Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Estación robótica musical con Multi-Robots ABB controlada desde Matlab

Autor:

Sánchez Calvo, Juana Li

Tutor:

Herreros López, Alberto

Departamento de Ingeniería de Sistemas y Automática

Valladolid, octubre 2025

RESUMEN

La robótica industrial se ha convertido en un pilar de la automatización gracias a su precisión, rapidez y capacidad de integración. Este Trabajo de Fin de Grado presenta la simulación de una estación robótica en RobotStudio, compuesta por dos ABB IRB 120 y un IRB 14000 YuMi. Los IRB 120 ejecutan secuencias musicales en un teclado de piano programadas en RAPID, mientras el IRB 14000 acompaña con movimientos sincronizados. La comunicación entre controladores se gestiona mediante Socket Messaging, y el control externo se realiza a través de una HMI en MATLAB conectada mediante OPC UA. El proyecto tiene un enfoque educativo, al integrar programación, coordinación multi-robot y diseño de interfaces, mostrando además una aplicación innovadora al vincular la robótica con la música.

Palabras clave: RobotStudio, RAPID, OPC UA, Socket Messaging, IRB.

ABSTRACT

Industrial robotics has become a cornerstone of automation thanks to its precision, speed, and integration capabilities. This Final Degree Project presents the simulation of a robotic workstation in RobotStudio, consisting of three ABB IRB 120 and an IRB 14000 YuMi. The IRB 120 units perform musical sequences on a piano keyboard programmed in RAPID, while the IRB 14000 accompanies with synchronized movements. Communication between controllers is managed through Socket Messaging, and external control is carried out via a MATLAB HMI connected using OPC UA. The project has an educational purpose, combining programming, multi-robot coordination, and interface design, while exploring an innovative application by linking robotics with music.

Keywords: RobotStudio, RAPID, OPC UA, Socket Messaging, IRB.

ÍNDICE DE CONTENIDO

1. INTRODUCCIÓN Y OBJETIVOS 16
   1.1 INTRODUCCION 16
   1.2 JUSTIFICACIÓN DEL PROYECTO 16
   1.3 OBJETIVOS 17
   1.4 ESTRUCTURA DE LA MEMORIA 18
2. MARCO TEÓRICO Y ESTADO DEL ARTE 21
   2.1 MARCO TEÓRICO 21
      2.1.1 Componentes básicos 25
         2.1.1.1 Controlador 27
         2.1.1.2 Esqueleto del robot 27
         2.1.1.3 Actuadores 27
         2.1.1.4 Sensores 28
         2.1.1.5 Manipulador 29
         2.1.1.6 Sistema de control 29
      2.1.2 Clasificación de los robots 30
   2.2 ESTADO DEL ARTE 33
3. METODOLOGÍA Y SOFTWARE UTILIZADO 35
   3.1 RobotStudio 35
      3.1.1 Propósito y Aplicaciones 35
      3.1.2 Características Principales 36
      3.1.3 Versiones y Compatibilidad 37
      3.1.4 Beneficios del Uso de RobotStudio 37
      3.1.5 Limitaciones y Desafíos 38
   3.2 Matlab 38
      3.2.1 Propósito y Aplicaciones 39
      3.2.2 Características Principales 40
      3.2.3 Versiones y Compatibilidad 41
      3.2.4 Beneficios del Uso de MATLAB 41
      3.2.5 Limitaciones y Desafíos 42
      3.2.6 Conclusión 42

3.3 ABB IRC5 OPC 43
3.3.1 Propósito y Aplicaciones 43
3.3.2 Características Principales 44
3.3.3 Versiones y Compatibilidad 44
3.3.4 Beneficios del Uso de ABB IRC5 OPC Configuration 45
3.3.5 Limitaciones y Desafíos 46
4. DESARROLLO DEL PROYECTO 48
4.1 PRIMERA FASE: RobotStudio 48
4.1.1 CONFIGURACIÓN Y MODELADO DE LA ESTACIÓN 49
4.1.1.1 Modelado de la estación 49
4.1.1.2 Controladores virtuales 58
4.1.1.3 Componentes Inteligentes 65
Sensor de colisión 66
Sensor plano 66
Linear Move 67
Logic Gate 69
Play Sound 69
Set Color 70
Pose Mover 78
4.1.1.4 Lógica de la estación 80
4.1.2 Trayectorias y puntos 82
Workobject 82
Robtarget 85
Joint tarjets 86
Path 88
4.1.3 RAPID 90
4.1.3.1 Importación de los datos 91
4.1.3.2 Generación de módulos 92
4.1.3.3 Definición de variables 94
Variables globales 94
Variables locales 94
Variables Constantes 95
Variables Normales 96
4.1.3.4 Creación de las interrupciones 96

4.1.3.5 Creación de las distintas funciones 101
Num Rand 101
Notas Aleatorias 102
Leer Partitura 103
Tocar Nota 106
Tocas notas Ascendentemente 107
Tocar Notas Descendentemente 108
Bailar Nota 109
Bailar Normal 110
Leer Nota de IRB120 110
4.1.3.6 Diagrama de flujo de las funciones main 111
4.2 ABB IRC5 OPC CONFIGURATION 115
4.3 MATLAB 118
5. COMUNICACIONES 125
5.1 Módulos de RAPID 125
5.2 Controladores 126
5.3 RobotStudio y sistemas externos 126
6. SIMULACIÓN 129
7. RESULTADOS 138
8. CONCLUSIONES 141
Líneas de trabajo futuro 143
9. Bibliografía 145
Referencias 145

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Robot IRB 1200 17
Ilustración 2: Robot IRB 1400 YuMi Dual Arm 18
Ilustración 3: La Paloma de Arquitas: primer autómata mecánico documentado (s. IV a.C) 21
Ilustración 4: La eolípila de Herón de Alejandría: primera máquina de vapor (10-70 a.C) 22
Ilustración 5: Torre de reloj astronómica de Su Sung (1088) 22
Ilustración 6: Autómatas musicales programables de Al-Jazari (1206) 23
Ilustración 7: Smart Art Componentes de un robot 26
Ilustración 8: Logo RobotStudio ABB 35
Ilustración 9: Beneficio del uso de RobotStudio según ABB (ABB, 2025) 38
Ilustración 10: Logo Matlab 39
Ilustración 11: Página inicio RobotStudio 49
Ilustración 12: Robots articulados de la biblioteca de ABB 50
Ilustración 13: Fijar posición de un robot en RobotStudio 50
Ilustración 14: Robots IRB 1200 duplicado en estación 51
Ilustración 15: Robots articulados de la biblioteca de ABB 51
Ilustración 16: Robot IRB 1400 YuMi en la estación de modelado 51
Ilustración 17: Creación de un tetraedro "Nota Blanca" 52
Ilustración 18: Creación de un tetraedro "Nota Negra" 52
Ilustración 19: Primera escala del teclado de nuestra estación de RobotStudio 53
Ilustración 20: Teclado en la estación de modelado de RobotStudio 53
Ilustración 21: Creación base de la pinza inteligente 54
Ilustración 22: Creación brida de la pinza inteligente 54
Ilustración 23: Posición fija de una pieza en RobotStudio 54
Ilustración 24: Definición de posiciones en RobotStudio 55
Ilustración 25: Cuerpo de nuestra pinza inteligente 55
Ilustración 26: Creación de unión de dos cuerpos en RobotStudio 56
Ilustración 27: Unión del cuerpo de nuestra pinza inteligente 56
Ilustración 28: Selección de color en RobotStudio 56
Ilustración 29: Colores básicos para nuestro cuerpo de la pinza inteligente 57
Ilustración 30: Modelado de nuestra Pinza Inteligente en RobotStudio 57
Ilustración 31: Estación modelada en RobotStudio 58
Ilustración 32: Creación de un controlador virtual en RobotStudio 59
Ilustración 33: Características de la creación de un nuevo controlador virtual en RobotStudio 59
Ilustración 34: Configuración de un controlador virtual para dos IRB 1200 en RobotStudio 61
Ilustración 35: Señales de entrada y salida del controlador IRB 1200 de RobotStudio 61
Ilustración 36: Configuración de un controlador virtual para un IRB 1400 en RobotStudio 64
Ilustración 37: Señales de entrada y salida del controlador IRB 1200 de RobotStudio 64
Ilustración 39: Propiedades del componente sensor de colisión de RobotStudio 66
Ilustración 40: Propiedades del componente inteligente sensor plano de RobotStudio 67
Ilustración 41: Propiedades del componente movimiento linear de RobotStudio 68
Ilustración 42: Propiedades del componente movimiento linear de RobotStudio 68
Ilustración 43: Propiedades del componente inteligente puerta lógica NOT de RobotStudio ... 69

Ilustración 44: Propiedades del componente inteligente reproducir sonido de RobotStudio ... 70
Ilustración 45: Ajuste de propiedades del componente inteligente reproducir sonido de RobotStudio ... 70
Ilustración 46: Propiedades del componente inteligente configurar color de RobotStudio ... 71
Ilustración 47: Lógica del componente inteligente nota do del teclado del RobotStudio ... 71
Ilustración 48: Estructuración y componentes del piano de RobotStudio ... 72
Ilustración 49: Creación de un mecanismo dentro de la ventana de modelado de RobotStudio ... 72
Ilustración 50: Creación de un mecanismo en RobotStudio ... 73
Ilustración 51: SmartArt de las partes que conforman un mecanismo en RobotStudio ... 73
Ilustración 52: Creación de un eslabón de un mecanismo en RobotStudio ... 73
Ilustración 53: Propiedades para la definición de un eslabón en RobotStudio ... 74
Ilustración 54: Eslabones que conforman una pinza inteligente en RobotStudio ... 74
Ilustración 55: Eje 1 del mecanismo pinza de RobotStudio ... 74
Ilustración 56: Eje 2 del mecanismo pinza de RobotStudio ... 75
Ilustración 57: Ejes del mecanismo pinza de RobotStudio ... 75
Ilustración 58: Propiedades de los datos de herramienta del mecanismo pinza de RobotStudio ... 76
Ilustración 59: Compilación mecanismo pinza de RobotStudio ... 76
Ilustración 60: Herramienta pinza de RobotStudio ... 77
Ilustración 61: Creación Pose de la posición pinza abierta de RobotStudio ... 77
Ilustración 62: Composición pinza de RobotStudio ... 77
Ilustración 63: PoserMovers del mecanismo pinza de RobotStudio ... 78
Ilustración 64: LogicSRLatch del mecanismo pinza de RobotStudio ... 78
Ilustración 65: Puerta lógica NOT del mecanismo pinza en RobotStudio ... 79
Ilustración 66: DigitalOutputs del mecanismo pinza de RobotStudio ... 79
Ilustración 68: Entrada digital pulsador del mecanismo pinza de RobotStudio ... 80
Ilustración 69: Diseño y lógica del mecanismo pinza de RobotStudio ... 80
Ilustración 70: Diseño de la lógica de la estación de modelado de RobotStudio ... 82
Ilustración 71: Creación de un objeto de trabajo en RobotStudio ... 83
Ilustración 72: Workobjects de la estación de modelado de RobotStudio ... 84
Ilustración 73: Definición del Workobject Teclado del módulo ModulePiano de RAPID ... 84
Ilustración 74: Robtargets del objeto teclado de la estación de RobotStudio ... 85
Ilustración 75: Definición de Robtargets del módulo ModulePiano de RAPID ... 85
Ilustración 76: Robot IRB 1200 en posición del robtarget ... 86
Ilustración 77: Robot 1400 YuMi en posición del jointtarget ... 87
Ilustración 78: Definición de los jointargets de nuestro módulo DanceL de RAPID ... 87
Ilustración 79: Path mostrado en nuestra estación de RobotStudio ... 89
Ilustración 80: Formas de sincronización con estación y RAPID en RobotStudio ... 91
Ilustración 81: Desglose de los módulos de RAPID de nuestro Controlador3 ... 93
Ilustración 82: Desglose de los módulos de RAPID de nuestro Controlador_Robots_IRB120 ... 93
Ilustración 83: Variables globales empleadas en el módulo ModulePiano del Controlador_Robots_IRB120 de RAPID ... 94
Ilustración 84: Variable Local del módulo ModulePiano del Controlador_Robots_IRB120 de RAPID ... 95

Ilustración 85: Variables CONST del módulo ModulePiano del Controlador_Robots_IRB120 de RAPID 95
Ilustración 86: Variables no persistentes del módulo ModulePiano del Controlador_Robots_IRB120 de RAPID 96
Ilustración 87: Función Init del módulo ModulePiano del Controlador_Robots_IRB120 de RAPID 97
Ilustración 88: Rutina TRAP de la nota DO del módulo ModulePiano del Controlador_Robots_IRB120 de RAPID 98
Ilustración 89: Interrupción TRAP para activación/desactivación del robot YuMi 98
Ilustración 90: Interrupción TRAP para tocar una o dos notas en el teclado a través de los modos de la pinza herramienta 99
Ilustración 91: Interrupciones TRAP de la activación/desactivación de los robots IRB120 99
Ilustración 92: Interrupción TRAP de activación del modo Escala Ascendente 100
Ilustración 93: Interrupción TRAP de activación del modo Escala Descendente 100
Ilustración 94: Interrupción TRAP de activación del modo Leer Partitura 100
Ilustración 95: Interrupción TRAP de activación del modo Aleatorio 100
Ilustración 96: Interrupción TRAP de finalización del programa 101
Ilustración 97: Función num rand para la generación de un numero aleatorio 102
Ilustración 98: Función de RAPID NotasAleatorias 102
Ilustración 99: Fichero partitura Sonrisas y Lágrimas.txt 103
Ilustración 100: Función de RAPID Leer Partitura 105
Ilustración 101: Relación entre la nomenclatura de las notas de un piano 105
Ilustración 102: Función de RAPID Tocar Nota 106
Ilustración 103: Función de RAPID Tocar Notas Ascendentes 107
Ilustración 104: Función de RAPID Tocar notas ascendentes en intervalos 107
Ilustración 105: Función de RAPID Tocar Notas Descendentes 108
Ilustración 106: Función Tocar Notas descendentes en Intervalos 108
Ilustración 107: Función de RAPID Bailar Nota 109
Ilustración 108: Función de RAPID Bailar Normal 110
Ilustración 109: Función de RAPID Leer nota de IRB120 111

Ilustración 110: Diagrama de flujo de la función main de ModulePiano de RAPID 112
Ilustración 111: Diagrama de flujo de la función main de DanceL de RAPID 113
Ilustración 112: Diagrama de flujo de la relación entre los módulos 114
Ilustración 113: Interfaz inicial del programa ABB IRC5 OPC Configuration 116
Ilustración 114: Creación de un nuevo alias para un controlador 116
Ilustración 115: Dispositivos escaneados por ABB IRC5 OPC Configuration 117
Ilustración 116: Creación de un controlador y sus criterios de conexión en ABB IRC5 OPC Configuration 117
Ilustración 117: Activación y desactivación del Server Control de los dispositivos de ABB IRC5 OPC Configuration 117
Ilustración 118: APPs de Matlab 118
Ilustración 119: Interfaz OPC Data Explorer de Matlab 119
Ilustración 120: Definición del hostname de un host en OPC Data Explorer 119
Ilustración 121: Creación de un cliente en OPC Data explorer de Matlab 120

Ilustración 122: Conexión al servidor de OPC 120
Ilustración 123: Creación de un grupo de variables en OPC Data Explorer de Matlab 121
Ilustración 124: Añadir Items en un grupo del localhost en OPC Data Explorer de Matlab 121
Ilustración 125: Items disponibles de nuestros controladores 122
Ilustración 126: Explorador de variables del Server Control a través de OPC Data Explorer de Matlab..... 122
Ilustración 127: PIANO DASHBOARD..... 124
Ilustración 128: Ruta para la creación de un estado inicial en la estación de RobotStudio 130
Ilustración 129: Configuración final del estado inicial POS_INIT de nuestra estación de RobotStudio..... 130
Ilustración 130: Estación de RobotStudio en modo escala ascendente con un IRB 120 activado en movimiento..... 131
Ilustración 131: Estación de RobotStudio en modo escala ascendente con dos IRB 120 activados en movimiento..... 131
Ilustración 132: Estación de RobotStudio en modo escala ascendente con dos IRB 120 activados en movimiento y el YuMi 1400 bailando (1) 132
Ilustración 133 : Estación de RobotStudio en modo escala ascendente con dos IRB 120 activados en movimiento y el YuMi 1400 bailando (2) 132
Ilustración 134 : Estación de RobotStudio en modo escala ascendente con dos IRB 120 activados en movimiento y el YuMi 1400 bailando (3) 133
Ilustración 135: Start del programa de la comunicación de comunicaciones OPC UA de ABB. 134
Ilustración 136: Lectura/Escritura de señales y variables de RobotStudio a través de ABB IRC5 OPC UA Configuration (1)..... 135
Ilustración 137: Lectura/Escritura de señales y variables de RobotStudio a través de ABB IRC5 OPC UA Configuration (2)..... 135
Ilustración 138: Ejecución de la pantalla HMi diseñada desde App Designer de Matlab 136
Ilustración 139: Representación real de la ejecución de la función escala ascendente con un robot IRB 120 activado y el YuMi 1400 de la estación a través de la activación de señales desde el HMI diseñado en Matlab (1) 136
Ilustración 140: Representación real de la ejecución de la función escala ascendente con dos robots IRB 120 activados y el YuMi 1400 de la estación a través de la activación de señales desde el HMI diseñado en Matlab (2) 137
Ilustración 141: Representación real de la ejecución de la función escala ascendente con un robot IRB 120 activado y el YuMi 1400 de la estación a través de la activación de señales desde el HMI diseñado en Matlab (3) 137

ÍNDICE DE TABLAS

Tabla 1: Smart Art de las leyes de la robótica 24
Tabla 2: Smart Art de hechos relevantes en la robótica cronológicamente 25
Tabla 3: Categorías principales de actuadores 28
Tabla 4: Tipos de Sensores 29
Tabla 5: Tipos de programación de los sistemas de control 29
Tabla 6: Generaciones de robots según su cronología 31
Tabla 7: Clasificación de robots según propósito 32
Tabla 8: I/O del controlador IRB 1200 62
Tabla 9: I/O del controlador IRB 1400 YuMi 65
Tabla 10: SmartArt de las funciones de los componentes inteligentes 65

Capítulo 1

1. INTRODUCCIÓN Y OBJETIVOS

1.1 INTRODUCCION

La robótica ha experimentado con los años una gran evolución, transformándose en una disciplina esencial en la ingeniería y la industria moderna. Los grandes avances se han impulsado en la electrónica, la informática, el control automático, permitiendo un desarrollo donde los robots son cada vez más sofisticados y eficientes.

Si nos adentramos en la robótica industrial podemos decir que ha tenido un impacto en la manufactura y producción, siendo capaz de optimizar procesos, mejorar la calidad de productos obteniendo unos costes operativos más reducidos. Robots como el IRB 120 con su capacidad para realizar movimientos más precisos y rápidos en espacios reducidos, son un ejemplo perfecto de esta transformación tecnológica.

El proyecto que vamos a realizar a continuación consiste en el desarrollo de una estación de trabajo virtual utilizando el software de RobotStudio, que consistirá en una programación y simulación de un robot industrial IRB 120 de ABB capaz de tocar un instrumento musical, concretamente. Acompañado de un IRB 1400 YuMi, que realizará un baile y será controlado con un controlador independiente. Las órdenes recibidas por ambos serán enviadas a través de una interfaz diseñada con MATLAB y conectada a través de OPC UA con las señales digitales y variables necesarias del lenguaje propio de RobotStudio, RAPID.

1.2 JUSTIFICACIÓN DEL PROYECTO

La justificación del proyecto radica en varios aspectos tanto técnicos como educativos, que prueban la relevancia de desarrollar estaciones de trabajo virtual en un entorno educativo que permita tanto a estudiantes como a profesionales adquirir conocimientos prácticos y avanzados en robótica y en programación de robots industriales.

La simulación de una tarea como es tocar un piano fomenta la innovación y la creatividad, promoviendo nuevas aplicaciones y soluciones dentro del campo de la robótica, vinculado a el área musical.

La programación y simulación del robot IRB 120 requiere de una comprensión de los principios de la robótica, así como del control automático. Conocimientos que hemos ido adquiriendo durante el estudio de asignaturas tales como Sistemas robotizados, donde abordamos cinemática, dinámica y algoritmos de control de robots, así como su implementación en esto entornos simulados y reales.

Este proyecto permite a los estudiantes aplicar sus conocimientos en la integración de sistemas electrónicos y de automatización, complementando con la tarea de tocar un piano que requiere de la implementación de sensores en el entorno a diseñar. También refleja la innovación y creatividad fomentados para emplear un robot capaz de tocar un instrumento musical.

En conclusión, este proyecto no solo valida y aplica los conocimientos teóricos y prácticos adquiridos en la carrera de ingeniería electrónica y automática industrial, sino que también proporciona una plataforma para la innovación, el análisis y la solución de problemas complejos.

1.3 OBJETIVOS

El objetivo principal de este proyecto será diseñar un programa en RAPID capaz de ejecutar una secuencia de movimientos que le permitan al robot tocar un instrumento musical, en concreto un teclado de piano, pudiendo demostrar así los conocimientos adquiridos durante el paso por el grado de Ingeniería de Electrónica y Automatización Industrial en la Escuela de Ingenieros Industriales de la Universidad de Valladolid.

Para ello vamos a crear una estación virtual en RobotStudio, donde emplearemos tres robots:

- Dos IRB 120, ambos controlados por un mismo controlador virtual (*Ilustración 1*).

Ilustración 1: Robot IRB 1200

- Un Robot IRB14000 YuMi (*Ilustración 2*).

Ilustración 2: Robot IRB 1400 YuMi Dual Arm

Con todos estos elementos seremos capaces de simular una estación donde los robots IRB1200 se encargarán de tocar distintas escalas en el teclado, mientras el IRB14000 YuMi le acompañará con movimientos al ritmo de las notas y velocidades determinadas previamente para así darle al proyecto un entorno musical y de aprendizaje para aquellos que los deseen.

1.4 ESTRUCTURA DE LA MEMORIA

El proyecto se ha organizado en seis capítulos, que son los que se presentan a continuación:

Capítulo 1

Introducción y objetivos.

En este capítulo se detallan los objetivos principales del proyecto y se justifica la importancia de su realización.

Capítulo 2

Marco teórico y estado del arte

Este capítulo recoge información relevante sobre la robótica, proporcionando un marco de referencia que permite comprender el contexto actual del tema tratado.

Capítulo 3

Metodología y softwares empleados

Aquí se describen las diferentes áreas de la ingeniería involucradas en el desarrollo del proyecto y se explican las herramientas de software utilizadas de cada una.

Capítulo 4

Desarrollo del proyecto

Este capítulo detalla las distintas fases por las que ha pasado el proyecto durante su ejecución.

Capítulo 5

Simulación y Resultados

Se presentan los resultados obtenidos, incluyendo la simulación realizada en RobotStudio y la interfaz creada a través de MATLAB.

Capítulo 6

Conclusiones y futuras líneas de trabajo

En este capítulo se evalúan los resultados alcanzados, se analiza el grado en que se cumplieron los objetivos y se sugieren posibles direcciones para trabajos futuros relacionados con el proyecto.

Capítulo 2

2. MARCO TEÓRICO Y ESTADO DEL ARTE

El marco teórico, así como el estado del arte de este trabajo se enfocará en un campo multidisciplinario como es la robótica, empezando por los conceptos fundamentales, teorías y desarrollos tecnológicos que nos han permitido evolucionar en los distintos aspectos de este ámbito. De esta manera seremos capaces de entender el contexto actual sobre el tema a tratar.

2.1 MARCO TEÓRICO

Un robot es una entidad virtual o mecánica artificial, un sistema electromecánico que, por su apariencia o sus movimientos es capaz de ofrecer una sensación de tener un propósito propio. La propia independencia de sus acciones son el motivo por el que son un estudio razonable.

La propia palabra puede referirse tanto a mecanismos físicos como a sistemas virtuales de software, aunque el segundo término suele aludirse a los segundos con el término de bots. (Wikipedia, 2024).

A continuación, les introduciremos algunos antecedentes en la historia:

- En el siglo IV antes de Cristo, el matemático griego Arquitas de Trento construyó un ave mecánica a la que llamó “La Paloma” (*Ilustración 3*) y que funcionaba con vapor.

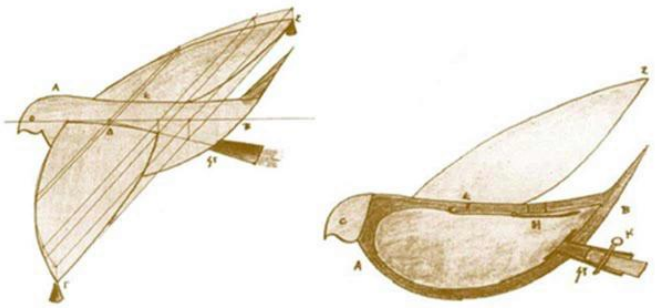


Ilustración 3: La Paloma de Arquitas: primer autómata mecánico documentado (s. IV a.C)

- En el año 10 – 70 antes de Cristo el ingeniero Herón de Alejandría creó numerosos dispositivos modificables por los usuarios y que las describió como máquinas accionadas a través de presión de aire, vapor y agua. Entre una de ellas se encontraba la eolípila (*Ilustración 4*).

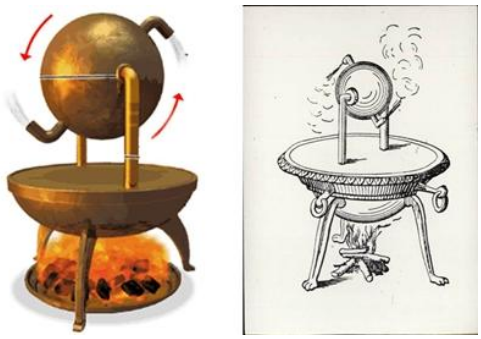


Ilustración 4: La eolípila de Herón de Alejandría: primera máquina de vapor (10-70 a.C)

- En 1088, el chino Su Sung levantó la torre de reloj (*Ilustración 5*) formado por figuras mecánicas que tocaban las campanadas de las horas.

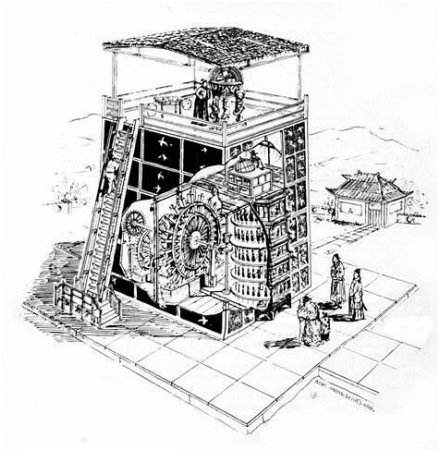


Ilustración 5: Torre de reloj astronómica de Su Sung (1088)

- En 1136-1206 el musulmán Artuqid fue capaz de diseñar y construir una serie de máquinas automatizadas, donde nos encontramos desde autómatas musicales que funcionaban con agua hasta útiles de cocina.

  En 1206 creó los primeros robots humanoides programables. La forma de estas máquinas se asemejaba al aspecto físico de cuatro músicos navegando en un bote sobre un lago. Dicho mecanismo contenía un tambor que era programable a través de unas clavijas que se chocaban con pequeñas palancas capaces de accionar los instrumentos de percusión. Gracias a la incorporación de las clavijas

en el mecanismo, el usuario, en este caso se trataba de dos tamborileros, era posible cambiar los ritmos y patrones del instrumento. La barca estaba formada por dos tamborileros, un arpista y un flautista (Jorge Elices, 2020) (*Ilustración 6*).

Ilustración 6: Autómatas musicales programables de Al-Jazari (1206)

La palabra "robótica" proviene del término "robot", que a su vez tiene su origen en la palabra checa "robota", que significa "trabajo forzado" o "servidumbre".

El término "robot" fue utilizado por primera vez en 1920 por el escritor checho Karel Capek en su obra de teatro "R.U.R" (Rossum´s Universal Robots). En la obra, los robots eran seres artificiales creados para realizar los trabajos de los humanos, pero durante el desarrollo de la obra, estos se relevan contra sus creadores.

El término "robótica" fue popularizada por el escritor de ciencia ficción Isaac Asimov en 1941. Asimov empleó el término en su cuento "Liar!" y *más* tarde en su obra "Runaround" en 1942, donde también introdujo las tres famosas Leyes de la Robótica definidas en la siguiente tabla (*Tabla 1*) . Desde entonces el término que se ha establecido como estándar para tratar del campo que estudia los robots, su diseño, construcción, operación y aplicación se ha convertido en "robótica" (Kak, 2011).

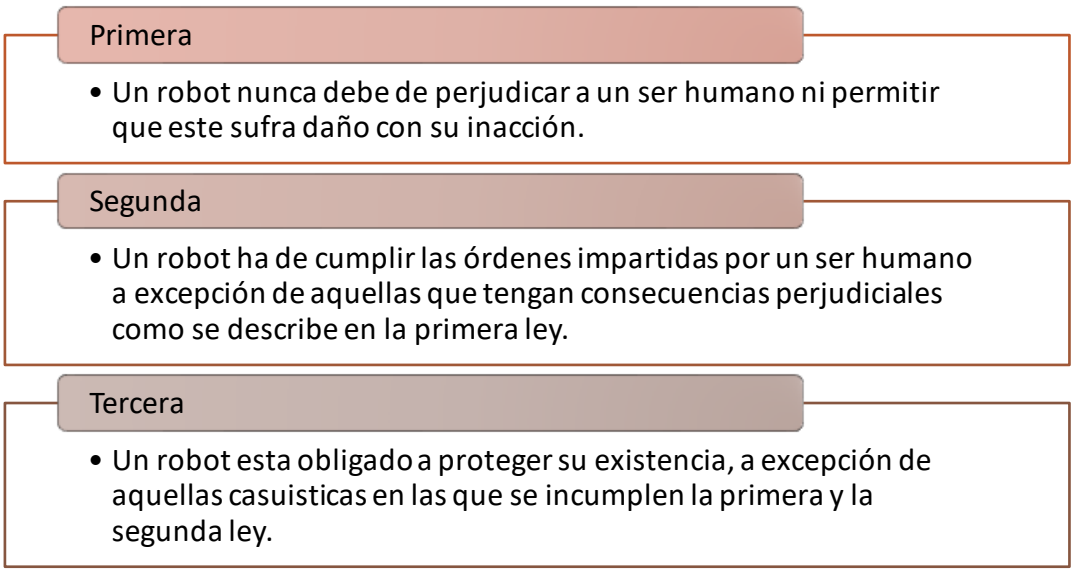


Tabla 1: Smart Art de las leyes de la robótica

Estos datos proporcionados previamente nos dan una base sólida para entender el origen del término, así como su evolución en el contexto histórico y literario.

La robótica moderna comienza en la década de 1950, aunque el concepto de autómatas ha existido desde la antigüedad, los hitos importantes en la evolución de la robótica incluyen los mostrados a continuación (*Tabla 2*):

Tabla 2: Smart Art de hechos relevantes en la robótica cronológicamente

2.1.1 Componentes básicos

La robótica es la rama de la ingeniería mecánica, electrónica y de las ciencias de la computación, que diseña, construya y opera robots (Robótica, 2023).

Un robot es una máquina o ingenio electrónico programable que es capaz de manipular objetos y realizar diversas operaciones. Gracias a la robótica los humanos somos capaces

de realizar diversas que nos permiten recopilar información, procesarla y tomar decisiones. Estos son los componentes (*Ilustración 7*) básicos que lo forman:

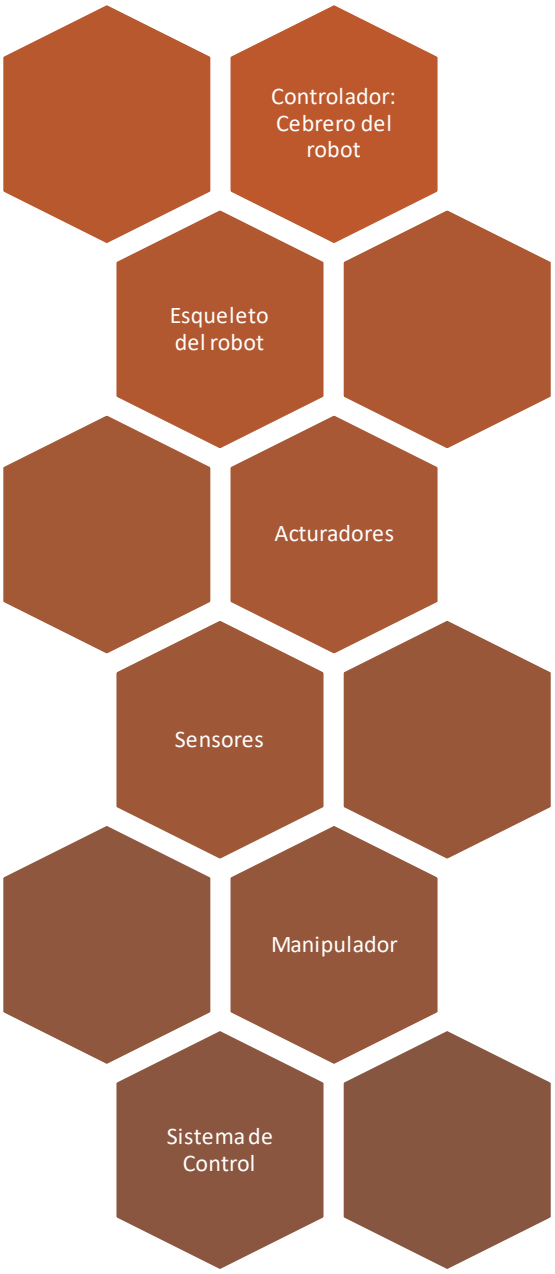


Ilustración 7: Smart Art Componentes de un robot

2.1.1.1 Controlador

El controlador es el núcleo central que gobierna todas las operaciones de un robot, incluyendo sus movimientos, cálculos y procesamiento de datos. Funciona a través de un microordenador equipado con una unidad central de procesamiento (CPU), memoria, sistemas de alimentación e interfaces que le permiten comunicarse con comando externos. Este sistema es esencialmente el "cerebro" del robot, coordinado y ejecutando las órdenes necesarias para que el robot realice sus tareas.

La función del controlador puede variar dependiendo de los parámetros que gestione. Por lo general, los controladores se clasifican en diferentes tipos, como controladores de posición, cinemáticos, dinámicos o adaptativos, cada uno especializado en gestionar aspectos específicos del movimiento y comportamiento del robot.

2.1.1.2 Esqueleto del robot

El esqueleto del robot, al igual que en el cuerpo humano, tiene la función crucial de sostener y dar estructura a todas las demás partes del sistema robótico. Su diseño debe estar adaptado a las necesidades específicas del robot, ya sea para proporcionar fuerza, velocidad, ligereza o maniobrabilidad.

La elección del material para el esqueleto es otro factor fundamental. Dependiendo de la función del robot, el esqueleto puede ser fabricado con materiales duros, pesados, flexibles, o una combinación de estos. Algunos robots pueden tener un esqueleto que sirve como base para sus componentes, mientras que otros pueden estar hechos de materiales más inusuales como cartón, madera, hierro o plástico, dependiendo de las necesidades de su diseño y aplicación.

2.1.1.3 Actuadores

Los actuadores son esenciales para los robots industriales, ya que proporcionan la fuerza necesaria para sus movimientos. Actúan como el "corazón" del robot, transformando señales eléctricas en acciones físicas que permiten a la máquina interactuar con su entorno.

Los actuadores se dividen en dos categorías principales:

Motores	Otros Actuadores
• Fundamentales para el funcionamiento del robot, permitiendo que se desplace, mueva sus brazos o manipule objetos a través de pinzas u otros mecanismos. Sin los motores, el robot sería incapaz de realizar cualquier movimiento físico.	Además de los motores, existen otros actuadores, que permiten al robot comunicarse y presentar información a su entorno. LCD, displays, altavoces, sincronizados de voz

Tabla 3: Categorías principales de actuadores

2.1.1.4 Sensores

Para que el robot funcione de manera autónoma y pueda interactuar eficazmente con su entorno, es indispensable que esté equipado con sensores. Estos dispositivos permiten al robot percibir su entorno y responder a él de manera adecuada, ajustando su comportamiento según las condiciones que detecta.

Los sensores varían en función de la tarea y el entorno del robot. Entre los más comunes se encuentran los sensores de luz, sonido, gravedad, temperatura, humedad, presión, velocidad, magnetismo y ubicación. También se utilizan sensores de proximidad, distancia, cámaras de video, y muchos otros, cada uno colocado estratégicamente para maximizar su efectividad en la función que debe cumplir.

La percepción es esencial para que los robots interactúen con su entorno. Los robots utilizan una variedad de sensores para recopilar datos sobre su entorno y su propio estado interno, lo que les permite tomar decisiones informadas.

Sensores de Proximidad y Visión	Sensores de Fuerza y Tacto	Sensores de Posición y Velocidad
• Permiten detectar obstáculos y reconocer objetos de entorno	• Utilizados en robots que requieren manipulación delicada o interacción precisa con objetos	• Proveen información crucial para el control de movimiento del robot

Tabla 4: Tipos de Sensores

2.1.1.5 Manipulador

El manipulador es la parte mecánica central del robot, compuesta por una serie de elementos sólidos o eslabones unidos mediante articulaciones que permiten el movimiento. Esta estructura, que se asemeja a un brazo humano con secciones con cuerpo brazo, muñeca y un actuador final, es crucial para la manipulación de objetos y la realización de tareas específicas.

2.1.1.6 Sistema de control

El sistema de control de un robot, integrado por software y hardware, es responsable de dirigir y coordinar sus movimientos. Este sistema puede ser programado para ejecutar tareas específicas de manera positiva o para adaptarse a cambios en el entorno mediante el uso de sensores, lo que permite al robot operar de forma autónoma.

Robots Pre-programados	Robots Autónomos
• Robots que siguen un conjunto de intruscciones fijas y realizan tareas repetitivas sin cambios	• Robots capaces demodificar su comportamiento en respuesta a la variaciones del entorno, utilizando datos obtenidos de sus sensores.

Tabla 5: Tipos de programación de los sistemas de control

Para que un robot funcione eficazmente, es crucial una coordinación precisa entre su esqueleto, sensores y actuadores. Además, el "cerebro" del robot, o su sistema de control, debe estar programado de manera que permita al robot cumplir con su función principal. Cada componente tiene un propósito específico y es esencial para el correcto funcionamiento de la máquina, haciendo que el robot sea una herramienta integral y altamente funcional.

2.1.2 Clasificación de los robots

Según la Asociación Francesa de Robótica Industrial – AFRI, fueron clasificados en generaciones según su cronología.

Primera Generación

- Robots manipuladores
- Repiten una o varias tareas de manera programada bajo un software, en secuencia.

Segunda Generación

- Robots en aprendizaje
- Aprenden los movimientos a realizar a través de los movimientos que ejecutan los operadores humanos.

Tercerca Generación

- Robots con sensores
- Son aquellos programables desde ordenadores, normalmente cuentan con sensores artificiales y otras piezas que permiten la visión y el tacto empleando lenguajes de programación

Cuarta Generación

- Robots móviles
- Aquellos que son capaces de tomar parte en diversos procesos gracias a la inteligencia artificial, también poseen sensores como la generación anterior pero se diferencian que pueden tomar decisiones y realizar mas movimientos

Quinta Generación

- Robots inteligentes
- Son aquellas máquinas dotadas de inteligencia artificial

Tabla 6: Generaciones de robots según su cronología

Los robots se pueden clasificar de diversas maneras, dependiendo del criterio utilizado. Una clasificación común es la siguiente:

	Robot Industriales Utilizados principalmente en la manufatura para tareas como ensamblaje, soldadura, pintura y manipulación de materiales. Son típicamente manipuladores con múltiples grados de libertad.
	Robots de Servicio Diseñados para interactuar con personas y ayudar en tareas diarias, como robots de limpieza o asistentes personales.
	Robots Móviles Incluyen vehículos autónomos y drones, que pueden desplazarse por el entorno. Estos robots son ampliamente utilizados en exploración, logística, y agricultura.
	Robots humanoides Imitan la forma y movimientos del cuerpo humano, con aplicaciones en la investigación de biónica, inteligencia artificial, y asistencia personal.
	Robots Colaborativos (Cobots) Trabajan junto a humanos en entornos compartidos, diseñados para ser seguros y fáciles de programar.

Tabla 7: Clasificación de robots según propósito

2.2 ESTADO DEL ARTE

Tras haber expuesto el marco teórico que sustenta este proyecto, en este apartado se recopilan algunos de los trabajos previos realizados en la Universidad de Valladolid que guardan relación con el ámbito de la robótica y la simulación en entornos industriales.

En el año 2015, Gonzalo Muinelo Garrido presentó un Trabajo de Fin de Grado centrado en la simulación de una célula robotizada destinada al tratamiento de piezas de aluminio. El propósito principal de su estudio fue desarrollar la programación del robot para gestionar de forma autónoma las piezas, las cuales debían atravesar tres procesos distintos, cada uno asociado a una máquina específica (Garrido, 2015).

Ese mismo año, Juan Antonio Ávila Herrero llevó a cabo el diseño de otra célula robótica, orientada en este caso al ámbito educativo. Su proyecto consistió en la creación de diversas prácticas formativas para la enseñanza del manejo del software RobotStudio, programando el robot con el fin de realizar tareas interactivas como jugar al tres en raya o escribir sobre una mesa inclinada. Este trabajo perseguía facilitar el aprendizaje de la programación y el control de robots industriales ABB (Herrero, 2015).

En 2019, Carlos Jiménez Jiménez desarrolló un sistema robótico educativo cuyo objetivo era permitir a un usuario jugar al ajedrez contra un robot industrial. El sistema incluía una interfaz gráfica que permitía al jugador introducir sus movimientos, mientras que el robot ejecutaba las jugadas desplazando las piezas sobre el tablero. Este proyecto integraba tanto la parte mecánica como la lógica de control y comunicación entre el robot y la interfaz, dando lugar a una experiencia interactiva y didáctica (Jiménez, 2019).

Finalmente, en 2022, Elena Pozas Mata desarrolló un proyecto en el que un robot YuMi interactuaba con un xilófono controlado mediante MATLAB, utilizando el protocolo OPC UA para la comunicación en tiempo real. Este trabajo destacó por la integración de la programación del robot en RAPID, la simulación en RobotStudio y el control externo a través de una interfaz HMI, combinando aspectos de coordinación multi-robot, ejecución musical e interacción educativa (Mata, 2022).

En conjunto, estos proyectos reflejan la evolución y el interés continuo de la Universidad de Valladolid en la integración de la robótica industrial, la simulación mediante RobotStudio y el desarrollo de aplicaciones educativas que favorecen el aprendizaje práctico en este campo.

Capítulo 3

3. METODOLOGÍA Y SOFTWARE UTILIZADO

En este proyecto, se ha seguido una metodología sistemática que abarca desde la conceptualización y diseño de un sistema robótico hasta la simulación, control y análisis de resultados. El enfoque metodológico se ha dividido en varias etapas clave, cada una respaldada por el uso de herramientas de software especializadas que permiten un desarrollo eficiente y preciso del trabajo. A continuación, se describen las principales etapas de la metodología y los tres softwares fundamentales utilizados: RobotStudio, MATLAB y ABB IRC5 OPC.

3.1 RobotStudio

RobotStudio es un software desarrollado por ABB Robotics, una de las principales empresas en el ámbito de la automatización industrial y robótica. Este entorno de simulación y programación offline es ampliamente utilizado en la industria para diseñar, simular, programar y optimizar sistemas robóticos antes de su implementación en un entorno real. RobotStudio es una herramienta poderosa que permite a los ingenieros y programadores trabajar en un entorno virtual, lo que reduce costos, minimiza riesgos y aumenta la eficiencia de los proyectos robóticos (ABB, 2025).

Ilustración 8: Logo RobotStudio ABB

3.1.1 Propósito y Aplicaciones

RobotStudio está diseñado para permitir a los usuarios crear modelos virtuales de células robóticas y simular su funcionamiento en un entorno tridimensional. Esta capacidad es fundamental en la fase de diseño, ya que permite identificar y resolver posibles problemas antes de que el sistema sea implementado físicamente. Además, la posibilidad de programar los

robots offline, es decir, sin interrumpir la producción en la planta, proporciona una gran ventaja en términos de eficiencia operativa.

Las aplicaciones de RobotStudio son diversas, abarcando desde la programación y simulación de robots industriales en fábricas de automóviles hasta su uso en líneas de producción de electrónica, ensamblaje, soldadura, pintura y manejo de materiales. Además, RobotStudio es utilizado en la formación y capacitación de ingenieros y operadores, dado que ofrece un entorno seguro y controlado para el aprendizaje.

3.1.2 Características Principales

RobotStudio incluye una amplia gama de características y herramientas que facilitan la creación, simulación y optimización de células robóticas. Entre las características más destacadas se encuentran:

- Simulación en 3D

  Permite crear modelos tridimensionales detallados de células de trabajo que replican fielmente el entorno de producción real, incluyendo robots, cintas transportadoras, herramientas y sensores. Esta capacidad es crucial para visualizar y optimizar la disposición y los movimientos dentro del espacio de trabajo.

- Programación Offline

  Una de las mayores ventajas de RobotStudio es la capacidad de programar robots sin detener la producción en la planta. Esto se traduce en una reducción significativa de tiempos de inactividad y una mejora en la productividad. La programación offline permite a los usuarios escribir, probar y depurar código en un entorno virtual antes de transferirlo al robot real.

- Virtual Commissioning

  Este módulo permite validar la lógica de control en un entorno virtual antes de implementarla físicamente. Es una herramienta vital para identificar y corregir errores de programación que podrían resultar costosos si no se detectan antes de la implementación.

- Path Optimization

  Esta herramienta optimiza las trayectorias de los robots, reduciendo el tiempo de ciclo y mejorando la eficiencia de los movimientos. El algoritmo de optimización ajusta las trayectorias para minimizar el tiempo de desplazamiento y evitar colisiones, lo que resulta en un sistema más rápido y seguro.

- Interfaz de Usuario Intuitiva

  RobotStudio cuenta con una interfaz gráfica de usuario (GUI) que es tanto potente como fácil de usar. La GUI permite a los usuarios arrastrar y soltar componentes en el espacio de trabajo virtual, facilitando la creación de modelos y la programación de los robots.

- Bibliotecas y Módulos Adicionales

  El software incluye una extensa biblioteca de componentes y módulos adicionales que permiten a los usuarios añadir robots, herramientas y equipos específicos a sus simulaciones. Además, ofrece soporte para la personalización y creación de componentes específicos según las necesidades del proyecto.

3.1.3 Versiones y Compatibilidad

La versión utilizada en este proyecto es RobotStudio 2023.1, una de las más recientes y avanzadas del software. Esta versión incorpora mejoras en la precisión de las simulaciones, nuevas herramientas de optimización y una interfaz de usuario mejorada que facilita aún más el uso del software.

RobotStudio es compatible con una amplia gama de controladores de robots ABB, incluyendo los controladores IRC5, que son uno de los más avanzados del mercado. Esta compatibilidad garantiza que las simulaciones y programaciones realizadas en RobotStudio se puedan transferir de manera efectiva a los robots reales sin problemas de integración.

3.1.4 Beneficios del Uso de RobotStudio

El uso de RobotStudio ofrece múltiples beneficios para los ingenieros y las empresas que implementan sistemas robóticos:

- Reducción de Costos

  Al permitir la programación y simulación offline, RobotStudio reduce los costos asociados con el tiempo de inactividad de la producción y los errores de programación.

- Mejora de la Eficiencia

  Las herramientas de optimización de trayectorias y simulación permiten a los usuarios mejorar la eficiencia operativa de los robots, reduciendo el tiempo de ciclo y aumentando la productividad.

- Reducción de Riesgos

  La capacidad de realizar pruebas y validaciones en un entorno virtual minimiza los riesgos asociados con la implementación de nuevos sistemas robóticos, lo que resulta en una transición más suave a la producción en vivo.

- Flexibilidad

  RobotStudio ofrece un alto grado de flexibilidad, permitiendo a los usuarios adaptar las simulaciones y programaciones a una amplia variedad de aplicaciones industriales.

- Formación y Capacitación

  El entorno seguro y controlado de RobotStudio es ideal para la formación de nuevos operadores y la capacitación continua de los ingenieros, permitiendo el desarrollo de habilidades sin riesgos para la producción real.

Ilustración 9: Beneficio del uso de RobotStudio según ABB (ABB, 2025)

3.1.5 Limitaciones y Desafíos

A pesar de sus numerosas ventajas, RobotStudio también presenta algunas limitaciones. Por ejemplo, aunque la simulación en 3D es altamente precisa, no siempre puede replicar con exactitud todos los aspectos de la física del mundo real, como la fricción o el desgaste de los materiales. Además, el software requiere una curva de aprendizaje, especialmente para aquellos que son nuevos en la programación de robots o en el uso de entornos de simulación avanzada.

3.2 Matlab

MATLAB (abreviatura de *Matrix Laboratory*) es un entorno de programación y una plataforma de cálculo numérico desarrollado por MathWorks. Es ampliamente utilizado en la ingeniería, la

ciencia y la economía para la resolución de problemas matemáticos, simulaciones, análisis de datos, visualización y desarrollo de algoritmos.

MATLAB es especialmente conocido por su capacidad de manipular matrices y realizar cálculos en gran escala con alta eficiencia y precisión, lo que lo convierte en una herramienta imprescindible en diversas áreas de investigación y desarrollo.

Ilustración 10: Logo Matlab

3.2.1 Propósito y Aplicaciones

MATLAB está diseñado para facilitar el trabajo con matrices, que son la base de muchas operaciones matemáticas en ingeniería y ciencia. Su principal propósito es proporcionar un entorno en el que los usuarios puedan realizar cálculos numéricos, desarrollar algoritmos, analizar datos y crear modelos y simulaciones de sistemas complejos.

MATLAB es utilizado en una amplia variedad de aplicaciones, que incluyen, pero no se limitan a:

- Procesamiento de Señales y Comunicaciones

  MATLAB se utiliza para el diseño, análisis y simulación de sistemas de comunicación y procesamiento de señales, como filtros, moduladores y demoduladores, análisis de espectros, etc.

- Control de Sistemas

  En ingeniería de control, MATLAB es una herramienta esencial para el diseño y análisis de sistemas de control. Los ingenieros utilizan MATLAB para modelar sistemas dinámicos, diseñar controladores y simular su comportamiento en un entorno virtual.

- Procesamiento de Imágenes y Visión Artificial

  MATLAB es ampliamente utilizado para el procesamiento de imágenes y el desarrollo de algoritmos de visión artificial, permitiendo la manipulación, mejora y análisis de imágenes y videos.

- Finanzas Computacionales

  MATLAB es utilizado en el ámbito financiero para la modelización de riesgos, análisis de mercados, y la simulación de estrategias de inversión y otros procesos financieros.

- Robótica y Automatización

  MATLAB, en combinación con otros productos de MathWorks, como Simulink, se utiliza para el modelado, simulación y control de sistemas robóticos, permitiendo a los ingenieros desarrollar algoritmos complejos y probarlos en un entorno de simulación antes de su implementación real.

3.2.2 Características Principales

MATLAB es un software extremadamente versátil, con una amplia gama de características y herramientas que lo hacen adecuado para un gran número de aplicaciones técnicas. Algunas de las características más destacadas incluyen:

- Manipulación de Matrices y Álgebra Lineal

  MATLAB ofrece una potente capacidad para la manipulación de matrices y el álgebra lineal, permitiendo realizar operaciones complejas con matrices de manera eficiente. Estas capacidades son fundamentales en áreas como la ingeniería de control, el procesamiento de señales y la modelización matemática.

- Lenguaje de Programación de Alto Nivel

  El lenguaje de programación de MATLAB es sencillo y fácil de aprender, con una sintaxis intuitiva que permite escribir código de manera eficiente. Además, MATLAB soporta programación orientada a objetos, lo que facilita la creación de estructuras de datos complejas y algoritmos personalizados.

- Toolboxes Especializados

  MATLAB ofrece una serie de toolboxes o conjuntos de herramientas especializados para diferentes disciplinas, como el procesamiento de señales, control de sistemas, redes neuronales, optimización, y mucho más. Estos toolboxes permiten a los usuarios aplicar técnicas avanzadas y resolver problemas específicos de sus campos de estudio o trabajo.

- Visualización de Datos

  MATLAB incluye herramientas avanzadas para la visualización de datos, que permiten a los usuarios crear gráficos en 2D y 3D, así como animaciones para representar sus datos de manera clara y efectiva. La capacidad de personalizar y exportar gráficos hace que MATLAB sea una herramienta poderosa para la presentación de resultados.

- Simulink

  Simulink es una extensión de MATLAB que proporciona un entorno para la simulación y modelado de sistemas dinámicos. Es especialmente útil para el diseño de sistemas de control, procesamiento de señales y sistemas embebidos, permitiendo a los usuarios construir modelos gráficos que simulan el comportamiento de sistemas reales.

- Interfaz Gráfica de Usuario (GUI)

  MATLAB permite a los usuarios diseñar y desarrollar interfaces gráficas de usuario personalizadas para sus aplicaciones, facilitando la creación de herramientas interactivas que pueden ser utilizadas por otros usuarios.

3.2.3 Versiones y Compatibilidad

La versión de MATLAB utilizada en este proyecto es MATLAB R2023a, la cual incluye las últimas actualizaciones y mejoras en términos de funcionalidad, eficiencia y compatibilidad. MATLAB R2023a introduce mejoras en el rendimiento, nuevas funciones matemáticas y gráficas, así como actualizaciones en los toolboxes especializados.

MATLAB es compatible con una amplia variedad de plataformas, incluyendo Windows, macOS y Linux, lo que facilita su integración en diferentes entornos de trabajo. Además, MATLAB es altamente compatible con otros lenguajes de programación como C, C++, Java y Python, lo que permite a los usuarios integrar MATLAB en flujos de trabajo más amplios y complejos.

3.2.4 Beneficios del Uso de MATLAB

MATLAB ofrece múltiples beneficios para ingenieros, científicos y profesionales de diferentes disciplinas:

- Eficiencia en el Cálculo

  La capacidad de MATLAB para manejar grandes volúmenes de datos y realizar cálculos complejos de manera rápida y precisa es uno de sus mayores beneficios. Esto permite a los usuarios abordar problemas complejos y obtener resultados de manera eficiente.

- Flexibilidad

  MATLAB es extremadamente flexible, permitiendo a los usuarios personalizar y extender su funcionalidad según las necesidades específicas del proyecto. Los toolboxes especializados y la posibilidad de crear funciones personalizadas amplían significativamente el alcance de lo que se puede lograr con MATLAB.

- Integración con Otros Sistemas

  MATLAB se integra fácilmente con otros sistemas y lenguajes de programación, lo que facilita su incorporación en flujos de trabajo existentes y la colaboración con otros profesionales que utilizan diferentes herramientas.

- Facilidad de Uso

  La interfaz intuitiva y el lenguaje de programación de alto nivel de MATLAB hacen que sea accesible para usuarios con diferentes niveles de experiencia en programación, desde principiantes hasta expertos.

- Amplio Soporte y Comunidad

  MATLAB cuenta con una extensa documentación y una activa comunidad de usuarios, lo que facilita el aprendizaje y la resolución de problemas. Además, MathWorks ofrece soporte técnico y formación para ayudar a los usuarios a maximizar el uso del software.

3.2.5 Limitaciones y Desafíos

A pesar de sus muchas ventajas, MATLAB también presenta algunas limitaciones. Una de las principales es su coste, que puede ser elevado, especialmente para usuarios individuales o pequeñas empresas. Además, aunque MATLAB es extremadamente versátil, algunos usuarios pueden encontrar que el rendimiento disminuye cuando se manejan conjuntos de datos extremadamente grandes o se ejecutan simulaciones muy complejas.

Otra limitación es que, aunque MATLAB es potente para el cálculo numérico y el análisis de datos, no está tan optimizado para el desarrollo de software a gran escala o para la integración en sistemas embebidos, donde otros lenguajes de programación, como C o Python, pueden ser más apropiados.

3.2.6 Conclusión

MATLAB es una herramienta integral que ha demostrado ser indispensable en muchas áreas de la ingeniería, la ciencia y la industria. Su capacidad para manejar cálculos complejos, junto con su flexibilidad, eficiencia y facilidad de uso, lo convierten en una opción preferida para muchos profesionales. En este proyecto, MATLAB ha sido crucial para el análisis de datos, el desarrollo de algoritmos y la simulación de sistemas, contribuyendo significativamente al éxito del proyecto.

3.3 ABB IRC5 OPC

ABB IRC5 OPC Configuration es una herramienta esencial utilizada en la configuración y gestión de comunicaciones entre el controlador de robots IRC5 de ABB y otros sistemas a través del protocolo OPC (OLE for Process Control). Esta herramienta facilita la integración de robots industriales en un entorno automatizado, permitiendo el intercambio de datos en tiempo real entre el controlador y sistemas SCADA, PLCs u otros dispositivos de automatización.

3.3.1 Propósito y Aplicaciones

El controlador IRC5 de ABB es uno de los controladores de robots más avanzados y ampliamente utilizados en la industria. Es el cerebro detrás de los robots industriales ABB, gestionando sus movimientos, operaciones y la interacción con otros sistemas. Para facilitar esta interacción, especialmente en entornos de automatización complejos, se utiliza la configuración OPC.

El propósito principal de la configuración OPC en el IRC5 es permitir la comunicación estándar y la interoperabilidad entre diferentes dispositivos y sistemas de software en una planta de producción. OPC es un protocolo abierto que facilita el intercambio de información entre dispositivos de diferentes fabricantes, lo que es crucial para la integración de robots ABB en un entorno de producción heterogéneo.

Las aplicaciones de ABB IRC5 OPC Configuration son diversas y abarcan múltiples industrias donde se requiere la integración de robots con sistemas de control de procesos, monitoreo y adquisición de datos. Algunos ejemplos incluyen:

- Automoción

  Integración de robots en líneas de ensamblaje donde la comunicación en tiempo real con sistemas SCADA es crítica para el control y monitoreo del proceso.

- Manufactura

  En plantas de manufactura, donde es necesario que los robots interactúen con otros equipos y sistemas de control para realizar tareas sincronizadas y garantizar la calidad del producto.

- Industria Farmacéutica

  Integración de robots en entornos controlados para la manipulación de materiales sensibles, donde la comunicación y el monitoreo en tiempo real son esenciales para cumplir con las normativas.

3.3.2 Características Principales

ABB IRC5 OPC Configuration proporciona una serie de características que facilitan la configuración, monitoreo y gestión de la comunicación entre el controlador IRC5 y otros sistemas. Algunas de las características más importantes incluyen:

- Interoperabilidad Estándar

  Utilizando el protocolo OPC, la herramienta asegura que el controlador IRC5 pueda comunicarse de manera efectiva con una amplia gama de dispositivos y sistemas, independientemente del fabricante. Esto es esencial para la integración en plantas con equipos heterogéneos.

- Configuración Sencilla y Flexible

  La herramienta permite configurar de manera intuitiva los puntos de datos que se compartirán entre el IRC5 y otros sistemas. Los usuarios pueden definir y mapear señales específicas del robot, como posiciones, velocidades, estados de herramientas, y otros parámetros operativos.

- Monitoreo en Tiempo Real

  Una vez configurada, la herramienta facilita el monitoreo en tiempo real de los datos del robot, lo que permite a los operadores y sistemas SCADA recibir y procesar datos instantáneamente. Esto es crucial para la toma de decisiones y el control en tiempo real de procesos automatizados.

- Compatibilidad con Múltiples Protocolos OPC

  ABB IRC5 OPC Configuration soporta varios estándares de OPC, incluyendo OPC DA (Data Access) y OPC UA (Unified Architecture), lo que ofrece flexibilidad en la integración con diferentes tipos de sistemas de automatización.

- Seguridad y Fiabilidad

  La herramienta está diseñada para garantizar la seguridad en la comunicación de datos, protegiendo la información crítica del robot contra accesos no autorizados y garantizando la integridad de los datos durante la transmisión.

3.3.3 Versiones y Compatibilidad

La herramienta de configuración OPC para IRC5 ha evolucionado a lo largo de los años, con cada nueva versión proporcionando mejoras en la funcionalidad, seguridad y facilidad de uso. La

versión utilizada en este proyecto es ABB IRC5 OPC Configuration 2023, que incluye soporte completo para OPC UA, mejorando la compatibilidad y la seguridad de las comunicaciones.

Esta versión es compatible con los controladores IRC5 más recientes y está diseñada para integrarse sin problemas con los sistemas SCADA y PLC más utilizados en la industria. Además, es compatible con las versiones anteriores de OPC, lo que facilita la integración en plantas que operan con diferentes generaciones de tecnología de automatización.

3.3.4 Beneficios del Uso de ABB IRC5 OPC Configuration

El uso de ABB IRC5 OPC Configuration ofrece múltiples beneficios, tanto en términos de integración como de operación:

- Integración Simplificada

  La herramienta simplifica la tarea de integrar los robots ABB en sistemas de automatización existentes, lo que reduce el tiempo de implementación y los costos asociados.

- Comunicación Eficiente y en Tiempo Real

  Al utilizar el protocolo OPC, la herramienta garantiza que los datos del robot se transmitan de manera eficiente y confiable, lo que es crucial para el control en tiempo real y la toma de decisiones.

- Flexibilidad en la Configuración

  Los usuarios pueden personalizar la configuración para adaptarse a las necesidades específicas de su aplicación, lo que permite una mayor flexibilidad y control sobre el sistema robótico.

- Mejora en la Productividad

  Al facilitar una comunicación fluida y en tiempo real, la herramienta ayuda a mejorar la productividad de la planta, ya que los sistemas de control pueden reaccionar rápidamente a los cambios en las operaciones del robot.

- Reducción de Errores

  La capacidad de monitorear y ajustar la configuración OPC de manera precisa reduce la posibilidad de errores en la comunicación, lo que a su vez mejora la fiabilidad del sistema en general.

3.3.5 Limitaciones y Desafíos

Aunque ABB IRC5 OPC Configuration ofrece muchas ventajas, también presenta algunos desafíos y limitaciones. Uno de los principales desafíos es la complejidad en la configuración inicial, especialmente en entornos con una gran cantidad de señales y dispositivos que deben ser integrados. La curva de aprendizaje puede ser empinada para los usuarios que no están familiarizados con el protocolo OPC o con la arquitectura del sistema IRC5.

Otra limitación es que, aunque OPC UA ofrece mejoras significativas en términos de seguridad y escalabilidad en comparación con OPC DA, su implementación puede requerir una infraestructura de red más robusta y un mayor conocimiento técnico.

Capítulo 4

4. DESARROLLO DEL PROYECTO

En este capítulo se detallan todas las etapas del desarrollo del proyecto, desde la fase de planificación hasta la implementación final. El propósito de esta sección es ofrecer una descripción exhaustiva del proceso llevado a cabo para alcanzar los objetivos establecidos en la fase inicial del proyecto.

El desarrollo del proyecto se ha estructurado en varias fases clave, que incluyen el diseño conceptual, la simulación y modelado, la programación y control del sistema robótico, y finalmente, la validación y pruebas del sistema en un entorno controlado. Cada una de estas fases ha sido fundamental para garantizar que el proyecto no solo cumple con los requisitos técnicos, sino que también se adapte a las necesidades prácticas y operativas del entorno de aplicación.

En este apartado detallaremos las secciones donde se explican los pasos realizados, comenzamos con el modelado de la estación, luego continuaremos con la programación de los módulos de la estación, la definición de puntos y trayectorias y su lógica.

Finalmente programaremos la interfaz con el usuario en Matlab y configuraremos la comunicación OPC UA necesaria para poder realizar la conexión entre estación y HMI.

4.1 PRIMERA FASE: RobotStudio

En esta primera fase vamos a explicar detalladamente como se ha diseñado la estación sobre la que se trabajará en este proyecto, contando con que previamente han sido definidas las características deseadas. Durante este proceso primero hemos generado una estación, hemos añadido nuestros robots virtuales, les hemos ido posicionando en las posiciones óptimas para un mejor desarrollo, hemos creado los controladores correspondientes y finalmente generado los componentes inteligentes y las trayectorias. Todas estas tareas definidas previamente son las que conformaran el modelado de nuestra estación.

4.1.1 CONFIGURACIÓN Y MODELADO DE LA ESTACIÓN

Como hemos indicado previamente lo primero que vamos a hacer es abrir el software RobotStudio y vamos a crear una nueva estación vacía.

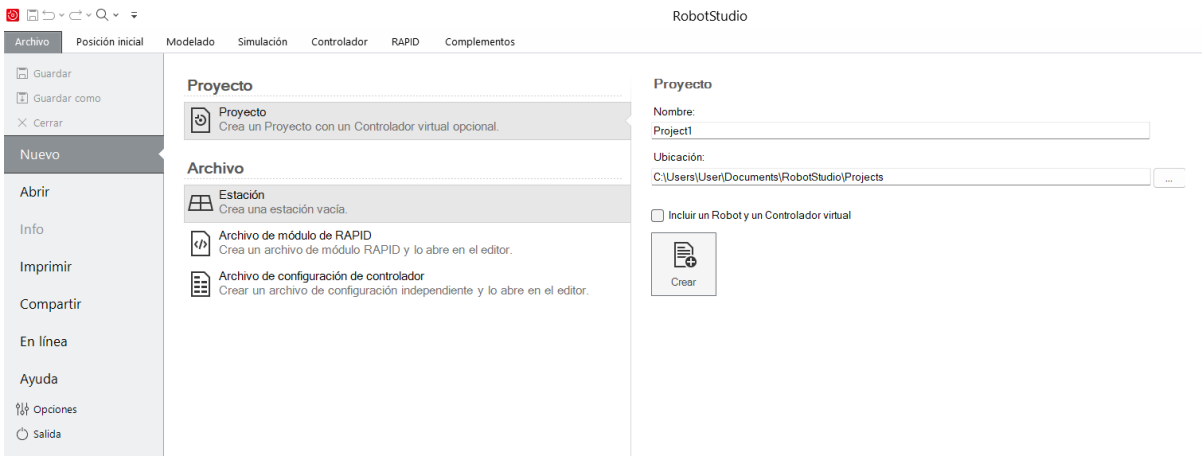


Ilustración 11: Página inicio RobotStudio

Una vez ya tenemos creada la estación vacía comenzaremos con el modelado en materia.

4.1.1.1 Modelado de la estación

Primero vamos a añadir el robot que vamos a emplear. Accedemos a la biblioteca ABB donde tendremos acceso a todos los robots proporcionados por el propio software. La oferta entre los distintos tipos es muy amplia: Articulados, Colaborativos, Paralelos, SCARA, etc. Como ya hemos dicho antes, añadiremos el robot IRB 120.

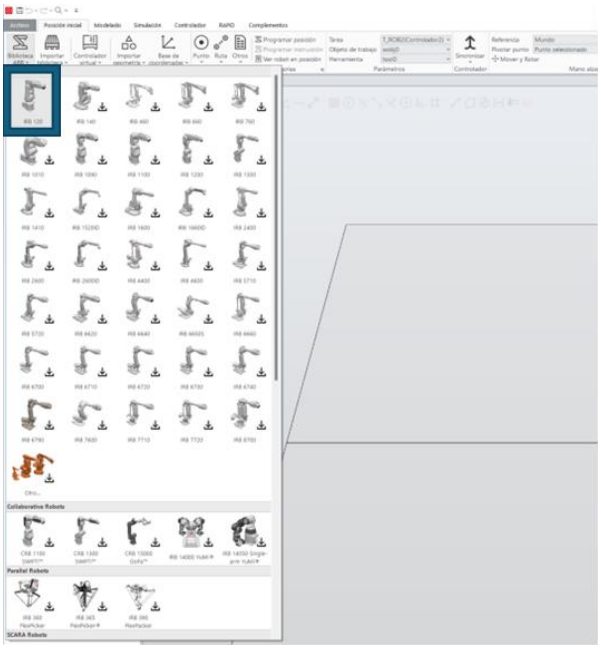


Ilustración 12: Robots articulados de la biblioteca de ABB

Colocamos el robot en la posición fija deseada y lo duplicamos.

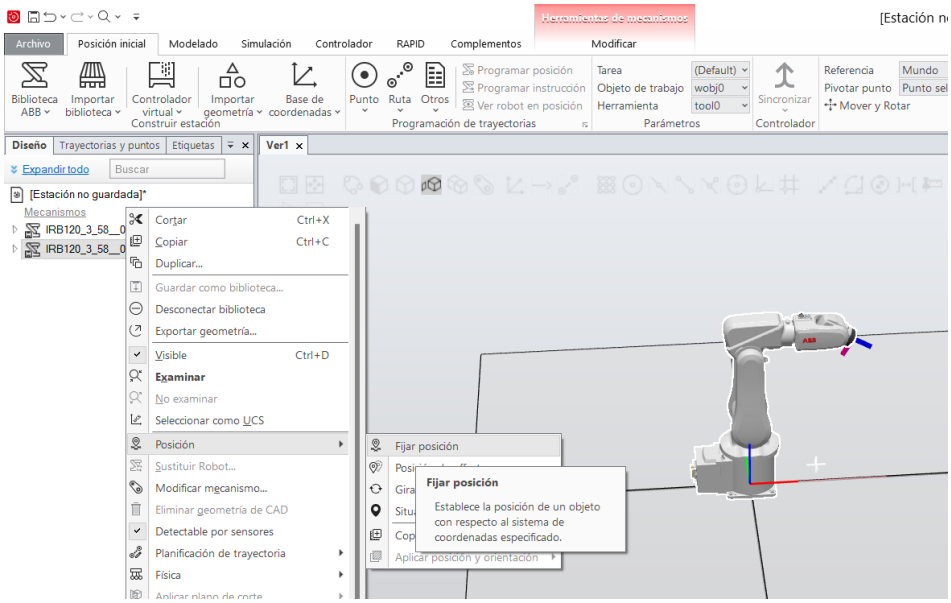


Ilustración 13: Fijar posición de un robot en RobotStudio

Una vez importados al proyecto nuestro robot, vamos a darles una posición fija respecto a la referencia mundo.

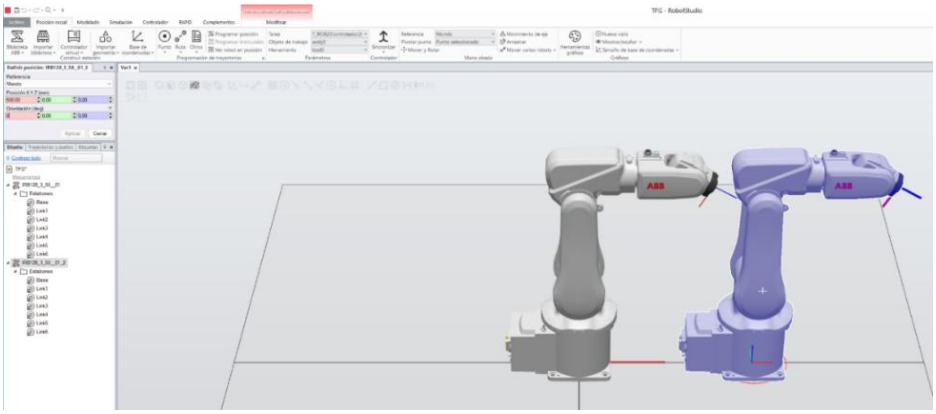


Ilustración 14: Robots IRB 1200 duplicado en estación

Una vez ya tenemos colocados nuestros robots manipuladores, añadiremos nuestro YuMi 1400

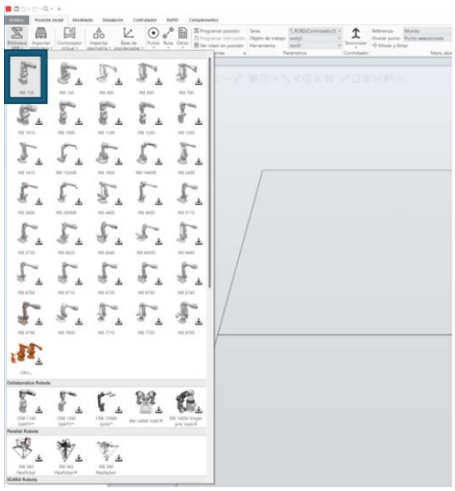


Ilustración 15: Robots articulados de la biblioteca de ABB

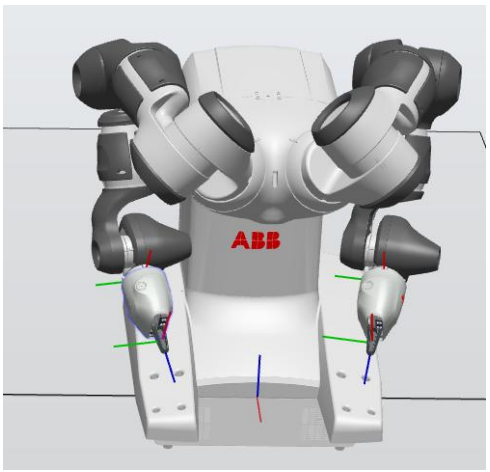


Ilustración 16: Robot IRB 1400 YuMi en la estación de modelado

A continuación, creamos el piano, estará conformado por varios grupos de solidos con sus correspondientes componentes inteligentes.

Comenzaremos generando un tetraedro para simular nuestras teclas blancas que tendrán las siguientes medidas: Nota blanca (100, 800, 30).

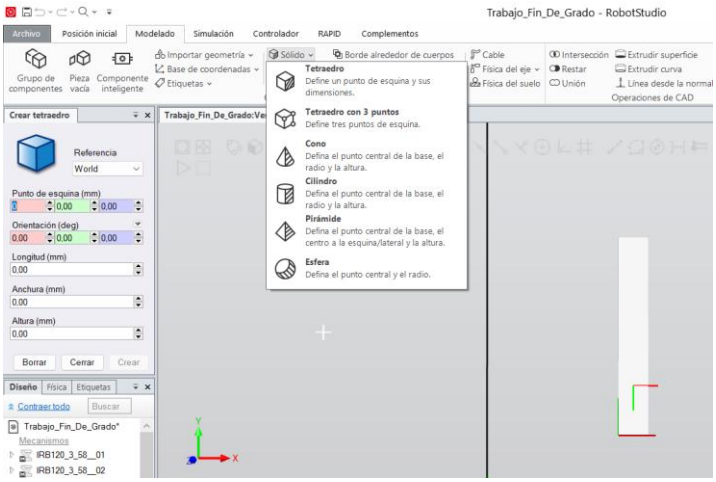


Ilustración 17: Creación de un tetraedro "Nota Blanca"

Luego repetiremos el proceso para crear las teclas blancas de nuestro teclado, pero con distintas dimensiones: Nota negra (100, 600, 30).

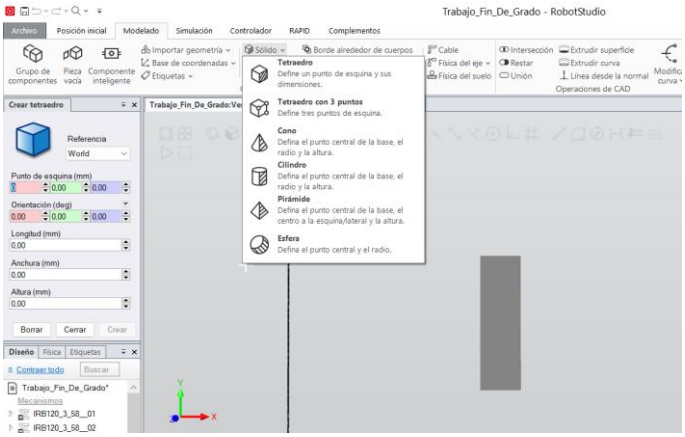


Ilustración 18: Creación de un tetraedro "Nota Negra"

Generamos tantos tetraedros como sean necesarios para poder formar nuestra primera escala, obteniendo la una parte de nuestro teclado.

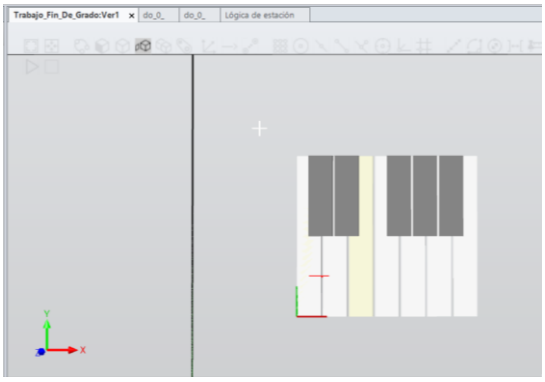


Ilustración 19: Primera escala del teclado de nuestra estación de RobotStudio

Normalmente los pianos están formados de ocho escalas, pero en nuestro caso estará compuesto por un total de tres. La primera definida para uno de los robots, la última para el otro y una común en el medio para ambas, donde tanto el primer IRB 1200 como el segundo tendrá acceso a ella.

Una vez ya tenemos la composición completa del teclado, le añadiremos otro sólido tetraedro con color negro, que será la base de nuestro instrumento.

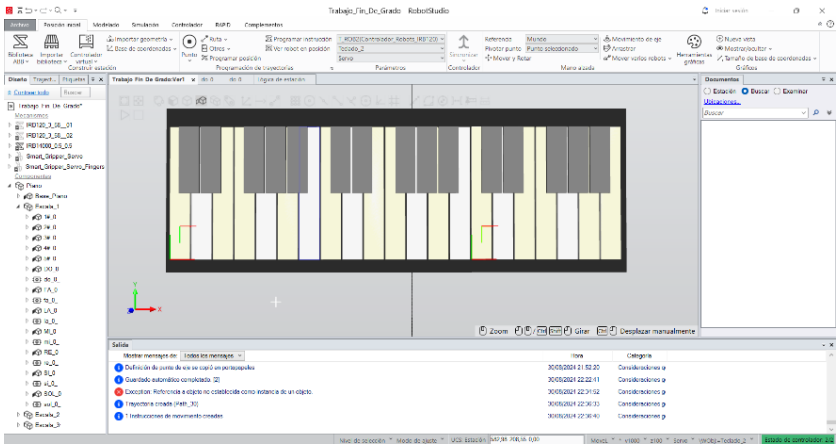


Ilustración 20: Teclado en la estación de modelado de RobotStudio

Ahora vamos a modelar la herramienta que vamos a emplear con nuestros IRB 1200. Se trata de una pinza inteligente con doble posición, encargada de pulsar las notas del teclado, nota a nota de manera individual (Posición cerrada) y tocar dos notas a la vez en un intervalo de segunda (Posición abierta). En este caso vamos a partir de primero hacer la geometría.

Primero crearemos la base de la pinza y para ello vamos a crear otro solido tetraedro de 80 mm de largo, 30 mm de ancho y 110 mm de alto.

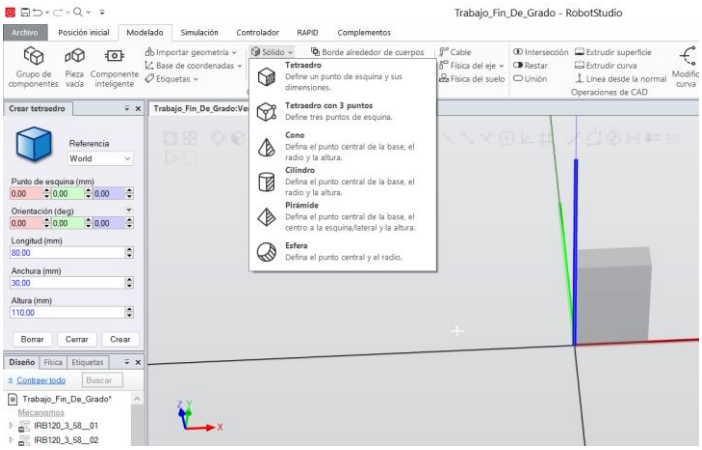


Ilustración 21: Creación base de la pinza inteligente

Ahora vamos a generar lo que es la brida creando un cilindro de radio 30 mm y altura 8 mm.

Ilustración 22: Creación brida de la pinza inteligente

Ya creadas la base y la brida vamos a modificar sus posiciones fijando nuevas coordenadas.

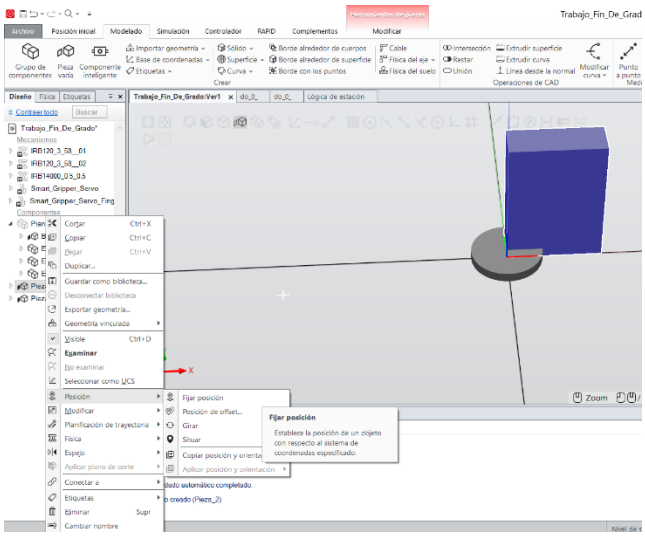


Ilustración 23: Posición fija de una pieza en RobotStudio

Vamos a fijar una posición respecto de la referencia mundo a la altura de la brida de 8 mm y luego vamos a centrándolo, desplazándolo –40 mm en el eje X y –15 mm en el eje Y.

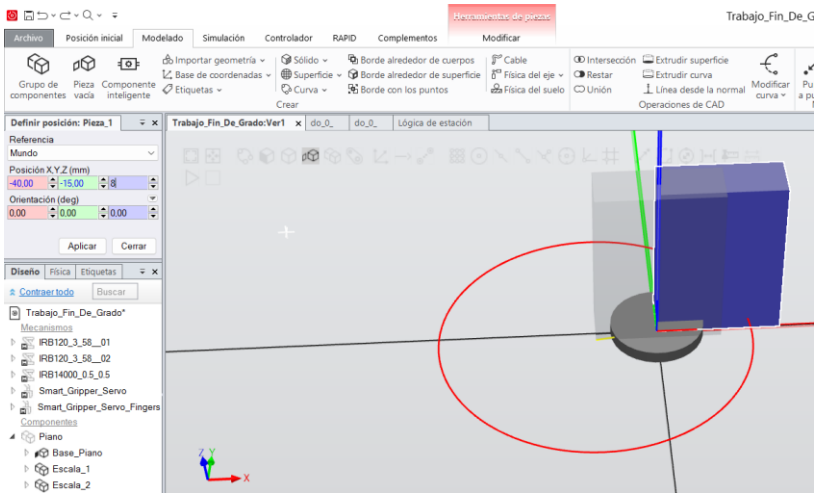


Ilustración 24: Definición de posiciones en RobotStudio

Aplicamos y podemos observar la forma de la base de nuestra pinza inteligente.

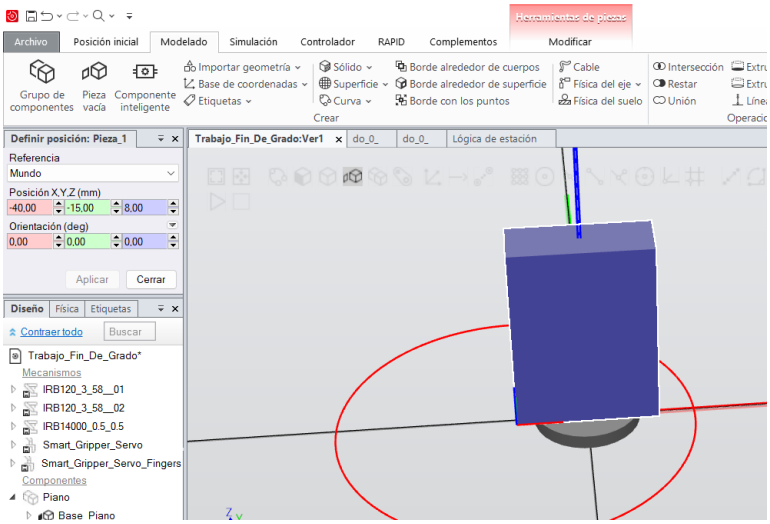


Ilustración 25: Cuerpo de nuestra pinza inteligente

Una vez ya tenemos la posición deseada, crearemos la unión a la que vamos a definir como cuerpo. Generamos una unión y definimos qué figuras son las que queremos que lo formen, quitando la opción de conservar.

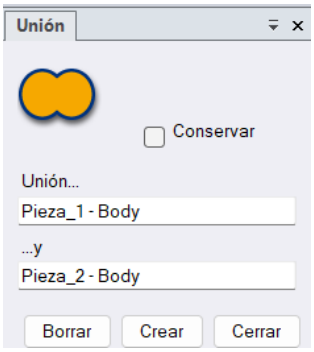


Ilustración 26: Creación de unión de dos cuerpos en RobotStudio

Ya tenemos la base completa de nuestra herramienta, a lo que llamaremos cuerpo.

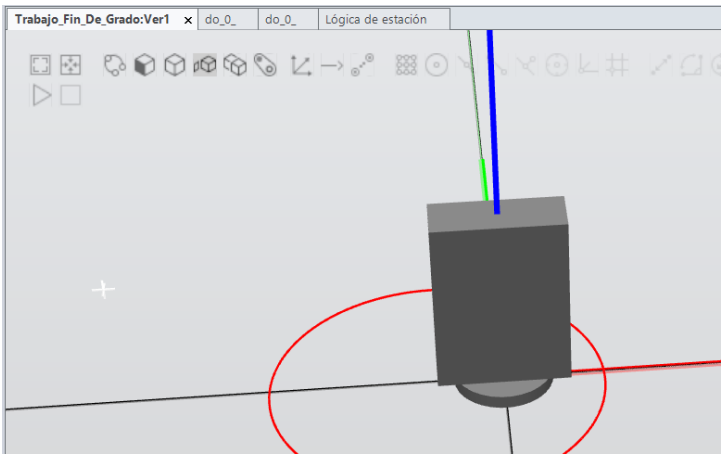


Ilustración 27: Unión del cuerpo de nuestra pinza inteligente

Ahora vamos a definir un color para la pinza y seleccionar un color.

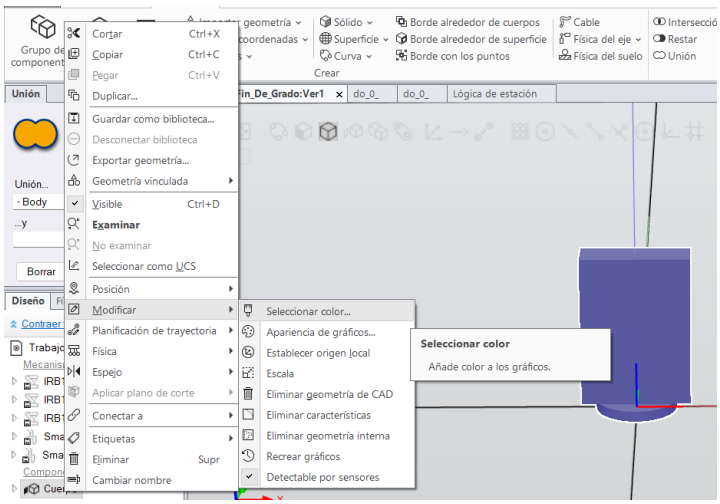


Ilustración 28: Selección de color en RobotStudio

Escogemos un color y damos a aceptar y guardar.

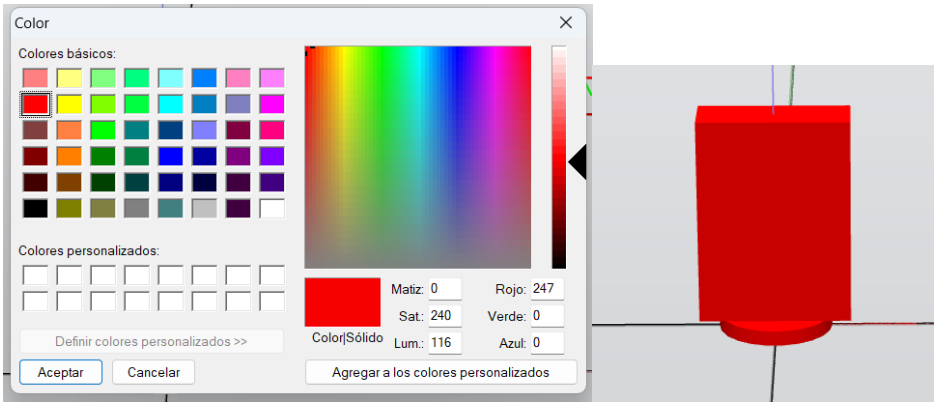


Ilustración 29: Colores básicos para nuestro cuerpo de la pinza inteligente

Ya tenemos la base, pero nos falta crear las pinzas de nuestra herramienta. Creamos dos sólidos tetraedros de 6 mm de largo, 30 mm de ancho y 90 de alto.

Una vez creada la primera, la tendremos que subir en altura 118 mm (8 mm de la base y 110 mm de la pinza) y la desplazaremos 40 mm a la derecha y para centrarla la llevaremos a –5 mm.

Para la otra garra de la pinza generamos un duplicado de esta última, la damos un offset respecto de la anterior de –80 mm.

Ya tenemos nuestra herramienta modelada al completo.

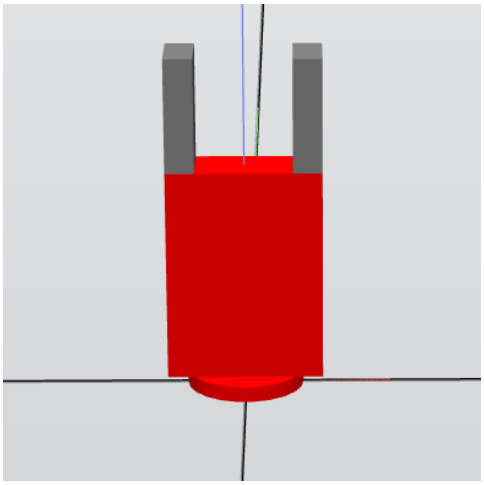


Ilustración 30: Modelado de nuestra Pinza Inteligente en RobotStudio

Una vez ya tenemos la herramienta diseñada en nuestra estación, y nuestra estación modelada tendrá la siguiente apariencia.

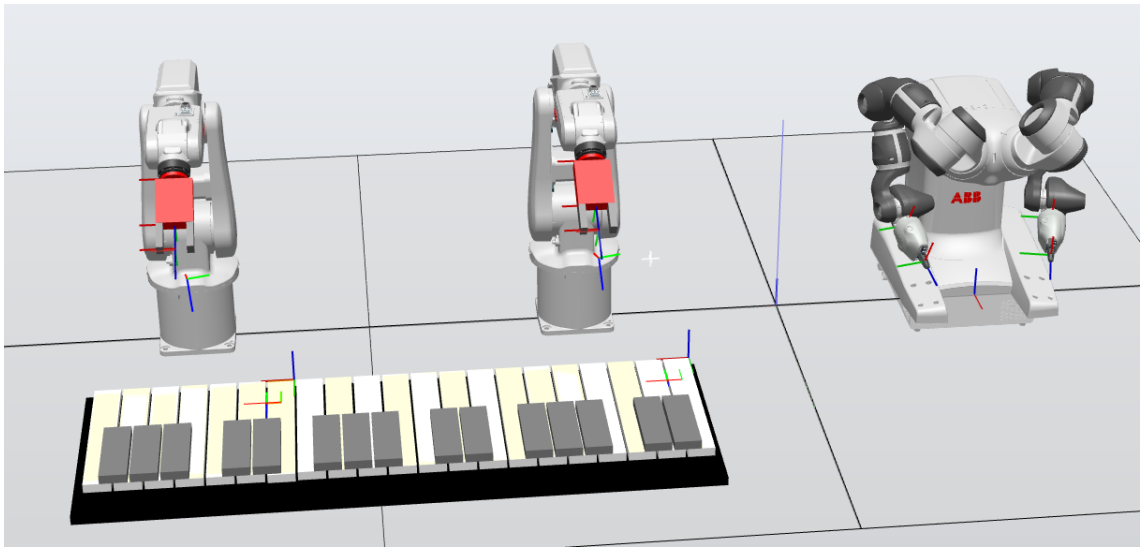


Ilustración 31: Estación modelada en RobotStudio

4.1.1.2 Controladores virtuales

Para crear el controlador virtual tendremos que ir a la pestaña Controlador que contiene los controles utilizados para gestionar un controlador real y los controles necesarios para la sincronización, configuración y tareas asignadas al controlador virtual. RobotStudio le permite trabajar con un controlador fuera de línea, que constituye un controlador IRC5 virtual que se ejecuta localmente en su PC. Este controlador fuera de línea también se conoce como el controlador virtual (VC) (ABB, 2025).

RobotStudio también le permite trabajar con un controlador IRC5 físico real, que simplemente se conoce como el controlador real. Las funciones de la pestaña Controlador pueden clasificarse de la siguiente forma:

- Funciones para controladores tanto virtuales como reales
- Funciones para controladores reales
- Funciones para controladores virtuales

La programación offline es la mejor forma de maximizar la rentabilidad de la inversión en sistemas robotizados. El software de simulación y programación offline de ABB, RobotStudio, permite programar los robots en un PC de la oficina sin necesidad de parar la producción, lo que permite realizar tareas como formación, programación y optimización.

La herramienta se basa en el controlador virtual de ABB, una copia exacta del software real que hace funcionar sus robots en la producción. Esto permite realizar simulaciones muy realistas, utilizando programas de robot reales y archivos de configuración idénticos a los que se utilizan en el taller (ABB, 2025).

Para llevar a cabo el proyecto necesitaremos crear dos controladores virtuales, uno para los dos robots IRB 1200 y otro para el robot IRB 1400 YuMi.

Ilustración 32: Creación de un controlador virtual en RobotStudio

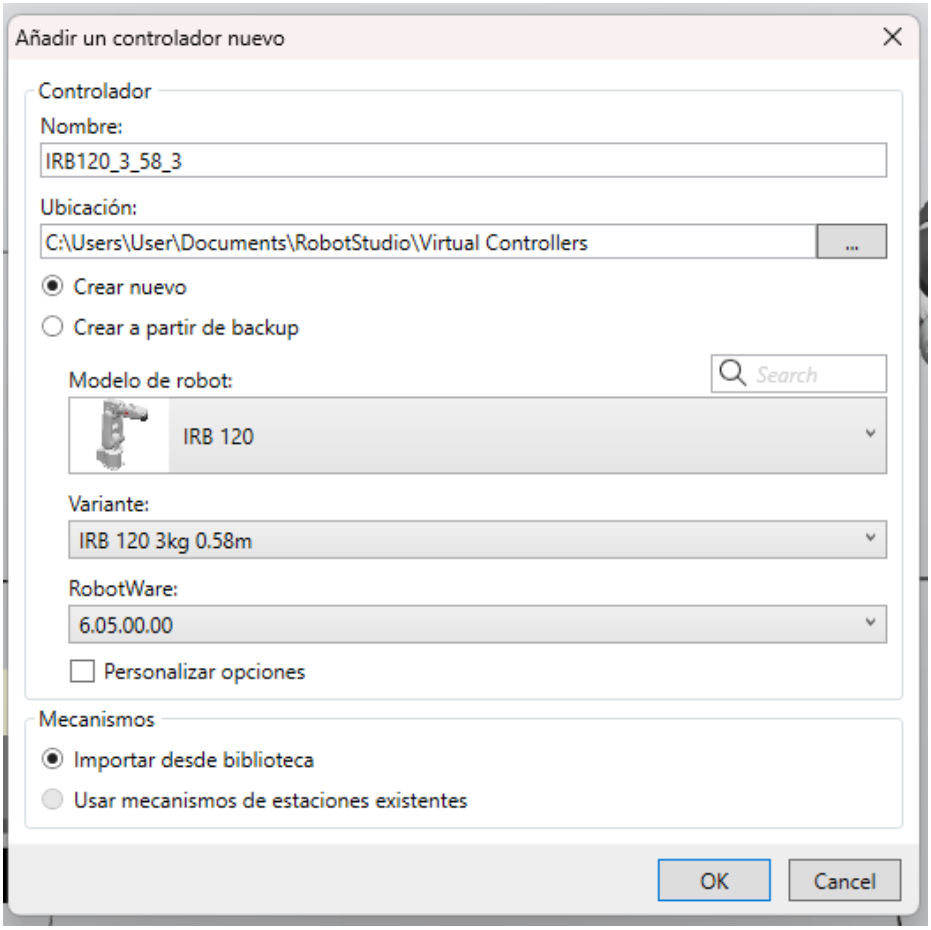
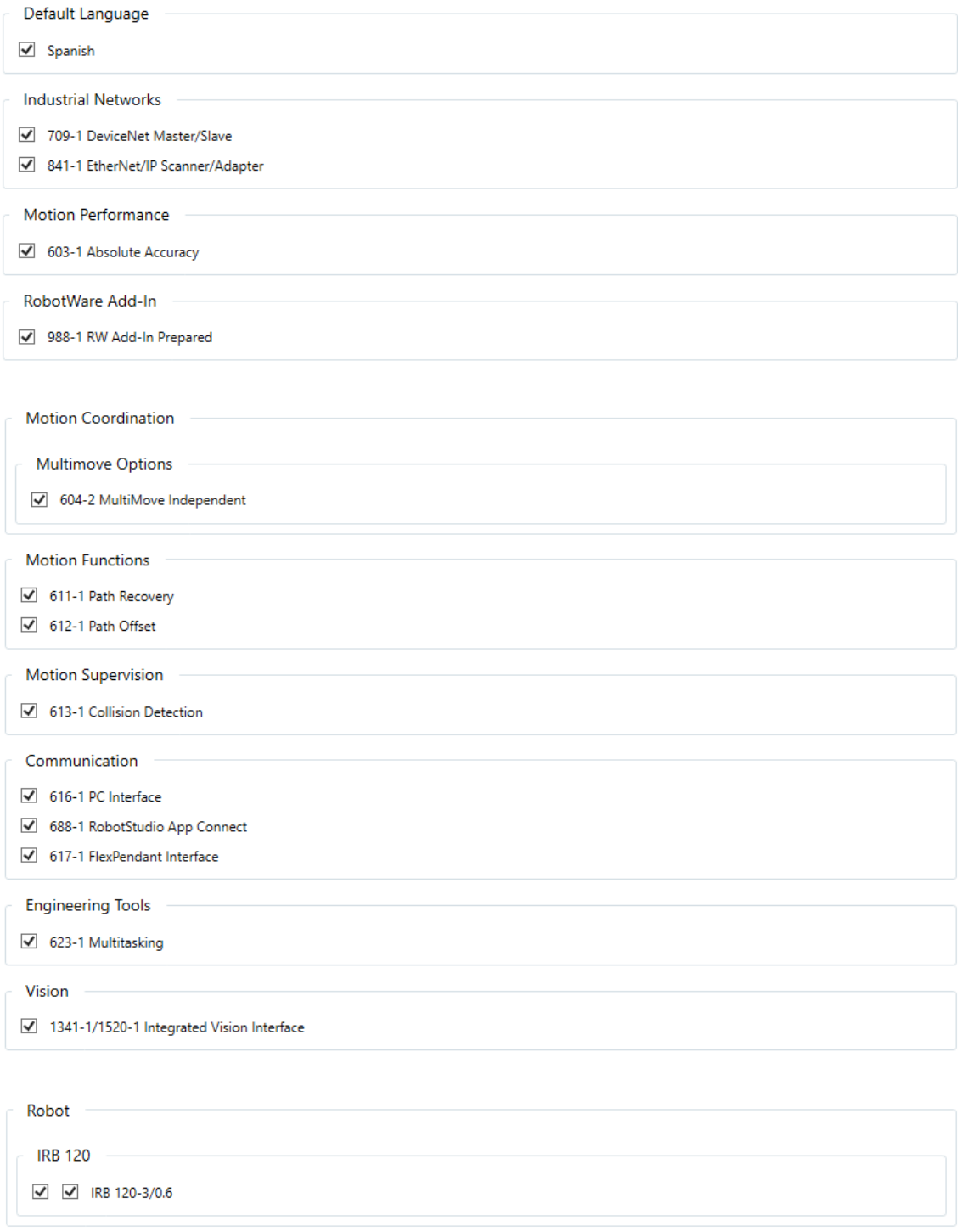


Ilustración 33: Características de la creación de un nuevo controlador virtual en RobotStudio

El controlador lo crearemos con las siguientes características:



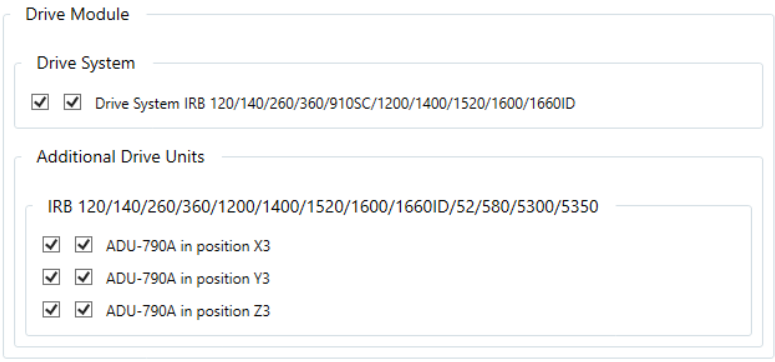


Ilustración 34: Configuración de un controlador virtual para dos IRB 1200 en RobotStudio

Podemos destacar la función 604-2 MultiMove Independent, función del sistema de RobotWare que permite que varios robots y ejes externos trabajen de forma independiente y no simultánea, aunque ambos estén vinculados a un mismo IRC5 o controlador virtual.

Una vez generado nuestro controlador, tendremos que crear nuestras señales de entrada y salida, así como un dispositivo que será d645.

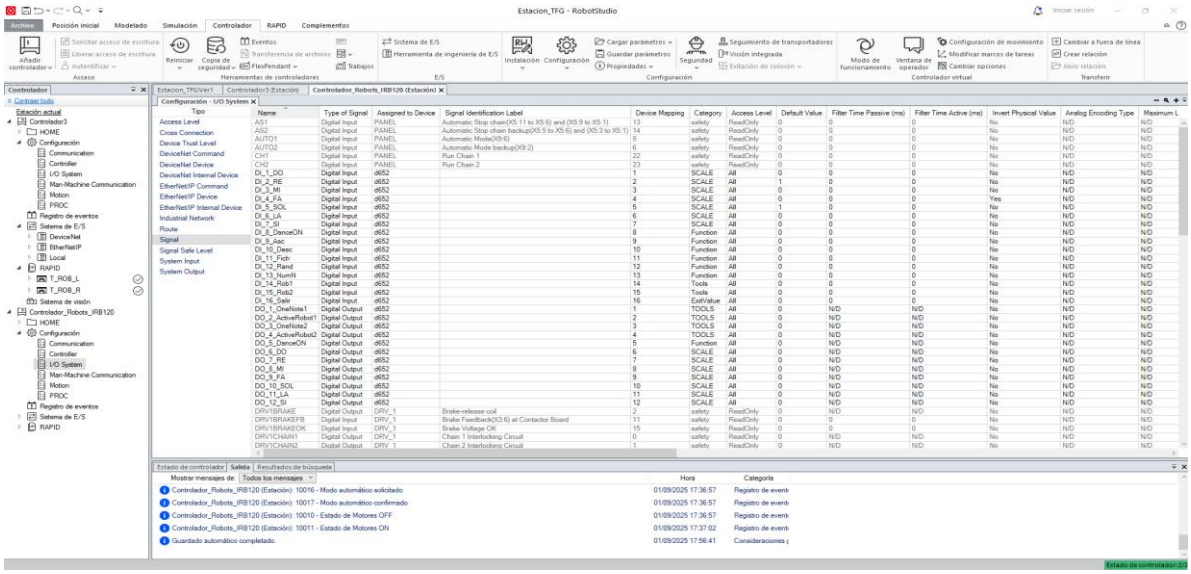


Ilustración 35: Señales de entrada y salida del controlador IRB 1200 de RobotStudio

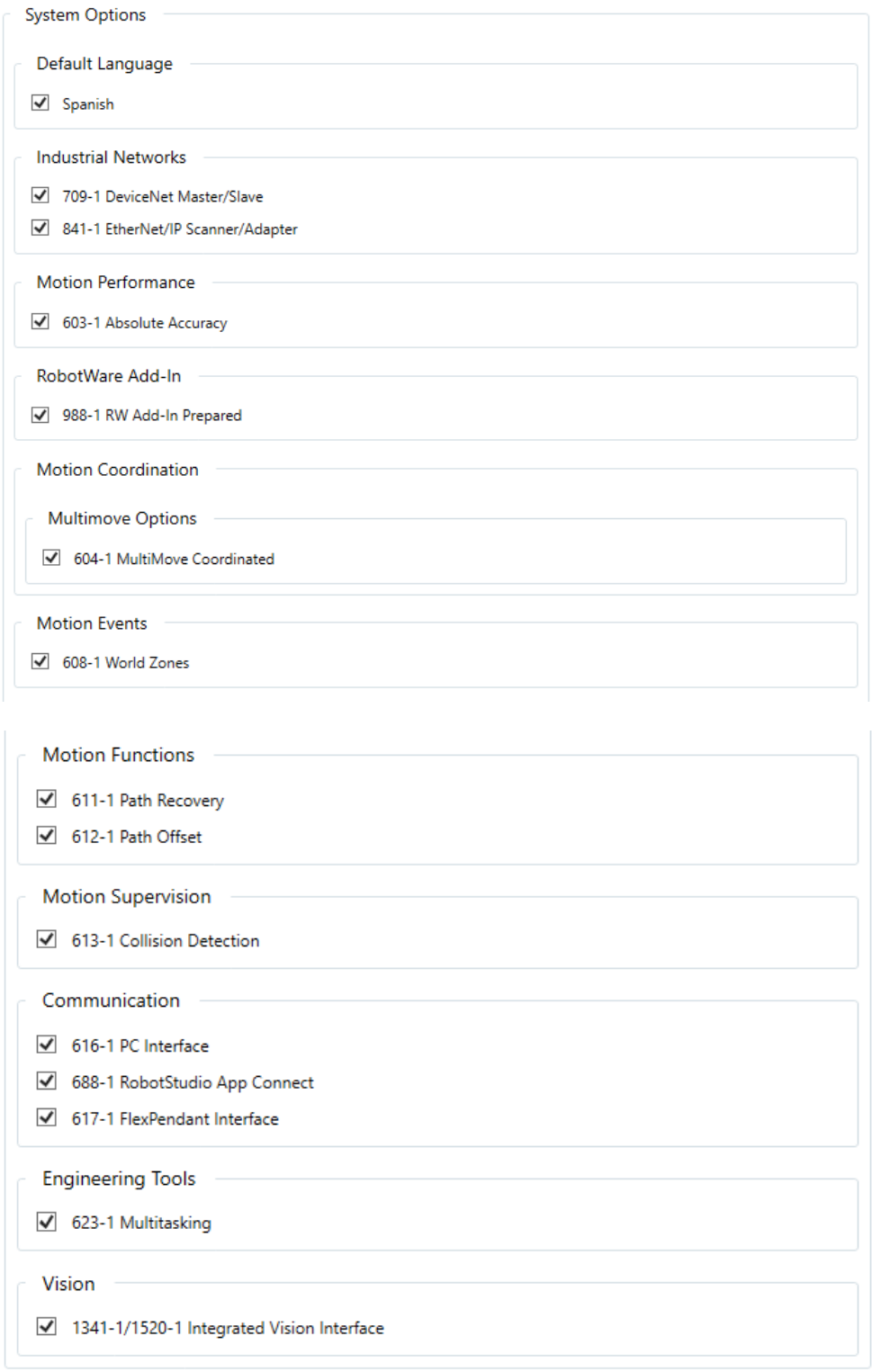
El total de señale generadas para nuestro controlador donde son 16 entradas digitales y 13 salidas.

Nombre	Tipo de señal	Mapeo de dispositivo	Categoría	Nivel de acceso	Función
DI_1_DO	Digital Input	1	SCALE	Todos	Entrada digital para la nota do
DI_2_RE	Digital Input	2	SCALE	Todos	Entrada digital para la nota re
DI_3_MI	Digital Input	3	SCALE	Todos	Entrada digital para la nota mi
DI_4_FA	Digital Input	4	SCALE	Todos	Entrada digital para la nota fa
DI_5_SOL	Digital Input	5	SCALE	Todos	Entrada digital para la nota sol
DI_6_LA	Digital Input	6	SCALE	Todos	Entrada digital para la nota la

DI_7_SI	Digital Input	7	SCALE	Todos	Entrada digital para la nota si
DI_8_DanceON	Digital Input	8	Function	Todos	Entrada digital que activa el YuMi
DI_9_Asc	Digital Input	9	Function	Todos	Entrada digital para escala ascendente
DI_10_Desc	Digital Input	10	Function	Todos	Entrada digital para escala descendente
DI_11_Fich	Digital Input	11	Function	Todos	Entrada digital para tocar partitura
DI_12_Rand	Digital Input	12	Function	Todos	Entrada digital para notas aleatorias
DI_13_NumN	Digital Input	13	Function	Todos	Entrada digital para nota o intervalo
DI_14_Rob1	Digital Input	14	Function	Todos	Entrada digital para activar IRB 1200_1
DI_15_Rob2	Digital Input	15	Function	Todos	Entrada digital para activar IRB 1200_2
DI_16_Salir	Digital Input	16	Exit	Todos	Entrada digital para salir de la simulación
DO_1_OneNote1	Digital Output	1	Tools	Todos	Salida digital al activar modo intervalo
DO_2_ActiveRobot1	Digital Output	2	Tools	Todos	Salida digital al activar IRB 1200_1
DO_3_OneNote2	Digital Output	3	Tools	Todos	Salida digital al activar modo intervalo
DO_4_ActiveRobot2	Digital Output	4	Tools	Todos	Salida digital al activar IRB 1200_2
DO_5_DanceON	Digital Output	5	Funcion	Todos	Salida digital al activar IRB 1400 YuMi
DO_6_DO	Digital Output	6	SCALE	Todos	Salida digital para la nota do
DO_7_RE	Digital Output	7	SCALE	Todos	Salida digital para la nota re
DO_8_MI	Digital Output	8	SCALE	Todos	Salida digital para la nota mi
DO_9_FA	Digital Output	9	SCALE	Todos	Salida digital para la nota fa
DO_10_SOL	Digital Output	10	SCALE	Todos	Salida digital para la nota sol
DO_11_LA	Digital Output	11	SCALE	Todos	Salida digital para la nota la
DO_12_SI	Digital Output	12	SCALE	Todos	Salida digital para la nota si

Tabla 8: I/O del controlador IRB 1200

Una vez ya tenemos agregado y configurado uno de los controladores, pasamos a configurar el controlador encargado de los movimientos del IRB 1400, que tendrá las siguientes características.


System Options
Default Language
Spanish
Industrial Networks
709-1 DeviceNet Master/Slave
841-1 EtherNet/IP Scanner/Adapter
Motion Performance
603-1 Absolute Accuracy
RobotWare Add-In
988-1 RW Add-In Prepared
Motion Coordination
Multimove Options
604-1 MultiMove Coordinated
Motion Events
608-1 World Zones
Motion Functions
611-1 Path Recovery
612-1 Path Offset
Motion Supervision
613-1 Collision Detection
Communication
616-1 PC Interface
688-1 RobotStudio App Connect
617-1 FlexPendant Interface
Engineering Tools
623-1 Multitasking
Vision
1341-1/1520-1 Integrated Vision Interface

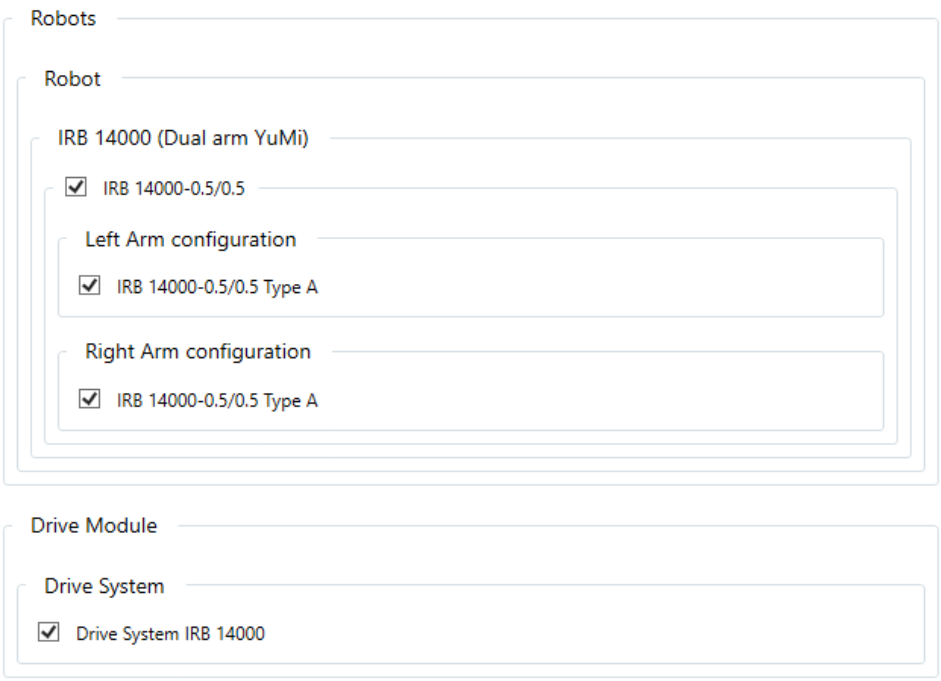


Ilustración 36: Configuración de un controlador virtual para un IRB 1400 en RobotStudio

Accedemos a las señales del dispositivo, con el fin de generar las entradas y salidas digitales y así poder comunicarnos con otros dispositivos, ya sean controladores, módulos de RAPID u otros softwares.

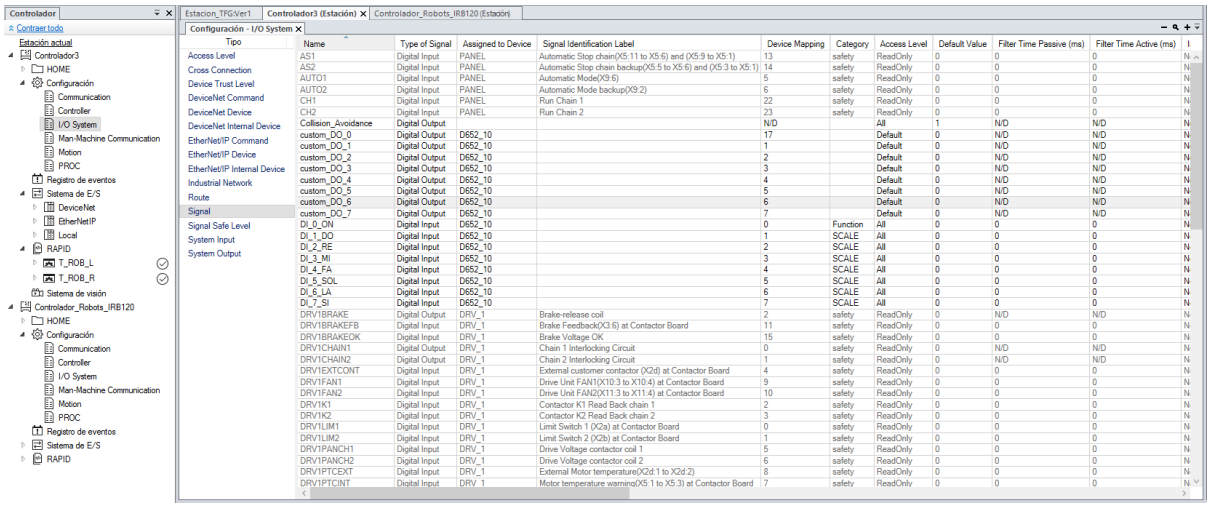


Ilustración 37: Señales de entrada y salida del controlador IRB 1200 de RobotStudio

En nuestro controlador para el YuMi el número de variables creadas es un total de ocho, donde solo contaremos con entradas digitales.

Nombre	Tipo de señal	Mapeo de dispositivo	Categoría	Nivel de acceso	Función
DI_0_ON	Digital Input	0	Function	Todos	Entrada digital para activar baile
DI_1_DO	Digital Input	1	SCALE	Todos	Entrada digital para la pose do
DI_2_RE	Digital Input	2	SCALE	Todos	Entrada digital para la pose re
DI_3_MI	Digital Input	3	SCALE	Todos	Entrada digital para la pose mi
DI_4_FA	Digital Input	4	SCALE	Todos	Entrada digital para la pose fa
DI_5_SOL	Digital Input	5	SCALE	Todos	Entrada digital para la pose sol
DI_6_LA	Digital Input	6	SCALE	Todos	Entrada digital para la pose la
DI_7_SI	Digital Input	7	SCALE	Todos	Entrada digital para la pose si

Tabla 9: I/O del controlador IRB 1400 YuMi

4.1.1.3 Componentes Inteligentes

Un componente inteligente es un objeto virtual que puede contener lógica interna, señales de entrada/salida y comportamientos personalizados, utilizado para simular el funcionamiento de equipos periféricos, mecanismos o procesos automatizados dentro de una celda robótica.

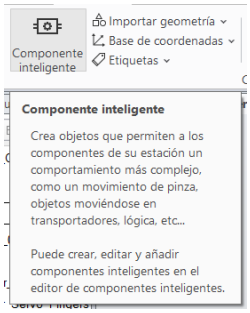


Ilustración 38: Componente inteligente en RobotStudio

Funciones de los componentes inteligentes:

Simulación de lógica	• Se puedes agregar comportamientos como movimiento, animaciones o respuestas a señales sin necesidad de programar directamente en RAPID
Interacción por señales	• Se conectan mediante señales de entrada y salida virtuales con el controlador del robot
Diseño modular	• Permite construir y reutilizar componentes personalizados en diferentes proyectos
Animaciones y eventos	• Se pueden realizar basadas en condiciones o señales

Tabla 10: SmartArt de las funciones de los componentes inteligentes

Para poder replicar el comportamiento lógico de nuestra estación, hemos creado varios componentes inteligentes. Vamos a comenzar explicando todos los que componen las distintas notas del piano.

Sensor de colisión

El sensor de colisión es un componente inteligente diseñado para simular la detección de contacto entre un objeto (como una herramienta o el brazo de un robot) y otro elemento de la celda virtual. Este componente se utiliza para detectar colisiones de forma visual o lógica, permitiendo ejecutar respuestas programadas cuando se produce una interacción física no deseada o intencional.

Características principales:

1. Detección automática de contacto físico entre el componente inteligente y otro objeto 3D dentro del entorno en RobotStudio, nuestra pinza “Pieza1_Intervalos”.

2. Salida digital activada cuando se detecta una colisión.

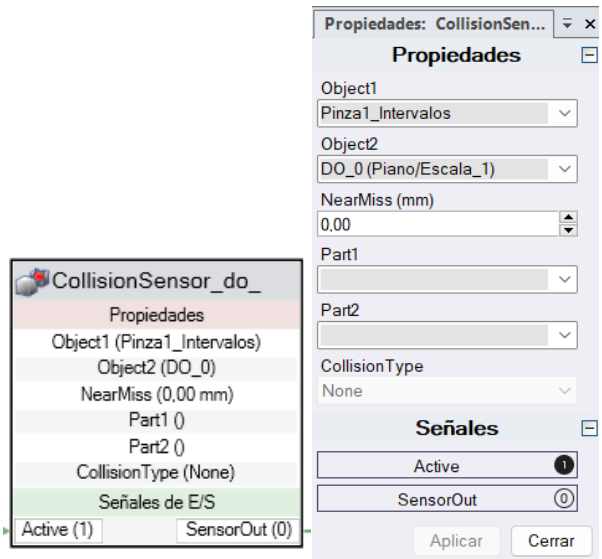


Ilustración 38: Propiedades del componente sensor de colisión de RobotStudio

Sensor plano

Un sensor plano en RobotStudio es un componente inteligente diseñado para la detección de la presencia o el paso de objetos dentro de un plano bidimensional específico dentro de nuestra estación robótica. A través de la simulación de la activación de señales, el sensor plano nos permitirá representar las interacciones en el entorno entre nuestro robot y nuestro

instrumento, considerando su posición en un plano, en este caso el sensor se encontrará situación sobre la superficie de la tecla del piano en cuestión, en este primer caso de la nota Do.

Características principales:

1. Área de detección definida: El sensor plano tiene un área rectangular como nuestras diferentes teclas, que será activado o desactivado según la presencia de objetos.

2. Señales de activación:
    a. Entrada: ObjectDetected (si se detecta un objeto dentro del área del sensor)
    b. Salida: Al activarse el sensor, puede enviar una señal como SensorActive = True, la cual puede ser utilizada por el robot para tomar decisiones.

3. Configuración en el entorno virtual: El componente se posicione en un plano, y la orientación del sensor puede ajustarse para que se alinee con la geometría de la celda o el entorno del robot.

4. Uso de la lógica de detección: El sensor puede programarse para responder a eventos específicos, como el paso de un objeto a través de su área de detección o la presencia continua de un objeto dentro del área.

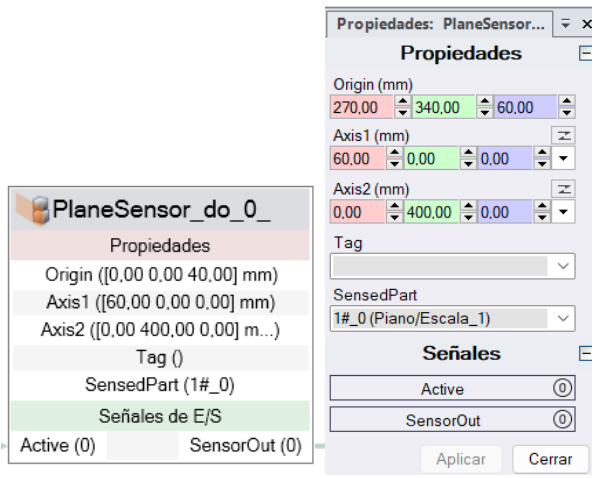


Ilustración 39: Propiedades del componente inteligente sensor plano de RobotStudio

Linear Move

Es una instrucción de movimiento que ordena al robot trasladar su herramienta desde un punto a otro siguiendo una línea recta en el espacio tridimensional con una velocidad constante y sin desviarse de esa trayectoria.

Características principales:

1. Mantiene la orientación de la herramienta constante (a menos que se especifique lo contrario).
2. Ideal para operaciones donde es crucial mantener una trayectoria exacta.
3. Es más exigente en términos de cálculo que un movimiento punto a punto (MoveJ), porque debe seguir el espacio cartesiano exacto.

Para ello emplearemos uno para mover linealmente hacia abajo la tecla al ser pulsada.

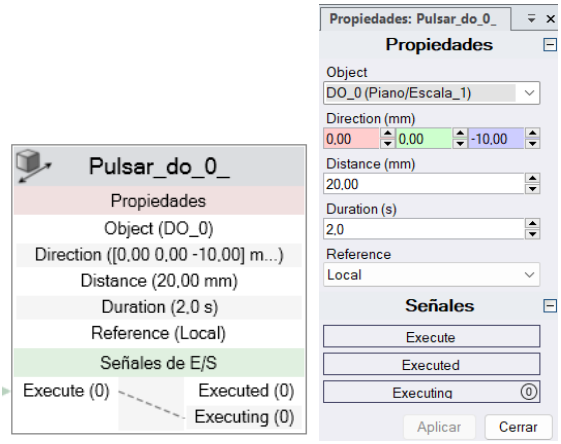


Ilustración 40: Propiedades del componente movimiento linear de RobotStudio

Una vez hemos desplazado el componente, lo devolvemos a su posición inicial.

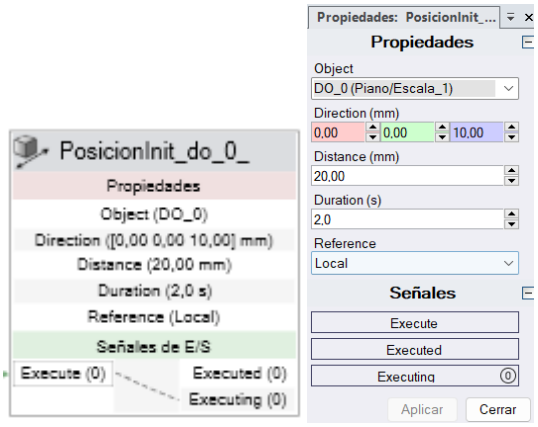


Ilustración 41: Propiedades del componente movimiento linear de RobotStudio

Logic Gate

Para que se puedan realizar las dos trayectorias del objeto de forma consecutiva y no de simultánea, le añadiremos una puerta lógica.

Es un bloque lógico virtual que se utiliza dentro del editor de comportamientos de un componente inteligente para procesar señales de entrada y producir salidas basadas en condiciones booleanas como AND, OR, NOT, entre otras.

Su función principal es permitir la construcción de lógicas condicionales internas, sin necesidad de escribir código RAPID, facilitando el diseño de respuestas automáticas en la simulación.

Tipos de puerta:

1. AND: Solo da salida True si todas las entradas son True
2. OR: Da salida True si al menos una entrada es True
3. NOT: Invierte el valor lógico de la entrada
4. XOR: True si solo una de las entradas es True

Para poder devolver la tecla a su posición inicial, emplearemos una puerta lógica NOT que nos permitirá saber cuando la nota puede volver a su posición inicial.

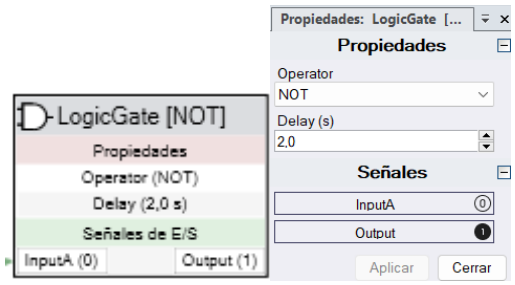


Ilustración 42: Propiedades del componente inteligente puerta lógica NOT de RobotStudio

Play Sound

Además de los movimientos vamos a añadirle sonido y color para una mayor iteración con la estación usando un Sound Asset.

Es un archivo de sonido (normalmente .wav) que se puede importar y vincular a un componente inteligente para que se reproduzca automáticamente en respuesta a una señal o evento específico durante la simulación.

Características clave:

1. Formato compatible: Generalmente .wav, aunque puede aceptar otros formatos básicos.
2. Reproducción controlada: Puedes vincular el sonido a eventos como una señal de entrada, una colisión, una animación o una condición lógica.
3. Ubicación sonora: El sonido se puede reproducir desde la ubicación del componente, simulando una fuente de sonido localizada.
4. No afecta el programa RAPID: El uso de sonidos es únicamente para efectos de simulación visual y auditiva, no tiene.

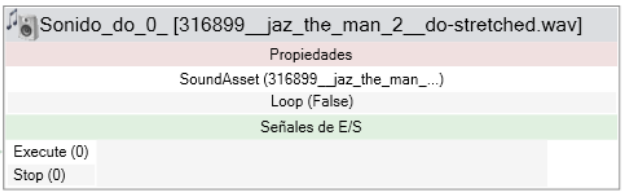


Ilustración 43: Propiedades del componente inteligente reproducir sonido de RobotStudio

Para añadir el sonido correspondiente a cada nota, tendremos que ir a la pestaña componer, y añadir el archivo manualmente, en este caso un .wav.

Para poder añadir el sonido en las propiedades del componente, tendremos que previamente tener en nuestro dispositivo el sonido en formato.wav. En archivos añadiremos el sonido de la nota.

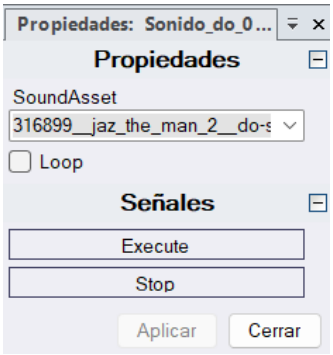


Ilustración 44: Ajuste de propiedades del componente inteligente reproducir sonido de RobotStudio

Set Color

Es un bloque de acción en RobotStudio que se utiliza dentro del componente inteligente para modificar dinámicamente el color de una parte del modelo, simulando cambios de estado visuales, en nuestro caso le dará un color verde a la tecla que sea pulsada por el robot durante la simulación.

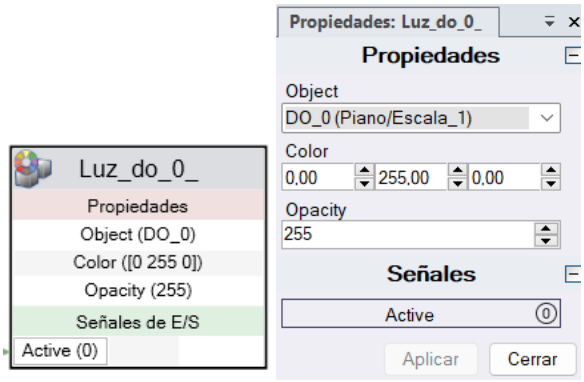


Ilustración 45: Propiedades del componente inteligente configurar color de RobotStudio

En este caso solo tendremos una entrada ON y ninguna salida, y lo conformarán los elementos mostrados en la imagen.

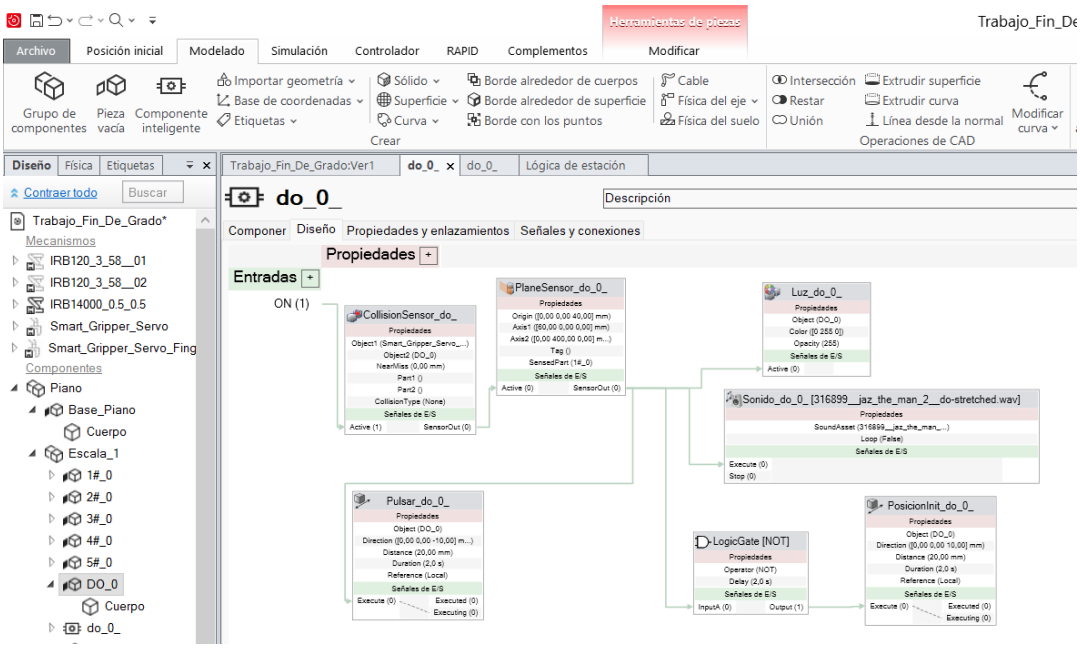


Ilustración 46: Lógica del componente inteligente nota do del teclado del RobotStudio

Ahora ya tenemos el teclado modelado, que estará formado por tres escalas en total.

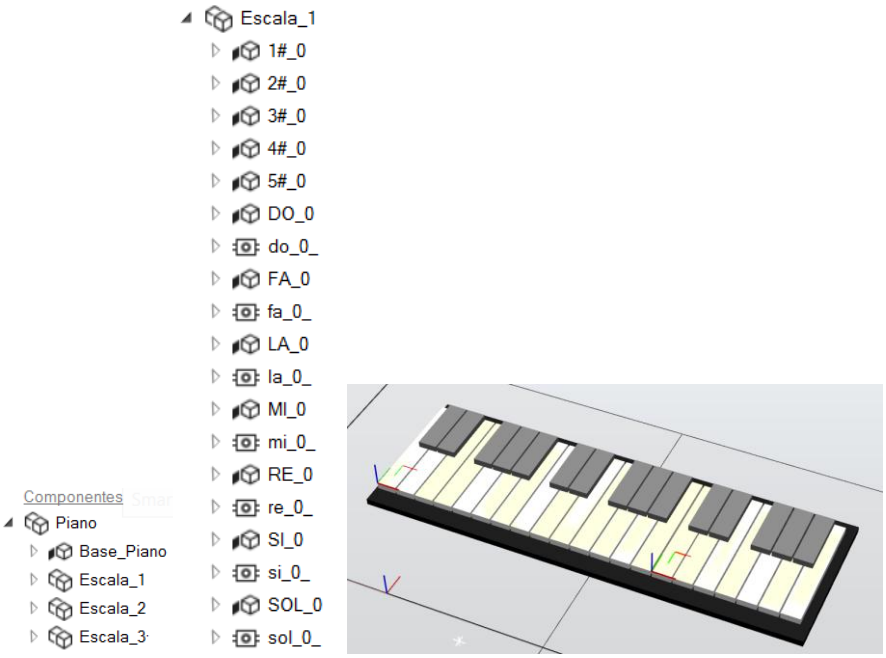


Ilustración 47: Estructuración y componentes del piano de RobotStudio

Nuestro piano estará formado por la base y luego de sus tres escalas, donde cada escala contará con siete notas blancas y cinco alteraciones, como se muestran en la imagen.

Configurado ya el teclado configuramos el componente pinza para convertirlo en un mecanismo que podremos usar durante la simulación como herramienta.

Un mecanismo en RobotStudio es una entidad cinemática simulada compuesta por varias partes conectadas mediante juntas (joints) que permiten movimientos definidos, como traslación o rotación, en respuesta a señales, comandos o condiciones dentro del entorno virtual. Los mecanismos se utilizan para representar y controlar componentes móviles de máquinas o sistemas automatizados, como pistones, compuertas, prensas, etc.

Dentro de Modelado >> Mecanismo >> Crear mecanismo.

Ilustración 48: Creación de un mecanismo dentro de la ventana de modelado de RobotStudio

Primero le daremos nombre al mecanismo, en nuestro caso lo vamos a llamar pinza y definimos el tipo de mecanismo como herramienta.

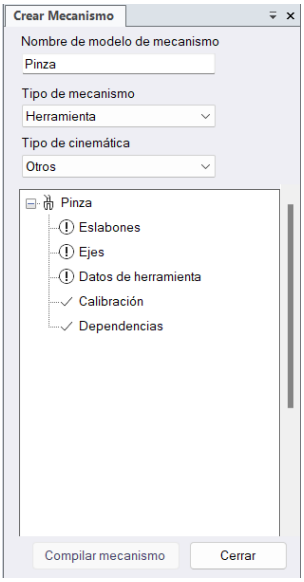


Ilustración 49: Creación de un mecanismo en RobotStudio

Definimos las distintas partes del mecanismo.

Ilustración 50: SmartArt de las partes que conforman un mecanismo en RobotStudio

Vamos a empezar por los eslabones.

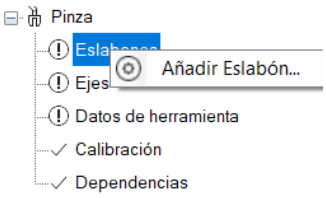


Ilustración 51: Creación de un eslabón de un mecanismo en RobotStudio

Primero elegiremos el cuerpo, ya que será nuestro eslabón base.

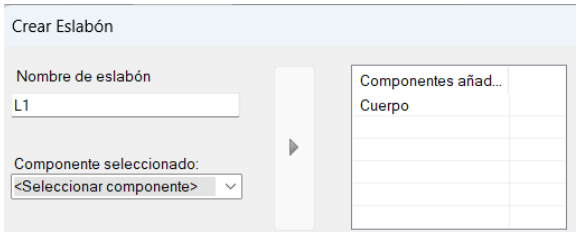


Ilustración 52: Propiedades para la definición de un eslabón en RobotStudio

Realizaremos lo mismo con el resto de los eslabones que conforman nuestra pinza, la pinza izquierda y la pinza derecha, y el cuerpo.

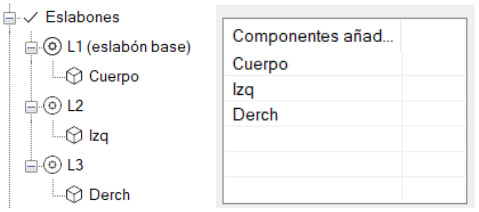


Ilustración 53: Eslabones que conforman una pinza inteligente en RobotStudio

Añadimos los ejes, el eje 1 donde el eslabón principal va a ser el eslabón base, nuestro cuerpo, y como eslabón segundario va a ser la pinza izquierda. Queremos que dicha articulación sea lineal, por lo que vamos a seleccionar tipo de eje prismático.

Como se trata del eslabón izquierdo, se tiene que mover a la derecha, por lo que los valores que daremos a las posiciones serán 0 y 1 positivo.

Le tenemos que definir también el recorrido, por lo que lo fijaremos en 0 y 25 mm.

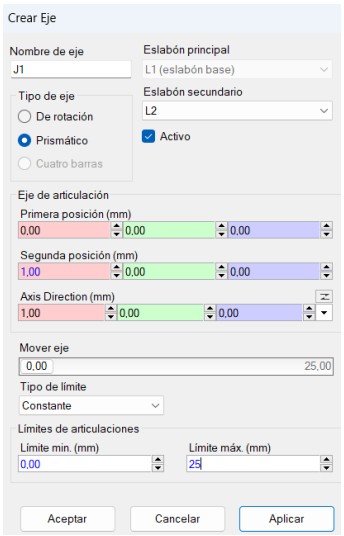


Ilustración 54: Eje 1 del mecanismo pinza de RobotStudio

Realizamos lo mismo con nuestro eslabón principal y el tercer eslabón (Pinza derecha) pero cambiando los límites, donde nuestro límite inferior será -25 y el máximo será 0.

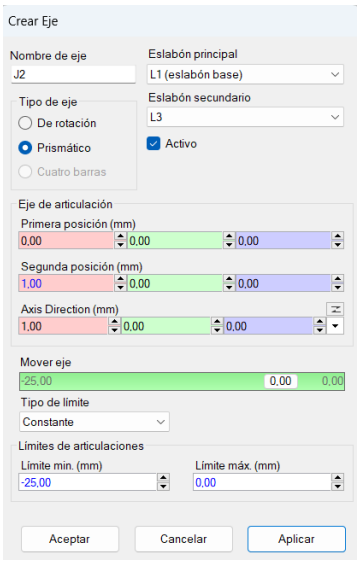


Ilustración 55: Eje 2 del mecanismo pinza de RobotStudio

Aplicados todos los cambios ya tendremos definidos todos los ejes de nuestro mecanismo

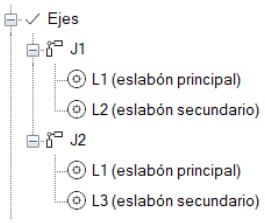


Ilustración 56: Ejes del mecanismo pinza de RobotStudio

Para configurar los datos generales de la herramienta Pinza le definiremos una masa de 1 Kg con centro de gravedad en las siguientes coordenadas (0,0,50).

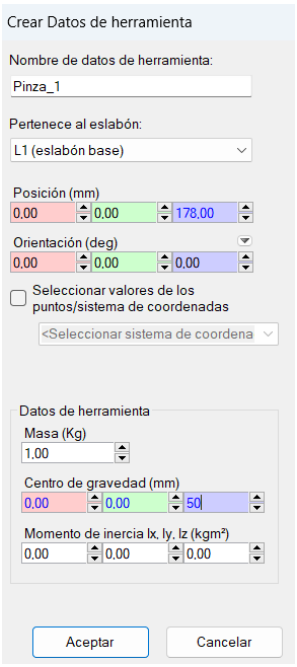


Ilustración 57: Propiedades de los datos de herramienta del mecanismo pinza de RobotStudio

Una vez la completada la configuración de nuestra pinza, compilamos nuestra herramienta.

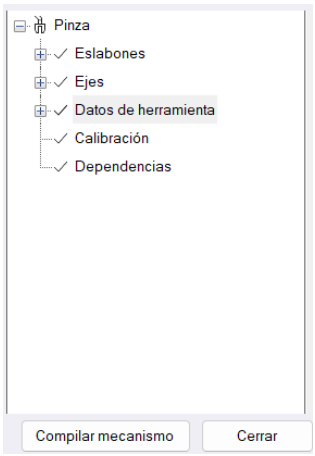


Ilustración 58: Compilación mecanismo pinza de RobotStudio

Ya tenemos nuestra pinza definida y modelada como podemos ver en la imagen a continuación con su TCP.

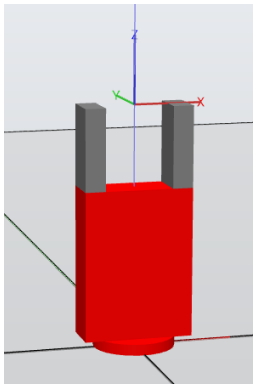


Ilustración 59: Herramienta pinza de RobotStudio

Ahora vamos a añadir dos posiciones: Pinza abierta y Pinza cerrada.

Para ello vamos a crear una pose Pinza_abierta y definimos valores de los ejes.

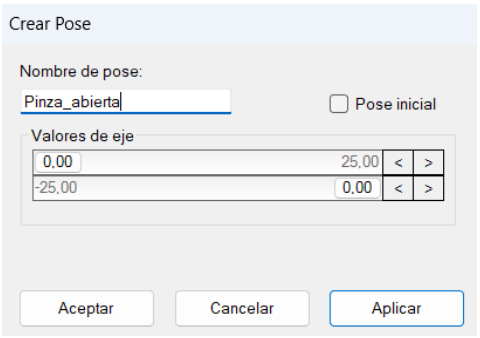


Ilustración 60: Creación Pose de la posición pinza abierta de RobotStudio

Cuando tenemos la primera posición, haremos lo mismo para la posición de pinza cerrada.

Ahora vamos a definir la lógica inteligente de la pinza.

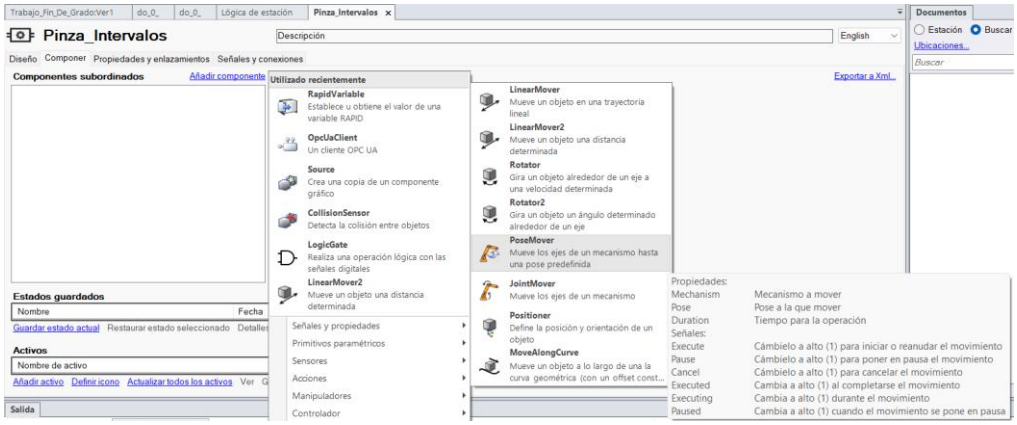


Ilustración 61: Composición pinza de RobotStudio

Pose Mover

El Pose Mover es un componente inteligente (Smart Component) que permite que el TCP de un robot se mueva a posición específica, en nuestro caso serán las poses definidas como pinza_abierta y pinza_cerrada.

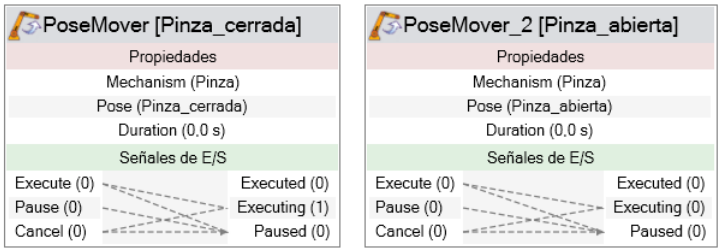


Ilustración 62: PoserMovers del mecanismo pinza de RobotStudio

Para poder completar la lógica de nuestro componente, vamos a añadir unas puertas lógicas.

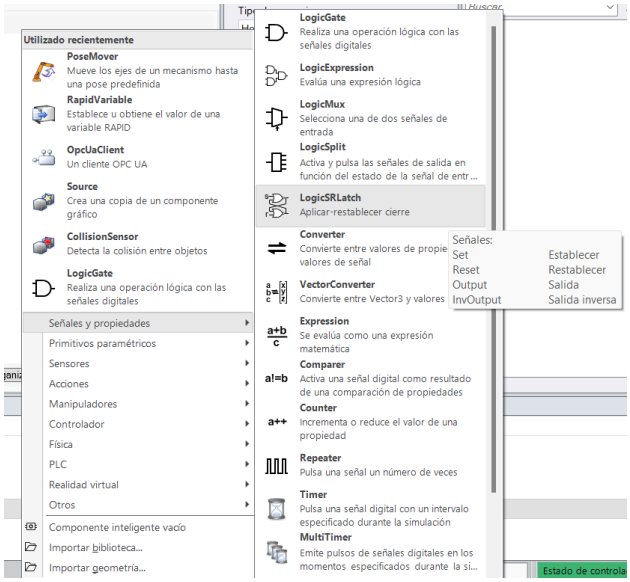


Ilustración 63: LogicSRLatch del mecanismo pinza de RobotStudio

La puerta lógica usada es NOT

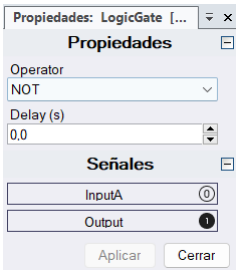


Ilustración 64: Puerta lógica NOT del mecanismo pinza en RobotStudio

El tipo de señal que emplearemos será un DigitalOutput funcionarán como nuestras dos salidas de la pinza. Las llamaremos Dos_Notas y Una_Nota.

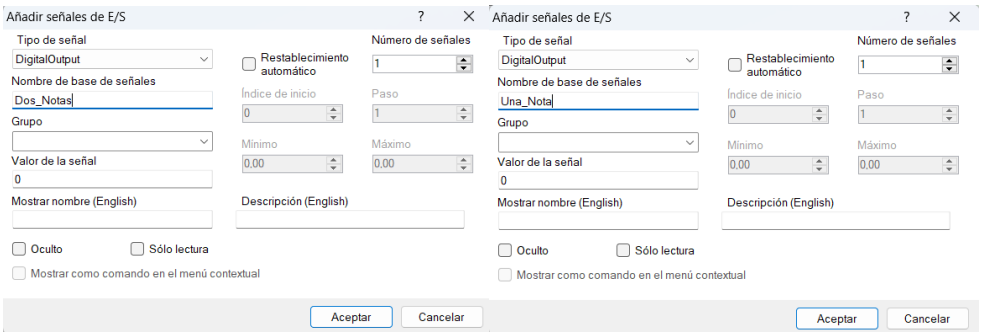


Ilustración 65: DigitalOutputs del mecanismo pinza de RobotStudio

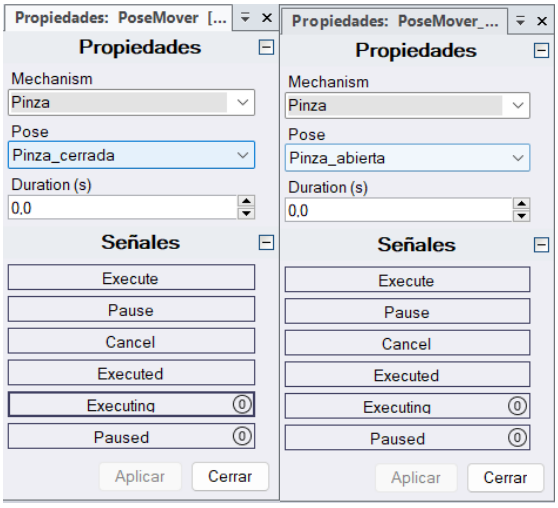


Ilustración 67: Propiedades de los DigitalOutputs del mecanismo pinza de RobotStudio

La señal de entrada será única, y la vamos a denominar Pulsador. Mediante la activación de este pulsador vamos a ser capaces de cambiar la posición de la pinza, a la deseada por el usuario.

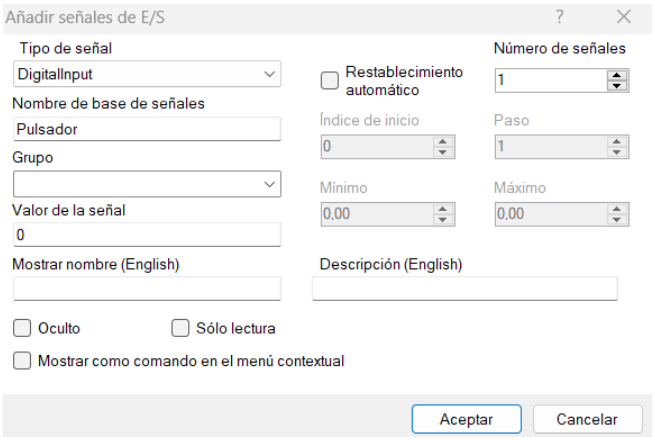


Ilustración 668: Entrada digital pulsador del mecanismo pinza de RobotStudio

Finalmente obtendremos la siguiente Pinza de intervalos.

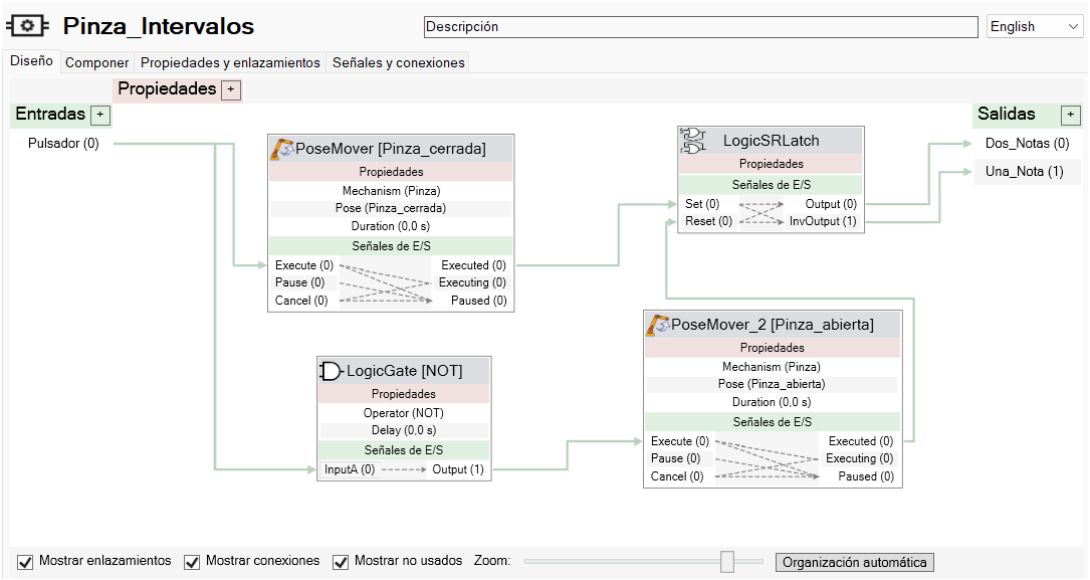


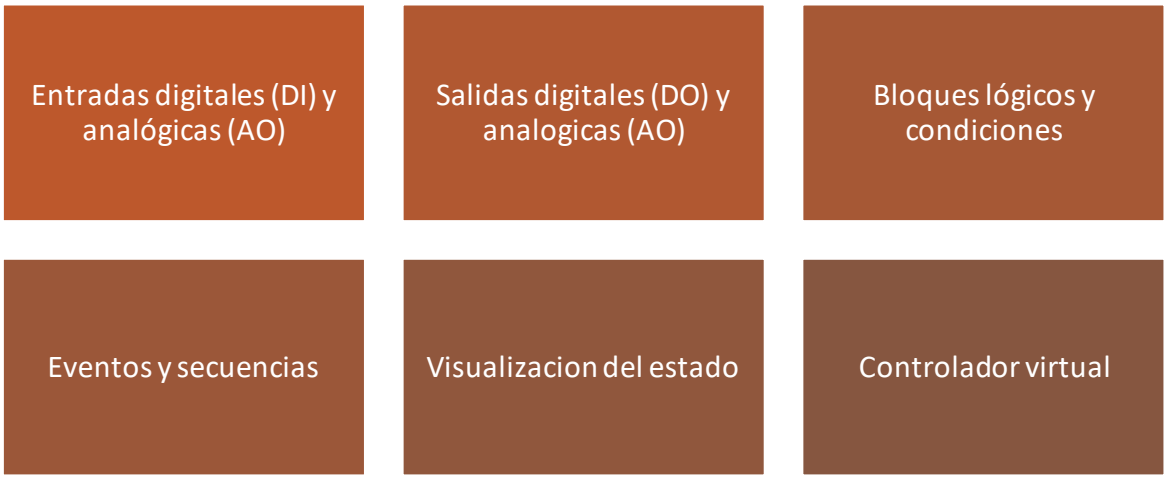
Ilustración 679: Diseño y lógica del mecanismo pinza de RobotStudio

4.1.1.4 Lógica de la estación

En RobotStudio, la lógica de la estación (Station Logic) es una herramienta que permite definir y visualizar el comportamiento lógico de los dispositivos y elementos de una celda robótica virtual. Funciona como un entorno de control basado en señales, en el que se establecen las relaciones entre entradas, salidas, sensores, actuadores y el robot simulado.

El objetivo principal de esta herramienta es coordinar la interacción del robot con los periféricos de la estación virtual, garantizando que la simulación represente fielmente las condiciones que existirán en el entorno físico.

Elementos que se muestran en la lógica de una estación:



Señales digitales y analógicas

- Entradas digitales (DI) y analógicas (AI): simulan la información de sensores, finales de carrera o detectores de pieza.
- Salidas digitales (DO) y analógicas (AO): controlan actuadores como pinzas, cilindros neumáticos, bandas transportadoras o sistemas de iluminación.

Bloques lógicos y condiciones

- Se representan mediante diagramas de lógica (puertas AND, OR, NOT).
- Se utilizan temporizadores, comparadores y reglas condicionales (if/then).

Eventos y secuencias

- Se definen reglas que desencadenan acciones: por ejemplo, "si un sensor detecta pieza → activar pinza".
- Permite sincronizar el ciclo de trabajo entre robot y periféricos.

Visualización del estado de la celda

- Se muestran animaciones en tiempo real, como la activación de un cilindro, la apertura de una garra o el movimiento de una cinta transportadora.

- Los cambios en las señales se reflejan con indicadores gráficos que facilitan la depuración.

Interacción con el controlador virtual

- La lógica de la estación puede conectarse al controlador virtual del robot, lo que permite que el programa RAPID y los periféricos funcionen en conjunto.

- Obtención de una simulación completa que reproduce el comportamiento del sistema físico antes de su implementación.

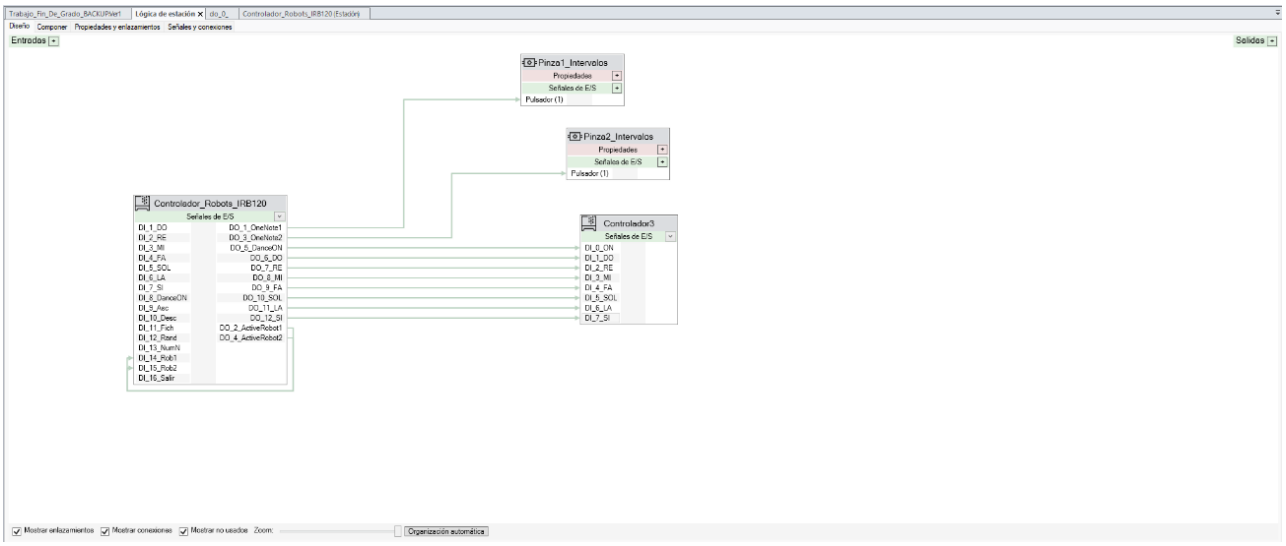


Ilustración 68: Diseño de la lógica de la estación de modelado de RobotStudio

4.1.2 Trayectorias y puntos

Workobject

Un workObject define un sistema de coordenadas local (un marco de referencia) respecto al cual se pueden programar las posiciones del robot

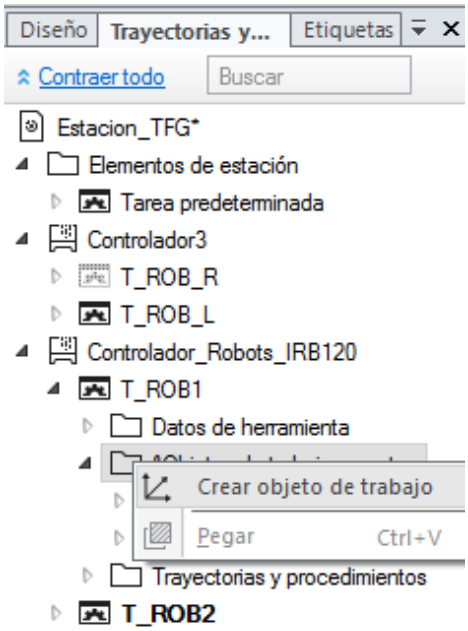


Ilustración 69: Creación de un objeto de trabajo en RobotStudio

Características:

- Representa un sistema de coordenadas móvil o fijo.
- Permite programas en términos relativos a una pieza, herramienta, otra estructura.
- Facilita la reutilización de programas sin modificar coordenadas absolutas.
- Incluye tanto la posición y orientación del origen del sistema, como una posible base móvil.

Tipos:

- Fijo: Coordenadas relativas a la celda del robot.
- Móvil: Asociado a un objeto en movimiento .

Nuestros workobjects serán los distintos teclados sobre los que trabajarán los IRB 1200.

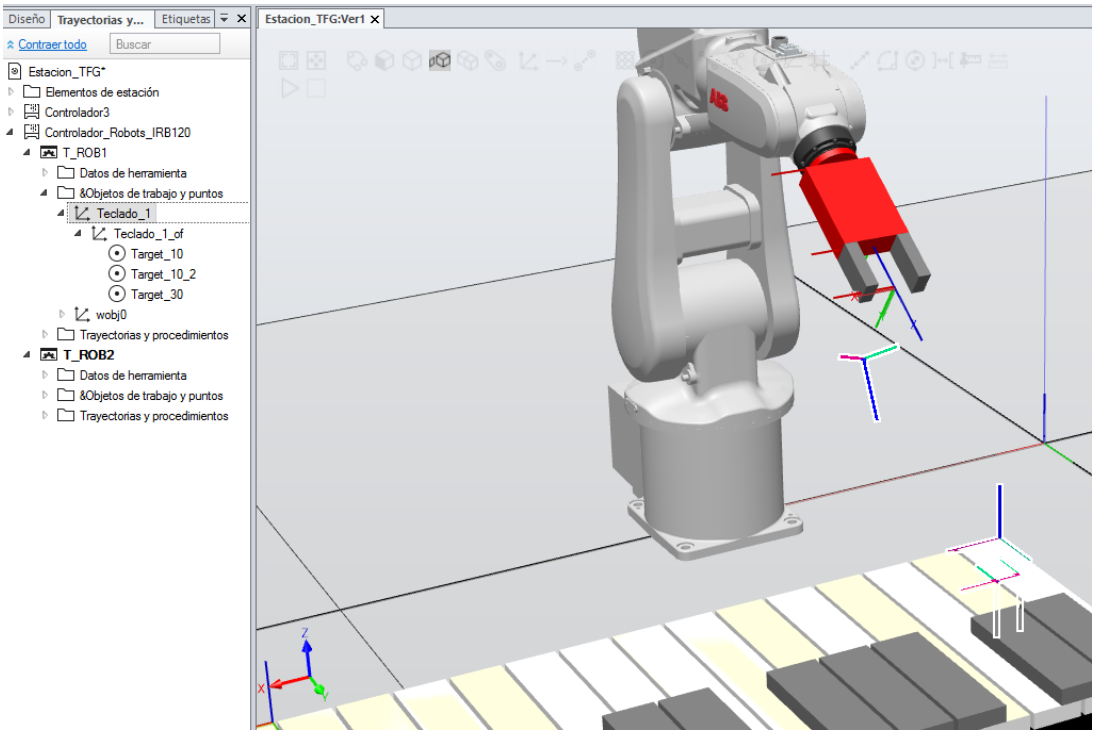


Ilustración 70: Workobjects de la estación de modelado de RobotStudio

Su definición en RAPID viene dada por los siguientes parámetros:

```
!Workobjects
TASK PERS wobjdata Teclado_1:=[FALSE,TRUE,"",[[340,230,60],[0.707107,0,0,-0.707107]],[[0,0,0],[1,0,0,0]]];
```

Ilustración 71: Definición del Workobject Teclado del módulo ModulePiano de RAPID

Los parámetros que lo definen son los siguientes:

- FALSE: El sistema de coordenadas del WorkObject está definido respecto al World (Mundo, Sistema global del robot).
- TRUE: El objeto se mueve con la pieza.
- " ": Nombre del userframe al que se referencia. Si está vacío el sistema de referencia será el World.
- Posición y orientación del User Frame respecto al sistemta mundial (World). Se define como un robtarget.

- Origen del objeto de trabajo (WorkObject Frame) dentro del uframe.

Robtarget

Un robtarget es una posición y orientación completa en el espacio caartesiano (X; Y, Z + orientación del TCP), junto con información adicional necesaria para ejecutar movimientos

Suele ser el tipo de dato más empleado para la definición de posiciones que tiene como objetivo el robot poder alcanzar.

Estos serán nuestros robtargets sobre el teclado_1.

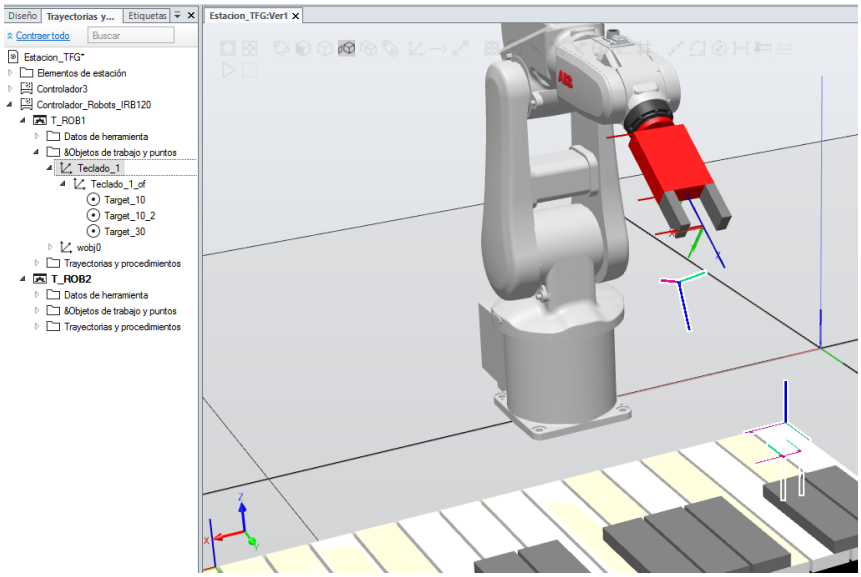


Ilustración 72: Robtargets del objeto teclado de la estación de RobotStudio

Uno de los robtargets empleados posteriormente en RAPID:

```
!Punto de partida de la primera escala
CONST robtarget Target_10:=[[30,100,0],[0,1,0,0],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Target_10_2:=[[62.5,100,0],[0,1,0,0],[0,0,-1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Ilustración 73: Definición de Robtargets del módulo ModulePiano de RAPID

Consta de 4 partes [Pos, Orient, ConfJ, ConfL]:

1. Pos (posición):
   Coordenadas cartesianas [X, Y, Z] en mm, respecto al sistema de referencia activo (por ejemplo, World, Tool, o Workobject)

2. Orient (orientación):
   Cuaterno que representa la orientación del TCP (herramienta). Se trata de un cuaterno unitario, ya que es más estable que los ángulos de Euler.

3. ConfJ (Configuración articular):
   Configuración del robot para alcanzar esa posición, con opción a múltiples configuraciones posibles.

   - Cf1 → Estado del eje 1 (frente/detrás)
   - Cf4 → Orientación de la muñeca (arriba/abajo)
   - Cf6 → Giro de la muñeca
   - Cfx → Configuración externa (ejes externos o séptimo eje)

4. ConfL (Configuración de la orientación del eje 6 / posición externa)
   Información adicional para movimientos con ejes externos o redundancia.

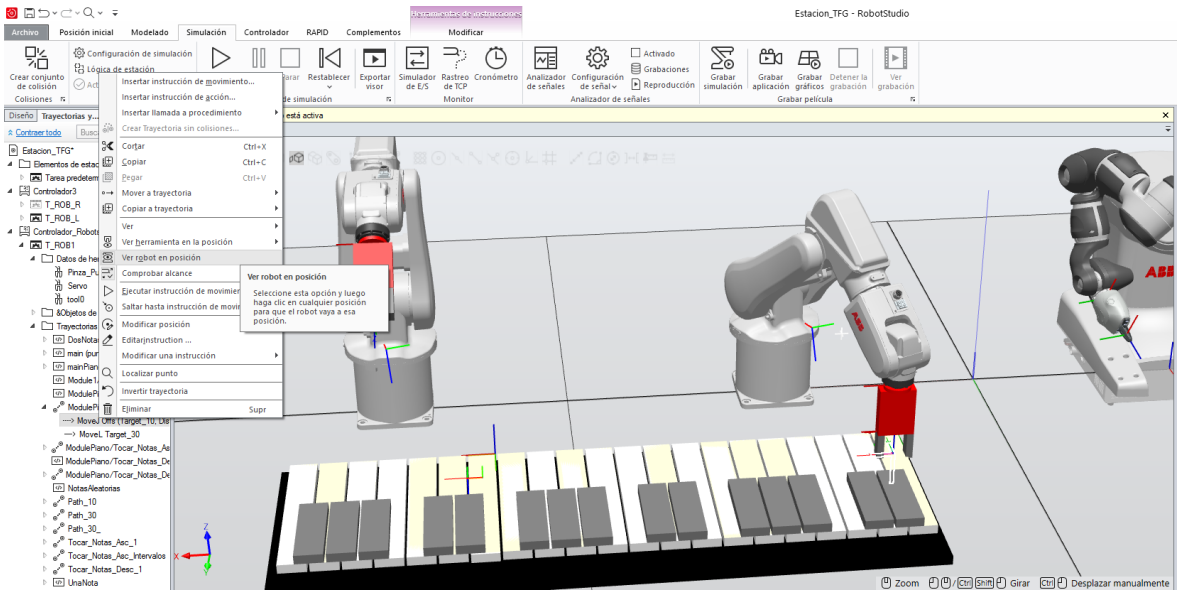


Ilustración 74: Robot IRB 1200 en posición del robtarget

Joint tarjets

Se define joint target como su propio nombre indica a la posición o target final queremos alcanzar posicionan y definiendo las posiciones de las juntas o articulaciones del robot para poder llegar a ella.

JointTarget es un tpo de posición definida por los ángulos individuales de cada eje del robot. En lugar de especificar una posición en el espacio cartesiano (X; Y; Z) defines directamente la configuración de las articulaciones del robot.

Características:

- Específica para un robot particular (ya que depende de su configuración de ejes).
- No se ve afectado por el sistema de coordenadas del WorkObject.
- Utilizado cuando se necesita precisión en las posiciones articulares, por ejemplo:
    1. Posiciones de referencia de seguridad.
    2. Movimiento de aproximación en espacios reducidos.

Estos son nuestros jointtargets:

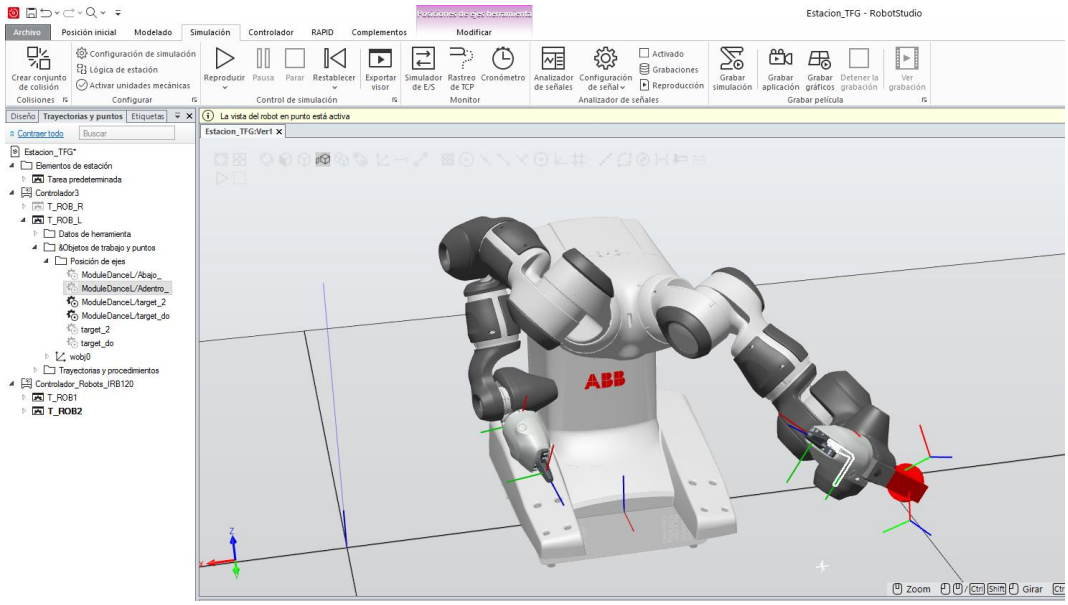


Ilustración 75: Robot 1400 YuMi en posición del jointtarget

Algunos de nuestros jointtargets en nuestros módulos de RAPID:

```
LOCAL CONST jointtarget target_do := [[-133.471326165,-46.983870968,-89.218637993,11.433691756,16.494623656,0.82078853],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
LOCAL CONST jointtarget target_2 := [[-133.471326165,-50.335125448,-61.501792115,11.433691756,16.494623656,0.82078853],[0,9E+09,9E+09,9E+09,9E+09,9E+09]];
LOCAL CONST jointtarget Abajo_ := [[-82.740143369,-84.517921147,-8.985663082,13.512544803,44.035842294,0.82078853],[33.216845878,9E+09,9E+09,9E+09,9E+09,9E+09]];
LOCAL CONST jointtarget Adentro_ := [[-105.689964158,-73.123655914,-8.985663082,13.512544803,44.035842294,0.82078853],[33.216845878,9E+09,9E+09,9E+09,9E+09,9E+09]];
LOCAL CONST jointtarget Afuera_ := [[-41.672043011,-73.123656915,-8.985663683,13.512544018,44.035838089,0.820788475],[33.216845878,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Ilustración 76: Definición de los jointargets de nuestro módulo DanceL de RAPID

Está compuesto por dos listas de valores:

1. Ejes del robot principal (j1–j6):
    - Representan los ángulos de cada articulación del robot, normalmente en grados.

- El número depende del robot (los de 6 ejes tienen j1–j6).

2. Ejes externos (e1–e6):

   - Representan los valores de los ejes adicionales que puedan estar conectados al controlador (por ejemplo, un séptimo eje lineal, una posición de mesa rotativa o un *track*).

   - Si no hay ejes externos, estos valores suelen ser 0.

Path

Un path (camino o trayectoria) en robótica es la forma en que el robot se mueve de un punto A a un punto B. Define el tipo de movimiento que el robot debe seguir para llegar a una posición destino.

El path determinara cómo el robot se mueve entre dos robtarget o jointtarget, elementos mencionados previamente, el tipo de movimiento que va a realizar, una línea o una curva, y el nivel de precisión o suavidad del movimiento.

Tipos:

- MoveL (Lineal) --> Movimiento lineal del TCP, sigue una línea recta entre puntos. Es ideal para aplicaciones que necesitan precisión en la trayectoria.

- MoveJ (Articular) --> El robot mueve sus articulaciones para llegar rápido a la posición destino, no garantiza trayectoria lineal en el espacio. Es más rápido y eficiente para moverse entre posiciones distantes.

- MoveC (Circular) --> El robot sigue un arco o curva circular, usando un punto intermedio para definir la curva.

Nosotros lo hemos empleado para alcanzar las siguientes posiciones:

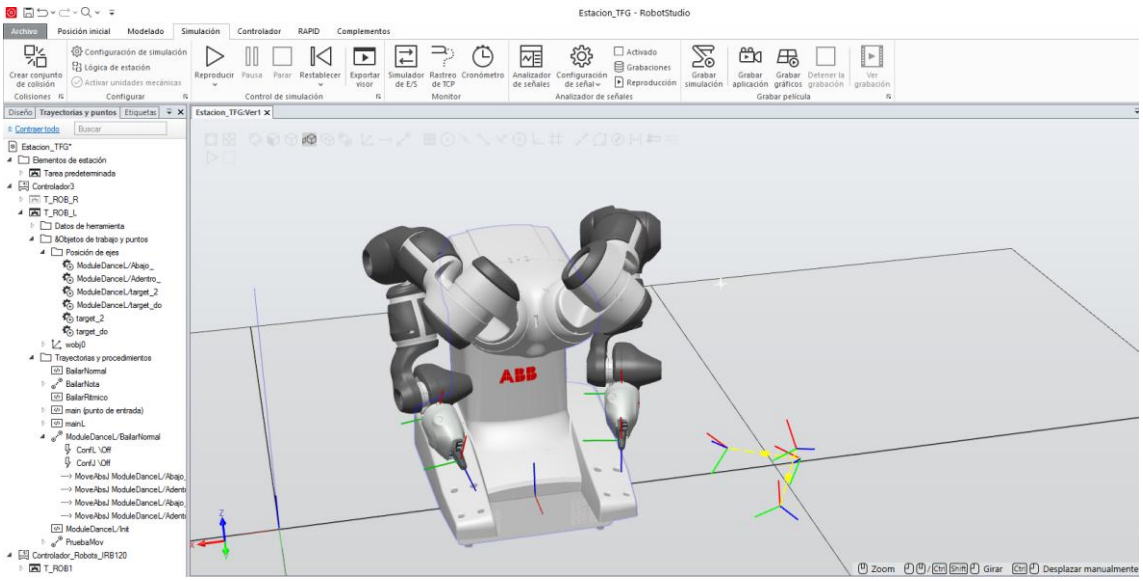


Ilustración 77: Path mostrado en nuestra estación de RobotStudio

4.1.3 RAPID

RAPID es el lenguaje de programación propietario desarrollado por ABB Robotics para sus controladores de robots industriales. Se introdujo junto con el controlador S4 en 1994 y desde entonces se ha consolidado como el estándar de programación en los sistemas de ABB (ABB, 2004-2017).

Se trata de un lenguaje de programación de alto nivel, diseñado específicamente para:

- Controlar la cinemática y el movimiento de robots industriales ABB.
- Gestionar entradas/salidas digitales y analógicas.
- Coordinar rutinas, ciclos y tareas complejas de producción.
- Integrar periféricos y sistemas externos dentro de una celda robótica.

Características principales:

- Sintaxis estructurada similar a lenguajes como Pascal o BASIC, lo que lo hace accesible para programadores.
- Estructuración modular, con procedimientos, funciones y rutinas.
- Manejo de movimientos robóticos, mediante instrucciones predefinidas como MoveJ (movimiento articular), MoveL (movimiento lineal) o MoveC (movimiento circular).
- Soporte de multitarea, lo que permite ejecutar rutinas paralelas para manejar periféricos o procesos en segundo plano.
- Compatibilidad con simulación y control real, ya que el mismo código se puede ejecutar en un controlador virtual (RobotStudio) o en un controlador físico del robot.

Usos principales:

- Programación de trayectorias y movimientos: definir la forma en que el robot se desplaza en el espacio de trabajo.
- Control de dispositivos externos: como pinzas, sensores, cintas transportadoras, mediante señales de E/S.
- Automatización de procesos: soldadura, ensamblaje, pintura, paletizado, entre otros.
- Simulación offline: probar y validar programas en RobotStudio sin necesidad de detener la producción real.
- Interacción con sistemas externos: a través de comunicación por buses de campo (Ethernet/IP, DeviceNet, Profibus, etc.).

4.1.3.1 Importación de los datos

Para poder emplear los puntos y trayectorias, definidas previamente, en nuestros módulos de RAPID tendremos que realizar la importación de estas

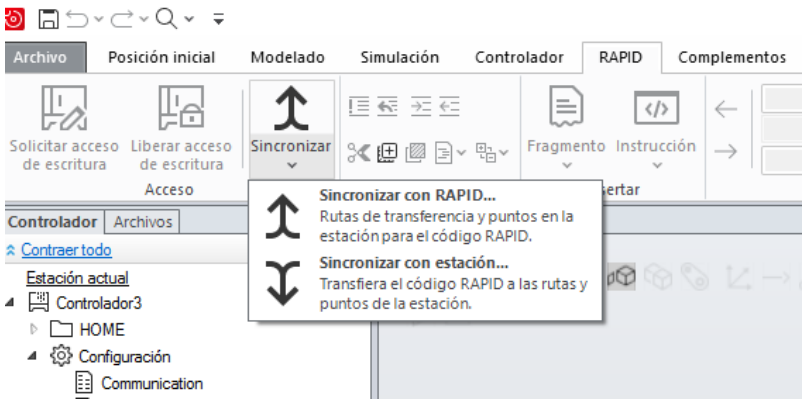


Ilustración 78: Formas de sincronización con estación y RAPID en RobotStudio

Sincronización con RAPID

La opción *Sincronizar con RAPID* en RobotStudio se utiliza cuando se requiere mantener actualizado el código RAPID entre el proyecto de simulación y el controlador virtual o físico. Esta sincronización garantiza que los módulos, procedimientos y variables RAPID cargados en el controlador se reflejen fielmente en el entorno de RobotStudio. Resulta especialmente útil cuando se realizan modificaciones directamente desde el FlexPendant o desde el propio controlador, como ajustes de rutinas, depuración de programas o cambios en variables persistentes, y posteriormente se desea que dichos cambios queden registrados en el proyecto de simulación. De esta forma se asegura la coherencia del programa entre el entorno de desarrollo y el controlador que ejecuta el robot (ABB, 2019).

Sincronización con estación

Por otro lado, la opción *Sincronizar con Estación* está orientada a la coherencia entre el modelo 3D de la celda virtual en RobotStudio y el controlador virtual asociado. Con este método se sincronizan aspectos físicos y de configuración, tales como los datos de herramientas (ToolData), marcos de referencia (WObjData), trayectorias, posiciones y configuraciones de los robots dentro del entorno de simulación. Es la opción adecuada cuando se realizan modificaciones en la estación 3D, como mover un objeto de trabajo, crear una nueva herramienta o reconfigurar la disposición de la celda, y se quiere garantizar que el controlador refleje exactamente la misma configuración. De esta forma, la simulación y la realidad virtual del sistema se mantienen consistentes, minimizando discrepancias durante la ejecución real (Kihlman, 2017; ABB, 2020).

4.1.3.2 Generación de módulos

En RAPID, un módulo es un archivo o bloque de código que agrupa funciones, rutinas, procedimientos y declaraciones de datos relacionadas, con el objetivo de organizar el programa y facilitar la reutilización del código en diferentes partes del programa o en distintos proyectos. Cada módulo puede contener procedimientos (PROC), que son bloques de instrucciones ejecutables, funciones (FUNC), que devuelven un valor, así como variables y constantes. Además, los módulos se pueden llamar desde otros módulos, lo que permite estructurar programas complejos y facilita el mantenimiento y la lectura del código al dividir un programa grande en partes más manejables.

Módulos del Controlador 3

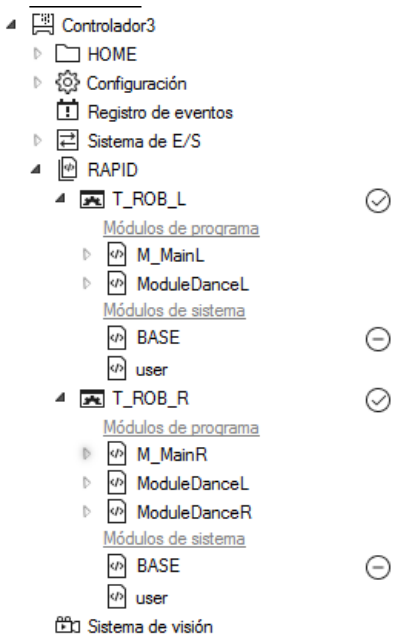


Ilustración 79: Desglose de los módulos de RAPID de nuestro Controlador3

Módulos del Controlador Irb120

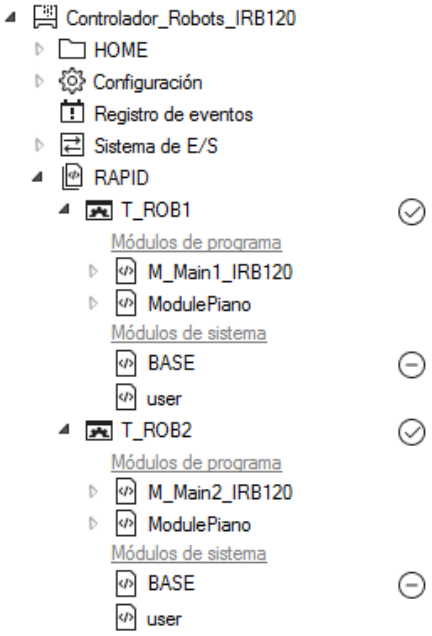


Ilustración 80: Desglose de los módulos de RAPID de nuestro Controlador_Robots_IRB120

4.1.3.3 Definición de variables

Variables globales

Una variable global es una variable que puede ser accedida y modificada desde cualquier rutina, procedimiento o módulo del programa, siempre y cuando se haya declarado con ese alcance (ABB, 2004-2017) .

Características principales de las variables globales en RAPID:

- Se declaran fuera de procedimientos o funciones, normalmente al inicio de un módulo o en un sistema de datos común.
- Están disponibles en todo el módulo donde fueron declaradas (y en otros, si se definen como *PERS* o mediante módulos de sistema compartidos).
- Su valor puede permanecer constante entre ejecuciones si se declaran con la palabra clave PERS (persistent).
- Si no son PERS, su valor se reinicia cada vez que se inicia el programa o se reinicia el controlador.
- Son útiles para almacenar estados, configuraciones o parámetros que se necesitan en varios lugares del programa.

```
!Variables del main
PERS bool nosalir;
PERS num opcion;
PERS num notaPulsada;

!Variable generales
PERS bool ActiveRobot1;
PERS bool ActiveRobot2;
PERS bool Dos_Notas := FALSE;
```

Ilustración 81: Variables globales empleadas en el módulo ModulePiano del Controlador_Robots_IRB120 de RAPID

Variables locales

Las variables locales son aquellas que Oslo existen dentro un de un procedimiento (PROC), o función (FUNC).

Características de las variables locales en RAPID:

- Se declaran dentro de una rutina (PROC o FUNC).

- Solo pueden ser usadas dentro de esa rutina; no son visibles fuera de ella.
- Su valor se pierde cuando la rutina termina.
- Se utilizan para cálculos internos o datos temporales que no es necesario guardar ni compartir.
- Ocupan menos memoria y evitan conflictos con variables globales.

```
LOCAL PERS num seed:=10426;
```

Ilustración 82: Variable Local del módulo ModulePiano del Controlador_Robots_IRB120 de RAPID

Variables Constantes

Las variables constantes son aquellas que se definen con la palabra clave CONST.

Características de las constantes (CONST) en RAPID:

- Su valor no puede cambiar durante la ejecución del programa.
- Se usan cuando un dato es fijo y no debe modificarse.
- Mejoran la legibilidad y la seguridad del código, evitando cambios accidentales.
- Se pueden usar en expresiones, movimientos o cálculos.
- Pueden declararse a nivel global, general o incluso local dentro de un procedimiento.

```
!Distancia entre notas
CONST num DistNota:=65;


CONST num Nota_DO := 0;
CONST num Nota_RE := 1;
CONST num Nota_MI := 2;
CONST num Nota_FA := 3;
CONST num Nota_SOL := 4;
CONST num Nota_LA := 5;
CONST num Nota_SI := 6;
CONST num Nota_Do_Alto := 7;
```

Ilustración 83: Variables CONST del módulo ModulePiano del Controlador_Robots_IRB120 de RAPID

Variables Normales

Las variables declaradas con VAR son las más comunes y se consideran variables normales (no persistentes).

Características de las variables VAR en RAPID:

- Se pueden modificar libremente durante la ejecución del programa.
- Se reinician a su valor inicial (o 0 si no se les asigna nada) cada vez que se reinicia el programa o el controlador.
- Pueden declararse como:
    - Locales → dentro de un procedimiento (PROC) o función (FUNC), visibles solo ahí.
    - Globales → fuera de procedimientos, accesibles desde todo el módulo.
    - Generales → en un módulo de datos compartido, accesibles desde cualquier módulo.
- Son útiles para cálculos temporales, estados momentáneos o datos que no necesitan guardarse permanentemente.

```
!Variables para la lectura de ficheros
VAR string lineRead;
VAR string filePath := "/home/cancion.txt";
VAR string note;
VAR num i;
```

Ilustración 84: Variables no persistentes del módulo ModulePiano del Controlador_Robots_IRB120 de RAPID

4.1.3.4 Creación de las interrupciones

El procedimiento Init en RAPID se encarga de preparar las interrupciones que va a usar el robot. Primero borra cualquier configuración previa con IDelete para empezar desde cero. Luego conecta cada interrupción con una rutina especial llamada TRAP, que es el bloque de código que se ejecutará automáticamente cuando ocurra un evento. Finalmente, asigna cada entrada digital del robot (DI_1 a DI_16) a su interrupción correspondiente mediante ISignalDI, de manera que, cuando una de esas entradas se active, se dispare la rutina asociada.

```
LOCAL PROC Init()

    IDelete Idi0;
    IDelete Idi1;
    IDelete Idi2;
    IDelete Idi3;
    IDelete Idi4;
    IDelete Idi5;
    IDelete Idi6;
    IDelete Idi7;
    IDelete Idi8;
    IDelete Idi9;
    IDelete Idi10;
    IDelete Idi11;
    IDelete Idi12;
    IDelete Idi13;
    IDelete Idi14;
    IDelete Idi15;

    CONNECT Idi0 WITH trap_0_DO;
    CONNECT Idi1 WITH trap_1_RE;
    CONNECT Idi2 WITH trap_2_MI;
    CONNECT Idi3 WITH trap_3_FA;
    CONNECT Idi4 WITH trap_4_SOL;
    CONNECT Idi5 WITH trap_5_LA;
    CONNECT Idi6 WITH trap_6_SI;
    CONNECT Idi7 WITH trap_7_DanceON;
    CONNECT Idi8 WITH trap_8_EscalaAscendente;
    CONNECT Idi9 WITH trap_9_EscalaDescendente;
    CONNECT Idi10 WITH trap_10_Partitura;
    CONNECT Idi11 WITH trap_11_Aleatorio;
    CONNECT Idi12 WITH trap_12_UnaDosNotas;
    CONNECT Idi13 WITH trap_13_ROB1;
    CONNECT Idi14 WITH trap_14_ROB2;
    CONNECT Idi15 WITH trap_15_NOSALIR;


     ISignalDI DI_1_DO, 1, Idi0;
     ISignalDI DI_2_RE, 1, Idi1;
     ISignalDI DI_3_MI, 1, Idi2;
     ISignalDI DI_4_FA, 1, Idi3;
     ISignalDI DI_5_SOL, 1, Idi4;
     ISignalDI DI_6_LA, 1, Idi5;
     ISignalDI DI_7_SI, 1, Idi6;
     ISignalDI DI_8_DanceON, 1, Idi7;
     ISignalDI DI_9_Asc, 1, Idi8;
     ISignalDI DI_10_Desc, 1, Idi9;
     ISignalDI DI_11_Fich, 1, Idi10;
     ISignalDI DI_12_Rand, 1, Idi11;
     ISignalDI DI_13_NumN, 1, Idi12;
     ISignalDI DI_14_Rob1, 1, Idi13;
     ISignalDI DI_15_Rob2, 1, Idi14;
     ISignalDI DI_16_Salir, 1, Idi15;

ENDPROC
```

Ilustración 85: Función Init del módulo ModulePiano del Controlador_Robots_IRB120 de RAPID

Las rutinas TRAP son diferentes a los procedimientos normales porque no se llaman directamente dentro del programa, sino que se lanzan solas en cuanto sucede el evento que las activa. Esto permite que el robot responda al instante, interrumpiendo lo que estaba haciendo para ejecutar la acción indicada y, después, volver al punto donde lo dejó. Gracias a esto, el sistema puede reaccionar a señales externas en tiempo real, como un sensor, un pulsador o una orden de seguridad, sin necesidad de estar comprobando esas condiciones dentro del ciclo principal. Los TRAPs empleados son los siguientes:

Para las distintas notas musicales del teclado, hemos creado un TRAP para cada uno de los sonidos de la escala.

```
TRAP trap_0_DO

    Nota_a_tocar := Nota_DO;
    opcion := 5;

ENDTRAP
```

Ilustración 86: Rutina TRAP de la nota DO del módulo ModulePiano del Controlador_Robots_IRB120 de RAPID

Descripción del funcionamiento

1. Asignación de la nota: Cada TRAP establece el valor de la variable Nota_a_tocar con la nota correspondiente (DO, RE, MI, FA, SOL, LA, SI). Esta variable es utilizada posteriormente por el sistema para reproducir el sonido asociado.
2. Control de opciones: La variable opción se asigna con el valor 5, indicando al sistema que se debe ejecutar la acción de reproducción de la nota seleccionada.
3. Activación de TRAPs: Cada TRAP se activa de forma independiente, generalmente como respuesta a la interacción del usuario, ya sea a través de un teclado o un interfaz gráfico, permitiendo un control preciso sobre qué nota se reproducirá en cada momento.

Además de usar TRAPs para la gestión de notas musicales, se han implementado para controlar distintas activaciones y desactivaciones de ciertas funciones:

- Activación del Robot IRB 1400 YuMi

```
TRAP trap_7_DanceON

    IF DO_5_DanceON = 1 THEN
        TPWRite "Activo DanceRobot -> Pasamos a desactivarlo";
        reset DO_5_DanceON;
        ActiveRobot1 := FALSE;
    ELSE
        TPWRite "Incativo DanceRobot -> Pasamos a Activarlo";
        set DO_5_DanceON;
        ActiveRobot1 := TRUE;
    ENDIF

ENDTRAP
```

Ilustración 87: Interrupción TRAP para activación/desactivación del robot YuMi

- Modo de empleo de la pinza instrumento

```
TRAP trap_12_UnaDosNotas

    IF DO_1_OneNote1 = 1 THEN
        TPWRite "Una nota, cambiamos a dos";
        reset DO_1_OneNote1;
        reset DO_3_OneNote2;
    ELSE
        TPWRite "Dos notas, cambiamos a una";
        set DO_1_OneNote1;
        set DO_3_OneNote2;
    ENDIF

ENDTRAP
```

Ilustración 88: Interrupción TRAP para tocar una o dos notas en el teclado a través de los modos de la pinza herramienta

- Activación de los distintos Robots IRB 120

```
TRAP trap_13_ROB1

    IF DO_2_ActiveRobot1 = 1 THEN
        TPWRite "Activo Robot1 -> Pasamos a desactivarlo";
        reset DO_2_ActiveRobot1;
        ActiveRobot1 := FALSE;
    ELSE
        TPWRite "Incativo Robot1 -> Pasamos a Activarlo";
        set DO_2_ActiveRobot1;
        ActiveRobot1 := TRUE;
    ENDIF

ENDTRAP

TRAP trap_14_ROB2

    IF DO_4_ActiveRobot2 = 1 THEN
        TPWRite "Activo Robot2 -> Pasamos a desactivarlo";
        reset DO_4_ActiveRobot2;
        ActiveRobot2 := FALSE;
    ELSE
        TPWRite "Incativo Robot2 -> Pasamos a Activarlo";
        set DO_4_ActiveRobot2;
        ActiveRobot2 := TRUE;
    ENDIF

ENDTRAP
```

Ilustración 89: Interrupciones TRAP de la activación/desactivación de los robots IRB120

Para mejorar la interacción del usuario y la dinámica del sistema, se implementaron TRAPs que permiten seleccionar distintos modos de ejecución musical y controlar la finalización del programa. Cada TRAP modifica la variable opción o nosalir según la acción deseada.

Estas variables función serán las que empleamos como los distintos CASE en nuestro menú principal main.

- Activación del modo Escala Ascendente.

```
TRAP trap_8_EscalaAscendente
    opcion := 1;
ENDTRAP
```

Ilustración 90: Interrupción TRAP de activación del modo Escala Ascendente

- Activación del modo Escala Descendente.

```
TRAP trap_9_EscalaDescendente
    opcion := 2;
ENDTRAP
```

Ilustración 91: Interrupción TRAP de activación del modo Escala Descendente

- Activación del modo Lectura de Partitura.

```
TRAP trap_10_Partitura
    opcion := 3;
ENDTRAP
```

Ilustración 92: Interrupción TRAP de activación del modo Leer Partitura

- Activación de modo Aleatorio.

```
TRAP trap_11_Aleatorio
    opcion := 4;
ENDTRAP
```

Ilustración 93: Interrupción TRAP de activación del modo Aleatorio

- Activación de Finalización de Programa.

```
TRAP trap_15_NOSALIR

    nosalir:=FALSE;

ENDTRAP
```

Ilustración 94: Interrupción TRAP de finalización del programa

4.1.3.5 Creación de las distintas funciones

Primero vamos a explicar las funciones principales usadas en los módulos del Controlador_Robots_IRB120.

Num Rand

Descripción del funcionamiento

1. Semilla y generación de número aleatorio:
   - La variable seed se actualiza mediante la fórmula (171 * seed) MOD 30269 para garantizar un flujo pseudoaleatorio.
   - random almacena un valor decimal entre 0 y 1, calculado a partir de la semilla.

2. Escalado al rango deseado:
   - Se multiplica el valor decimal random por posibilidad (el límite superior del rango) y se suma 1, para obtener un número entre 1 y posibilidad.
   - La función Trunc elimina la parte decimal, asegurando que el resultado sea un número entero.

3. Retorno del valor:
   - La función devuelve el número entero generado, que puede ser utilizado para seleccionar notas aleatorias u otras acciones dentro del sistema.

```
! Generar un numero aleatorio
LOCAL FUNC num rand(num posibilidad)
    VAR num random;
    VAR num valor:=0;
    VAR num ej:=0;
    VAR num inicio:=0;
    VAR num fin:=0;

    seed:=(171*seed) MOD 30269;
    random:=seed/30269;
    valor:=random*posibilidad+1;
    valor:=Trunc(valor\Dec:=0);

    RETURN valor;

ENDFUNC
```

Ilustración 95: Función num rand para la generación de un numero aleatorio

Notas Aleatorias

Descripción del funcionamiento

1. Generación de nota aleatoria:
   - Se utiliza la función rand(6) para generar un valor aleatorio entre 0 y 6, correspondiente a las siete notas de la escala (DO, RE, MI, FA, SOL, LA, SI).
   - El valor generado se asigna a la variable Nota_a_tocar, que indica la nota que se reproducirá.

2. Reproducción de la nota:
   - La función TocarNota(Nota_a_tocar) recibe la nota seleccionada aleatoriamente y la reproduce en el sistema, permitiendo al usuario escuchar una nota diferente cada vez que se ejecuta el procedimiento.

```
! Tocar notas Aleatorias
PROC NotasAleatorias()

    TPWrite "Tocar nota aleatoria";
    Nota_a_tocar:=rand(6);
    TocarNota(Nota_a_tocar);

ENDPROC
```

Ilustración 96: Función de RAPID NotasAleatorias

Leer Partitura

Descripción del funcionamiento

1. Apertura y lectura del fichero:
   - Se utiliza Open para abrir el fichero de texto ubicado en "HOME:" y ReadStr para leer cada línea del fichero.
   - La variable saltodelinea controla si se debe avanzar a la siguiente línea de la partitura.
2. Interpretación de las notas:
   - Cada línea del fichero se analiza carácter por carácter mediante la estructura TEST ... CASE.
   - Según el carácter leído (C, D, E, F, G, A, B), se asigna la nota correspondiente a la variable Nota_a_tocar.
3. Reproducción de las notas:
   - La función TocarNota(Nota_a_tocar) se invoca tras determinar la nota, permitiendo su reproducción inmediata.
4. Control de finalización:
   - Si opción = 0 o la línea está vacía, el procedimiento finaliza mediante GOTO last.
   - La estructura WHILE línea <> EOF garantiza que se procesen todas las líneas del fichero hasta llegar al final.

Ruta del fichero:

`C:\Users\User\Documents\RobotStudio\VirtualControllers\Controlador_Robots_IRB120\HOME`

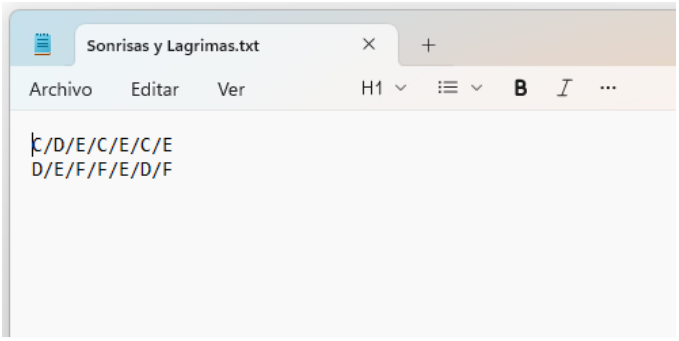


Ilustración 97: Fichero partitura Sonrisas y Lágrimas.txt

Para trabajar con partituras almacenadas en ficheros de texto, se utiliza Open para abrir el fichero en modo lectura y Close para cerrarlo al finalizar, garantizando que los recursos del sistema se gestionen correctamente. La función ReadStr se emplea para leer el contenido línea por línea o hasta un delimitador específico, lo que permite procesar cada nota de la partitura de manera secuencial y asignarla directamente a la variable correspondiente. Estas funciones se usan porque facilitan la lectura controlada del fichero, permiten recorrer todas las notas de manera ordenada y aseguran que el fichero se cierre correctamente al terminar, evitando errores o bloqueos de recursos.

```
! Leer partitura de un fichero .txt
PROC LeerPartitura(string nombre)
    VAR string caracter;
    VAR string linea;
    VAR num caso;
    VAR num fila :=0;
    VAR iodev fic;

    Close fic;
    Open "HOME:"\File:=nombre,fic\Read;


  WHILE linea <> EOF DO

      TPWrite "Empezamos por la primera linea";

      IF saltodelinea THEN
          linea:=ReadStr(fic);
          saltodelinea:=FALSE;
          fila:=0;
          GOTO init;
      ENDIF

      init: linea:=ReadStr(fic\Delim:="\2F");

      TPWrite linea;
```

```
IF opcion = 0 OR StrLen(linea)=0 THEN
    GOTO last;
ENDIF

TEST linea
CASE "C":
  TPWrite "Tocamos el DO";
  Nota_a_tocar := Nota_DO;

CASE "D":
  TPWrite "Tocamos el RE";
  Nota_a_tocar := Nota_RE;

CASE "E":
  TPWrite "Tocamos el MI";
  Nota_a_tocar := Nota_MI;

CASE "F":
  TPWrite "Tocamos el FA";
  Nota_a_tocar := Nota_FA;

CASE "G":
  TPWrite "Tocamos el SOL";
  Nota_a_tocar := Nota_SOL;

CASE "A":
  TPWrite "Tocamos el LA";
  Nota_a_tocar := Nota_LA;

CASE "B":
  TPWrite "Tocamos el SI";
  Nota_a_tocar := Nota_SI;

DEFAULT:

ENDTEST


          TocarNota(Nota_a_tocar);

    ENDWHILE
```

Ilustración 98: Función de RAPID Leer Partitura

Se usan caracteres en lugar de strings completos porque cada nota de la escala (C, D, E, F, G, A, B) se puede representar con un solo símbolo, lo que hace que el código sea más simple y rápido de procesar. Con caracteres, podemos comparar directamente cada uno usando un CASE o un IF sin tener que hacer operaciones de parsing de strings más largas, lo que también ahorra memoria y tiempo de ejecución, algo importante en sistemas con recursos limitados. Además, al leer la partitura línea por línea, procesar los caracteres uno a uno permite tocar las notas en secuencia de manera directa y manejar mejor los espacios o saltos de línea. En resumen, usar caracteres hace que todo sea más eficiente, sencillo y fácil de entender.

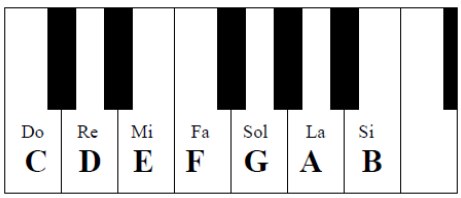


Ilustración 99: Relación entre la nomenclatura de las notas de un piano

Tocar Nota

Descripción del funcionamiento

1. Movimiento del robot:
   - MoveJ mueve el robot a la posición de la tecla correspondiente según la nota pulsada (notaPulsada) multiplicando la distancia base DistNota.
   - Una vez pulsada la tecla, MoveL devuelve la pinza a la posición inicial (Target_30).
2. Asignación de la nota:
   - Según el valor de notaPulsada, se asigna el nombre de la nota ("Do", "Re", etc.) a la variable nota y se activa la salida digital correspondiente (DO_6_DO, DO_7_RE, etc.) para indicar que la nota se ha pulsado.
3. Comunicación con el servidor:
   - SocketSend envía la nota reproducida al cliente o servidor conectado, permitiendo registrar o mostrar la nota en tiempo real.

```
PROC TocarNota(num notaPulsada)

    MoveJ Offs (Target_10, DistNota*notaPulsada, 0,0),v300,z0,Pinza_Pulsador\WObj:=Teclado_1;

    IF notaPulsada = 0 THEN
        nota := "Do";
        Set DO_6_DO;
    ELSEIF notaPulsada = 1 THEN
        nota := "Re";
        Set DO_7_RE;
    ELSEIF notaPulsada = 2 THEN
        nota := "Mi";
        Set DO_8_MI;
    ELSEIF notaPulsada = 3 THEN
        nota := "Fa";
        Set DO_9_FA;
    ELSEIF notaPulsada = 4 THEN
        nota := "Sol";
        Set DO_10_SOL;
    ELSEIF notaPulsada = 5 THEN
        nota := "La";
        Set DO_11_LA;
    ELSEIF notaPulsada = 6 THEN
        nota := "Si";
        Set DO_12_SI;
    ENDIF

    !iniciamos el server

    SocketSend client_L \Str:=nota;

    MoveL Target_30,v500,z0,Pinza_Pulsador\WObj:=Teclado_1;

ENDPROC
```

Ilustración 100: Función de RAPID Tocar Nota

Tocas notas Ascendentemente

Descripción del funcionamien

1. Escala ascendente completa (Tocar_Notas_Asc):
   - Se recorre un bucle desde 0 hasta 6, correspondiente a las siete notas de la escala (Do a Si).
   - MoveJ desplaza la pinza a la tecla correspondiente según el índice cont y la distancia entre notas DistNota.
   - MoveL devuelve la pinza a la posición inicial después de pulsar cada tecla.
   - Este procedimiento permite tocar la escala de manera secuencial y continua.

2. Escala ascendente por intervalos (Tocar_Notas_Asc_Intervalos):
   - Similar al anterior, pero se saltan teclas de manera que el robot toca cada segunda nota de la escala, generando intervalos.
   - El bucle recorre un rango menor (0 a 3) y multiplica la distancia entre notas por 2 (DistNota*2).

```
LOCAL PROC Tocar_Notas_Asc()

    VAR intnum cont;

    TPWrite "Empezamos a tocar la escala ascendentemente";
    FOR cont FROM 0 TO 6 DO
        MoveJ Offs (Target_10, DistNota*cont, 0,0),v300,z0,Pinza_Pulsador\WObj:=Teclado_1;
        MoveL Target_30,v500,z0,Pinza_Pulsador\WObj:=Teclado_1;
        !TocarNota(cont);
    ENDFOR

ENDPROC
```

Ilustración 101: Función de RAPID Tocar Notas Ascendentes

```
PROC Tocar_Notas_Asc_Intervalos()

    VAR intnum cont;

    FOR cont FROM 0 TO 3 DO
        MoveJ Offs (Target_10_2, (DistNota*2)*cont, 0,0),v300,z0,Pinza_Pulsador\WObj:=Teclado_1;
        MoveL Target_30,v300,z0,Pinza_Pulsador\WObj:=Teclado_1;
    ENDFOR

ENDPROC
```

Ilustración 102: Función de RAPID Tocar notas ascendentes en intervalos

Tocar Notas Descendentemente

Descripción del funcionamiento

1. Escala descendente completa (Tocar_Notas_Desc):
   - Se recorre un bucle desde 6 hasta 0, correspondiente a las notas de la escala de Si a Do.
   - Se invoca el procedimiento TocarNota(cont) para pulsar cada nota en orden descendente.

2. Escala descendente por intervalos (Tocar_Notas_Desc_Intervalos):
   - El procedimiento toca cada segunda nota descendente, generando intervalos dentro de la escala.
   - El bucle recorre un rango menor (3 a 0) y multiplica la distancia entre notas por 2 (DistNota*2) para saltar teclas.

```
LOCAL PROC Tocar_Notas_Desc()

    VAR intnum cont;

    TPWrite "Empezamos a tocar la escala descendentemente";

    FOR cont FROM 6 TO 0 DO
        TocarNota(cont);
    ENDFOR

ENDPROC
```

Ilustración 103: Función de RAPID Tocar Notas Descendentes

```
LOCAL PROC Tocar_Notas_Desc_Intervalos()

    VAR intnum cont;

    FOR cont FROM 3 TO 0 DO
        MoveJ Offs (Target_10_2, (DistNota*2)*cont, 0,0),v300,z0,Pinza_Pulsador\WObj:=Teclado_1;
        MoveL Target_30,v300,z0,Pinza_Pulsador\WObj:=Teclado_1;
    ENDFOR

ENDPROC
```

Ilustración 104: Función Tocar Notas descendentes en Intervalos

A continuación, vamos a explicar las distintas funciones principales usadas en los módulos del Controlador3.

Bailar Nota

Descripción del funcionamiento

1. Configuración inicial:
   - ConfL\Off y ConfJ\Off desactivan configuraciones de seguridad o limitaciones de eje, permitiendo movimientos libres del robot.

2. Movimientos según la nota:
   - Se evalúa la variable nota y se ejecuta un movimiento absoluto (MoveAbsJ) hacia una posición específica (target_do o target_2).
   - Esto provoca que la pinza del robot se desplace de forma coordinada con la nota que se está tocando, creando un efecto visual de baile.

3. Velocidad y precisión:
   - La velocidad se ajusta con v100 y la precisión con fine, asegurando movimientos suaves y sincronizados.
   - La pinza utilizada es Pinza_Yumi_L, indicando que el brazo izquierdo del robot realiza los movimientos.

```
PROC BailarNota()

    ConfL\Off;
    ConfJ\Off;

   IF nota = "Do" THEN
        MoveAbsJ Abajo_, v100, fine, Pinza_Yumi_L;
   ELSEIF nota = "Re" THEN
        MoveAbsJ Adentro_, v100, fine, Pinza_Yumi_L;
    ELSEIF nota = "Mi" THEN
        MoveAbsJ Abajo_, v100, fine, Pinza_Yumi_L;
    ELSEIF nota = "Fa" THEN
        MoveAbsJ Adentro_, v100, fine, Pinza_Yumi_L;
    ELSEIF nota = "Sol" THEN
        MoveAbsJ Abajo_, v100, fine, Pinza_Yumi_L;
    ELSEIF nota = "La" THEN
        MoveAbsJ Adentro_, v100, fine, Pinza_Yumi_L;
    ELSEIF nota = "Si" THEN
        MoveAbsJ Abajo_, v100, fine, Pinza_Yumi_L;
   ENDIF

ENDPROC
```

Ilustración 105: Función de RAPID Bailar Nota

Bailar Normal

Descripción del funcionamiento

1. Configuración inicial:
   - ConfL\Off y ConfJ\Off desactivan restricciones de seguridad o limitaciones de eje, permitiendo movimientos libres y fluidos del robot.

2. Patrón de movimiento:
   - El robot se desplaza alternativamente entre las posiciones target_do y target_2.
   - Cada movimiento se realiza con velocidad v100 y precisión fine, asegurando desplazamientos suaves.

3. Efecto visual:
   - Este patrón repetitivo genera un "baile" sencillo del brazo izquierdo (Pinza_Yumi_L) del robot, agregando dinamismo a la presentación musical.

```
LOCAL PROC BailarNormal()

    ConfL\Off ;
    ConfJ\Off;

    MoveAbsJ Abajo_, v100, fine, Pinza_Yumi_L;
    MoveAbsJ Adentro_, v100, fine, Pinza_Yumi_L;

    MoveAbsJ Abajo_, v100, fine, Pinza_Yumi_L;
    MoveAbsJ Adentro_, v100, fine, Pinza_Yumi_L;


ENDPROC
```

Ilustración 106: Función de RAPID Bailar Normal

Leer Nota de IRB120

Descripción del funcionamiento

1. Recepción de datos:
   - SocketReceive se utiliza para recibir información desde el socket sock, asignando el valor recibido a la variable notaSocket.
   - Esta variable contiene el nombre de la nota que el robot IRB1200 ha pulsado o enviado.

2. Retorno de la nota:

   - La función devuelve notaSocket, permitiendo que otros procedimientos del sistema conozcan qué nota fue recibida y puedan procesarla, tocarla con el robot Yumi o activar movimientos de baile.

```
FUNC string LeerNotaDeIRB1200()
    SocketReceive sock \Str:=notaSocket;
    RETURN notaSocket;
ENDFUNC
```

Ilustración 107: Función de RAPID Leer nota de IRB120

4.1.3.6 Diagrama de flujo de las funciones main

La función mainPiano se encarga de controlar el piano robótico y de recibir comandos para ejecutar distintas acciones musicales. Primero, inicializa variables y hardware mediante la llamada a Init, y establece un bucle controlado por la variable nosalir que determina cuándo salir de la rutina. A continuación, configura un socket servidor en la IP local y puerto 5000, lo que permite que otros programas o robots se conecten y envíen órdenes. Dentro del bucle principal, la función verifica si ActiveRobot1 está activo y, según el valor de la variable opcion, ejecuta diferentes acciones: tocar escalas ascendentes o descendentes, leer partituras desde un archivo, tocar notas aleatorias o tocar una nota específica. Tras cada acción, opcion se reinicia, y el bucle espera medio segundo antes de continuar, asegurando un ciclo continuo de recepción de órdenes y ejecución musical.

MODULEPIANO

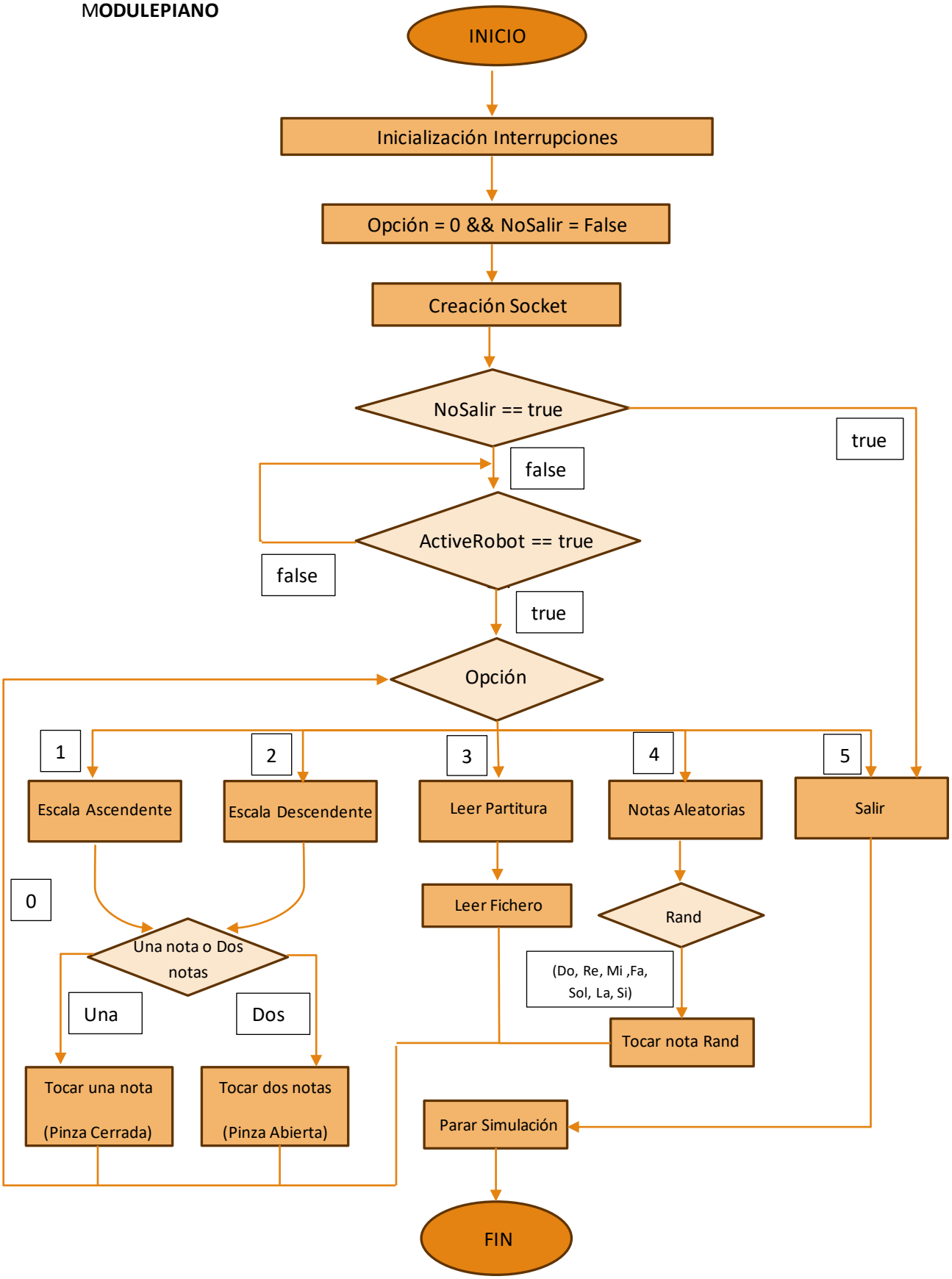


Ilustración 108: Diagrama de flujo de la función main de ModulePiano de RAPID

Por su parte, la función mainL controla el robot bailarín y actúa como cliente que se conecta al servidor creado por mainPiano. Al inicio, también inicializa variables y hardware, y establece un socket cliente que se conecta a la IP y puerto donde corre mainPiano. Su bucle principal se activa cuando un sensor (DI_0_ON) o una variable de control (dance_ON) indica que el robot debe bailar. En cada iteración, muestra en pantalla un mensaje con la nota correspondiente y, según la variable opcion, decide si debe leer la nota proveniente del piano y ejecutar un baile específico, o si debe realizar un patrón de baile normal. Cada iteración del bucle espera 0.1 segundos, y al finalizar, el socket se cierra para liberar la conexión.

MODULEDANCE

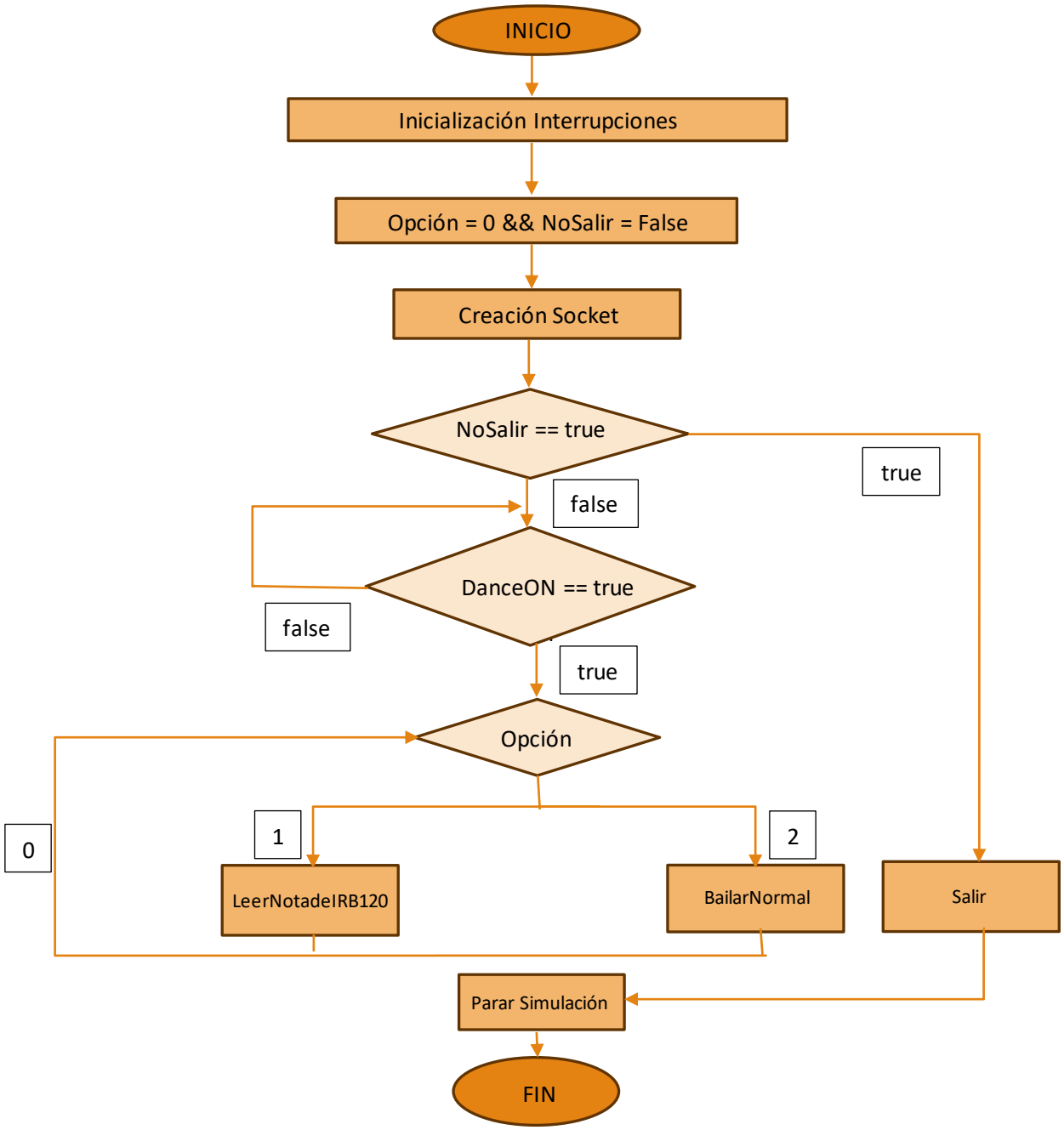


Ilustración 109: Diagrama de flujo de la función main de DanceL de RAPID

La relación entre ambas funciones se da a través del socket TCP/IP: mainPiano funciona como servidor que toca notas y envía información, mientras que mainL actúa como cliente que recibe las notas y realiza los movimientos de baile correspondientes. La variable opcion es el mecanismo que coordina qué acción se ejecuta en cada robot, asegurando que el piano y el bailarín trabajen de forma sincronizada. En conjunto, este sistema permite que las acciones del piano robótico influyan directamente en los movimientos del robot bailarín, creando una interacción musical y coreográfica entre ambos.

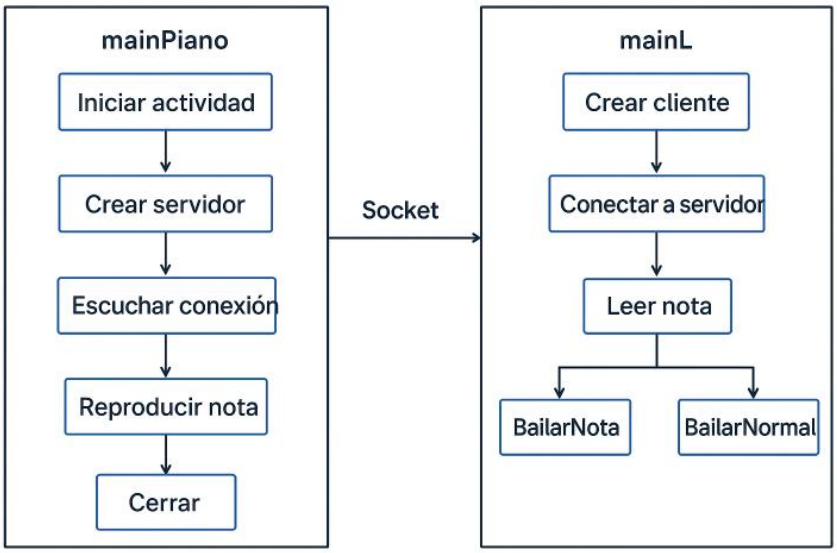


Ilustración 110: Diagrama de flujo de la relación entre los módulos

4.2 ABB IRC5 OPC CONFIGURATION

La arquitectura OPC UA (OPC Unified Architecture) es un estándar de comunicación para entornos industriales que proporciona un modelo seguro, independiente del fabricante y de la plataforma para el intercambio de información. A diferencia de los antiguos OPC basados en COM/DCOM, OPC UA define según manual (ABB, 2020-2022):

- Un modelo de información orientado a objetos: los datos se exponen como un espacio de direccionamiento jerárquico de *nodos* (Nodes). Cada nodo puede representar una variable, un objeto, un método, un tipo de dato o una referencia.

- Namespaces y NodeIds: cada servidor organiza sus nodos en *namespaces* y cada nodo tiene un identificador único (NodeId) que el cliente puede explorar.

- Mecanismos de acceso flexibles: lectura/escritura puntual (Read/Write), *subscriptions* (publicación de cambios) y llamada a métodos (Method Call).

- Seguridad integrada: canales cifrados, firmas, certificados para autenticar servidor/cliente y políticas de seguridad configurables.

- Portabilidad: funciona sobre TCP/UA Binary, WebSockets, HTTPS, etc., y no depende del sistema operativo ni del lenguaje.

Gracias a este enfoque, OPC UA se usa tanto como pasarela de datos entre sistemas heterogéneos (PLC, robots, SCADA, MES) como modelo de información que permite describir estructura, tipos y comportamiento de los datos de forma explícita y autodescriptiva.

ABB proporciona un servidor OPC (IRC5 OPC) que puede ejecutarse en el controlador IRC5 real o en el Controlador Virtual dentro de RobotStudio. Este servidor expone nodos que representan:

- Señales digitales y analógicas (I/O).
- Variables RAPID (numeric, bool, string, robtargets, wobjdata, ...).
- Estados del controlador y del robot (estados de seguridad, alarmas, posición, programa en ejecución).
- Eventos y alarmas.

En RobotStudio se dispone de una herramienta de OPC Configuration que permite seleccionar qué variables y señales del controlador se exportan (mapearlas) al espacio OPC UA.

Abriremos el programa ABB IRC5 OPC Configuration.

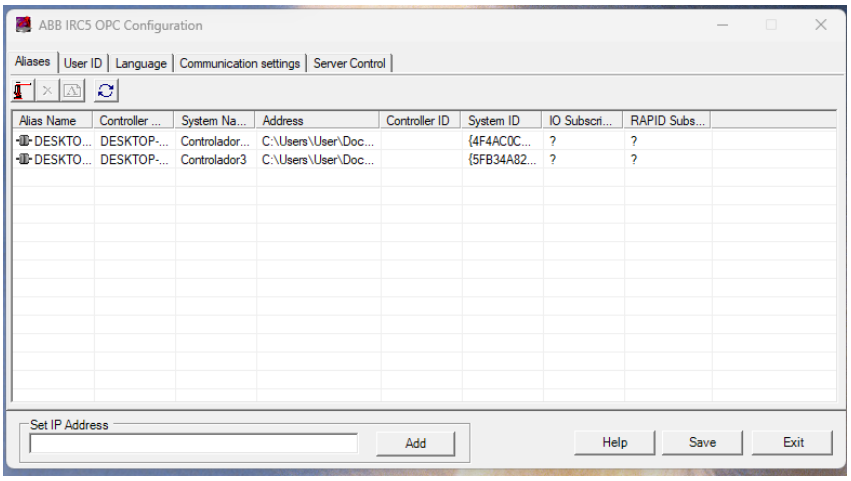


Ilustración 111: Interfaz inicial del programa ABB IRC5 OPC Configuration

Escaneamos para ver los controladores disponibles en nuestro PC para generar un nuevo alias.

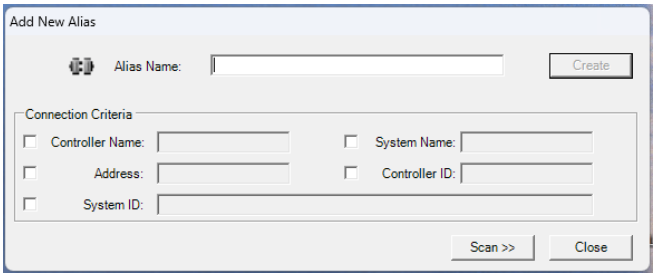


Ilustración 112: Creación de un nuevo alias para un controlador

Observamos a continuación los dispositivos encontrados por el programa.

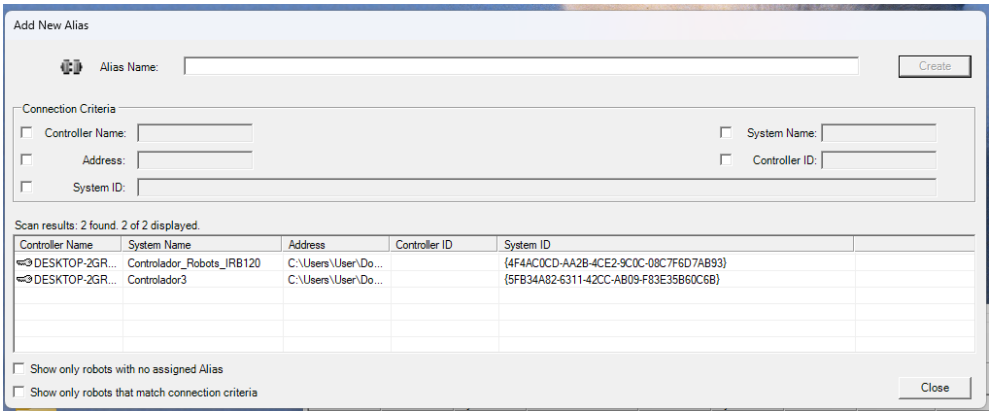


Ilustración 113: Dispositivos escaneados por ABB IRC5 OPC Configuration

Seleccionamos uno de ellos, y sus criterios de conexión, y lo creamos.

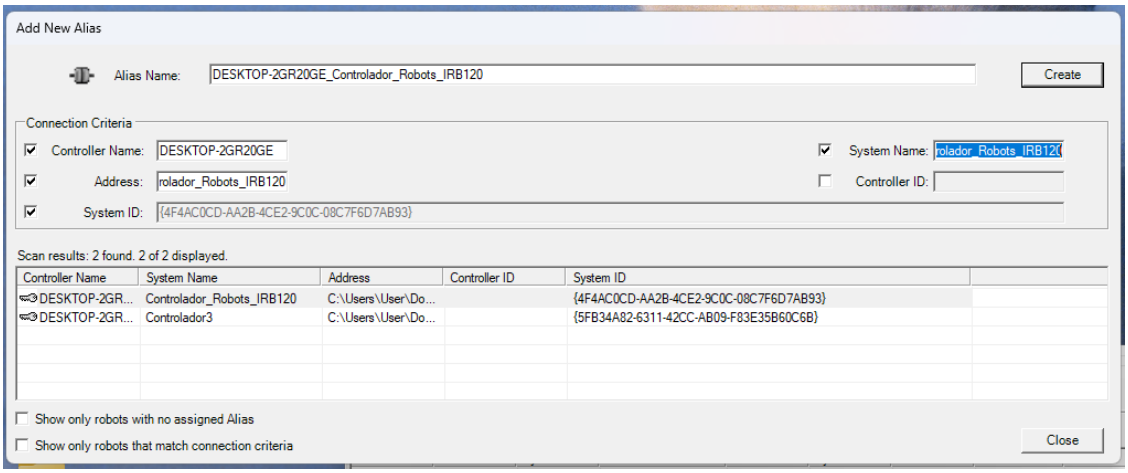


Ilustración 114: Creación de un controlador y sus criterios de conexión en ABB IRC5 OPC Configuration

Iniciamos el OPC Server

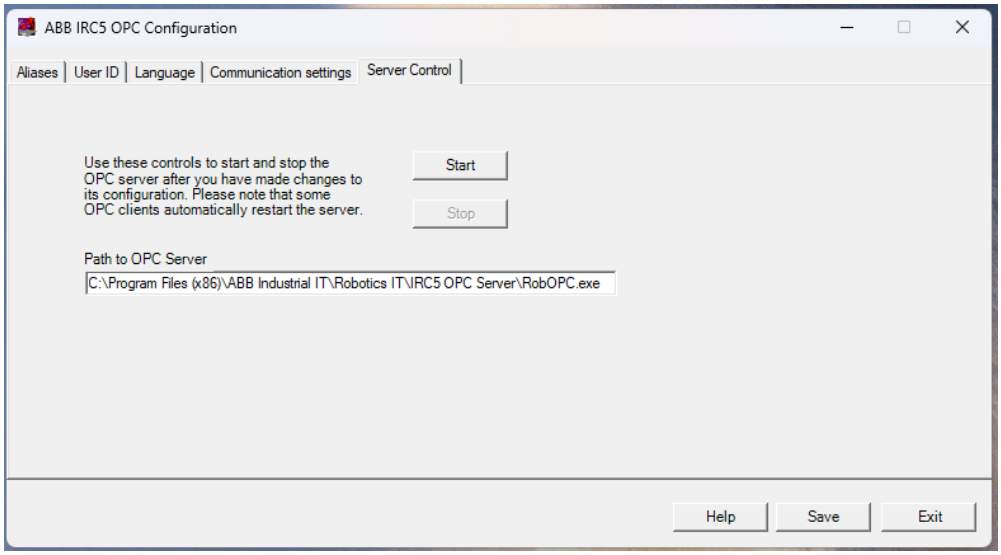


Ilustración 115: Activación y desactivación del Server Control de los dispositivos de ABB IRC5 OPC Configuration

4.3 MATLAB

OPC DA (Data Access) es uno de los estándares de la familia OPC Clásico, pensado principalmente para el acceso a variables en tiempo real de controladores, PLCs, SCADA o sistemas de simulación como RobotStudio.

En el caso de MATLAB, la comunicación con un servidor OPC DA se realiza mediante el OPC Toolbox, que actúa como cliente OPC. Esto permite:

1. Conectarse a un servidor OPC DA (en este caso, el ABB OPC Server de RobotStudio).
2. Leer variables en tiempo real (tags, señales de sensores, estados del robot).
3. Escribir variables (enviar consignas, activar flags, modificar parámetros de control).
4. Monitorizar la calidad de la comunicación (estado bueno, malo o incierto).

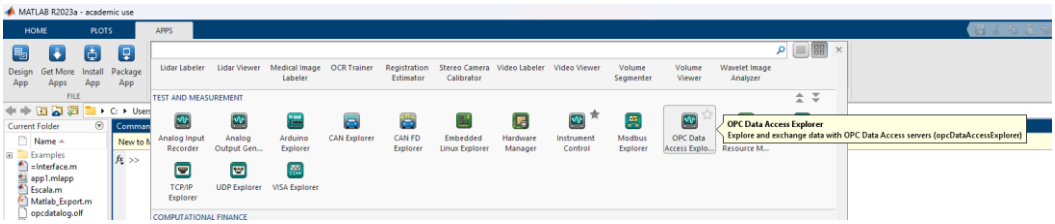


Ilustración 116: APPs de Matlab

Ventajas de usar OPC DA en MATLAB

- Integración sencilla con RobotStudio (ya que ABB incluye un servidor OPC DA).
- No requiere configuraciones complejas de seguridad (a diferencia de OPC UA).
- Permite trabajar en tiempo real con los robots (ej. enviar consignas desde MATLAB y recibir estados instantáneamente).
- MATLAB puede procesar datos en paralelo (filtrado, control avanzado, análisis matemático).

Abrimos la aplicación y generamos un nuevo Host.

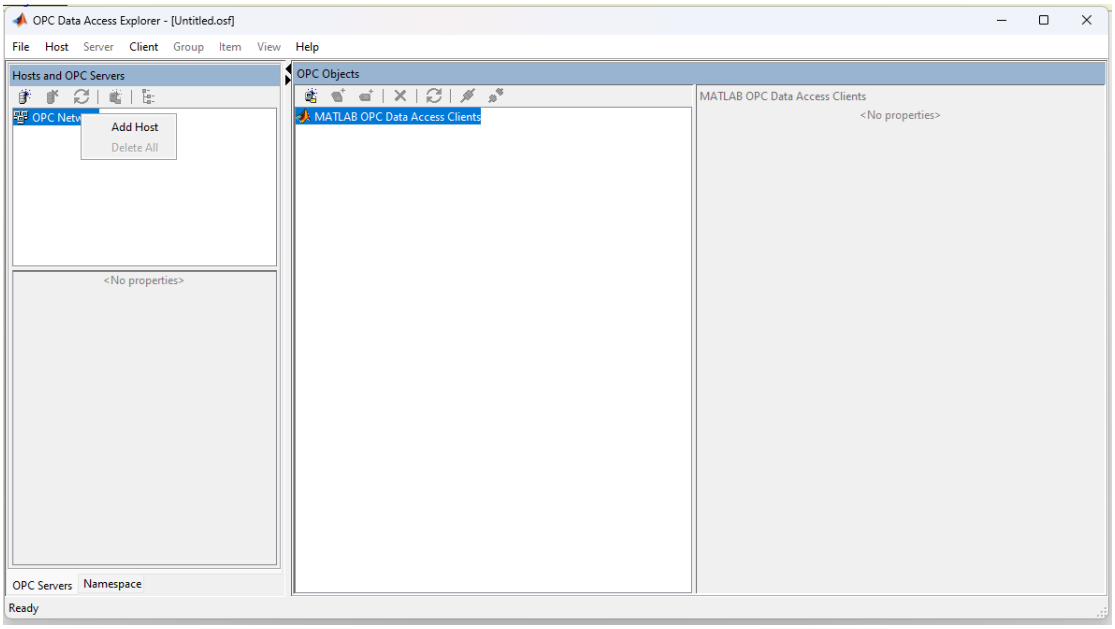


Ilustración 117: Interfaz OPC Data Explorer de Matlab

Le damos un nombre como 'localhost'.

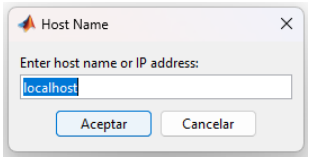


Ilustración 118: Definición del hostname de un host en OPC Data Explorer

Creamos un Cliente en el local host previamente definido.

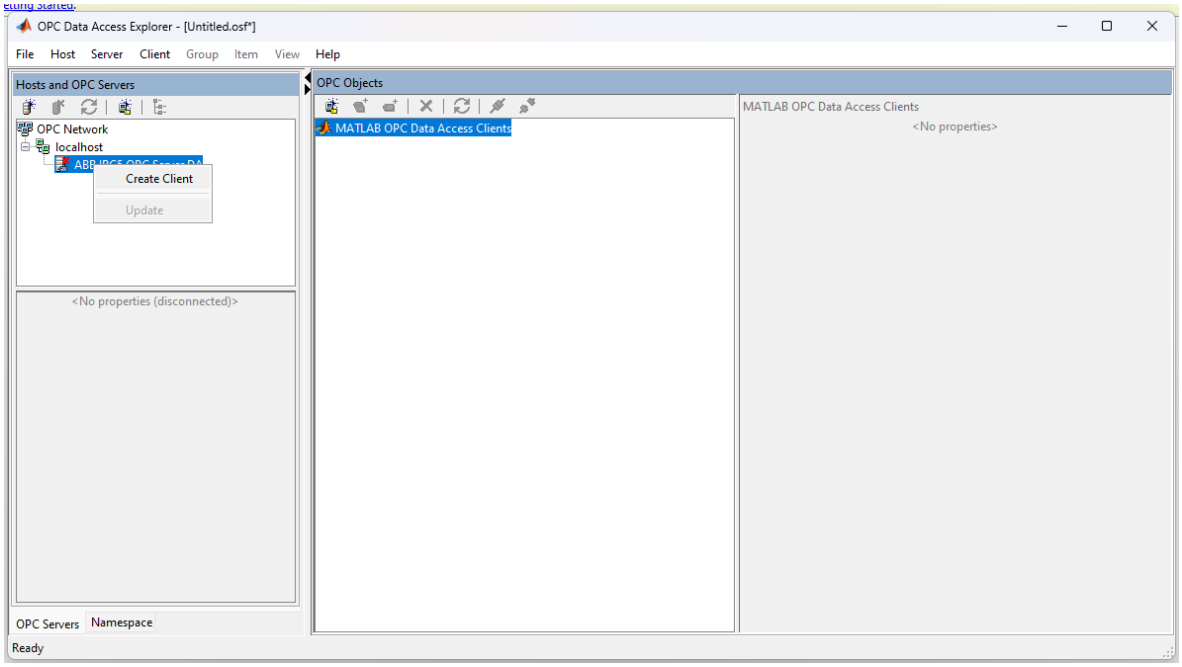


Ilustración 119: Creación de un cliente en OPC Data explorer de Matlab

Ya conectado, lo conectamos con el servidor.

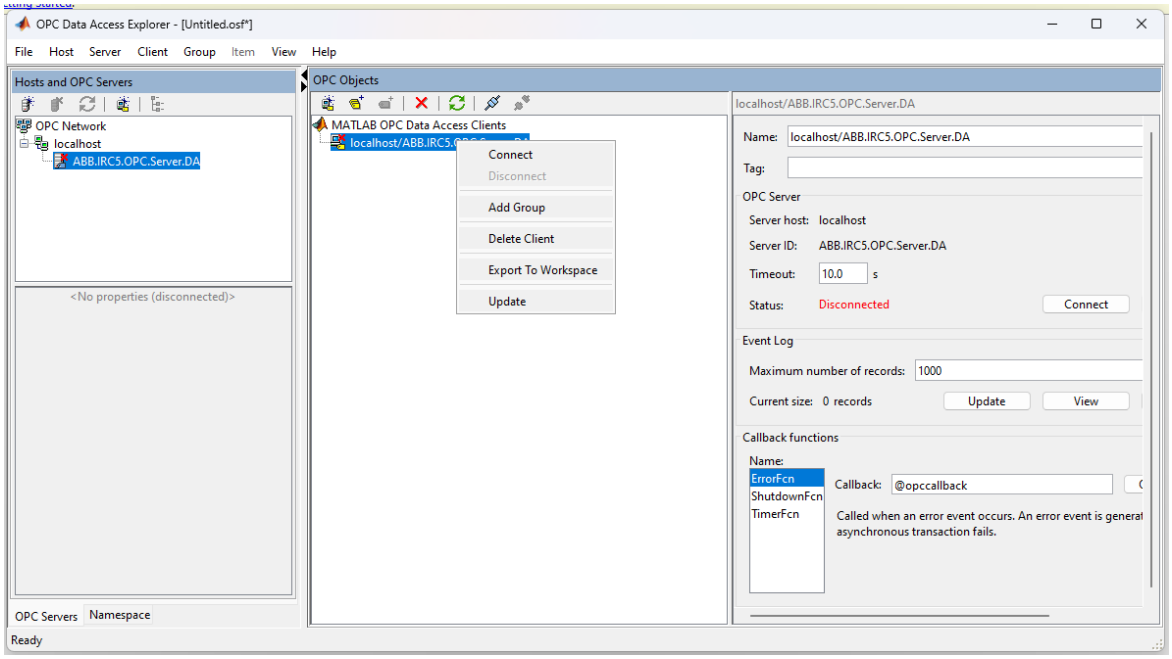


Ilustración 120: Conexión al servidor de OPC

Creamos el Grupo, que posteriormente emplearemos para la conexión OPC.

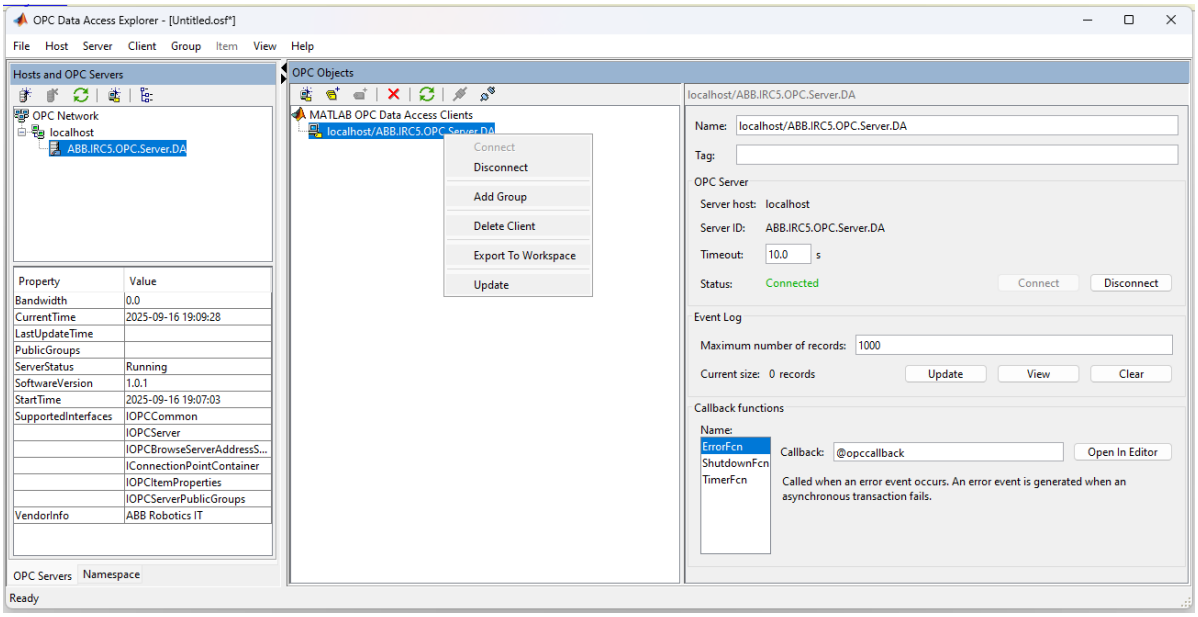


Ilustración 121: Creación de un grupo de variables en OPC Data Explorer de Matlab

Con el grupo ya creado, añadiremos las variables (Items).

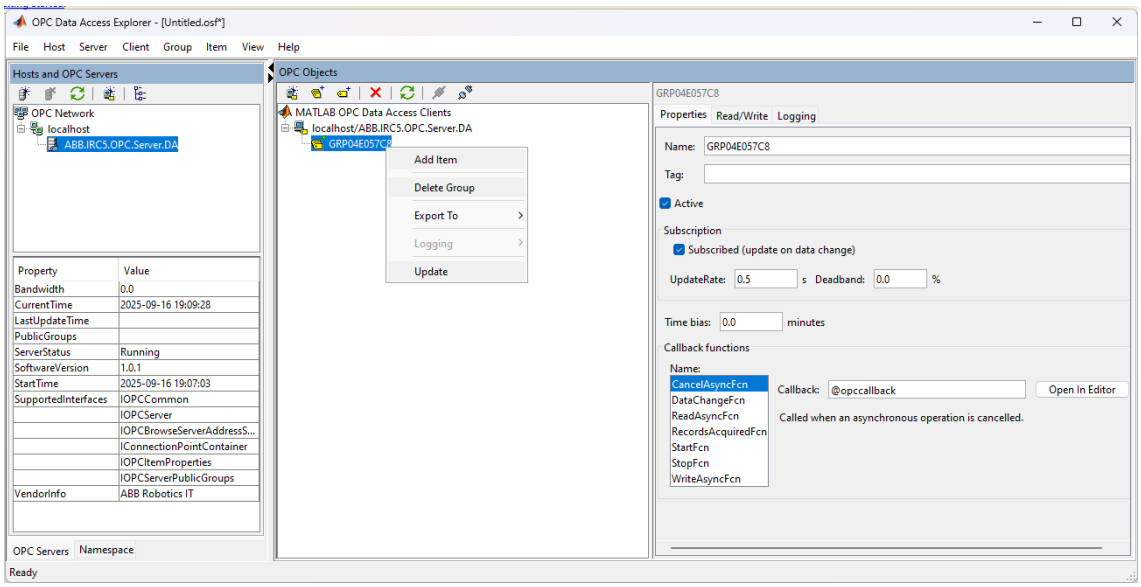


Ilustración 122: Añadir Items en un grupo del localhost en OPC Data Explorer de Matlab

Seleccionamos las variables que vamos a modificar a través de la conexión creada con el servidor de ABB IRC5 OPC.

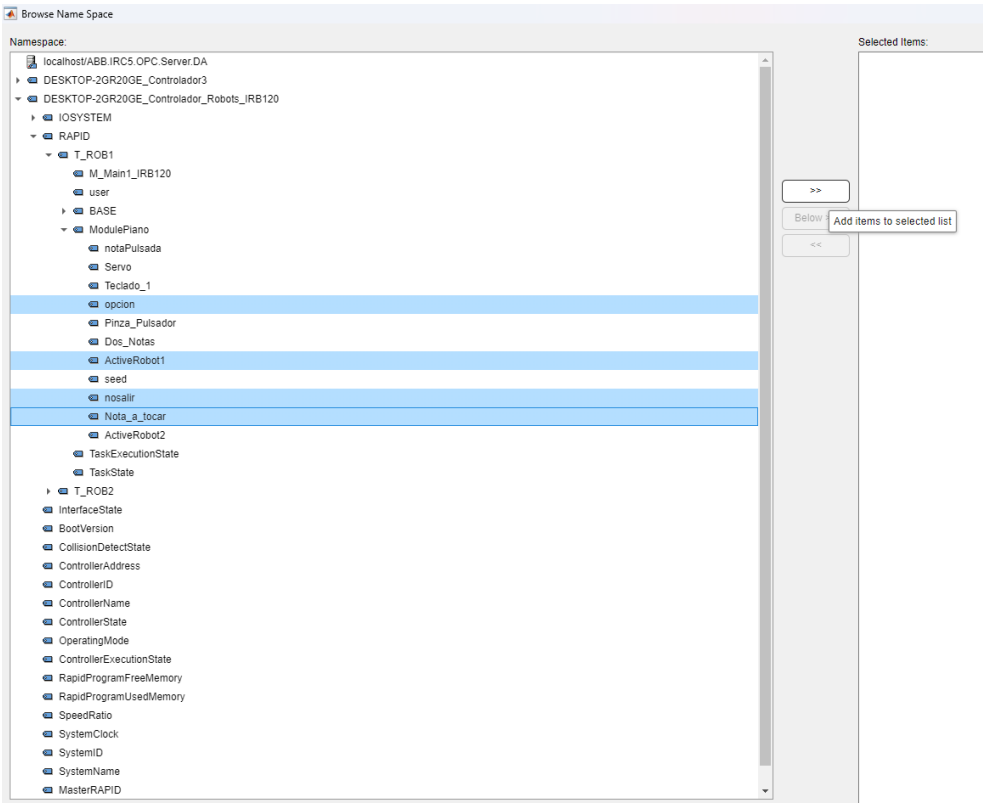


Ilustración 123: Items disponibles de nuestros controladores

Ya seleccionados los Items elegidos, podremos comenzar con la simulación.

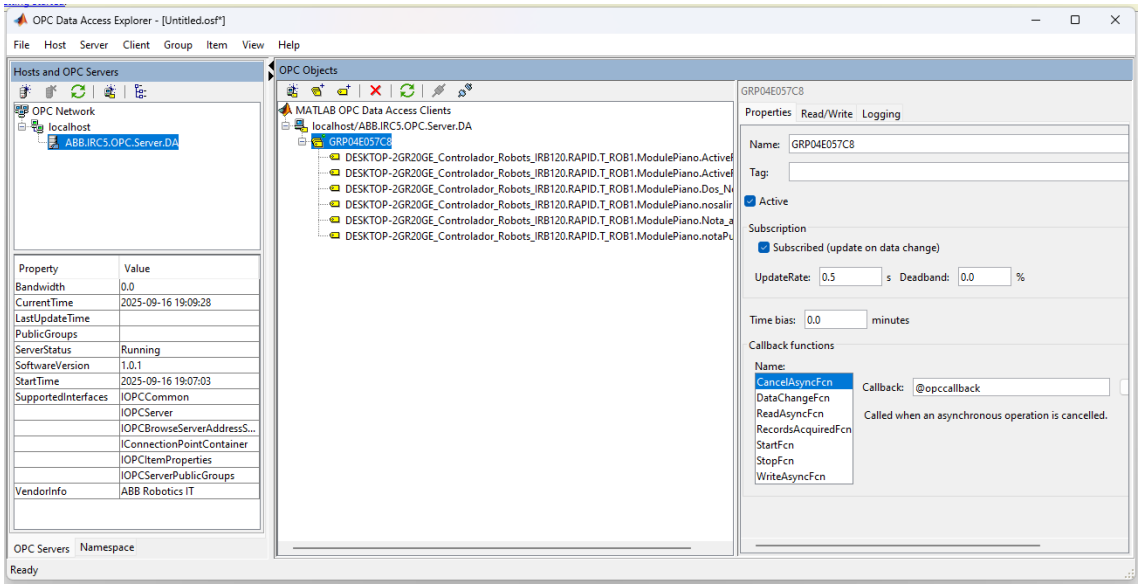


Ilustración 124: Explorador de variables del Server Control a través de OPC Data Explorer de Matlab

La definición y nombre de las variables e Items añadidos los emplearemos en el diseño de nuestra pantalla de iteracción con App Designer.

App Designer es la herramienta integrada de MATLAB para crear aplicaciones gráficas interactivas (GUIs), combinando diseño visual con programación en MATLAB para definir el comportamiento de la aplicación. Permite diseñar interfaces mediante un editor visual (arrastrar y soltar componentes) y luego añadir lógica mediante callbacks, clases, propiedades y métodos (MathWorks, 2025a).

Características principales

1. Diseño visual de la interfaz: Se pueden arrastrar y soltar componentes como botones, menús, controles deslizantes, tablas, gráficos, paneles y pestañas (MathWorks, 2025a).
2. Vistas de desarrollo:
    - *Design View* para organizar visualmente la interfaz.
    - *Code View* para programar el comportamiento de la aplicación (Matworks, 2025).
3. Programación orientada a objetos: Las aplicaciones creadas en App Designer son clases que heredan de matlab.apps.AppBase. Se utilizan propiedades para los componentes de la interfaz y métodos o callbacks para la lógica de interacción (MathWorks, 2025a).
4. Biblioteca de componentes modernos: Incluye componentes como botones, listas desplegables, paneles, pestañas, indicadores, lámparas, interruptores y ejes para gráficos 2D/3D (MathWorks, 2025c).
5. Componentes personalizados: El usuario puede crear y configurar sus propios componentes para integrarlos en la biblioteca de App Designer (MathWorks, 2025d).
6. Propiedades y organización de datos: Se pueden definir propiedades privadas para datos internos, públicas para accesibilidad y constantes para valores inmutables. Esto facilita la organización de aplicaciones complejas (MathWorks, 2025e).
7. Compartir y desplegar aplicaciones: Las aplicaciones se pueden empaquetar en archivos .mlappinstall, exportar como aplicaciones web o compilarlas en ejecutables con MATLAB Compiler o Web App Server (MathWorks, 2025a).

En App Designer de MATLAB, la Code View es la vista en la que se escribe y organiza el código asociado a la aplicación.

La *Code View* es el entorno donde se define la lógica funcional de la aplicación. En esta vista, el usuario programa el comportamiento de los componentes a través de callbacks, funciones y propiedades, que determinan cómo responde la aplicación ante acciones del usuario o eventos internos del sistema (MathWorks, 2025a).

Los callbacks son bloques de código o métodos que se ejecutan automáticamente cuando ocurre un evento asociado a un componente, como presionar un botón o mover un control deslizante; son el vínculo entre la interfaz visual y la lógica de la aplicación. Las funciones son secciones de código reutilizables que realizan tareas específicas dentro de la aplicación, ayudando a estructurar y modular el programa. Por último, las propiedades son variables definidas dentro de la clase de la aplicación que almacenan datos o estados de los componentes, pudiendo ser públicas (accesibles desde otros scripts o apps) o privadas (solo disponibles dentro de la propia aplicación). Esta organización orientada a objetos facilita el desarrollo de aplicaciones complejas, robustas y mantenibles en MATLAB (MathWorks, 2025b).

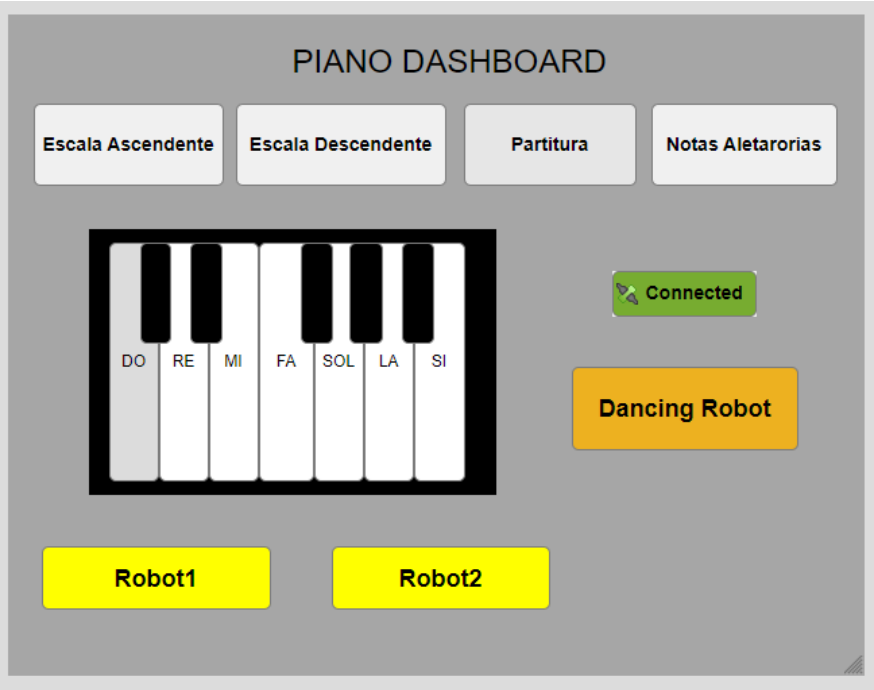


Ilustración 125: PIANO DASHBOARD

5. COMUNICACIONES

La comunicación dentro de un entorno ABB puede darse en tres niveles principales:

1. Entre módulos de RAPID dentro de un mismo controlador.
2. Entre varios controladores o con dispositivos externos mediante sockets.
3. Con plataformas externas de supervisión e integración industrial a través de OPC UA.

5.1 Módulos de RAPID

Dentro de un mismo controlador IRC5, el software RAPID organiza los programas en módulos. Cada módulo es una unidad que contiene:

- Procedimientos (PROCs): instrucciones ejecutables.
- Funciones (FUNCs): bloques que devuelven un valor.
- Datos locales (VAR, PERS, CONST).

Cuando varios módulos necesitan compartir información, se usan variables globales.

Características de las variables globales:

- Si una variable se declara con el modificador GLOBAL, puede ser accedida desde cualquier otro módulo del sistema RAPID.
- Pueden ser de tipo VAR (volátil, se reinicia con cada ciclo) o PERS (persistente, mantiene su valor incluso después de reinicios del programa).
- Esto permite que distintos módulos trabajen de manera coordinada sin necesidad de duplicar información.

Aplicaciones típicas:

- Compartir estados de máquina (ej. "Robot en espera", "Ciclo iniciado").
- Implementar contadores globales (ej. piezas producidas, ciclos completados).
- Guardar parámetros de proceso (velocidades, límites de tolerancia, banderas de seguridad).

En resumen, las variables globales facilitan la comunicación interna dentro de un mismo sistema RAPID, mejorando el modularidad y el mantenimiento del programa.

5.2 Controladores

Cuando se necesita que dos o más controladores IRC5 trabajen juntos, o bien que un robot intercambie datos con un PC u otro dispositivo, se utiliza la tecnología Socket Messaging de RAPID.

Características del Socket Messaging:

- Basado en el estándar TCP/IP (red Ethernet).
- Uno de los equipos asume el rol de servidor y el otro de cliente.
    - El servidor abre un puerto y espera conexiones.
    - El cliente inicia la comunicación y envía/recibe datos.
- Los datos pueden ser:
    - Cadenas de texto (mensajes, instrucciones simples).
    - Números (coordenadas, estados).
    - Mensajes estructurados (tablas de datos con un formato acordado).

Ejemplo de uso:

- Coordinación de robots en una celda:
    - Un robot actúa como servidor, informando su estado de producción.
    - Otro robot, como cliente, consulta ese estado y decide cuándo iniciar su tarea.
- Comunicación con un PC:
    - El PC envía comandos de proceso al robot.
    - El robot responde con estados, alarmas o resultados de producción.

Esto permite crear celdas robotizadas distribuidas sin necesidad de un PLC central, reduciendo costos

5.3 RobotStudio y sistemas externos

En entornos de automatización industrial más amplios, se requiere que los robots se integren con sistemas de supervisión y control como SCADA, MES o aplicaciones de Industria 4.0. Para esto, se utiliza el estándar OPC UA (Open Platform Communications Unified Architecture).

Características de OPC UA en ABB:

- Es un protocolo cliente-servidor usado ampliamente en la industria.
- ABB RobotStudio (o directamente un IRC5 real) puede actuar como servidor OPC UA, exponiendo variables y estados del robot.
- Los sistemas externos actúan como clientes OPC UA, leyendo y escribiendo datos en tiempo real.
- Permite intercambiar información como:
    - Entradas y salidas digitales/analógicas.
    - Estados del robot (en movimiento, parado, en error).
    - Variables RAPID (parámetros de proceso, valores de sensores, contadores).

Ejemplo de uso:

- Un SCADA puede leer en tiempo real las posiciones y estados de un robot en RobotStudio.
- Un sistema MES puede recibir datos de producción simulados para validar flujos antes de la puesta en marcha real.
- Una fábrica digital puede simular el comportamiento de toda una celda robotizada conectada a un sistema central.

Esto convierte al robot ABB en un componente integrado dentro de un ecosistema digital, facilitando la transición hacia Industria 4.0.

Capítulo 5

6. SIMULACIÓN

Para controlar las funciones de los robots se ha desarrollado un menú programado en RAPID, empleando interrupciones que permiten al usuario seleccionar diferentes modos de ejecución en tiempo real. Este menú constituye la base de la lógica de la estación, gestionando la interacción musical de los IRB 120 y el acompañamiento del IRB 1400.

Las funciones implementadas fueron las siguientes:

1. Escala ascendente: los robots recorren las teclas del piano en orden ascendente, de notas graves a agudas.
2. Escala descendente: ejecución en sentido contrario, de agudos a graves.
3. Lectura de partitura: interpretación de una secuencia de notas predefinidas en RAPID, emulando una melodía.
4. Notas aleatorias: los robots ejecutan notas al azar, generando un patrón impredecible.

Las primeras pruebas del menú se realizaron utilizando el simulador de señales de E/S digitales de RobotStudio. Este entorno permitió verificar de manera controlada que cada entrada activaba correctamente la función correspondiente en el programa RAPID.

- Se comprobó la correspondencia entre entradas digitales y funciones musicales.
- Se validó el correcto funcionamiento de las interrupciones.
- Se garantizó que la lógica del menú respondía de manera inmediata a los cambios de estado en las señales.

Para nuestra simulación en RobotStudio usaremos un estado inicial. El estado inicial hace referencia a la condición o configuración del robot y su entorno de simulación antes de iniciar la ejecución de un programa o ciclo de trabajo. Este estado es esencial para garantizar que las simulaciones y pruebas se realicen de manera coherente, repetible y segura, ya que define la posición, orientación y estado de los distintos elementos del sistema (robots, herramientas, piezas, transportadores, etc.) al comienzo de una operación.

De manera general, el estado inicial incluye:

1. Posición y configuración del robot: El robot parte de una postura o posición conocida, normalmente definida mediante un robtarget o una instrucción MoveJ hacia una posición de “home” o “inicio”. Esto asegura que todas las trayectorias comiencen desde un punto seguro y controlado.

2. Configuración de ejes y herramientas: Se especifican las herramientas activas (ToolData) y los sistemas de referencia (WObjData) que se usarán al iniciar la simulación.

3. Estado de las señales y dispositivos externos: Las señales digitales o analógicas que comunican el robot con otros sistemas (como PLCs o sensores) pueden tener valores iniciales predefinidos mediante la configuración de Device Mapping o directamente desde el entorno virtual.

4. Restablecimiento del entorno: En simulaciones complejas, el estado inicial puede incluir la posición de piezas, mesas o cintas transportadoras, asegurando que el entorno virtual esté en condiciones idénticas a las del comienzo del proceso real.

En la práctica, establecer un estado inicial correcto permite que la simulación sea reproducible, que las trayectorias no generen colisiones inesperadas y que la ejecución en el controlador físico coincida con lo ensayado en RobotStudio. Nosotros vamos a crear nuestro estado inicial que llamaremos POS_INIT (*Ilustración 128).*

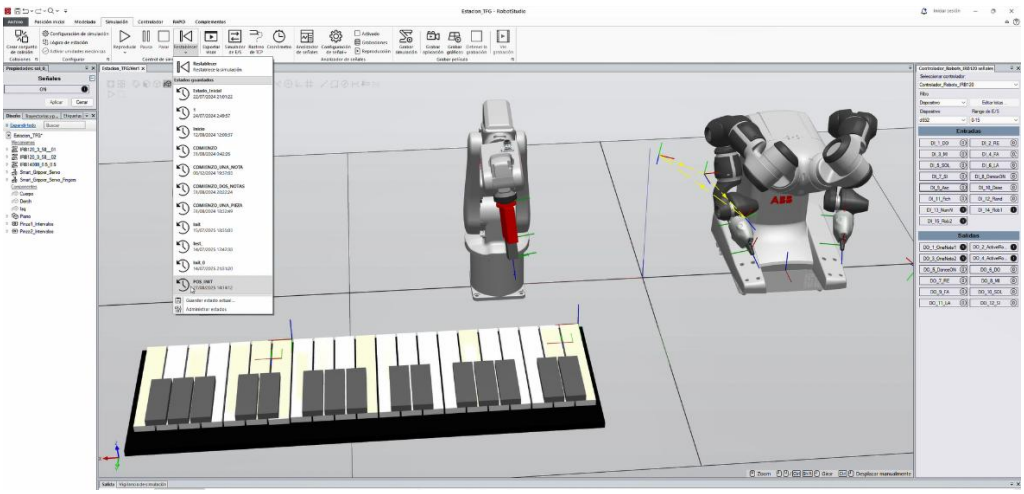


Ilustración 126: Ruta para la creación de un estado inicial en la estación de RobotStudio

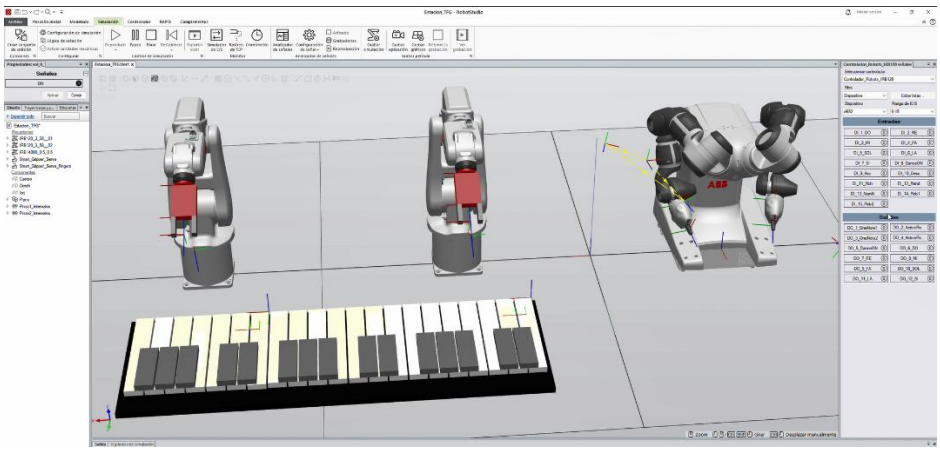


Ilustración 127: Configuración final del estado inicial POS_INIT de nuestra estación de RobotStudio

A continuación, se muestran la correspondencia y coherencia de la lógica entre algunas de las señales durante las primeras pruebas.

Señales de entrada activas: Escala ascendente Activación un IRB 120.

Señales de salida activas: Robot1 Activado.

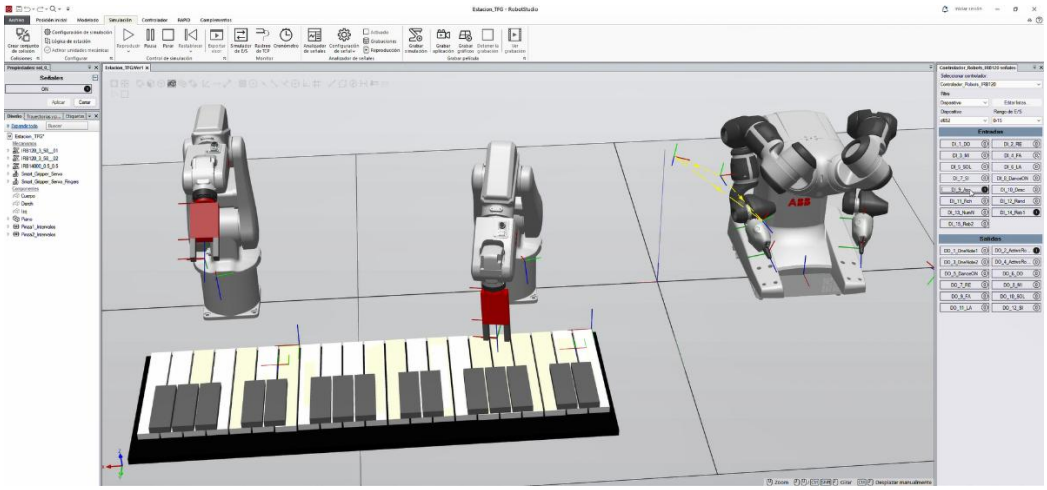


Ilustración 128: Estación de RobotStudio en modo escala ascendente con un IRB 120 activado en movimiento

Señales de entrada activas: Escala ascendente, Activación ambos IRB 120 con Pinza cerrada.

Señales de salida activas: Ambos robots (ABB, 2025) IRB 120 activados y ambas pinzas cerradas.

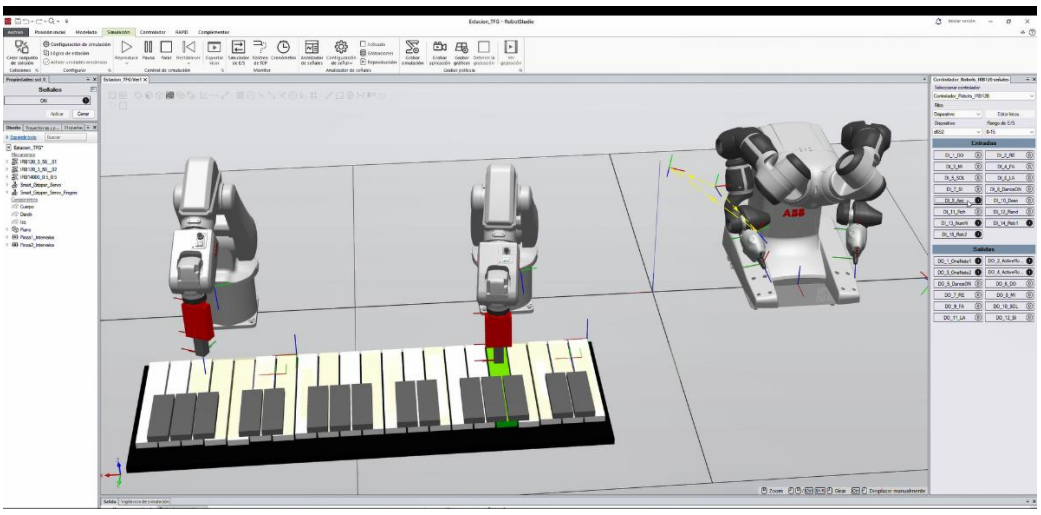


Ilustración 129: Estación de RobotStudio en modo escala ascendente con dos IRB 120 activados en movimiento

Señales de entrada activas: Nota Re , Activación ambos IRB 120 con Pinza cerrada.

Señales de salida activas: Ambos robots (ABB, 2025) IRB 120 activados y ambas pinzas cerradas, Nota Re y Dance del YuMi 1400 que hemos activado mediando pulsador.

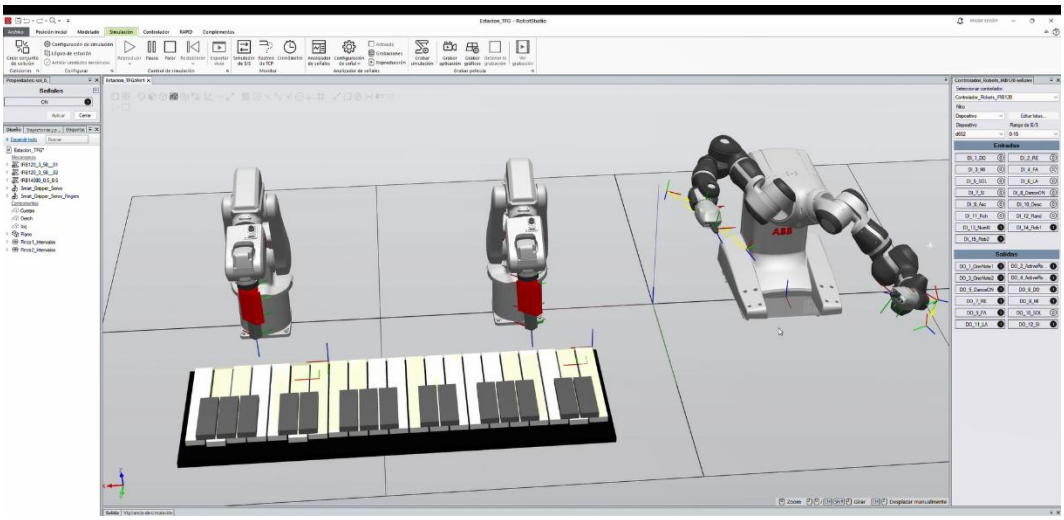


Ilustración 130: Estación de RobotStudio en modo escala ascendente con dos IRB 120 activados en movimiento y el YuMi 1400 bailando (1)

Señales de entrada activas: Nota Re , Activación ambos IRB 120 con Pinza cerrada.

Señales de salida activas: Ambos robots (ABB, 2025) IRB 120 activados y ambas pinzas cerradas, Nota Re y Dance del YuMi 1400 que hemos activado mediando pulsador.

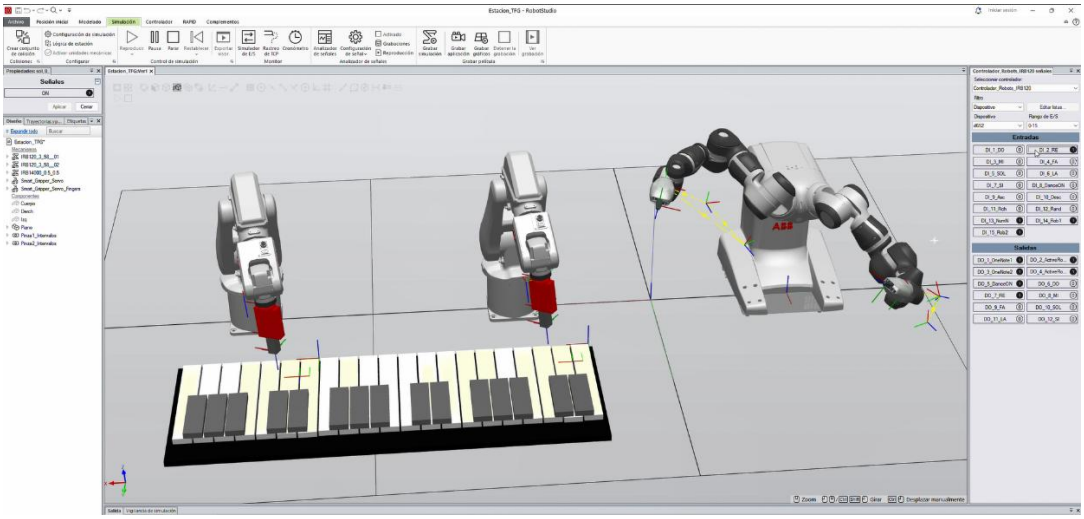


Ilustración 131 : Estación de RobotStudio en modo escala ascendente con dos IRB 120 activados en movimiento y el YuMi 1400 bailando (2)

Señales de entrada activas: Nota Sol , Activación ambos IRB 120 con Pinza cerrada.

Señales de salida activas: Ambos robots (ABB, 2025) IRB 120 activados y ambas pinzas cerradas, Nota Sol y Dance del YuMi 1400 que hemos activado mediando pulsador.

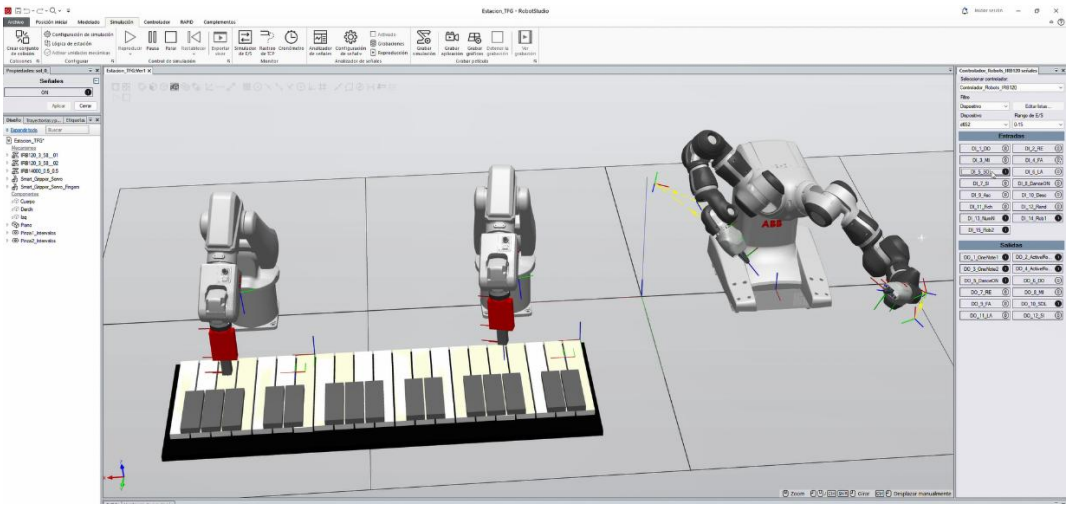


Ilustración 132 : Estación de RobotStudio en modo escala ascendente con dos IRB 120 activados en movimiento y el YuMi 1400 bailando (3)

Este paso resultó esencial para depurar el código RAPID y asegurar la estabilidad del sistema antes de la integración con sistemas externos.

La simulación en RobotStudio permitió comprobar la integración completa del sistema:

- Ejecución precisa de las trayectorias de los IRB 120 en cada modo de funcionamiento.
- Sincronización entre los robots mediante comunicación por socket.
- Respuesta del IRB 1400, que ejecuta movimientos de baile coordinados con la música generada.
- Activación correcta de funciones desde la HMI, confirmando la comunicación fluida vía OPC UA.
- Ausencia de colisiones y correcta coordinación en la estación virtual.

Una vez validado el menú con el simulador de E/S digitales, se procedió a su integración con una pantalla HMI desarrollada en MATLAB App Designer, conectada mediante OPC UA al controlador IRC5.

- El usuario puede elegir desde la pantalla el tipo de ejecución deseada (ascendente, descendente, partitura o aleatoria).

- También puede activar o desactivar el movimiento de los robots.
- Cada selección en la HMI se traduce en señales enviadas al controlador, que RAPID interpreta en tiempo real.

Este avance permitió disponer de una interfaz gráfica intuitiva para interactuar con el sistema, aumentando la usabilidad y acercando el proyecto a un contexto real de supervisión industrial.

Primero iniciamos el programa ABB IRC5 OPC UA Configuration, y confirmamos que somos capaces de leer y escribir tanto variables de nuestro programa de RAPID como entradas y salidas de nuestros controladores.

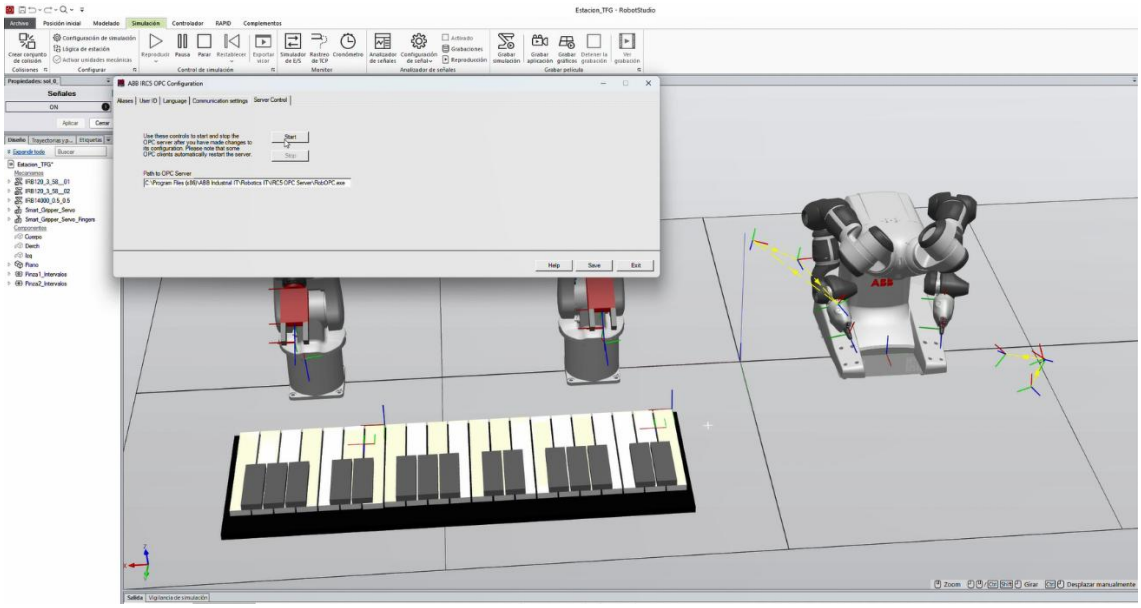


Ilustración 133: Start del programa de la comunicación de comunicaciones OPC UA de ABB

Como se muestra en la Ilustración, desde el programa somos capaces de modificar los valores y visualizarlos.

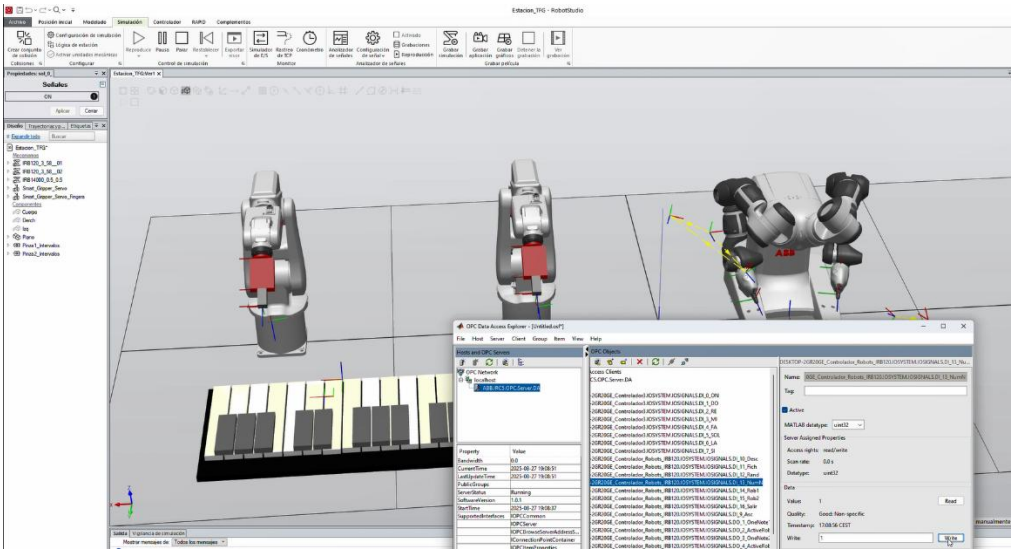


Ilustración 134: Lectura/Escritura de señales y variables de RobotStudio a través de ABB IRC5 OPC UA Configuration (1)

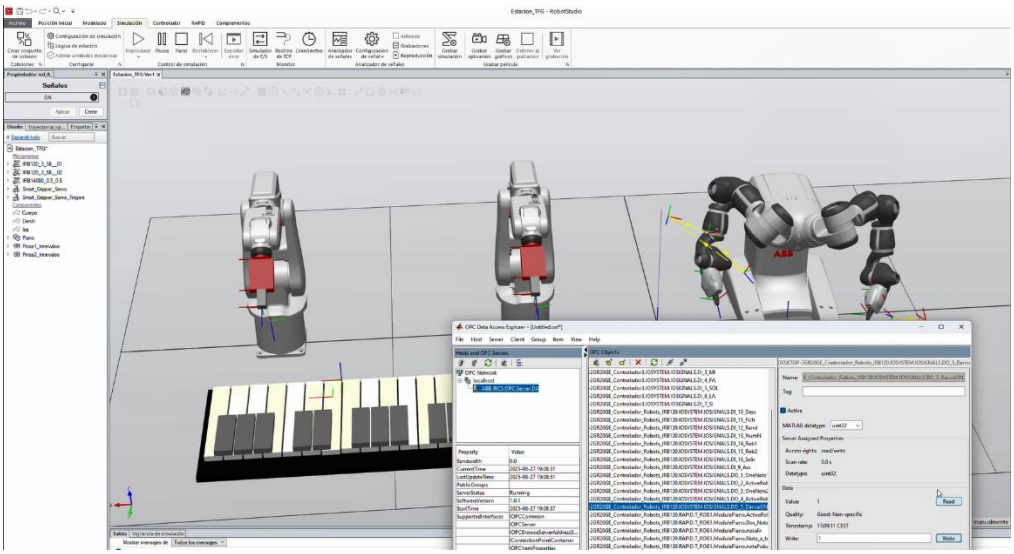


Ilustración 135: Lectura/Escritura de señales y variables de RobotStudio a través de ABB IRC5 OPC UA Configuration (2)

Una vez verificado el correcto funcionamiento de las comunicaciones, ejecutaremos nuestra HMI desde Matlab.

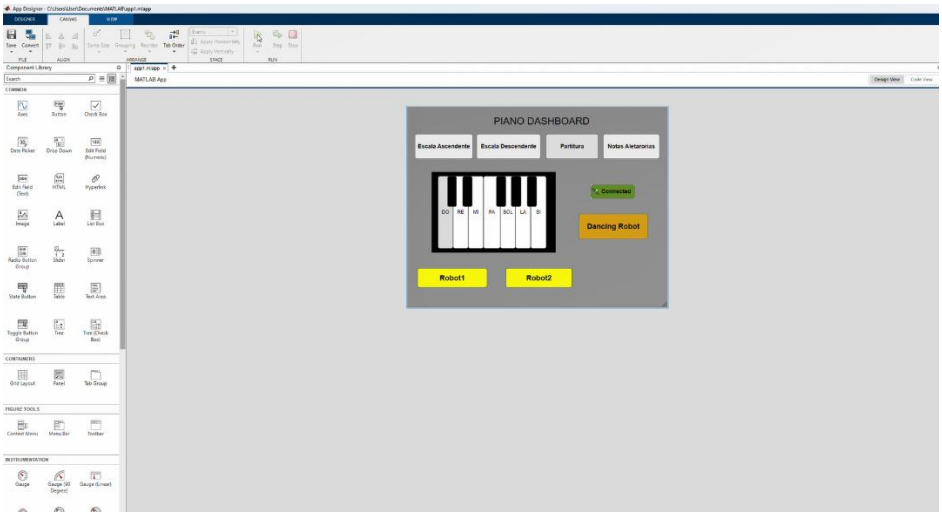


Ilustración 136: Ejecución de la pantalla HMi diseñada desde App Designer de Matlab

Demostración del correcto funcionamiento de nuestra interfaz.

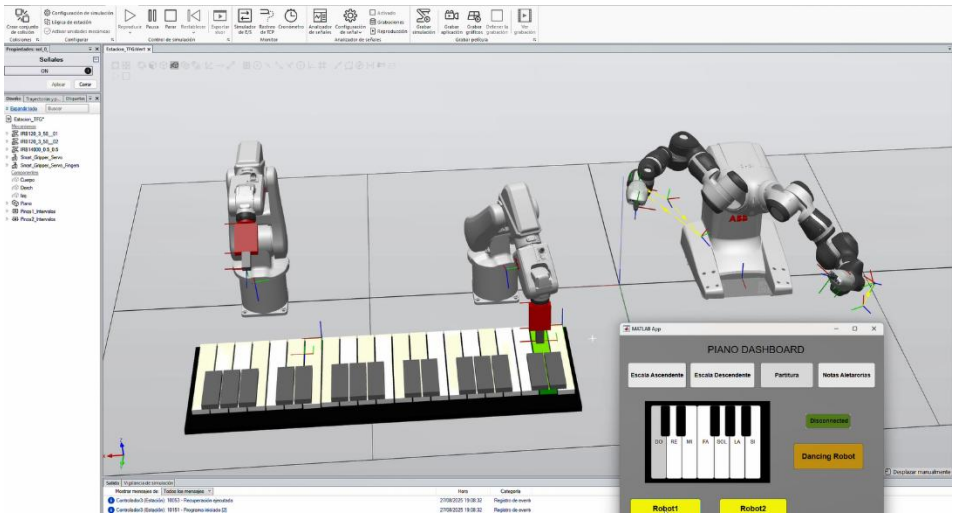


Ilustración 137: Representación real de la ejecución de la función escala ascendente con un robot IRB 120 activado y el YuMi 1400 de la estación a través de la activación de señales desde el HMI diseñado en Matlab (1)

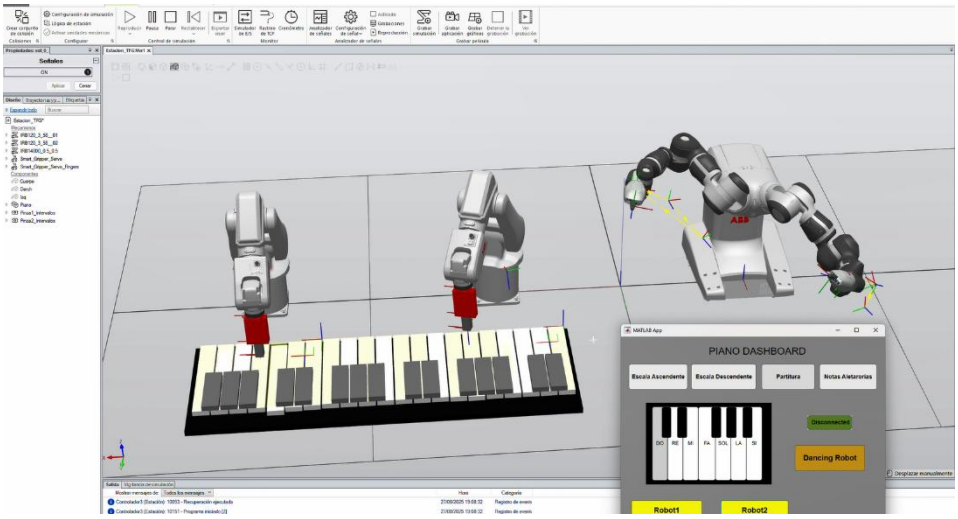


Ilustración 138: Representación real de la ejecución de la función escala ascendente con dos robots IRB 120 activados y el YuMi 1400 de la estación a través de la activación de señales desde el HMI diseñado en Matlab (2)

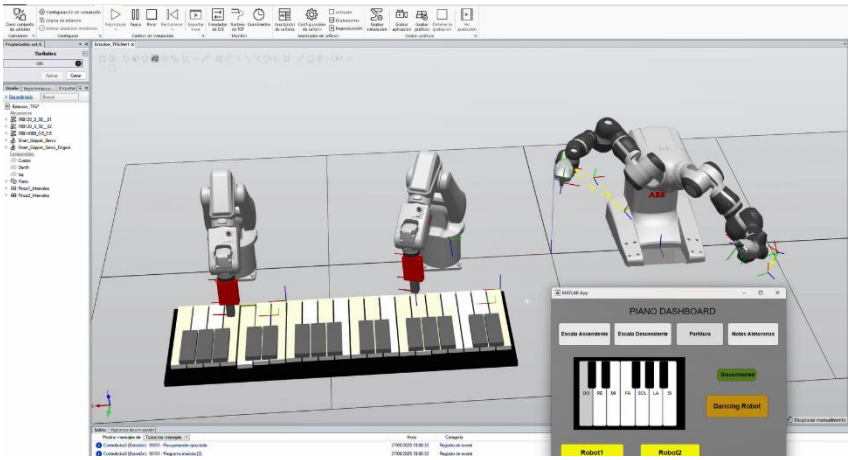


Ilustración 139: Representación real de la ejecución de la función escala ascendente con un robot IRB 120 activado y el YuMi 1400 de la estación a través de la activación de señales desde el HMI diseñado en Matlab (3)

7. RESULTADOS

El proyecto ha permitido desarrollar y validar una estación robótica completa en RobotStudio, integrada por dos robots IRB 120 y un IRB 1400, destacando varios resultados relevantes desde el punto de vista técnico, educativo y de simulación industrial.

En primer lugar, se ha demostrado la eficacia de la comunicación por socket entre los controladores IRC5, logrando un intercambio de datos confiable y de baja latencia. Esta arquitectura permitió coordinar las acciones de los IRB 120 encargados de tocar el teclado de piano y del IRB 1400 encargado de realizar movimientos de baile, sin necesidad de un PLC intermediario. La independencia de la comunicación socket asegura una arquitectura más flexible y escalable, apta para futuras ampliaciones de la estación con más robots o dispositivos externos.

El menú programado en RAPID se ha validado mediante un proceso en dos etapas: inicialmente con el simulador de señales de E/S digitales de RobotStudio, y posteriormente integrando la HMI desarrollada en MATLAB. Esto garantizó que todas las funciones —escala ascendente, escala descendente, lectura de partitura, notas aleatorias y control de pinza abierta/cerrada, así como la activación y desactivación de los distintos robots que componen la estación, respondieran correctamente a las órdenes del usuario en tiempo real. La implementación de interrupciones en RAPID permitió cambios inmediatos entre modos, asegurando un comportamiento predecible y confiable de los robots en todas las situaciones.

El IRB 1400 se integró exitosamente como robot acompañante, mostrando que es posible asignar tareas diferenciadas dentro de la misma celda robótica. Este robot interpreta las señales enviadas por los IRB 120 y genera movimientos sincronizados con la música, demostrando la eficacia de la coordinación entre múltiples robots en tiempo real. La simulación confirmó la correcta sincronización y la ausencia de conflictos o colisiones, asegurando la seguridad y eficiencia de la operación conjunta.

Por otro lado, la interfaz HMI en MATLAB, conectada a los controladores mediante OPC UA, permitió una gestión remota e intuitiva del sistema. A través de esta interfaz, el usuario puede monitorizar el estado de los robots, activar o desactivar funciones del menú y controlar la pinza de cada robot. Esto no solo valida la integración de sistemas externos con los controladores IRC5, sino que también evidencia la aplicabilidad de OPC UA como estándar industrial para comunicación en tiempo real.

La simulación en RobotStudio se mostró altamente fiable, permitiendo validar trayectorias, tiempos de ciclo, secuencias de movimiento y sincronización de todos los robots antes de cualquier implementación física. Esto asegura que el sistema puede trasladarse posteriormente a un entorno real sin modificaciones estructurales significativas, reduciendo riesgos y costos de puesta en marcha.

Finalmente, desde el punto de vista educativo, el proyecto constituye una herramienta didáctica de alto valor, permitiendo a los estudiantes comprender de manera práctica conceptos avanzados de programación en RAPID, comunicación entre controladores y diseño de interfaces HMI. La combinación de interacción musical y movimiento robótico genera un entorno motivador, facilitando el aprendizaje aplicado y fomentando la creatividad en la resolución de problemas industriales.

En conjunto, los resultados muestran que el sistema desarrollado es robusto, escalable y pedagógicamente valioso, capaz de integrarse tanto en contextos educativos como en proyectos de automatización industrial, demostrando su potencial como base para futuras ampliaciones y aplicaciones en entornos de fábrica digital o Industria 4.0.

Capítulo 6

8. CONCLUSIONES

El desarrollo de la estación robótica en RobotStudio con dos manipuladores IRB 120 y un IRB 1400 ha permitido demostrar la viabilidad de integrar múltiples robots ABB en un mismo entorno colaborativo, coordinados tanto a nivel de programación como de comunicaciones industriales.

En primer lugar, se ha validado el uso de Socket Messaging como mecanismo de intercambio de información entre los distintos controladores, estableciendo una arquitectura distribuida sin necesidad de un PLC intermedio. Este enfoque ha permitido que los IRB 120 ejecuten la interacción con el componente inteligente diseñado —el teclado de piano—, mientras que el IRB 1400 adapta su comportamiento a modo de baile en función de la información recibida. La utilización de interrupciones y menús programados en RAPID ha facilitado la creación de un sistema flexible y modular, capaz de responder en tiempo real a las distintas entradas del proceso.

En segundo lugar, la integración de la estación con un entorno externo de supervisión a través del estándar OPC UA ha ampliado las capacidades del sistema, posibilitando la comunicación entre los controladores IRC5 y una interfaz gráfica desarrollada en MATLAB App Designer. Esto ha permitido dotar al usuario de una herramienta intuitiva y práctica, desde la cual puede monitorizar el estado de los robots y enviar órdenes directamente al programa, acercando el proyecto a un contexto real de Industria 4.0.

Desde una perspectiva educativa, el proyecto persigue como propósito fundamental familiarizar a los estudiantes con las tecnologías y metodologías propias de la robótica industrial, utilizando RobotStudio como entorno de diseño, simulación y validación. La estación creada constituye un recurso didáctico innovador que permite comprender de manera práctica conceptos clave como la programación en RAPID, la comunicación entre controladores, la integración de interfaces externas y el diseño de aplicaciones robóticas interactivas. De este modo, se potencia el aprendizaje aplicado y se fomenta la creatividad en la resolución de problemas complejos.

El uso combinado de RobotStudio como plataforma de simulación y programación, junto con las tecnologías de comunicación implementadas, ha demostrado el potencial de los entornos virtuales para validar celdas robóticas antes de su implementación física. Asimismo, este trabajo

pone de manifiesto la importancia de la interoperabilidad entre robots, controladores y sistemas externos, condición indispensable en entornos productivos actuales y futuros.

En conclusión, el proyecto ha logrado no solo la simulación de un sistema innovador que combina interacción musical y movimiento robótico, sino también la creación de una infraestructura de comunicación robusta y escalable. Este enfoque abre la puerta a futuras aplicaciones en ámbitos educativos, de entretenimiento e industriales, en los que la colaboración entre múltiples robots y la interacción hombre-máquina son cada vez más relevantes.

Líneas de trabajo futuro

Con el objetivo de ampliar y mejorar el sistema desarrollado, se plantean diversas líneas de evolución:

1. Implementación física de la estación: trasladar la simulación a un entorno real con robots ABB, validando el comportamiento de las comunicaciones y la interfaz en un escenario productivo.

2. Optimización de la interacción musical: perfeccionar el reconocimiento y traducción de notas musicales hacia movimientos robóticos más complejos y sincronizados.

3. Incorporación de visión artificial: integrar sistemas de visión y algoritmos de procesamiento de imagen para que los robots puedan reaccionar a estímulos visuales o gestuales, además de los sonoros.

4. Ampliación de la interfaz de usuario: evolucionar la aplicación desarrollada en MATLAB hacia plataformas más accesibles, como HMI industriales, aplicaciones web o móviles.

5. Orientación hacia entornos educativos: consolidar el sistema como herramienta didáctica en laboratorios universitarios o de formación profesional, permitiendo a los estudiantes experimentar con programación, simulación y control de robots.

9. Bibliografía

Referencias

ABB. (2004-2017). *Tecnichal reference manual, RAPID Instructions, Functions and Data types.* Obtenido de chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://library.e.abb.com/public/b227fcd260204c4dbeb8a58f8002fe64/Rapid_instructions.pdf

ABB. (2020-2022). *Application manual - IRC5 OPC UA Server.* Obtenido de chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://library.e.abb.com/public/68db219ad97b4119b1d42170a2872a5a/3HAC074394%20AM%20IRC5%20OPC%20UA%20Server-en.pdf?x-sign=buiuR9xHIrMxYsX2rUUIBdfycerHrvQXxlrOEBDhcwJnJEPuRrdVD/MjSEMi7Wuv

ABB. (2022). *Technical reference manual - RAPID kernel.* Obtenido de https://library.e.abb.com/public/f23f1c3e506a4383b635cff165cc6993/3HAC050946%20TRM%20RAPID%20Kernel%20RW%206-en.pdf?x-sign=Pshy8WRY3W/ZXxWomdcYd7x/zaMRc8NYHY5S/ybBrKSUa638uSsiKt1Rlq27VeH9

ABB. (2025). *Manual del operador - RobotStudio.* Obtenido de https://search.abb.com/library/Download.aspx?DocumentID=3HAC032104-005&LanguageCode=es&DocumentPartId=&Action=Launch

ABB. (2025). *Operating manual*. Obtenido de RobotStudio: https://search.abb.com/library/Download.aspx?Action=Launch&DocumentID=3HAC032104-001&DocumentPartId=&LanguageCode=en&utm_source=chatgpt.com

ABB. (2025). *RobotStudio Desktop*. Obtenido de RobotStudio Desktop: https://new.abb.com/products/robotics/es/software-y-digital/robotstudio/robotstudio

Bruno Siciliano, L. S. (2010). *Robotics Modelling, Planning and Control.* Springer.

Garrido, G. M. (2015). *TFG-P-252.* Obtenido de En conjunto, estos proyectos reflejan la evolución y el interés continuo de la Universidad de Valladolid en la integración de la robótica industrial, la simulación mediante RobotStudio y el desarrollo de aplicaciones educativas que favorecen el aprendizaj: https://uvadoc.uva.es/handle/10324/852

Herrero, J. A. (Septiembre de 2015). *Modelado de una célula robótica con fines educativos usando el programa Robot-Studio.* Obtenido de chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://uvadoc.uva.es/bitstream/handle/10324/14441/TFG-P-309.pdf;jsessionid=E92FCB1651F6A4CC177887374BDD91FA?sequence=1

Jiménez, C. J. (1 de Julio de 2019). *DISEÑO DE UN SISTEMA ROBÓTICO EDUCATIVO PARA JUGAR AL AJEDREZ CON ROBOTS INDUSTRIALES .* Obtenido de chrome-

extension://efaidnbmnnnibpcajpcglclefindmkaj/https://uvadoc.uva.es/bitstream/handle/10324/36877/TFG-I-1168.pdf?sequence=1&isAllowed=y

Jorge Elices. (30 de Julio de 2020). *National Geographic*. Obtenido de https://www.nationalgeographic.com/history/history-magazine/article/ismail-al-jazari-muslim-inventor-called-father-robotics

Kak, S. (2011). En *The Beginning of Robotics.* (págs. 142 - 146). IEEE Access.

Mata, E. P. (2022). *Simulación de un Robot Colaborativo YuMi (ABB) en entorno RobotStudio comandado desde MATLAB mediante protocolo OPC UA para tocar un Xilófono.* Obtenido de https://www.bing.com/ck/a?!&&p=bda7e830a913214ed0a157038b87c285354ce983f35b0bf14084ba0839b2fcfbJmltdHM9MTc2MTE3NzYwMA&ptn=3&ver=2&hsh=4&fclid=0411548b-ddd3-6dbc-15c2-4138dc676c93&psq=elena+pozas+mata+tfg&u=a1aHR0cHM6Ly91dmFkb2MudXZhLmVzL2hhbmRsZS8xMDMyNC8

MathWorks. (2025). *Code view in App Designer*. Obtenido de MATLAB & Simulink Documentation: https://es.mathworks.com/help/matlab/creating_guis/about-app-designer.html

Mathworks. (2025). *Create and edit apps in App Designer*. Obtenido de MATLAB & Simulink Documentation: https://es.mathworks.com/help/matlab/creating_guis/code-view.html

Matworks. (2025). *Mathworks.* Obtenido de https://es.mathworks.com/help/matlab/index.html?s_tid=CRUX_lftnav

Robótica. (5 de Septiembre de 2023). *¿Cúales son las partes de un robot?* Obtenido de https://www.esneca.lat/blog/partes-robot-caracteristicas/

Wikipedia. (27 de Agosto de 2024). *Wikipedia* . Obtenido de https://es.wikipedia.org/wiki/Robot