

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ARQUITECTURA
SANTIAGO-CHILE



**PROTOTIPO DE SISTEMA DE CONTROL POR VOZ PARA LA
COLABORACIÓN HUMANO-ROBOT EN LA
CONSTRUCCIÓN ROBOTIZADA EN MADERA**

Tamara Josefina Pérez Brunel

Memoria de Titulación

para optar a título de Arquitecto

Modalidad I+D

Profesor referente: Francisco Quitral

NOVIEMBRE 2025



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Prototipo de sistema de control por voz para la Colaboración Humano-Robot en la Construcción Robotizada en Madera

Nombre del candidato(a): Tamara Josefina Pérez Brunel

Carrera / Grado: Arquitectura

Campus: Santiago San Joaquín **Departamento:** Arquitectura

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Francisco Javier Quitral Zapata, en mi calidad de profesor guía/director del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 10/01/2026

Firma:

Digitally signed by Francisco Quitral
Date: 2026.01.10 18:10:06-03'00'
Foxit PDF Editor Version: 13.1.7

Estudiante o Candidato(a):

Fecha: 10/01/2026

Firma:

**PROTOTIPO DE SISTEMA DE CONTROL POR VOZ PARA LA
COLABORACIÓN HUMANO-ROBOT EN LA CONSTRUCCIÓN
ROBOTIZADA EN MADERA**

RESUMEN

El presente proyecto I+D tiene como objetivo principal desarrollar un prototipo de sistema de control por voz para la configuración y ajuste del sistema de coordenadas en un robot industrial KUKA, aplicado procesos de construcción robotizada en madera. Este sistema integra reconocimiento de voz (*Speech-To-Text, STT*), programación visual en Grasshopper y control de trayectorias mediante KUKA|prc y RoboDK, permitiendo una calibración más rápida, intuitiva y segura, prescindiendo del uso intensivo del controlador principal SmartPAD.

El sistema considera un flujo de trabajo propuesto que inicia con la captura de audio mediante un micrófono inalámbrico con cancelación de ruido, cuya transcripción es procesada en Python para interpretar comandos direccionales y rotacionales. Estos comandos se simulan primero en KUKA|prc, generando una validación visual que, junto a una retroalimentación auditiva del sistema (*Text-To-Speech, TTS*), permite confirmar la instrucción proporcionada antes de su ejecución física. Una vez validada, la trayectoria se envía a RoboDK para hacer la conexión online al robot real y ejecutar la acción.

La metodología de desarrollo se estructura en cuatro fases: (1) Integración de reconocimiento de voz a Grasshopper, evaluando cinco motores STT (Microsoft Azure Speech SDK; Google Speech Recognition, Vosk, Vosk small y Open AI Whisper); (2) Simulación offline de comandos y detección de errores, donde se simulan movimientos dados por instrucciones por voz para identificar la tasa de aciertos de ejecución y errores. (3) Simulación online, en la que se hace la conexión al robot KUKA KR 210 R3100 2-C mediante RoboDK para probar trayectorias en condiciones controladas y (4) Validación experimental en entorno real de trabajos de mecanizado en madera, midiendo precisión y efectividad del flujo de trabajo en condiciones reales.

ABSTRACT

This R&D project aims to develop a voice-controlled system prototype for configuring and adjusting the workpiece coordinate system in a KUKA industrial robot, intended for robotic timber construction processes. The system integrates Speech-to-Text (STT) recognition, visual programming in Grasshopper, and trajectory control via KUKA|prc and RoboDK, enabling faster, more intuitive, and safer calibration while reducing reliance on the SmartPAD.

The project follows a workflow that begins with audio capture through a wireless microphone with noise cancellation, whose transcription is processed in Python to interpret both translational and rotational commands. These commands are first simulated in KUKA|prc, providing visual validation which, combined with Text-to-Speech (TTS) auditory feedback, allows the operator to confirm the instruction before physical execution. Once validated, the trajectory is sent to RoboDK to establish an online connection with the real robot and execute the commanded movement. The development methodology is structured into four stages:

(1) Integration of voice recognition into Grasshopper, evaluating five STT engines (Microsoft Azure Speech SDK, Google Speech Recognition, Vosk, Vosk Small, and OpenAI Whisper); (2) Offline simulation of commands and error detection, where voice-given instructions are simulated to assess execution accuracy and identify errors. (3) Online simulation, in which a connection is established with the KUKA KR 210 R3100-2C robot via RoboDK to test trajectories under controlled conditions; and (4) Experimental validation in a real timber machining environment, measuring precision and workflow effectiveness under actual working conditions.

GLOSARIO

Archivo .src

Archivo de código fuente editable que contiene las instrucciones lógicas y de movimiento escritas en KRL para controlar un robot KUKA.

Azure STT:

Un servicio en la nube de Microsoft Azure que forma parte de sus capacidades de inteligencia artificial (IA), diseñado para transcribir el lenguaje hablado en texto escrito en tiempo real o a partir de archivos de audio pregrabados.

BASE (sistema de coordenadas de la pieza de trabajo):

Sistema de coordenadas que describe la posición y orientación de la pieza de trabajo en el espacio del robot. Se define a partir de puntos medidos sobre la pieza y se utiliza como referencia para ejecutar el mecanizado con precisión.

Construcción robotizada en madera:

Área de investigación dentro de la fabricación digital en madera que emplea robots industriales para asistir a carpinteros humanos en el corte de uniones complejas y el ensamble de estructuras entramadas a partir de modelos digitales, manteniendo precisión geométrica pese a la variabilidad del material.

Componente (Grasshopper):

Elemento gráfico básico e interactivo, representado como un bloque en el lienzo de Grasshopper, que encapsula una función o cálculo específico. Estos componentes tienen puertos de entrada y puertos de salida

Coordenadas XYZ:

Sistema de referencia tridimensional que define la posición absoluta de un punto en el espacio mediante los ejes X, Y y Z.

Ejecución online:

Fase en la que las trayectorias validadas se envían, mediante la API de RoboDK, al robot real para su ejecución en tiempo casi real. La comunicación en red permite que el sistema lea estados del robot (TCP, valores articulares) y ejecute los mismos movimientos probados en la simulación offline.

FRAME (Marco de referencia):

Conjunto de datos que describen la posición (coordenadas XYZ) y la orientación (ángulos ABC) de un sistema o herramienta respecto de otro sistema de referencia dentro del espacio de trabajo robótico.

Grasshopper:

Entorno de programación visual basado en nodos que opera como plugin dentro de Rhinoceros y permite construir definiciones paramétricas conectando componentes lógicos y geométricos.

Hardware:

Componente material o físico de un sistema informático.

KR C4 (KUKA Robot Controller 4):

Sistema de control utilizado para operar los robots industriales fabricados por KUKA.

KRL (KUKA Robot Language):

Lenguaje de programación estructurado y de alto nivel utilizado para definir y ejecutar tareas en robots industriales KUKA.

KUKA|prc:

Plugin de Grasshopper que permite generar, simular y postprocesar trayectorias para robots KUKA a partir de geometría y parámetros definidos en el entorno paramétrico. Proporciona componentes específicos para movimientos cartesianos y articulares, análisis de colisiones y exportación de programas.

Movimiento MoveJ:

Instrucción de movimiento articular en los robots KUKA que interpola el desplazamiento del robot en el espacio de sus articulaciones. Instrucción de RoboDK.

Movimiento MoveL:

Instrucción de movimiento lineal en robots KUKA que fuerza al TCP a desplazarse en línea recta entre dos poses en el espacio cartesiano. Instrucción de RoboDK.

Plugin:

Complemento de una aplicación para funciones específicas.

Pose:

Posición y orientación de un objeto, herramienta o del TCP del robot en el espacio tridimensional, descrita usualmente mediante coordenadas (X, Y, Z) y un conjunto de parámetros de orientación (por ejemplo, ángulos ABC).

Python:

Lenguaje de programación interpretado, de alto nivel y orientado a objetos, diseñado con una filosofía que enfatiza la legibilidad del código.

Rhinoceros 8:

Software comercial de modelado geométrico 3D desarrollado por Robert McNeel & Associates, utilizado como entorno principal para la definición de la geometría de trabajo y como plataforma base de Grasshopper.

RoboDK:

Software de simulación y programación offline de robots industriales que permite crear estaciones de trabajo virtuales, importar modelos 3D y enviar programas al robot físico a través de su API.

Simulación offline:

Etapas en la que los comandos de voz interpretados se convierten en movimientos del robot que se visualizan en un entorno virtual (Rhinoceros + Grasshopper + KUKA|prc), sin conexión al robot físico.

SmartPAD:

Controlador físico de robots KUKA, que permite su configuración y manipulación.

Software:

Componente lógico e intangible de los sistemas informáticos, que comprende todos los programas, aplicaciones, procedimientos, rutinas e instrucciones necesarias para que el hardware realice operaciones y tareas específicas.

Speech to Text (STT):

Tecnología de reconocimiento automático de voz que convierte audio en texto escrito. El motor STT segmenta la señal de audio, la transforma a representación espectral y utiliza modelos estadísticos o neuronales para identificar patrones, devolviendo la transcripción.

Technology Readiness Level (TRL):

Escala estandarizada que indica el grado de madurez de una tecnología, desde conceptos iniciales hasta aplicaciones en entorno real.

Text to Speech (TTS):

Tecnología de asistencia que convierte texto escrito de una interfaz digital en audio con sonido natural o voz humana sintética.

Tool Center Point (TCP):

Punto de referencia que representa el punto central de la herramienta montada en el robot. Utilizado como punto traslacional de los movimientos del robot.

TRL 4:

Nivel que indica que los componentes de una tecnología han sido integrados y validados en un entorno de laboratorio como un sistema funcional básico.

Trayectoria:

Secuencia ordenada de poses (posición y orientación del TCP) que describe el camino que seguirá el robot entre un punto inicial y uno final.

ÍNDICE

1. INTRODUCCIÓN	12
1.1 PROBLEMA U OPORTUNIDAD	15
1.2 SOLUCIÓN PROPUESTA	18
1.2.1 Campo de aplicación	24
1.3 ESTADO DEL ARTE	27
1.3.1 Colaboración humano robot (HRC)	28
1.3.2 Interfaces de control por voz	35
1.3.3 Aplicaciones en construcción o carpintería robotizada	41
1.4 OBJETIVOS	48
1.4.1 Objetivo general	48
1.4.2 Objetivos Específicos	48
1.5 METODOLOGÍA	50
1.5.1 Etapas de desarrollo	52
1.5.2 Componentes del sistema	54
2. DESARROLLO DEL TEMA	56
2.1 TECNOLOGÍAS DE RECONOCIMIENTO DE VOZ	56
2.1.1 Motores STT	57
2.1.2 Pruebas de motores	60
2.1.3 Resultados Parciales	66
2.2 PROTOTIPO DE CONTROL POR VOZ	68
2.2.1 Integración STT en Rhinoceros y Grasshopper	68
2.2.2 Simulación Offline	70
2.2.3 Integración de control online con RoboDK	77
2.2.4 Configuración RoboDK	78

2.2.5	Integración Grasshopper	84
2.2.6	Flujo de Trabajo	90
2.2.7	Resultados parciales	93
2.3	EXPERIMENTO DE VALIDACIÓN TRL4	94
2.3.1	Condiciones experimentales y configuración del sistema	95
2.3.2	Registros del experimento	100
2.3.3	Resultado	107
3.	CONCLUSIONES	112
3.1	DISCUSIÓN	113
3.2	PROYECCIÓN	116
4.	REFERENCIAS	119
5.	ANEXOS	122

1. INTRODUCCIÓN

La construcción robotizada en madera se presenta como un ámbito emergente de investigación aplicada que integra la precisión de los robots industriales con la tradición artesanal de la construcción en madera. Este cambio ha sido relevante en procesos de fabricación que involucran madera y uniones carpinteras, donde la precisión geométrica es fundamental debido a la variabilidad dimensional inherente del material.

Este enfoque posibilita que los robots actúen en la ejecución de uniones complejas y montajes de estructuras entramadas, ampliando las capacidades del diseño arquitectónico más allá de las restricciones de las herramientas manuales. Los robots industriales aportan precisión milimétrica en operaciones de corte fresado y perforación, a partir de datos obtenidos de modelos digitales desarrollados en entornos paramétricos como *Rhinoceros 3D* y *Grasshopper*.

En estos procesos, el sistema de coordenadas de la pieza (SC) actúa como referencia tridimensional, determinando con exactitud la posición y orientación del material dentro del espacio operativo del robot (Fig. 1). Dicho sistema es esencial para garantizar que las trayectorias se ajusten fielmente al modelo de simulación digital. Sin embargo, la variabilidad geométrica inherente a la madera exige recalibraciones periódicas para corregir desviaciones y conservar la precisión del mecanizado.

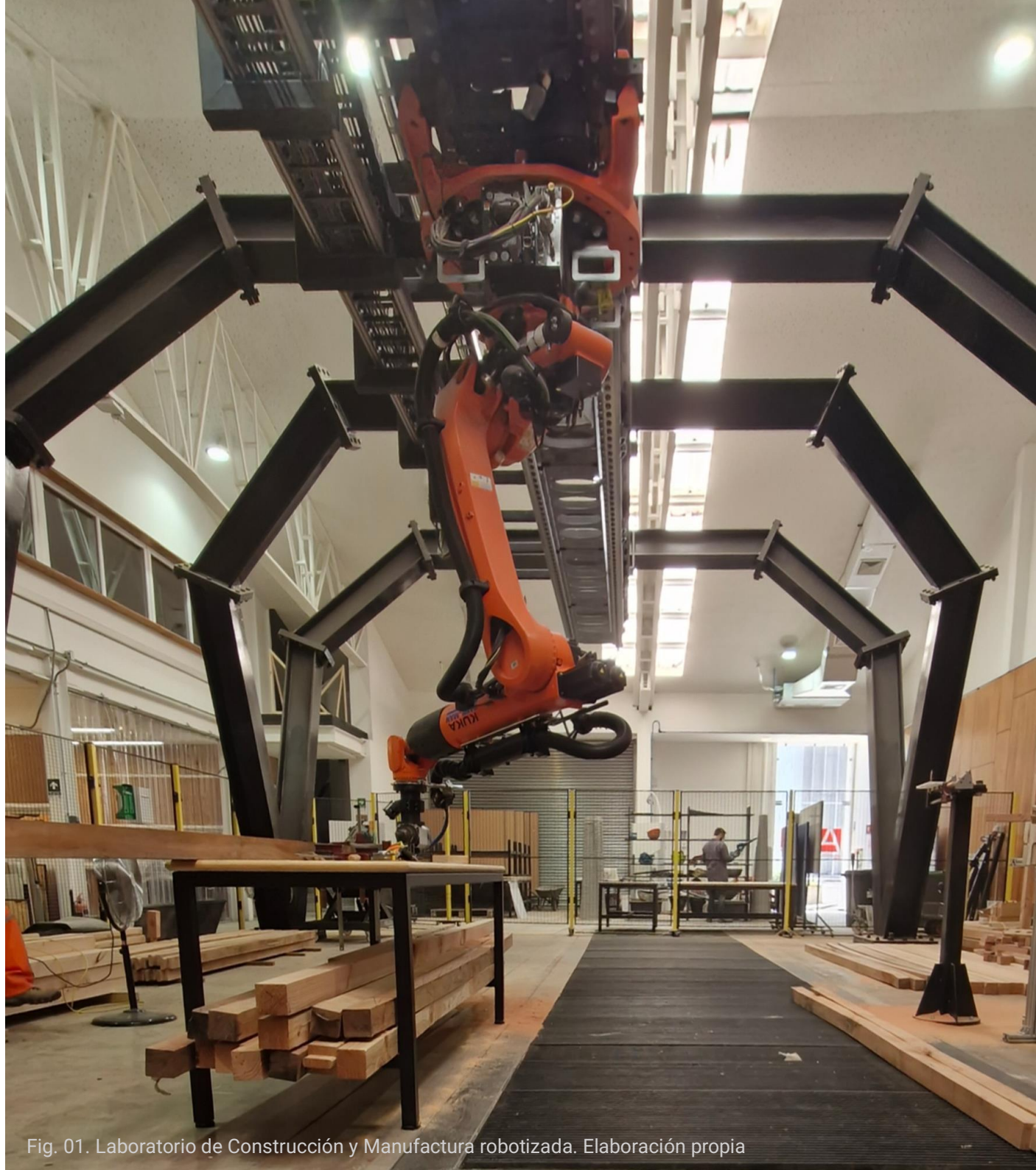


Fig. 01. Laboratorio de Construcción y Manufactura robotizada. Elaboración propia

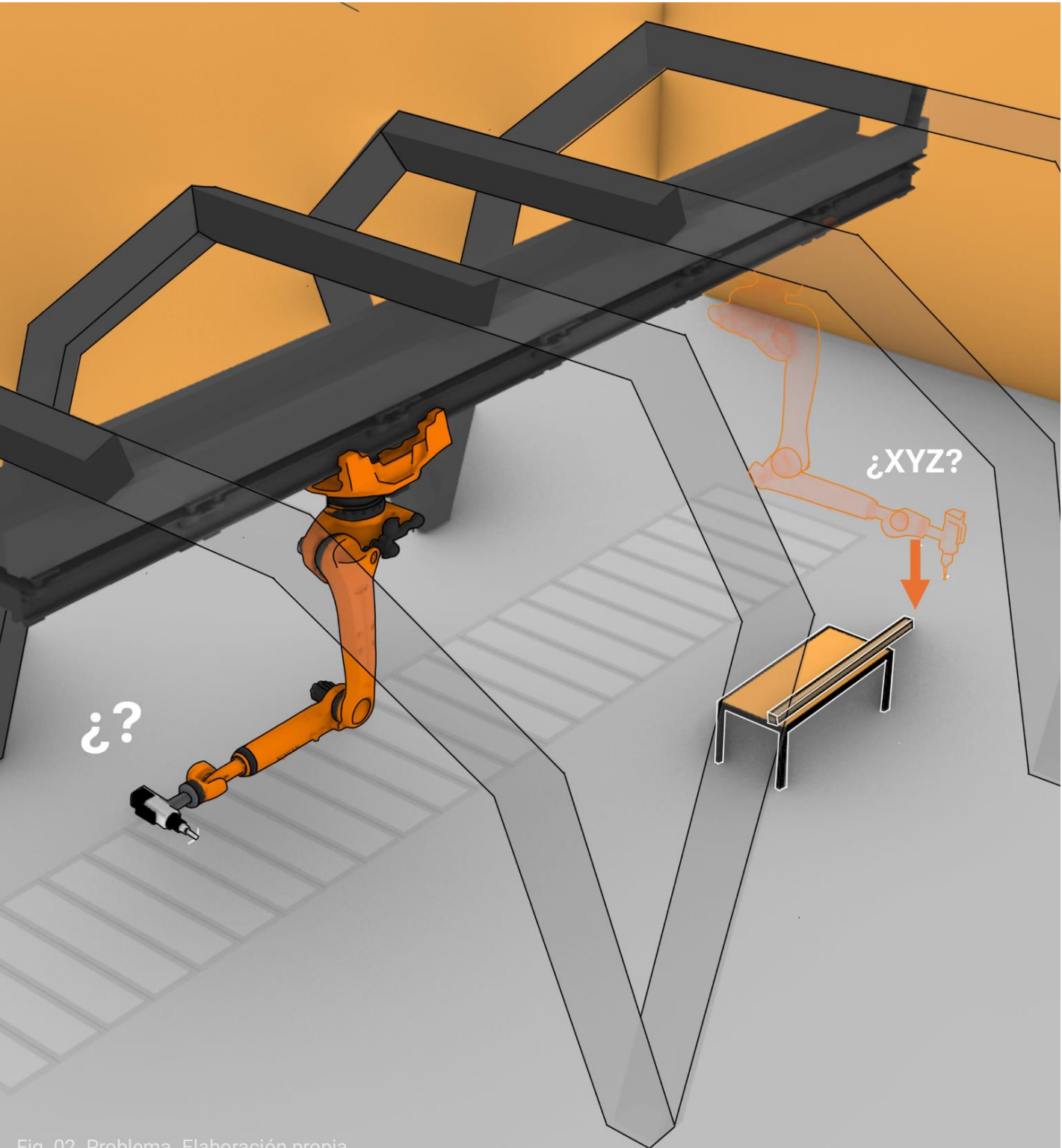


Fig. 02. Problema. Elaboración propia

1.1 PROBLEMA U OPORTUNIDAD

A pesar de los avances de la construcción robotizada en madera y su integración con herramientas digitales de fabricación, persisten limitaciones operativas que restringen su implementación extensiva y continua en el ámbito arquitectónico. En particular, la dependencia del controlador primario *SmartPAD* constituye una barrera de acceso significativa. Su interfaz técnica y las secuencias de calibración manual en tres etapas lo convierten en un instrumento diseñado principalmente para especialistas en robótica industrial. Esta condición restringe su utilización por parte de usuarios provenientes del ámbito del diseño o la arquitectura —como arquitectos, estudiantes o artesanos digitales— que carecen de formación avanzada en programación o control de robots.

Esta situación limita la integración directa del arquitecto en el proceso constructivo digital, reduciendo la posibilidad de exploración formal, experimentación proyectual y aplicación práctica de tecnologías robóticas en entornos académicos, talleres o laboratorios de arquitectura.

A lo anterior se suma la complejidad física de la madera como material de trabajo: su anisotropía, las variaciones dimensionales provocadas por la humedad y temperatura, y las deformaciones propias del mecanizado. Estas condiciones exigen recalibraciones constantes del sistema de coordenadas (SC) para garantizar la correspondencia precisa con el modelo digital de referencia.

En los flujos de trabajo actuales, el reajuste del SC se realiza manualmente mediante el *SmartPAD* utilizando el método de tres puntos, que obliga a posicionar el *Tool Center Point* (TCP) del robot sobre referencias físicas específicas y confirmar secuencialmente cada una. Este procedimiento puede demorar entre cinco y siete minutos por pieza, interrumpiendo la continuidad del flujo de trabajo, incrementando la carga cognitiva del operador y elevando la probabilidad de errores humanos que afectan la precisión de las uniones carpinteras. (Fig. 05)

En la construcción en madera de media y gran escala, se visualiza una oportunidad en las uniones híbridas madera-acero mediante placas ocultas o exteriores han adquirido amplia aceptación, ya que permiten transferir grandes cargas en secciones esbeltas y mejorar el comportamiento estructural frente a fuego y sismo. Cada una de estas placas requiere la ejecución de múltiples perforaciones con tolerancias inferiores a 1 mm, necesarias para garantizar el ajuste preciso de los pasadores y la capacidad portante de la unión. En la actualidad, este proceso se realiza mediante plantillas manuales y taladros portátiles (Fig. 03), donde el operario debe alinear cada plantilla, asegurar la ortogonalidad del taladro y evacuar viruta en cada perforación, repitiendo la operación múltiples veces por componente.

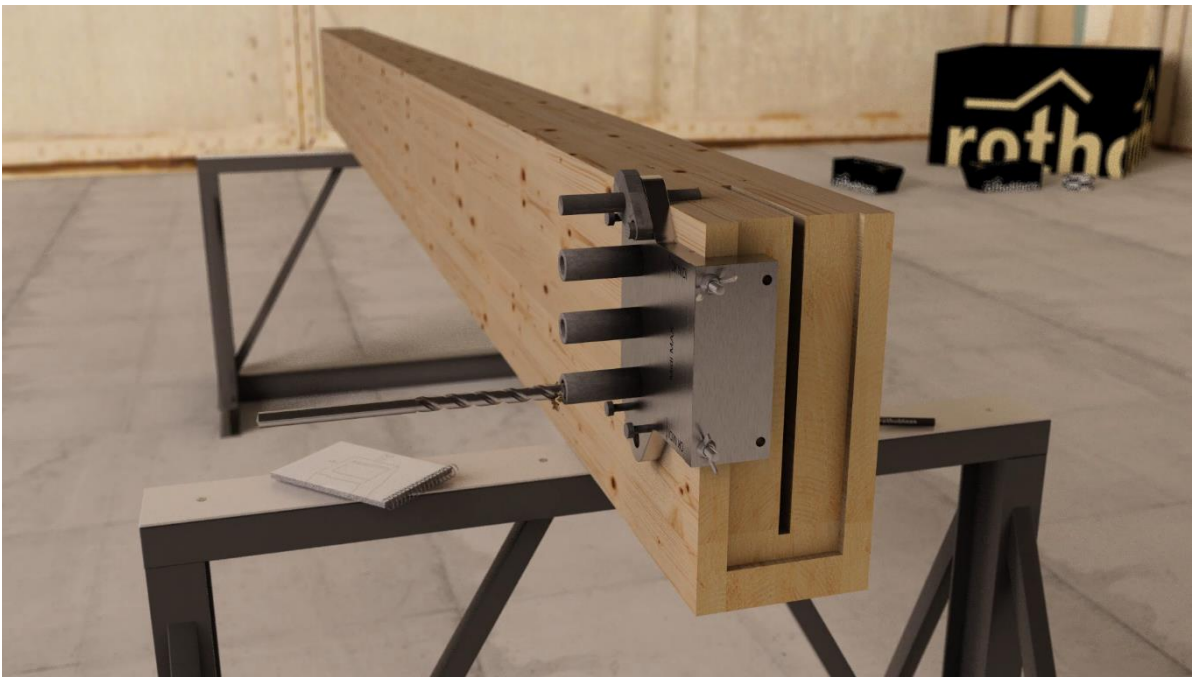


Fig. 03. Plantilla de perforación. Captura de video en YouTube [en línea]. Disponible en: <https://www.youtube.com/watch?v=Vwm02Hjz0-w>

Estos procesos deben ser realizados por personal capacitado múltiples veces en obra, lo que presenta desafíos en su ejecución (Fig. 06)

En estas operaciones, las desviaciones angulares mínimas generan desajustes en la inserción de los pasadores y pueden provocar fallas estructurales prematuras. Además, las herramientas de alto par requeridas exigen presión constante sobre el taladro, aumentando la fatiga del operario y el riesgo de error acumulativo. El resultado es una producción lenta, con retrabajos frecuentes y variabilidad dimensional que encarece las estructuras industrializadas de madera (Fig. 04).

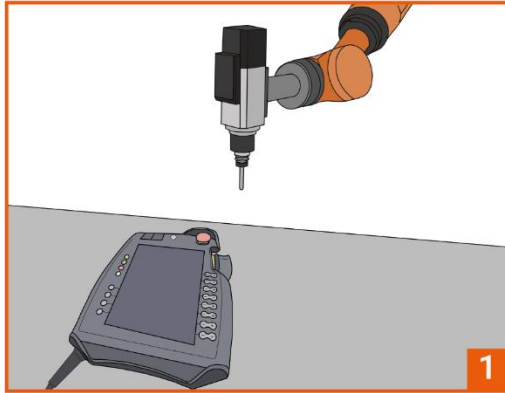


Fig. 04. Montaje incorrecto en obra. captura de video en YouTube [en línea]. Disponible en: <https://www.youtube.com/watch?v=0z8KYmgOnhs&t=160s>

En síntesis, la brecha productiva descrita evidencia la necesidad de estrategias que agilicen la recalibración del SC y la interacción operador-robot, dejando planteado el escenario para considerar alternativas de mejora expuestas en la Sección 1.2, ya que el flujo actual de recalibraciones es altamente sensible a variaciones del material y al criterio del operario, introduce interrupciones por cada pieza y carece de verificaciones estandarizadas que garanticen precisión y repetibilidad.

1.2 SOLUCIÓN PROPUESTA

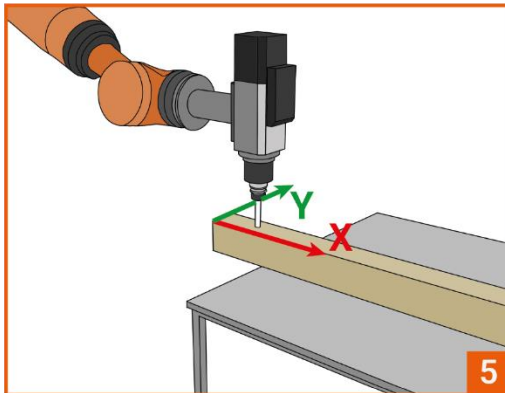
AJUSTE MANUAL Y REPETITIVO DEL SISTEMA DE



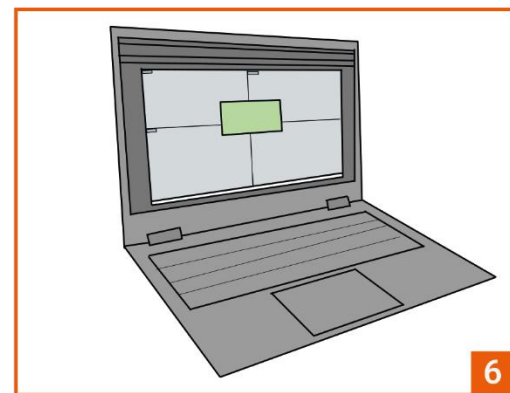
Encender robot y llevarlo a posición



Recorrer menús del SmartPAD para registrar puntos



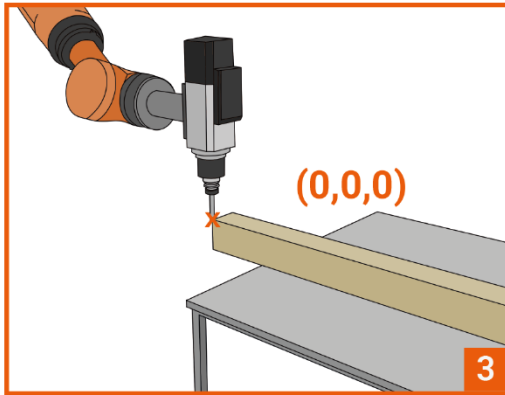
Indicar punto en plano XY de la pieza



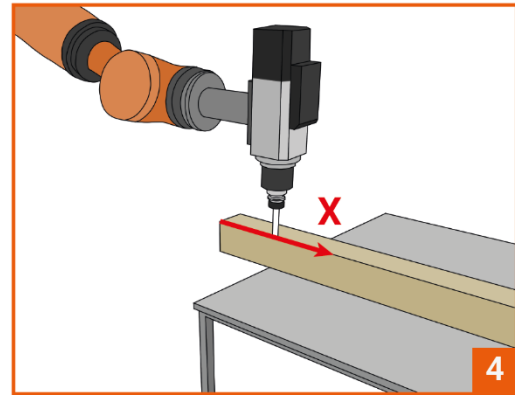
Ingresar SC a KUKA|prc

Fig. 05. Ajuste manual y repetitivo del sistema de coordenadas de la pieza. Elaboración Propia

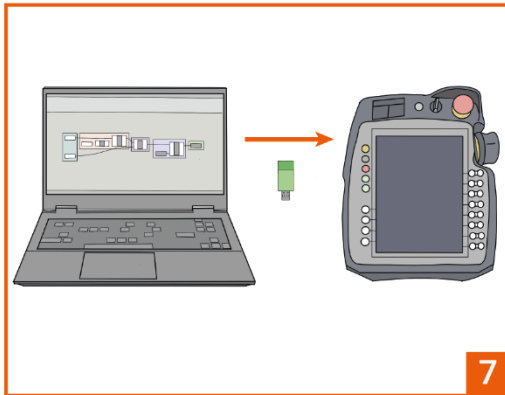
COORDENADAS (SC) DURANTE EL MECANIZADO



Indicar Origen $(0,0,0)$



Indicar punto en eje X de la pieza



Exportar archivo *.src a USB



Recorred menús del SmartPAD
para ejecutar programa

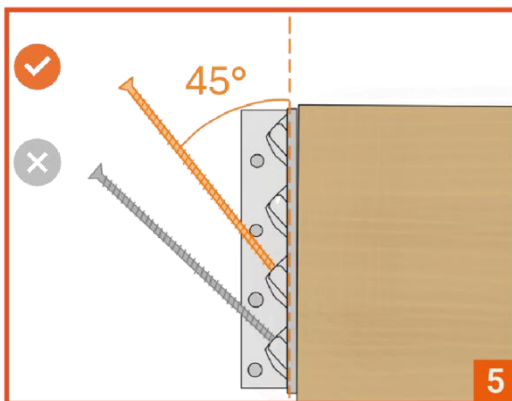
DESAFÍOS EN LA PERFORACIÓN Y FIJACIÓN



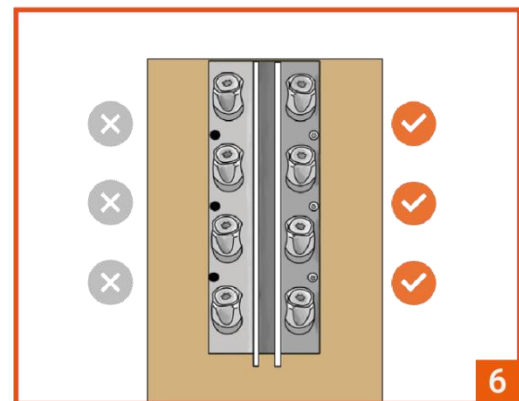
Precisión: Ubicación exacta en ángulo, profundidad y posición



Capacitación: Necesidad de operarios entrenados



Homogeneidad: Variaciones humanas acumulan tolerancias



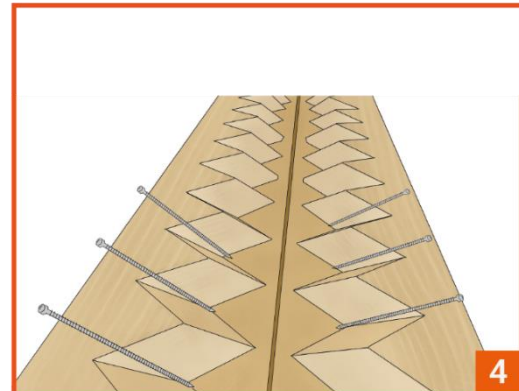
Normativa: Distancias mínimas entre tornillos y bordes

Fig. 06. Desafíos en la perforación y fijación de uniones en estructuras de madera. Elaboración Propia

DE UNIONES EN ESTRUCTURAS DE MADERA



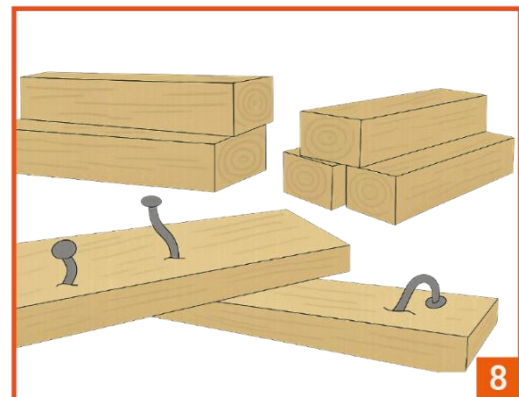
Fatiga: Repetición de tareas,
pérdida de precisión



Tiempo: Miles de perforaciones
ralentizan el montaje



Herramienta: Desgaste de
herramienta, desgarros de
fibra.



Reprocesos: Correcciones y
desperdicio de material

INTEGRACIÓN DE CONTROL POR VOZ PARA REAJUSTES Y CONTROL DEL SISTEMA ROBOTICO

A partir de las limitaciones observadas del uso de robots industriales en el campo de la construcción robotizada en madera, tales como la dependencia del *SmartPAD*, la complejidad de su interfaz, los tiempos de recalibración, la reducida accesibilidad para usuarios no técnicos y la alta carga cognitiva que requiere el operador para la ejecución de estas tareas, se plantea el desarrollo de un sistema de control por voz integrado a un flujo de trabajo para la calibración del SC, control y ejecución de tareas en un robot industrial KUKA, específicamente orientado a procesos de mecanizado y perforado en madera.

Con este sistema, se propone crear un entorno operativo más intuitivo y eficiente, que permite la integración directa de arquitectos, diseñadores y artesanos en el flujo de fabricación robotizada, reduciendo a grandes rasgos la brecha operativa entre el diseño y fabricación.



Fig. 07. Características de propuesta. Elaboración Propia

El sistema propuesto implementa una interfaz de control interactiva basada en el reconocimiento de voz, que traduce comandos verbales en instrucciones robóticas estructuradas a través del plugin *KUKA/prc* en *Grasshopper*, complementado con una retroalimentación auditiva generada por el sistema, que informa el estado de la operación o posibles advertencias antes de su ejecución física a través del programa *RoboDK* que hace conexión vía red para ejecutar movimientos a tiempo real.

De esta forma, se elimina el uso de menús, ingreso manual de coordenadas o programación directa con el *SmartPAD*, la interacción se produce exclusivamente mediante el lenguaje natural, sustituyendo la interfaz tradicional durante las etapas de recalibración y ajuste; ofreciendo una alternativa más rápida y accesible a los procedimientos con un robot industrial (Fig. 07).



Fig. 07. Desafíos en la perforación y fijación de uniones en estructuras de madera. Elaboración Propia

1.2.1 Campo de aplicación

La implementación de un sistema de control por voz integrado a un flujo de trabajo de programación visual y simulación robótica abre un campo de aplicación amplio en la arquitectura y la construcción robotizada en madera, particularmente en contextos donde la precisión geométrica, la adaptabilidad operativa y la continuidad del flujo de trabajo resultan determinantes.

En la construcción robotizada en madera, la necesidad de recalibrar frecuentemente el sistema de coordenadas de la pieza de trabajo, adaptar la posición del robot frente a variaciones dimensionales del material y ejecutar mecanizados repetitivos convierte a la voz en una interfaz estratégica para estandarizar procedimientos, reducir la complejidad de operación y disminuir la dependencia de interfaces técnicas como el *SmartPAD*. El sistema desarrollado articula estas necesidades mediante un conjunto de funciones específicas: medición de la BASE por voz, desplazamiento del robot mediante comandos hablados, obtención en tiempo real del TCP (*Tool Center Point*), programación de perforaciones y almacenamiento de poses predefinidas para distintas fases del proceso.

En primer lugar, el sistema presenta un alto potencial de aplicación en laboratorios universitarios y entornos académicos de fabricación arquitectónica, como el Laboratorio de Construcción y Manufactura Robotizada de la Universidad Técnica Federico Santa María, así como en otros centros de investigación vinculados a la robótica en arquitectura, el diseño paramétrico y la fabricación digital. En estos contextos, la interfaz por voz constituye una herramienta didáctica que permite a estudiantes y docentes interactuar con un robot industrial sin requerir experiencia avanzada en programación de controladores ni dominio exhaustivo del entorno del *SmartPAD*. La función de medición de BASE por voz simplifica la definición del sistema de coordenadas de la pieza de trabajo, transformando una tarea típicamente técnica y secuencial en un procedimiento guiado mediante instrucciones habladas, con apoyo de simulación visual en *Grasshopper* y *KUKA|prc*. Esto facilita que los usuarios se concentren en comprender la relación entre el modelo digital y el espacio físico, más que en la navegación de menús complejos. En estos laboratorios, la función de desplazamiento controlado por voz y la obtención continua del TCP en tiempo real

permiten explorar de forma segura conceptos como el espacio de trabajo del robot, las trayectorias cartesianas, los movimientos por ejes y la relación entre distintos sistemas de referencia. El monitoreo permanente del TCP, combinado con la validación previa de los movimientos en simulación, contribuye a reforzar la comprensión de la cinemática del robot y de las implicancias geométricas de cada comando. De este modo, el sistema actúa no solo como una interfaz operativa, sino como un recurso pedagógico que materializa en tiempo real los vínculos entre programación visual, robótica y geometría arquitectónica.



Fig. 08. Montaje de estructura en madera. Captura de video en YouTube [en línea]. Disponible en: (https://www.youtube.com/watch?v=RobJ9s_UIKw).

Por otra parte, la programación de perforaciones por voz amplía este campo de aplicación académico hacia la experimentación con uniones carpinteras y conectores metálicos específicos utilizados en la construcción en madera. Al permitir que el usuario defina matrices de perforaciones mediante comandos hablados (número de filas y columnas, distancias en X e Y, profundidad, etc.), el sistema hace posible simular y ejecutar patrones de taladrado asociados a uniones metálicas. En un entorno de obra, esto habilita la exploración de soluciones de unión, el estudio de tolerancias y la evaluación de distintos criterios de diseño estructural, reduciendo la barrera de

entrada para quienes no dominan el código de bajo nivel o las interfaces tradicionales del robot. La voz se convierte así en una puerta de acceso a la fabricación robotizada de elementos vinculados directamente a la práctica constructiva contemporánea en madera al ser la forma más natural e intuitiva de comunicación para el ser humano.

En un segundo nivel, el sistema encuentra un campo de aplicación relevante en talleres de carpintería especializados, laboratorios de prototipado y centros de fabricación a pequeña escala, donde se producen piezas únicas o series cortas de elementos de alta complejidad. En estos contextos, las variaciones inherentes de la madera, la diversidad de secciones y la necesidad de ajustes frecuentes de la pieza de trabajo hacen que la recalibración del sistema de coordenadas sea una tarea recurrente y, muchas veces, tediosa. La función de medición de BASE por voz permite redefinir rápidamente el sistema de referencia del robot frente a cada nueva pieza o lote, reduciendo tiempos de preparación y minimizando errores derivados de la manipulación manual del *SmartPAD*. La operación manos libres posibilita que el carpintero o el operador mantenga la atención sobre la inspección visual, la manipulación física del elemento y las condiciones de sujeción, sin necesidad de navegar repetidamente el controlador.

La capacidad de guardar y posicionar el robot en poses específicas aporta una ventaja adicional en estos talleres. Es posible, por ejemplo, registrar posiciones asociadas a un “punto de espera” seguro, a una zona de cambio de herramienta o a una posición de acercamiento estándar a la pieza de trabajo. Mediante comandos de voz, el operador puede llevar el robot a estas poses sin navegar por listas de puntos ni escribir código adicional, lo que disminuye la carga cognitiva y reduce la probabilidad de errores de selección. En combinación con la función de desplazamiento por voz, esta característica facilita el desarrollo de rutinas de trabajo semi-estandarizadas que pueden adaptarse rápidamente a distintos proyectos, manteniendo al mismo tiempo la flexibilidad necesaria para la construcción de prototipos y elementos singulares.

1.3 ESTADO DEL ARTE

El desarrollo de sistemas interactivos para la operación de robots industriales ha avanzado significativamente en la última década, impulsado por la necesidad de mejorar la colaboración entre humanos y máquinas en entornos productivos, educativos y constructivos. Estos avances han permitido explorar nuevas formas de interacción que superan a las interfaces tradicionales basadas en controladores físicos, integrando controles de reconocimiento de gestos, comandos de voz, programación visual, asistencia auditiva y visión artificial. En particular, en ámbitos como la fabricación digital, la construcción y la carpintería robotizada, estas tecnologías han sido aplicadas para optimizar la calibración, supervisión y ejecución de tareas donde la precisión geométrica y la adaptabilidad son fundamentales.

Así es como disciplinas como la construcción robotizada y la carpintería han comenzado a incorporar estas tecnologías para enfrentar desafíos propios del trabajo con materiales variables como lo es la madera, donde la calibración y el reajuste constante debido a su variabilidad geométrica resultan esenciales para asegurar la máxima precisión en mecanizados. Esto ha permitido explorar modelos de trabajo híbridos, donde los robots colaboran con los operarios o diseñadores, apoyando en tareas repetitivas, de alto detalle o que requieren precisión sostenida para reforzar la comprensión del operario de las acciones de la máquina y disminuir la carga cognitiva asociada al uso de dispositivos tradicionales.

De este modo, el estado del arte a continuación reúne investigaciones que abordan la colaboración humano robot (HRC), los métodos de control por voz y su implementación en flujos de trabajo constructivos, ofreciendo el marco conceptual y tecnológico que sustenta la propuesta de un sistema de control por voz para un robot industrial KUKA orientado a procesos de fabricación en madera.

1.3.1 Colaboración humano robot (HRC)

La colaboración humano-robot (HRC) se entiende como el conjunto de configuraciones en que personas y robots comparten un mismo entorno de trabajo, que combina las capacidades de las personas con la eficiencia y precisión de una máquina, coordinando acciones de manera simultánea o alternada con distintos grados de proximidad física e intercambio de información. A diferencia de las celdas industriales tradicionales —donde el robot opera aislado y con trayectorias rígidamente preprogramadas— la HRC busca combinar la flexibilidad cognitiva humana con la precisión, repetitividad y capacidad de carga de los robots, habilitando flujos de trabajo más adaptativos, seguros y eficientes en tareas de manufactura, inspección y montaje. En el contexto de la arquitectura y, en particular, de la construcción y la fabricación digital en madera, este paradigma resulta especialmente relevante, ya que permite abordar procesos que requieren tanto decisiones de alto nivel (alineamiento, inspección, reajuste geométrico) como operaciones repetitivas o físicamente demandantes (mecanizados, perforaciones, desplazamientos guiados en obra).

En la literatura reciente se identifican tendencias convergentes: la incorporación de robots colaborativos capaces de compartir espacio físico con personas; el desarrollo de interfaces más intuitivas basadas en lenguaje natural, gestos y visión 3D; la integración de modelos de lenguaje y técnicas de aprendizaje profundo para traducir instrucciones habladas en acciones robóticas seguras y repetibles. Sobre este trasfondo, se revisan a continuación tres trabajos que abordan la HRC desde distintas configuraciones tecnológicas reconocimiento de voz, modelos de lenguaje y generación de trayectorias; que resultan directamente pertinentes para la definición de un flujo de trabajo controlado por voz aplicado a robots industriales KUKA en procesos de fabricación en madera.

El trabajo de Bingol y Aydogmus (2020) desarrolla un sistema de interacción humano-robot basado en reconocimiento de voz en idioma turco para ejecutar tareas predefinidas en un robot industrial KUKA KR Agilus KR6 R900 sixx. El objetivo es que

personas sin experiencia en programación robótica puedan operar el sistema mediante órdenes habladas, activadas por una palabra clave y procesadas por un módulo de reconocimiento automático del habla (ASR) basado en redes neuronales profundas. Dicho módulo convierte el audio en texto, corrige ambigüedades propias del turco y, en función del comando interpretado, activa un flujo que combina visión artificial e imagen digital para localizar objetos y un controlador difuso para ejecutar movimientos de recolección, alineamiento o perforación. En este sistema se destacan:

- El uso de un módulo ASR basado en *deep neural networks* (p-DNN) entrenado específicamente para turco, que alcanza una tasa de acierto de palabras en torno al 90,37 %, mostrando que es posible utilizar el habla como canal principal de comunicación en un entorno industrial ruidoso.
- La integración de visión artificial de bajo costo con un esquema de control difuso, que permite determinar con precisión la posición del objeto y guiar al robot con errores inferiores a 1 mm, aun utilizando una cámara económica.
- La demostración de que usuarios no expertos pueden programar y poner en marcha aplicaciones complejas de HRC mediante comandos de voz, reduciendo la dependencia de interfaces tradicionales del controlador y abriendo la puerta a esquemas de programación más accesibles y multimodales.
- Retroalimentación: Se dispone de una pantalla de información donde se tiene la vista de la cámara, la simulación del movimiento, una retroalimentación en texto y emojis para el operador (Fig. 09)

Esta aproximación presentada por Bingol y Aydogmus mejora la accesibilidad y eficiencia de manera segura en el entorno de trabajo, que incluye la simulación *offline* y validación visual de cada ejecución de trayectorias. El flujo de trabajo presentado

presenta un antecedente técnico directo que respalda la propuesta de la implementación de control por voz para un robot industrial KUKA. Donde la voz se convierte en comandos espaciales ejecutables por el robot. Además, la obtención de tasas de acierto cercanas al 90% del reconocimiento y proceso de palabras respalda el control por voz en entornos robustos de trabajo propensos al ruido y las variaciones dialectales. De este modo, su investigación no solo confirma la viabilidad técnica de la colaboración humano-robot mediada por voz en un robot industrial, sino que aporta un precedente directo para extrapolar este paradigma a procesos de calibración y mecanizado en madera, donde la continuidad del flujo de trabajo, la reducción de errores humanos y la accesibilidad para usuarios no expertos son igualmente críticas.



Fig. 09. Ejemplo de interacción humano-robot. Bingol y Aydogmus (2020)

Más recientemente, Tasneem y Pieters proponen un sistema colaborativo de inspección visual que utiliza modelos de lenguaje de gran escala (LLMs) desplegados localmente para generar código robótico a partir de instrucciones habladas. En esta aproximación, el operario se comunica con un robot colaborativo Franka Emika equipado con una cámara RGB-D Intel RealSense D435 montada en el efector final, emitiendo instrucciones en lenguaje natural que se transcriben mediante un modelo *Speech-to-Text* (Whisper). El texto resultante se entrega a un LLM generador de código (Qwen2.5-Coder-1.5B-Instruct), que, siguiendo un *prompt* estructurado, produce funciones específicas para percepción y planificación de trayectorias de inspección. Estas trayectorias se ejecutan a través de una arquitectura híbrida en la que el LLM resuelve las instrucciones de alto nivel, mientras que ROS se encarga del control determinista de bajo nivel, la cinemática y el planificado de movimiento. La propuesta se valida tanto en entornos simulados como en un caso industrial real de inspección de rodillos de succión a escala, mostrando un sistema de asistencia robótica capaz de manejar objetos de gran tamaño y geometría compleja. En este caso se pone en evidencia que:

- La interacción por lenguaje natural puede ser utilizada para definir recorridos de inspección complejos, donde las rutas se generan en relación a ejes espaciales o direcciones específicas según la intención del usuario, cerrando el ciclo mediante retroalimentación auditiva (TTS).
- Un LLM local, ajustado a generación de código, permite mantener la privacidad de los datos industriales, al tiempo que traduce comandos hablados en secuencias de inspección sobre nubes de puntos y modelos 3D sin requerir entrenamiento específico para cada tarea.
- La arquitectura híbrida que separa planificación de alto nivel (LLM) y control de bajo nivel (ROS) ofrece un compromiso robusto entre flexibilidad, seguridad y ergonomía cognitiva, reduciendo el esfuerzo de programación y facilitando la adopción de la HRC en entornos de fabricación avanzada.

En síntesis, este sistema basado en LLMs constituye una referencia valiosa, ya que demuestra cómo la integración de modelos de lenguaje, percepción 3D y control robótico híbrido puede habilitar interacciones más intuitivas, seguras y eficientes entre humanos y robots.

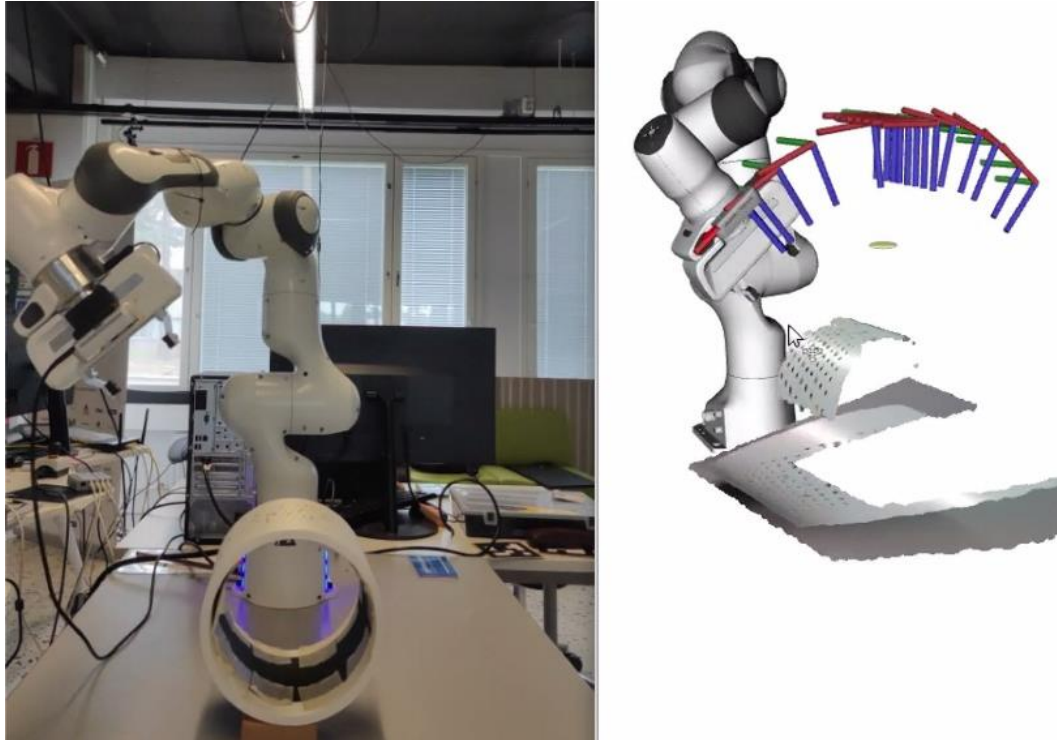


Fig. 10. Resultado de inspección visual en espacio real de trabajo y simulación digital. Tasneem y Pieters (2025).

Por su parte, Bamani et al. introducen el Directive Language Model (DLM), un marco de “speech-to-trajectory” que aborda la colaboración humano-robot desde la generación directa de trayectorias de movimiento a partir de comandos hablados, sin depender de estructuras verbales predefinidas. El sistema se entrena a partir de demostraciones en simulación, donde participantes humanos guían un robot mediante un controlador mientras emiten instrucciones verbales; cada trayectoria se asocia luego a múltiples paráfrasis generadas con GPT-4 para capturar la variabilidad del lenguaje. Las instrucciones se transcriben y se codifican con BERT, y una política

de difusión se encarga de producir trayectorias continuas coherentes con la intención lingüística. El modelo se evalúa frente a diversas arquitecturas de referencia y se valida en un robot cuadrúpedo Unitree Go2, donde es capaz de ejecutar desplazamientos guiados por órdenes verbales explícitas o implícitas (por ejemplo, moverse hacia el operador) manteniendo una alta tasa de éxito y errores de posición y orientación significativamente reducidos.

Gracias al sistema de visión artificial, el control por voz es capaz de adaptarse al espacio de trabajo de la máquina, siendo capaz de aprender sobre su entorno y actuar

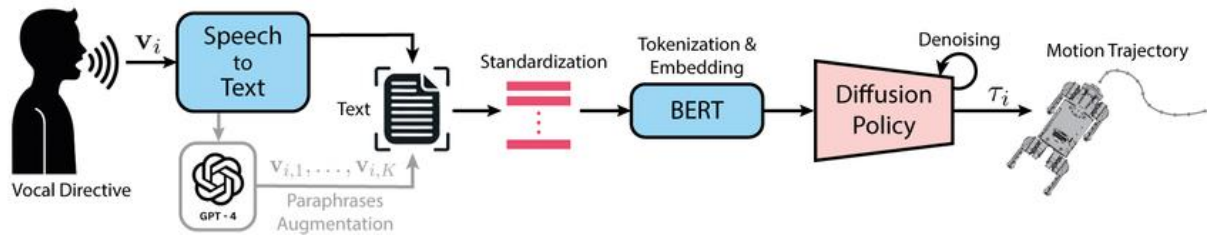


Fig. 11. Illustration of the proposed Directive Language Model (DLM) framework. Bamani, et al. (2025).

sobre él. Lo que permite realizar tareas más complejas cada vez que el sistema se entrena con otras acciones al tener la capacidad de interpretar al operario y poder traducir el contexto a acciones robóticas. Dentro de este marco destacan:

- La traducción directa de comandos hablados en trayectorias de bajo nivel, lo que permite un guiado continuo del robot basado en lenguaje, más allá de listas de órdenes discretas o lenguajes de comandos rígidos.
- El uso combinado de transcripción (Whisper), aumento semántico con GPT-4 y *embeddings* BERT, que dota al modelo de capacidad para generalizar frente a múltiples paráfrasis y expresiones incompletas o ruidosas, manteniendo comportamientos robóticos consistentes.

- Una arquitectura de difusión para la generación de trayectorias, que alcanza tasas de éxito superiores y errores menores que otros modelos comparativos, sin sacrificar la viabilidad de operación en tiempo casi real.

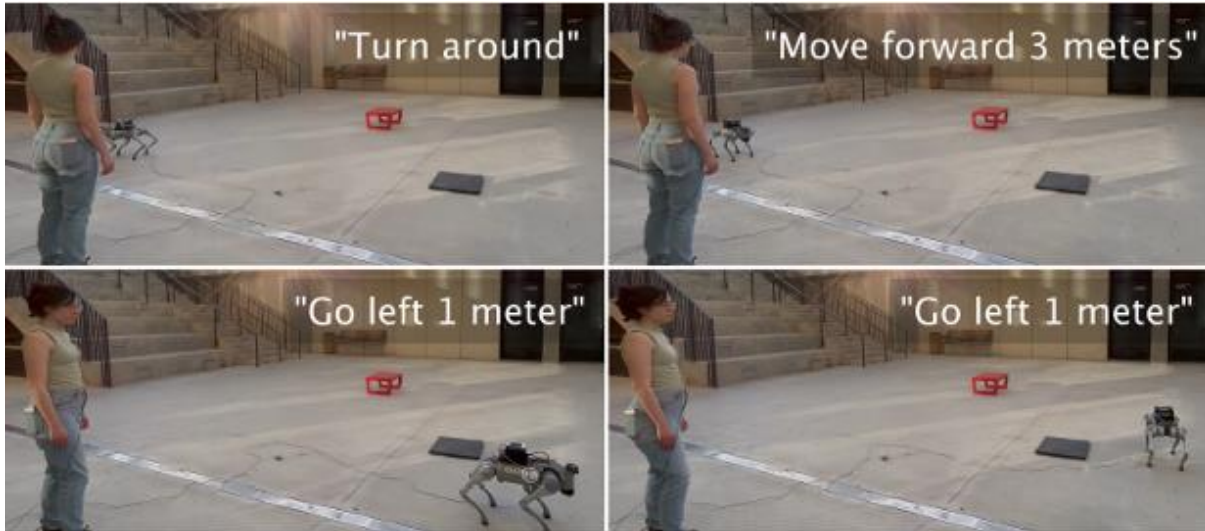


Fig. 12. Sequential robot navigation towards a target square, achieved through four iterations of verbal command input processed by the DLM. Bamani, et al. (2025).

En conjunto, estos tres trabajos evidencian la consolidación de la colaboración humano-robot mediada por voz y lenguaje natural, desde sistemas basados en comandos predefinidos asistidos por visión y control difuso, hasta arquitecturas híbridas con modelos de lenguaje generativos y marcos de aprendizaje *“speech-to-trajectory”*. Todos ellos coinciden en reducir la complejidad de la programación robótica para el usuario no experto, aumentar la naturalidad de la interacción y mejorar la eficiencia y seguridad en tareas de manufactura e inspección. La presente memoria se sitúa sobre este marco, trasladando dichos avances al dominio específico de la construcción robotizada en madera, donde aún no existe una integración explícita entre control por voz, flujos de trabajo paramétricos y ajustes geométricos de alta precisión en robots industriales KUKA, lo que justifica el desarrollo y validación de un prototipo orientado a este contexto.

1.3.2 Interfaces de control por voz

Las interfaces de control por voz corresponden a sistemas de interacción en los que el canal principal entre el usuario y el robot es el habla. Técnicamente, se estructuran en torno a un motor de reconocimiento automático del habla (*Speech-to-Text*, STT) que convierte el audio capturado por un micrófono en texto, un módulo de interpretación que traduce ese texto a comandos estructurados y, en muchos casos, un sistema de síntesis de voz (*Text-to-Speech*, TTS) que entrega retroalimentación auditiva al operador. Este tipo de interfaz presenta ventajas evidentes frente a los métodos tradicionales basados en *teach pendants* o paneles de control: permite operar manos libres, reduce la curva de aprendizaje para usuarios no expertos y hace posible mantener la atención visual en la pieza de trabajo o el entorno físico en lugar de en una pantalla. No obstante, también introduce limitaciones relevantes, como la sensibilidad al ruido ambiental, la necesidad de restringir o entrenar el vocabulario para evitar ambigüedades, los errores de reconocimiento que pueden comprometer la seguridad si no existen capas de validación, y en algunos casos la dependencia de servicios en la nube y la latencia asociada.

En el contexto de la fabricación digital y la construcción, las interfaces por voz se han explorado principalmente como mecanismo de programación o activación de tareas repetitivas, y como complemento a otros canales de interacción como la visión artificial o la programación visual. Los dos trabajos revisados a continuación —un prototipo académico orientado a la inclusión en contextos de diseño arquitectónico y un manipulador orientado a aplicaciones industriales— permiten identificar patrones comunes en la arquitectura de estos sistemas y ofrecen antecedentes directos para el diseño del flujo de trabajo propuesto en esta memoria.

El proyecto Blabber Bot, desarrollado en el Institute for Advanced Architecture of Catalonia (IAAC), plantea un sistema de colaboración humano-robot cuyo objetivo es crear un entorno inclusivo y fácil de usar, donde usuarios sin experiencia en robótica puedan interactuar con un brazo colaborativo mediante comandos de voz y recibir respuestas habladas del sistema. La interfaz se basa en un flujo relativamente

compacto: el usuario emite instrucciones a través de un micrófono, el audio se captura en tiempo real (mediante *PyAudio* en un entorno de programación visual como *Grasshopper*), se transcribe a texto mediante un motor STT y este texto se publica en un *broker* mediante el protocolo MQTT; desde allí, un *script* de control robótico suscrito a los mensajes interpreta el comando y envía las órdenes correspondientes al robot. Paralelamente, un módulo TTS procesa cadenas de texto generadas por el sistema (confirmaciones, alertas, mensajes de error) y las reproduce en tiempo real, cerrando el ciclo de diálogo entre humano y máquina.

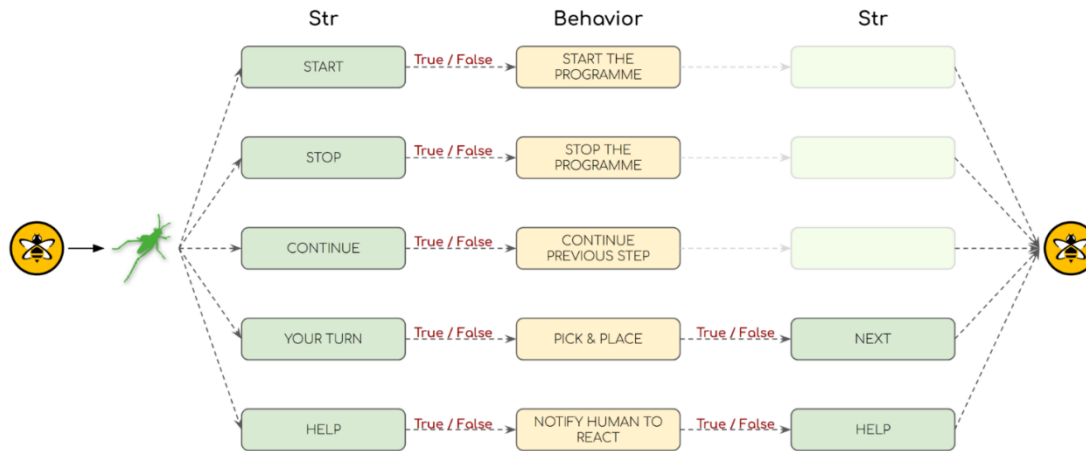


Fig. 13. Conjunto de comandos del sistema Blabber Bot. Pardo et al. (2024)

El comportamiento del sistema se organiza en torno a un conjunto acotado de comandos simbólicos (“*START*”, “*STOP*”, “*CONTINUE*”, “*YOUR TURN*”, “*HELP*”), cada uno asociado a una condición lógica y a una acción específica sobre el robot, como iniciar o detener un programa, repetir una secuencia de *pick and place* o solicitar intervención humana (Fig. 13). Dentro de esta arquitectura se destacan:

- El uso de STT y TTS como capa de interfaz principal, lo que permite que la interacción se produzca casi exclusivamente mediante lenguaje hablado, con un conjunto reducido de palabras clave fáciles de recordar para usuarios no expertos.

- La integración de MQTT como *middleware* desacoplado entre la captura de comandos y el control del robot, facilitando que distintos módulos (interfaz de usuario, lógica de control, simulación en *Grasshopper*) se comuniquen mediante mensajes de texto sin compartir un mismo entorno de ejecución.
- La representación gráfica del flujo en un entorno de programación visual (*Grasshopper*), que permite a estudiantes y diseñadores comprender de forma transparente la cadena completa desde el audio hasta el movimiento robótico, y adaptar el sistema a nuevos comportamientos sin necesidad de reprogramar desde cero (Fig. 14).

Aunque se trata de un prototipo académico y no de un sistema industrial robusto, Blabber Bot demuestra que las tecnologías de STT/TTS pueden integrarse de forma relativamente ligera con un brazo colaborativo a través de *Grasshopper*, constituyendo una referencia directa para esta memoria en cuanto a la estructura básica de la interfaz: captura de voz, interpretación de comandos, publicación de mensajes a un sistema de control visual-paramétrico y retroalimentación auditiva. Permitiendo aplicar e integrar esta tecnología en el software seleccionado como interfaz para el proyecto desarrollado en esta memoria, incorporando los elementos de reconocimiento de voz, retroalimentación auditiva y programación visual.

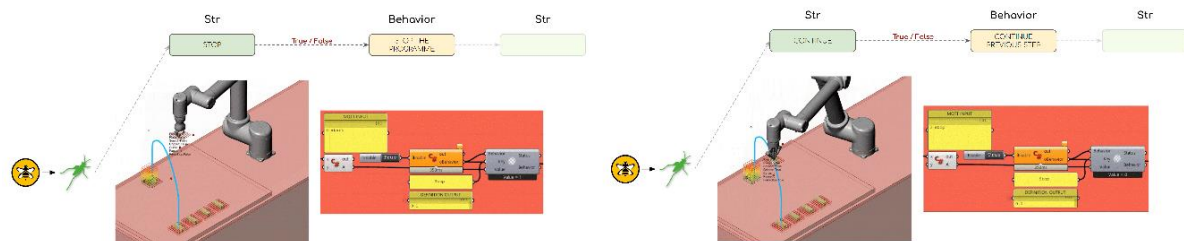


Fig. 14. Entorno de programación Grasshopper. Pardo et al. (2024)

Un enfoque más orientado a aplicaciones industriales se presenta en el trabajo de Priyadarshana et al. (2022), donde se desarrolla un manipulador controlado por voz para tareas de *pick and place* y enseñanza de trayectorias, utilizando un brazo Kinova MICO de 6 grados de libertad y un sensor *Kinect V2* como sistema de visión. El sistema emplea *Windows Speech Recognition* como motor de STT, accedido mediante un programa en C# que define el conjunto de gramáticas válidas, y combina esta capa de reconocimiento con un módulo de comprensión de palabras clave (*Keyword Understanding Module*, KUM) que filtra y clasifica las órdenes de usuario ("*go to*", "*pick the cube*", "*place the cube*", "*repeat the task*", etc.). En paralelo, el *Kinect* captura imágenes de color y profundidad; estas se procesan mediante el SDK de *Kinect 2.0* y *OpenCV 3.1* para segmentar los objetos de interés, eliminar el fondo mediante información de profundidad, detectar formas circulares mediante transformada de *Hough* y calcular la posición 3D del objeto en el sistema de referencia de la cámara.



Fig. 15. Estación de trabajo. Priyadarshana et al. (2022)

Toda esta información converge en el *Interaction Decision Making Module* (IDMM), que define la trayectoria, el punto final y la velocidad del manipulador en función de los comandos de voz y de la escena observada. La lógica de funcionamiento puede describirse como un flujo secuencial:

- Procesamiento de voz: los comandos se capturan por micrófono, se transcriben mediante *Windows Speech Recognition* y se analizan en el KUM, donde se comparan con un conjunto de frases esperadas para determinar la intención.
- Procesamiento de visión: las imágenes RGB y de profundidad del *Kinect* se sincronizan y se filtra el entorno para aislar el área de trabajo; se eliminan los píxeles fuera del rango útil, se convierte el espacio de color a HSV y se segmenta el objeto por umbralización y operaciones morfológicas, obteniendo coordenadas 3D en el espacio de la cámara.
- Integración y planificación: el IDMM recibe los comandos interpretados y las posiciones de los objetos, evalúa si el objetivo se encuentra dentro del alcance del manipulador y, en caso afirmativo, genera la trayectoria correspondiente (por ejemplo, acercarse, tomar el objeto, desplazarlo y soltarlo), almacenando en algunos casos las trayectorias para su repetición futura.
- Ejecución y evaluación: las pruebas realizadas muestran una tasa de reconocimiento de voz cercana al 85% bajo condiciones normales y una probabilidad de detección de objetos en torno al 0,9 tras ajustar los valores HSV, indicadores suficientes para considerar el enfoque más intuitivo y fácil de usar que el *teach pendant* tradicional en los escenarios ensayados.

Este tipo de sistema tiene una alta complejidad y desglose para la ejecución de acciones robóticas, que sirve de referencia para entender cómo se distribuye la información para poder traducirla y ejecutarla en un sistema robótico (Fig. 16). La

experiencia reportada indica que la combinación de STT con un sistema de visión robusto aumenta la fiabilidad del manipulador controlado por voz y facilita la enseñanza de tareas repetitivas, al igual que hace posible más accesible la colaboración humano-robot en entornos en que el operario mantiene las manos ocupadas o debe desplazarse en el espacio de trabajo. Si bien el manipulador utilizado es una plataforma de investigación y no un robot industrial de gran carga, los autores enfatizan que el concepto es transferible a aplicaciones industriales, siempre que se mejore la tasa de reconocimiento y se incorporen mecanismos adicionales de seguridad.

Fig. 16. Flujo de trabajo. Priyadarshana et al. (2022)

1.3.3 Aplicaciones en construcción o carpintería robotizada

En el ámbito de la construcción, la robótica se ha aplicado principalmente a tareas de manipulación, ensamblaje y mecanizado, con especial énfasis en operaciones de *pick and place*, montaje de paneles y asistencia en la manipulación de elementos pesados o de gran formato. En madera, la automatización se ha concentrado históricamente en la prefabricación seriada mediante CNC y sistemas robotizados en planta, mientras que las fases de montaje en obra, ajuste fino de uniones y tareas de replanteo siguen siendo mayoritariamente manuales debido a la variabilidad del material y al carácter no estructurado del entorno. Ejemplos representativos incluyen robots articulados utilizados para el mecanizado de uniones de alta precisión en elementos de madera, brazos robotizados sobre bases móviles para asistencia en el montaje de cubiertas, y sistemas de manipulación robotizada de paneles (yeso-cartón, tableros o módulos de fachada) en operaciones repetitivas de instalación.

En este contexto, Park et al. (2024) proponen un marco para la interacción humano-robot basada en instrucciones de lenguaje natural aplicado a operaciones de *pick and place* en obras de construcción, validado en la instalación de paneles de yeso cartón. El sistema se estructura en tres módulos: (1) un módulo de *Natural Language Understanding* (NLU), que etiqueta secuencialmente cada palabra de la instrucción para extraer el objeto objetivo, el destino y la orientación de montaje; (2) un módulo de *Information Mapping* (IM), que mapea estas entidades sobre información de componentes constructivos proveniente de un modelo BIM y del historial de acciones; y (3) un módulo de *Robot Control* (RC), que genera y ejecuta trayectorias libres de colisión para un brazo KUKA de 6 grados de libertad sobre un eje lineal, simulado mediante ROS, *Gazebo* y *Movelt*. Las figuras de configuración del caso de estudio, el robot entre una pared de montantes y el área de acopio de paneles, así como los distintos patrones de colocación vertical y horizontal, ilustran la escena y las estrategias de instalación ensayadas

El sistema se apoya en un modelo BERT ajustado para etiquetado de secuencias, el cual alcanza precisiones cercanas al 100% a nivel de palabra e instrucción en la identificación de las entidades relevantes de cada comando, incluso con conjuntos de datos relativamente acotados. La arquitectura permite describir las piezas mediante

identificadores, dimensiones o relaciones espaciales y traducir dichas descripciones en comandos formales de control que el RC ejecuta en el gemelo digital del robot. En este trabajo se destacan:

- La formalización de un flujo completo desde instrucciones verbales hasta trayectorias ejecutables, encapsulado en la tríada NLU–IM–RC, que integra lenguaje natural, información BIM y planificación de movimiento en un entorno de robot KUKA.
- La generación de un *dataset* específico de instrucciones de instalación de paneles con anotación fina de atributos (ID, dimensiones, localización y método de colocación), lo que permite entrenar modelos de lenguaje capaces de extraer información relevante para tareas constructivas reales.
- La demostración de secuencias completas de montaje de paneles en un gemelo digital con elevadas tasas de éxito, mostrando que las operaciones de instalación de tabiques, cielos y otros elementos planos pueden ser comandadas mediante instrucciones de lenguaje natural sin recurrir a interfaces técnicas tradicionales.

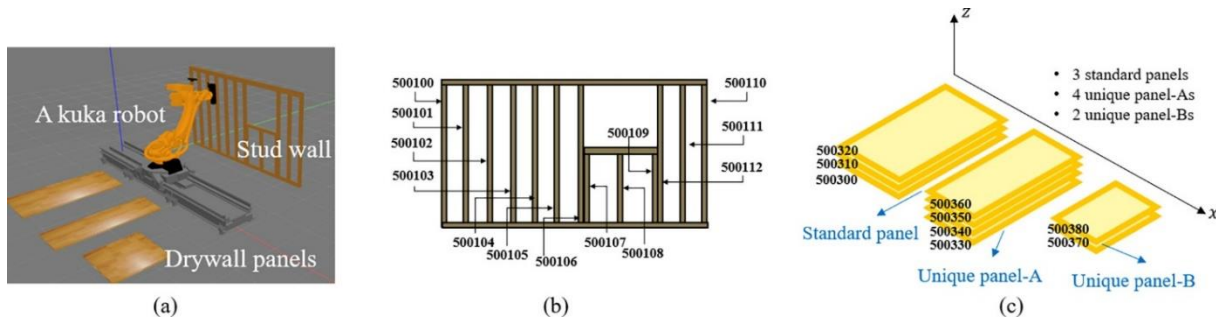


Fig. 17. Tipos de paneles para montaje con robot KUKA. Park et al. (2024)

Aunque el caso de estudio se centra en paneles de yeso, el marco planteado es extensible a otras operaciones de montaje repetitivo en construcción como instalación de paneles de fachada, revestimientos o elementos modulares, y constituye un referente directo para pensar sistemas de asistencia robotizada en , donde la selección del elemento, la posición de montaje y la orientación de la pieza pueden expresarse discursivamente y resolverse mediante un modelo de lenguaje y un planificador geométrico.

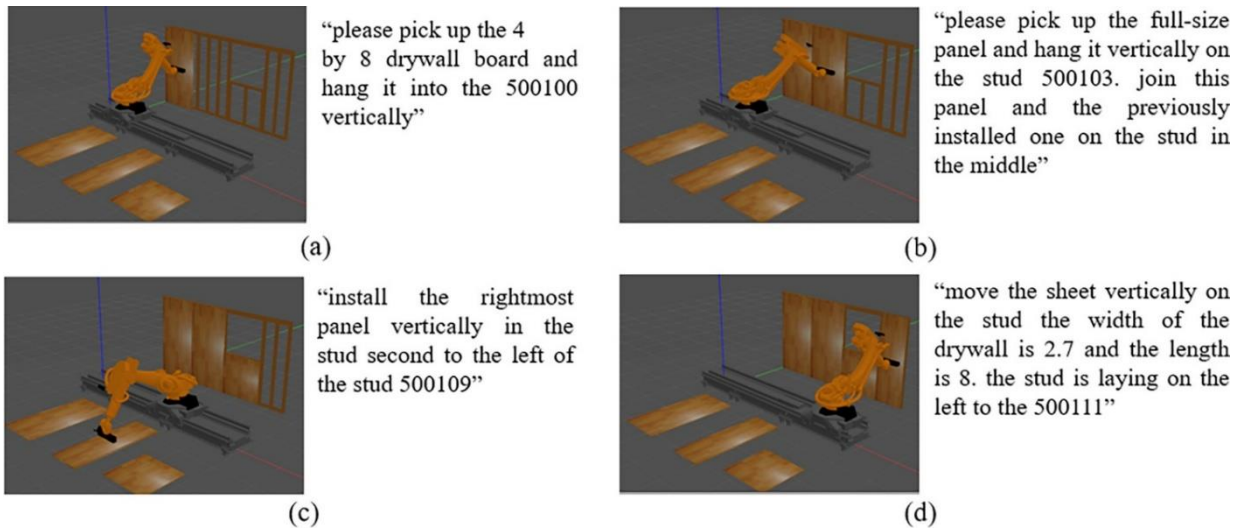


Fig. 18. Montaje de paneles de yeso cartón según instrucciones habladas. Park et al. (2024)

Este trabajo da una aplicación relacionada con la construcción que apoya la proyección de la investigación desarrollada, utilizando un robot KUKA de características similares al que es objeto de investigación en este caso, lo cual implica que efectivamente resulta en una adición funcional de la máquina al área de construcción.

En el ámbito específico de la construcción en madera, la investigación de Landázuri (2022) examina el estado del arte de la automatización con robots industriales y propone escenarios de Colaboración Humano-Robot (HRC) para procesos tradicionalmente manuales, con especial foco en estructuras de cubierta. La autora identifica que la automatización actual se concentra en la prefabricación seriada en

taller mediante CNC y células robotizadas mientras que tareas como el mecanizado de uniones no estandarizadas, el transporte interno de piezas y el ensamblaje in situ de cerchas o pórticos siguen recayendo en mano de obra altamente especializada. Se argumenta que los robots articulados de fabricantes como ABB y KUKA, gracias a su precisión, carga útil y alcance, son candidatos idóneos para abordar estas tareas en entornos no estructurados, siempre que se integren sensores, cámaras y *LiDAR* que habiliten la comunicación multimodal y mecanismos seguros de HRC.

Uno de los casos más relevantes es el ensamblaje in situ de un pabellón (Fig 19) y de cerchas *king post* de madera, donde se propone un flujo de trabajo híbrido que combina mecanizado robotizado y asistencia robótica en montaje. La cercha y el flujo de trabajo tradicional que distinguen etapas *off-site* (*layout*, corte, limpieza, lijado, aplicación de acabado y ensamblaje de cerchas en taller) y *on-site* (transporte, posicionamiento y montaje final)— muestran cómo la incorporación de un robot KUKA KR270 sobre una plataforma móvil KUKA omniMove permitiría relocalizar gran parte de la fabricación y montaje directamente en obra, eliminando etapas de transporte y reorganización de elementos.



Fig. 19. Montaje de estructura en madera. Landázuri (2022)

En este escenario, el robot asume tanto el mecanizado de uniones como el transporte y posicionamiento de piezas pesadas, mientras que el operario mantiene el control de alto nivel mediante comandos multimodales.

La propuesta enfatiza que, para fases críticas como el acople de uniones tipo caja y espiga en la cercha, los comandos de alto nivel; particularmente voz y control háptico, podrían utilizarse para reprogramar y ajustar la trayectoria del robot en tiempo real, de modo que el elemento de madera se alinee con precisión respecto de la pieza ya instalada. De esta manera, se plantea un esquema de HRC en el que:

- Los robots industriales articulados se encargan del mecanizado de uniones complejas y del transporte de elementos pesados, reduciendo el esfuerzo físico y el riesgo para los trabajadores.
- La obra se configura como un “taller ampliado”, donde la materia prima llega directamente al sitio y las tareas de corte, ajuste y ensamblaje se organizan en torno a la celda robotizada, acortando la cadena logística tradicional.
- La interacción multimodal (voz, feedback háptico, sensores de proximidad y LiDAR) se considera un requisito para lograr un control intuitivo y seguro del robot en procesos de ensamblaje de precisión, especialmente en , donde los ajustes milimétricos y las tolerancias geométricas son determinantes para la estabilidad de la estructura.

En conjunto, los trabajos revisados en esta subsección muestran que la robótica en construcción y carpintería se orienta crecientemente hacia esquemas de HRC donde el robot asiste en tareas de mecanizado, manipulación y montaje, mientras que el humano conserva la planificación de alto nivel y las decisiones de ajuste fino. Park et al. demuestran que operaciones de *pick and place* en obra pueden ser comandadas mediante instrucciones de lenguaje natural estructuradas, apoyadas en modelos de lenguaje y gemelos digitales; Landázuri demuestra que los robots industriales en

madera pueden migrar desde la prefabricación seriada a escenarios de ensamblaje in situ, siempre que se disponga de interfaces de control intuitivas, incluida la voz, y de un andamiaje de sensores que asegure la operación segura en entornos no estructurados.

En conjunto, el estado del arte revisado permite afirmar que la Colaboración Humano-Robot y las interfaces de control por voz constituyen hoy un campo tecnológicamente maduro y metodológicamente robusto sobre el cual se apoya directamente la propuesta de esta memoria. Por una parte, los distintos trabajos analizados demuestran de forma consistente que es posible delegar en la voz buena parte de la interacción con robots industriales y colaborativos, tanto en tareas de manufactura (*picking*, alineamiento, perforación), inspección visual avanzada o montaje en obra, validando tasas de reconocimiento superiores al umbral necesario para un uso fiable en entornos ruidosos y no controlados, y confirmando que usuarios no expertos pueden operar sistemas robóticos complejos sin recurrir siempre al *teach pendant* ni a lenguajes de programación especializados.

Por otra parte, se observa una convergencia clara en las tecnologías de base: *Python* se consolida como lenguaje de orquestación para integrar módulos de reconocimiento de voz, visión artificial, planeamiento de trayectorias y control de robots; entornos como ROS, *MoveIt* y diversas APIs de fabricantes articulan la capa de control de bajo nivel; y, en el ámbito más cercano a la arquitectura, *frameworks* visuales como *Rhino* y *Grasshopper* se utilizan para construir flujos de trabajo paramétricos en los que se simulan, verifican y ajustan las trayectorias de robots que en muchos casos marca KUKA, antes de su ejecución física.

Esta coincidencia tecnológica otorga una validación indirecta a las decisiones de diseño del sistema desarrollado en la memoria: la elección de un robot industrial KUKA, la integración con *Grasshopper* y *KUKA|prc*, el uso de *Python* como puente hacia plataformas de simulación y ejecución como *RoboDK*, así como la incorporación de motores STT/TTS y, cuando corresponde, de modelos de lenguaje para estructurar comandos. A nivel de arquitectura de sistemas, los trabajos revisados coinciden en un patrón modular que se vuelve referencia directa: captura de audio mediante micrófonos, transcripción con motores STT especializados, normalización y análisis

de comandos, mapeo de esa información a estructuras espaciales (coordenadas, nubes de puntos, modelos BIM o definiciones paramétricas), validación previa en un entorno simulado y, finalmente, ejecución controlada sobre el robot físico con mecanismos explícitos de seguridad. Esta lógica se extiende también a los casos específicos de construcción y carpintería robotizada, donde se exploran gemelos digitales, bases móviles y estrategias de ensamblaje in situ que incorporan comandos de voz como capa de alto nivel para reajustar trayectorias y afinar el montaje de elementos de madera. Sin embargo, el mismo cuerpo de literatura permite identificar vacíos precisos que enmarcan el aporte de esta memoria: la mayoría de las soluciones se orientan a manipuladores de laboratorio o *cobots* de menor escala, sin abordar de manera directa el problema de la calibración geométrica y el reajuste continuo del sistema de coordenadas de la pieza de trabajo en procesos de construcción robotizada ; en pocos casos se integran de forma explícita las capacidades de control por voz con flujos de trabajo paramétricos en Grasshopper vinculados a robots KUKA de gran capacidad; y la variabilidad dimensional de la madera, así como las implicancias materiales del mecanizado de uniones complejas, suele quedar subsumida bajo supuestos de geometría ideal o procesos de prefabricación altamente estandarizados.

Frente a este panorama, la propuesta de un prototipo de sistema de control por voz para un robot industrial KUKA orientado a procesos de fabricación en madera se sustenta en un conjunto de evidencias sólidas que validan la accesibilidad, eficacia y seguridad de la voz como interfaz, pero al mismo tiempo se posiciona en un nicho poco explorado: la integración directa entre control por voz, simulación visual-paramétrica y calibración en tiempo real del sistema de coordenadas en construcción robotizada, con énfasis en mejorar la continuidad del flujo de trabajo, reducir errores humanos y aumentar la usabilidad para estudiantes, diseñadores y operarios no expertos en robótica industrial.

1.4 OBJETIVOS

1.4.1 Objetivo general

Desarrollar y validar un prototipo de sistema de control por voz que permita una colaboración segura y adaptable entre un operador humano y un robot industrial, **aplicado a un proceso de construcción con madera.**

1.4.2 Objetivos Específicos

- **Objetivo Específico 1: Analizar y seleccionar tecnologías de reconocimiento de voz** compatibles con la colaboración humano-robot en construcción con madera, evaluando métodos de entrada de audio, procesamiento y traducción de voz a comandos de control.
- **Objetivo específico 2: Diseñar e implementar el prototipo de control por voz**, integrando hardware y software con el robot industrial, asegurando su **funcionamiento en condiciones seguras y adaptables al proceso de construcción con madera.**
- **Objetivo Específico 3: Validar experimentalmente la integración funcional del prototipo en un entorno de laboratorio (TRL 4)**, aplicado a un proceso de construcción con madera, demostrando su funcionamiento integrado como sistema y garantizando una colaboración humano-robot segura y adaptable.

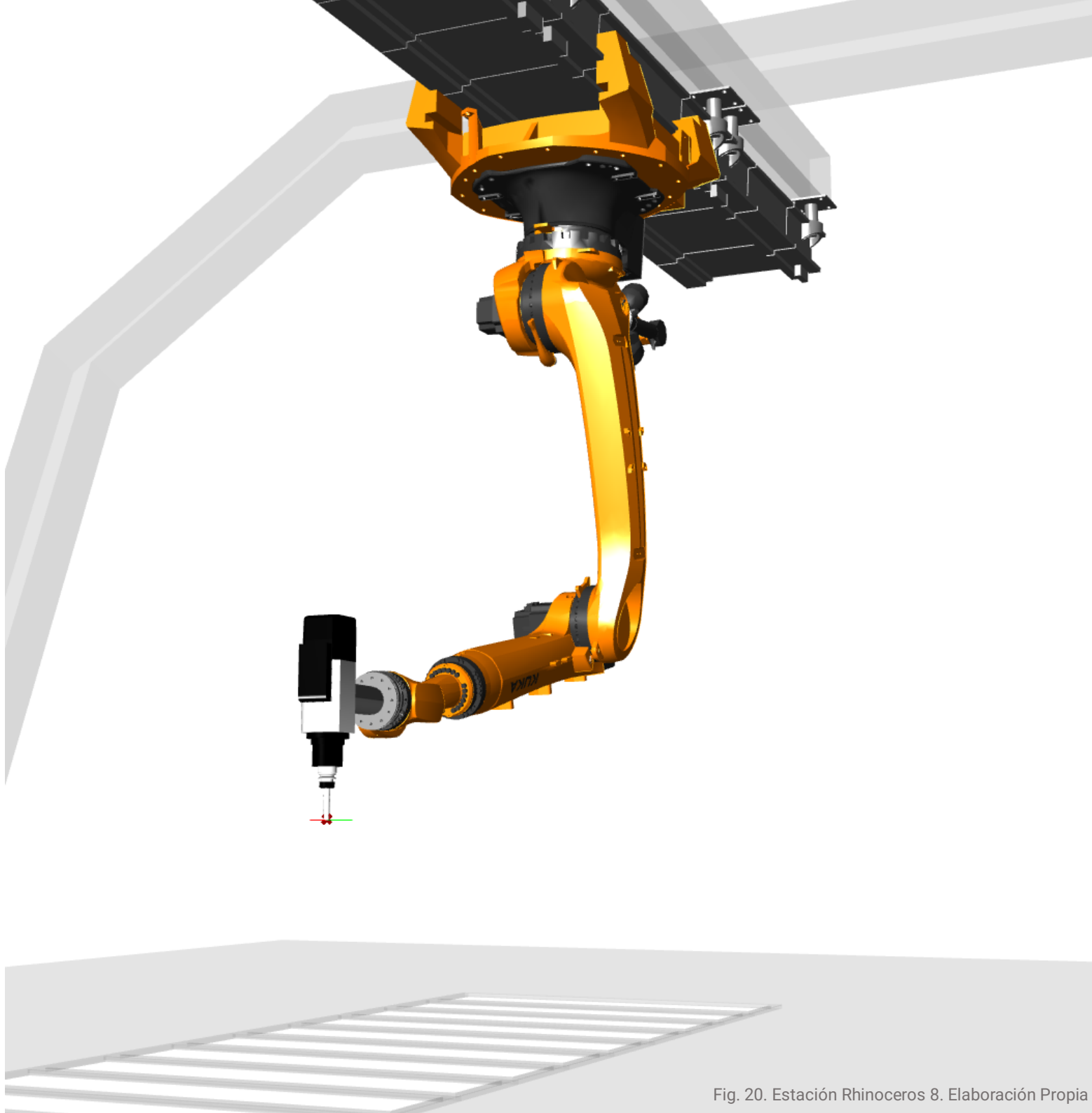


Fig. 20. Estación Rhinoceros 8. Elaboración Propia

1.5 METODOLOGÍA

Este sistema comienza con la captura y reconocimiento de audio mediante un motor *Speech To Text* (STT), que transcribe el audio en texto. Este texto es interpretado como un comando robótico que se simula de manera *offline* en el entorno de programación visual *Grasshopper*, utilizando el plugin *KUKA|prc* para visualizar y validar la acción antes de su ejecución online a través de un programa en *Python* integrado en *RoboDK*. Esta integración busca no solo facilitar el control operativo, sino también lograr una interacción humano-robot más fluida, empleando el lenguaje natural como interfaz principal de control.

A partir de esta base, la arquitectura del sistema se concibe como un flujo modular en el que cada componente cumple una función específica para garantizar precisión, seguridad y continuidad operativa. El micrófono inalámbrico captura la señal de voz incluso en entornos ruidosos; el motor STT procesa la transcripción con un formato estandarizado; un módulo de interpretación semántica en *Python* traduce las instrucciones a vectores de movimiento; y la simulación en *KUKA|prc* ofrece validación visual previa a la ejecución física. Finalmente, un sistema de retroalimentación auditiva basado en *Text-to-Speech* (TTS) confirma al operador la acción interpretada, mientras la API de *RoboDK* gestiona la comunicación directa con el robot. Esta estructura integral no solo optimiza la calibración y control del robot, sino que sienta las bases para una expansión futura hacia operaciones más complejas en la construcción robotizada en madera.



Fig. A. Medición tradicional de SC. Elaboración Propia

1.5.1 Etapas de desarrollo

1. **Integración de reconocimiento de voz a Grasshopper:** Se inicia con la **selección comparativa de motores STT** en *Python*, importándolos como librerías en componentes *GhPython*. A partir de ahí se delimita un **vocabulario operativo de comandos** y se programa la lógica de interpretación dentro de *Grasshopper*, asegurando que cada frase hablada se traduzca a un formato de datos coherente para el sistema.
2. **Simulación offline:** Los comandos ya interpretados se asocian a trayectorias del robot generadas con *KUKA|prc*; dichas trayectorias se prueban en un entorno virtual, evaluando tanto la correcta transcripción como la viabilidad cinemática. En esta fase se monitorean latencias, errores de reconocimiento y posibles colisiones, depurando el flujo antes de cualquier ejecución física.
3. **Simulación Online:** Se define la infraestructura de conexión directa con el robot mediante *RoboDK* y se ensayan los mismos comandos en condiciones controladas de taller. El énfasis recae en corroborar la respuesta inmediata del robot, la robustez frente a ruido ambiental y la activación de salvaguardas de seguridad ante entradas erróneas.
4. **Validación y documentación:** Finalmente, el sistema se somete a un **experimento colaborativo real con tareas de construcción en madera**, midiendo eficiencia, precisión y seguridad con usuarios finales. Los datos obtenidos permiten ajustar el léxico de comandos, los umbrales de reconocimiento y los parámetros cinemáticos, cerrando el ciclo de mejora continua antes de su transferencia a entornos productivos.

1.5.2 Componentes del sistema

SOFTWARE

- **API Speech to Text (STT):** Se utiliza el motor **Microsoft Azure STT SDK** para el reconocimiento de voz y transcripción de instrucciones en español.
- **Procesamiento del lenguaje:** Se implementa un código en *Python* que recibe la transcripción del audio y procesa el texto mediante la librería **Text2Num** para establecer números en cifras y dar un formato predeterminado.
- **Interpretación comandos:** A través de un diccionario de acciones predeterminadas se traduce el texto a vectores de movimiento que luego se convierten a movimientos del robot, utilizando **KUKA/prc**. De esta manera, se puede simular el movimiento antes de ejecutarlo.
- **Simulación offline:** A través de **Grasshopper** conjunto a **Rhinoceros 8** se previsualiza el movimiento del robot y verificar si el comando fue correctamente interpretado y si es seguro ejecutar la acción.
- **Respuesta Texto a Voz (TTS):** El módulo **Python Text to Speech (PyTTsX3)** reproduce voces en español, entregando una respuesta rápida a la persona usuaria. Esta retroalimentación auditiva proporciona información inmediata que confirma el comando recibido.
- **Ejecución Online:** El comando interpretado se traduce en un vector de movimiento, el cual se ejecuta en línea a través de la **API de RoboDK en Python**.

HARDWARE

- **Micrófono:** Para la captura de voz se utilizó el micrófono inalámbrico **DJI Mic Mini**, el cual ofrece gran cobertura, tecnología de cancelación de ruido y alta fidelidad, incluso en ambientes ruidosos.
- **Robot:** El robot utilizado corresponde al **modelo KUKA KR 210 R3100-2 C**, montado a cielo en el laboratorio de construcción y manufactura robotizada de la Universidad Técnica Federico Santa María Campus Santiago San Joaquín

2. DESARROLLO DEL TEMA

Este capítulo sigue la lógica de la metodología para cumplir los objetivos del trabajo: primero se evalúan tecnologías de reconocimiento de voz en español enfocadas en comandos técnico-numéricos y robustez al ruido; luego se implementa un prototipo que integra STT, *parsing* en *Python*, confirmación TTS y enlace con el flujo visual paramétrico para operar el robot KUKA; finalmente se valida en laboratorio (TRL 4) mediante tareas de calibración de BASE y maniobras de mecanizado simuladas y ejecutadas, verificando precisión, tiempos de respuesta y estabilidad.

2.1 TECNOLOGÍAS DE RECONOCIMIENTO DE VOZ

El reconocimiento de voz Speech to Text (STT) es, en esencia, un sistema que escribe lo que escucha. Primero toma el audio y filtra la voz, luego lo transforma en un espectrograma, que permite ver como cambian los tonos y energía en el tiempo, esa representación entra a un modelo que reconoce patrones dentro del espectrograma que se definen como palabras o sílabas, luego este se procesa según las pausas y tonos para el espaciado y uso de grafías (Jurafsky, & Martin, 2025). Existen sistemas que funcionan a tiempo real, por nube vía internet, analizando grabaciones preexistentes o con modelos locales descargables.

En torno al contexto de prototipo, se prioriza hacer pruebas con motores STT que funcionen en español y estos no sean de pago para poder evaluar su replicabilidad a futuro y que no sea una limitante para su campo de aplicación.

2.1.1 Motores STT

Para seleccionar el motor STT más adecuado, se evaluaron cinco modelos: *Google STT*, *Microsoft Azure STT SDK*, *Whisper (Open AI)*, *Vosk* y *Vosk Small* (Tabla 1). Para evaluar la precisión de transcripción de cada modelo, se definió una lista de 100 instrucciones que abarca cinco formatos de comandos (Fig.22), donde cada comando se compone de una indicación, una letra para especificar el eje del movimiento y un número en milímetros o grados sexagesimales para establecer la magnitud de cada acción con el fin de analizar la eficiencia y precisión de cada uno y realizar los ajustes necesarios.

Tabla 1. Características modelos STT

Modelo STT	Conexión	Consumo de recursos	Idioma	Costo
Azure	Online	Bajo	esES; esMX	Gratis con opción de pago
Google	Online	Bajo	esES; esMX	Gratis con opción de pago
Vosk	Offline	Alto	Español avanzado	Gratis
Vosk small	Offline	Medio	Español básico	Gratis
Whisper	Online/ Offline	Alto	Multilenguaje	Gratis

Fuente: Elaboración propia

COMANDOS DIRECCIONALES

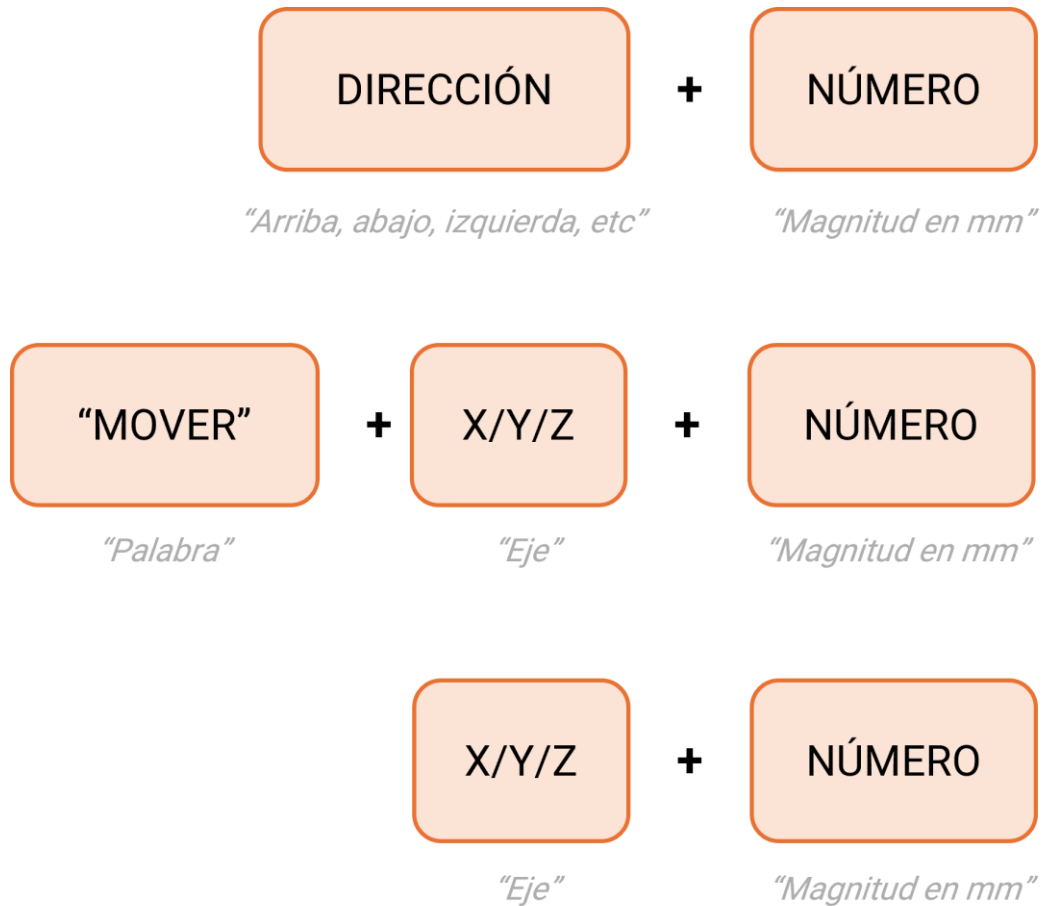


Fig. 22. Tipos de comandos. Elaboración Propia

COMANDOS ROTACIONALES

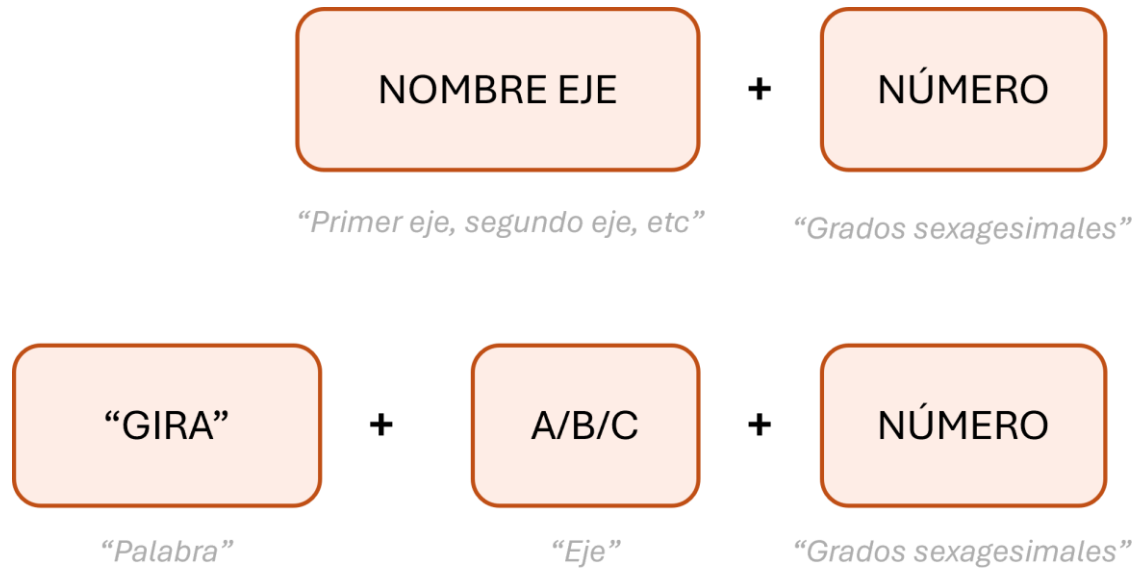


Fig. 22. Tipos de comandos. Elaboración Propia

2.1.2 Pruebas de motores

Las pruebas se organizaron en tres etapas secuenciales con distintas variaciones en las instrucciones: primero, una prueba preliminar utilizando el micrófono nativo del computador, destinada a detectar patrones generales de transcripción; segundo, una prueba con micrófono inalámbrico en condiciones reales, enfocada en identificar errores comunes; y tercero, una prueba final con ajustes al listado de instrucciones, orientada a mejorar el rendimiento del sistema.

Todas las pruebas fueron realizadas a través de un código Python que proporcionaba el listado de 100 instrucciones que el usuario debía repetir con micrófono abierto, comprobando su transcripción y otorgando un porcentaje de aciertos:

```
Function EjecutarReconocimientoVoz()  
    Inicializar variables globales y lista de instrucciones  
    Mientras no se complete la lista y el usuario no escriba  
    'salir'  
        Mostrar instrucción a pronunciar  
        Configurar micrófono y motor STT  
        Capturar audio una vez  
        Si se reconoce la instrucción  
            Convertir texto a minúsculas  
            Quitar puntuación y normalizar espacios  
            Convertir palabras numéricas a cifras  
            Comparar con la instrucción esperada  
            Si coincide, contar como acierto  
            Si no coincide, registrar como error  
        Si no se reconoce, mostrar advertencia  
        Avanzar a siguiente instrucción  
    Mostrar resumen final de resultados  
End
```

1. PRUEBA PRELIMINAR

La primera prueba trata de una evaluación preliminar del sistema para entender el funcionamiento y proceso del lenguaje de cada motor. Esta prueba se realizó con el micrófono nativo del computador a través de un programa de *Python* que contenía el listado de cien instrucciones a evaluar. Con esta prueba se determinó que cada motor tenía su propia manera de escribir como se muestra en la Tabla 2, por lo que se tuvo que establecer un formato predeterminado y procesar la transcripción para coincidir con el listado textual del programa de prueba. Esta etapa preliminar es clave para identificar las diferencias de cada modelo en su formato nativo y procesar los posibles escenarios dentro del código que afecten los resultados de cada prueba.

Tabla 2. Patrones de escritura

Modelo STT	Precisión	Formato	Números	Letras	Velocidad respuesta	% Aciertos
Azure	Muy Alta	Mayúscula y grafías	Cifras	Letra en mayúscula	Rápida (4s)	62%
Google	Media	Tipo oración	Cifras	Letra con número minúscula	Rápida (3s)	33%
Vosk	Muy Alta	Tipo oración	Palabras	Nombre de la letra	Media (5s)	39%
Vosk small	Alta	Tipo oración	Palabras	Nombre de la letra	Rápida (4s)	14%
Whisper	Baja	Incluye símbolos	Cifras	Letra en mayúscula	Lenta (9s)	28%

Fuente: Elaboración propia

Según los resultados de la tabla 2 es que se realiza un procesamiento del texto de la transcripción original de cada modelo, para que el resultado sea lo más textual posible al listado original dentro del código. Como en español existen diversas reglas gramáticas y grafías es que se aplica el siguiente proceso:

- 1. Convertir todo el texto a minúsculas:** Esto se realiza para que no existan confusiones entre las oraciones, las letras que determinan coordenadas o ejes. De modo que quede todo de manera más uniforme. Esto se realiza a través de una fórmula que procesa el texto dentro de *Python*.
- 2. Quitar signos de puntuación:** Algunos modelos incluían signos dentro de cada oración siguiendo las reglas gramaticales del idioma, sin embargo, el código al no tener un procesamiento o inteligencia artificial es que interpreta estos caracteres como un error de precisión del STT al no coincidir textualmente con el listado de instrucciones esperadas. Se incorpora otra fórmula dentro del procesamiento de texto en *Python* que elimina estos caracteres de la transcripción original.
- 3. Escribir números en cifras:** Como se menciona anteriormente, el código interpreta como acierto solamente las transcripciones que coinciden textualmente con el listado de instrucciones; dos modelos escriben los números en palabras, y para solucionarlo se utiliza la librería *text2num* que es capaz de reconocer números escritos en español y pasarlos a cifras dentro del código.

Ante este tipo de errores es que en las siguientes pruebas se aplica una corrección en los porcentajes de acierto de cada modelo, de forma que solo se evalúe su capacidad de transcribir correctamente cada instrucción independientemente del formato textual de la lista de prueba.

2. SEGUNDA PRUEBA

La segunda prueba se realizó en el mismo entorno de trabajo en las condiciones de ruido normales del taller. La entrada de audio fue recibida por el micrófono inalámbrico con cancelación de ruido DJI Mic Mini con el mismo listado de cien instrucciones utilizado en la primera prueba. Ya con el formato de texto corregido es que se obtuvieron resultados más precisos de cada modelo, sin embargo, se notaron patrones de errores según algunas palabras, combinaciones de letras con números, etcétera como se muestra en la tabla 3. Por lo que se optó por reemplazar algunas palabras del listado de instrucciones por sinónimos para obtener mejores resultados.

Tabla 3. Patrones de errores prueba 2

Modelo STT	Palabras	Números	Espaciado	% Aciertos
Azure	Confunde pocas veces “derecha” con “derecho”	Sin errores	Une números	62%
Google	Omite palabras	Omite números	Junta números con letras	33%
Vosk	Confunde “Mueve” con “Nueve”	Sin errores	Une números	39%
Vosk small	Confunde “Mueve” con “Nueve”	Aún escribe “uno” y “cuatro”	Une números	14%
Whisper	El multilinguaje detecta palabras en otros idiomas	Sin errores al entender presencia de números	Une palabras o inventa otras	28%

Fuente: Elaboración propia

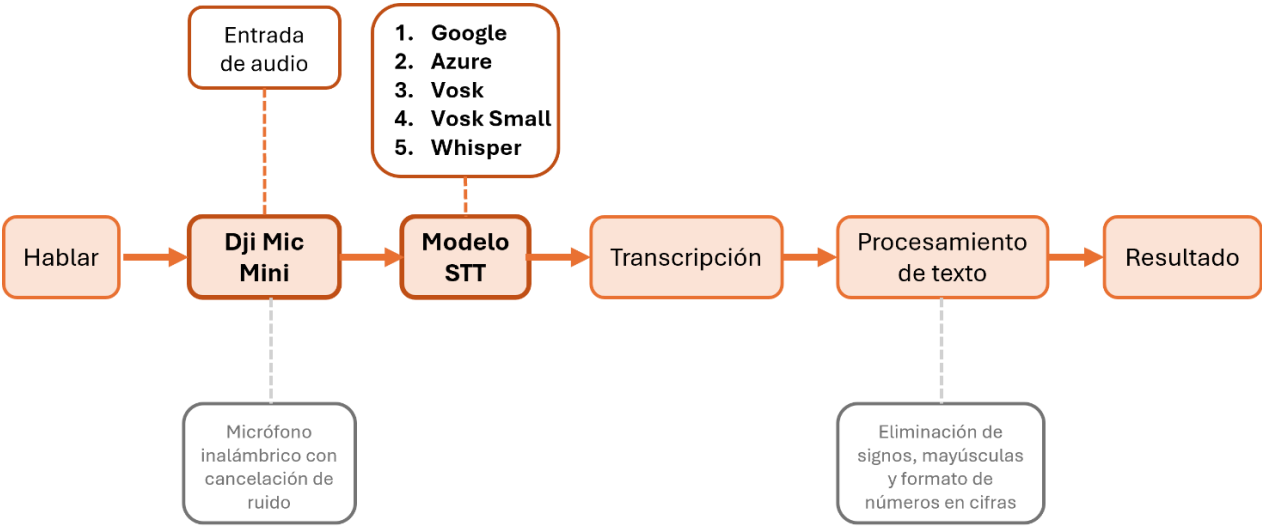


Fig. 23. Flujo de pruebas STT. Elaboración Propia

Con los resultados obtenidos en esta prueba es que se realizan algunos cambios en los formatos de un par de comandos; en este caso, se opta por modificar la palabra “Mueve” por “Mover” y se modifica el comando de rotación de ejes, dándole un “nombre” a cada eje para acompañarlo de un solo número. De esta forma, se erradican errores asociados al reconocimiento de palabras en un formato y también se establece un formato de comando de ejes más sencillo de transcribir para cada modelo.

Tabla 4. Modificaciones de comandos

Listado Original	Listado Final
“Mueve” + coordenada + número	“Mover” + coordenada + número
“Mueve Z 500”	“Mover Z 500”
Eje + n° eje + número	“Nombre eje” + número
“Eje 6 120”	“Sexto eje 120”

Fuente: Elaboración propia

3. PRUEBA FINAL

Para cerrar el proceso de evaluación, se realizó una tercera ronda de prueba bajo las mismas condiciones e implementos utilizados previamente, incorporando una modificación clave: la actualización parcial del listado de instrucciones.

Esta modificación buscó eliminar combinaciones específicas que habían generado patrones reiterativos de error en las pruebas anteriores, lo que permitió obtener resultados más favorables en todos los modelos, tal como se muestra en la Tabla 5. Si bien aún persisten ciertos errores en combinaciones particulares de comandos o sílabas, los resultados evidencian una mejora sustancial en la tasa de reconocimiento.

Dentro de los modelos que no mostraron un buen rendimiento está Whisper de OpenAI, el cual funciona a través del análisis de una grabación previa, lo que resultaba en un proceso más lento al tener que crear un archivo temporal de audio por cada instrucción hablada. Además, tiene la función de multilinguaje, donde en muchas ocasiones provocaba que ciertas pronunciaciones o palabras las interpretara en otro idioma y no coincidiera con el listado, a pesar de que el modelo recibe contexto para saber qué tipo de palabras o idioma analizar, este fue el único que no logró un mejor porcentaje a causa de esto.

El modelo con el mayor desempeño fue *Microsoft Azure*, alcanzando un porcentaje corregido de aciertos del 96%, con errores de transcripción mínimos y sin patrones de fallo evidentes. Gracias a su capacidad para interpretar términos técnicos en español y diferenciar correctamente números, letras y coordenadas, se seleccionó como el motor principal de reconocimiento para la implementación del sistema.

2.1.3 Resultados Parciales

Con las pruebas ejecutadas anteriormente, se selecciona *Microsoft Azure* como motor STT principal por su capacidad de reconocer comandos técnicos en español con alta precisión y sin errores sistemáticos. Este modelo logra interpretar formatos mixtos de palabras, letras y números, especialmente en contexto de lenguaje técnico con coordenadas o instrucciones vectoriales.

Tabla 5. Resultados porcentajes de aciertos

Modelo STT	Prueba 1	Prueba 2	Prueba 3
Azure	62%	90%	96%
Google	33%	61%	69%
Vosk	39%	89%	91%
Vosk small	14%	77%	65%
Whisper	28%	51%	58%

Fuente: Elaboración propia

Acorde a los resultados obtenidos, se ve una mejora secuencial luego de cada prueba gracias a las correcciones. Esto asegura una mejor ejecución al momento de programar las trayectorias de movimiento y además se tiene la certeza de que hay dos modelos con sobre el 90% de precisión, lo cual asegura más de una opción viable en la integración de STT para la transcripción de instrucciones verbales.

A pesar de que existen maneras más complejas de catalogar instrucciones y corregir transcripciones, se opta por hacer la parte de programación lo más acotada posible, ya que, en este caso se tiene una serie de movimientos limitados. Al ser un prototipo, se crea un entorno mucho más seguro limitando a solo un formato de comando sin

involucrar correcciones a través de inteligencia artificial o modelos más complejos del procesamiento del lenguaje, ya que, estos pueden actuar de forma no deseada o tener variaciones al tener todos estos subprocesos.

Según la tabla 6, el sistema responde de forma correcta mayoritariamente a las instrucciones por voz, sin múltiples subprocesos de la transcripción original. Los comandos con menos del 70% de aciertos se deben comúnmente a problemas de espaciado, de escritura de letras o números como se especifica en la Tabla 3.

Tabla 6. Precisión de modelos según tipo de comando

Tipo de comando	Prueba 2	Prueba 3
Dirección + Número	84%	86%
Coordenada (X/Y/Z) + Número	61%	65%
Mover + Coordenada (X/Y/Z) + Número	37%	67%
Gira + A/B/C + Número	45%	41%
Nombre de eje + Número	49%	70%

Fuente: Elaboración propia

Aún existen dificultades para comandos rotacionales de ejes que pueden requerir más programación en el procesamiento del lenguaje, por lo que para efectos de esta investigación no serán utilizados en el entorno de prueba. Los resultados de la Tabla 6 indican que el formato de “Dirección + Número” presenta mayor porcentaje de aciertos y no se detectaron patrones de errores específicos respecto a este formato, por lo que será la forma principal de ejecutar movimientos direccionales utilizando el formato más intuitivo.

2.2 PROTOTIPO DE CONTROL POR VOZ

2.2.1 Integración STT en Rhinoceros y Grasshopper

La lógica que conecta el reconocimiento de voz con la ejecución robótica se implementa en componentes *GH-Python*. Este componente nativo de *Grasshopper* permite código en *Python* 3. En este caso se utiliza la versión 3.9.11 que viene integrado en *Grasshopper* de *Rhino* 8.

Para poder acceder al motor STT desde *Grasshopper* es que se debe instalar la librería de *Azure* a *Python* dentro de *Grasshopper*, de esta forma, se tiene acceso para conectarse al servidor del STT seleccionado. Para acceder a la retroalimentación auditiva es que se instala la librería *PyTTSx3* (*Python Text to Speech*) que es capaz de leer y emitir voz según el texto que reciba. Inicialmente, se configura el sistema para que reciba comandos direccionales, en este caso se utiliza el formato de “dirección + número” en donde según la posición del usuario (Fig.B) se define el valor vectorial para cada dirección.

VECTORES SEGÚN DIRECCION

- ARRIBA: (0, 0, 1) Z+
- ABAJO: (0, 0, -1) Z-
- IZQUIERDA (0, 1, 0) Y+
- DERECHA (0, -1, 0) Y-
- ADELANTE (1, 0, 0) X+
- ATRÁS (-1, 0, 0) X-

Fig. B. Referencia direccional en Laboratorio. Elaboración Propia



2.2.2 Simulación Offline

Para poder realizar la simulación *offline*, se integra el proceso de STT dentro de *Python* con *KUKA|prc*. El vector que resulta del código *Python* funciona como vector traslacional del *Tool Center Point* (TCP) del robot, donde se utiliza un comando *Move* que mueve el TCP actual del robot que se define manualmente con el vector resultado del STT, obteniendo como resultado un plano que determina la nueva posición del TCP que es conectado a un componente *LINEAR Movement* de *KUKA|prc* que es capaz de simular en *Rhino* cuál sería el movimiento por ejecutar. (Fig. 24)

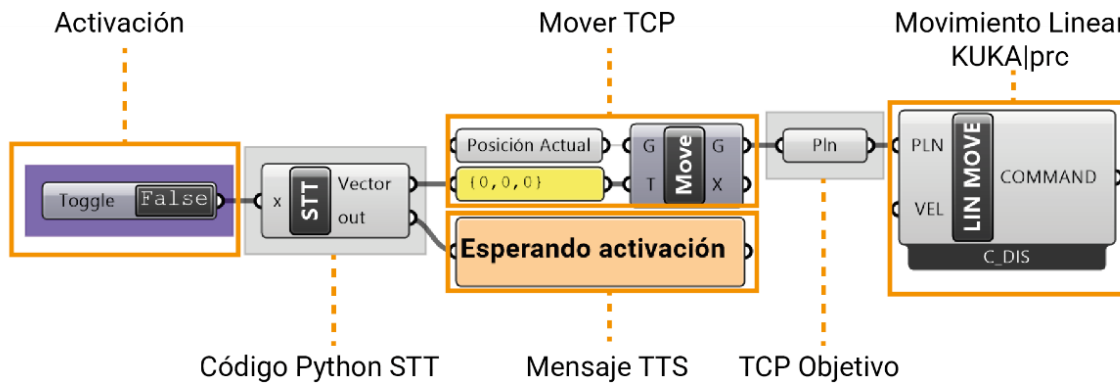


Fig. 24. Algoritmo preliminar STT offline para movimientos direccionales. Elaboración Propia.

En este flujo, la simulación *offline* cumple la función de validar de manera anticipada cada comando hablado antes de que sea enviado al robot físico, lo que permite verificar su seguridad y viabilidad dentro del entorno geométrico de Rhinoceros. Una vez calculado el vector de movimiento mediante STT, este se incorpora al plano actual del TCP para generar un nuevo plano objetivo, el cual es procesado por los componentes de *KUKA|prc* para reconstruir la trayectoria completa del robot considerando tanto la orientación de la herramienta como la base activa. Este proceso no solo visualiza la trayectoria en términos de desplazamiento cartesiano, sino que también actualiza la cinemática completa del robot dentro de la simulación, mostrando el movimiento de cada eje, los cambios de orientación y la posición relativa del robot respecto de su entorno.

El sistema entrega una retroalimentación auditiva del movimiento y la distancia en milímetros que se moverá el TCP del punto inicial al final. También, gracias al análisis que ofrece *KUKA/prc*, se incorpora una advertencia en caso de que se detecte colisión o si el punto es inalcanzable (Fig. 25)

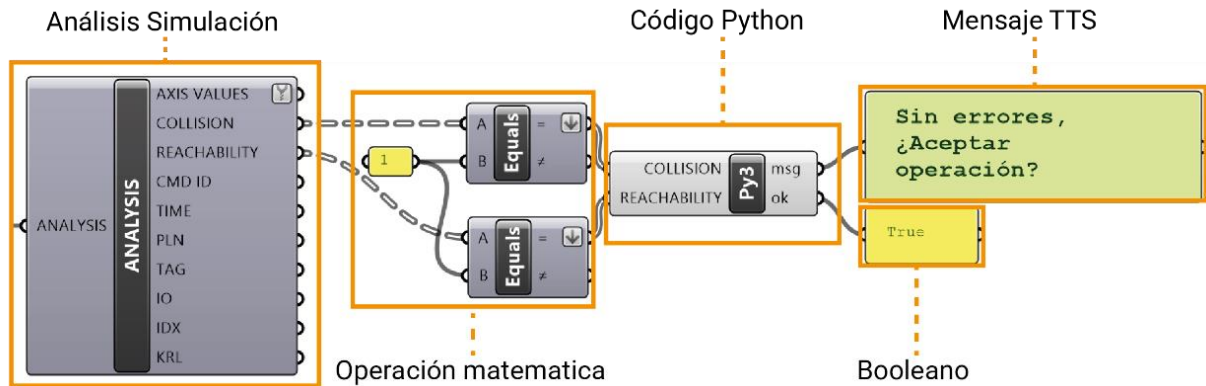


Fig. 25. Algoritmo Advertencia de colisión offline. Elaboración Propia.

El análisis de la simulación funciona dando valores de 0 o 1, donde si existe algún valor 1 en la salida de Colisión (*Collision*) o Alcance (*Reachability*) significa que durante el movimiento del robot existe un movimiento o articulación que no se puede ejecutar.

Para sistematizar aquello, se utilizan componentes "Equals" (igual a) donde se analizan ambas salidas, de colisión y alcance, se evalúa si existe algún valor 1 en cada lista y se ingresa a un código *Python*. El código *Python* analiza cada lista y otorga un mensaje según corresponda.

- Si existe valor 1 en Collision → TTS: "Colisión detectada"
- Si existe valor 1 en Reachability → TTS: "El robot no puede alcanzar el punto"
- Si no existe ningún valor 1 → TTS: "Sin errores, ¿Aceptar Operación?"
- Si existen valores 1 en ambas listas → TTS: "Colisión y Punto inalcanzable"

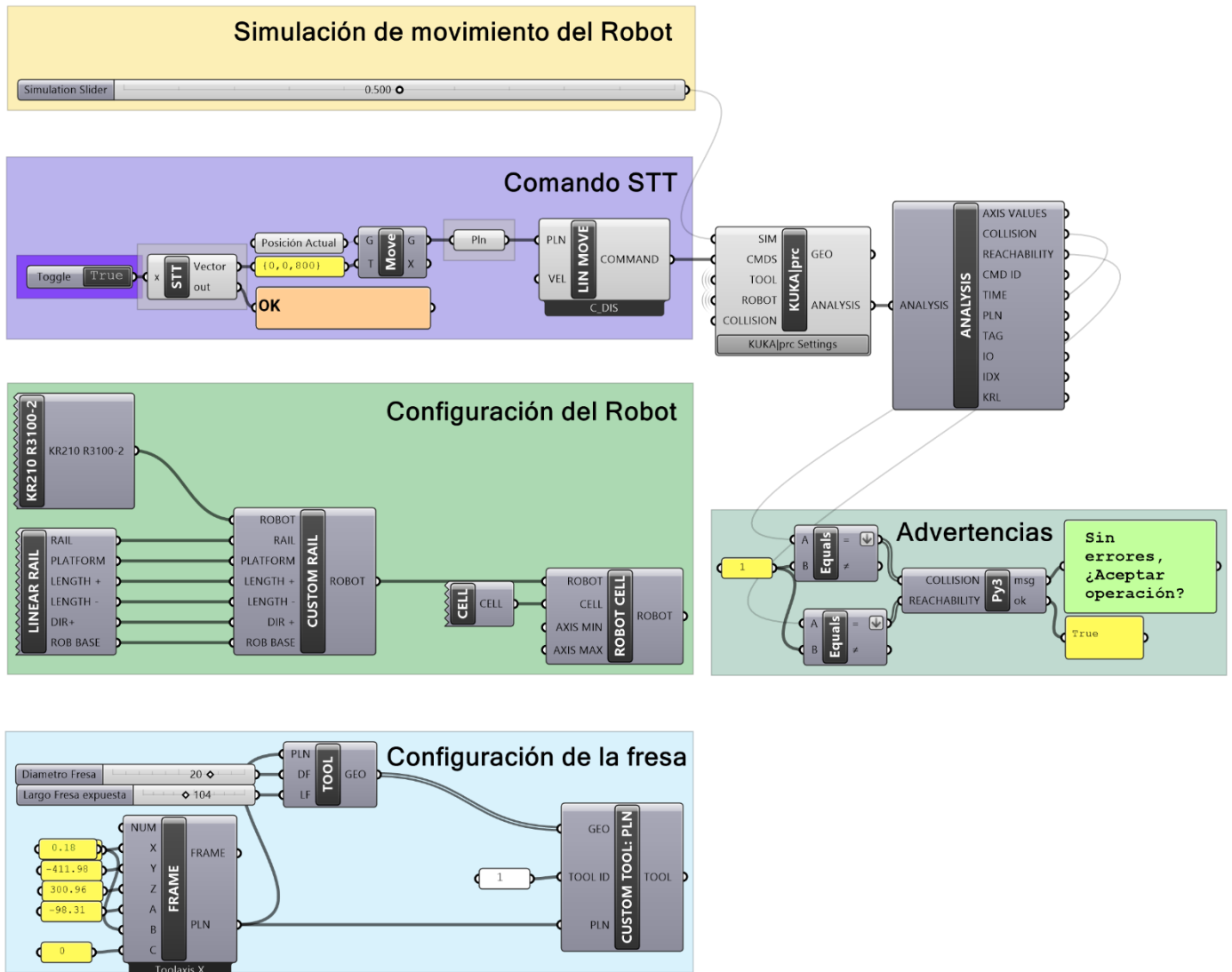


Fig. 26. Algoritmo completo de simulación offline. Elaboración Propia



Fig. 27. Simulación offline de movimiento direccional Rhino 8. Elaboración propia

Otra característica del sistema es enviar al robot a posiciones predeterminadas. Existen ubicaciones y poses del robot que resultan útiles a la hora de mecanizar, medir o guardar la máquina, en este caso, por el uso que se le da al robot en el Laboratorio de construcción y manufactura robotizada de la UTFSM es que se determinan 4 poses para simular de forma *offline* con el sistema de control por voz. Estas 4 posiciones son: Posición de trabajo (mecanizado); Posición de Inicio (medición de base) y Posición de guardado (almacenamiento) y Reposo (por defecto) (Fig. 30).

Para determinar estas posiciones, se requiere de los valores en grados de cada eje del robot; en este caso son 6 ejes y un eje lineal externo, para efectos de este trabajo no se considera el eje lineal externo. Para obtener los valores, de forma experimental, se ubica el robot físico en la pose deseada con un componente *AXIS MOVE* de *KUKA/prc* dentro de *Grasshopper* observando la simulación para que coincida con la posición del robot deseada. Luego, los valores de cada eje se ingresan a un Código *Python* (Fig. 28) que actúa de diccionario, donde determina que una frase clave significa ciertos valores de eje (Fig. 29).

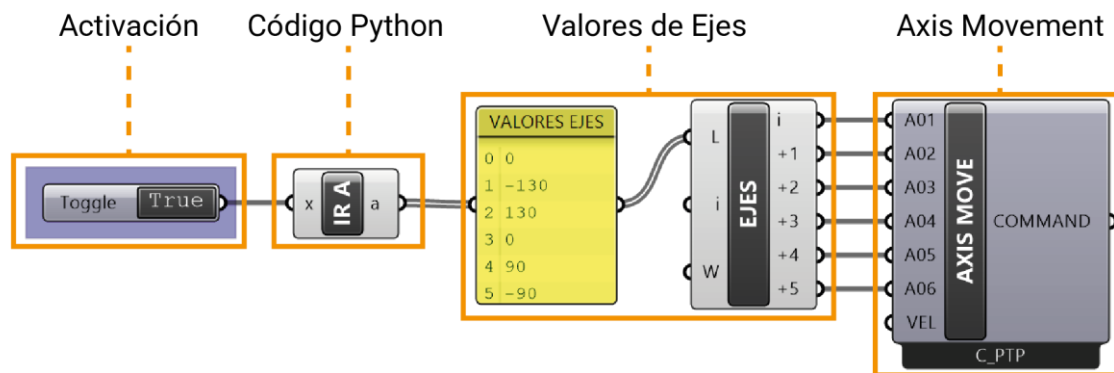
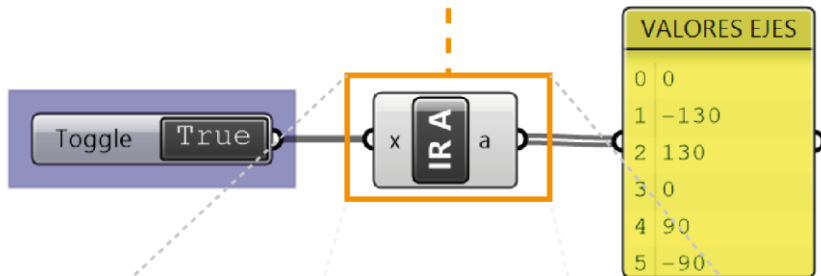


Fig. 28. Algoritmo Movimiento Axial. Elaboración Propia

Código Python



```

_refresh()

raw, clean = reconocer_una_vez()

if raw is None:
    texto_resultado = ""
    estado_resultado = "No se entendió el audio."
    valores_ejes = [0, -90, 90, 0, 0, 0] # reposo
else:
    texto_resultado = raw
    estado_resultado = "OK"

# =====
# DICCIONARIO DE POSICIONES
# =====
diccionario_posiciones = {
    "posición guardado": [0, -130, 130, 0, 90, -90],
    "posición de trabajo 1": [90, -130, 130, 80, 0, 0],
    "posición de inicio": [160, -115, 100, -83, 65, -205],
}

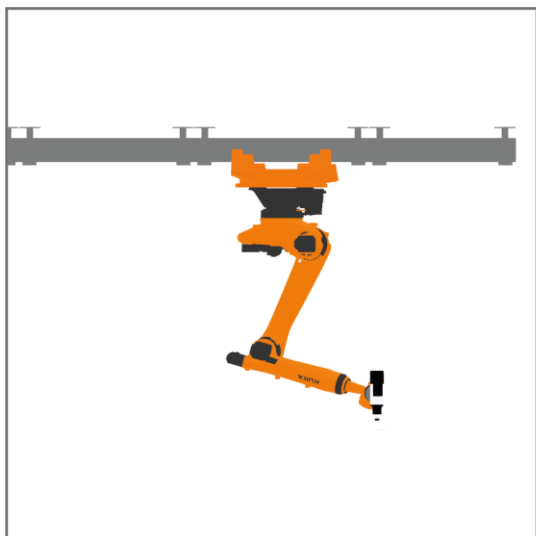
valores_ejes = [0, -90, 90, 0, 0, 0] # default reposo
for clave, valores in diccionario_posiciones.items():
    if clave in clean:
        valores_ejes = valores
        break

escuchando = False
resultado_listo = True
winsound.Beep(800, 200) # beep fin
_refresh()
    
```

Valores de ejes según posición

Pose Reposo por defecto

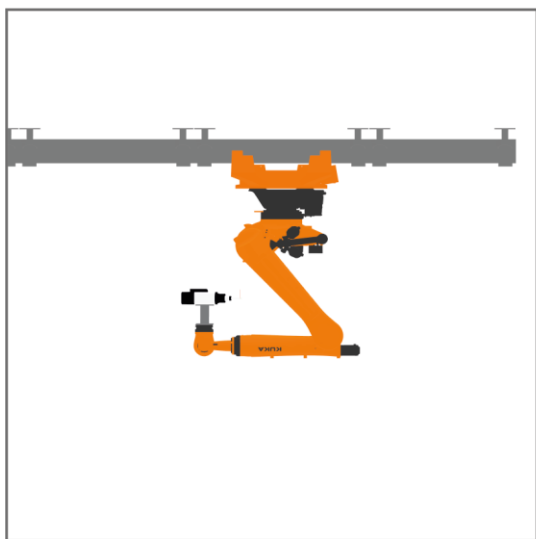
Fig. 29. Código Python para pose por ejes. Elaboración Propia



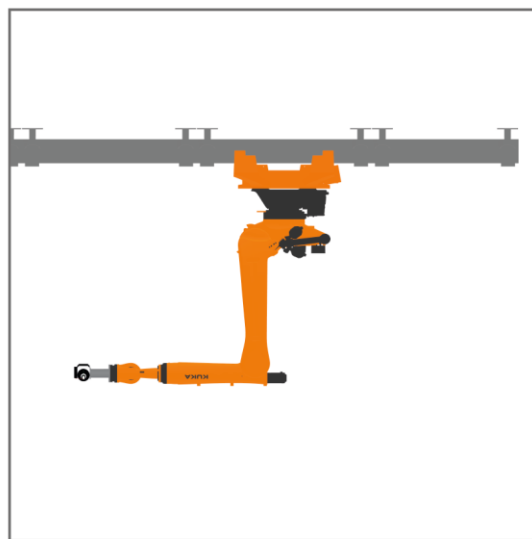
Posición de Trabajo



Posición de Inicio



Posición de Guardado



Posición de Reposo

Fig. 30. Poses predefinidas. Elaboración Propia.

2.2.3 Integración de control online con RoboDK

Para ejecutar los movimientos en el robot físico, se requiere un tipo de conexión con la máquina, en este caso, *Grasshopper* no tiene la capacidad directa de conectarse con el plugin utilizado, por lo que se opta por utilizar el software de *RoboDK*.

RoboDK (*Robot Development Kit*) es un software que permite enviar programas hacia un robot físico mediante una conexión red. Este programa admite Python y tiene un API que puede ser integrada dentro de *Grasshopper*, por lo que queda un flujo donde *Grasshopper* integra el STT, se simula en *Rhinoceros* a través de *KUKA|prc*, se envía el programa a *RoboDK* mediante *Python* y se ejecuta en físico vía red (Fig. 31).



Fig. 31. Selección de software. Elaboración Propia

Gracias a este flujo es que será posible obtener datos del robot físico, como su TCP y valores de ejes, además de poder ejecutar todo lo que se ha simulado fuera de línea dentro de *Grasshopper* anteriormente.

2.2.4 Configuración RoboDK

La ejecución *online* requiere una estación previamente configurada en *RoboDK* que actúe como intermediario entre *Grasshopper* y el robot físico. Para ello, se creó una estación base que replica el entorno de trabajo, incorporando el modelo cinemático del robot KUKA KR 210 R3100-2 C y los elementos de referencia necesarios para la operación. Dado que el robot KUKA KR 210 R3100-2 C no se encuentra disponible en la librería nativa de *RoboDK*, fue necesario modelarlo íntegramente para habilitar su simulación y control online. El proceso se inició descargando el modelo 3D oficial del robot en formato *STEP* desde la plataforma *my.KUKA*, junto con su ficha técnica, la cual contiene los rangos articulares, *offsets*, dimensiones del espacio de trabajo y parámetros cinemáticos requeridos para construir el mecanismo (Anexo A).

El archivo *STEP* fue importado en una estación vacía de *RoboDK* y se preparó para su articulación. Posteriormente, se separó el modelo 3D en las siete partes que componen su estructura: base fija y los seis eslabones correspondientes a los ejes del robot (Fig. 32). Esto se realizó mediante la herramienta *Split Object*, seguida por la agrupación manual de las piezas asociadas a cada eje utilizando *Merge*. El resultado fue un conjunto de geometrías limpias y organizadas en el árbol de estación, cada una representando un eslabón del mecanismo real.

Con este despiece finalizado, se procedió a definir el robot como mecanismo articulado utilizando la herramienta *Model Mechanism or Robot*. En este módulo se ingresaron los valores de la ficha técnica: límites articulares, *offsets* angulares ($\Delta\theta$), sentidos de giro, posiciones “*Home*” y ajustes de alineación (“*Build Joints*”) para hacer coincidir la cinemática del modelo con la del robot físico. Al actualizar el mecanismo, *RoboDK* generó un robot completamente articulado, con la movilidad correcta en cada junta y con una correspondencia directa entre su pose en la simulación y la del robot real (Fig. 33).

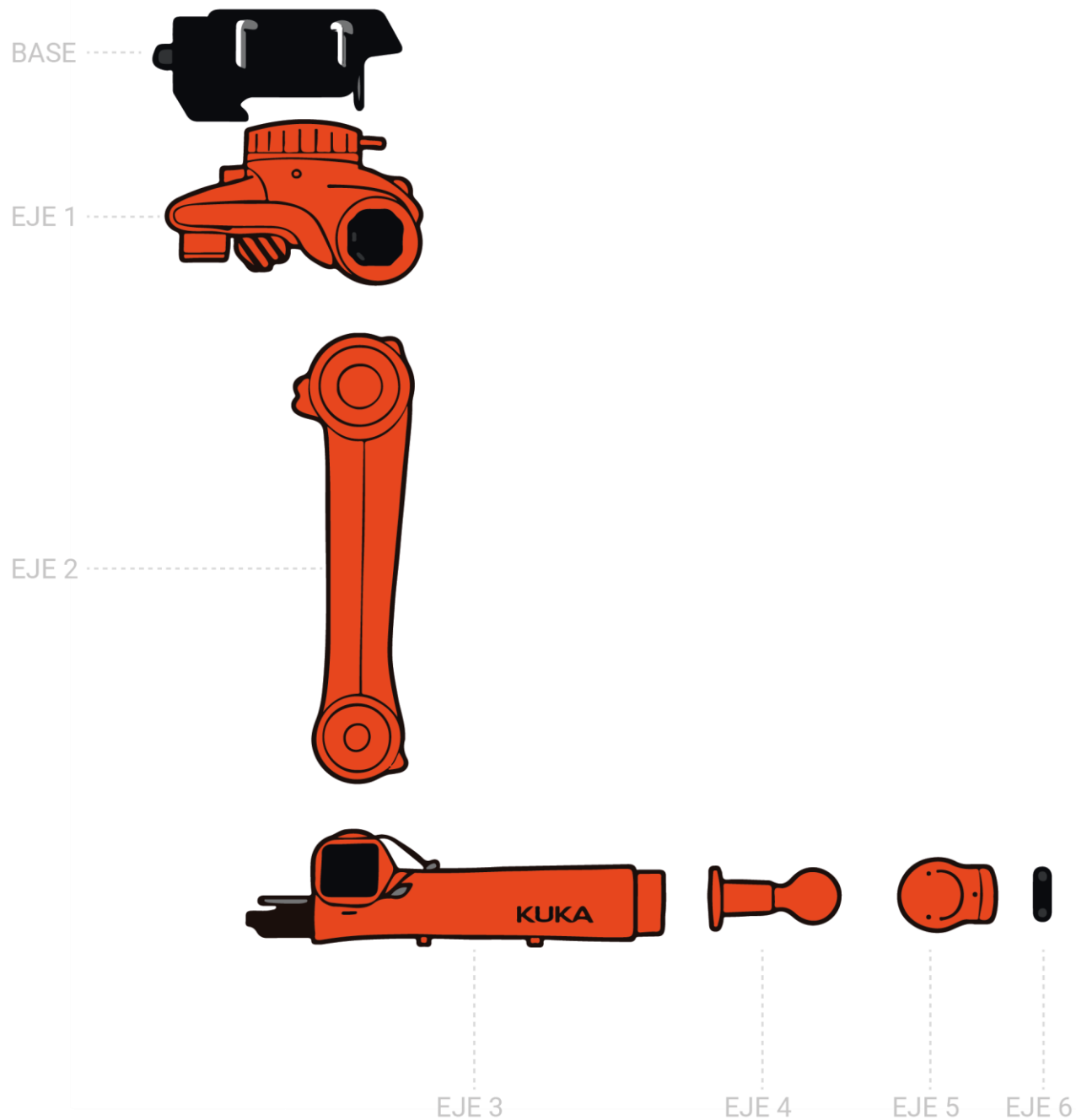


Fig. 32. Despiece por ejes. Elaboración Propia

Además, se integró el conjunto riel-carro del eje lineal externo KL4000, ya que el robot del laboratorio opera montado sobre un eje de traslación. Dado que tampoco existía en la librería oficial, se generó un modelo tridimensional del riel y carro a partir del archivo CAD descargado desde KUKA Xpert. Este fue previamente organizado y posicionado en *Rhino 8* utilizando puntos de referencia específicos para asegurar su coincidencia exacta con el mecanismo robótico al importarlo en *RoboDK*. El mecanismo lineal se definió como un eje adicional con límites entre -8000 mm y 2000 mm, y se vinculó al robot mediante la función *Synchronize External Axes*, permitiendo así un movimiento coordinado entre el carro y el manipulador (Fig.34)

Finalmente, se importó la estructura de soporte del laboratorio para completar la estación digital (Fig. 35). Con ello se obtuvo una estación en *RoboDK* que replica la configuración física del laboratorio, que permite enviar movimientos, simular trayectorias y leer el estado real del robot mediante su API de comunicación.

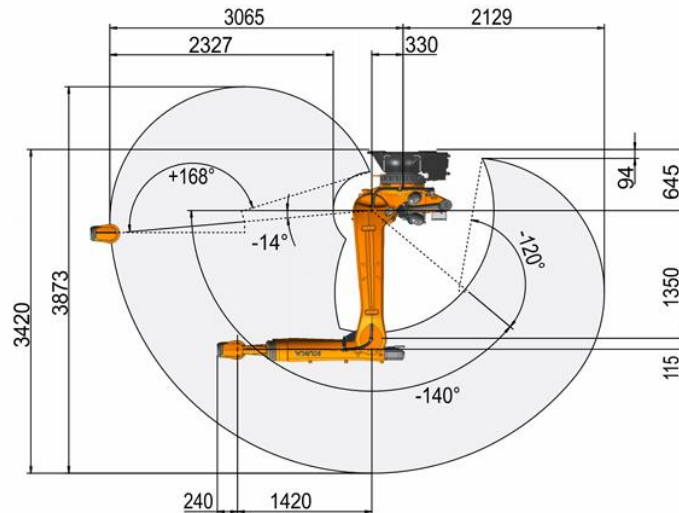


Fig. 33. Medidas robot. Elaboración Propia en base a Ficha técnica KUKA KR210 R3100-2 C

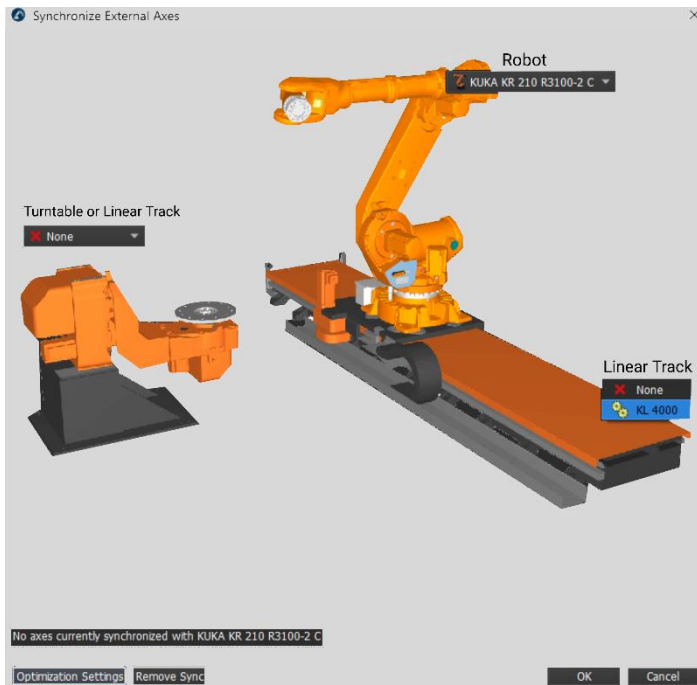


Fig. 34. Sincronización de eje externo RoboDK. Elaboración Propia

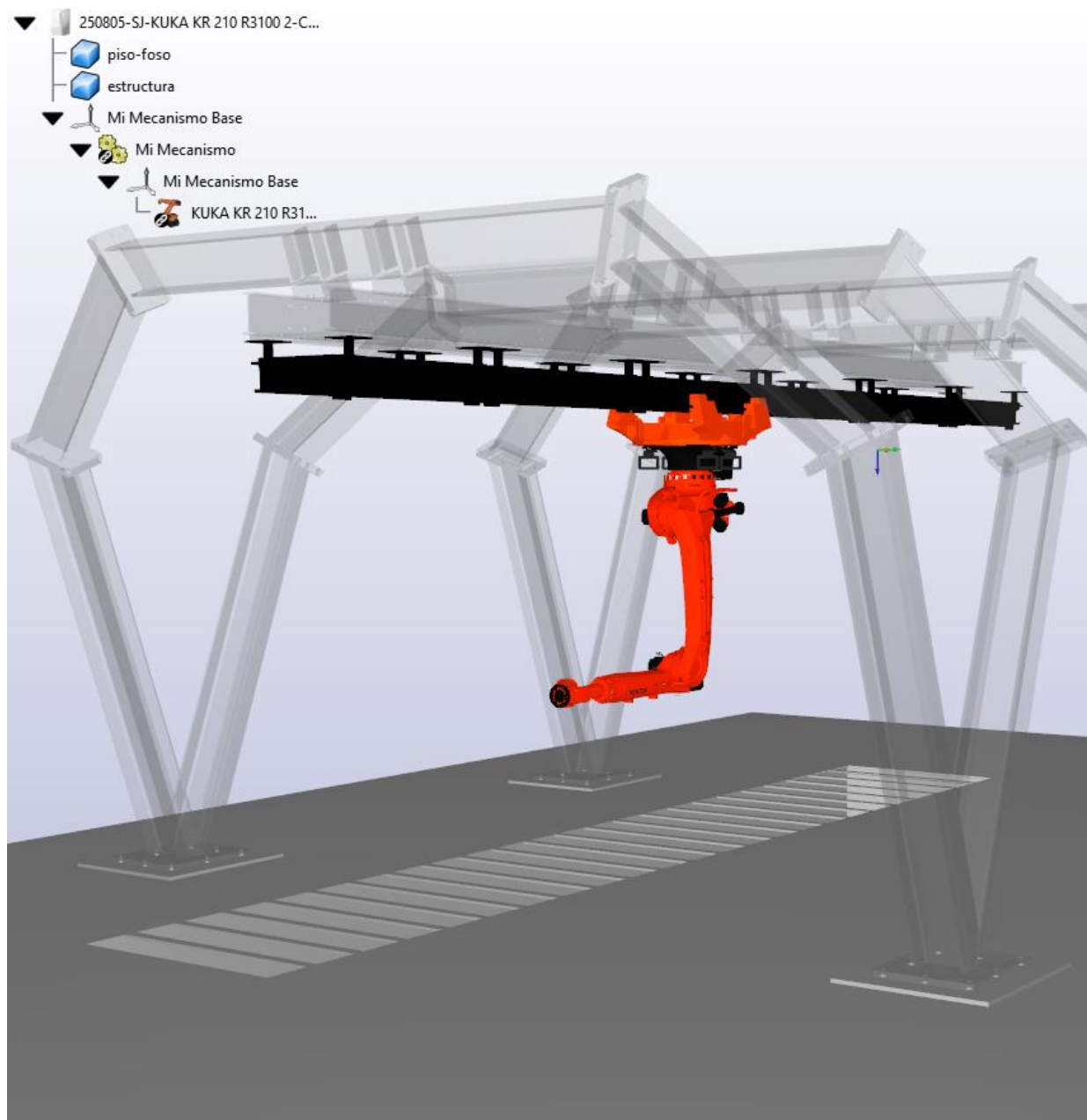
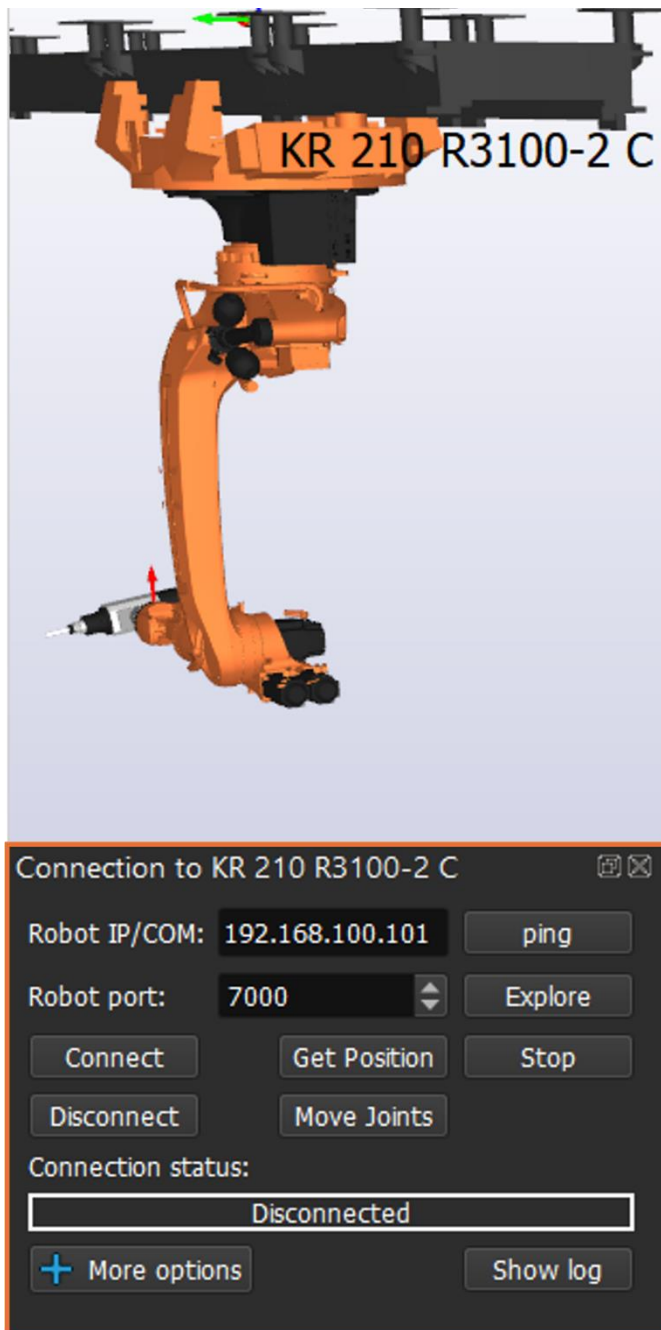


Fig. 35. Estación Laboratorio de Construcción y Manufactura Robotizada en RoboDK. Elaboración Propia

Una vez configurado el modelo cinemático en *RoboDK*, la ejecución online requiere habilitar una comunicación directa entre el computador *host* y el controlador KR C4 (*KUKA Robot Controller 4*). En términos simples, el KR C4 es el gabinete que contiene la electrónica y el sistema operativo del robot (*Windows* embebido con el *software* de control *KUKA System Software KSS*), mientras que el *SmartPAD* es el *teach pendant*, es decir, el dispositivo portátil con pantalla táctil desde el cual el operador selecciona programas cambia modos de operación y mueve el robot manualmente.

En este trabajo se emplea el *driver kukabridge* de *RoboDK*, un programa que se instala dentro del KR C4 y que actúa como servidor, recibiendo mensajes por la red y exponiendo una “puerta de entrada” para leer y escribir variables internas del controlador. La comunicación se realiza sobre el protocolo TCP/IP, es decir, el mismo conjunto de reglas que utilizan los computadores para comunicarse en Internet, configurando el controlador y el computador del laboratorio con direcciones IP compatibles dentro de la misma subred y habilitando un puerto de comunicación para que el *C3 Bridge* pueda aceptar conexiones desde *RoboDK*.

Una vez instalado el *C3 Bridge* en el KR C4, desde el *SmartPAD* se ejecuta un programa KRL generado por *RoboDK*, que se corre en bucle continuo (*loop*): esto significa que el programa no termina, sino que permanece “escuchando” cambios en un conjunto de variables globales especiales, las cuales son actualizadas por el *driver kukabridge* a través de la red. Así, cuando desde el computador se ordena un movimiento, el *driver* escribe en esas variables la instrucción correspondiente; el programa KRL *RoboDKsync* la interpreta y ejecuta los comandos de movimiento nativos de KUKA, respetando los modos de operación y las seguridades del controlador. De este modo, la arquitectura completa puede entenderse como tres capas encadenadas: por un lado, el nivel de alto nivel donde *Grasshopper* y *RoboDK* generan trayectorias a partir de comandos por voz; en el centro, el *C3 Bridge* más el *driver kukabridge*, que transforman esas órdenes en mensajes de red y actualizaciones de variables internas; y en la base, el programa KRL en *loop* que corre en el KR C4, encargado de transformar esas variables en movimientos físicos del robot mientras el *SmartPAD* se utiliza únicamente para poner el sistema en el modo adecuado y lanzar el programa de sincronización.



Estación RoboDK

Conexión a robot
físico KUKA KR
210 R3100-2 C

Fig. 36. Conexión por red a robot físico desde RoboDK. Elaboración Propia

2.2.5 Integración Grasshopper

La conexión entre *Grasshopper* y *RoboDK* se hace a través de *Python*. En el entorno de *Grasshopper* se instaló la librería de *RoboDK*, que permite enlazar ambos programas para enviar o recibir información.

Según las simulaciones *offline*, se tienen movimientos direccionales y movimientos de ejes, que se traducen en Movimientos Lineales (*MoveL*) y Movimientos Axiales (*MoveJ*) los cuales se deben ingresar a *RoboDK* con esa misma tipología.

Todos los movimientos se realizan desde el TCP actual del robot, el cual hay que extraer del robot físico para generar el movimiento real. Para ello, primero se crea un código de conexión al robot desde *Grasshopper* a *RoboDK*, donde se ingresa el nombre del modelo del robot para que enlace con el espacio de trabajo de *RoboDK* que contiene el modelo. El *script* gestiona la conexión a *RoboDK*: cuando se activa la conexión crea o reutiliza una conexión persistente, busca el robot por nombre, guarda el objeto robot para que otros componentes puedan usarlo y reporta el estado a la interfaz. Si ocurre un error elimina cualquier estado parcial. Cuando se desactiva la conexión, se cierra y limpia la conexión compartida. Esto permite compartir una sola conexión a *RoboDK* entre múltiples componentes de *Grasshopper/Rhino* sin reinstanciarla constantemente.

Para esta conexión se utiliza el modelo creado con el modelo del robot correspondiente en *RoboDK*, donde se ingresaron los datos de red que permiten hacer la conexión con el robot físico y permitir la transferencia de datos.

Se activa el monitor del TCP que actualiza la posición de este por cada movimiento realizado por el robot físico, de modo que se pueda ver la simulación del movimiento que hará el robot a través de su posición real en el espacio de trabajo (Fig.37). De esta forma, se asegura la correcta conexión al robot real y se puede previsualizar el movimiento exacto a realizar desde *Rhino* para luego hacer su ejecución real. El componente mantiene un estado persistente (*latch*) con la última pose/juntas publicadas (dónde está y cómo están sus ejes). Cuando está activo, corre un proceso en segundo plano que decide si revisar con más frecuencia (cuando el robot se está moviendo) o con menos frecuencia (cuando está en reposo). En cada revisión

compara las lecturas nuevas con las anteriores; si detecta movimiento empieza a contar ciclos y, cuando ese conteo llega a cierto valor, asume que el movimiento terminó. En ese momento guarda los valores finales de posición y ángulos y los entrega; esos valores se mantienen iguales hasta que vuelva a detectarse otro movimiento completado.

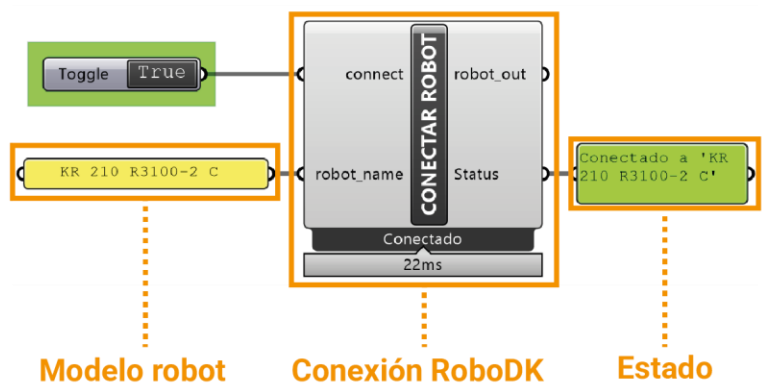
Para la ejecución de movimientos, se configuraron tres componentes específicos en *Grasshopper* que actúan como interfaz entre el flujo de comandos por voz, *RoboDK* y el robot físico: uno para movimientos axiales (*MoveJ*), otro para movimientos lineales (*MoveL*) y un tercero para trayectorias de mecanizado basadas en listas de planos. En los tres casos, el principio de funcionamiento es común: cada componente recibe una señal booleana *run_move* que opera como disparador del movimiento; mientras *run_move* está en falso, el componente permanece inactivo, actualiza únicamente estados de conexión y se mantiene a la espera del siguiente comando, y cuando *run_move* pasa a verdadero interpreta los datos de entrada, construye la instrucción correspondiente y la envía a *RoboDK* mediante la API de *Python*. En paralelo, todos comparten la misma lógica de modos: en modo simulación la instrucción se ejecuta únicamente en la estación virtual de *RoboDK*, sin mover el robot físico, mientras que en modo online se utiliza la IP configurada del controlador KR C4 para enviar los movimientos a través del *driver*, de modo que sean ejecutados en el robot real. Antes de ejecutar cualquier desplazamiento, los componentes intentan unificar el contexto del robot ajustando la herramienta activa (*tool*) y la base o sistema de referencia (*base frame*) cuando estos parámetros están disponibles, asegurando que todos los movimientos se calculen en el mismo sistema de coordenadas y con el mismo TCP. Además, cada componente mantiene un canal de mensajes de estado en texto (por ejemplo, “Conectado”, “Simulación”, “Movimiento ejecutado”, “Error en conexión”), que se devuelve a *Grasshopper* para informar al usuario sobre el resultado de la operación. Las diferencias entre los tres tipos de movimiento se encuentran principalmente en el tipo de datos que reciben y en cómo construyen la trayectoria que se envía a *RoboDK*.

- **Conexión MoveJ:** Recibe una señal booleana *run_move* que opera como disparador del movimiento; mientras *run_move* está en falso, el componente permanece inactivo y cuando *run_move* pasa a verdadero interpreta los datos recibidos y envía la instrucción correspondiente a *RoboDK* mediante la API de *Python*. Este componente trabaja en el espacio articular del robot y genera movimientos de tipo *MoveJ*; para ello recibe directamente seis ángulos en grados asociados a cada articulación (J1...J6), junto con un parámetro *speed_percent* que define la velocidad como porcentaje de la velocidad máxima de giro permitida. Funciona tanto en modo simulación como en modo online, y en ambos casos utiliza el robot entregado como *robot_in* desde el componente de conexión previa, manteniendo la misma herramienta y base activas, sin intervenir señales digitales ni tiempos de espera adicionales.
- **Conexión MoveL:** recibe una señal booleana *run_move* que opera como disparador del movimiento; mientras *run_move* está en falso, el componente permanece inactivo, y cuando pasa a verdadero interpreta los datos de entrada, construye la instrucción correspondiente y la envía a *RoboDK* mediante la API de *Python*. Este componente opera en el espacio cartesiano y genera movimientos *MoveL*; para ello recibe un único destino, que puede ser un plano de *Rhino* (que contiene simultáneamente posición y orientación del TCP) o un vector de coordenadas tipo [X, Y, Z, A, B, C] en el sistema KUKA, donde X, Y, Z representan la posición y A, B, C los ángulos de Euler que representan la orientación. La velocidad se define como velocidad lineal en mm/s y se incorpora un parámetro de redondeo (*rounding o blending*) que permite suavizar las trayectorias evitando paradas bruscas entre segmentos consecutivos cuando corresponde. Al igual que en el caso anterior, trabaja en modo simulación u online utilizando el robot entregado como *robot_in*, respetando la herramienta y base activas fijadas por el componente de conexión.
- **Conexión programa de Mecanizado:** también se activa mediante la señal booleana *run_move*, de forma que el componente permanece inactivo mientras el valor es falso y, al cambiar a verdadero, interpreta la información recibida y

ejecuta la trayectoria completa. A diferencia de los anteriores, no recibe un único destino sino una lista de planos que describen posiciones consecutivas de trabajo, a partir de la cual construye internamente una secuencia de movimientos lineales *MoveL* que siguen la trayectoria de mecanizado (por ejemplo, una matriz de perforaciones). La velocidad se define igualmente en mm/s, pero el parámetro de redondeo se fija en cero, de modo que el robot se detiene de forma precisa en cada punto, condición más adecuada para tareas de taladrado o corte. Este componente incorpora además control de una señal digital de salida (por ejemplo, DO2) para activar y desactivar la herramienta antes y después de la trayectoria, e incluye tiempos de espera (*WAIT*) al inicio y al final de la secuencia como medida de seguridad. A nivel de conexión, en vez de depender de *robot_in*, inicializa su propia instancia de *Robolink*, busca el robot por nombre dentro de la estación de *RoboDK* y establece de forma autónoma la conexión en modo simulación u *online*, manteniendo la coherencia con la herramienta y base definidas para el proceso de mecanizado.

De esta manera, la combinación de la estación modelada en *RoboDK*, la conexión online con el controlador KR C4 y los tres componentes de movimiento desarrollados en *Grasshopper* permite establecer una comunicación completa entre el flujo de trabajo paramétrico y el robot físico. *Grasshopper* deja de ser únicamente un entorno de simulación geométrica para convertirse en la interfaz principal de control: los comandos por voz se traducen a instrucciones numéricas, estas se estructuran en movimientos axiales, lineales o trayectorias de mecanizado según corresponda, y finalmente se envían a *RoboDK*, que los transmite al controlador del robot para su ejecución en tiempo real. De este modo, se garantiza que cada movimiento ejecutado por el robot KUKA corresponda a una instrucción previamente validada en la simulación, manteniendo coherencia entre el modelo digital y el comportamiento físico, y habilitando un flujo integrado de programación visual, control por voz y fabricación robotizada en madera.

CONEXIÓN ROBOT



MONITOR TCP

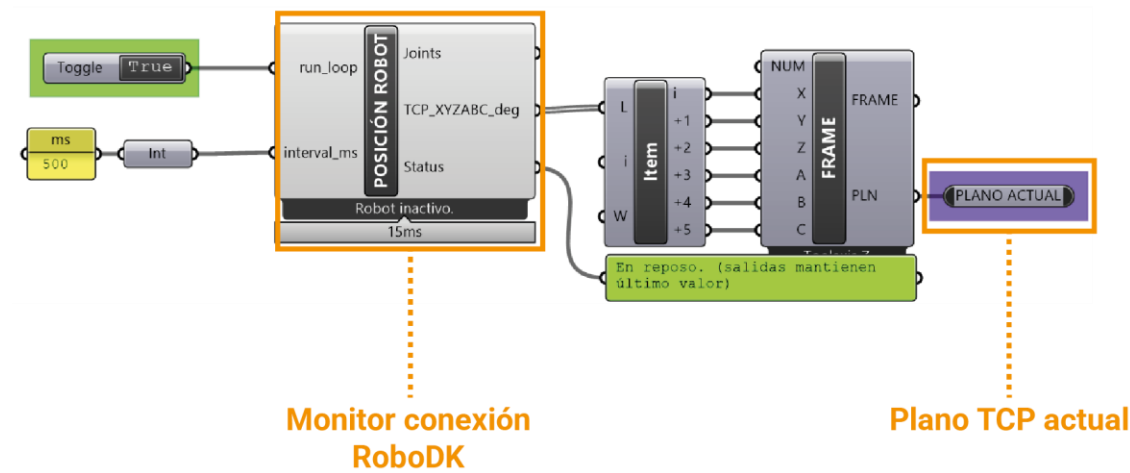
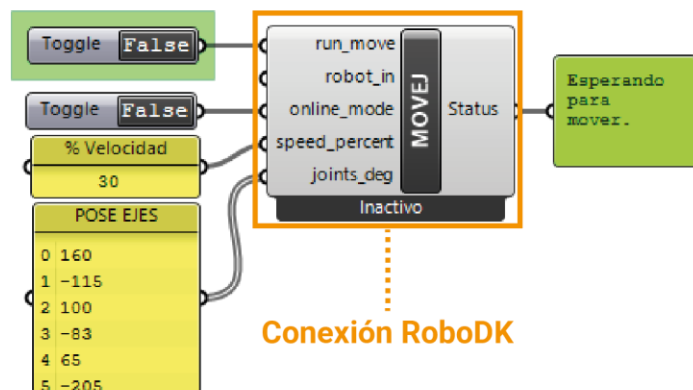


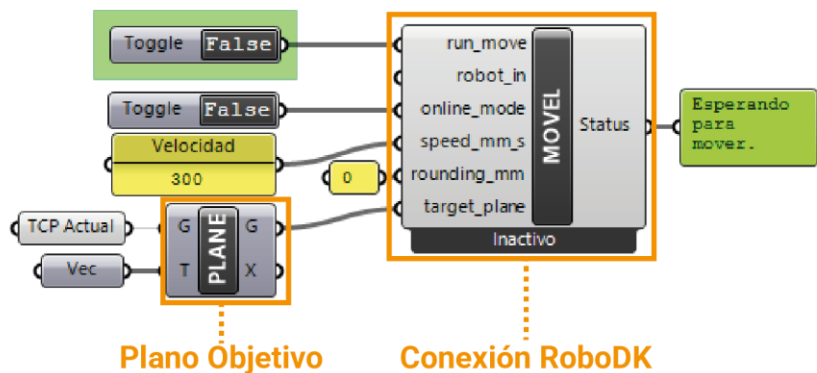
Fig. 37 Algoritmos de conexión inicial a RoboDK-Robot físico. Elaboración Propia

CONEXIÓN MOVEJ



Conexión RoboDK

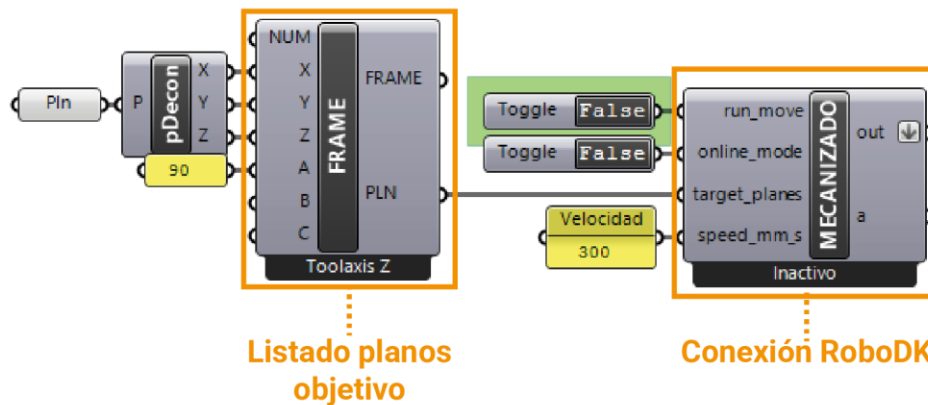
CONEXIÓN MOVEL



Plano Objetivo

Conexión RoboDK

CONEXIÓN MECANIZADO



Listado planos objetivo

Conexión RoboDK

Fig. 38. Algoritmos de conexión. Elaboración Propia.

2.2.6 Flujo de Trabajo

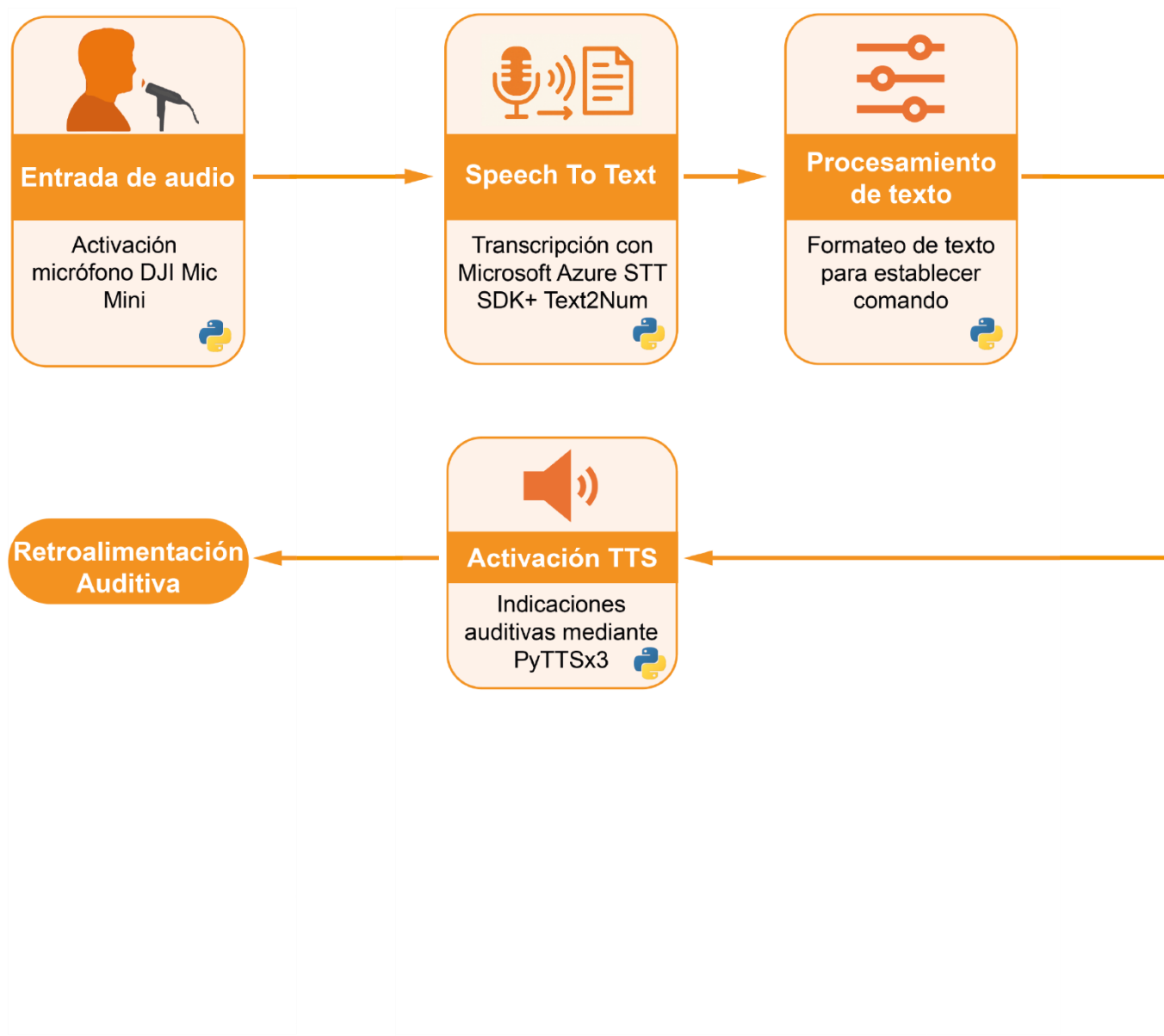
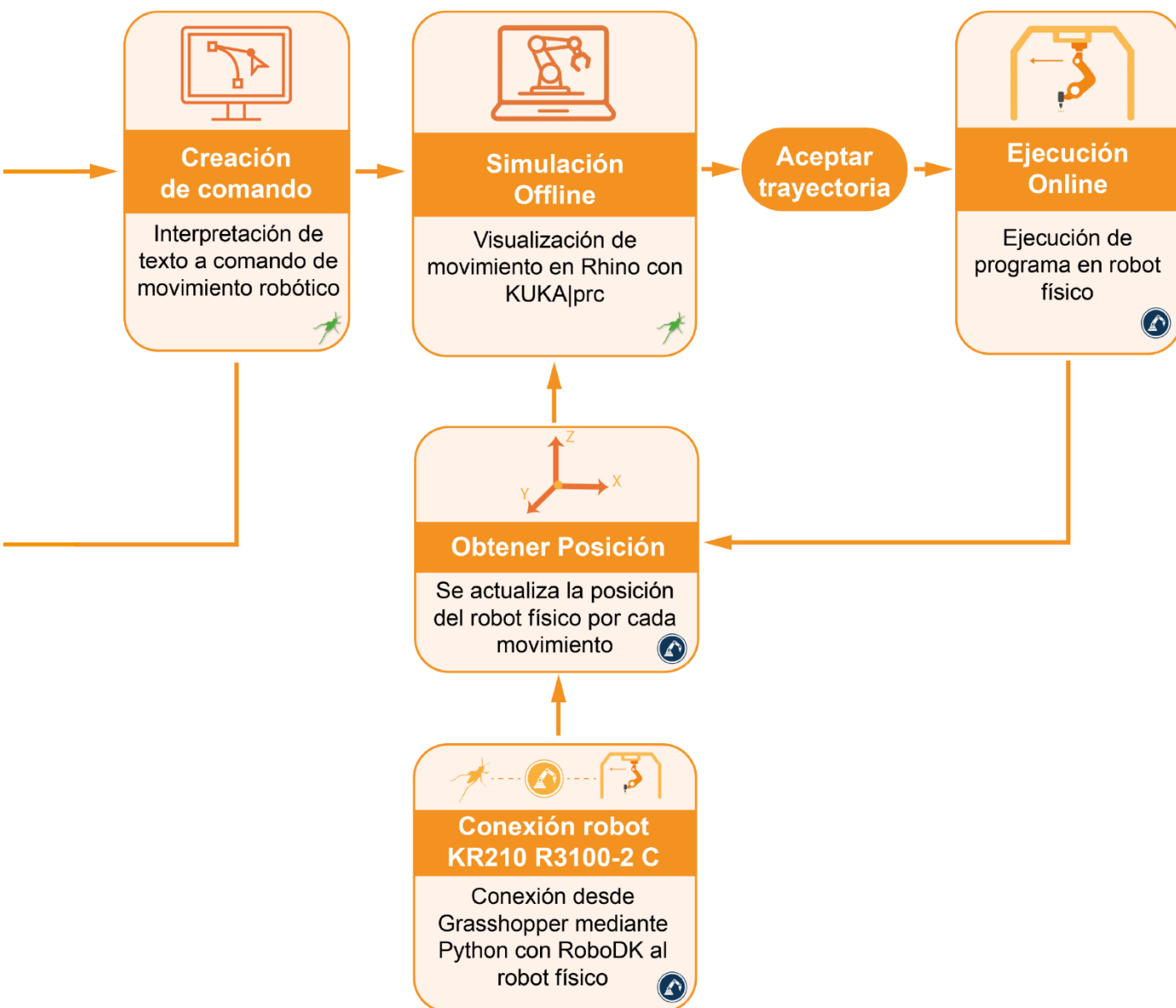


Fig. 39. Flujo de Trabajo. Elaboración Propia



El flujo de trabajo del sistema de control por voz se organiza como una cadena de módulos que van desde la captura de audio hasta la ejecución del movimiento en el robot físico. En primer lugar, el usuario activa el micrófono inalámbrico DJI Mic Mini, que captura la instrucción hablada y la envía al módulo de *Speech To Text*. Allí, el motor *Microsoft Azure STT SDK* transcribe la señal de audio a texto y, mediante la librería *Text2Num*, convierte los números expresados en palabras a su representación numérica, normalizando la instrucción para su posterior tratamiento.

A continuación, el módulo de procesamiento de texto formatea la transcripción, identifica la estructura del comando y la traduce a los parámetros requeridos por el sistema robótico (dirección, magnitud, tipo de movimiento, etc.). Esta información se envía al bloque de creación de comando, donde el texto procesado se interpreta como una orden de movimiento robótico y se convierte en vectores y poses dentro del entorno de *Grasshopper*. Utilizando *KUKA/prc*, dichos comandos se simulan en modo offline sobre el modelo digital del robot en *Rhinoceros*, permitiendo visualizar la trayectoria propuesta y verificar colisiones, alcances y coherencia con el sistema de coordenadas definido. Solo cuando la trayectoria simulada es validada por el usuario a través del nodo de aceptar trayectoria, el flujo continúa hacia la fase de ejecución.

Una vez aceptado el comando, este se transmite al programa en Python que gestiona la ejecución online mediante la API de *RoboDK*, enviando las instrucciones al robot KUKA KR210 R3100-2 C conectado al sistema. En paralelo, el módulo de conexión y monitoreo actualiza de forma continua la posición del robot físico después de cada movimiento, de modo que la pose real del TCP se devuelve a *Grasshopper* para mantener sincronizados el modelo digital y la situación física en el laboratorio. De manera complementaria, el bloque de activación TTS utiliza *PyTTSx3* para generar mensajes de texto a voz que informan al usuario del comando interpretado, del estado de la ejecución y de posibles errores, cerrando así un bucle de retroalimentación auditiva que refuerza la comprensión y la seguridad operativa durante todo el flujo de trabajo controlado por voz.

2.2.7 Resultados parciales

En esta etapa se evaluó el comportamiento del sistema tanto en su integración offline, a nivel de simulación en *Grasshopper* y *KUKA/prc*, como en su integración online a través de la conexión con *RoboDK* y el robot físico. En el entorno offline no hubo errores, el motor STT reconoció correctamente la mayoría de las instrucciones y es capaz de crear simulaciones. Cuando un comando no se entendía, el sistema pide al operador repetir la instrucción o no enviaba ningún comando a simularse. Las operaciones probadas fueron de movimientos direccionales y poses determinadas, en donde no se registraron errores por parte del sistema; por otra parte, si el operario comete un error al dar un comando (sea dirección o número erróneo) se tiene la validación de la pantalla que simula el movimiento y además cualquier movimiento que deba ser enviado al robot debe aceptarse manualmente, por lo que se tiene esta precaución.

En la integración *online*, se consiguió establecer una comunicación funcional entre *Grasshopper*, *RoboDK* y el robot industrial, restringida en esta etapa a un subconjunto acotado de instrucciones. Específicamente, se implementaron y probaron comandos de movimiento direccional de tipo “dirección + número” que permitieron verificar el envío y la ejecución de desplazamientos lineales en el espacio de trabajo. Durante estas pruebas se identificaron problemas iniciales en la orientación del TCP y en la interpretación de las trayectorias, derivados de una configuración inconsistente del *ToolAxis* en el modelo de Rhinoceros respecto del *tool* definido en *RoboDK*. Una vez ajustada la orientación del eje de herramienta en *Rhino* para hacerla coincidir con la configuración de *RoboDK*, los desplazamientos lineales dejaron de presentar anomalías y se obtuvo un comportamiento coherente entre el comando hablado, la simulación y el movimiento físico del robot.

Si bien en esta fase únicamente se ensayaron movimientos direccionales básicos, los resultados parciales permiten concluir que la infraestructura de conexión online y el esquema de interpretación de comandos son compatibles con la incorporación progresiva del resto de las funciones definidas en el flujo de trabajo sin requerir cambios estructurales en la arquitectura del sistema.

2.3 EXPERIMENTO DE VALIDACIÓN TRL4

El experimento de validación se centra en un caso de uso representativo de la construcción robotizada en madera: la medición y definición de la BASE (sistema de referencia de trabajo asociado a la pieza) y el mecanizado de perforaciones en vigas para la instalación de uniones metálicas. En los procesos de construcción y fabricación en madera, la precisión en la localización de estas perforaciones es determinante para garantizar el correcto montaje, la alineación de los elementos estructurales y la transferencia adecuada de cargas. Tradicionalmente, estas tareas se realizan mediante métodos manuales o con maquinaria que exige configuraciones complejas y tiempos de ajuste elevados, lo que introduce riesgos de error humano, sobrecostos y dificultades para adaptar el proceso a geometrías variables.

Cabe precisar que, en su formulación original, este experimento se planteó como una validación en nivel TRL 3, es decir, como una prueba de concepto donde el énfasis estaba puesto en demostrar, a nivel analítico y experimental, la viabilidad del flujo de control por voz integrado al entorno de simulación. Sin embargo, el grado de avance alcanzado durante el desarrollo del proyecto; particularmente la integración efectiva entre el sistema de reconocimiento de voz permite reclasificar el experimento de validación en el nivel TRL 4, correspondiente a una validación de tecnología en entorno de laboratorio mediante un prototipo integrado y funcional.

En este contexto, el experimento busca validar en un entorno de laboratorio (TRL 4) la factibilidad de utilizar un sistema de control por voz integrado a un flujo de trabajo visual paramétrico para calibrar la BASE del robot sobre una viga de madera y ejecutar el patrón de perforaciones requerido para uniones metálicas. La prueba no solo verifica que el robot KUKA pueda realizar el mecanizado, sino que además evalúa si un operador puede conducir todo el procedimiento, desde la conexión del sistema hasta la ejecución del taladrado mediante comandos hablados, reduciendo la dependencia del controlador *SmartPAD* y simplificando la interacción con el robot. Para ello, el experimento se estructura como una secuencia de fases definida en un algoritmo de *Grasshopper*, que organiza el flujo de control y comunicación entre el reconocimiento de voz, los scripts en *Python*, la estación de *RoboDK* y el robot físico.

2.3.1 Condiciones experimentales y configuración del sistema

El experimento se desarrolla en el Laboratorio de Construcción y Manufactura Robotizada, ubicado en el Campus San Joaquín de la Universidad Técnica Federico Santa María, sobre un caso representativo de construcción en madera: el mecanizado de perforaciones en vigas estructurales para la instalación de uniones metálicas. La pieza de trabajo corresponde a una viga de madera laminada de sección 90x180, fijada mecánicamente sobre una mesa de trabajo para evitar desplazamientos durante la operación. Esta configuración reproduce de manera simplificada una situación de taller u obra, donde la precisión en la ubicación de los orificios condiciona la correcta alineación de los herrajes y, por extensión, la geometría final del montaje estructural.

El sistema robótico empleado está compuesto por un robot KUKA KR 210 R3100-2 C de seis ejes en un carro KL4000 en un eje lineal externo montado a cielo, equipado con un controlador KR C4 y operado mediante el dispositivo *SmartPAD*, el cual en este trabajo se utiliza solo como interfaz de seguridad y supervisión. En el extremo del brazo se monta una fresa helicoidal de 20mm de diámetro, configurada como efector final del robot. La viga se ubica dentro del alcance de trabajo del manipulador, con un volumen de seguridad despejado a su alrededor para permitir movimientos de aproximación, retirada y maniobras de calibración sin riesgo de colisión.

El sistema de control se articula a través de un computador conectado a la misma red que el robot, que integra tres componentes de *software* principales: *Rhinoceros* con *Grasshopper* como entorno de simulación y programación visual; *RoboDK* como plataforma de comunicación con el controlador del robot; y un conjunto de *scripts* en *Python* dentro de *Grasshopper* que implementan el reconocimiento de voz (STT), retroalimentación auditiva (TTS), la gestión de los datos de movimiento y configuración. El reconocimiento de voz se realiza mediante el micrófono DJI Mic Mini conectado al computador, de forma que el operador pueda emitir comandos hablados a una distancia del robot, manteniendo contacto visual con la pieza y el efector. El operario se ubica en la misma zona utilizada para las simulaciones offline, por ende, las direcciones habladas corresponden a las mismas coordenadas que se hicieron en las pruebas offline.

Desde el punto de vista de la configuración del sistema, *Grasshopper* actúa como núcleo de orquestación del flujo: en él se implementa el algoritmo secuenciado en fases, que define las etapas de conexión, monitoreo, posicionamiento inicial, movimientos direccionales, registro de BASE, definición de origen de programa y generación del patrón de perforaciones. Cada uno de estos módulos se comunica con *RoboDK* mediante los componentes y scripts en *Python* mencionados en la sección 2.2.3, los cuales traducen las instrucciones derivadas de los comandos de voz en trayectorias y posiciones articulares compatibles con el robot KUKA.

0- Conectar: - *Python* que conecta con la estación de *RoboDK*, inicializando la comunicación entre *Grasshopper* y el robot virtual. En esta fase se carga la estación correspondiente (robot KUKA y herramienta), se establece el enlace con el controlador mediante el *driver* de KUKA y se verifica que la IP, el puerto y el estado del servidor permitan el envío de órdenes *online*. El objetivo es asegurar que todas las capas de *software* compartan la misma escena de trabajo antes de mover el robot físico.

1- Monitor TCP: *Python* que actualiza la posición del robot real en *Grasshopper*, consultando de forma periódica a *RoboDK* la pose actual del robot. Esta pose se obtiene como un *FRAME* en formato XYZABC otorgando el plano TCP (*Tool Center Point*), y se envía a *Grasshopper* para que el modelo en *Rhino* replique en tiempo real la orientación y posición del manipulador. Así, la simulación actúa como espejo del robot físico, permitiendo visualizar su movimiento a partir de su posición real.

2- Ir a posición: Se utiliza un STT para ir a posiciones predeterminadas como en la simulación offline, donde el reconocimiento de voz traduce comandos “ir a” como posiciones según un diccionario de poses. Este diccionario asocia nombres (Guardado, trabajo, inicio, *default*) a configuraciones articulares específicas. Luego *Python* conecta *RoboDK* con el componente de conexión de movimiento articular (*MoveJ*) y, a través de este, con el robot físico para ejecutar

el movimiento hacia la pose indicada. Esta fase garantiza que el robot parta siempre desde estados conocidos y repetibles antes de ajustar su posición respecto a la viga de madera.

3- Movimiento Direccional: STT en *Python* que recibe direcciones y números para hacer movimientos traslacionales del TCP, interpretando comandos de formato “dirección+ número” en desplazamientos incrementales en milímetros. Estos incrementos se aplican sobre el eje local del TCP, permitiendo ajustar constantemente la posición del efector final. Se utiliza el componente de *Python* para movimientos lineales (*MoveL*).

4- Registrar BASE: STT donde, mediante direcciones y confirmaciones por voz, se registran los 3 puntos necesarios para definir la BASE sobre la viga. El robot es guiado por comandos direccionales hasta cada punto de referencia, y el operador emite un comando para registrar el origen, la coordenada X y el plano XY. Con estos tres puntos, el *script* en *Python* calcula el nuevo sistema de coordenadas BASE (origen y orientación de los ejes) y lo guarda en *Grasshopper*.

5- Origen Programa: STT de movimientos direccionales que permite registrar un punto de inicio para mecanizado de perforaciones, correspondiente a la ubicación de la primera perforación en la matriz. El operador ajusta el TCP sobre el punto deseado en la superficie de la viga mediante comandos de voz, y luego dicta un comando para registrar origen de programa. *Python* almacena esa pose como referencia inicial del patrón de perforaciones.

6- Mecanizar perforaciones: STT tipo *chatbot* que recibe números por cantidades de perforaciones, distanciamientos (mm) y profundidad de perforaciones (mm). El sistema valida los parámetros dictados por el operador y genera un ordenamiento rectangular de perforaciones (*Rectangular Array*) como una matriz de planos objetivo en *Grasshopper*. Estos planos se envían a *RoboDK* y al robot físico como trayectorias de taladrado, incluyendo

movimientos de aproximación, perforación a la profundidad indicada y retirada, así como el encendido y apagado de la herramienta según corresponda. De este modo, la secuencia completa de perforaciones se ejecuta de forma automatizada a partir de instrucciones definidas únicamente por voz.

RoboDK, por su parte, mantiene una estación virtual que replica la disposición física del experimento con el modelo del robot y herramienta. Esta estación cumple la función de servir como puente de comunicación en tiempo real con el controlador KR C4, permitiendo la ejecución online de las trayectorias generadas desde *Grasshopper*.

En conjunto, las condiciones experimentales y la configuración del sistema buscan aproximarse a un escenario realista de fabricación en madera, pero bajo control de laboratorio, donde es posible observar con detalle la respuesta del prototipo: la robustez de la comunicación entre capas, la capacidad del reconocimiento de voz para interpretar correctamente los comandos y la precisión con que el robot registra la BASE y ejecuta el patrón de perforaciones requerido para la instalación de uniones metálicas.

Para la ejecución del experimento se sigue una secuencia operativa que articula el sistema robótico nativo KUKA con la estación virtual de *RoboDK* y el flujo de programación visual en *Grasshopper*. En primer lugar, se prepara el sistema robótico en su configuración nativa: se energiza el robot KUKA KR 210 R3100-2 C y, mediante el *SmartPAD*, se carga el programa para habilitar la comunicación externa con *RoboDK*, dejando el robot referenciado y disponible para recibir órdenes *online*. Una vez verificado el estado del sistema el robot queda en una posición segura de espera.

En paralelo, en el computador de control se abre la estación del laboratorio en *RoboDK*, que contiene el modelo del robot, la herramienta de fresado y la viga en la posición correspondiente al montaje físico. A continuación, se abre el archivo de *Grasshopper* que implementa el algoritmo de control por voz y las fases 0 a 6. El operador activa en primer término las fases 0-Conectar y 1-Monitor TCP, con lo cual se establece la comunicación entre *Grasshopper* y *RoboDK* y se comienza a actualizar en tiempo real la pose del robot físico en la escena de *Rhino*. Solo una vez comprobado que el modelo

virtual refleja correctamente los movimientos del robot real se continúa con las siguientes fases.

Antes de detallar la secuencia operativa, es pertinente señalar que el experimento se ejecutó en tres ocasiones. En las dos primeras, la fase 2 “Ir a posición” no incorporaba aún el control por voz, sino que el robot era enviado a una posición predeterminada directamente desde la interfaz de *Grasshopper/RoboDK*, sin mediación de comandos hablados; en la primera y tercera ejecución los comandos del resto de las fases fueron emitidos por la autora mientras que en la segunda fueron emitidos por el profesor referente. Solo en la tercera ejecución, correspondiente a la versión final del experimento, se trabajó con un libreto basado en una trayectoria previamente probada e incorporando el módulo de STT en la fase 2, de modo que la orden de ir a la pose de trabajo se emitiera mediante comandos de voz de la autora, tal como se describe en los párrafos siguientes.

Para iniciar la operación, se habilita la fase 2 “Ir a posición” y, mediante un comando de voz, se ordena al robot ir a la “pose de trabajo” predefinida en el diccionario de posiciones. *Grasshopper*, a través de *Python* y *RoboDK*, envía la orden articular (*MoveJ*) al controlador, situando al robot en una configuración estándar desde la cual resulta seguro aproximarse a la viga. Alcanzada esta pose, se activa la fase 3 “Movimiento Direccional”, permitiendo al operador acercar el TCP a la pieza mediante comandos de voz de formato “dirección+ número”, hasta posicionar la herramienta en las proximidades del área donde se registrará la BASE.

Una vez que el robot se encuentra suficientemente próximo a la viga, se activa la fase 4 “Registrar BASE”. Guiando al robot con los mismos movimientos direccionales controlados por voz, el operador posiciona el TCP sobre los tres puntos de referencia necesarios (origen, eje X y plano XY) y, en cada caso, dicta el comando de registro correspondiente. Con ello se calcula y almacena en *Grasshopper* el nuevo sistema de coordenadas BASE asociado a la pieza. A continuación, se habilita la fase 5 “Origen Programa”, en la cual el operador, nuevamente mediante movimientos direccionales por voz, ajusta la herramienta hasta la ubicación exacta de la primera perforación de la matriz y dicta el comando para registrar el origen del programa de mecanizado.

Definidos tanto la BASE como el origen del programa, se activa la fase 6 “Mecanizar perforaciones”. En esta etapa, el módulo de STT tipo *chatbot* solicita y recibe los parámetros necesarios (número de perforaciones, distancias entre ellas según eje y profundidad en milímetros), genera la matriz rectangular de objetivos y envía al robot, a través de *RoboDK*, la secuencia completa de movimientos de taladrado. El robot ejecuta entonces el ciclo de perforaciones sobre la viga, coordinando los movimientos de aproximación, perforación y retirada de la herramienta según los parámetros indicados por voz. Finalizado el mecanizado, el operador vuelve a habilitar la fase 3- Movimiento Direccional para alejar el TCP de la pieza y situarlo en una posición segura de retiro. Finalmente, se reactiva la fase 2-Ir a posición, esta vez ordenando al robot ir a la “posición de guardado”, desde donde queda listo para detener el sistema o repetir el procedimiento con una nueva pieza.

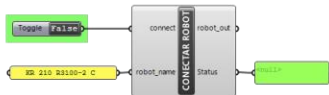
Esta secuencia paso a paso asegura que el experimento se desarrolle de manera controlada y reproducible, manteniendo una relación clara entre las fases del algoritmo de *Grasshopper*, las acciones del operador y el comportamiento del robot físico durante la validación TRL 4.

2.3.2 Registros del experimento

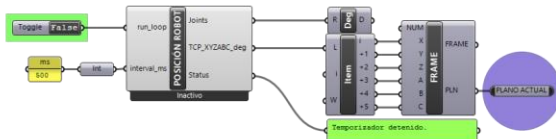
La presente subsección reúne los registros visuales de la tercera ejecución del experimento, correspondiente a la versión realizada con libreto y con el módulo de reconocimiento de voz plenamente integrado en todas las fases. Las imágenes se organizan de manera secuencial, la preparación del sistema, la aproximación del robot a la viga, el registro de la BASE, la definición del origen de programa y el desarrollo completo del mecanizado de perforaciones sobre la pieza de madera. Esta selección se realiza sobre la tercera ejecución por ofrecer la demostración gráfica más clara y consistente del flujo propuesto, al tratarse de una trayectoria previamente ensayada y ejecutada bajo las condiciones finales de validación TRL 4.

ALGORITMO GRASSHOPPER

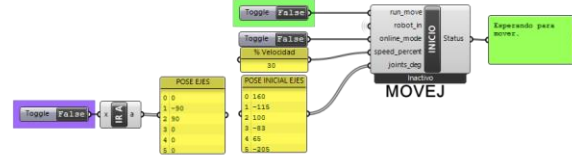
0 - CONECTAR



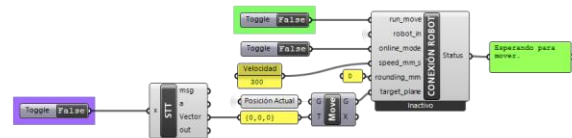
1 - MONITOR TCP



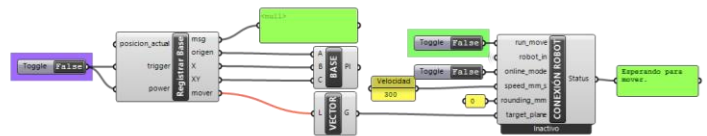
2 - IR A POSICIÓN



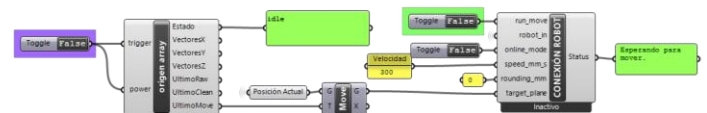
3 - MOVIMIENTO DIRECCIONAL



4 - REGISTRAR BASE



5 - ORIGEN PROGRAMA



6 - MECANIZAR PERFORACIONES

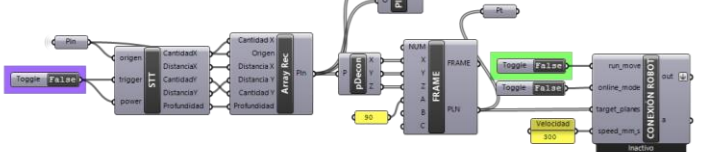


Fig. 40. Algoritmo completo experimento de validación. Elaboración Propia

ENTORNO DE TRABAJO

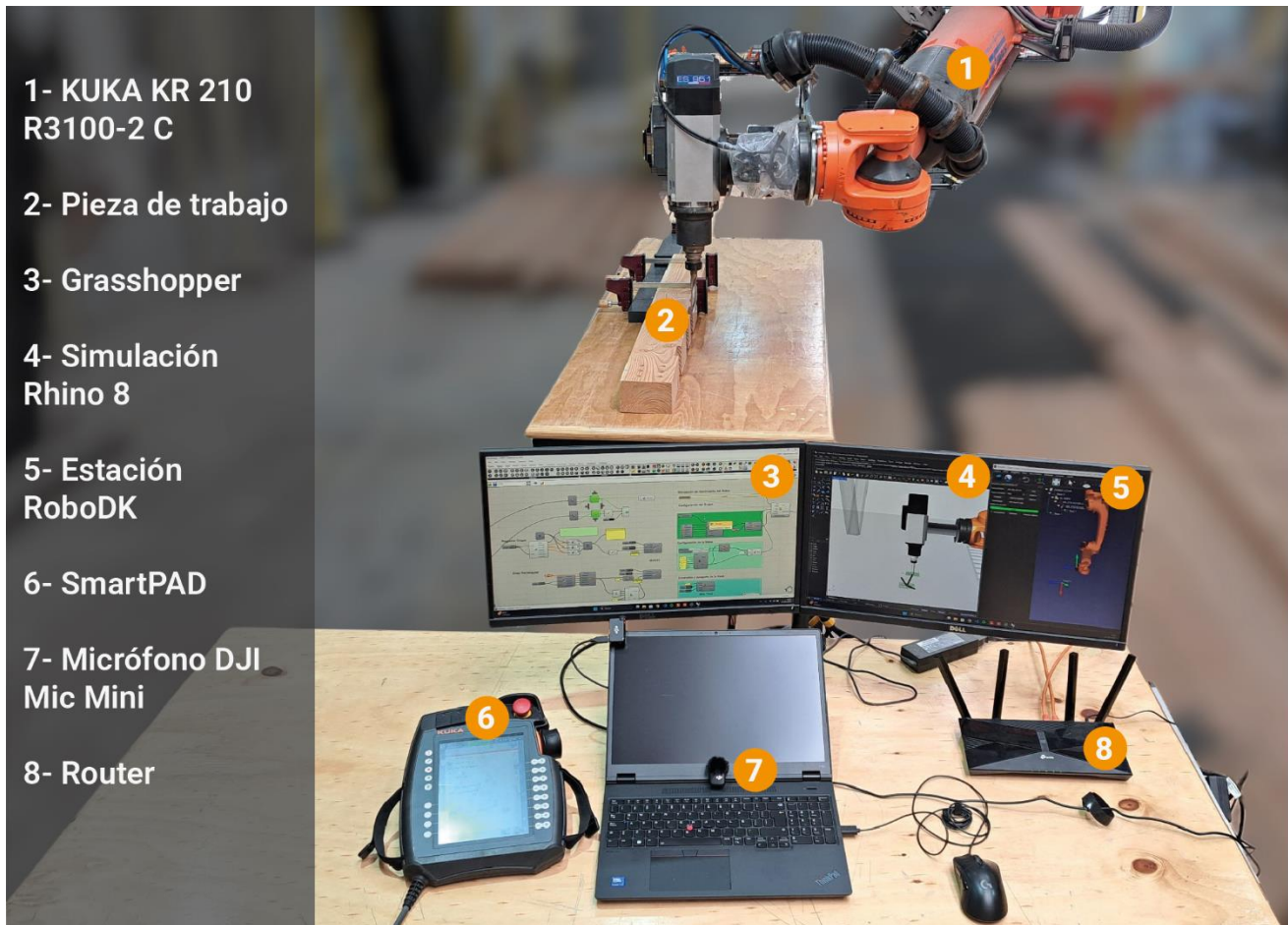


Fig. 41. Componentes del entorno de trabajo. Elaboración Propia

POSICIÓN DE TRABAJO

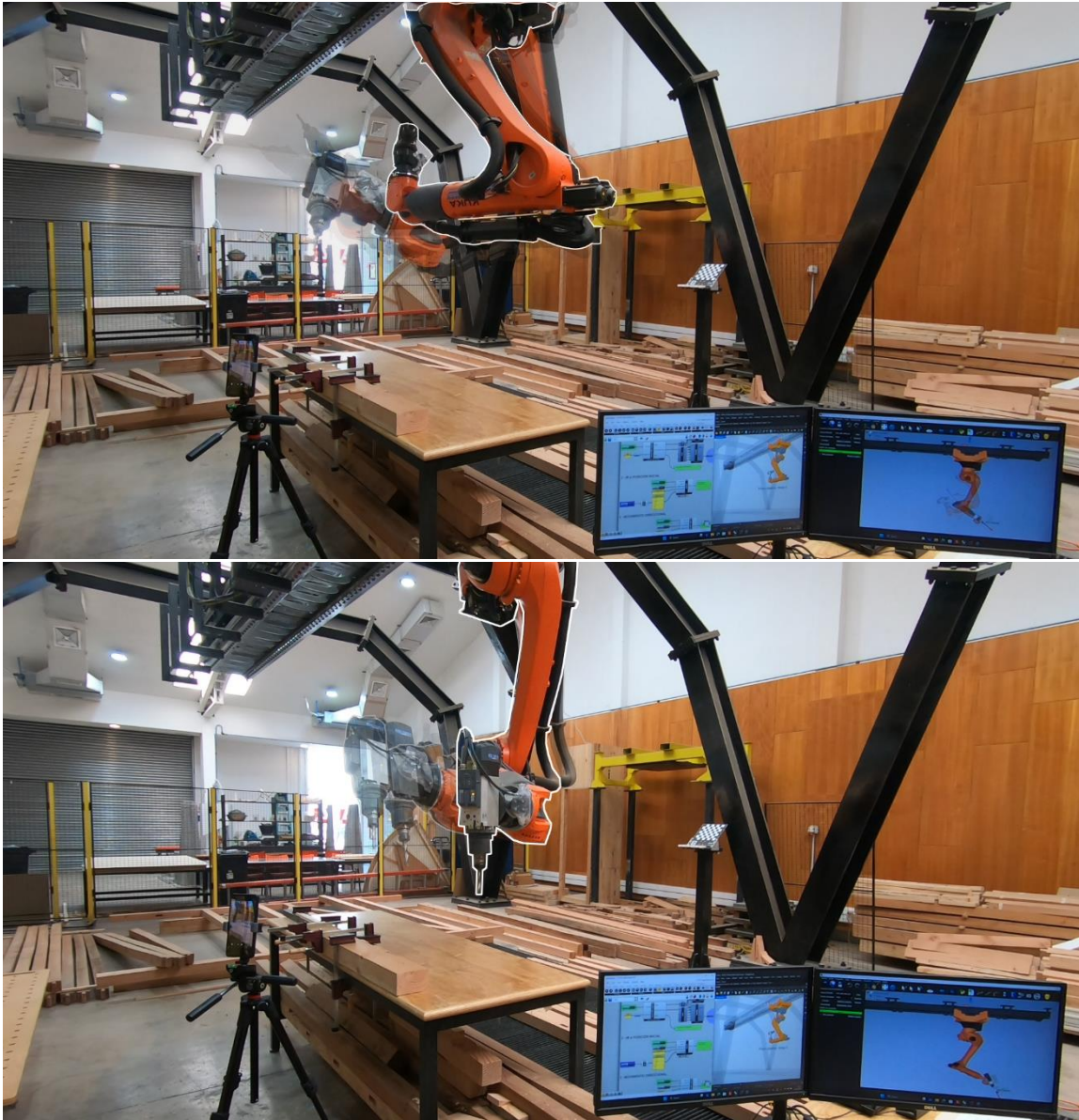


Fig. 42. Posición de trabajo. Elaboración Propia

MEDICIÓN BASE

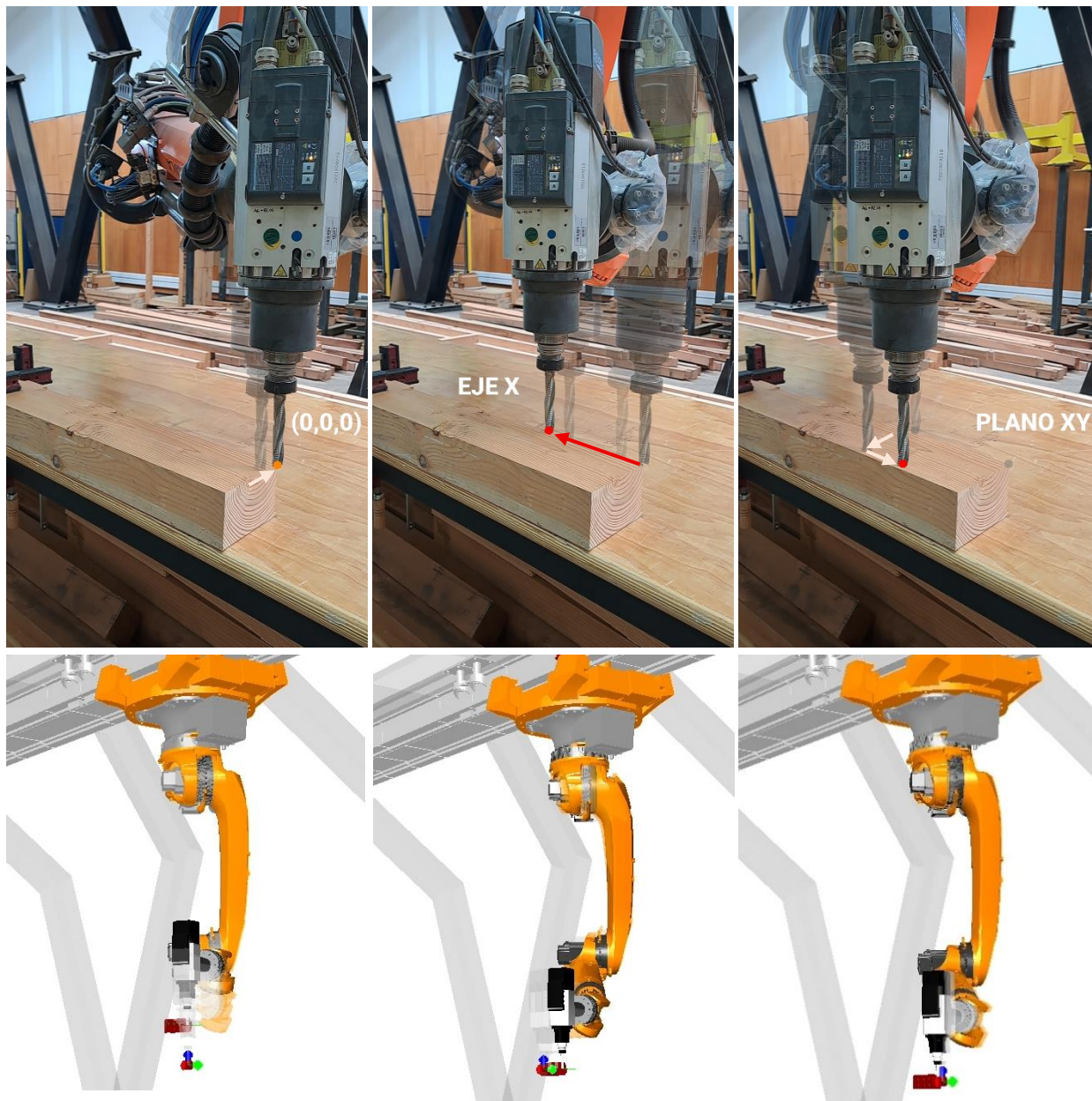


Fig. 43. Registro de SC de la pieza de trabajo. Elaboración Propia

MECANIZADO PERFORACIONES

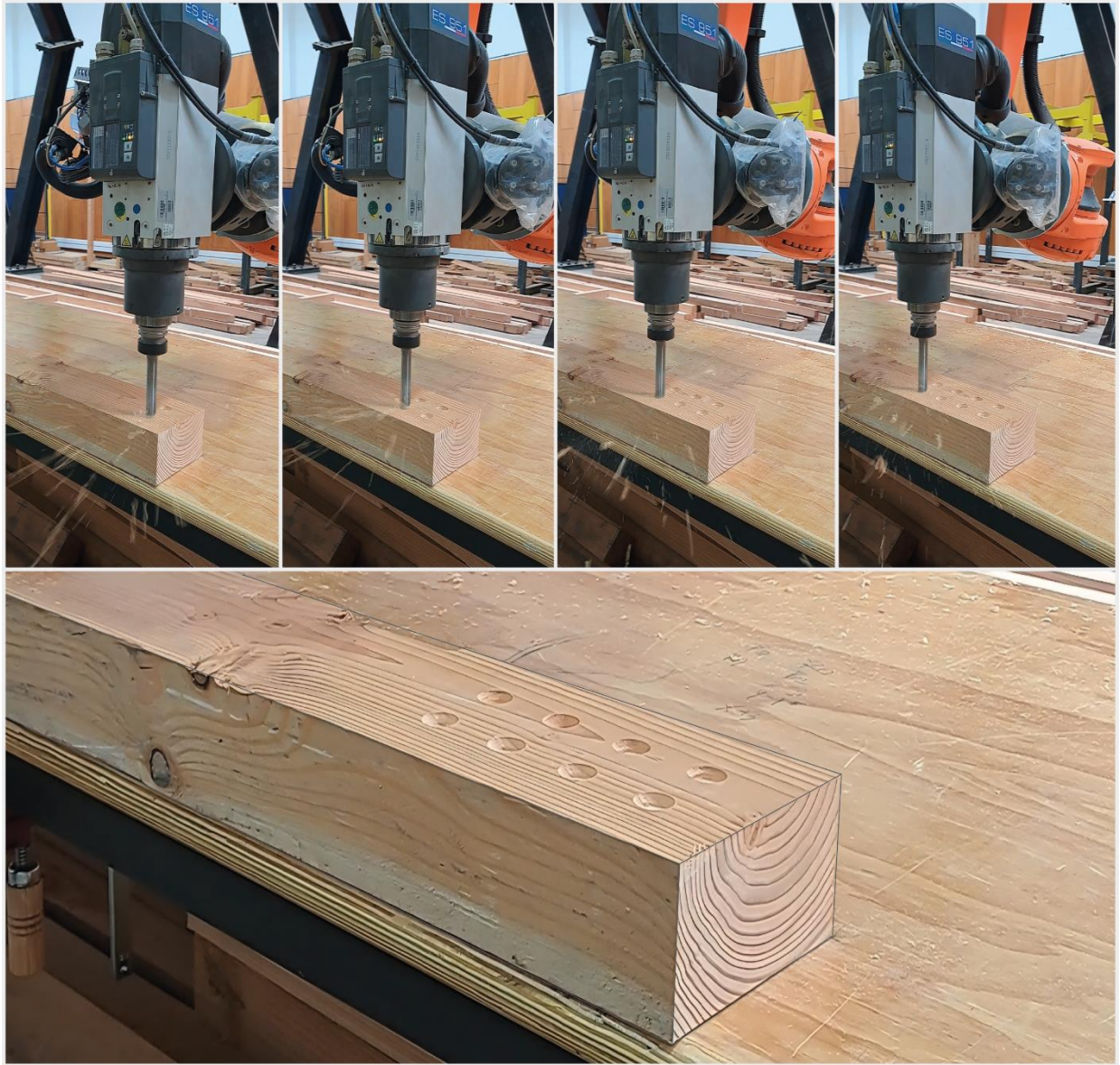


Fig. 44. Mecanizado de perforaciones. Elaboración Propia

POSICIÓN DE GUARDADO

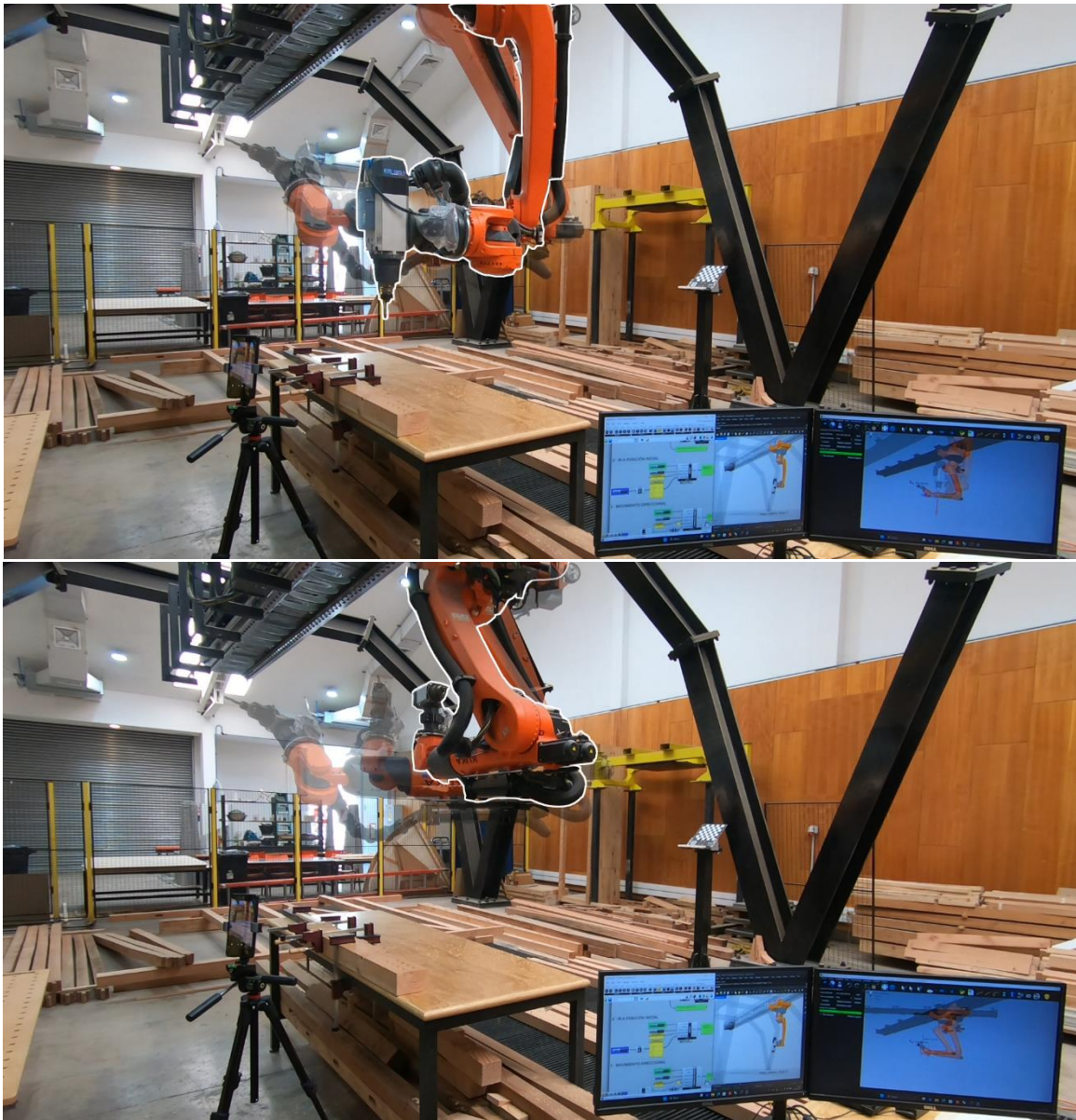


Fig. 45. Posición de guardado. Elaboración Propia

2.3.3 Resultado

El experimento de validación se ejecutó utilizando el conjunto completo de funciones desarrolladas en el prototipo de control por voz, siguiendo la secuencia definida en el flujo de trabajo: posicionamiento inicial del robot en una posición de trabajo segura, medición de la base mediante el sistema de monitoreo del TCP, definición del origen de referencia para el patrón de perforaciones y posterior ejecución del mecanizado. En la ejecución final del experimento, todas las etapas se llevaron a cabo de forma continua y sin interrupciones, evidenciando una integración correcta del reconocimiento de voz, la lógica de interpretación de comandos en *Python*, el entorno paramétrico en *Grasshopper* y la comunicación con *RoboDK* con el robot físico. (Anexo B).

En términos operativos, el robot fue capaz de desplazarse a la posición de trabajo indicada, registrar los puntos necesarios para la base y establecer un sistema de coordenadas, para luego generar el patrón de perforaciones a partir de las instrucciones verbales del operador. La generación y el envío de trayectorias se realizaron conforme a la definición paramétrica prevista, de modo que las perforaciones se ejecutaron en las posiciones esperadas, sin desviaciones apreciables respecto de la simulación y sin requerir ajustes manuales durante la secuencia. No se registraron errores de comunicación ni fallas en la interpretación de los comandos de voz dentro de la prueba, y los movimientos del robot se mantuvieron dentro de las condiciones de seguridad consideradas para el experimento. Sin embargo, es recomendable poder medir algunas trayectorias en el espacio físico antes de ejecutarlas por voz, ya que, por ejemplo, bajar en el eje Z puede comprometer la pieza o el espacio de trabajo, pues, a pesar de la retroalimentación hablada y la simulación en pantalla, no se sabe de los elementos dentro del espacio (como la pieza de trabajo, mesa, prensas etc.) por lo que se debe tener precaución al ejecutar una acción en el eje vertical.

Estos resultados permiten afirmar que el sistema de control por voz logró cumplir el objetivo principal del experimento: demostrar que es posible ejecutar un flujo de trabajo completo de tareas de procesos de fabricación en madera, que incluye calibración básica del sistema de referencia, definición de un origen de trabajo y

activación de trayectorias parametrizadas mediante comandos hablados, sin recurrir a la programación directa en el controlador ni a interfaces complejas. Si bien se trata de un caso de uso acotado a una tarea específica y a un entorno controlado de laboratorio, la ejecución exitosa de todas las funciones programadas valida la arquitectura propuesta y sienta una base sólida para extender el sistema a otros procesos de mecanizado y aplicaciones de la construcción robotizada en futuras etapas de desarrollo.



Fig. 46. Resultado de HRC en el entorno de trabajo. Elaboración Propia

Sobre la base de esta validación experimental se elaboró y presentó un artículo al congreso SIGraDi 2025 (Anexo C), en el que se sintetizan la arquitectura del sistema, la metodología de desarrollo y los resultados obtenidos en el caso de estudio de

perforación asistida por voz. Dicho trabajo se enmarca en las temáticas de fabricación digital y flujo de trabajo paramétrico aplicados a la construcción en madera, y propone el sistema como una interfaz alternativa de baja fricción entre diseñadores, arquitectos y robots industriales KUKA para tareas de construcción robotizada en madera. El artículo describe el acoplamiento entre reconocimiento de voz, definición paramétrica en Grasshopper y ejecución robótica vía RoboDK, detalla el protocolo del experimento y discute el desempeño del sistema en términos de precisión, repetibilidad y usabilidad percibida por el operador, posicionando esta contribución como un paso intermedio hacia herramientas más accesibles de programación robótica en contextos de arquitectura y construcción.

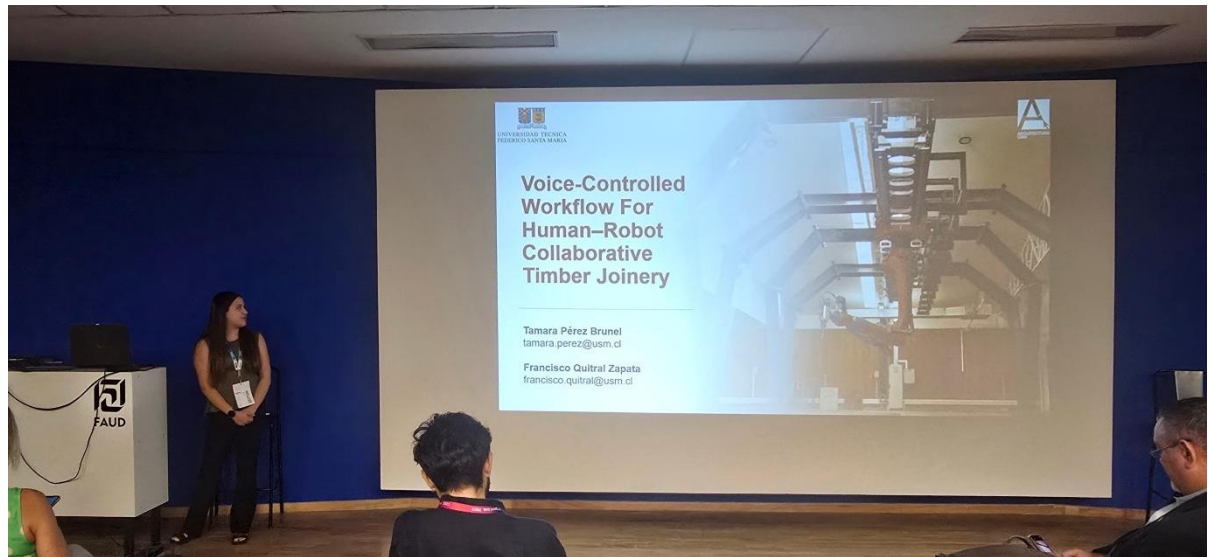


Fig. 47. Presentación de trabajo en SIGraDi 2025- Córdoba, Argentina. Elaboración Propia

A partir de la ejecución del experimento se consideran validadas, en condiciones reales de operación con el robot físico, las siguientes características principales del sistema de control por voz, las cuales se articularon de manera coherente dentro del flujo de trabajo y experimento propuesto:

CARACTERÍSTICAS DEL SISTEMA



Fig. 48. Características del sistema. Elaboración Propia

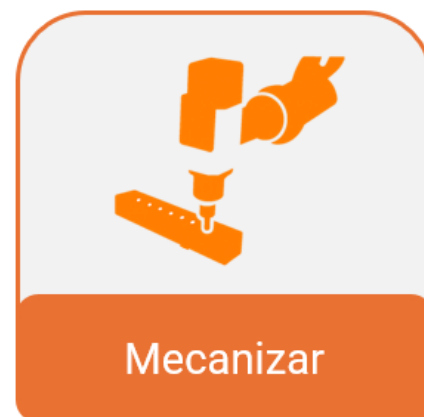
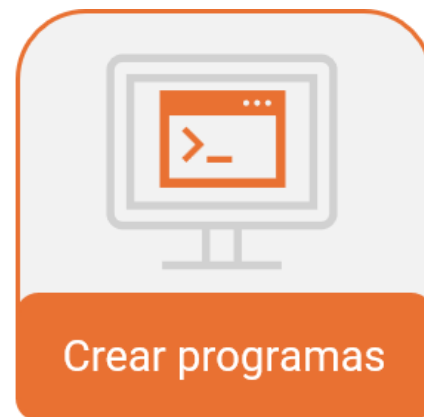


Fig. 48. Características del sistema. Elaboración Propia

3. CONCLUSIONES

En síntesis, el desarrollo del prototipo de sistema de control por voz para un robot industrial KUKA permite concluir que la integración de reconocimiento de voz, programación visual y simulación robótica constituye una alternativa válida y eficiente al uso intensivo del controlador *SmartPAD* para la calibración y manipulación del sistema de coordenadas en contextos de construcción robotizada en madera. El flujo de trabajo implementado demuestra que es posible traducir instrucciones habladas en español a comandos estructurados que se simulan y validan previamente en entorno digital, reduciendo pasos intermedios de reajuste, disminuyendo la carga operativa y mejorando la ergonomía del proceso, especialmente en tareas prolongadas o repetitivas vinculadas a la medición de la pieza de trabajo y a la ejecución de patrones de perforación. La combinación de simulación visual y retroalimentación auditiva configura un modelo de control bidireccional comprensible para usuarios no expertos, incrementando la seguridad operativa al verificar virtualmente cada trayectoria antes de su ejecución física sobre la pieza. La validación experimental en laboratorio, mediante la ejecución de una matriz de perforaciones en madera utilizando exclusivamente comandos de voz, confirma la viabilidad técnica del sistema y su capacidad para sostener niveles de precisión compatibles con los requerimientos de la construcción robotizada, al tiempo que evidencia su potencial para reducir barreras de acceso a la robótica industrial en ámbitos académicos y profesionales. Sin embargo, también se constata que el desempeño del reconocimiento de voz sigue condicionado por factores como el ruido ambiental en caso de no tener un hardware apto para ello, la dependencia de servicios en la nube al no utilizar STT locales y estrategias de filtrado acústico más robustas para escenarios de obra. En conjunto, los resultados obtenidos permiten afirmar que el prototipo desarrollado cumple el objetivo de habilitar un flujo de trabajo de calibración y operación básica del robot mediante comandos de voz, sienta una base operativa

escalable hacia procesos de mecanizado más complejos y abre una línea de continuidad en la investigación aplicada sobre colaboración humano–robot en la construcción en madera y en la formación de arquitectos en entornos de fabricación digital.

En relación con los objetivos planteados, es posible afirmar que el proyecto cumple de manera ordenada y coherente con cada uno de ellos. En primer lugar, se alcanza el objetivo general de diseñar e implementar un sistema de control por voz integrado a un flujo de trabajo visual paramétrico para la calibración y manipulación del sistema de coordenadas de un robot KUKA, mediante la articulación funcional entre motor de reconocimiento de voz, entorno de programación visual y simulación robótica. En segundo lugar, se satisfacen los objetivos específicos asociados a la definición de un vocabulario de comandos hablados, la estructuración de un protocolo de uso y la implementación de un flujo de calibración ejecutable por voz, los cuales se consolidan en el experimento de perforación sobre piezas de madera. En tercer lugar, la validación secuencial en simulación y en robot físico demuestra que el sistema es capaz de ejecutar trayectorias con precisión suficiente para tareas básicas de construcción robotizada, confirmando su viabilidad técnica. Finalmente, se verifica el objetivo de mejorar la accesibilidad y reducir la barrera técnica de entrada a la robótica industrial: el operador deja de depender de la programación directa en el controlador, interactúa mediante lenguaje natural y puede concentrarse en la lógica constructiva antes que en la sintaxis del código. De este modo, el prototipo no solo cumple con los objetivos del trabajo, sino que se configura como una herramienta de base sobre la cual es posible escalar a operaciones de mecanizado más complejas, incorporar nuevas herramientas y extender el modelo de colaboración humano–robot hacia escenarios de obra y de enseñanza de la fabricación digital en arquitectura.

3.1 DISCUSIÓN

Considerando los resultados obtenidos y la experiencia acumulada, el impacto del sistema de control por voz puede leerse en tres planos complementarios: académico, de obra y de fabricación en madera. En el ámbito académico, el prototipo se consolida

como una herramienta didáctica de baja barrera que reduce de forma efectiva la complejidad asociada al *SmartPAD* y a las interfaces de programación tradicionales, permitiendo que estudiantes y docentes se concentren en comprender la lógica geométrica de la calibración y la relación entre sistema de referencia, pieza de trabajo y espacio de trabajo del robot. La posibilidad de formular comandos en lenguaje natural, observar inmediatamente su traducción a trayectorias simuladas y recibir una retroalimentación auditiva del sistema favorece un aprendizaje experiencial, donde es posible visualizar consecuencias espaciales, anticipar colisiones potenciales y reflexionar sobre alcances y limitaciones del robot sin arriesgar equipamiento ni material. En este contexto, el sistema no solo introduce conceptos fundamentales de robótica industrial (coordenadas, rotaciones, trayectorias, espacio de trabajo), sino que también habilita la exploración temprana de escenarios de colaboración humano-robot, en los que el estudiante asume un rol de coordinación y toma de decisiones mientras el robot ejecuta operaciones de precisión. De este modo, el control por voz opera como un mediador entre la abstracción del modelo digital en *Grasshopper* y la acción física del robot, acercando la formación de arquitectos y arquitectas a prácticas de fabricación digital más próximas a la realidad profesional contemporánea.

En obra, el sistema introduce una modalidad de interacción sin contacto físico y más natural para realizar perforaciones y el reajuste del sistema de coordenadas, que adquiere relevancia en contextos donde la posición de las piezas de madera puede variar por tolerancias de montaje, deformaciones o cambios de secuencia constructiva. La reducción de pasos intermedios y la disminución de desplazamientos frente al controlador permiten ajustar el sistema de referencia mientras se mantienen las manos libres para supervisar el montaje, manipular herramientas auxiliares o coordinar al equipo de trabajo, lo que mejora la ergonomía y la continuidad del flujo de trabajo, especialmente en procesos prolongados o repetitivos. La validación mediante simulación visual antes de ejecutar sobre la pieza se vuelve un filtro crítico para anticipar interferencias con elementos ya instalados, andamios u otros oficios en el frente de trabajo, reforzando la seguridad operativa y la coordinación entre agentes. El tener la capacidad de programar perforaciones con voz permite reducir el error humano y realizar en serie las miles de perforaciones que puede tener un edificio hecho de madera sin la necesidad de tener todo el conocimiento técnico, asegurando

un resultado preciso a través de comandos simples que resultan más intuitivos que las interfaces tradicionales. Sin embargo, esta potencialidad debe leerse a la luz de las condiciones reales de obra: el ruido ambiental, el uso de equipos de protección personal y la presencia de múltiples interlocutores pueden afectar el rendimiento del reconocimiento de voz, lo que podría requerir micrófonos especializados, protocolos de activación más estrictos o incluso zonas de comando delimitadas. Aun así, el hecho de que supervisores o jefes de terreno, con un conocimiento moderado del flujo de trabajo, puedan dirigir calibraciones, mecanizados y ajustes a través de comandos verbales, sugiere un escenario en el que el control por voz se integra paulatinamente en la lógica cotidiana de una obra, donde las tareas rutinarias pueden ser automatizadas o adaptables al replanteo de perforaciones o la adaptación de patrones de unión frente a contingencias en sitio.

En el ámbito de la fabricación robotizada en madera, el sistema se perfila como una base escalable hacia flujos de trabajo más complejos, en la medida en que la arquitectura actual centrada en calibración y movimientos básicos puede extenderse a comandos de corte, perforación y ensamblaje, e incluso articularse con sistemas de visión computacional o módulos de inteligencia artificial y comprensión del lenguaje natural. En procesos de mecanizado prolongados, como la generación de matrices de perforaciones o la preparación de uniones metálicas con múltiples ángulos, el control por voz reduce la carga de entradas manuales repetitivas, disminuye la fatiga del operario y permite que este supervise simultáneamente la calidad del mecanizado y el entorno de trabajo. La combinación de simulación visual y retroalimentación auditiva configura un modelo de control bidireccional en el que es posible iterar rápidamente variaciones de patrones, evaluar su impacto en tiempos de ciclo y consumo de material y solo entonces autorizar la ejecución física, lo que mejora tanto la precisión técnica como la eficiencia productiva. A futuro, la incorporación de modelos STT locales, sistemas NLU para clasificar comandos y asignarles contexto, y la integración con flujos paramétricos más amplios abre la posibilidad de celdas robotizadas donde la programación de operaciones se derive directamente de un diálogo estructurado entre el operario y el sistema. En conjunto, estas proyecciones sugieren que, aun con las limitaciones actuales vinculadas al ruido, la conectividad y el alcance acotado a tareas básicas, el prototipo constituye un punto de partida consistente para discutir

cómo articular, en términos concretos, la integración de control por voz, simulación visual y robótica industrial en la formación académica, en la obra y en la fabricación arquitectónica en madera.

3.2 PROYECCIÓN

La versión actual del sistema ya habilita un conjunto de proyecciones concretas que se desprenden directamente de sus características funcionales. En primer lugar, el simple hecho de contar con una conexión estable entre *Grasshopper* y *RoboDK*, basada en un modelo digital del robot, implica que el flujo de trabajo es replicable en otras estaciones con distintos robots KUKA, siempre que exista un gemelo digital válido dentro de RoboDK. En términos estructurales, la lógica de comunicación no está acoplada a un único modelo físico, sino al uso de referencias de robot y herramientas en el entorno de simulación, por lo que el mismo esquema de control por voz podría trasladarse a otras configuraciones de celda robotizada. De manera análoga, el sistema de monitoreo del TCP es extrapolable a otros modelos de robot, convirtiéndose en una herramienta genérica dentro de *Grasshopper* para capturar posiciones y orientaciones del efector final, registrar secuencias de poses y utilizarlas como base para modelar trayectorias, mecanizados o estrategias de montaje a partir de posiciones conocidas. Esto abre la posibilidad de construir bibliotecas de poses asociadas a tareas recurrentes (montajes, aproximaciones, cambios de herramienta, etc.) que se integren de forma nativa en el flujo paramétrico.

La función “ir a” y el registro de bases ofrecen, a su vez, un campo amplio de expansión funcional. Actualmente, la capacidad de guardar y recuperar poses permite definir posiciones de referencia que, a futuro, pueden vincularse con el eje lineal externo, con posiciones específicas de cambio de herramienta, poses seguras, posiciones de exhibición o puntos de captura de imágenes para sistemas de visión o escaneo. Complementariamente, la introducción de un control fino de movimiento de ejes ajustando el valor individual de cada articulación mediante comandos de voz

permitiría generar nuevas poses en tiempo real, afinar aproximaciones y hacer aún más interactiva la exploración del espacio de trabajo por parte del operario. Por otra parte, los movimientos direccionales, hoy orientados a aproximar el robot a la pieza, pueden especializarse hacia rutinas de entrada y salida segura, procedimientos estandarizados de cambio de herramienta o incluso secuencias de posicionamiento para montaje, donde el robot se desplaza a “zonas seguras” predefinidas antes de interactuar con el entorno. En paralelo, el registro de la base, si bien responde inicialmente a la necesidad de mantener la calibración y precisión del sistema, puede evolucionar hacia la definición de múltiples espacios de trabajo dentro de *Grasshopper*, asociados a distintos dispositivos o fases del proceso, almacenados como referencias geométricas modificables según ajustes menores en obra o en taller.

En cuanto a la fabricación propiamente tal, la programación de perforaciones puede complejizarse incorporando ángulos variables de perforación, distanciamientos asimétricos y patrones adaptativos que respondan a exigencias estructurales o a detalles específicos de uniones carpinteras. Sobre esta misma base, resulta factible parametrizar tipologías de uniones de carpintería de madera (por ejemplo, cajas y espigas, empalmes, alojamientos para herrajes) y dotar al sistema STT-*chatbot* de un “diccionario de mecanizados”, mediante el cual el robot solicite la información mínima necesaria al operario (dimensiones, profundidades, orientación de la pieza, tipo de unión) para generar automáticamente el programa correspondiente. El hecho de haber demostrado el uso de la herramienta por voz para tareas de calibración y perforación indica que esta misma lógica puede extenderse a otras operaciones y herramientas (fresadoras, sierras, herramientas de marcado), e incluso a otros materiales, siempre que se ajusten los parámetros de proceso y la configuración del sistema robótico. Una línea de desarrollo particularmente relevante es la integración de sistemas de visión artificial y de inteligencia artificial para la asistencia en seguridad y toma de decisiones: por ejemplo, detección de colisiones y zonas de riesgo antes de aproximar, medición de profundidades de corte o perforación, verificación de la posición real de la pieza frente al modelo digital y generación de advertencias a través de la

retroalimentación TTS, incorporando mecanismos de razonamiento sobre las acciones a ejecutar.

Todas estas proyecciones se encuentran dentro del rango de desarrollo posible a partir de los resultados obtenidos en esta memoria; sin embargo, no han sido implementadas debido a la restricción temporal propia del proceso de titulación, acotado a un periodo de dos semestres académicos. En ese marco, la investigación se concentró en asegurar la robustez del flujo de trabajo de calibración y en validar experimentalmente la viabilidad del control por voz en una tarea representativa de fabricación en madera, priorizando la consolidación de un núcleo funcional sólido por sobre la incorporación de un conjunto amplio de extensiones. No obstante, el sistema actual deja instalada una base técnica y metodológica suficientemente estructurada para que trabajos futuros, ya sea en el ámbito académico o profesional, pueda complejizar el sistema, incorporando nuevas herramientas, funciones avanzadas de programación por voz, integración con visión artificial y despliegue en escenarios de prefabricación y obra a escala real.

4. REFERENCIAS

- Alpha Cephei Inc. Vosk Small Model – Lightweight Speech Recognition Model. [En línea]. 2024. Disponible en: <https://alphacephei.com/vosk/models> [Consulta: 16-jun-2025].
- Alpha Cephei Inc. Vosk Speech Recognition Toolkit. [En línea]. 2024. Disponible en: <https://alphacephei.com/vosk> [Consulta: 16-jun-2025]
- Association for Robots in Architecture. KUKA|prc – Parametric Robot Control. [En línea]. Viena, 2024. Disponible en: <https://www.robotsinarchitecture.org/kuka-prc> [Consulta: 14-abr-2025].
- Bamani, E. B., Nissinman, E., Atari, R., Saadon, N. H., & Sintov, A. "Speech-to-Trajectory: Learning Human-Like Verbal Guidance for Robot Motion". IEEE Robotics and Automation Letters, 2025. [En línea]. Disponible en: <https://arxiv.org/abs/2504.05084> [Consulta: 4 ago 2025].
- Bingol, M. C., & Aydogmus, O. "Performing predefined tasks using the human–robot interaction on speech recognition for an industrial robot". Engineering Applications of Artificial Intelligence, vol. 95, pp. 103903, 2020. [En línea]. Disponible en: <https://doi.org/10.1016/j.engappai.2020.103903> [Consulta: 4 ago 2025].
- Google LLC. Google Speech Recognition API. [En línea]. Mountain View, CA: Google, 2024. Disponible en: <https://cloud.google.com/speech-to-text> [Consulta: 16-abr-2025].
- IAAC – Institute for Advanced Architecture of Catalonia. BLABBER BOT. [En línea]. Barcelona: IAAC, 2024. Disponible en: <https://blog.iaac.net/blabber-bot/> [Consulta: 03-nov-2025].
- Kaczmarek, W., Panasiuk, J., Borys, S., & Banach, P. "Industrial robot control by means of gestures and voice commands in off-line and on-line mode". Sensors, vol. 20, no. 21, pp. 6358, 2020. [En línea]. Disponible en: <https://doi.org/10.3390/s20216358> [Consulta: 4 ago 2025].

- KARPAGAVALLI, S. y CHANDRA, E. "A Review on Automatic Speech Recognition Architecture and Approaches," International Journal of Signal Processing, Image Processing and Pattern Recognition, vol. 9, n.º 4, pp. 393–404, 2016
- KUKA. Cooperación hombre-robot. [En línea]. KUKA AG, 2024. Disponible en: <https://www.kuka.com/es-es/la-produccion-del-futuro/cooperacion-hombre-robot> [Consulta: 20-nov-2025]
- Landázuri Alulema, V. Timber Structures for Automation and Human-Robot Collaboration. Construction Robotics, Research Paper, RWTH Aachen University, 2022. [En línea]. Disponible en: <https://publications.rwth-aachen.de/record/850812/files/850812.pdf>
- Microsoft Corporation. Azure Cognitive Services – Speech to Text. [En línea]. Microsoft Azure, 2024. Disponible en: <https://azure.microsoft.com/services/cognitive-services/speech-to-text/> [Consulta: 20-jun-2025].
- Microsoft Corporation. Visual Studio Code. [En línea]. 2024. Disponible en: <https://code.visualstudio.com> [Consulta: 16-may-2025].
- OpenAI. Whisper – Automatic Speech Recognition System. [En línea]. OpenAI, 2024. Disponible en: <https://github.com/openai/whisper> [Consulta: 20-jun-2025].
- Park, S., Wang, X., Menassa, C. C., Kamat, V. R., & Chai, J. Y. "Natural language instructions for intuitive human interaction with robotic assistants in field construction work". Automation in Construction, vol. 161, pp. 105345, 2024. [En línea]. Disponible en: <https://doi.org/10.1016/j.autcon.2024.105345> [Consulta: 4 ago 2025].
- PortAudio / PyAudio Development Team. PyAudio. [En línea]. 2024. Disponible en: <https://people.csail.mit.edu/hubert/pyaudio/> [Consulta: 14-abr-2025].
- Python Software Foundation. Python Programming Language, Version 3.x. [En línea]. 2024. Disponible en: <https://www.python.org> [Consulta: 14-abr-2025].

- PyTTsX3 Contributors. Pytttsx3 – Text-to-Speech Conversion Library. [En línea]. 2024. Disponible en: <https://pytttsx3.readthedocs.io/> [Consulta: 16-abr-2025].
- Robert McNeel & Associates. Grasshopper. [En línea]. Seattle, WA: Robert McNeel & Associates, 2024. Disponible en: <https://www.grasshopper3d.com> [Consulta: 14-abr-2025].
- Robert McNeel & Associates. Rhinoceros 8. [En línea]. Seattle, WA: Robert McNeel & Associates, 2024. Disponible en: <https://www.rhino3d.com> [Consulta: 14-abr-2025].
- RoboDK Inc. RoboDK – Robot Simulation and Offline Programming. [En línea]. Quebec, Canadá, 2024. Disponible en: <https://robodk.com> [Consulta: 22-abr-2025].
- Tasneem, O., y Pieters, R. Human–robot collaborative visual inspection with Large Language Models. *Robotics and Computer-Integrated Manufacturing*, vol. 98, 2026, art. 103154. Disponible en: <https://doi.org/10.1016/j.rcim.2025.103154>

5. ANEXOS

ANEXO A- MANUAL DE MODELADO ROBODK

https://usmcl-my.sharepoint.com/:b:/g/personal/tamara_perez_usm_cl/IQB8BvsLaZ8OT6NxaeSsorTYAQmflNIDg48ct89wfuvkDZo?e=ifkOzU

ANEXO B- EXPERIMENTO

https://usmcl-my.sharepoint.com/:f:/g/personal/tamara_perez_usm_cl/IgCe-YMc6ng-SLKNocS77LGSAQsz7DuAIQ4p8QX4tkNhYIk?e=i5UDhl

ANEXO C- VOICE-CONTROLLED WORKFLOW FOR HUMAN-ROBOT COLLABORATIVE TIMBER JOINERY

https://usmcl-my.sharepoint.com/:b:/g/personal/tamara_perez_usm_cl/IQDqclugOhrQRqCgr7nCNMvAZITaUuaW84T2YfnibVK7zl?e=bsNOu3