en la sección de configuración del servidor OPC.

Una vez empieza a ejecutarse el programa en Rapid, este entra en un ciclo donde espera que la variable “start” cambie su valor. Un valor de False en la variable “start” indica que el robot debe permanecer detenido, y un valor de True significa que el movimiento debe empezar.

Una vez que el valor de “start” se pone en True, el algoritmo lee la variable “mode” que es la encargada de informar que tipo de movimiento debe realizar el robot y ordena a esta ejecutar una función específica asignada para cada caso o valor de la variable “mode”.

```
40          IF start=1 THEN
41              TPWrite "Reconocimiento de voz iniciado";
42              !Comando stop para parar el robot
43              ciclorobot:
44              !Posicion HOME
45              IF mode=2 THEN
46                  go_home;
47                  GOTO ciclorobot;
48                  !Posicion UP
49              ELSEIF mode=3 THEN
50                  go_up;
51                  GOTO ciclorobot;
52                  !Posicion DOWN

PROC go_point_a()
    MoveJ Target_10,v1000,z10,tool0\WObj:=wobj0;
ENDPROC
```

Figura 65. Opciones de movimiento del robot en Rapid

La Figura 65 muestra un segmento del código del aplicativo en Rapid, donde se toman las decisiones sobre que movimiento debe realizar. En este caso se observa que, si la variable mode es igual a 2, el robot debe ejecutar la secuencia go_home que envía al robot a la posición inicial de movimiento.

3.3.2.2. Configuración de puntos de trabajo

El primer punto que se define es la posición Home mostrado en la Figura 66, esta posición es en la que la trayectoria del robot comienza al arrancar el aplicativo y en la que termina al finalizar el aplicativo. el siguiente comando es la definición del punto de trabajo Home.

```
CONST robtarget home:=[[515,0,712.000000147],[0.707106782,0.000000001,0.70710678,0.000000001],
                [0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Figura 66. Punto de trabajo Home

Adicionalmente a este punto, se define otros utilizados para verificar el campo de movimiento del robot, estos puntos han sido verificados por medio del uso de la Flex Pendant, y están verificados en el área de trabajo del laboratorio donde se realiza este proyecto. Se puede crear tantos puntos como se necesite, pero en la Figura 67 solamente se detalla 5 de ellos para entender cómo se utilizan.

```
!Puntos fijos de trabajo
CONST robtarget Target_10:=[[320,-200,210],[0,0,1,0],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Target_20:=[[320,395,210],[0,0,1,0],[0,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Target_30:=[[700,395,210],[0,0,1,0],[0,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Target_40:=[[700,-200,210],[0,0,1,0],[-1,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Target_50:=[[500,0,500],[0,0,1,0],[-1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Figura 67. Puntos fijos de trabajo del robot

Además de las variables constantes, se necesita definir puntos RobTarget del tipo variables ya que estos puntos no son fijos en la ejecución del programa y se ira asignando valores diferentes a ellos a medida que el aplicativo funcione. La Figura 68 muestra los puntos definidos en él programa.

```
!Puntos variables para movimiento
VAR robtarget moveup;
VAR robtarget movedown;
VAR robtarget moverigth;
VAR robtarget moveleft;
VAR robtarget movefront;
VAR robtarget moveback;
```

Figura 68. Puntos variables de trabajo del robot

3.3.2.3. Configuración de funciones

El programa dispone de funciones de movimiento a puntos fijos, que solamente cuentan de una instrucción como es el caso de la función go_point_a que se muestra en la Figura 69. Esta función mueve el robot al punto "Target_10" y lo deja posicionado en ese lugar hasta que se ejecute algún otro comando. El punto al que se mueve el robot puede ser modificado únicamente desde la programación del robot, y no mientras se está ejecutando el programa.

```
PROC go_point_a()
    MoveJ Target_10,v1000,z10,tool0\WObj:=wobj0;
ENDPROC
```

Figura 69. Función "go point a'

En el programa también se tienen funciones de mayor complejidad, estas son movimientos por trayectorias cerradas como por ejemplo la función go_recognize que se muestra en la Figura 70. Esta función hace que el robot recorra una trayectoria cerrada desde el punto "home" por 4 puntos fijos y vuelva a home al finalizar.

```
PROC go_recognize()
    MoveJ home,v1000,z10,tool0\WObj:=wobj0;
    MoveJ Target_10,v1000,z10,tool0\WObj:=wobj0;
    MoveJ Target_20,v1000,z10,tool0\WObj:=wobj0;
    MoveJ Target_30,v1000,z10,tool0\WObj:=wobj0;
    MoveJ Target_40,v1000,z10,tool0\WObj:=wobj0;
    MoveJ home,v1000,z10,tool0\WObj:=wobj0;
ENDPROC
```

Figura 70. Función go_recognize

La trayectoria de la función go_recognize se muestra en la Figura 71.

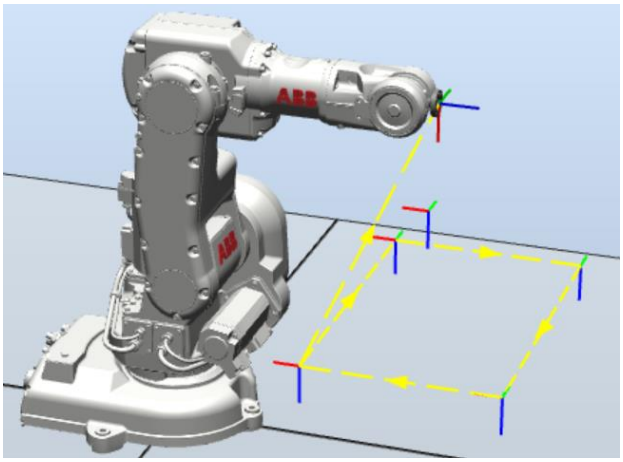

Figura 71. Trayectoria de la función go_recognize

Adicional a las otras funciones, el programa también ejecuta movimientos relativos al punto donde está ubicado en un determinado momento. Estas funciones leen la posición donde se encuentra el robot, modifica uno de los parámetros de posición como la posición en los ejes (x,y,z), y envía el robot hacia la nueva posición calculada.

Una de las funciones de movimiento a posiciones variables está mostrada en la Figura 72, la función go_up mueve el robot 2 mm arriba de la posición actual. Esta función obtiene la información del RobTarget actual del robot, modifica el parámetro de la posición en z, y envía el robot hacia allá.

```
PROC go_up()

    moveup:=CRobT();
    moveup.trans.z:=moveup.trans.z+20;
    If IsReachable(moveup,tool0,wobj0) THEN
        MoveJ moveup,v100,z10,tool0\WObj:=wobj0;
        WaitTime 1;
    ENDIF

ENDPROC
```

Figura 72. Función go_up

El comando CRobT() sirve para leer la posición actual, y esta se almacena en la variable llamada moveup. Luego en esta variable se modifica el valor del eje z lo que se hace con el comando moveup.trans.z. En este punto se usa la función IsReachable que será descrita más adelante para verificar si la nueva posición es alcanzable con la actual configuración del robot, y si es alcanzable se envía el robot hacia el nuevo punto calculado.

La función IsReachable sirve para determinar si los puntos variables pueden ser alcanzados por el robot en la configuración actual, esto significa si el robot puede llegar a posicionarse en las coordenadas especificas en la forma que se indica. El código de la función IsReachable se encuentra en la Figura 73.

```
FUNC bool IsReachable(robtarget pReach,PERS tooldata ToolReach,PERS wobjdata WobjReach)

    VAR bool bReachable;
    VAR jointtarget jntReach;

    bReachable:=TRUE;

    jntReach:=CalcJointT(pReach,ToolReach\Wobj:=WobjReach);

    RETURN bReachable;

ERROR
    IF ERRNO=ERR_ROBLIMIT THEN
        bReachable:=FALSE;
        TRYNEXT;
    ENDIF
ENDFUNC
```

Figura 73. Función IsReachable

La función se alimenta del punto que se quiere verificar, la herramienta que se está usando, y el sistema de coordenadas. Si esta función devuelve el valor de True, entonces el punto es alcanzable, pero si devuelve un valor False, no lo es y no se debe enviar al robot a moverse hacia este nuevo punto porque se genera un error.

3.3.3. Funcionamiento del aplicativo en Rapid

El aplicativo desarrollado en Rapid, integra todas las funciones descritas en el apartado anterior y tiene por objetivo que el robot realice los movimientos indicados por la aplicación de reconocimiento de voz. Para este propósito, el programa sigue la lógica descrita a continuación.

1. Inicia el aplicativo
2. Lee la variable “start”, si el valor es start=0 el robot no realiza ningún movimiento y se repite el ciclo de lectura de “start”, pero si el valor es start=1 entra en un bucle interno.
3. En el bucle interno, se lee la variable “mode”, y se compara con las opciones programadas. Si encuentra un valor de la variable “mode” que haya sido preestablecido, ejecuta la función correspondiente ese valor.

4. La función de movimiento evalúa si el punto al que se está enviando al robot es alcanzable con su configuración actual, y en caso de ser alcanzable realiza el movimiento. Caso contrario se termina el aplicativo para evitar posibles fallas.
5. Una vez terminado el movimiento, se repite el bucle exterior esperando que se le indique nuevamente otro comando y la variable “start” vuelva a tener el valor de 1.

La lógica descrita en el apartado anterior, se puede observar en el diagrama de flujo del programa en la Figura 74.

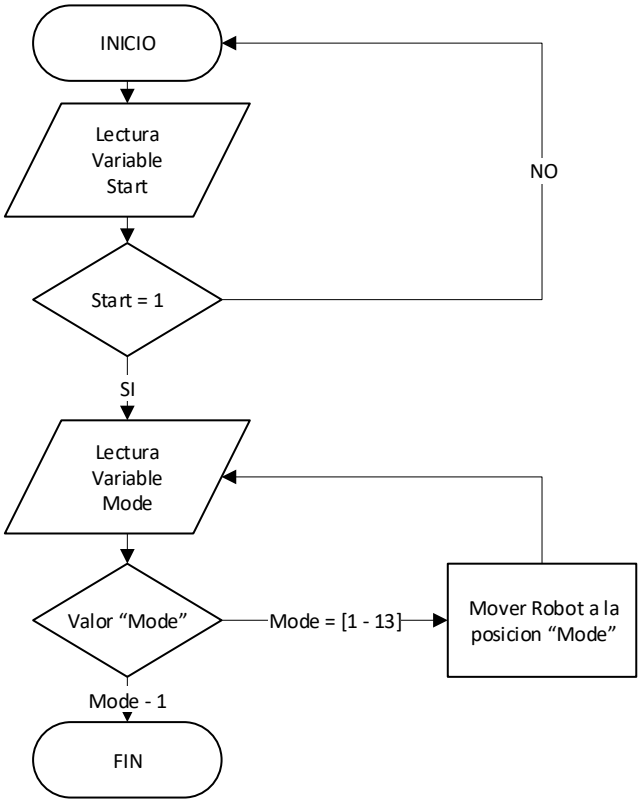

Figura 74. Diagrama de flujo del funcionamiento del aplicativo de movimiento del robot

6. RESULTADOS Y TRABAJO FUTURO

6.1. RESULTADOS

Durante el desarrollo de este proyecto se ha llevado a cabo el control de movimiento de un robot ABB IRB140 por medio de un aplicativo de reconocimiento de voz creado en el lenguaje Python. Para ello se ha utilizado diversas herramientas típicas de la robótica industrial.

6.1.1. Aplicativo de reconocimiento de voz en Python

El aplicativo de reconocimiento de voz, funciona correctamente, pero presenta ciertas características que son condicionantes al momento de usarlo en un entorno industrial. A continuación, se detallan los problemas encontrados y las soluciones dadas.

6.1.1.1. Inicio del aplicativo

Al iniciar el aplicativo, este empezaba inmediatamente escuchar cualquier sonido en el entorno, esto causa un problema de seguridad ya que si alguien se encuentra cerca del micrófono puede provocar un movimiento no deseado del robot.

Para evitar que el aplicativo reconozca comandos no deseados, el operario debe decir una palabra clave que inicie la comunicación con el robot. Esta palabra clave es “OK ROBOT”

6.1.1.2. Idioma de reconocimiento de voz

El lenguaje de reconocimiento que se ha utilizado en este proyecto es el inglés, ya que es el que presenta mayor soporte en la API de reconocimiento de voz de Google. El uso de este idioma conlleva un problema, y es que si el operario no tiene una buena pronunciación el aplicativo reconoce palabras que no tienen sentido. Por ejemplo, la palabra clave para iniciar el aplicativo es “OK ROBOT”, pero al pronunciar esta expresión generalmente reconoce las palabras “OKEY ROBERT” por lo que en inicio el aplicativo era muy difícil de usar.

Para disminuir el impacto que se tenía al utilizar el idioma inglés para el reconocimiento de voz, se realizó el proceso de entrenamiento, este integra las respuestas más comunes que la API da como respuesta en cada comando, y crea una lista de posibles opciones que corresponden a ese comando. Por ejemplo, las expresiones “okey Robot”, “ok robot”, “okey robert”, “hey robert”, ahora forman el comando “OK ROBOT” y con cualquiera de esas opciones el aplicativo inicia.

6.1.1.3. Lenguaje natural

Uno de los objetivos en las aplicaciones de reconocimiento de voz es el lograr integrar capacidades de lenguaje natural, esto significa integrar al aplicativo la funcionalidad de reconocer varias expresiones como una misma expresión, en inicio el aplicativo no poseía esta habilidad, y era necesario decir el comando exactamente como estaba programado. Por ejemplo, el aplicativo reconocía solamente una orden en específico como el comando “move left” para moverse a la izquierda y si se le daba otra orden similar como “left” “go left” “move to your left” “go to the left”, el aplicativo no las reconocía.

Se implemento un proceso de entrenamiento similar al utilizado para disminuir los efectos del idioma, con esto se integró la capacidad de reconocer varias expresiones como un mismo comando. Por ejemplo, el aplicativo ahora es capaz de reconocer los comandos “left” “go left” “move to your left” “go to the left” como el comando moverse a la izquierda.

6.1.1.4. Ruido ambiental

En un inicio se requería un entorno muy silencioso para que el reconocimiento de voz funcione, esto era debido a que el ruido interfería mucho en la señal de audio de voz y el aplicativo no era capaz de reconocerlo.

Se implemento un filtro de ruido ambiental, este hace que la señal de audio de voz mejore y que la API de Google sea capaz de reconocer las palabras que el usuario dice al aplicativo

6.1.1.5. Interfaz del aplicativo

El interfaz de usuario se muestra en la siguiente imagen

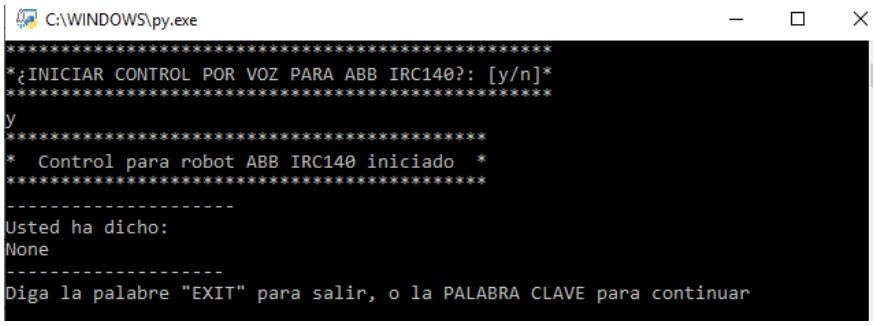

Figura 75. Interfaz del aplicativo de reconocimiento de voz

Para arrancar es necesario ingresar la letra “y” por medio del teclado, de esta manera se inicia el reconocimiento de voz, esto se hace como medida de seguridad para que no se controle por error el robot.

Una vez iniciado el control para el robot ABB IRC140, el aplicativo empieza a reconocer comandos de voz, pero espera a que se diga la palabra clave para iniciar la comunicación por medio del servidor OPC.

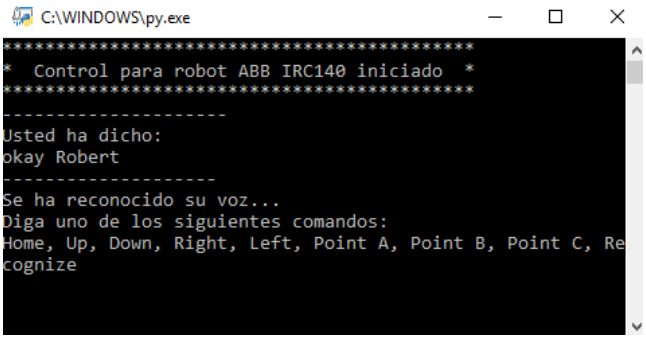

Figura 76. Reconocimiento de palabra clave

Una vez se ha identificado la palabra clave, establece comunicación con la controladora del robot y se puede empezar a controlar el movimiento por medio de comandos de voz.

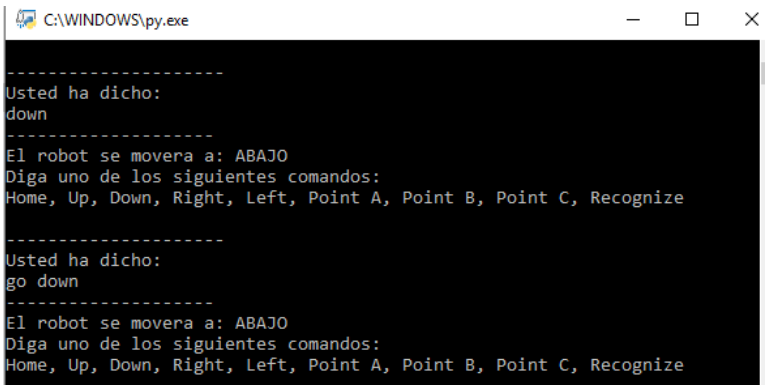

Figura 77. Reconocimiento de comandos

Al decir un comando, el aplicativo procede a enviar la señal a la controladora del robot, además es capaz de reconocer lenguaje natural por lo que existe más de una manera de indicar los comandos. En la imagen se muestra que el comando “down” y “go down” son reconocidos como la función de movimiento hacia abajo.

La interfaz es sencilla, pero muy funcional y se ha realizado pruebas de uso con algunos usuarios. El mejor resultado se ha obtenido cuando la persona que entreno el reconocimiento de voz es la que

utiliza el aplicativo, esto resulta obvio ya que está diseñado para funcionar con esta persona. A pesar de esto, el proceso de entrenamiento es relativamente sencillo, y el sistema puede ser configurado para un nuevo usuario en aproximadamente 23 minutos.

6.1.2. Comunicación OPC

El protocolo de comunicación utilizado (OPC) establecido por medio del software ABB IRC5 OPC SERVER funciona correctamente ya que la aplicación implementada en Python proporciona la información del movimiento que el robot debe realizar, y el robot lo ejecuta.

Fue necesario establecer la comunicación entre la controladora del robot, y el programa ABB IRC OPC SERVER, y luego realizar la comunicación entre el servidor OPC y el programa en Python por medio de un cliente OPC ejecutado en el aplicativo.

Se vio la necesidad de usar dos variables para establecer la comunicación entre la controladora del robot y el aplicativo de reconocimiento de voz, y con estas dos variables se logra enviar toda la información necesaria para poder controlar el movimiento del robot.

6.1.3. Aplicativo de movimiento del robot en Rapid

El robot ejecuta los movimientos que son indicados desde la aplicación de reconocimiento de voz. Se han configurado una variedad de movimientos para ver la funcionalidad del aplicativo, entre estas está el realizar desplazamientos a puntos fijos, realizar desplazamientos a puntos relativos, y realizar movimiento por trayectorias complejas. Esto muestra que el proyecto ha cumplido sus objetivos, e incluso es escalable ya que se pueden programar nuevos movimientos con relativa facilidad.

6.1.3.1. Puntos variables

Al inicio, el movimiento del robot presentaba un problema al desplazarse a puntos variables. Este problema era debido a que al realizar movimientos como por ejemplo “UP” que desplaza el robot hacia arriba en el eje z con movimiento lineal, llega un punto en el que el robot sale del área de trabajo, y la controladora devuelve un código de error indicando que el punto no se puede alcanzar.

Para evitar que esto pase, fue necesario programar una función de verificación para determinar si el robot puede o no alcanzar un cierto punto variable. Ahora antes de realizar un desplazamiento la controladora del robot primero determina si el punto es alcanzable y para generar el movimiento, si el punto no se puede alcanzar el robot no realiza ninguna acción.

6.1.3.2. Trayectorias y puntos de movimiento

El aplicativo de reconocimiento de voz se probó con movimientos a puntos fijos, puntos variables, y trayectorias. Se tienen programados cuatro puntos fijos, que son los que se muestran en la Figura 78, Figura 79, Figura 80, Figura 81. Como se aprecia en las imágenes, se logró alcanzar todos los puntos establecidos en la simulación.

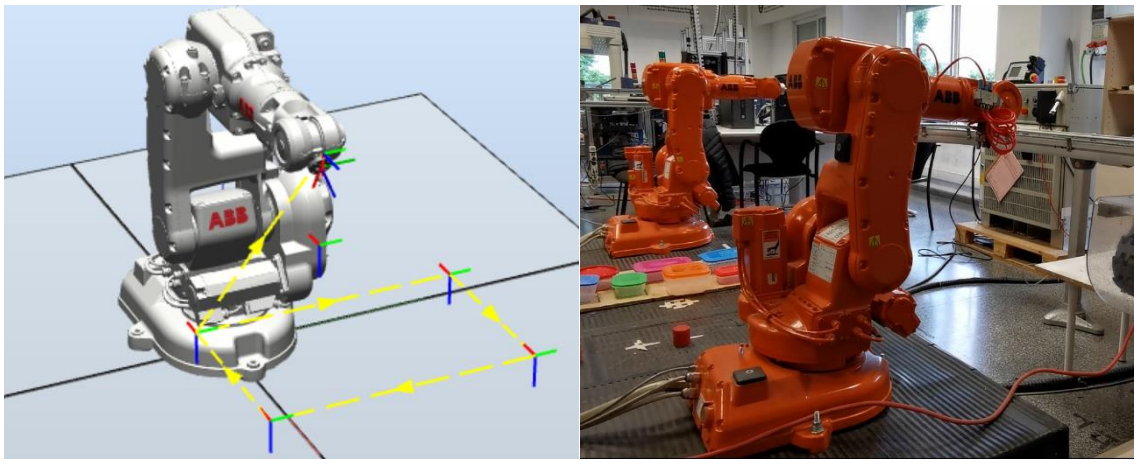

Figura 78. Posición Home

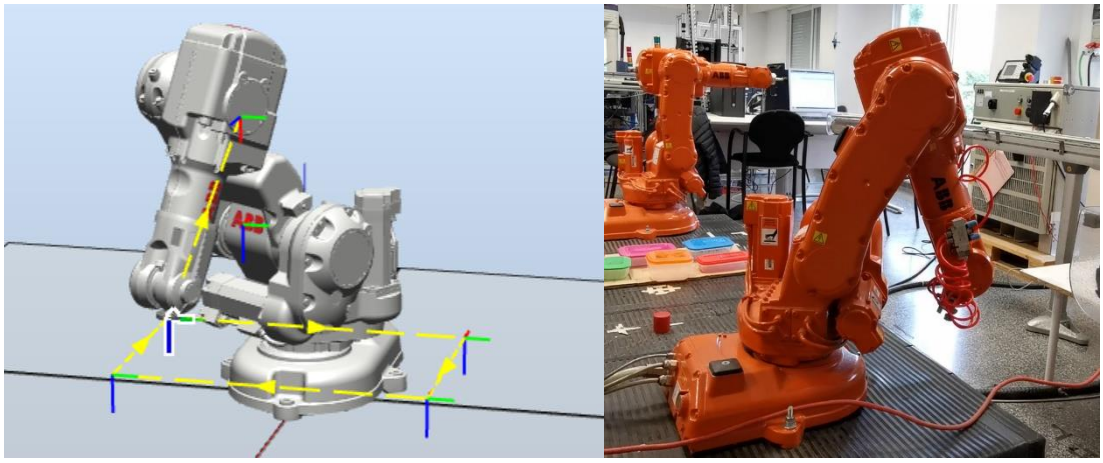

Figura 79. Posición Punto A

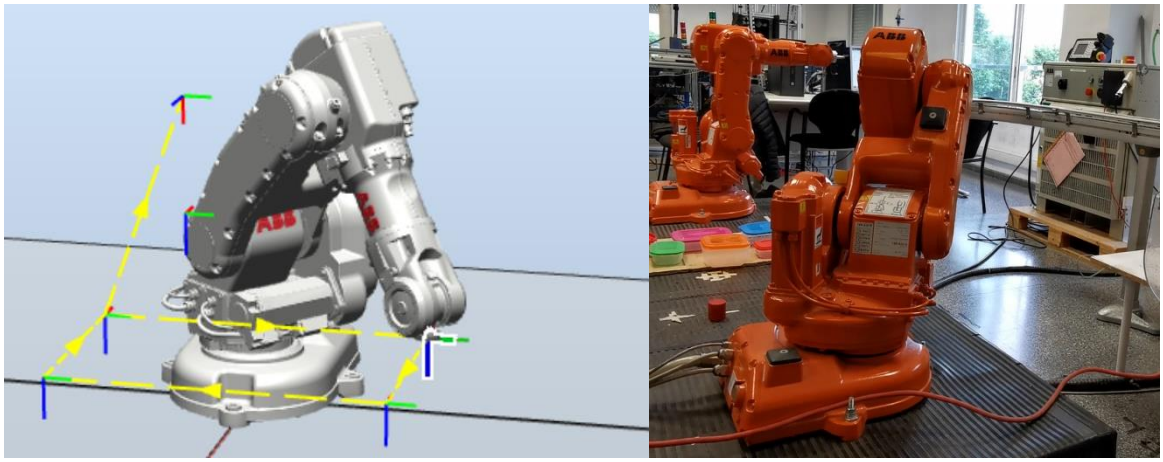

Figura 80. Posición Punto B

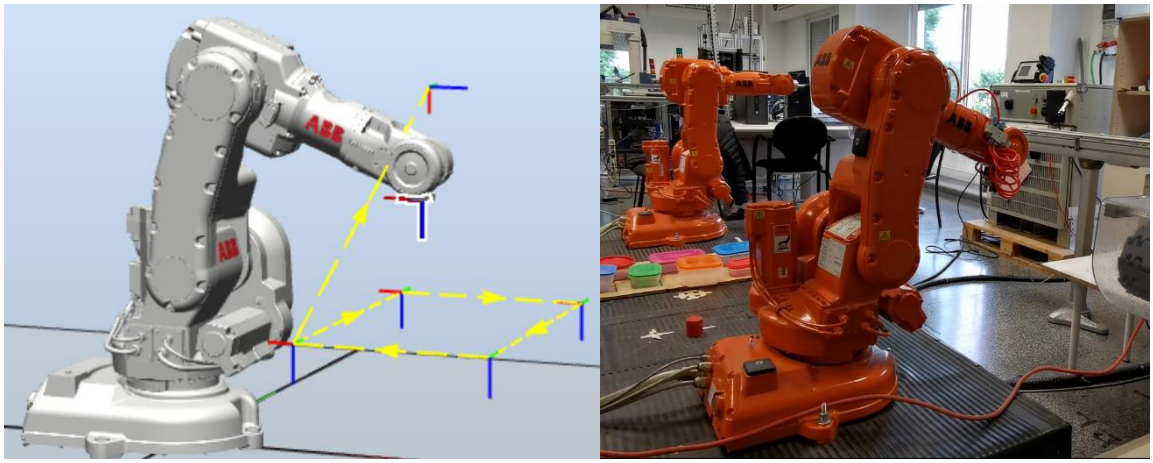

Figura 81. Posición Punto C

Además, se realizó pruebas con trayectorias como la Recognize que envía el robot desde la posición inicial home a cuatro puntos de manera secuencial, y de vuelta a la posición inicial.

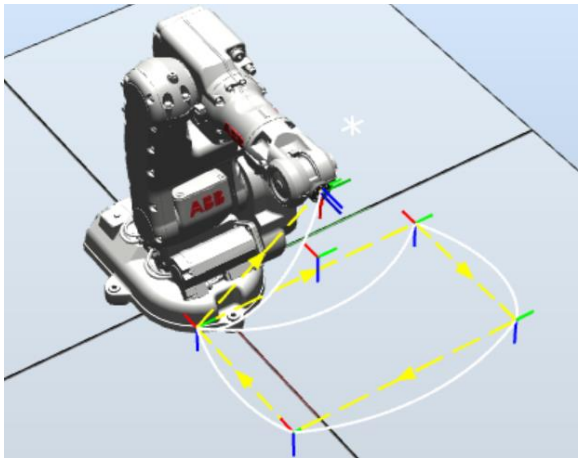

Figura 82. Simulación trayectoria Recognize

6.1.4. Integración del sistema

Para integrar todo el sistema y poder en marcha el aplicativo de control de movimiento de un robot ABB IRB140, lo primero que se hizo fue conectar todos los dispositivos necesarios a la placa Lattepanda como se muestra en la Figura 83. Se conecto teclado, mouse, y un monitor HDMI como en un computador normal. Además, es necesario conectar a la alimentación eléctrica para tener energía. La conexión más importante es la del cable ethernet con la controladora del robot, y el micrófono externo.

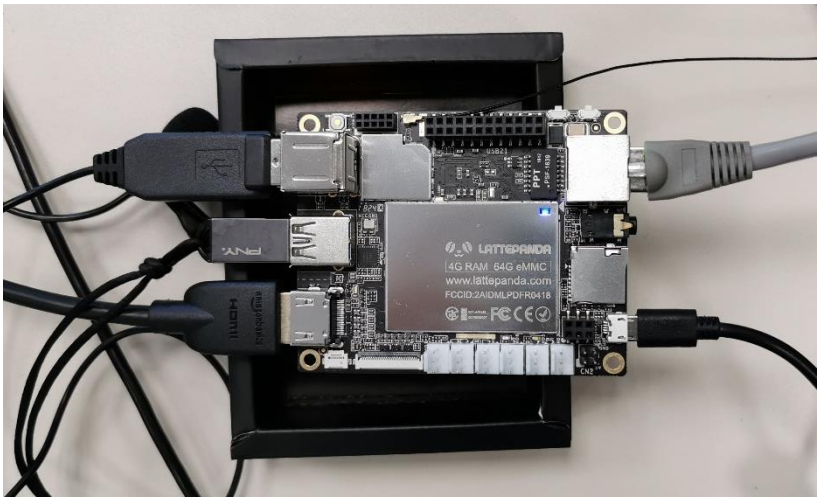

Figura 83. Conexión física de la placa Lattepanda

La configuración de la placa Lattepanda con la dirección IP del controlador del robot se muestra en la Figura 84.

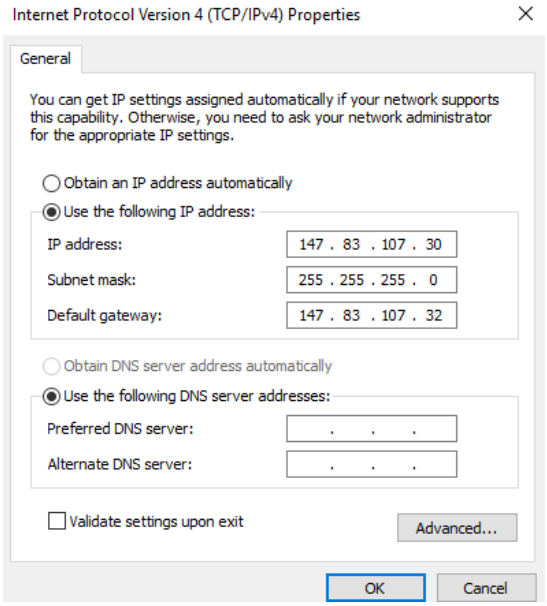

Figura 84. Configuración de la IP para la conexión con el servidor del robot

La configuración del servidor OPC de ABB se muestra en la Figura 85.

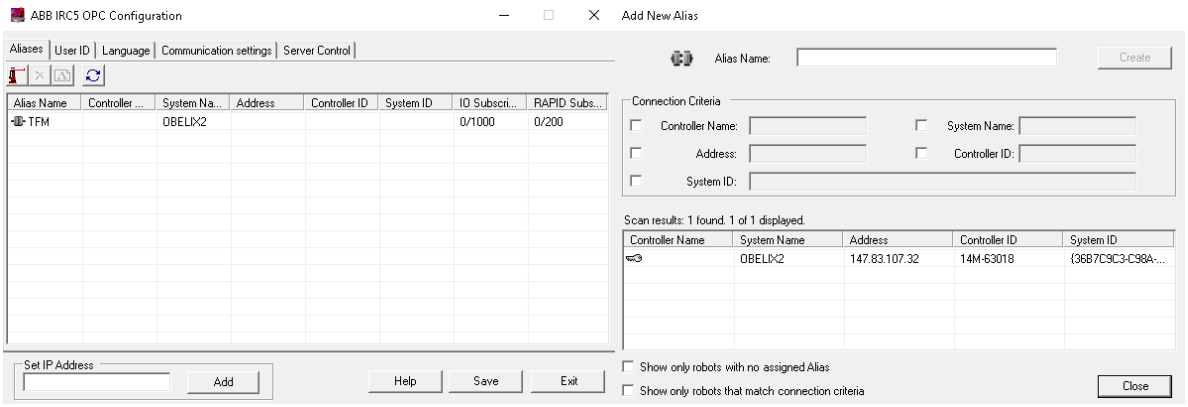

Figura 85. Configuración del servidor OPC para la conexión con el robot Obelix

La Figura 86 muestra el funcionamiento del aplicativo de reconocimiento de voz conectador al servidor del robot, se puede observar las variables de comunicación leídas por el cliente OPC al lado derecho.

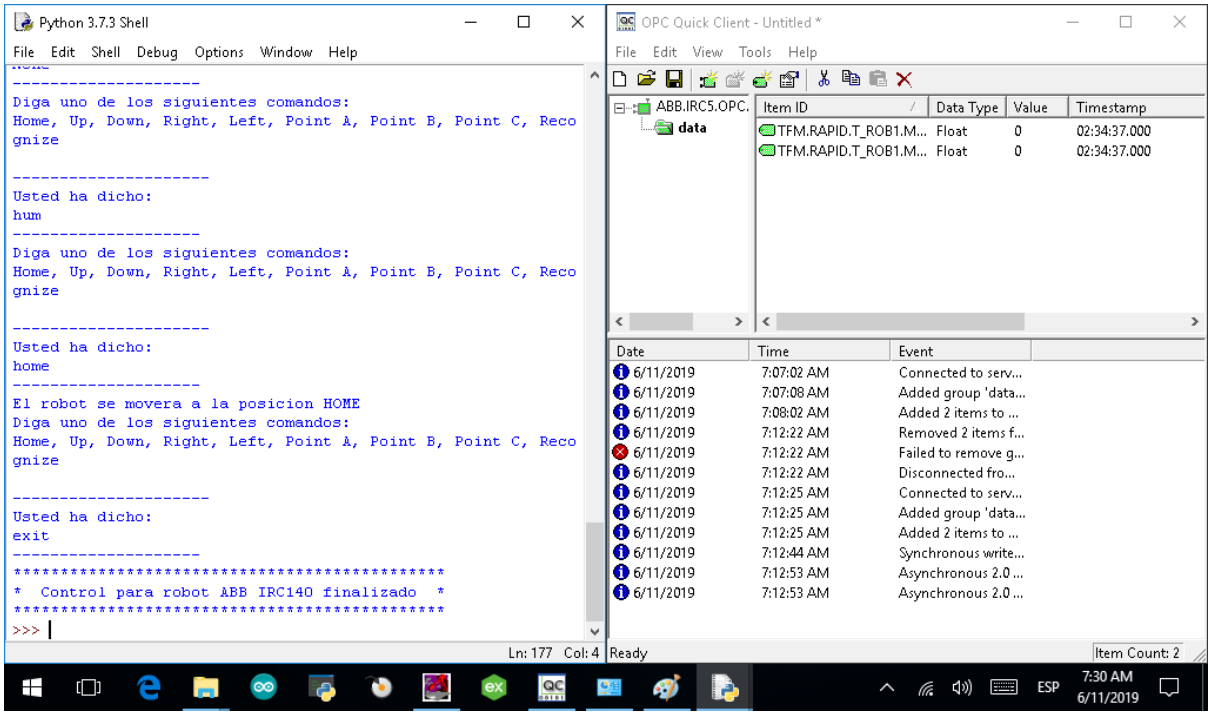

Figura 86. Funcionamiento del aplicativo

En la Figura 87 se puede observar todo el sistema integrado durante su funcionamiento. En el computador se ejecuta el aplicativo de reconocimiento de voz, y la señal de voz es obtenida por medio de un micrófono. El robot realiza los movimientos indicados al hablar.

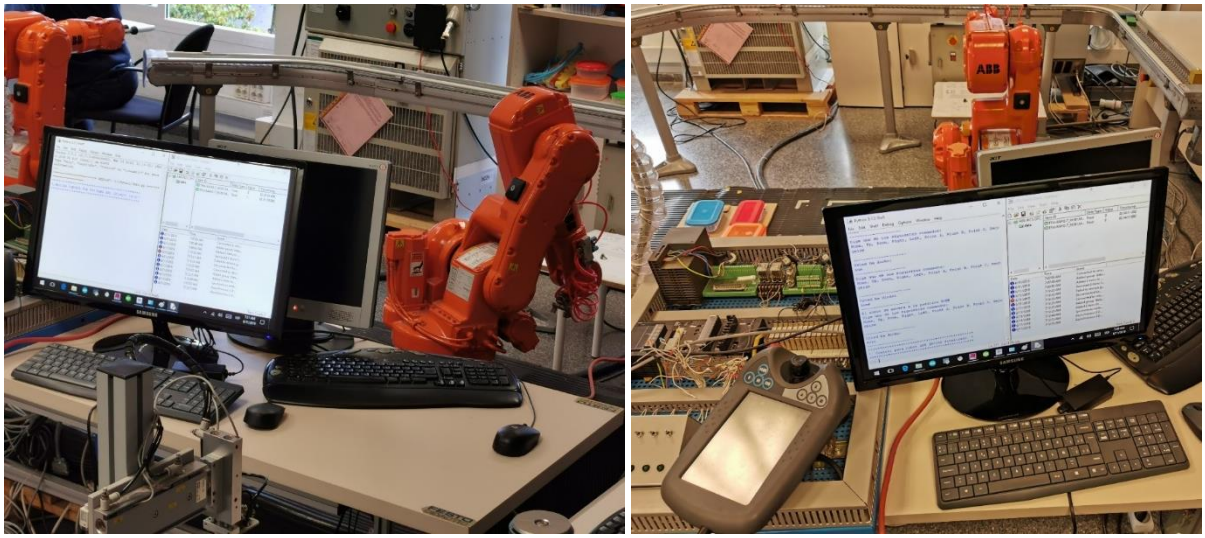

Figura 87. Sistema de control de movimiento con reconocimiento de voz

6.2. Trabajo futuro

En el desarrollo del proyecto de control de movimiento de un robot por medio de voz, se han encontrado ciertos puntos que podrían mejorarse para optimizar el sistema y en algún punto llegar a utilizar esta tecnología para aplicaciones industriales reales. Estos aspectos de mejora pueden ser puntos de partida de futuros estudios y se detallan a continuación.

El uso de la API de Google, tiene un inconveniente y es el tiempo de respuesta del servidor. Generalmente el tiempo que tarda en analizar una orden esta entre 5 y 10 segundos, lo cual causa que el control del robot tenga un retraso. Es por esto que en el presente proyecto no se ha programado un paro de emergencia por voz, ya que por el tiempo de respuesta el robot no alcanzaría a reconocer esta situación. Una mejora interesante sería el trabajar con sistemas de reconocimiento de voz que no requieran el utilizar servicios de procesamiento externo, con esto se mejoraría el tiempo de respuesta.

Adicionalmente, el reconocimiento de voz utilizado en el proyecto tiene el inconveniente que al dar una orden busca todas las posibles respuestas, por ejemplo, para el comando up da respuestas como app, ap, apa, lo cual causa que no se reconozca el comando fácilmente. Una propuesta de trabajo es que el entrenamiento del reconocimiento de voz este encaminado a buscar solamente las posibles ordenes que puede recibir el robot, de esta manera se ejecutara el comando más similar al que se ha ordenado mejorando también el tiempo de respuesta y la experiencia del usuario.

7. PRESUPUESTO

Este capítulo desarrolla el estudio de los costes económicos asociados a la elaboración de este proyecto. Se incluye los precios del material empleado, y el tiempo dedicado al desarrollo de los programas necesarios para la resolución de la simulación y la documentación.

Se realiza un estudio económico de los costes relacionados a la elaboración de este proyecto ya que forman parte del total de inversión necesaria para la puesta en marcha del mismo, si se considera que ha sido realizado por un ingeniero facturando las horas de trabajo. A continuación, se detalla el desglose de los costes asociados al proyecto en función de su naturaleza.

- Coste de mano de obra
- Coste de equipos
- Coste de materiales
- Coste total

7.1. Costes directos

Los costes directos que se van a considerar en la elaboración del proyecto son referidos al personal, materiales y maquinaria. A continuación, se detalla cada uno de los costes económicos directos que conllevan la realización del proyecto y estos son, costes de personal, de materiales amortizables y no amortizables y costes directos totales.

7.1.1. Coste de personal

Estos costes se calculan en función del número de horas empleadas por el ingeniero técnico industrial y del coste por hora de trabajo. A continuación, en la Tabla 14, se detalla un desglose del tiempo usado para el desarrollo de cada actividad del proyecto

Concepto	**Horas**
Investigación	80 horas
Programación de los aplicativos	100 horas
Integración del proyecto	60 horas
Pruebas de funcionamiento	40 horas
Elaboración de la documentación	95 horas
Total horas dedicadas	**375 horas**

Tabla 14. Tiempo de dedicación a las actividades

Con el cálculo del coste de hora de trabajo y el del tiempo de dedicación total para la realización del proyecto se obtiene el coste total de personal mostrado en la Tabla 15.

Costes de personal		
Total coste persona/año	30	€ / hora
Total horas por año trabajadas	375	horas
Total coste por hora	**11250**	**€**

Tabla 15. Costes de personal

7.1.2. Costes de materiales no amortizables

Estos costes son referidos al material de oficina como papel, programas, copias, etc. Los materiales no amortizables se detallan en la Tabla 16.

Concepto	**Importe**
Placa Lattepanda	150 €
Micrófono	20 €
Teclado y ratón	30 €
Total costes materiales	**200 €**

Tabla 16. Costes de materiales no amortizables

7.1.3. Costes directos totales

La Tabla 17 muestra el cálculo total de los costes directos que son el resultado del total de materiales y el total de costes de personal.

Concepto	**Importe**
Costes de personal	11250 €
Costes de materiales	200 €
Total costes directos	**11450** €

Tabla 17. Costes directos no amortizables

7.1.4. Costes de materiales amortizables

Los costes amortizables son los correspondientes a los equipos informáticos y software utilizados en la elaboración del proyecto. En este caso se estima una amortización a 4 años tanto para el software como para el hardware, por lo que el factor de amortización es el 25% sobre el importe de la inversión realizada, estos costes se muestran en la Tabla 18.

Concepto	Importe	Amortización (25%)
Sistema Operativo Windows10	200 €	50 €
Microsoft Office 2016	200 €	50 €
Total materiales oficina	**400 €**	**100 €**

Tabla 18. Costes de materiales amortizables

7.1.5. Costes directos totales

En la Tabla 19 se muestra el total de los costes directos, esto es el resultado de la suma de los gastos antes calculados, costes de personal, amortización de programas de equipos, y material empleado.

Concepto	Importe
Costes de personal	11250 €
Costes de materiales	200 €
Costes de amortización	100 €
Total costes directos	**11550 €**

Tabla 19. Costes directos

7.2. Costes indirectos

Este punto incluye los gastos ocasionados por la elaboración del proyecto pero que no están directamente relacionas con él. Aunque no se ven directamente en el proyecto deben ser tenidos en consideración, por ejemplo, el consumo eléctrico o los servicios administrativos. Estos costes se encuentran en la Tabla 20.

Concepto	Importe
Internet	140 €
Total costes indirectos	**140 €**

Tabla 20. Costes indirectos

7.3. Costes totales

Finalmente, el coste total del proyecto es la suma de los costes directos e indirectos calculados. Estos valores se detallan en la Tabla 21.

Concepto	Importe
Total costes directos	11550 €
Total costes indirectos	140 €

Total coste proyecto	**11690 €**

Tabla 21. Coste total del proyecto

El proyecto de desarrollo de un aplicativo de control de voz para el control de movimiento de un robot ABB IRB140 tiene un coste final de 11690 €

8. CONCLUSIONES

8.1. Conclusiones

El proyecto logro integrar de manera satisfactoria los tres campos de desarrollo planteados inicialmente que son la inteligencia artificial con el algoritmo de reconocimiento de voz, la robótica con la interfaz de pruebas, y las comunicaciones industriales con la comunicación OPC, en un aplicativo funcional capaz de controlar el movimiento de un robot por medio de comandos de voz.

Partiendo de un conocimiento muy básico sobre el reconocimiento de voz y sin ninguna experiencia en su uso, se ha logrado comprender de manera integral su funcionamiento, y como se pueden utilizar, en la creación de nuevas aplicaciones.

Los experimentos realizados con la interfaz de pruebas funcionaron muy bien, incluso con cierto grado de ruido ambiental se obtuvo una respuesta satisfactoria. lo que indica claramente que la tecnología es adecuada para usar en aplicaciones industriales donde la cooperación entre personas y máquinas es necesaria o donde la intervención del operador es mínima.

Se ha logrado superar problemas del reconocimiento de voz al entrenar el sistema con las posibles opciones de reconocimiento, y se ha dotado el sistema de la capacidad de lenguaje natural con lo cual es capaz de reconocer comandos dichos de diferentes maneras. Esto hace que el uso sea mucho más sencillo para el usuario de este aplicativo.

Finalmente, gracias a este proyecto se ha profundizado en temas de programación de robots con funciones avanzadas, y se ha familiarizado con el entorno de simulación de RobotStudio que es muy útil para el uso de robots industriales. Además, se ha aprendido sobre el lenguaje de programación Python que es uno de los de mayor crecimiento en la actualidad, por lo que el saber usarlo es una gran herramienta en el desarrollo de futuros trabajos.

9. BIBLIOGRAFÍA

ABB Robotics. (2004). *Especificaciones del producto IRB 140.* Vasteras: ABB.

ABB Robotics. (2007). *Manual de Referencia Técnica, Descripcion general de RAPID.* Vasteras: ABB.

ABB Robotics. (2010). *Technical Reference Manual, Instructions, Functions and Data types.* Vasteras: ABB.

ABB Robotics. (2011). *Manual del Operador, RobotStudio.* Vasteras: ABB.

ABB Robotics. (2013). *Especificaciones del producto, Controller IRC5 with FlexPendant.* Vasteras: ABB.

ABB Robotics. (2015). *Application Manual, IRC5 OPC Server help.* Vasteras: ABB.

Beigi, H. (2011). *Fundamentals of Speaker Recognition.* New York: Springer.

Clark, A., Fox, C., & Lappin, S. (2010). *The Handbook of Computational Linguistics and Natural Language Processing.* Oxford: Wiley Blackwell.

Indurkhya, N., & Damerau, F. (2010). *Handbook of Natural Language Processing.* Boca Raton: CRC Press.

Jurafsky, D., & Martin, J. (2018). *Speech and Language Processing.* Palo Alto: Stanford University.

Kepware. (06 de 2019). *Guia de comunicacion remota por OPC DA.* Obtenido de Kepserverxopx: www.kepserverexopc.com

Keshet, J., & Bengio, S. (2009). *Automatic Speech and Recognition.* Mountain View: Wiley.

LattePanda. (01 de junio de 2019). *LattePanda 4G/64GB – LattePanda*. Obtenido de Lattepanda.com: https://www.lattepanda.com/products/3.html

Lutz, M. (2010). *Programming Python.* Cambridge: O'Reilly.

Lynch, K., & Park, F. (2017). *Modern Robotics Mechanics, Planning, and Control.* Cambridge: Cambridge University.

Murphy, R. (2000). *Introduction to AI Robotics.* Cambridge: Bradford Book.

Perez, J. A., Deligianni, F., Ravi, D., & Yang, G. Z. (2016). *Artificial Intelligence and Robotics.* UK-RAS Network.

Pires, J. (2010). *Robot by Voice: Experiments on commanding an industrial robot using the human voice.* Coimbra: University of Coimbra.

Wilamowski, B., & Irwin, J. (2011). *The Industrial Electronics Handbook.* Boca Raton: CRC Press.

Wisskirchen, G., Thibault, B., & Bormann, U. (2017). *Artificial Intelligence and Robotics and their Impact on the Workplace.* International Bar Association.

Yu, D., & Deng, L. (2015). *Automatic Speech Recognition a Deep Learning Approach.* London: Springer.

10. ANEXOS

10.1. Anexo 1: Código de programación en Python

```
#*************************************************************************
#     UNIVERSIDAD POLITECNICA DE CATALUNYA
#     MASTER EN INGENIERIA DE SISTEMAS AUTOMATICOS Y ELECTRONICA INDUSTRIAL
#     TRABAJO DE FIN DE MASTER
#     DISEÑO DE UN APLICATIVO PARA EL CONTROL DE MOVIMIENTO
#     DE UN ROBOT ABB MEDIANTE RECONOCIMIENTO DE VOZ
#*************************************************************************

#  Librerias y modulos
import random
import os
import time
import speech_recognition as sr
import OpenOPC
import datetime, threading, time

#  Entrenamiento de comandos de voz
ok_robot = ["okay Robert", "okay robot", "okay rabbit", "okay or what", "hey
Robert", "okay.", "okay remote"]
go_home = ["home", "come", "crumb", "Chrome"]
go_up = ["up", "app", "Go app", "morph app", "move app", "go up", "move up"]
go_down = ["down", "hoedown", "go down", "Bow Down", "most down", "moth down",
"more down", "move down", "wolfdown", "morph down", "mulch down"]
go_right = ["right", "bright", "move right", "go right", "gold right"]
go_left = ["left", "go left", "move left"]
go_front = ["front", "go to front", "how front", "go front", "move to front",
"move front"]
go_back = ["back", "go back", "move back", "move to back", ""]
go_point_a = ["point a", "go to point a", "2. day", "move to point a", "go to
point the a", "move point a", "goldpoint a", "go point a"]
go_point_b = ["point B", "going to be", "Tubi", "how to point B", "go to point
B", "move to point B", "move point B", "goldpoint be", "gold pointy"]
go_point_c = ["Point C", "go to point C", "how to point C", "move to point C"]
go_recognize = ["recognize"]


#  Establecimiento de la comunicacion con el servidor OPC de ABB
opc = OpenOPC.client()
opc.connect('ABB.IRC5.OPC.Server.DA')

#  Definicion clase de recopilacion de voz con el microfono
def recognize_speech_from_mic(recognizer, microphone):
    """Escribe lo escuchado y grabado desde `microphone`.

    Retorna un diccionario con tres keys:
```

```
    "success": un valor boolean que indica si la respuesta de la API fue o no
exitosa
    "error":   `None` si no hay error, o un string que contiene
               un mensaje de error si no se pudo acceder a la API o
               si la voz no fue reconocida
    "transcription": `None` si la voz no se puede transcribir,
               o un string que contiene el texto transcrito
    """
    # Verificar que los argumentos recognizer y microphone son del tipo
apropiado
    if not isinstance(recognizer, sr.Recognizer):
        raise TypeError("`recognizer` debe ser una instancia `Recognizer`")

    if not isinstance(microphone, sr.Microphone):
        raise TypeError("`microphone` debe ser una instancia `Microphone`")

    # Ajustar recognizer sensitivity para el ruido ambiental del audio grabado
    # Desde el microfono "microphone"
    with microphone as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)

    # Configurar el objeto response
    response = {
        "success": True,
        "error": None,
        "transcription": None
    }

    # try recognizing the speech in the recording
    # if a RequestError or UnknownValueError exception is caught,
    #     update the response object accordingly
    try:
        response["transcription"] = recognizer.recognize_google(audio)
    except sr.RequestError:
        # API was unreachable or unresponsive
        response["success"] = False
        response["error"] = "API unavailable"
    except sr.UnknownValueError:
        # speech was unintelligible
        response["error"] = "Unable to recognize speech"

    return response


# Aplicacion de reconocimiento

# Crear instancias recognizer y microphone
recognizer = sr.Recognizer()
```

```
microphone = sr.Microphone()

#Funcion texto inicio de aplicacion
def inicio_app():
    print("*********************************************")
    print("*  Control para robot ABB IRC140 iniciado  *")
    print("*********************************************")
    return

#Funcion texto final de aplicacion
def final_app():
    print("*********************************************")
    print("*  Control para robot ABB IRC140 finalizado  *")
    print("*********************************************")
    return

#Funcion texto Iniciar aplicacion
def iniciar_app():
    print("*****************************************************")
    print("*¿INICIAR CONTROL POR VOZ PARA ABB IRC140?: [y/n]*")
    print("*****************************************************")
    return

#Funcion de reconocimiento de voz
def reconocer_voz():
    guess = recognize_speech_from_mic(recognizer, microphone)
    print("---------------------")
    print("Usted ha dicho:")
    respuesta=str(guess["transcription"])
    print(respuesta)
    print("--------------------")
    return respuesta


#Funcion para establecer los comandos
def comandos_robot():
    commands = ["Home", "Up", "Down", "Right", "Left", "Point A", "Point B",
"Point C","Recognize"]

    instructions = (
    "Diga uno de los siguientes comandos:\n"
    "{commands}\n"
    ).format(commands=', '.join(commands))

    key1 = True

    while key1:
        print(instructions)
        comando = reconocer_voz()
```

```
#  Comando Home
        elif comando in go_home:
            print("El robot se movera a la posicion HOME")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 2) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
#  Comando Arriba
        elif comando in go_up:
            print("El robot se movera a: ARRIBA")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 3) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
#  Comando Abajo
        elif comando in go_down:
            print("El robot se movera a: ABAJO")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 4) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
#  Comando Derecha
        elif comando in go_right:
            print("El robot se movera a: DERECHA")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 5) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
#  Comando Izquierda
        elif comando in go_left:
            print("El robot se movera a: IZQUIERDA")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 6) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
#  Comando Adelante
        elif comando in go_front:
            print("El robot se movera a: ADELANTE")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 7) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
#  Comando Atras
        elif comando in go_back:
            print("El robot se movera a: ATRAS")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 8) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
#  Comando Punto A
        elif comando in go_point_a:
            print("El robot se movera a: PUNTO A")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 9) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
```

```
#  Comando Punto B
        elif comando in go_point_b:
            print("El robot se movera a: PUNTO B")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 10) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
#  Comando Punto C
        elif comando in go_point_c:
            print("El robot se movera a: PUNTO C")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 11) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
#  Comando Reconocer
        elif comando in go_recognize:
            print("El robot reconocera el espacio de trabajo")
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 12) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
#  Comando Salir
        elif comando == "exit":
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 2) )
            time.sleep(3)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
            opc.write( ('TFM.RAPID.T_ROB1.Module1.start', 0) )

            key1 = False
    return

#  Texto que pregunta si se desea iniciar la aplicacion
iniciar_app()
#  Lectura de ingreso por teclado
acceso = input()
#  Si se ingresa la letra y "yes" por el teclado arranca el aplicativo
if acceso == 'y':
    key = True
    inicio_app()

    while key:
        respuesta = reconocer_voz()
        if respuesta == "exit":
            key = False
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 2) )
            time.sleep(1)
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
            opc.write( ('TFM.RAPID.T_ROB1.Module1.start', 0) )
            opc.close()
        elif respuesta in ok_robot:
            print("Se ha reconocido su voz...")
```

```
            opc.write( ('TFM.RAPID.T_ROB1.Module1.start', 1) )
            opc.write( ('TFM.RAPID.T_ROB1.Module1.mode', 0) )
            comandos_robot()
            key = False
            final_app()
        else:
            print('Diga la palabre "EXIT" para salir, o la PALABRA CLAVE para
continuar')
else:
    final_app()
```

10.2. Anexo 2: Código de programación en Rapid

```
MODULE Module1

    !Variables de comunicacion
    PERS num start;
    PERS num mode;

    !Posicion Home
    CONST robtarget
home:=[[515,0,712.000000147],[0.707106782,0.000000001,0.70710678,0.000000001],[0,0
,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !Puntos fijos para movimiento
    CONST robtarget Target_10:=[[320,-200,250],[0,0,1,0],[-
1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_20:=[[320,395,250],[0,0,1,0],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_30:=[[700,395,250],[0,0,1,0],[0,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_40:=[[700,-200,250],[0,0,1,0],[-1,-
1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    CONST robtarget Target_50:=[[500,0,500],[0,0,1,0],[-
1,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

    !Puntos variables para movimiento
    VAR robtarget moveup;
    VAR robtarget movedown;
    VAR robtarget moverigth;
    VAR robtarget moveleft;
    VAR robtarget movefront;
    VAR robtarget moveback;

    !***********************************************************
    ! Module:  Module1
    ! Description:
    ! Trabajo de Fin de Master
    ! Master en Sistemas Automaticos y Electronica Industrial
    ! Author: Diego Mosquera
    ! Version: 5.0
    !***********************************************************

    PROC main()
        go_recognize;

        IF start=1 THEN
            TPWrite "Reconocimiento de voz iniciado";
            !Comando stop para parar el robot
            ciclorobot:
            !Posicion HOME
            IF mode=2 THEN
                go_home;
                GOTO ciclorobot;
                !Posicion UP
            ELSEIF mode=3 THEN
                go_up;
                GOTO ciclorobot;
                !Posicion DOWN
```

```
        ELSEIF mode=4 THEN
            go_down;
            GOTO ciclorobot;
            !Posicion RIGHT
        ELSEIF mode=5 THEN
            go_rigth;
            GOTO ciclorobot;
            !Posicion LEFT
        ELSEIF mode=6 THEN
            go_left;
            GOTO ciclorobot;
            !Posicion FRONT
        ELSEIF mode=7 THEN
            go_front;
            GOTO ciclorobot;
            !Posicion BACK
        ELSEIF mode=8 THEN
            go_back;
            GOTO ciclorobot;
            !Posicion A
        ELSEIF mode=9 THEN
            go_point_a;
            GOTO ciclorobot;
            !Posicion B
        ELSEIF mode=10 THEN
            go_point_b;
            GOTO ciclorobot;
            !Posicion C
        ELSEIF mode=11 THEN
            go_point_c;
            GOTO ciclorobot;
        ELSEIF mode=12 THEN
            go_recognize;
            GOTO ciclorobot;
        ENDIF
    ENDIF
ENDPROC

PROC go_home()
    MoveJ home,v500,z10,tool0\WObj:=wobj0;
ENDPROC

PROC go_up()
    moveup:=CRobT();
    moveup.trans.z:=moveup.trans.z+20;
    If IsReachable(moveup,tool0,wobj0) THEN
        MoveJ moveup,v100,z10,tool0\WObj:=wobj0;
        WaitTime 1;
    ENDIF
ENDPROC

PROC go_down()
    movedown:=CRobT();
    movedown.trans.z:=movedown.trans.z-20;
    If IsReachable(movedown,tool0,wobj0) THEN
        MoveJ movedown,v100,z10,tool0\WObj:=wobj0;
        WaitTime 1;
    ENDIF
ENDPROC
```

```
PROC go_rigth()
    moverigth:=CRobT();
    moverigth.trans.y:=moverigth.trans.y+20;
    If IsReachable(moverigth,tool0,wobj0) THEN
    MoveJ moverigth,v100,z10,tool0\WObj:=wobj0;
    WaitTime 1;
    ENDIF
ENDPROC

PROC go_left()
    moveleft:=CRobT();
    moveleft.trans.y:=moveleft.trans.y-20;
    If IsReachable(moveleft,tool0,wobj0) THEN
    MoveJ moveleft,v100,z10,tool0\WObj:=wobj0;
    WaitTime 1;
    ENDIF
ENDPROC

PROC go_front()
    movefront:=CRobT();
    movefront.trans.x:=movefront.trans.x+20;
    If IsReachable(movefront,tool0,wobj0) THEN
    MoveJ movefront,v100,z10,tool0\WObj:=wobj0;
    WaitTime 1;
    ENDIF
ENDPROC

PROC go_back()
    moveback:=CRobT();
    moveback.trans.x:=moveback.trans.x-20;
    If IsReachable(moveback,tool0,wobj0) THEN
    MoveJ moveback,v100,z10,tool0\WObj:=wobj0;
    WaitTime 1;
    ENDIF
ENDPROC

PROC go_point_a()
    MoveJ Target_10,v1000,z10,tool0\WObj:=wobj0;
ENDPROC

PROC go_point_b()
    MoveJ Target_20,v1000,z10,tool0\WObj:=wobj0;
ENDPROC

PROC go_point_c()
    MoveJ Target_50,v1000,z10,tool0\WObj:=wobj0;
ENDPROC

PROC go_recognize()
    MoveJ Target_10,v1000,z10,tool0\WObj:=wobj0;
    MoveJ Target_20,v1000,z10,tool0\WObj:=wobj0;
    MoveJ Target_30,v1000,z10,tool0\WObj:=wobj0;
    MoveJ Target_40,v1000,z10,tool0\WObj:=wobj0;
    MoveJ Target_10,v1000,z10,tool0\WObj:=wobj0;
    MoveJ home,v1000,z10,tool0\WObj:=wobj0;
ENDPROC
```

```
    FUNC bool IsReachable(robtarget pReach,PERS tooldata ToolReach,PERS wobjdata
WobjReach)

        ! Check if specified robtarget can be reach with given tool and wobj.
        !
        ! Output:
        !  Return TRUE if given robtarget is reachable with given tool and wobj
        !  otherwise return FALSE
        !
        ! Parameters:
        !  pReach     - robtarget to be checked, if robot can reach this robtarget
        !  ToolReach  - tooldata to be used for possible movement
        !  WobjReach  - wobjdata to be used for possible movement

        VAR bool bReachable;
        VAR jointtarget jntReach;

        bReachable:=TRUE;

        jntReach:=CalcJointT(pReach,ToolReach\Wobj:=WobjReach);

        RETURN bReachable;

    ERROR
        IF ERRNO=ERR_ROBLIMIT THEN
            bReachable:=FALSE;
            TRYNEXT;
        ENDIF
    ENDFUNC

ENDMODULE
```