Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Desarrollo de Estación de Impresión 3D con Arcilla para un Robot ABB IRB 4600

Autor: David Toro Coronado
Tutor: Dr. Federico Cuesta Rojo

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado
Ingeniería Electrónica, Robótica y Mecatrónica

Desarrollo de Estación de Impresión 3D con Arcilla para un Robot ABB IRB 4600

Autor:
David Toro Coronado

Tutor:
Dr. Federico Cuesta Rojo
Profesor titular

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Trabajo Fin de Grado: Desarrollo de Estación de Impresión 3D con Arcilla para un Robot ABB IRB 4600

Autor: David Toro Coronado

Tutor: Federico Cuesta Rojo

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2023

El Secretario del Tribunal

A mis padres

A mis compañeros

A Enrique y Juan Carlos

A David Gómez

Agradecimientos

En el transcurso de esta enriquecedora experiencia por la Escuela de Ingeniería, deseo expresar mi profundo agradecimiento a quienes han sido mi inquebrantable apoyo a lo largo de este camino académico. En primer lugar, quiero destacar la importancia de mis padres, cuya dedicación constante y fe inquebrantable en mí han sido la fuente de inspiración detrás de cada logro alcanzado. Asimismo, deseo agradecer a mis amigos, compañeros de viaje en esta emocionante etapa académica, sus palabras de aliento, su amistad y apoyo moral que han sido un pilar sobre el que apoyarse.

Además, mi gratitud se extiende a dos figuras fundamentales en mi desarrollo académico y el de este proyecto: Federico Cuesta, mi tutor, cuyas orientaciones y enseñanzas han sido cruciales para la culminación de este proyecto y, también, a Enrique Vázquez, director del FabLab, a quien le estoy profundamente agradecido por confiar en mí y brindarme la libertad necesaria para abordar de manera independiente este desafío. Esta autonomía no solo ha permitido mi crecimiento como profesional, sino que también ha sido un motor que impulsó mi creatividad y mi capacidad para asumir la responsabilidad de mis propias decisiones. Su confianza ha sido un factor determinante en el éxito de este trabajo y en mi desarrollo personal y académico.

David Toro Coronado

Sevilla, 2023

Resumen

Este proyecto, nace de la necesidad de crear una estación de fabricación aditiva con arcilla en el Laboratorio de Fabricación de la Facultad de Arquitectura de la Universidad de Sevilla

Los equipos que se adquirieron para este proyecto son, primero, un Robot ABB IRB4600 con capacidad de carga de 60Kg, gobernado por un control IRC5 de ABB y, segundo, un extrusor WASP3D XL para impresoras 3D de arcilla.

El desafío surge cuando se busca integrar ambos equipos, ya que ninguno de los fabricantes proporciona una compatibilidad nativa, lo que implica la necesidad de una solución personalizada para lograr que trabajen conjuntamente. Además, no existe software comercial disponible que permita la impresión directa con estos robots, lo que ha requerido el desarrollo de un software dedicado basado en la integración de diversas aplicaciones.

El propósito de esta investigación es analizar el estado actual de la impresión con robots ABB y arcilla, y a partir de este análisis, diseñar el hardware abordando los desafíos de compatibilidad y seleccionando los componentes más adecuados. Además, se desarrollará el software necesario para operar la Estación de Impresión desde un programa CAD que permitirá tanto el diseño de modelos como el envío de instrucciones a la estación para su impresión. Esta iniciativa busca establecer una base sólida para la fabricación con arcilla en el entorno del FabLab, proporcionando una plataforma desde la cual se podrán realizar avances significativos en este campo.

El trabajo presentado en este documento muestra el estado del arte de la impresión con Robots y arcilla. También, la adaptación del extrusor al brazo ABB mediante una anilla impresa en 3D, la interconexión efectiva entre ambos equipos y la incorporación de un Arduino Uno para el control del motor paso a paso del extrusor. Asimismo, se detalla el diseño y la configuración de la estación en RobotStudio, el uso de la librería COMPAS FAB y la creación de software personalizado para habilitar la impresión desde Rhino 7 utilizando el robot IRB4600.

Los objetivos alcanzados en este proyecto han sido los siguientes:

1. Integración del extrusor al sistema del ABB IRB4600.
2. Comunicación mediante ROS exitosa entre Rhino 7 y el controlador IRC5.
3. Diseño del programa de procesado de modelos 3D e impresión en Grasshopper.

Abstract

This project arises from the need to create an additive manufacturing station for clay in the Manufacturing Laboratory of the School of Architecture at the University of Seville.

The equipment acquired for this project consists of, firstly, an ABB IRB4600 robot with a 60Kg payload capacity, controlled by an ABB IRC5 controller, and secondly, a WASP3D XL extruder for clay printers.

The challenge appears when attempting to integrate both pieces of equipment since neither of the manufacturers provides native compatibility, necessitating a custom solution to make them work together. Furthermore, there is no commercial software available that allows direct printing with these robots, which has required the development of dedicated software based on the integration of various applications.

The purpose of this research is to analyze the current state of printing with ABB robots and clay, and based on this analysis, design the hardware to address compatibility challenges and select the most suitable components. Additionally, the necessary software will be developed to operate the Printing Station from a CAD program that will enable both model design and sending instructions to the station for printing. This initiative aims to establish a solid foundation for clay manufacturing within the FabLab environment, providing a platform from which significant advancements in this field can be made.

The work presented in this document showcases the state of the art in printing with Robots and clay. It also covers the adaptation of the extruder to the ABB arm using a 3D-printed ring, effective interconnection between both pieces of equipment, and the incorporation of an Arduino Uno for controlling the extruder's stepper motor. Furthermore, it details the design and configuration of the station in RobotStudio, the use of the COMPAS FAB library, and the creation of custom software to enable printing from Rhino 7 using the IRB4600 robot.

The objectives achieved in this project are as follows:

1. Integration of the extruder into the ABB IRB4600 system.

2. Successful ROS communication between Rhino 7 and the IRC5 controller.

3. Design of the 3D model processing and printing program in Grasshopper.

Índice

Índice de Figuras

Índice de Tablas

Notación

FabLab	Laboratorio de Fabricación Digital
US	Universidad de Sevilla
ETSI	Escuela Técnica Superior de Ingeniería
3D	3 Dimensiones
CF	COMPAS FAB
RRC	Rapid Robot Comunication
ABB	Asea Brown Boveri
IRC5	Industrial Robot Controller
IRB	Industrial Robot
ETH	Swiss Federal Institute of Technology in Zürich
IAAC	Institute for Advance Architecture of Catalonia
CAD	Computer Aided Design
CNC	Computer Numerical Control
FDM	Fused Deposition Modeling
SLA	Stereolithography

1. INTRODUCCIÓN Y OBJETIVOS

1.1. Estación de prototipado en FabLab

Una estación de prototipado tiene la función de llevar a la realidad los modelos, diseños e ideas de los ingenieros, artistas y cualquier usuario de esta. El desarrollo de prototipos puede realizarse siguiendo múltiples técnicas que puedan adecuarse mejor a los resultados esperados existiendo opciones en un amplio rango de precisión y fidelidad respecto al modelo digital. Esto es un paso clave en el desarrollo de productos porque permite a los ingenieros y diseñadores probar diferentes ideas y conceptos antes de invertir grandes cantidades de tiempo y recursos en la producción de un sistema completo. Además, los prototipos de piezas pueden ayudar a identificar errores existentes que hayan escapado a la fase de verificación del producto durante su diseño permitiendo mejorar la funcionalidad y desempeño del producto.

En la Universidad de Sevilla tenemos múltiples talleres y laboratorios donde los estudiantes, profesores e investigadores pueden llevar a cabo estudios y proyectos, así como crear versiones físicas de sus diseños. En estos espacios, se dispone de una amplia variedad de herramientas y maquinaria dependiendo del ámbito educativo en el que se encuentren.

Un ejemplo de esto es **FabLab** en la Universidad de Sevilla, situado en la facultad de arquitectura, el Laboratorio de Fabricación Digital cuenta con herramientas avanzadas, como cortadoras láser para madera, cartón y metal; fresadoras CNC, impresoras 3D, software de diseño asistido por ordenador (CAD) además de talleres para la construcción, refinamiento y optimización de maquetas y estructuras [1].

En este espacio, los estudiantes, profesores e investigadores pueden llevar a cabo proyectos de diseño y fabricación digital en diversas áreas, desde la arquitectura y el diseño industrial hasta la electrónica y la robótica. Además, el FabLab fomenta la colaboración y el intercambio de conocimientos, lo que permite a los usuarios aprender de sus compañeros y mejorar sus habilidades técnicas y creativas.

Para ampliar los campos de investigación y estudio, y mejorar la capacidad de producción del FabLab se ha implementado una estación de prototipado con un robot ABB IRB4600. Este robot se ha configurado para realizar impresiones en 3D con arcilla, lo que permite explorar nuevas posibilidades en el ámbito del diseño arquitectónico y la fabricación de piezas artísticas y funcionales gracias al nuevo material disponible. El robot es capaz de realizar figuras 3D de gran tamaño comparado con las posibilidades que ofrece una impresora 3D convencional. Además de esto, también amplía la formación y oportunidades del laboratorio al permitir avanzar en el campo de impresión robótica con otro tipo de algoritmos, configuraciones o herramientas.

La modificación, ajuste y programación de un robot ABB para realizar esta tarea no es para nada sencilla ni convencional, especialmente si tenemos en cuenta la versatilidad que debe permitir en su funcionamiento más básico y los componentes que debemos añadir al propio robot. Este proyecto recoge todo el trabajo que se ha realizado para conseguir este objetivo partiendo desde un robot de fábrica.

1.2. Estado del arte

La impresión 3D, también llamada **fabricación aditiva**, es una tecnología que consiste en segmentar un modelo digital en capas para poder crear objetos físicos mediante la adición de material, estas capas pueden disponerse de múltiples formas y con distinta metodología resultando en los diferentes tipos de impresión 3D que existen actualmente. Uno de los más extendidos es la impresión por FDM, por sus siglas en inglés, Fused Deposition Modeling; utiliza habitualmente un material plástico fundido el cual se extruye por capas creando la figura deseada. La facilidad de uso junto con su bajo coste permitió un avance rápido en el mercado situándose como referencia a la hora de realizar proyectos que incluyan impresión 3D.

Dentro del mercado de impresoras 3D FDM encontramos una gran variedad de opciones con precios desde 100-200€ para las más económicas y hasta 1000-3000€ para las de alta gama, en el sector doméstico o particular, si buscamos para el sector industrial los precios ascienden hasta los 100000€ como la serie FDM de stratasys [2].

Figura 1.1. Prusa i3 (Impresora FDM).

Otro tipo de impresión bastante extendido es el **SLA** (StereoLithograohy Apparatus) donde se utilizan resinas fotosensibles que se solidifican mediante un láser UV de alta potencia, de igual manera que en las impresoras FDM, el modelo digital se segmenta en capas para que la impresora pueda unir lámina a lámina la resina formando el objeto tridimensional. El funcionamiento de este tipo de impresoras es más complejo y costoso que el anterior ya que la impresión se realiza, pero podemos obtener resultados mucho más precisos.

Si comparamos los precios con las anteriores el rango se eleva significativamente encontrando las opciones más económicas cerca del millar de euros, y la gama alta pudiendo alcanzar cinco cifras.

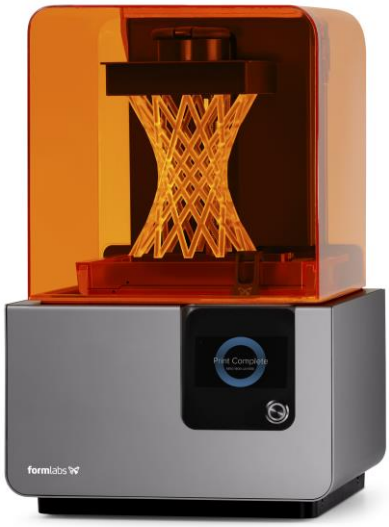


Figura 1.2. Form 2 (Impresora SLA).

Estos dos tipos son los más extendidos, pero existen algunos menos populares como el SLS (Selective Laser Sintering) utilizado habitualmente en la industria debido a su alta velocidad y gran volumen de fabricación. Utiliza un láser para solidificar un material en polvo, típicamente polvo metálico o nylon, capa a capa dando forma a una pieza 3D [3].

Esta tecnología permite imprimir objetos con un alto grado de detalle además de geometrías complejas, por lo que sumado a que permiten utilizar metal como material de impresión las hace la opción predilecta en industrial como la automovilística o aeronaval. Si bien los equipos de impresión SLS son más costosos que el resto de las opciones tienen la ventaja de poder crear piezas duraderas y de gran tamaño [4].

1.2.1. Impresión con arcilla y materiales similares

Anteriormente hemos explicado algunos de los métodos de impresión más extendidos y pese a que tiene grandes diferencias y aplicación tienen un factor común, los materiales que emplean. Tanto las impresoras FDM como las SLA utilizan polímeros plásticos, pero en distintas formas; para las impresoras FDM existen filamentos compuestos de ABS (Acrilonitrilo butadieno estireno) de igual manera que para las impresoras SLA existen resinas con base ABS.

Si bien existen materiales con características únicas como conductividad térmica, resistencia a productos químicos o incluso propiedades antibacterianas [5]; todos comparten características comunes. En primer lugar, todos los materiales de impresión son capaces de fundirse y volverse maleables a temperaturas relativamente bajas. En segundo lugar, la fuerza que pueden soportar antes de romperse no es muy elevada por lo que pueden no servir si van a estar sometidos a esfuerzos constantemente.

Una limitación existente en todas las impresoras existentes en el mercado es el **tiempo de impresión**. Las impresoras 3D se caracterizan por emplear largos periodos de tiempo para completar un trabajo y eso es en parte debido al volumen máximo que pueden extruir. La mayoría de extrusores tienen una boquilla de 0.4mm con lo que se pueden conseguir 15mm^3/s, con esta velocidad de impresión si queremos imprimir un cubo macizo de 5cm de lado necesitaríamos aproximadamente 2h y 30min [6].

Debido a estas limitaciones y a la necesidad de llevar la impresión 3D a nuevos sectores, se está avanzando en la impresión con materiales alternativos a los plásticos como pueden ser la arcilla, la cera u otros compuestos de origen orgánico. Estos materiales tienen ventajas respecto a los plásticos, como la capacidad de producir piezas más grandes y con una geometría más atractiva, una mayor eficiencia en este tipo de impresiones y un acabado y textura más natural.

La **impresión con arcilla** se está popularizando en los últimos años, especialmente en el campo de la cerámica y la escultura gracias a que permite producir formas que serían imposibles de manera manual y, además, obtener un objeto final de calidad en comparación con los plásticos.

El proceso de crear una pieza de arcilla es más complejo que con materiales plásticos ya que después de la impresión es necesario hornear las piezas a alta temperatura en un horno de cerámicas para que el material expulse la humedad y se vuelva poroso, este proceso puede variar en función del material exacto y la figura desarrollada. Una vez está completado, se puede decorar con diferentes pinturas que le pueden aportar propiedades como la resistencia al agua. Para finalizar hay que introducirla en el horno de nuevo para que se endurezca completamente [7].

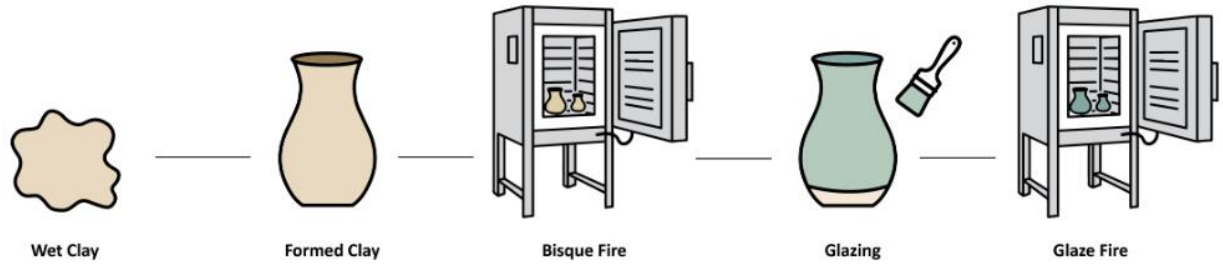


Figura 1.3. Proceso de fabricación de piezas con arcilla y cerámica [8].

Las impresoras 3D de cerámica son muy similares a las convencionales FDM ya que utilizan el mismo principio depositando capas de material sucesivamente. En el mercado existen varias empresas que se dedican a la impresión con arcilla como WASP3D, una compañía italiana que comercializa impresoras y extrusores para arcilla, así como todas las piezas necesarias como los tanques de almacenamiento o conectores necesarios.

Una de sus impresoras más populares es la Delta WASP 2040, la cual tienen una configuración delta, utiliza un sistema de aire comprimido para bombear la arcilla desde un tanque presurizado a 8 bares hasta el extrusor. El extrusor está especialmente diseñado para esta tarea por lo que tienen un tornillo sin fin que empuja el material hasta la boquilla.

Existen múltiples configuraciones, así como diferentes tamaños de boquilla que permitirán obtener resultados más precisos, con boquillas de menor diámetro, o impresiones más rápidas, con mayor diámetro. Los materiales que emplean pueden ser arcillas, porcelanas, cerámicas e incluso compuestos más avanzados como zirconio [8].

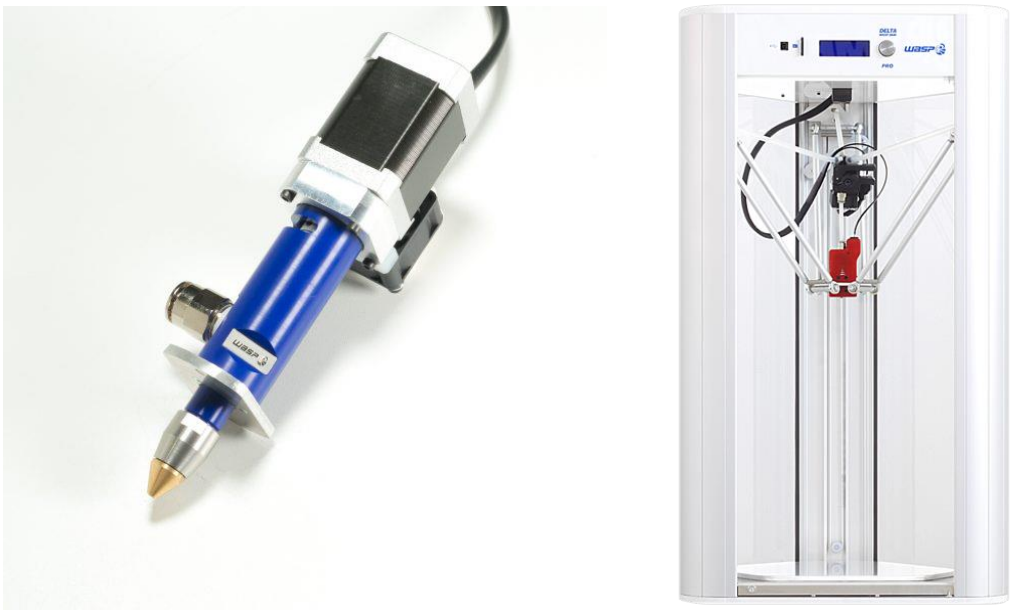


Figura 1.4. Extrusor WASP 3.0 e Impresora Delta WASP 2040 PRO

1.2.2. Impresión 3D con brazos robóticos

Los brazos robóticos comenzaron a desarrollarse en la industria hace aproximadamente 60 años, en ese momento desarrollaban tareas muy sencillas y repetitivas, pero con los avances en el control y programación de estos, se les ha dotado de una mayor flexibilidad, siendo capaces de ser integrados en sistemas totalmente autónomos y de realizar tareas de precisión.

Entre las últimas innovaciones que encontramos en este sector podemos destacar la impresión 3D con brazos robóticos. Esta técnica está todavía en desarrollo y existen multitud de universidades y empresas privadas que están enfocando investigaciones en esta nueva rama. El uso de brazos para la impresión tiene un gran potencial gracias a que combina la capacidad de un brazo de moverse con precisión y rapidez con la capacidad de una impresora o extrusor de añadir material progresivamente permitiendo crear construcción más grandes, complejas y baratas.

Una impresora 3D **FDM** utiliza un sistema de movimiento cartesiano que permite al cabezal moverse a lo largo de los ejes X, Y, y Z. En cambio, un brazo robótico tiene, habitualmente, 6 articulaciones rotativas permitiendo 6 grados de libertad en lugar de 3. Este amplio rango de movimiento permite utilizar un sistema de posicionamiento cartesiano para el extrusor, situado el final del brazo, pero además permite controlar la orientación con la este alcanza los puntos objetivos. Esto aumenta las posibilidades de impresión, pero para su uso requiere de un sistema de control mucho más complejo que una impresora 3D, requiriendo habitualmente de varias capas de software para hacerlo funcionar [9].

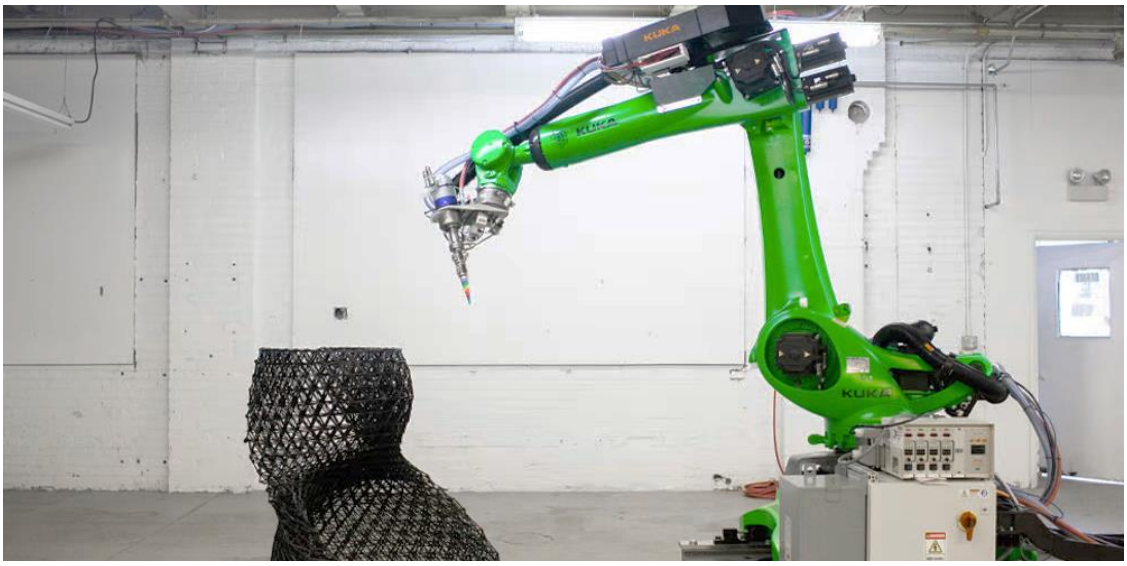


Figura 1.5. Brazo KUKA utilizado por Branch Technology para impresión 3D.

Esta complejidad de software tiene también una parte positiva ya que permite utilizar diferente metodología para alcanzar los resultados deseados, por ejemplo, no es necesario ceñirse a la impresión por capas ya que los robots pueden hacer movimientos más complejos y, con el software adecuado, realizar movimientos tridimensionales para construir objetos sin soportes, uno de los grandes problemas de las impresoras 3D y que hasta la fecha sigue en investigación para reducir su uso.

Las posibilidades se abren hasta el punto de poder utilizar software CAD para generar los diseños e imprimirlos directamente, además de esto, y sobre todo al tratarse una rama muy reciente, los equipos y sistemas utilizados no están diseñados específicamente para esta labor por lo que es necesaria una integración de sistemas de empresas diferentes, lo que aumenta la complejidad pero permite crear un sistema que se adapte mejor a las necesidades específicas de cada cliente [10].

Al tratarse de impresión de gran tamaño no es habitual encontrar robots que impriman con filamento plástico al igual que las impresoras 3D, sí que es posible conseguir un sistema robótico de ese estilo, pero estaría limitado por volumen que es capaz de calentar y enfriar además del elevado coste en piezas grandes. En estos casos se suelen utilizar robots de menor tamaño como los ABB IRB120.

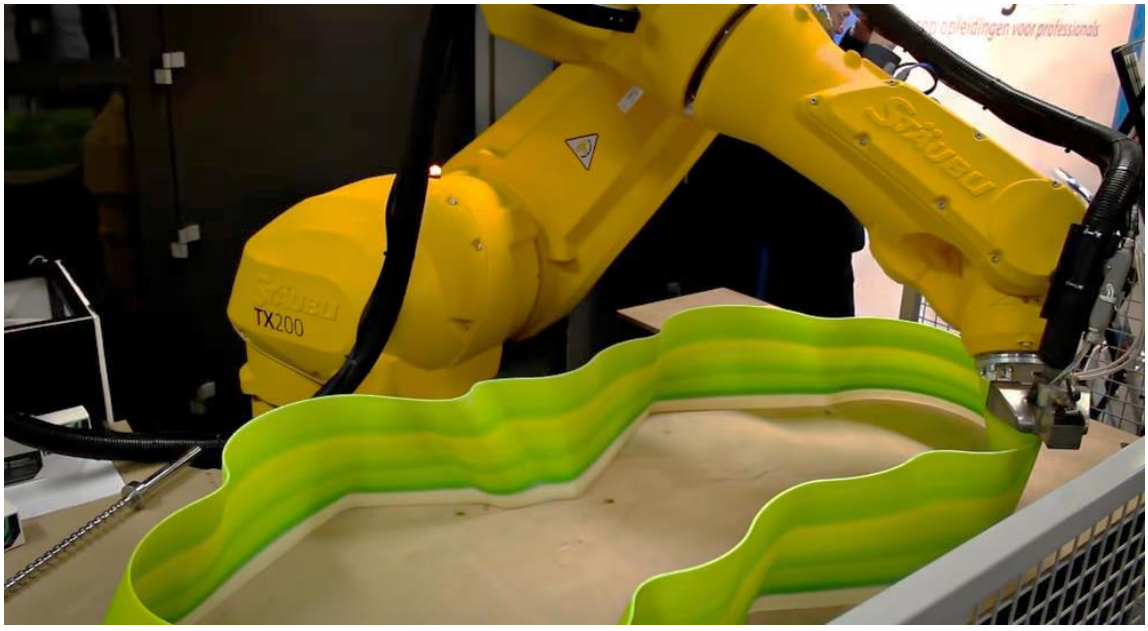


Figura 1.6. Robot Stäubli imprimiendo en 3D con plástico.

Existen ejemplos de robots de múltiples empresas instalados en configuración que les permiten imprimir en 3D con una gran diversidad de materiales, una de estas empresas es ABB. Es multinacional sueca que se dedica principalmente a la automatización y robótica siendo una de las empresas con mayor presencia en el sector de los brazos robóticos. En la **conferencia Automate 2022**, ABB exhibió un sistema de impresión 3D que utilizaba un IRB4400 equipado con un extrusor Massive Dimension MDPE10, este sistema utilizaba el software RobotStudio junto con la extensión Printing PowerPac [11].

Un caso particular de una empresa que basa su negocio en desarrollar e instalar soluciones de impresión 3D con robots de manera integral es CEAD, una compañía fundada en Países Bajos en 2014 y que proporciona diferentes soluciones para la fabricación aditiva. La diferencia principal con ABB es que CEAD sí fabrica extrusores para sus brazos robóticos además del resto de equipamiento necesario como hornos para secar las piezas terminadas o habitáculos de impresión ("printing chamber" en inglés).

Flexbot es una instalación de impresión 3D desarrollada por CEAD compuesta por un brazo robótico Comau y un extrusor o cabezal CNC para la fabricación aditiva o substractiva. El control del robot se realiza a través de un control externo con Siemens Sinumerik lo que les permite programar los movimientos directamente creando trayectorias más precisas y eficientes de manera más sencilla y directa que utilizando un robot con un control propio como los ABB que necesitan un controlador IRC5 para funcionar [12].

Los extrusores que tienen disponibles varían en tamaño y especificaciones siendo el más pequeño el E25 con una altura aproximada de 1 metro y una capacidad de extrusión de 12 kg/h, el módulo más grande es el E50 que mide más de 2 metros de alto y puede extruir 84 kg de material por hora. Las boquillas de todos los extrusores pueden variar en su diámetro permitiendo la impresión con diferentes materiales y obteniendo diferentes resultados.

En cuando a los materiales con los que pueden trabajar los extrusores, existe una amplia gama de materiales termoplásticos en forma de pellets que son fundidos a la hora de la impresión. Sin embargo, actualmente no hay disponible una opción predeterminada para imprimir con arcilla o materiales cerámicos, por lo que requeriría una solución personalizada.

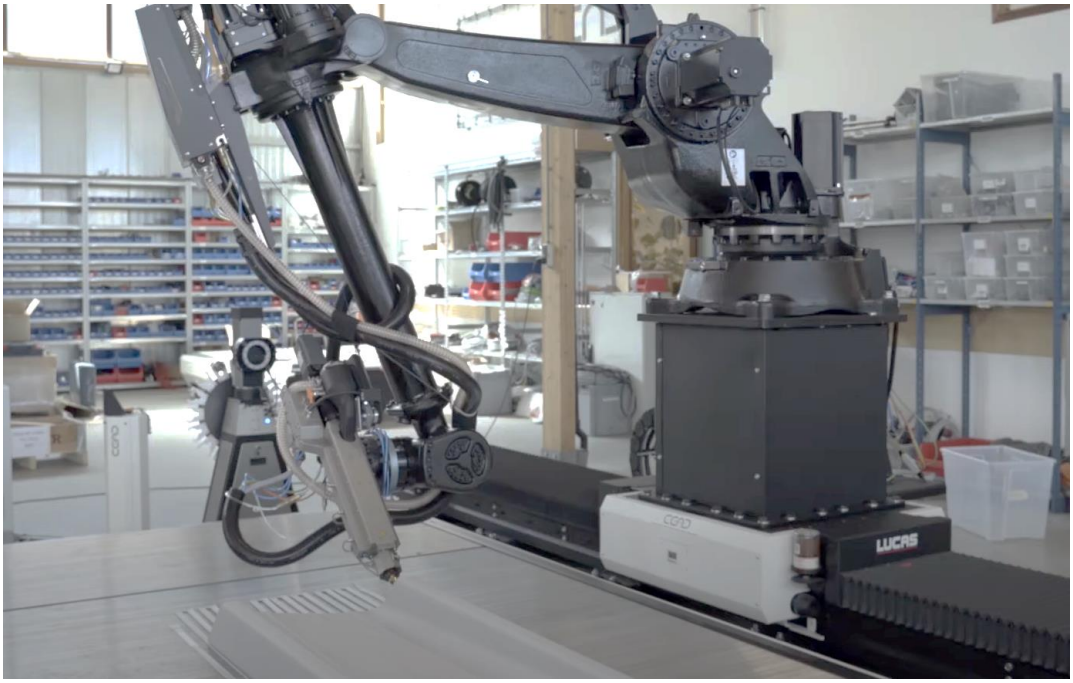


Figura 1.7. CEAD AM Flexbot Imprimiendo una pieza de gran tamaño.

1.2.3. Instalaciones similares

La impresión con brazos robóticos es un sector que todavía se encuentra en desarrollo por lo que no hay muchos ejemplos de estas instalaciones en España, en especial, si centramos la búsqueda en el uso de materiales arcillosos o cerámicos. El mejor ejemplo es el proyecto Pylos del IAAC (Institute for Advanced Architecture of Catalonia), en este proyecto usan un brazo robótico para imprimir en 3D con un material desarrollado en el IAAC. Pese a centrar el proyecto en las propiedades de los materiales y los resultados que se pueden obtener con ellos, Pylos es un claro ejemplo del potencial de la impresión con brazos tanto cuando se usan como una herramienta para la investigación como cuando se utilizan para la producción o impresión en serie [13].

Si ampliamos la búsqueda a nivel europeo también podemos encontrar algunos ejemplos de empresas o universidades que cuentan con esta infraestructura, el mejor ejemplo lo encontramos en Suiza, en el ETH Zürich (Instituto Federal Suizo de Tecnología en Zúrich).

En el **ETH Zürich,** de aquí en adelante ETH, han realizado dos proyectos de impresión 3D utilizando brazos robóticos y los extrusores de CEAD que hemos visto anteriormente, ambas investigación han sido realizadas por Gramazio Kohler Research, un grupo de investigación del ETH fundado por el Profesor Fabio Gramazio y el Profesor Matthias Kohler que junto con un gran número de ingenieros, arquitectos e investigadores buscan encontrar formas de integrar la fabricación aditiva en la arquitectura [14].

A lo largo de 2021 y 2022 han llevado varias investigaciones están explorando las posibilidades de impresión 3D de encofrados para estructuras de hormigón y de fachadas integradas, para lo que han utilizado el extrusor de CEAD que puede ser montado directamente en un brazo robótico y además tiene una alta capacidad de impresión como comentó antes [15]. Esta no es la única investigación que está llevando a cabo el ETH sobre robots en la arquitectura, también están investigando la construcción con brazos robóticos colaborativos que colocan y atornillan varios listones de madera totalmente autónomos [16].

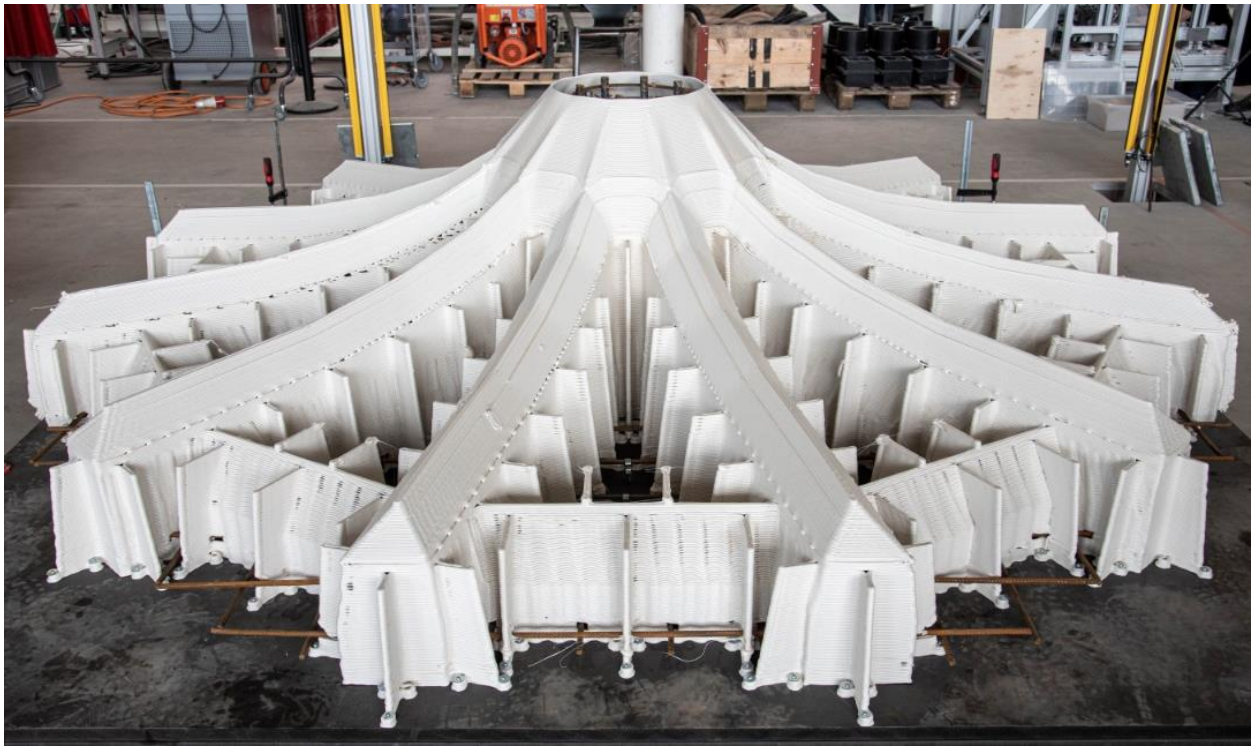


Figura 1.8. RIBB3D: Pisos de hormigón acanalados sostenibles con encofrados impresos en 3D

1.3. Solución particular

Para realizar la estación de prototipado con arcilla en el FabLab de la Universidad de Sevilla se ha escogido un brazo y un extrusor WASP XL 3.0. Los brazos robóticos de ABB son unos de los más versátiles y fiables en la industria **ABB IRB4600**, nuestro modelo es capaz de levantar hasta 60 kg y de trabajar en un amplio rango de movimientos. El brazo se encuentra montado sobre una plataforma móvil de 8m con un grado de libertad que permite al robot desplazarse en un eje. Gracias a su capacidad de alcance, se pueden crear piezas de gran tamaño sin sacrificar la precisión y la calidad del resultado final.

Por otro lado, el **extrusor WASP XL 3.0** es una buena opción para la fabricación aditiva de grandes piezas con alta capacidad de producción, gracias a su boquilla de 8 mm de ancho, además, la estación cuenta con un tanque de arcilla de 8kg que se sitúa en el eje 4 del brazo robótico. Con esta combinación de tecnologías, la estación de prototipado es capaz de realizar piezas de gran tamaño y de alta calidad de manera rápida y eficiente.

Además, la capacidad de ambos equipos junto con el software diseñado específicamente para esta aplicación y para trabajar de manera autónoma permitirá el funcionamiento continuo de la estación, lo que se traduce en una mayor productividad y una reducción de los tiempos de espera. Esto permitirá al usuario experimentar y crear prototipos en un corto período de tiempo, acelerando así el proceso de diseño y desarrollo de nuevos productos.

Figura 1.9. Estación de prototipado con ABB IRB4600 en el FablLab US.

Este equipamiento fue adquirido específicamente para realizar esta estación de prototipado **previamente** a la realización de este proyecto por lo que la programación, configuración del equipo e instalación del hardware añadido están supeditadas al equipo existente buscando la mejor solución en cada caso y minimizando posibles costes adicionales.

El equipo usado para el control de la estación lo constituyen un controlador IRC5 de ABB para el control de los movimientos del IRB4600, un Arduino Uno para el control del extrusor y un PC en el que se procesan las piezas 3D, calculan las trayectorias necesarias para imprimir el objeto y se envían al IRC5. Todos estos equipos deben estar interconectados físicamente para permitir su comunicación, esto tiene como ventaja el no depender de una conexión inalámbrica que pueda sufrir interferencias e interrumpir impresiones.

1.4. Metodología

La realización de este proyecto se ha llevado a cabo a lo largo de varios meses en los que ha pasado por diferentes etapas comenzando por una **investigación** previa sobre instalaciones similares que puedan ser de utilizada a la hora de desarrollar nuestra estación. En esta investigación nos hemos centrado en buscar información sobre el control y programación del brazo, así como la conexión entre los diferentes softwares (algunos libres y otros privados) que han sido necesarios para completar la instalación. Podemos distinguir también la **fase de diseño hardware** y montaje de los componentes necesarios ya que ha habido que diseñar el cableado de los diferentes elementos añadidos al brazo.

Este proyecto ha sido propuesto por el director del FabLab de la Universidad de Sevilla Dr. Enrique Vázquez Vicente en conjunto con el Dr. Federico Cuesta Rojo en la Escuela Técnica Superior de Ingeniería. Las bases del proyecto consistían en ampliar las capacidades del FabLab para posteriormente realizar investigaciones sobre el uso de arcillas en la arquitectura y el uso de robot para la construcción.

El proyecto venía con algunos requisitos de software que han marcado las decisiones en cuanto a la programación de este, la principal de estas es el uso del software CAD Rhinocero 7 ya que es el programa de diseño utilizado en la Facultad de Arquitectura para desarrollar los modelos 3D. Este programa cuenta con una extensión llamada Grasshopper que permite utilizar el entorno de Rhino, así como los modelos y piezas creadas por este en conjunto con un entorno de programación con bloques y Python. Otra de las restricciones existentes es que parte del hardware utilizado ya estaba adquirido a la hora de comenzar por lo que se adaptó el desarrollo del control a estos componentes.

Además del contenido que hemos obtenido de esta investigación online y que se encuentran en las referencias del proyecto y los anexos, hemos tenido acceso a diferentes documentos y libros que nos han permitido contrastar y profundizar la información, así como tener una guía sobre las diferentes técnicas existentes para el control del brazo y el procesado de piezas 3D.

- **Documentación de COMPAS**. Como se explica en el capítulo de desarrollo software, es un entorno de trabajo desarrollado por el grupo de investigación de Gramazio Kolher Research en conjunto con la universidad de Zúrich, pese a estar desarrollado con software libre su uso requiere una licencia gratuita para investigadores y universidades. De manera adicional ofrecen documentación sobre la instalación del software y programación de los controladores necesarios para utilizar este entorno de trabajo con robots ABB.
- **Advanced 3D Printing with Grasshopper**. Es un libro escrito por Diego García Cuevas y Gianluca Pugliese en el que se discuten diferentes técnicas para imprimir figuras en 3D utilizando el programa Grasshopper.
- **Documentación de ABB**. Para realizar el cableado de los elementos añadidos al brazo original se han utilizado toda la documentación relativa al brazo IRB4600, así como el soporte técnico de ABB para verificar las conexiones realizadas.

A continuación, se detalla todo el desarrollo realizado estructurado en hardware y software, en ambas partes se tratan los componentes necesarios y sus características junto con todo el trabajo necesario de cada sección. También se tratan el uso de la instalación finalizada y funcionando, así como la posible línea de desarrollo e investigación para continuar el proyecto.

2. DESCRIPCIÓN DEL HARDWARE

Analizando el alcance del proyecto, se establecen 4 grupos de equipamiento que requieren una adaptación, ya sea de manera física con instalación de partes adicionales o vía software con una programación específica, siento estos grupos, el brazo robótico **IRB 4600**, el controlador de ABB **IRC5**, el extrusor **WASP XL 3.0** y un **Arduino UNO** para la conexión del extrusor. Por esto, es necesario realizar una descripción detallada del hardware explicando las modificaciones realizadas, así como la programación necesaria únicamente para el funcionamiento de tal equipo ya que el software desarrollado para realizar la impresión será revisado en el próximo capítulo.

2.1. IRB 4600.

El brazo robótico IRB 4600 pertenece a la nueva generación ABB de alta precisión y capacidades mejoradas y añadidas. Su diseño ha sido optimizado para conseguir prestaciones superiores en aplicaciones específicas como el manejo de material, corte con láser o soldadura, lo que lo hace muy adecuado para un trabajo de impresión 3D ya que cuenta con la precisión necesaria, control de velocidad y un amplio rango de movimiento.

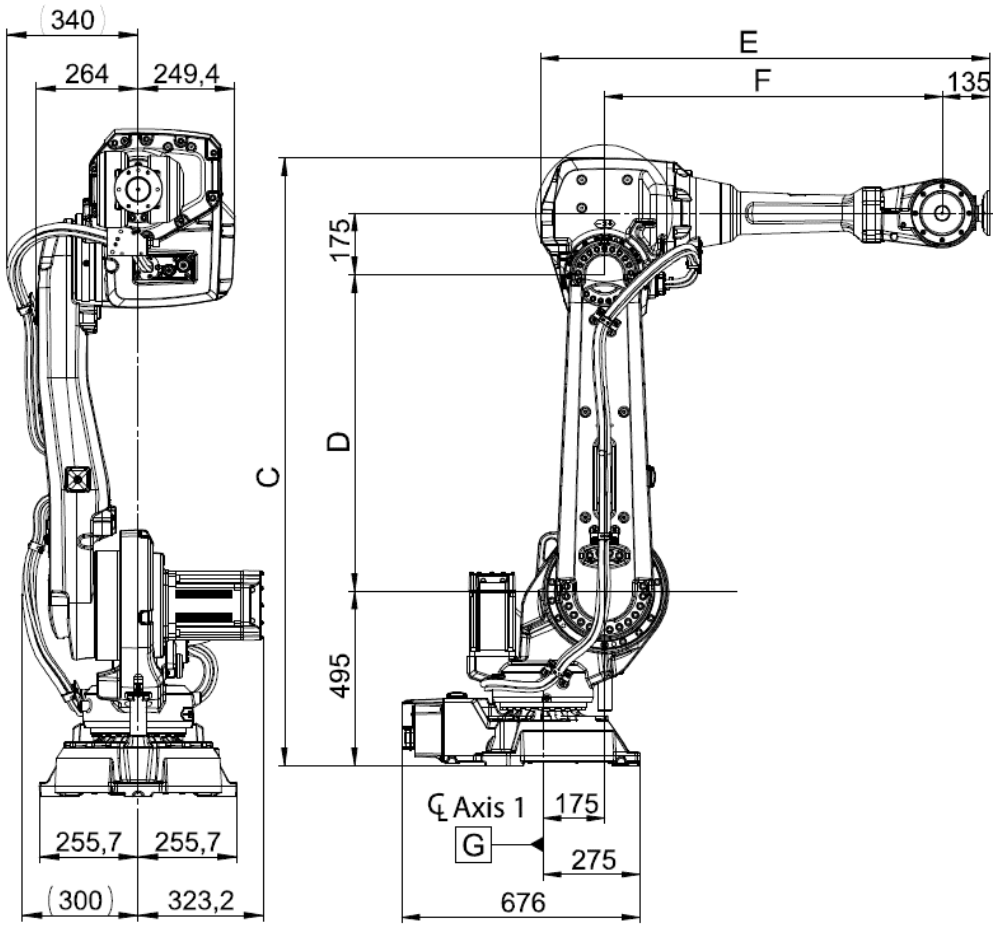


Figura 2.1 Dimensiones del IRB 4600-60/2.05. [17]

Existen varias versiones de este modelo de robot, en nuestro caso se trata de la **4600-60-2,05** (4600 con 60Kg de capacidad de manejo y alcance de 2,05m; E = 1276mm y F = 960mm).

A continuación, se muestra el diagrama de carga de esta versión.

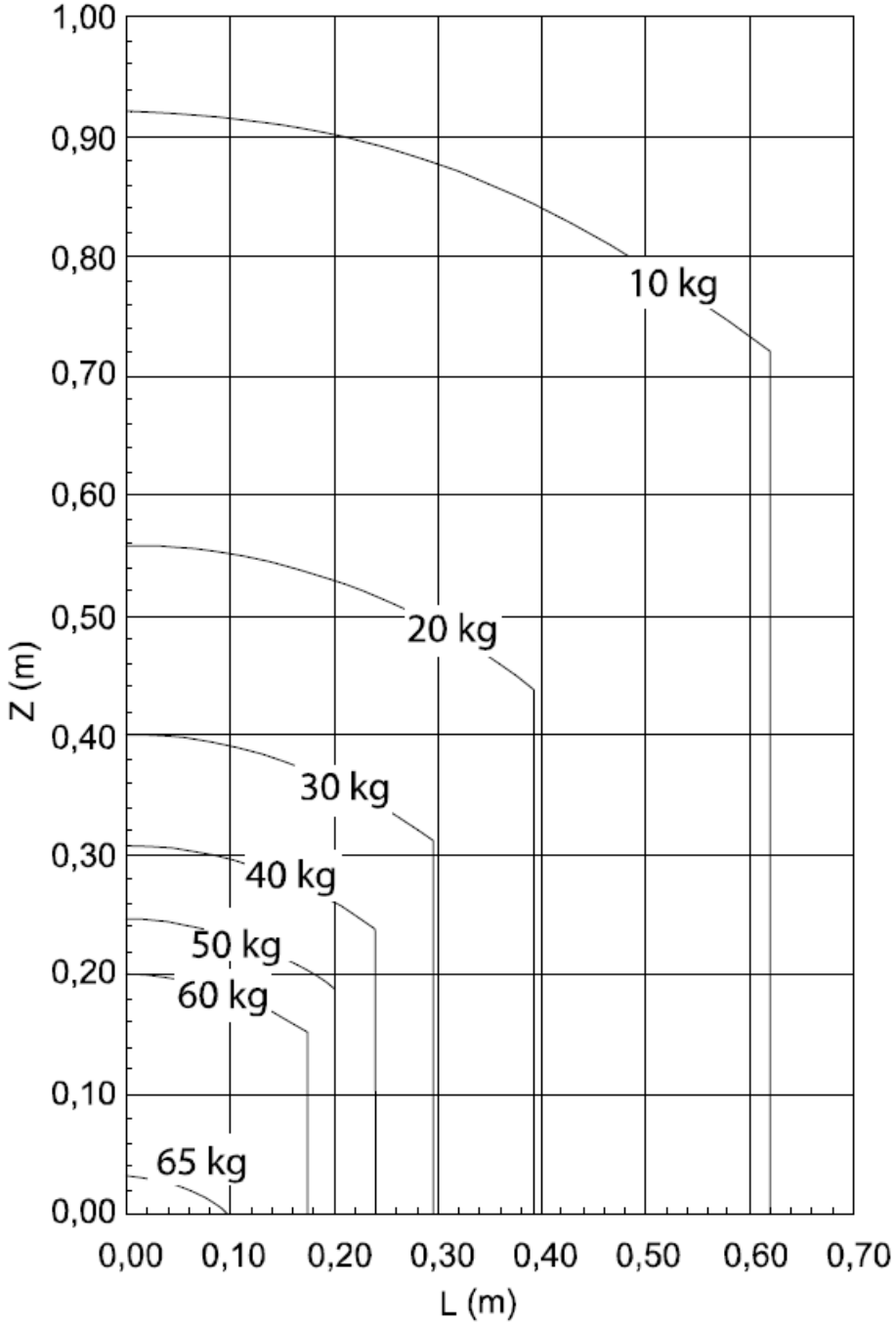


Figura 2.2. Diagrama de carga del manipulador. [17]

La forma correcta de interpretar este gráfico es la siguiente: El punto (0,0) del diagrama indica la posición de la muñeca del robot, el eje vertical Z(m) indica la desviación **del centro de masas de la herramienta (A)** utilizada en este eje de igual manera que ocurre con el eje horizontal, aunque en este caso se nombra L de longitud en lugar de utilizar el nombre del eje correspondiente ya que es la distancia en valor absoluto siguiendo esta configuración de ejes.

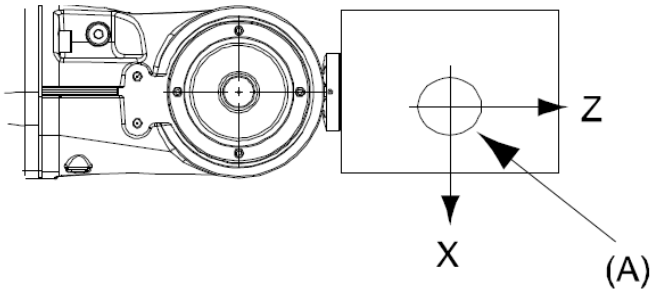


Figura 2.3 Sistema de ejes de la muñeca del manipulador. [17]

Es importante conocer el diseño de la **muñeca del robot** ya que vamos a tener que realizar una **anilla de acople** para poder instalar el extrusor WASP XL 3.0, esta anilla se describe en el apartado relativo al extrusor y aquí se citan los datos utilizados para su diseño.

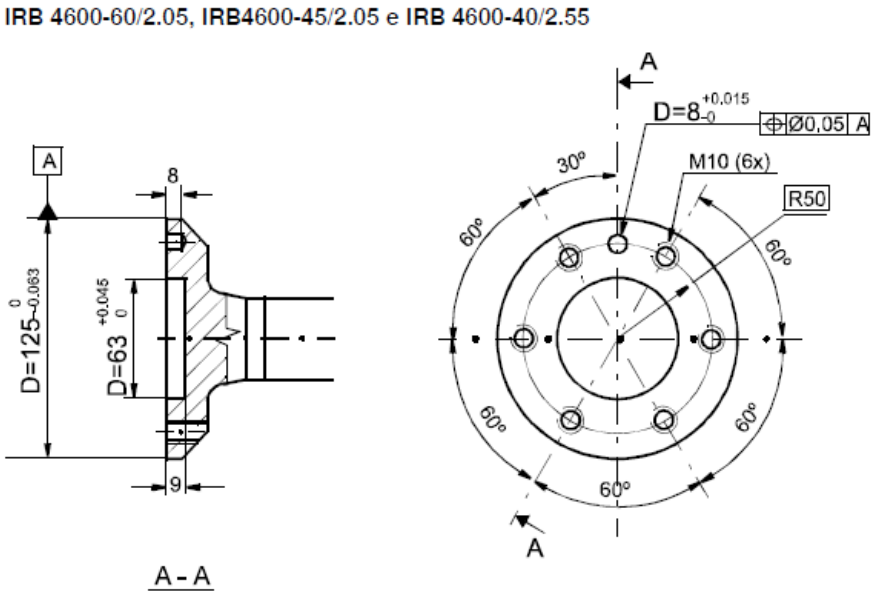


Figura 2.4 Brida de herramientas del manipulador. Esquema Acotado. [17]

El área de trabajo con un montaje sobre suelo del robot se especifica en la siguiente tabla, teniendo en cuenta que a esas medidas debemos de añadir la **posición final de la herramienta** que instalemos, en nuestro caso el extrusor WASP junto con la anilla de acople que hemos diseñado.

Tabla 1. Área de trabajo del manipulador. [17]

Variante	Pos. A	Pos. B	Pos. C	Pos. D	Pos. E	Pos. F
IRB 4600-60/2.05	2371 mm	1260 mm	1028 mm	593 mm	1701 mm	2051 mm
IRB 4600-45/2.05	2371 mm	1260 mm	1028 mm	593 mm	1701 mm	2051 mm
IRB 4600-40/2.55	2872 mm	1735 mm	1393 mm	680 mm	2202 mm	2552 mm
IRB 4600-20/2.50	2833 mm	1696 mm	1361 mm	665 mm	2163 mm	2513 mm

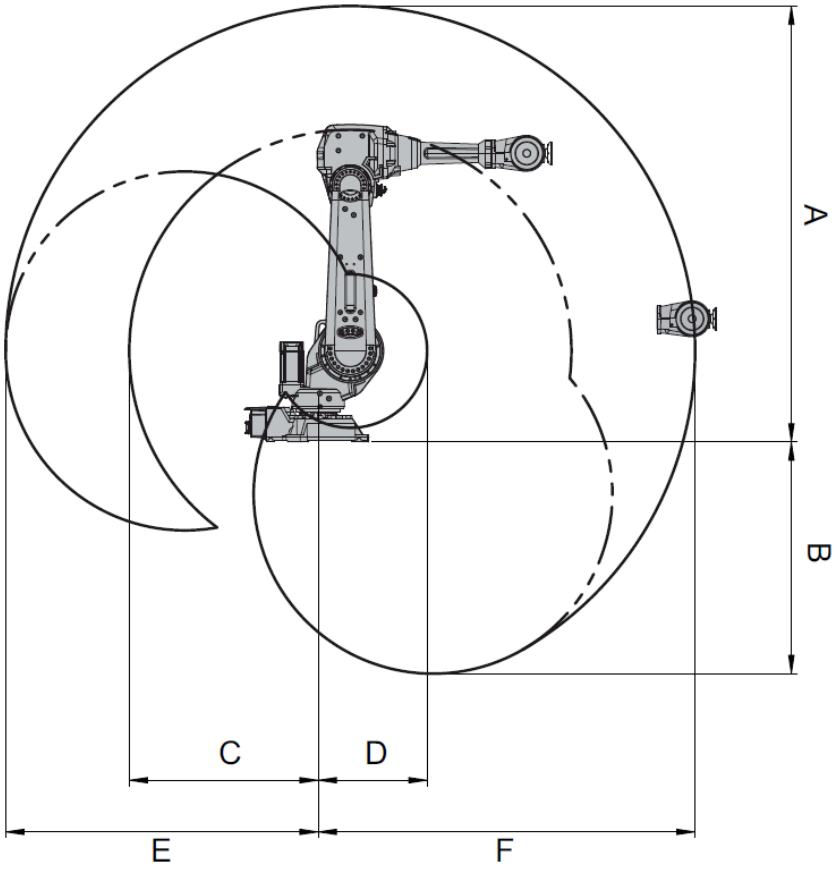


Figura 2.5 Área de trabajo del manipulador. [17]

Tabla 2 : Especificaciones relevantes del IRB 4600-60

Exactitud de posicionamiento	Media: 0,5 mm en un 98%
Aceleración máxima en movimiento controlado	35 m/s^2
Exactitud de trayectoria lineal	0.74 mm
Presión máxima de aire comprimido	8 bares
Grados de libertad del manipulador (sin TRACK)	6 GLD

2.1.1. Conexiones de usuario en el robot.

Para la conexión de equipos adicionales al robot se incorporan diferentes conectores eléctricos y de aire comprimido en la parte delantera del brazo superior. En nuestro modelo disponemos de las siguientes conexiones:

A) Aire comprimido M16x1.5.

B) **R2.CP**. Esta conexión se llama CUSTOMER POWER ya que será la encargada de proporcionar la alimentación a los equipos que instalemos en la muñeca.

C) **R2.CS**. Esta conexión se llama CUSTOMER SIGNAL ya que será la encargada de transmitir los datos necesarios para el funcionamiento del equipo.

D) R2.ETHERNET. En el caso que deseemos conectar el equipo de la muñeca a una red industrial.

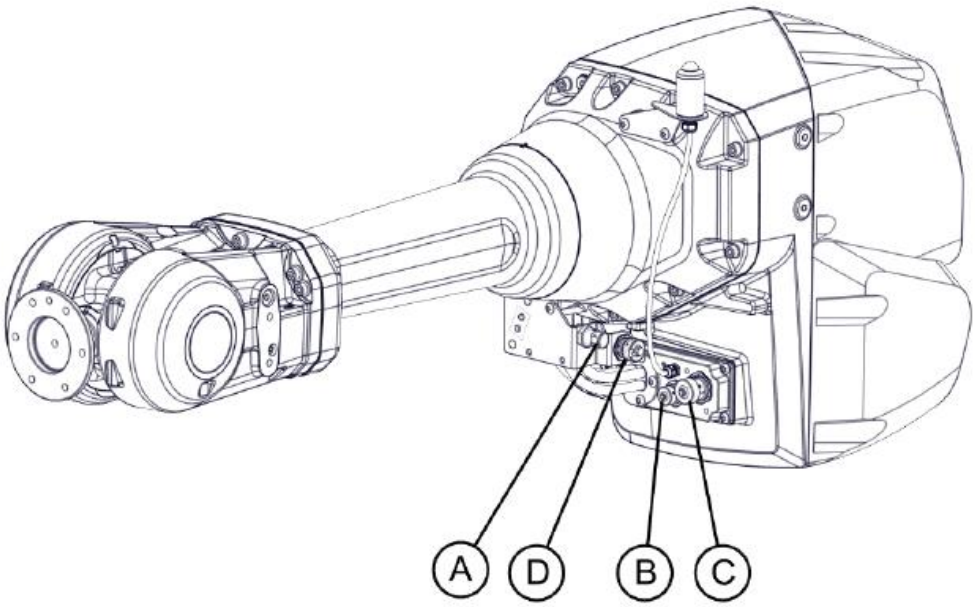


Figura 2.6 Ubicación de las conexiones de usuario en el manipulador. [17]

Estos cables están internamente conectados a la base del manipulador donde a través de la caja de conexiones R1 llegarán hasta el controlador IRC5. En nuestro caso, el brazo está montado sobre una cinta que permite mover el brazo a lo largo de su eje, constituyendo un séptimo eje y grado de libertad. La base de conexiones de usuario tiene la siguiente configuración en la instalación.

A) R1.CP/CS

B) R1.PROC1 (Aire comprimido M16.1,5)

C) R1.ETHERNET

D) R1.SMB

E) R2.FB7

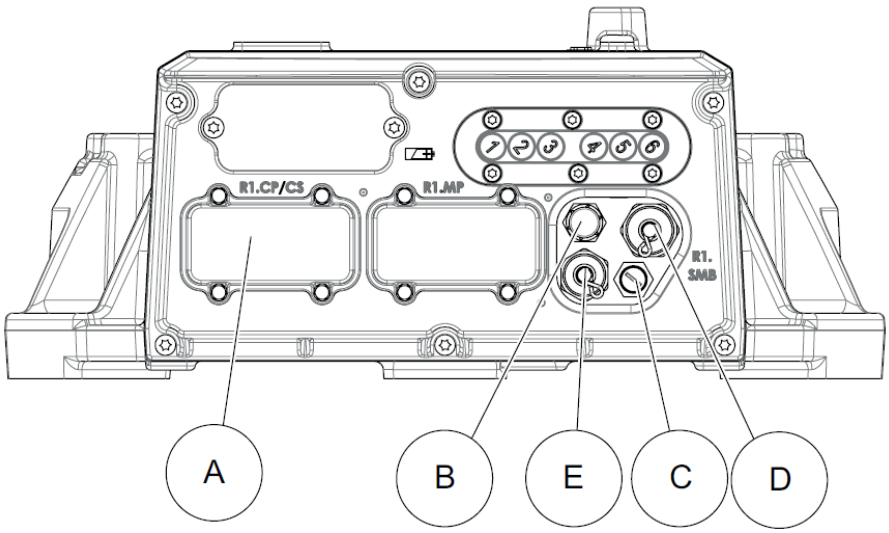


Figura 2.7 Base de conexiones de usuario con el 7º eje. [17]

Pese a que, en el **manual del brazo** [17], en la **pag74** dentro de la sección de conexiones de usuario, se detalle el número y potencia de las conexiones R2.CP y R2.CS disponibles en cada versión esto no significa que tengamos todas esas conexiones disponibles en nuestro robot ya **que cada unidad viene con un kit de conexiones (cableado interno) diferente.**

Para poder determinar que pines de conexión están disponibles debemos mirar los diagramas de circuitos de nuestra unidad y de esta manera asegurarnos de utilizar las conexiones adecuadas. Además de esto, cuando en el manual se detalla la potencia de los cables se refieren a la certificación de la que disponen los mismo, no quiere decir que estén listos para suministrar dicha potencia ya que no se encuentran conectados a ninguna alimentación dentro del controlador IRC5. La instalación que se ha realizado en nuestro caso se detalla en el apartado del extrusor.

2.1.2. TRACK IRBT 2005

El IRBT 2005 es un Track Motion lineal que, al igual que los robots ABB, se gestiona mediante el controlador IRC5. El movimiento del Track Motion puede ser controlador como un eje más desde el FlexPendant del robot o la estación de RobotStudio. En nuestro caso es un track estándar con carro para 1 robot montados de la siguiente forma:

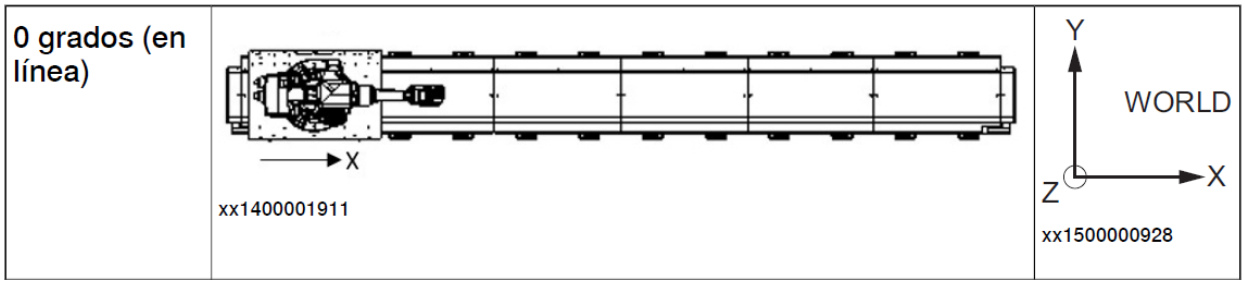


Figura 2.8 Orientación del robot con la cadena porta cables estándar. [18]

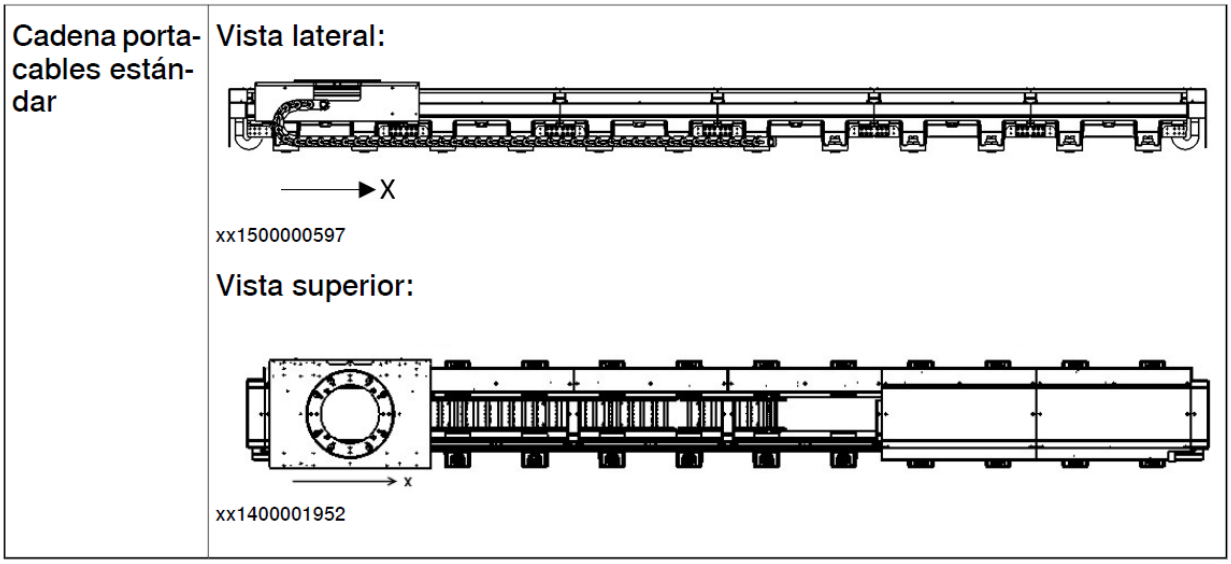


Figura 2.9 Vista lateral y superior del Track Motion. [18]

La aceleración máxima del Track es de 4m/s^2 y la velocidad máxima es de 2 m/s. El IRBT 2005 se ha diseñado para un funcionamiento intermitente. No se ha concebido para una aceleración/deceleración continua. Esta última puede dar lugar a un sobrecalentamiento del motor del track, por lo que no utilizaremos este eje como uno cualquiera del robot a la hora de alcanzar posiciones deseadas, sino únicamente para extender el rango de operación del brazo cuando la posición deseada esté fuera del alcanza actual.

2.2. IRC5 CONTROLLER

El IRC5 es el controlador estrella de ABB capaz de operar un gran número de manipuladores de la compañía. Además del control de movimientos, aporta flexibilidad, seguridad, modularidad, interfaces de aplicación, control multirobot y un puerto de conexión para un PC de servicio.

Es este proyecto vamos a hacer uso de todas estas cualidades que nos van a permitir programar con un software alternativo el control del robot gracias a que el IRC5 hará de puente entre el PC de servicio y el manipulador, también se le instalarán los complementos de hardware que sean necesarios para controlar el extrusor permitiendo continuar el proyecto con un controlador único que integre todas las funciones requeridas. Esto no es solo más práctico para el mantenimiento y futuras modificaciones del mismo, sino que engloba todos los sistemas empleados bajo los sistemas de seguridad que incorpora evitando posibles riesgos.

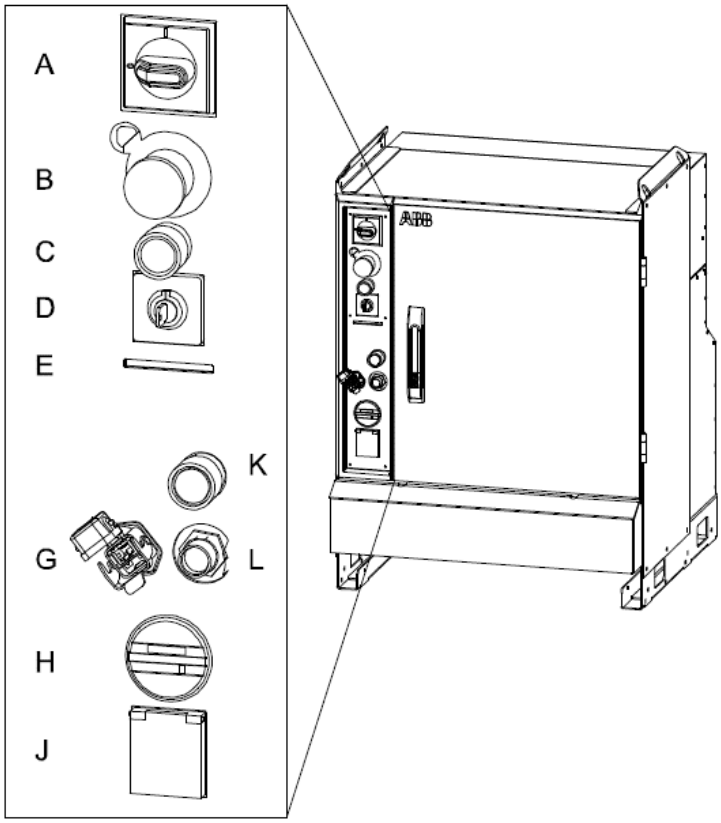


Figura 2.10 IRC5 Controller. [19]

El panel de control externo sirve para establecer el funcionamiento del controlador y del manipulador permitiendo alternar con seguridad entre los distintos modos de operación.

A) Interruptor de encendido
B) Paro de emergencia
C) Botón de motores ON
D) Selector de modo
G) Conector de Ethernet para PC de servicio
L) Conector para FlexPendant

En el siguiente apartado se describirán los diagramas de circuitos del IRC5 utilizados para instalar las conexiones entre el extrusor y el controlador además de los elementos añadidos a la estructura como el Arduino o la fuente de alimentación del extrusor.

2.3. EXTRUSOR Y TANQUE DE ARCILLA

En la estación de impresión vamos a utilizar el extrusor WASP 3.0 XL, 3DWASP [20] es una empresa líder en el campo de la impresión 3D con arcilla que ha destacado por su compromiso con la innovación y la calidad en el desarrollo de sus productos. Su enfoque principal radica en proporcionar soluciones avanzadas de impresión 3D para la creación de objetos utilizando este material específico tanto para particulares como sectores profesionales brindando en ambos casos soporte al cliente.

El extrusor WASP 3.0 XL es uno de sus productos destacados, este extrusor se caracteriza por su diseño robusto y su capacidad para manejar grandes volúmenes de arcilla. Con una boquilla de alta precisión y un sistema de extrusión controlado, el extrusor WASP 3.0 XL permite la impresión de objetos de gran tamaño con una alta calidad de acabado y alta velocidad.

Tabla 3 Especificaciones del extrusor WASP 3.0 XL

Printing technology:	Liquid Deposition Modeling (LDM)
Material maximum granulometry:	1 mm
Compatible nozzle dimensions:	4 mm, 6 mm, 8 mm
Energy consumption:	40 W
Suggested stepper voltage:	24 V
Cooling fan voltage:	12 V
Connector:	MODU
Extruder components materials:	Aluminum, POM
Net extruder weight:	1.3 Kg
Net tank weight:	4.6 Kg
Tank Capacity	5 L

En el manual del producto se especifica que no se proporcionará soporte con la instalación del equipo con el sistema del cliente por lo que tenemos que integrarlo nosotros, desarrollando los acoples necesarios para montarlo en el brazo y los cables de conexión.

El motor que utiliza el extrusor es un motor paso a paso de 24v (con una corriente máxima de 2A) que al no estar conectado a una placa base de impresora tenemos que conectarlo a un **Stepper driver** para poder controlar la velocidad de giro. Además, requiere de un Arduino Uno ya que el controlador IRC5 no es capaz de proporcionar una salida PWM, este proceso se explica en el siguiente apartado.

El tanque de arcilla va a ser instalado en el brazo de la **articulación 4** del robot, siendo este el mejor lugar para su ubicación debido a la facilidad del montaje en esta zona al tratarse de un brazo recto sin obstáculos y que en la mayoría de los movimientos no supone un riesgo de colisión con otras partes del brazo. Su montaje se va a realizar con dos abracaderas personalizadas en cada extremo, que asegurarán un montaje firme, dejando la boquilla de conexión con la manguera del extrusor en la parte más cercana a este.

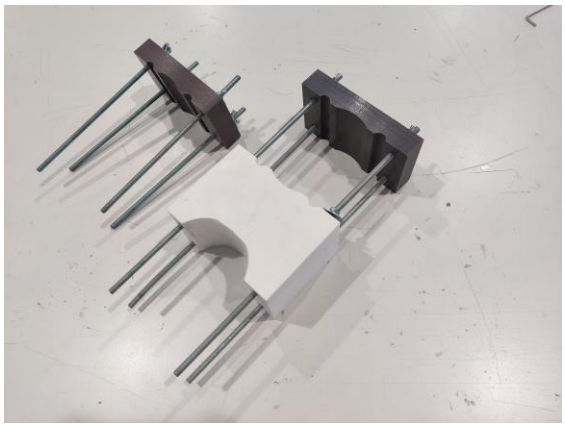


Figura 2.11 Abrazadera superior

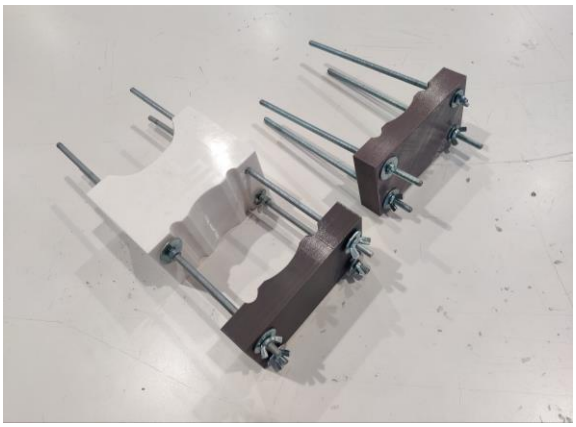


Figura 2.12 Abrazadera Inferior

El taque requiere de 8 bares de presión para funcionar correctamente que se le serán suministrados desde la salida A de la parte superior del brazo que se muestran en la Figura 2.6. A su vez esta toma es una prolongación de la entrada para aire comprimido que se encuentra en la base del manipulador, concretamente la R1.PROC1 (B) que se muestra en la Figura 2.7, aquí conectaremos un compresor externo que sea capaz de suministrar la presión requerida por nuestro sistema.

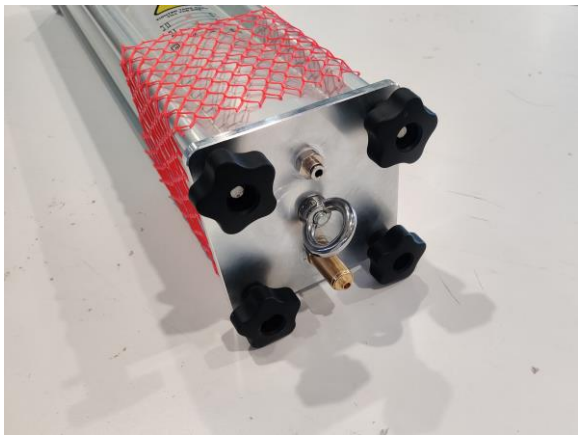


Figura 2.13 Boquilla superior (entrada de aire comprimido)

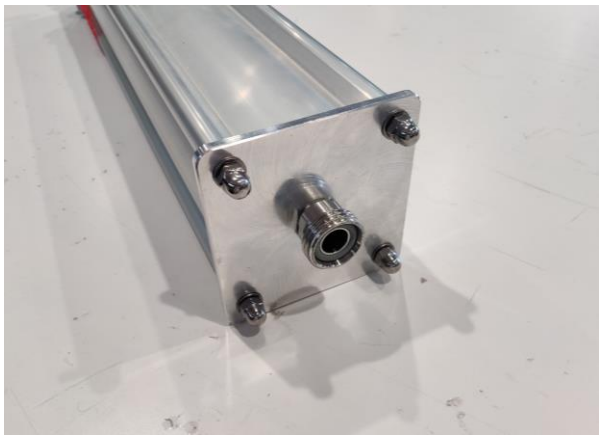


Figura 2.14 Boquilla inferior (Salida de material)

2.3.1. Anilla de acople del extrusor.

Para instalar el extrusor en la articulación 6 del manipulador debemos crear una anilla de acople que permita ser atornillada al brazo y que el extrusor quede en posición vertical, además de esto, el extrusor debe quedar a una altura tal que su boquilla esté por debajo de la parte más baja del conjunto de articulaciones de la muñeca (Figura 2.6) para que a la hora de imprimir estas no puedan colisionar durante su movimiento con el objeto imprimido. Si consideramos como origen de coordenadas el centro de la anilla como se muestra en esta Figura 2.4, la boquilla del extrusor debe quedar 16 cm por debajo del origen.

A continuación, se muestran los diseños en perspectiva, planta, alzado y perfil de la anilla y posteriormente la pieza impresa e instalada en el robot.

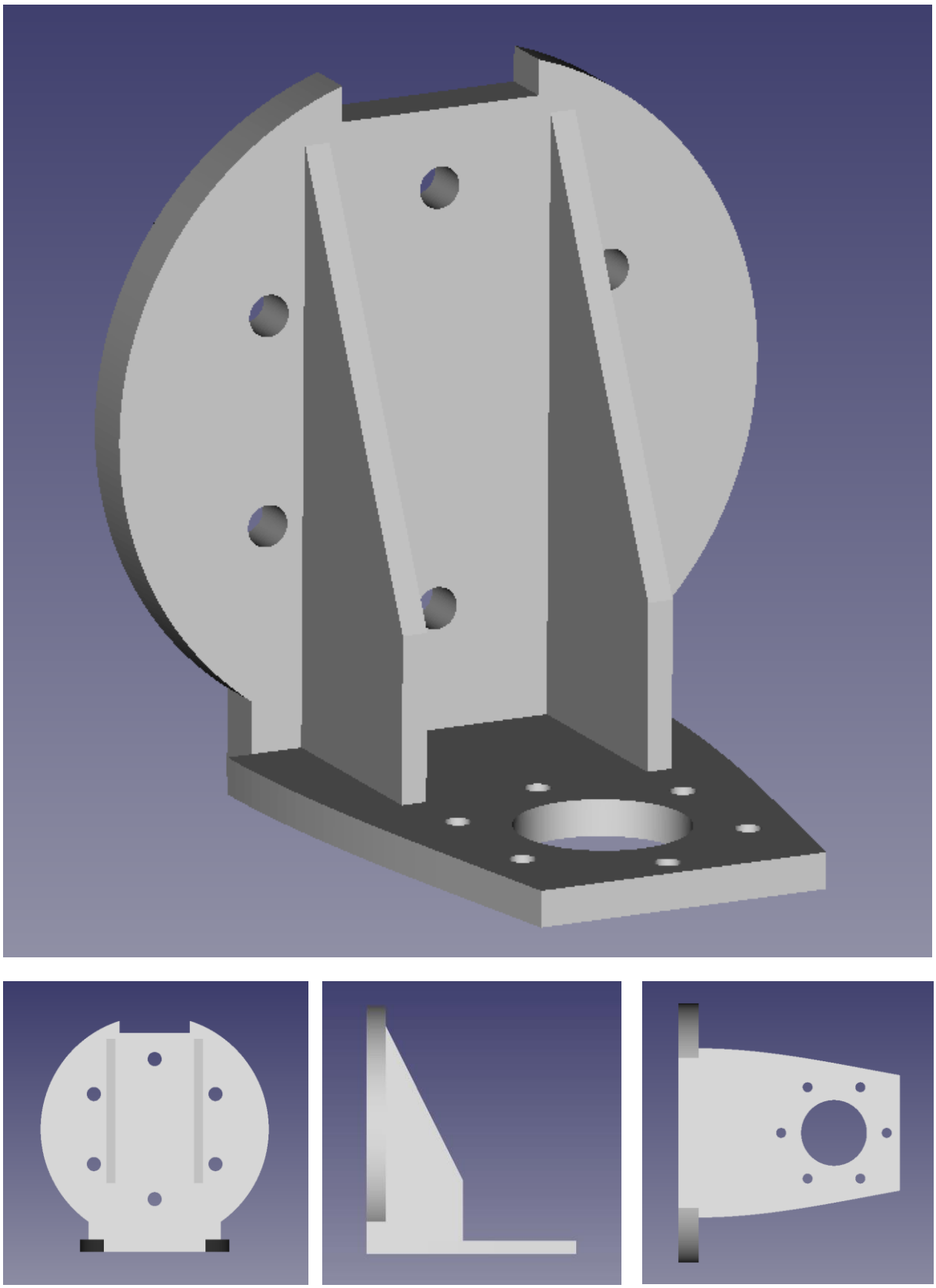


Figura 2.15 Diseños de la anilla de acople.

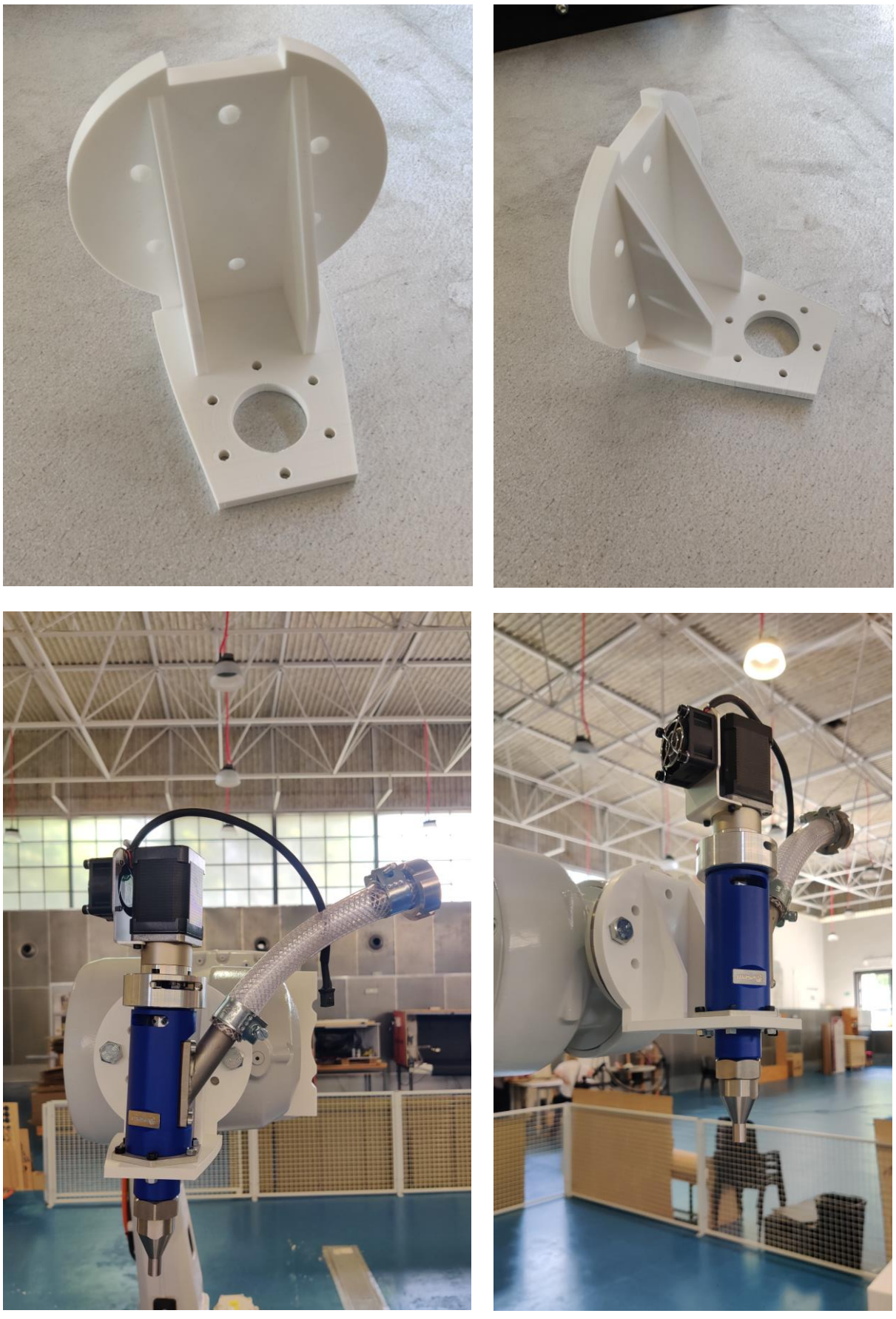


Figura 2.16 Imágenes del extrusor y la anilla de acople instalada

2.4. CONEXIÓN DEL EXTRUSOR WASPXL CON IRC5

La complejidad de la conexión entre el extrusor y el manipulador radica en la arquitectura seguida por cada uno de los fabricantes, teniendo la primera un enfoque electrónico con conectores de pequeño tamaño utilizados de manera habitual en proyectos que involucran microcontroladores y la segunda, un enfoque industrial, con conectores robustos y conexiones segmentadas.

El extrusor tiene un cable de conexión MODU de 8 pines que sigue el siguiente esquema:

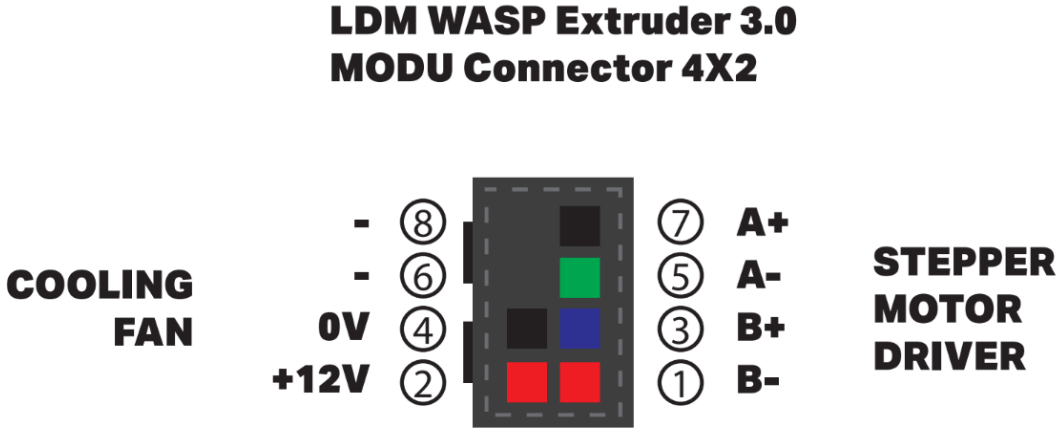


Figura 2.17 Conector MODU del extrusor. Pinout.

A la izquierda encontramos la alimentación para el ventilador incorporado en el extrusor y que requiere de 12v para funcionar. A la derecha se encuentran los 4 pines de conexión al motor paso a paso, utilizando estos pines debemos conectar el motor a un controlador y un Arduino Uno para poder ajustar la velocidad y hacerlo funcionar correctamente.

En el lado del manipulador vamos a utilizar la conexión R2CS de la parte superior del manipulador (Figura 2.6 Ubicación de las conexiones de usuario en el manipulador. Figura 2.6 conexión C). El conector R2CS que dispone nuestro brazo tiene 6 pines instalados, justo los que necesitamos para el extrusor. El número de pines y la disposición de estos varían entre cada unidad y es necesario seguir los esquemas eléctricos del manual (disponibles en MyABB) para ver que pines tenemos disponibles.

El cable que une estos dos componentes ha sido fabricado específicamente para esta instalación siguiendo este esquema de conexión:

CONECTOR MODU:		:CONECTOR R2 ABB
	VENTILADOR 12V	A
	VENTILADOR 0V	R
	A+	P
	A-	N
	B+	M
	B-	Z

Para seleccionar los pines del conector R2CS que se han utilizado para cada pin del conector MODU se han seguido los diagramas de circuitos del brazo hasta el controlador IRC5 donde los cables procedentes del manipulador se dividen en diferentes tarjetas de entrada pudiendo algunas proveer señales y potencia o

únicamente potencia.

2.4.1. Conexión eléctrica

Una vez tenemos conectado el extrusor al manipulador debemos realizar el cableado necesario en el controlador IRC5 para que cada pin del conector MODU del extrusor reciba la potencia y señal que necesita para su funcionamiento. Lo primero que debemos saber es en qué tarjeta de entradas del controlador se ubican los cables que hemos utilizado en la toma R2.CS.

Los diagramas de circuitos utilizados son confidenciales ya que se encuentran dentro de los esquemas eléctricos disponibles para los propietarios de robots ABB y de ellos podemos extraer que el conector **R2.CS** (Figura 2.6) de la parte superior del brazo es una prolongación del conector **R1.CP/CS** (Figura 2.7) en la base del robot. A partir de aquí los cables se dirigen al controlador IRC5 y se ubican en dos tarjetas de entrada:

A) **XP/XT5.1**, donde llegan a sus entradas 1 y 2 los pines A y R del R2.CS

B) **XP/XT5.2**, donde llegan a sus entradas 1,2,3 y 4 los pines P, N, M y Z del R2.CS

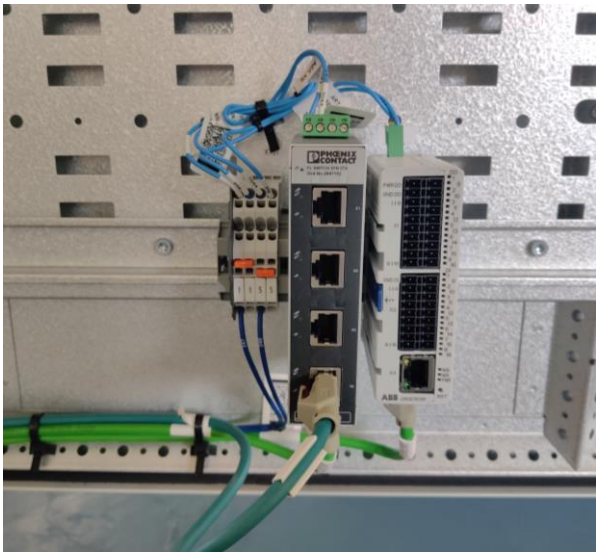


Figura 2.18 Tarjeta XT35 (a la izquierda)

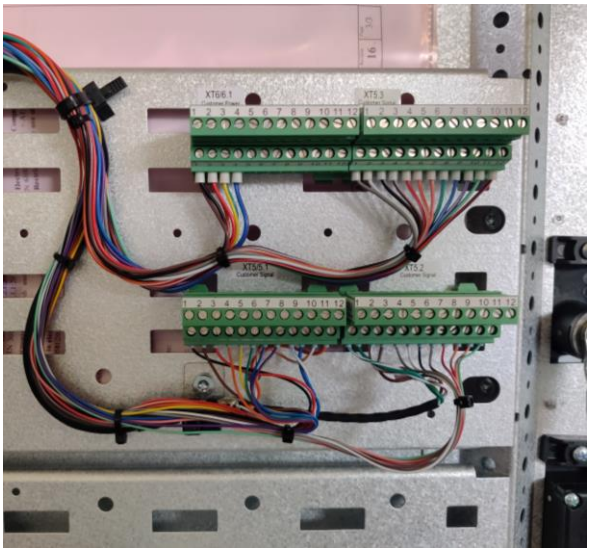


Figura 2.19 Tarjetas XT5.1 y XT5.2

Los pines A y R son los encargados de proporcionar la alimentación al ventilador del extrusor que necesita 12v para funcionar, como el controlador IRC5 no es capaz de proporcionar 12v **necesitamos instalar una fuente de alimentación externa** que irá montada en la puerta del controlador. Esta fuente tiene que estar conectada con los pines de 12v del conector MODU y para ello se utilizará la tarjeta de conexiones de potencia de usuario **XT5.1** disponible en la puerta del controlador.

Como se explicó anteriormente es necesario un Arduino Uno y un Stepper Driver para controlar el movimiento del extrusor ya que el controlador IRC5 no tiene salidas PWM. Para evitarnos instalar una fuente de 5v externa vamos a alimentar al Arduino con la fuente externa de 12v mediante el puerto de alimentación estándar (no USB) que dispone.

2.4.2. Stepper driver

La necesidad de un controlador en la operación de un motor, en general, radica en la gestión eficiente de la energía y el control preciso del movimiento. Los motores, especialmente aquellos como los motores paso a paso,

requieren corrientes y secuencias de pulsos específicas para funcionar de manera óptima. Un controlador actúa como el intermediario que traduce las señales eléctricas del sistema de control (como un microcontrolador o un PLC) en las acciones mecánicas del motor. En nuestro caso y para no complicar la instalación usaremos un Arduino en vez de un PLC como controlador, aunque ambos serían válidos.

En el caso del controlador específico para motores paso a paso, como el **A4988**, este se convierte en una pieza clave para alcanzar la máxima eficiencia y precisión. El A4988 regula la corriente que fluye a través de las bobinas del motor, evitando sobrecalentamiento y garantizando un rendimiento seguro y sostenible.

La incorporación de un controlador como el A4988 no solo asegura una operación segura y eficiente del motor, sino que también facilita la implementación de sistemas de control más complejos, mejorando la calidad y exactitud de los movimientos en aplicaciones que requieren niveles precisos de posicionamiento, como la impresión 3D.

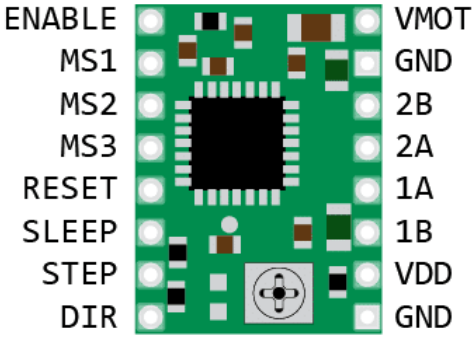


Figura 2.20 A4988 Pinout

Los pines VMOT y GND serán donde conectemos la alimentación del motor que debe ser de 24v, esta vez sí disponemos de una toma de 24v directamente en el IRC5 por lo que no será necesario una fuente adicional. Esta toma se encuentra en la tarjeta **XT35** situada en la parte inferior de la puerta del controlador.

Los pines 1ª, 1B, 2ª y 2B deben ser conectados a los pines del mismo nombre del conector MODU del extrusor, para eso se utiliza la tarjeta de conexiones de señales de usuario **XT5.2** disponible en el IRC5 y situada en la puerta del controlador.

Respecto a la conexión entre el driver **A4988 y el Arduino Uno**:

- "DIR" y "STEP" del driver se conectarán a pines digitales del Arduino (por ejemplo, pines 2 y 3). "DIR" determina la dirección del motor, y "STEP" indica un paso del motor.
- "GND" del driver debe estar conectado al GND del Arduino.
- "ENABLE" en el driver puede dejarse conectado a GND si no se desea usar la función de habilitación del motor mediante software. En caso afirmativo puede conectarse a cualquier pin digital del Arduino.

De esta manera tenemos interconectado todos los componentes que hacen posible la extrusión de material, quedando de esta manera el esquema de conexiones referenciado a las entradas de las tarjetas del IRC5 utilizadas.

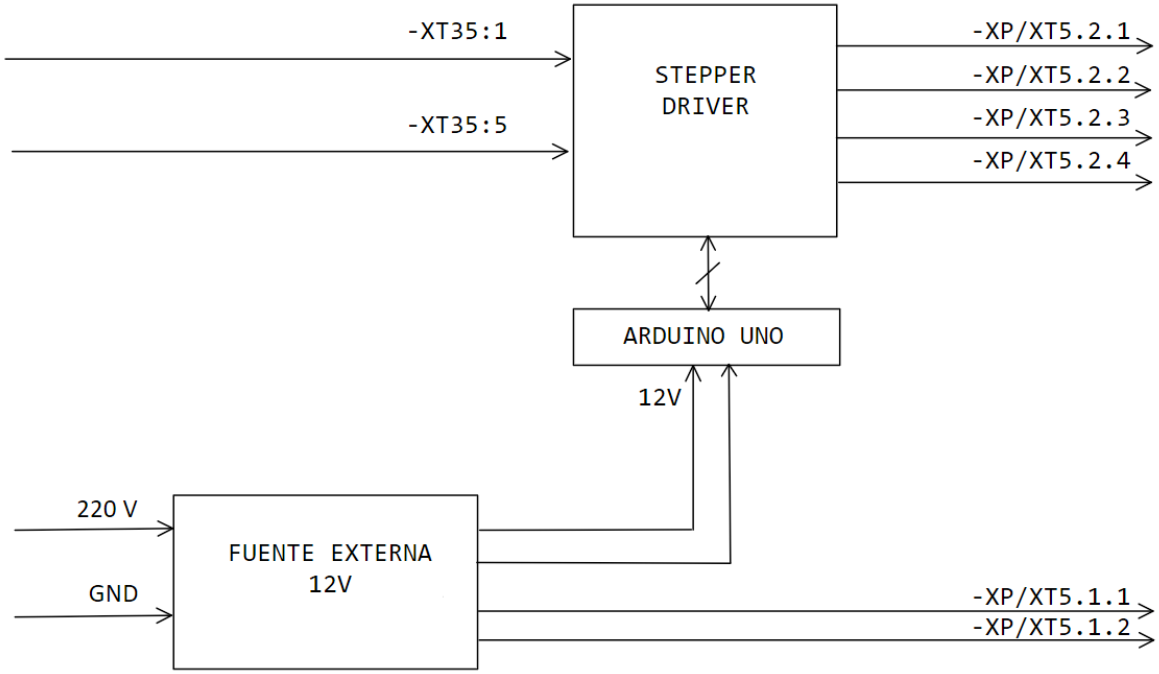


Figura 2.21 Esquema eléctrico para el control del motor paso a paso referenciado a las tomas del IRC5

Este esquema recoge las conexiones que realizarán dentro del IRC5, además, debemos conectar el cable desarrollado anteriormente para finalizar la instalación. En este caso, se ha escogido una fuente de **alimentación externa 220v/12v** pero también sería una opción adecuada utilizar una **fuente 24v/12v**, así podemos conectarla a las tomas XT35 donde conectamos el Stepper Driver y evitar una toma de corriente fuera del sistema IRC5

2.4.3. Arduino Uno

La función que cumple el microcontrolador Arduino es generar las señales necesarias para que el controlador del motor paso a paso pueda funcionar correctamente, además es el encargado de controlar la velocidad del motor, así como su activación. Existen múltiples formas de controlar la velocidad a través del Arduino y la más sencilla consiste en utilizar un potenciómetro que regule esta variable. Pese a ser una solución temporal que requiere de un ajuste manual a la hora de experimentar con el sistema completo y en funcionamiento es más sencillo modificar los valores de extrusión de material hasta encontrar una configuración adecuada para el material que se esté utilizando es ese momento o la velocidad de impresión deseada de la pieza completa.

Una vez se haya experimentado con el sistema se pueden fijar uno o varios valores de velocidad y asignarlos a una entrada digital del Arduino de modo que cuando se active una entrada el configure automáticamente la velocidad programada de esta manera se pueden tener varios perfiles de velocidad, uno para cada material, por ejemplo. Estas entradas digitales serán conectadas entonces a un puerto I/O del controlador IRC5 para que sea capaz de seleccionar la velocidad de extrusión.

De esta manera se solucionan los problemas de sincronización entre el controlador y el Arduino y el segundo pasará a ser un elemento secundario de la instalación que no debe requerir programación adicional o ajustes.

A continuación, se muestra el código utilizado para configurar el Arduino con un potenciómetro.

```
//Definimos las variables
const int stepPin = 6;
const int dirPin = 5;
int customDelay, customDelayMapped;

void setup() {
  // Sets the two pins as Outputs
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
  digitalWrite(dirPin,HIGH); //Sentido de giro horario → HIGH, antihorario → LOW
}

void loop() {
  customDelayMapped = speedUp(); // Obtiene el valor de espera a través de la función de lectura del potenciometro
  // Generamos los pulsos para el motor a la velocidad selecionada
  digitalWrite(stepPin, HIGH);
  delayMicroseconds(customDelayMapped);
  digitalWrite(stepPin, LOW);
  delayMicroseconds(customDelayMapped);
}

// Funcion para controlar el potenciometro
int speedUp() {
  int customDelay = analogRead(A0); // Leemos el valor del potenciometro
  int newCustom = map(customDelay, 0, 1023, 400,4000);
// convertimos los valores del CAD (de 0 a 1023) en un rango que nos sea de utilidad por ejemplo 400 a 4000
  return newCustom;
}
```

3. DESARROLLO SOFTWARE

En este capítulo se aborda todo el software necesario para el funcionamiento del manipulador como un brazo de impresión con arcilla. Para llevar a cabo esta adaptación ha sido necesario seleccionar un software 3D que reúna todas las características que el equipo de FabLab puede necesitar a la hora de diseñar y fabricar piezas, este requisito, al ser una de las restricciones más importante del proyecto, modifica el camino a seguir y el resto del software necesario que será el que haga posible la integración.

El programa CAD seleccionado es **Rhino 7**, un programa bajo licencia que permite crear modelos 3D con gran precisión, una de sus ventajas radica en su versatilidad ya que a diferencia de otros softwares CAD más especializados puede utilizarse en diferentes sectores con herramientas de programación integradas.

Una de las herramientas de programación que está integrada en Rhino 7 es **Grasshopper**, una herramienta de programación por bloques basada en Python que permite dar un gran salto en el diseño paramétrico y la modelización generativa. Esta potente herramienta de programación visual permite realizar escenarios completos, en arquitectura, por ejemplo, puede generar una maqueta de una ciudad a criterio del programador o generar estructuras complejas ya que permite el uso de funciones matemáticas aplicándolas al diseño.

Estas razones fundamentan la elección de este software para diseñar la plataforma que permita transformar diseños 3d en algo que pueda entender y replicar nuestro manipulador. El uso de este software complica el resto del diseño ya que ni ABB y Rhino proporcionan ningún tipo de integración, sino que además utilizan estructuras de programación completamente diferentes.

Durante la fase de investigación previa a este proyecto, se identificaron algunos proyectos realizados por ingenieros de la Universidad de Zúrich, quienes han desarrollado una plataforma o, más precisamente, un conjunto de bibliotecas para Grasshopper que permiten la transmisión de datos a los robots de ABB. Estas librerías necesitan de software adicional para establecer la comunicación entre ambos equipos, pero son la opción más viable que se ha encontrado. El uso académico del software se permite de manera gratuita bajo una autorización previa que ya ha sido otorgada a FabLab.

3.1. DISEÑO DE LA ESTACIÓN EN ROBOTSTUDIO

Para operar con cualquier robot de ABB es necesario utilizar el programa RobotStudio, software bajo licencia de ABB. Los clientes que posean un brazo tendrán a su disposición diferentes tipos de licencias que permitirán acceder a todas o algunas de las funciones disponible. En caso de FabLab dispone de la licencia estándar.

Cuando programemos en robot de ABB deberemos hacerlo desde una estación de RobotStudio, una estación es una versión virtual de la instalación sobre la que trabajamos representando fielmente las dimensiones de los equipos y la física de estos.

El proceso de creación de una estación es sencillo y no se va a detallar como se realiza paso a paso, en cambio, se muestra la estación que creada y la configuración de los equipos.

3.1.1. Elementos básicos

Los elementos principales de la instalación son el manipulador IRB4600 y la TRACK 2005 de 6 metros de longitud. Ambos se encuentran en integrados en RobotStudio dentro de la Biblioteca de ABB. Una vez añadidos debemos posicionar el robot encima del Track arrastrándolo y enlazándolo. Además de estos equipos vamos a añadir una mesa de trabajo, que necesitaremos para ajustar la herramienta y a partir de su superficie superior se

tomará el sistema de referencia para los puntos que enviemos al robot.

Para crear la mesa utilizaremos la pestaña de modelado desde donde añadimos un sólido con la forma deseada, las medidas en este caso no son las definitivas y pueden ser modificadas en un futuro cuando se decida la superficie real sobre la que imprimirá. Hay que tener en cuenta que RobotStudio toma como Z=0 la base del robot y no la base del track por lo que debemos tener en cuenta el offset a la hora de dimensionar la mesa virtual respecto a la real.

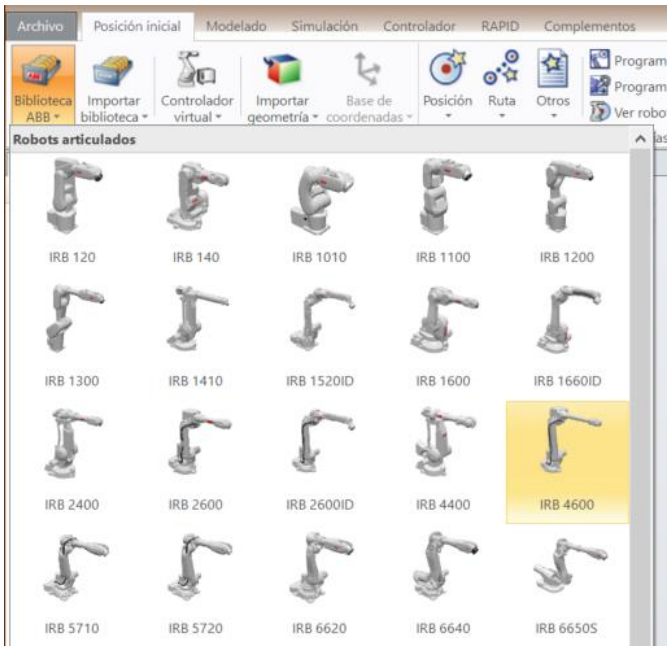


Figura 3.1 RobotStudio. IRB4600 Biblioteca.

Figura 3.2 RobotStudio. IRBT2005 Biblioteca.

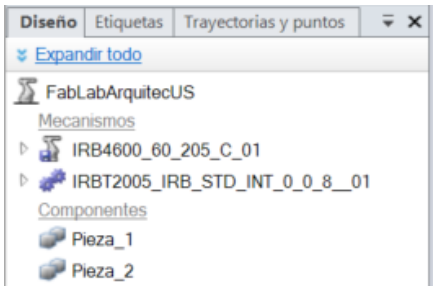


Figura 3.3 RobotStudio. Elementos estación inicial.

Estos elementos crean la estación inicial sobre la cual vamos a añadir la herramienta de trabajo y el tanque de arcilla. La mayoría de los ajustes de los equipos utilizados desde las librerías vienen configurados correctamente, pero los sólidos que generamos nosotros pueden tener comportamientos diferentes, por ejemplo, pueden computarse en la **física de colisiones** haciendo que el software los tenga en cuenta a la hora de generar las trayectorias. En todos los objetos que vamos a generar vamos a configurarlos para que se tengan en cuenta en la planificación de trayectorias. Esto se configura haciendo clic derecho sobre la pieza en la pestaña Diseño de la ventana vertical que se sitúa a la derecha de la pantalla.

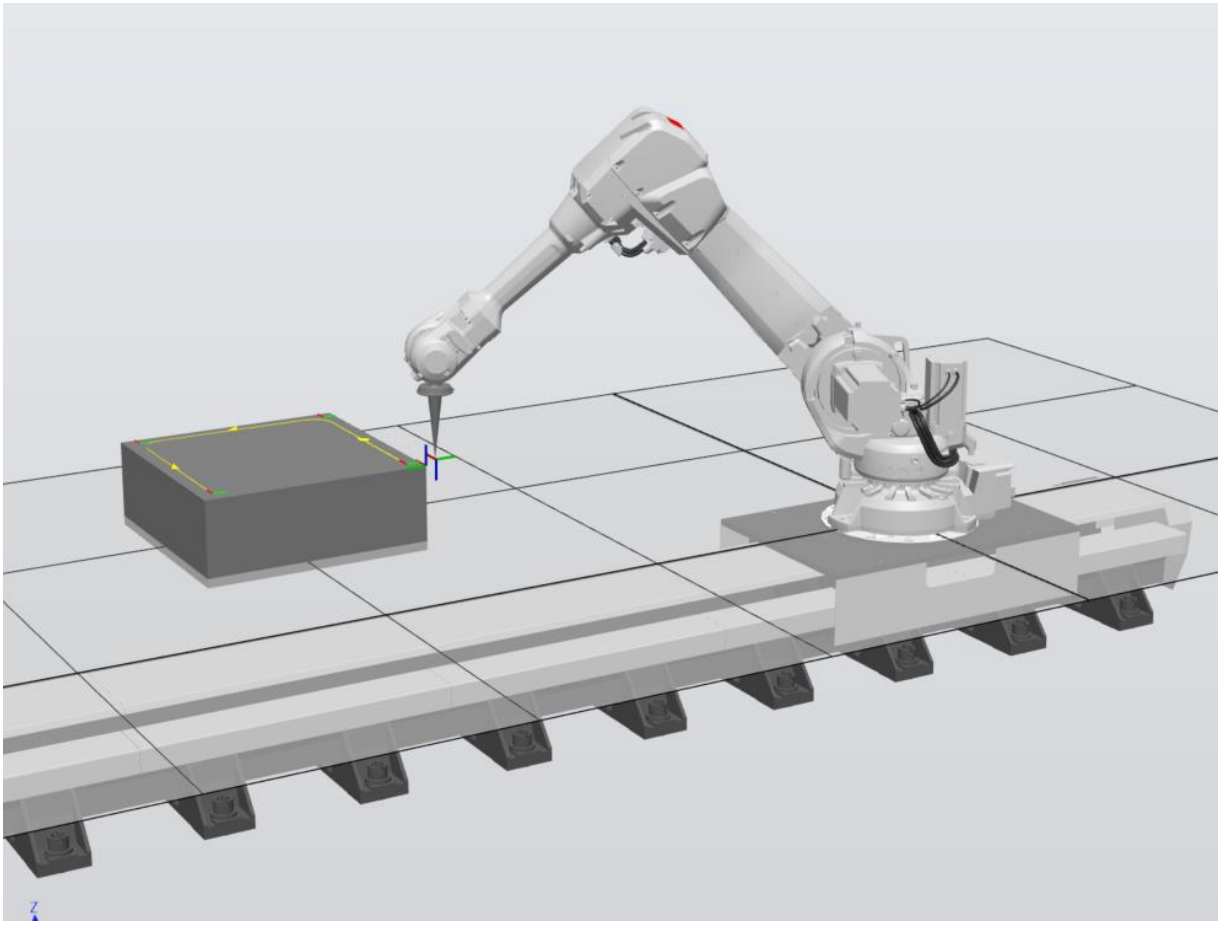


Figura 3.4 RobotStudio. Estación con brazo y track.

3.1.2. Creación de la herramienta

Pese a que disponemos de multitud de herramientas para seleccionar desde la librería de ABB y que pueden servir para darnos una idea de cómo podemos realizar ciertos movimientos, es necesario crear una herramienta propia que cumpla con las dimensiones precisas que se requiere para imprimir en 3D ya que el software calculará las trayectorias necesarias para posicionar el punto final de la herramienta en la ubicación deseada.

Para crear una nueva herramienta en RobotStudio podemos utilizar dos geometrías: una propia que diseñemos y que represente fielmente la herramienta real o una geometría generada automáticamente por RobotStudio que no será una representación de la realidad, pero sí mostrará fielmente el punto final de la herramienta y su centro de masas.

El segundo método se puede llevar a cabo desde el menú Modelado en Crear herramienta. Ahora debemos especificarle los parámetros de posición del Centro de gravedad y TCP de la herramienta real, así como el peso. Esto debemos hacerlo en función del sistema de referencia de RobotStudio para que la herramienta se posicione correctamente.

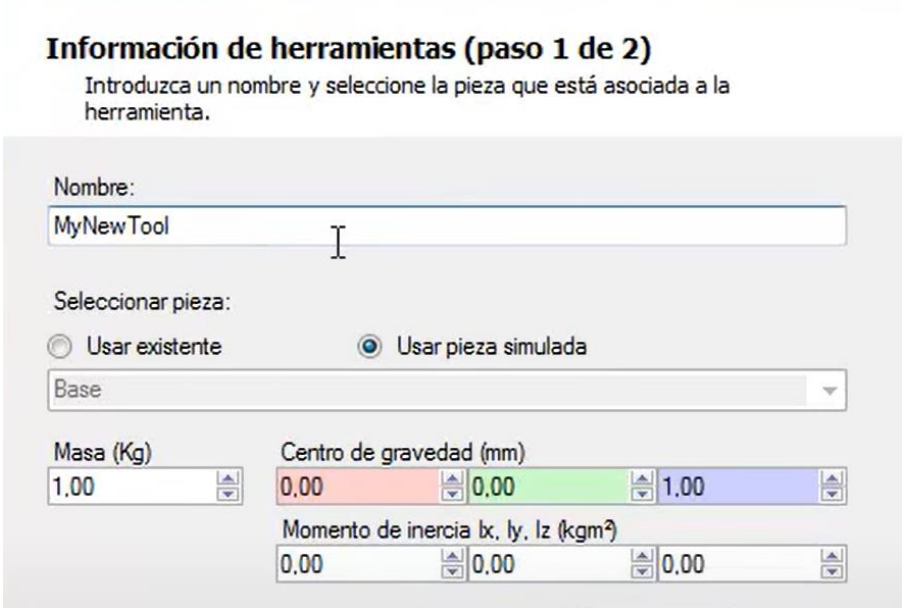


Figura 3.5 RobotStudio. Creación de herramientas (A)

El segundo paso es indicar la posición del TCP, siendo este el punto situado en el extremo de la herramienta y donde ocurren todas las operaciones y movimientos. Una vez seleccionado TCP es necesario añadir con el botor de la flecha derecha (->) para que quede registrado.

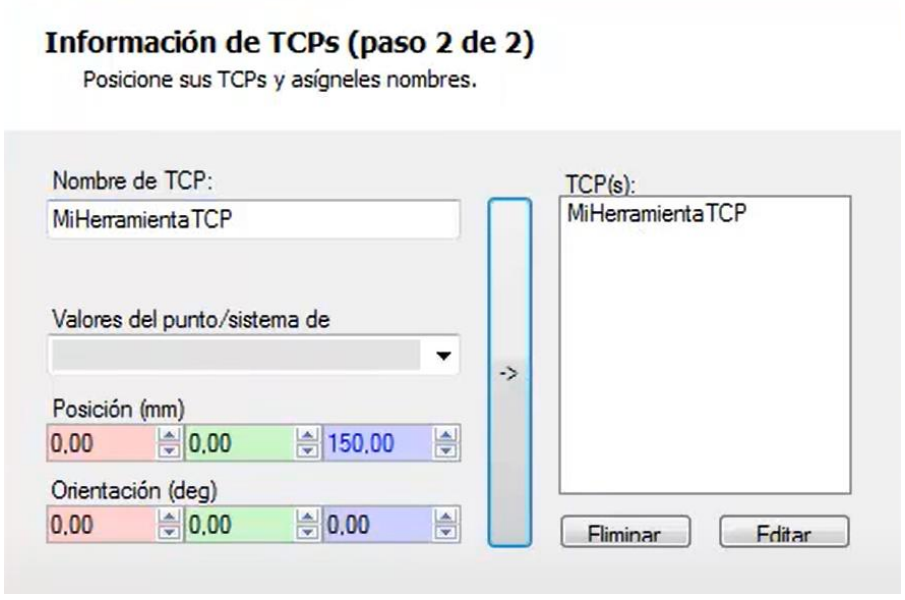


Figura 3.6 RobotStudio. Creación de herramientas (B)

Una vez realizado esto nuestra herramienta aparecerá en el espacio de trabajo, para que la herramienta sea funcional debemos enlazarla con el manipulador arrastrando su icono en la pestaña de diseño hasta el del IRB4600, ahora sí forma parte del brazo.

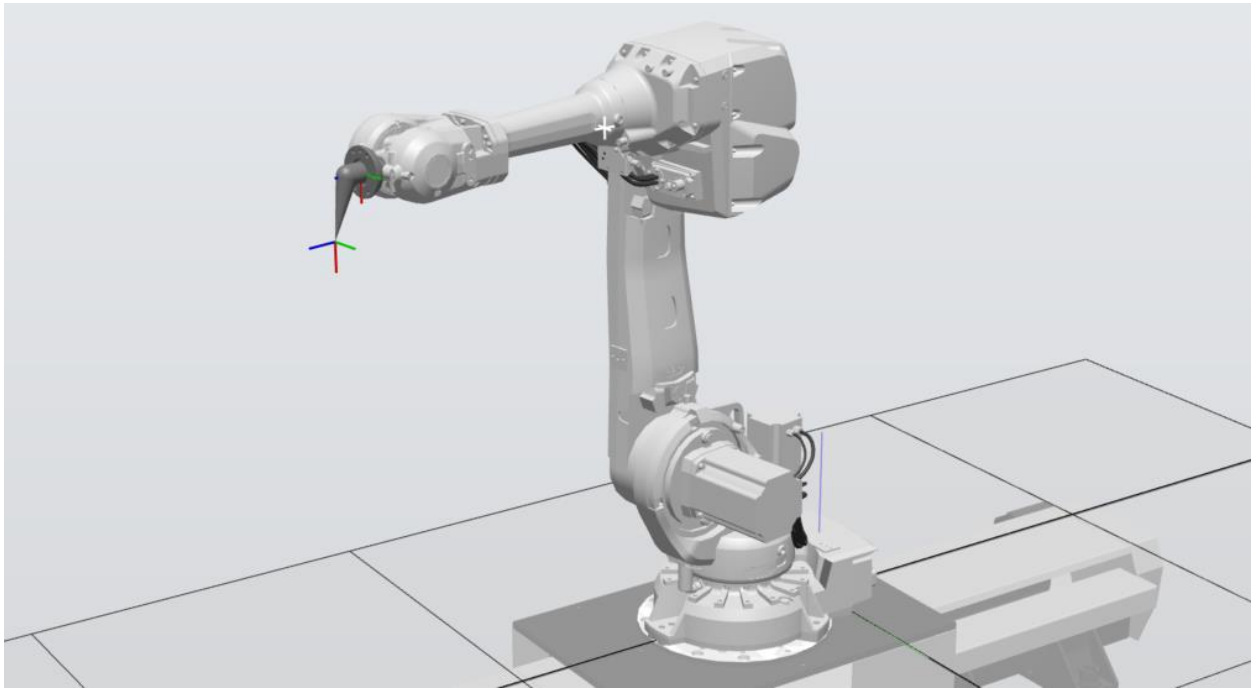


Figura 3.7 RobotStudio. IRB4600 con herramienta propia.

3.1.3. Tanque de arcilla

Al robot real, como ya se ha explicado, se le ha añadido un tanque de arcilla en el brazo de 4ª articulación. Para poder visualizar el movimiento de este en la simulación de RobotStudio vamos a crear un objeto que simule esta estructura y que tenga un comportamiento de colisiones.

Para ello vamos a generar un objeto rectangular de las medidas de tanque y otro objeto más pequeño que servirá de enlace (esto no es técnicamente necesario ya que se puede enlazar el bloque del depósito "en el aire" pero visualmente es más coherente utilizar un bloque de enlace)

Primero vamos a enlazar el bloque de pequeño con la articulación 4, esto se puede hacer con un clic derecho sobre el objeto y seleccionando la opción conectar. Antes de conectar el bloque debe estar en la ubicación que deseemos y al aparecer el mensaje para actualizar la posición de la pieza, tendremos que seleccionar "no". De esta manera el objeto seguirá el movimiento de la articulación.

El objeto que simula el depósito debe quedar 15cm desplazado en sentido positivo del eje y para nuestra instalación, configurándolo de la misma manera que el anterior excepto en la física de colisiones ya que el primer bloque no es necesario que se compute mientras que el bloque del depósito sí es necesario.

Al tener el depósito dentro de la simulación cuando indiquemos un recorrido o un punto a alcanzar al controlador automáticamente se generará una ruta sin colisiones del depósito con el brazo o con otros objetos del entorno como puede ser la mesa de impresión.

 xx0200000022	**ATENCIÓN**: La unidad sobre la que se realiza el proyecto no cuenta con el complemento **Collision Avoidance** de ABB, este complemento permite al controlador IRC5 utilizar algoritmos de planificación de trayectorias seguras con los objetos del entorno (y en nuestro caso del tanque contra el propio robot). Al no disponer de esta función **es necesario comprobar todos los recorridos en la simulación y verificar que no hay riesgo de colisión antes de enviar a imprimir cualquier pieza al robot real.**

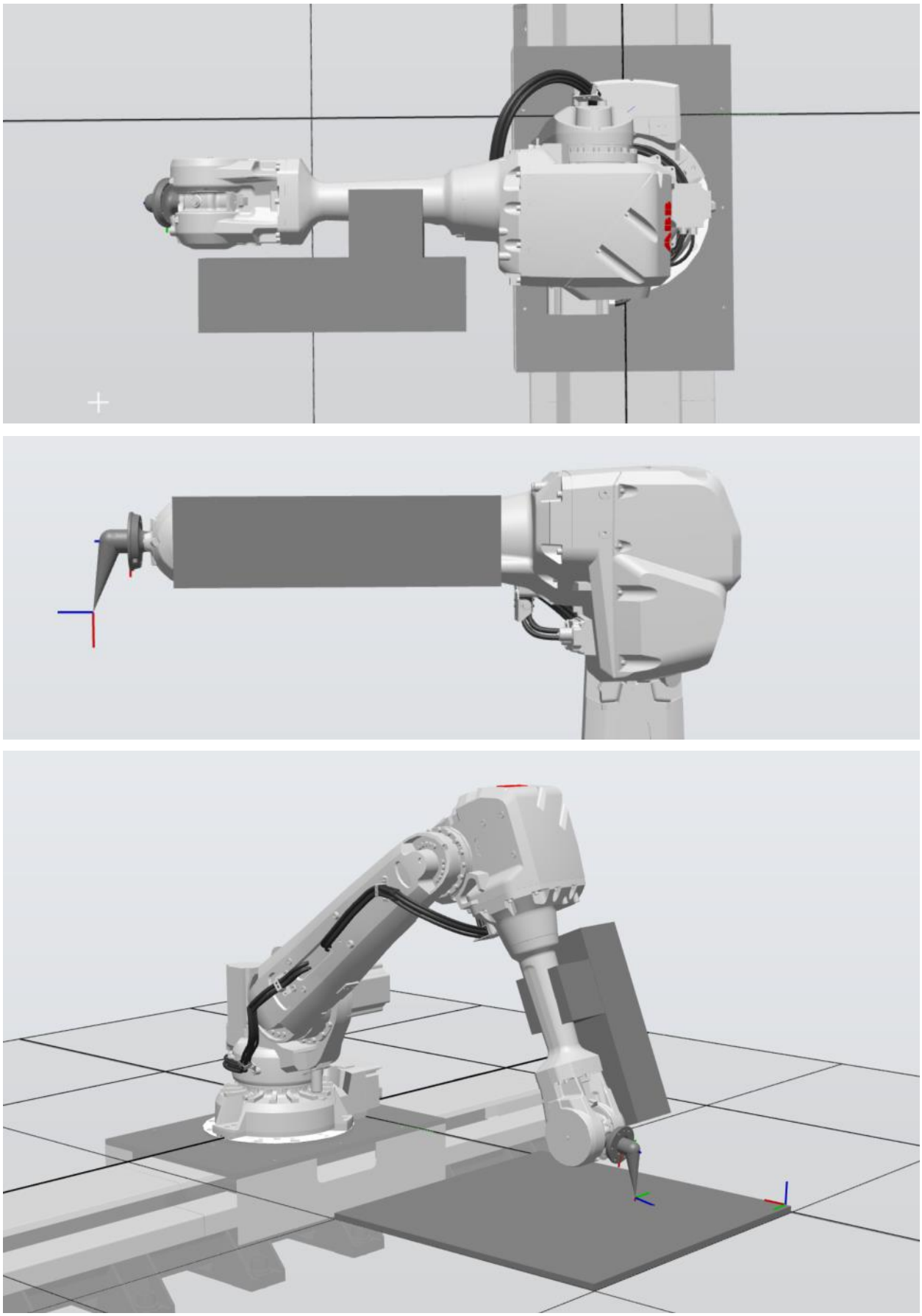


Figura 3.8 RobotStudio. Galería de imágenes del brazo con el Taque de arcilla simulado.

3.2. COMPAS

Uno de los hitos del proyecto es conectar correctamente el robot real IRB4600 de ABB con la plataforma de diseño paramétrico Rhino, crear un canal de comunicación entre estas dos estructuras requiere de múltiples capas de software que vamos a analizar.

Esta iniciativa proviene de la investigación Gramazio Kohler en la ETH Zúrich, [14] donde han creado un paquete llamado COMPAS FAB, basado en python y que facilita la planificación y ejecución de procesos de fabricación con robots directamente desde un entorno de diseño paramétrico [21]. Este entorno de programación se construye sobre COMPAS, un ecosistema de librerías de propósito general de Python integradas en un único núcleo. De esta manera se tiene acceso a resultados de múltiples investigaciones y herramientas que han sido revisadas para facilitar su uso y avance ya que permite construir extensiones que utilicen las librerías existentes y que aumenten de esta manera la capacidad del ecosistema.

COMPAS se centra en el análisis de datos y estructuras para proporcionar el soporte fundamental que se necesita para trabajar en los campos de la arquitectura, diseño geométrico y paramétrico, e incluye soporte para control de robots y fabricación autónoma.

Con estas bases el equipo de investigación de la ETH Zúrich ha desarrollado el conjunto de librerías que permiten comunicar RobotStudio con Rhino 7 y que se explica en el siguiente apartado como se han utilizado en nuestro caso. COMPAS FAB va a permitir comunicar nuestra herramienta de diseño CAD con RobotStudio y a partir de aquí, se desarrollan los programas que permiten procesar los modelos 3D y convertirlos en estructuras para imprimir en 3D y que nuestro controlador de ABB pueda comprender.

3.2.1. COMPAS FAB

El entorno de comunicación de COMPAS FAB (en adelante CF) es un conjunto de librerías que permiten la transferencia de datos entre los procesos de RobotStudio y Rhino 7, haciendo uso de las tecnologías de Visual Studio y ROS como base de la comunicación. RobotStudio y Rhino se encuentran disponibles exclusivamente para el sistema operativo Windows. Sin embargo, la implementación completa y óptima de ROS, a fecha de 2023, está limitada a sistemas Linux.

La necesidad de integrar ROS se origina en la utilización de MoveIt, una librería de planificación de movimientos muy extendida. Esta dependencia, motiva la incorporación de ROS en nuestro ecosistema y, por tanto, de un software que permita ejecutarlo correctamente en Windows. La mejor opción es Docker, una plataforma de virtualización y compartimentación de sistemas que nos permite ejecutar una distribución de Linux en segundo plano. Gracias a Docker podemos establecer una ejecución segura de ROS.

Además de esto, en el lado CAD de Rhino, debemos utilizar un complemento llamado Grasshopper que nos permite añadir una capa de programación visual con bloques y Python para así poder hacer uso de la potencia CAD de Rhino 7 de manera automática siendo Grasshopper donde se ejecutará el programa de impresión 3D

Para entender mejor la relación que existe entre estos programas disponemos del siguiente gráfico proporcionado por el equipo de CF en la documentación que tuvimos acceso durante el proyecto. Aquí se aprecia visualmente la secuencia de comunicación que sigue el ecosistema.

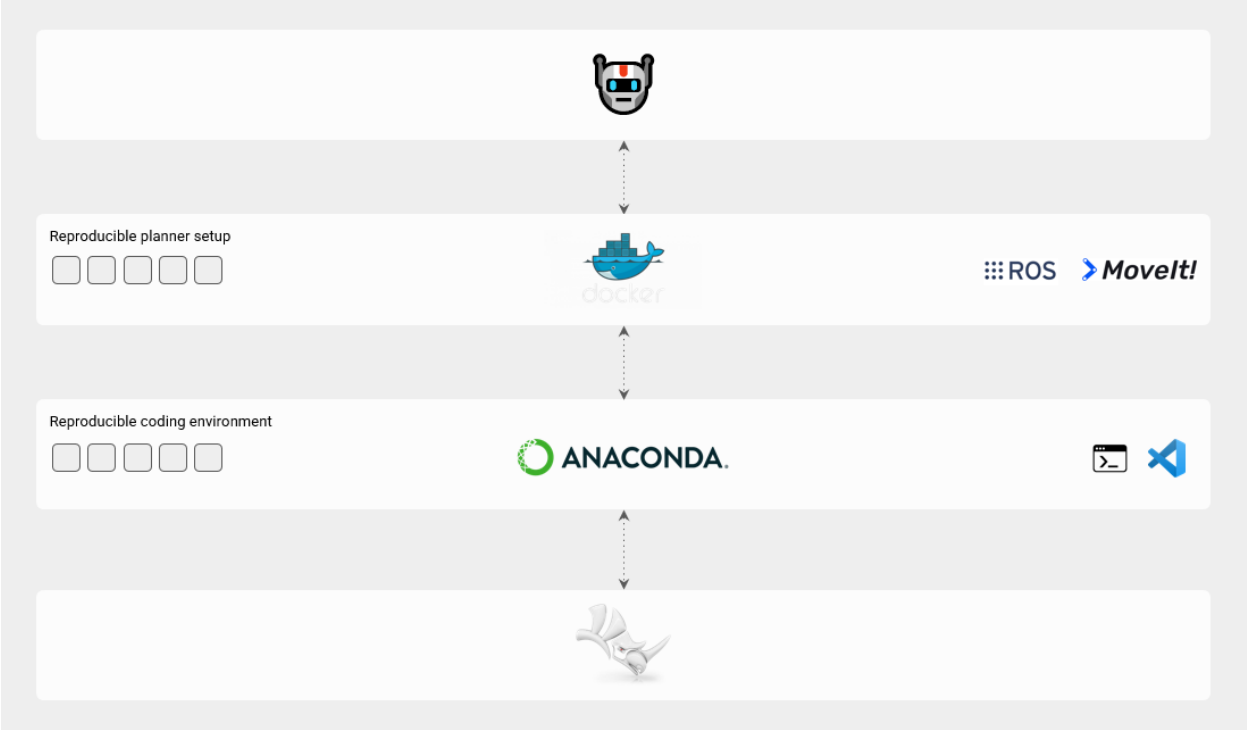


Figura 3.9 Estructura del software utilizado para comunicar Rhino y RobotStudio. [22]

Para comenzar la explicación detallada de CF vamos a explicar a continuación la librería de Python, algunas de sus capacidades y como utilizarla.

Esta librería está disponible para su uso gratuito en GitHub [23], para hacer uso necesitamos tener instalado el software **Anaconda** y desde su ventana de comandos podremos instalar el paquete como añadiríamos cualquier otro paquete desde **GitHub**. Previamente a la instalación debemos tener creado un entorno (enviroment) de Anaconda para poder añadir la librería dentro de este.

Una vez este instalado ya tendremos acceso y podemos ver los archivos que incluye desde Visual Studio seleccionando el entorno donde hemos añadido el paquete, así como la ventana de comandos a utilizar siendo la ventada de Anaconda.

La librería de CF se centra en la manipulación de robots por lo que incluye códigos que permiten realizar transformadas, definiciones de puntos y marcos de referencia, concatenaciones e incluso calcular la cinemática directa e inversa. El paquete completo se asemeja al **Robotic Toolbox para MATLAB** que hemos utilizado en varias ocasiones. De esta manera somos capaces de realizar algunos de los cálculos y simulaciones que se facilitan en el toolbox de MATLAB, pero desde otros entornos, como uno CAD en nuestro caso, ya que este paquete es compatible con diferentes softwares.

En estos ejemplos se muestra como posicionar un punto respecto un marco o sistema de referencia y como se pueden realizar las típicas transformaciones de Rotación, Traslación y Escalado directamente igual que en el toolbox, por ejemplo, con el comando trot(). En el tercer ejemplo se muestra la opción de crear nuestro propio robot, con el comando LINK, ajustando el tipo de articulación para cada eslabón.

Cuando tengamos definido un robot podemos obtener directamente la cinemática directa para una configuración de articulaciones dada y la cinemática inversa para una posición deseada.

```
"""Example: 'point in frame'
"""
from compas.geometry import *

point = Point(146.00, 150.00, 161.50)
xaxis = Vector(0.9767, 0.0010, -0.214)
yaxis = Vector(0.1002, 0.8818, 0.4609)

F = Frame(point, xaxis, yaxis)  # coordinate system F
P = Point(35., 35., 35.)  # point in F (local coordinates)

P_ = F.to_world_coordinates(P)  # point in global (world) coordinates
print("The point's world coordinates: {!r}".format(P_))

P2 = F.to_local_coordinates(P_)
print("The point's local coordinates: {!r}".format(P2))  # should equal P
print(allclose(P2, P))
```

Figura 3.10 Ejemplo de COMPAS FAB. Definición de un punto.

```
from compas.geometry import *

axis, angle = [0.2, 0.4, 0.1], 0.3
R = Rotation.from_axis_and_angle(axis, angle)
print("Rotation:\n", R)

translation_vector = [5, 3, 1]
T = Translation.from_vector(translation_vector)
print("Translation:\n", T)

scale_factors = [0.1, 0.3, 0.4]
S = Scale.from_factors(scale_factors)
print("Scale:\n", S)

point, normal = [0.3, 0.2, 1], [0.3, 0.1, 1]
R = Reflection.from_plane((point, normal))
print("Reflection:\n", R)
```

Figura 3.11 Ejemplo de COMPAS FAB. Transformaciones.

```
# create robot model
model = RobotModel("robot", links=[], joints=[])

# add links
link0 = model.add_link("world")
link1 = model.add_link("link1", visual_mesh=mesh1,)
link2 = model.add_link("link2", visual_mesh=mesh2,)

# add the joints between the links
model.add_joint("joint1", Joint.REVOLUTE, link0, link1, origin1, axis1)
model.add_joint("joint2", Joint.REVOLUTE, link1, link2, origin2, axis2)
```

Figura 3.12 Ejemplo COMPAS FAB. Creación de robot con LINK.

```
# Create config
config = model.zero_configuration()

# Get FK for tip
print (model.forward_kinematics(config))
# Get FK for base
print (model.forward_kinematics(config, link_name=model.get_base_link_name()))


  from compas_fab.backends.kinematics.solvers import UR5Kinematics

  f = Frame((0.417, 0.191, -0.005), (-0.000, 1.000, 0.00), (1.000, 0.000, 0.000))
  solutions = UR5Kinematics().inverse(f)
```

Figura 3.13 Ejemplo COMPAS FAB. Cinemática directa e inversa.

3.2.2. Docker

Como hemos explicado antes, Docker es una plataforma de virtualización basada en **contenedores** que permite desarrollar, implementar y ejecutar aplicaciones en diferentes entornos informáticos. La principal diferencia con un software de virtualización tradicional es que en Docker no existe la necesidad de virtualizar un sistema operativo completo sino únicamente entornos aislados que contienen los componentes necesarios para que la aplicación a ejecutar funcione correcta e independientemente.

Esto libera una gran carga computacional al equipo, teniendo potencia libre para el resto de tareas (el sistema completo del proyecto es relativamente pesado y para que funcione con fluidez necesitará igualmente un equipo potente con bastante RAM).

Dependiendo de la versión de Windows en la que se ejecute Docker o el proyecto concreto la virtualización se realizará de dos maneras, con Hyper-V o WSL, en nuestro caso usaremos el **WSL 2** (subsistema de Windows para Linux). WSL es una característica de Windows 10 en adelante que permite a los usuarios ejecutar un sistema operativo Linux en paralelo con Windows.

Para utilizar WSL, primero debemos habilitarlo desde la configuración de Windows en el panel de control, después debemos descargar e instalar una distribución de Linux, lo más sencillo es utilizar la que nos proporciona Microsoft desde la Microsoft Store. Cabe destacar que WSL ejecutará las distribuciones "en segundo plano" es decir sin interfaz de usuario gráfica o de texto a la que podamos acceder ya que sirve únicamente de motor para otras aplicaciones como Docker.

Al estar la máquina virtual dentro de Windows este puede gestionar los recursos una mejor manera que utilizando un software separado. En el caso de que queramos limitar la potencia consumida, ya sea en núcleos del procesador o memoria RAM podemos hacerlo como se explica en este enlace: Limitar WSL2 (para nuestro proyecto se ha limitado la memoria RAM disponible a 6GB para WSL, de 16GB totales en el equipo).

En cuando al uso de Docker en nuestro proyecto, será a través de Visual Studio, necesitando instalar la extensión Docker. Desde Visual abriremos los archivos **yml** que contienen la versión de ROS, MoveIt y ABB que vamos a ejecutar. Sobre el archivo abierto en Visual podemos hacer clic derecho y seleccionar **Compose Up**, esto ejecutará el código en Docker creando los contenedores necesarios.

Los archivos .yaml o .yml se utilizan para definir la configuración de los contenedores que se ejecutarán en un entorno de ROS incluyendo información como la imagen a utilizar, los puertos que se deben exponer, los volúmenes de datos que se deben montat o las variables de entorno necesarias.

3.2.3. COMPAS RRC

Una vez que hemos configurado el entorno necesario para ejecutar ROS, podemos cargar el controlador del robot en Docker. Este controlador se presenta como un nodo de ROS que facilita la comunicación entre RobotStudio o un controlador de ABB (IRC5) y la plataforma CAD Rhino 7.

COMPAS RRC es desarrollado por CF, actúa como el driver fundamental para establecer la comunicación con ROS y utiliza las bibliotecas previamente mencionadas de CF, que incluyen un puente en Python para interactuar con ROS. El controlador RRC desempeña el papel de cliente en RobotStudio y, además, actúa como servidor en Rhino.

A este conjunto se agrega MoveIt!, un nodo dedicado a la planificación de trayectorias. En el ámbito de los controladores empleados en robótica, generalmente se prefiere aquellos que funcionan en tiempo real y proporcionan una alta frecuencia de realimentación. Sin embargo, el control RRC difiere en este aspecto, ya que se clasifica como un control no en tiempo real con una frecuencia de realimentación más baja. Esto se explica debido a que el control RRC no gestiona directamente los movimientos del robot, tarea que recae en el IRC5 de ABB.

El control RRC podría considerarse como un **nodo de comunicación**, ya que su función principal es transmitir, recibir y procesar datos de manera que el IRC5 pueda interpretarlos. Dada la complejidad que implica la integración de múltiples capas de programación, diferentes software y equipos con diversas capacidades, mantener una alta frecuencia de realimentación se ve limitado debido a la **latencia** que se genera en esta solución.

El control RRC se divide en tres partes diferenciadas:

- COMPAS (Python Framework), instalado como una librería de CF en Grasshopper (Rhino 7).
- ROS (Integration Layer), que opera como nodo de ROS encargado de establecer la comunicación entre ambos equipos.
- Execution Layer (Machine Interface), que se instala en el propio controlador IRC5 físico (o en la estación de RobotStudio).

Este control bastante completo y abarca una amplia gama de características, que incluyen el control de movimientos, la gestión de señales de entrada/salida, el control del flexpendant, la gestión de tiempos de espera y, además, brinda la opción de agregar comandos personalizados directamente en el lenguaje Rapid.

3.3.3. Grasshopper

Grasshopper es una extensión de Rhino 7 que añada una ventana de programación al sistema, permitiendo integrar el diseño CAD con la programación con bloques y en Python. Aquí es donde vamos a usar las librerías de COMPAS FAB, que incluyen varios bloques básicos que permiten por ejemplo crear una representación del robot en Rhino, comunicarnos con el nodo de ROS para enviar los puntos o ubicaciones a alcanzar o algunos bloques de cálculos de cinemáticas directas e inversas.

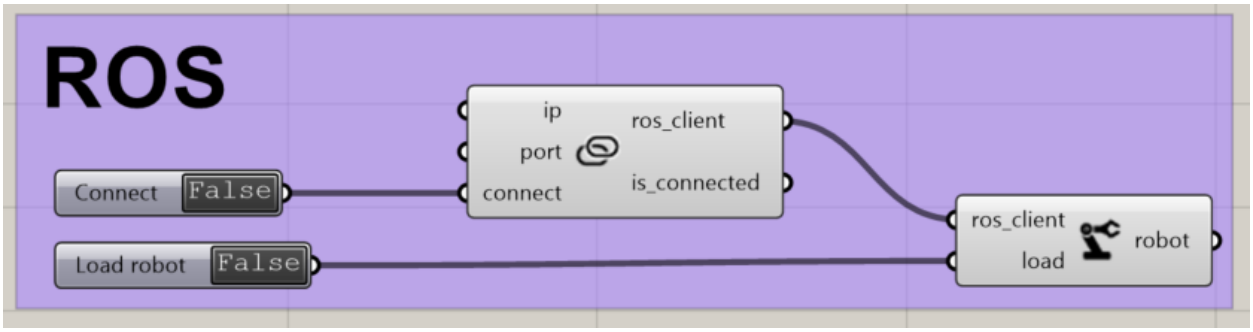


Figura 3.14 Grasshopper. Ejemplo básico.

Los bloques que se ven en la imagen anterior son bloques incluidos en la librería de CF que permiten crear un enlace con ROS para obtener la información del robot, bloque central, y que proyecta la representación gráfica del mismo en Rhino para poder ver los movimientos y ubicaciones en las que se encuentre.

Cada uno de los bloques tiene algunas entradas que permiten introducir los datos necesarios, aunque no siempre es obligatorio que están conectados, de igual forma que las salidas. La librería tiene bloques de conexión con ROS, bloques de cálculo de cinemática directa e inversa, así como planificación de movimiento, pero estos últimos no serán utilizados en nuestro proyecto ya que el control de los movimientos del robot se realizará directamente en el IRC5

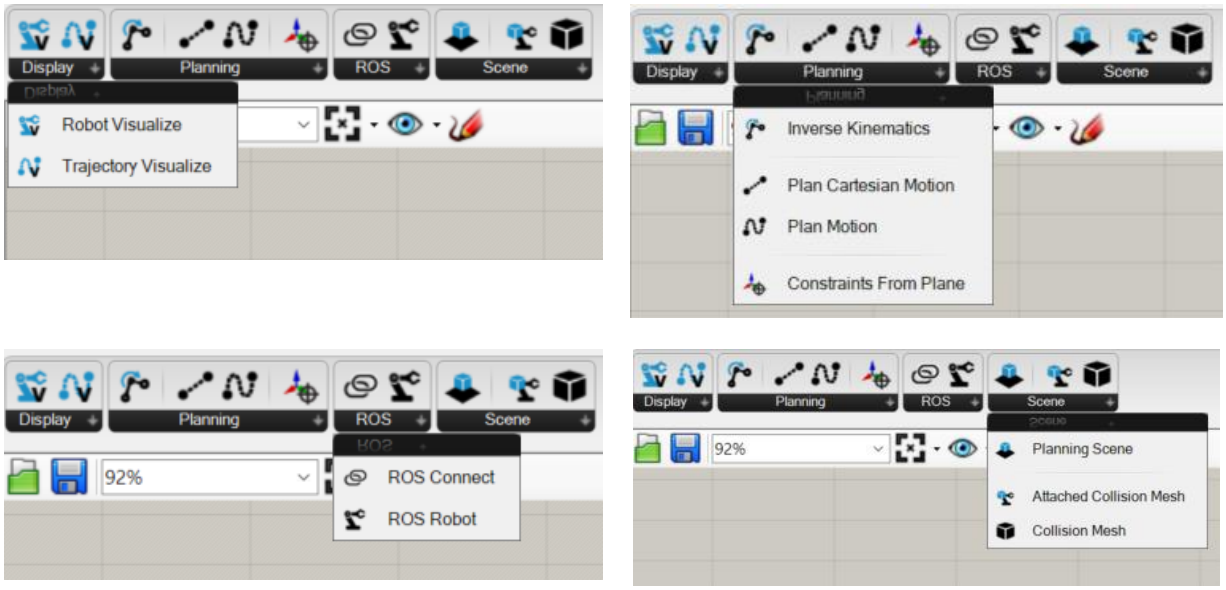


Figura 3.15 Grasshopper. Bloques disponibles en la librería de COMPAS FAB.

Hay que mencionar el tratamiento de los datos en Grasshopper ya que utiliza listas en vez de matrices y tipos de datos. Las **listas** son estructuras de datos más versátiles, dinámicas y fáciles de usar en comparación con las matrices. Las matrices tienen un tamaño fijo y requieren una gestión manual de la memoria, mientras que las listas se adaptan mejor a las necesidades cambiantes y ofrecen una amplia gama de operaciones integradas para la manipulación de datos. Esta forma de tratar la información nos dará la posibilidad de descomponer los datos (como un punto, dado por sus coordenadas) de manera sencilla.

3.4. PROCESADO DE MODELOS 3D

Ya habiendo realizado el apartado de comunicación y pudiendo enviar comandos de movimiento a nuestro IRB 4600, tenemos que realizar el programa capaz de transformar una pieza en 3D de Rhino en comandos de impresión. Este programa se ha realizado en Grasshopper siguiendo una programación mixta de bloques y Python, aunque principalmente por bloques, algunos de estos bloques provienen de la librería CF ya explicada, otros son bloques nativos de Grasshopper y algunos han sido desarrollados específicamente para esta aplicación.

El proceso de transformar una estructura, modelo o diseño tridimensional en comandos de impresión es habitualmente realizado por softwares específicos denominados **Slicers** en el entorno de impresoras 3D, estos softwares son capaces de descomponer el modelo completo en centenares o miles de capas del grosor de la boquilla que se va a utilizar para la impresión de manera que superponiendo estas capas se forme el objeto. Estas capas a su vez deben ser procesadas en secuencias de movimiento para que el firmware de la impresora pueda guiar a la cabeza de extrusión al lugar correcto.

Las capas se transformas en una secuencia de puntos y el Slicer envía los comandos para alcanzar esas ubicaciones a la impresora en Control Numérico, actualmente codificado como un archivo .gcode.

El programa desarrollado para el IRB4600 debe hacer un proceso similar descomponiendo los modelos en **capas** de una altura determinada y posteriormente esas capas en **secuencias de puntos** que serán enviados el control RRC del IRC.

A continuación, se muestra el programa completo en el que se pueden apreciar 4 grandes secciones, la primera llamado “Generar robot”, es la encargada de mostrar la representación del IRB4600, las secciones “Procesado de modelos 3D” y “Procesado de modelos 3D (método para impresión de esferas y similares)” son las encargadas de transformar los modelos en capas y, posteriormente, en secuencias de puntos. Finalmente, la sección “Procesado de puntos y envío a RobotStudio” ajusta estos puntos a un formato legible por RobotStudio y cambia algunas características importantes antes de enviarlos.

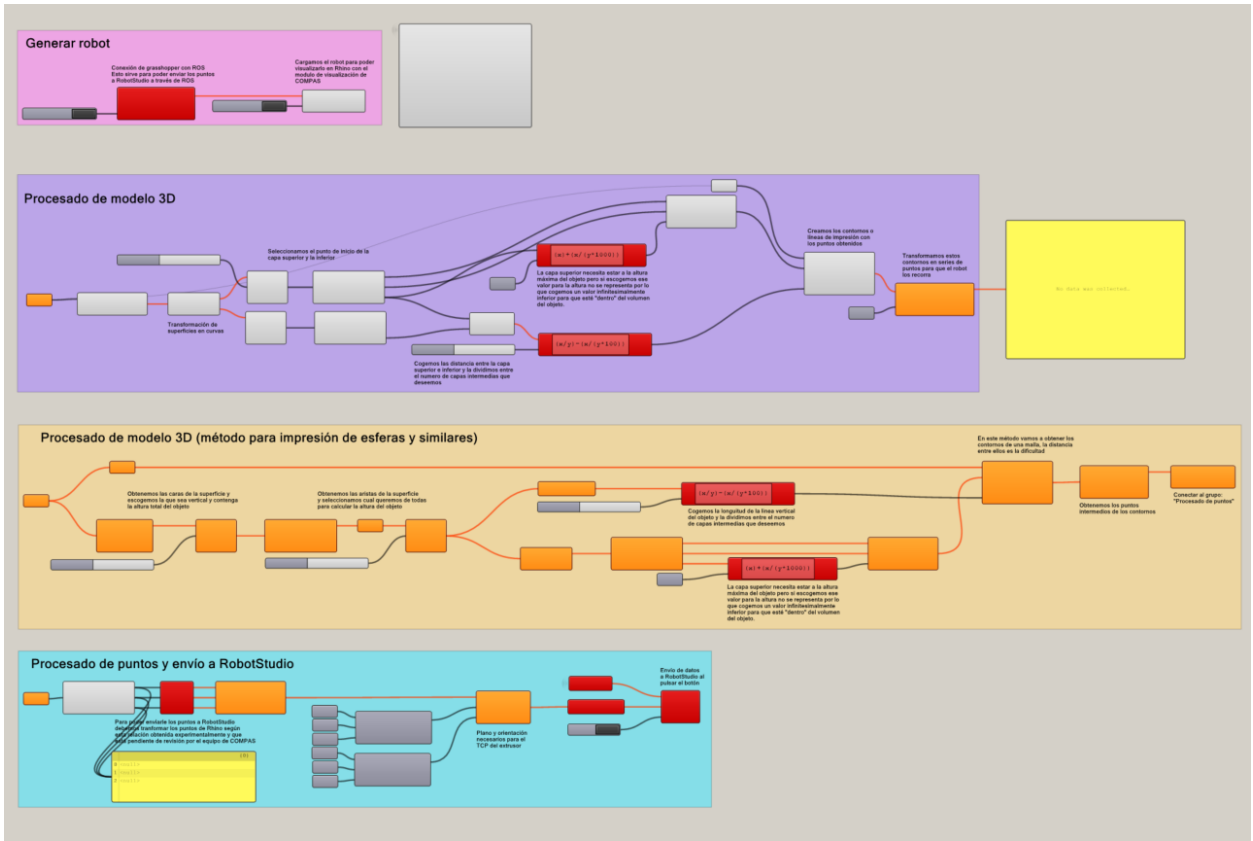


Figura 3.16 Programa de impresión 3D desarrollado. Vista completa.

3.4.1. Generar robot

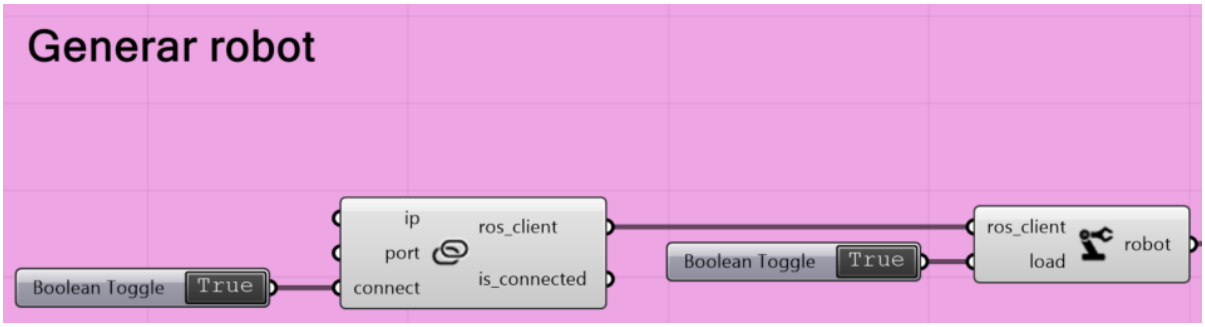


Figura 3.17 Grasshopper. Sección generar robot.

Esta primera sección permite funcionar a todo el programa ya que **establece la conexión con ROS**, que está ejecutado en Docker. De izquierda a derecha vamos a explicar el funcionamiento de las piezas presentes.

El primer bloque es un interruptor, que habilita o deshabilita la conexión, durante la programación del código es conveniente tener desactivada la conexión ya que ralentiza en gran medida el funcionamiento de Grasshopper al estar calculando los resultados constantemente.

El segundo, establece la conexión entre Grasshopper y ROS, durante la ejecución de ROS en Docker se especificó la dirección IP y puerto para acceder el servidor de ROS, así que usando esta dirección y puerto se puede encontrar al servidor y comprobar si acepta la conexión. Este bloque es parte de la librería CF.

Por último, encontramos el bloque encargado de obtener la información del Robot desde ROS solicitando el modelo y la geometría del robot. Este bloque es parte de la librería CF.

Ahora que Grasshopper tiene la información del servidor de ROS y del Robot utilizado podemos representarlo en Rhino gracias al bloque Robot Visualize incluido en la librería que utiliza el modelo del robot que se encuentre en el servidor de ROS para realizar la representación en Rhino.

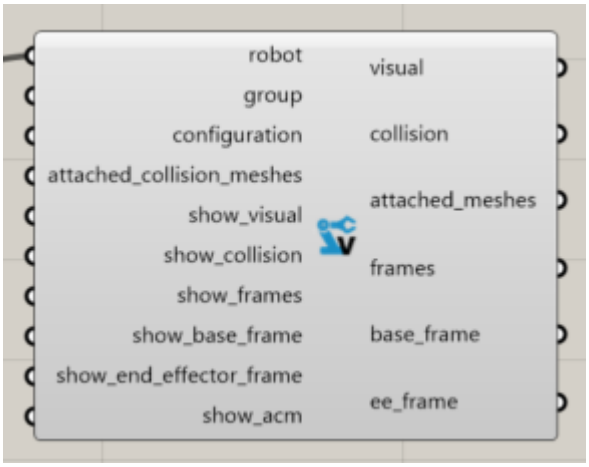


Figura 3.18 Grasshopper. Bloque Robot Visualize

Para terminar de explicar la conexión con ROS vamos a detallar el código que genera el servidor de ROS y que se carga en Docker desde VisualStudio, este código proviene de la librería CF y se ha adaptado a nuestro modelo de robot y versión de ROS.

```
version: '2'
services:
  abb-driver:
    image: gramaziokohler/ros-kinetic-dfab:21.12 #Vamos a usar la version de
KINETIC ya que el control rrc está en esta version
    container_name: abb-driver
    environment:
      - ROS_HOSTNAME=abb-driver
      - ROS_MASTER_URI=http://ros-core:11311
    depends_on:
      - ros-core
    command:
      - roslaunch
      - --wait
      - compas_rrc_driver
      - bringup.launch
     # - robot_ip:=192.168.125.1  #Control real
      - robot_ip:=host.docker.internal #Robot simulado
      - robot_streaming_port:=30101
      - robot_state_port:=30201
      - namespace:=rob1

  moveit-demo:
    image: gramaziokohler/ros-kinetic-dfab:21.12
    container_name: moveit-demo
    environment:
      - ROS_HOSTNAME=moveit-demo
      - ROS_MASTER_URI=http://ros-core:11311
      - ros-core
      # To use the web-based GUI, uncomment the following line
      # - gui
    command:
      - roslaunch
      - --wait
      - abb_irb4600_60_205_moveit_config
      - demo.launch
      - use_rviz:=false

  ros-core:
    image: gramaziokohler/ros-kinetic-dfab:21.12
    container_name: ros-core
    ports:
      - "11311:11311"
    command:
      - roscore

  ros-bridge:
    image: gramaziokohler/ros-kinetic-dfab:21.12
    container_name: ros-bridge
```

```
    environment:
      - "ROS_HOSTNAME=ros-bridge"
      - "ROS_MASTER_URI=http://ros-core:11311"
    ports:
      - "9090:9090"
    depends_on:
      - ros-core
    command:
      - roslaunch
      - --wait
      - rosbridge_server
      - rosbridge_websocket.launch
      - unregister_timeout:=28800

  ros-fileserver:
    image: gramaziokohler/ros-kinetic-dfab:21.12
    container_name: ros-fileserver
    environment:
      - ROS_HOSTNAME=ros-fileserver
      - ROS_MASTER_URI=http://ros-core:11311
    depends_on:
      - ros-core
    command:
      - roslaunch
      - --wait
      - file_server
      - file_server.launch
```

Este código es un archivo YAML que define un conjunto de servicios y contenedores para configurar ROS en Docker. Los diferentes servicios que se definen se ejecutarán como contenedores separados, aquí definimos el nombre del servicio u los detalles de la imagen que se utilizara.

- **abb-driver.** Utilizamos la imagen usada por el equipo de COMPAS en la versión KINETIC ya que el controlador RRC se encuentra en esa versión. Entre los comandos que debemos especificar se encuentre la IP del robot aquí tenemos dos opciones, especificar **la IP del controlar IRC5** que esté conectado físicamente por ethernet a nuestro portátil o indicar **host.docker.internal** para usar el robot simulado en RobotStudio
- **moveit-demo.** Lanzamos el paquete MoveIt! En los comandos especificamos el modelo exacto de robot que vamos a usar (IRB4600 60(kg) 205(cm)).
- **ros-core.** Nodo maestro de ROS.
- **ros-bridge.** Permite la comunicación entre ROS y aplicaciones con el protocolo WebSocket.
- **ros-fileserver.** Se utiliza para entregar archivos del entorno de ROS.

El código debe ser ejecutado en VisualStudio con la función Compose Up (extensión de docker) teniendo previamente abierto la aplicación de escritorio de Docker. Una vez estén los contenedores activos podemos utilizar los bloques de ROS en Grasshopper (sección Generar Robot) y obtendremos una representación de nuestro IRB4600 en Rhino.

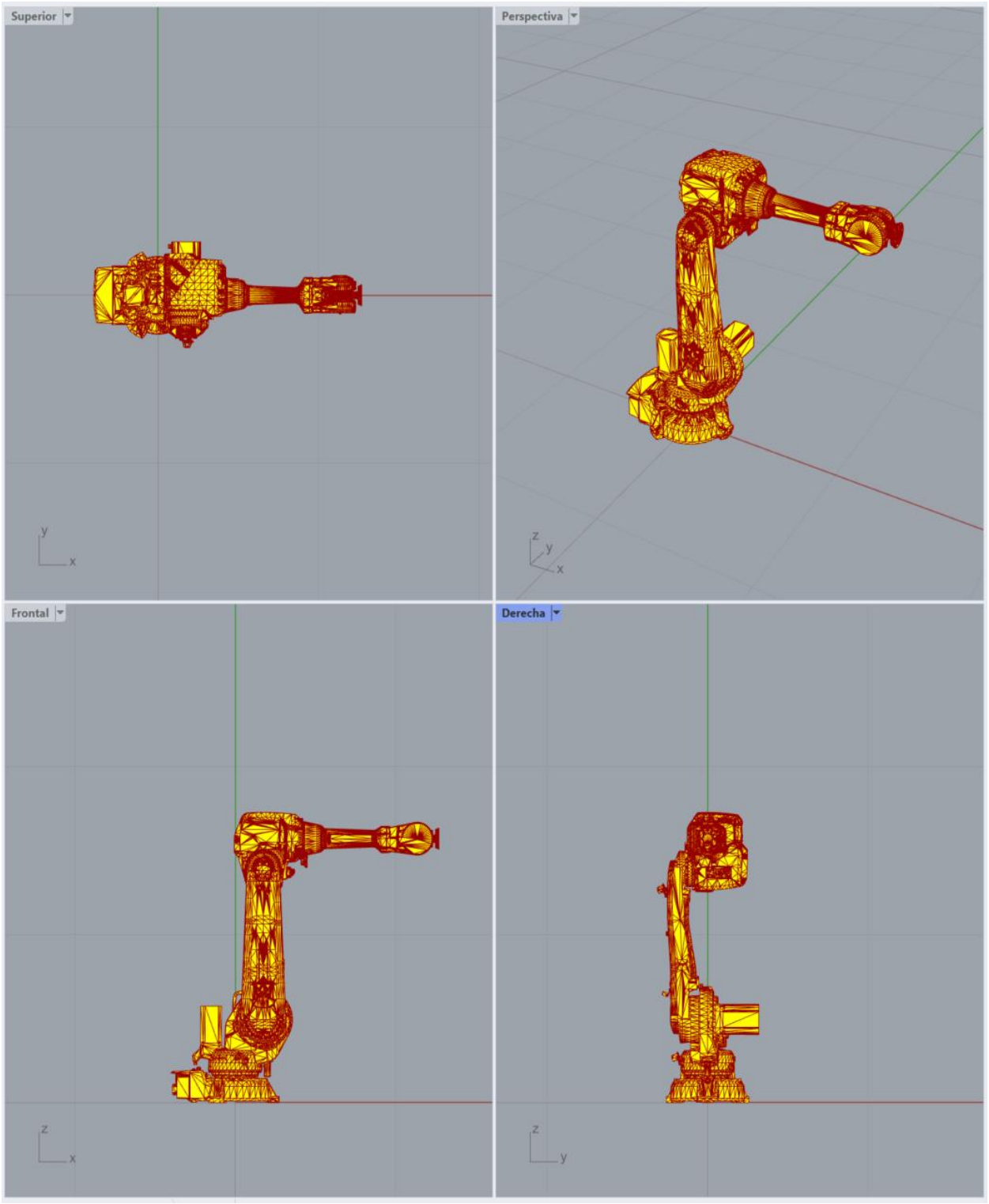


Figura 3.19. IRB4600 en Rhino

3.4.2. Procesado de modelos 3D

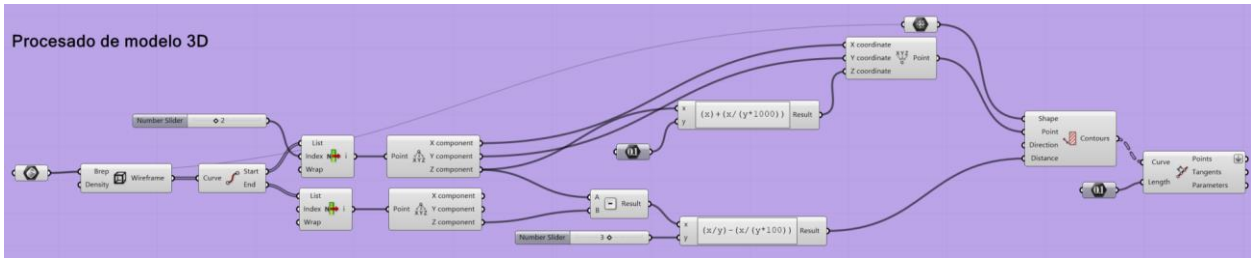


Figura 3.20 Grasshopper. Procesado de modelos 3D

En esta sección trabajaremos con modelos 3D realizados o importado a Rhino, concretamente modelos con paredes y bases lisas (sin curvatura, estos objetos se tratarán más adelante). Para explicar cómo descomponemos un objeto tridimensional en una lista de puntos vamos a utilizar de ejemplo un cilindro.

En Rhino existen diferentes tipos de sólidos: sólido, malla y brep; el primero es el más genérico y no incluye una representación con polígonos de las caras del objeto, una malla añade esta información. El brep (Boundary Representation) permite representar objetos más complejos combinando superficies y bordes. Estos tres tipos pueden ser transformados entre ellos y nosotros usaremos únicamente solidos para este ejemplo.

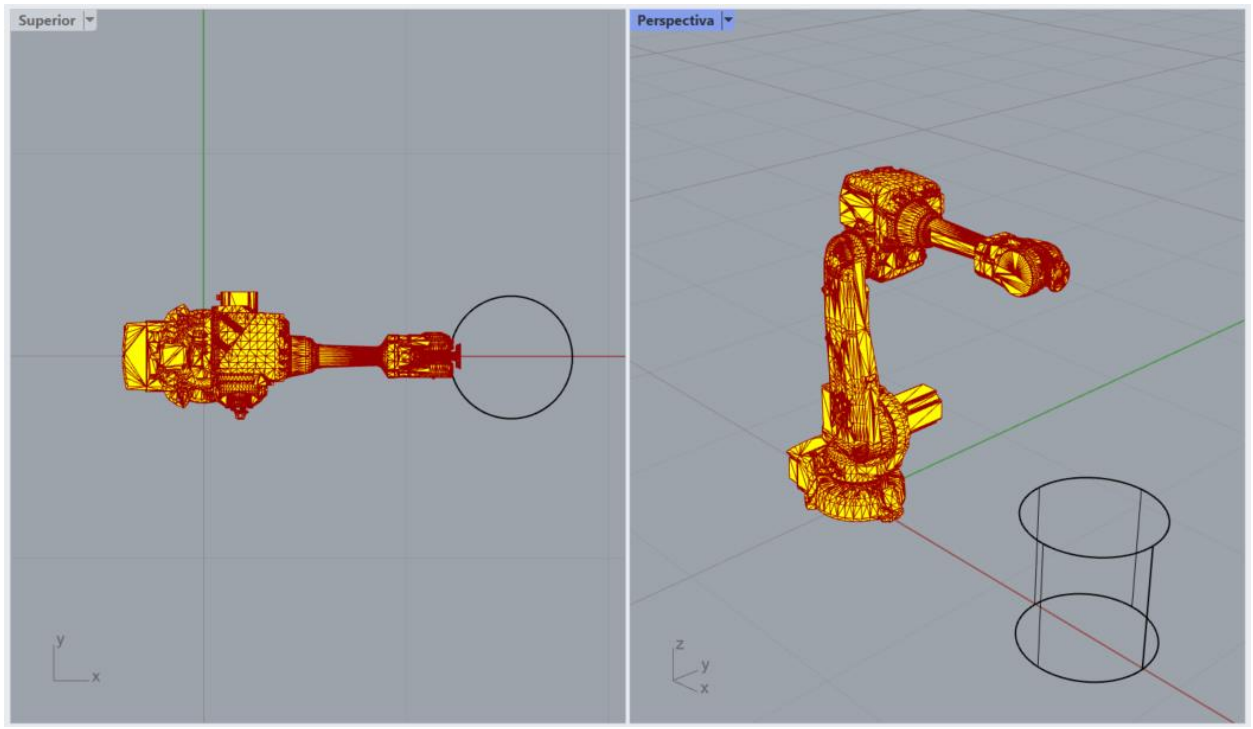


Figura 3.21 Cilindro de ejemplo en Rhino

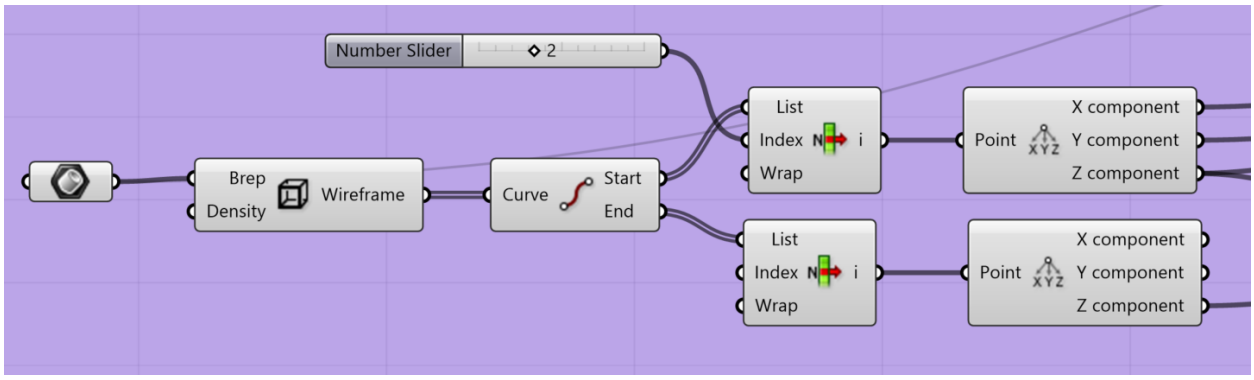


Figura 3.22 Grasshopper. Procesado de modelos 3D. Zoom 1

Para comenzar el tratamiento debemos seleccionar el objeto en Rhino y añadirlo al primer bloque con la opción “Set one Brep” en el clic derecho.

El bloque Wireframe extrae los contornos del objeto, esto contornos son las circunferencias superiores en inferiores del cilindro y la recta vertical que entrega la altura, tal y como se observe en la Figura 3.23 marcados con la línea verde.

En el bloque Curva se descomponen las circunferencias obteniendo un punto para cada una, la capa superior en Start y la capa inferior en End.

Los bloques de listas seleccionan únicamente un punto en cada caso, arriba el punto de inicio de la capa superior y abajo la inferior. Finalmente se descomponen esos puntos en sus componentes X, Y, Z.

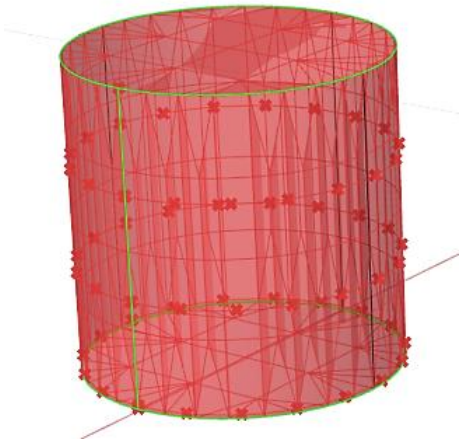


Figura 3.23. Modelo 3D. Cilindro con Wireframe en verde

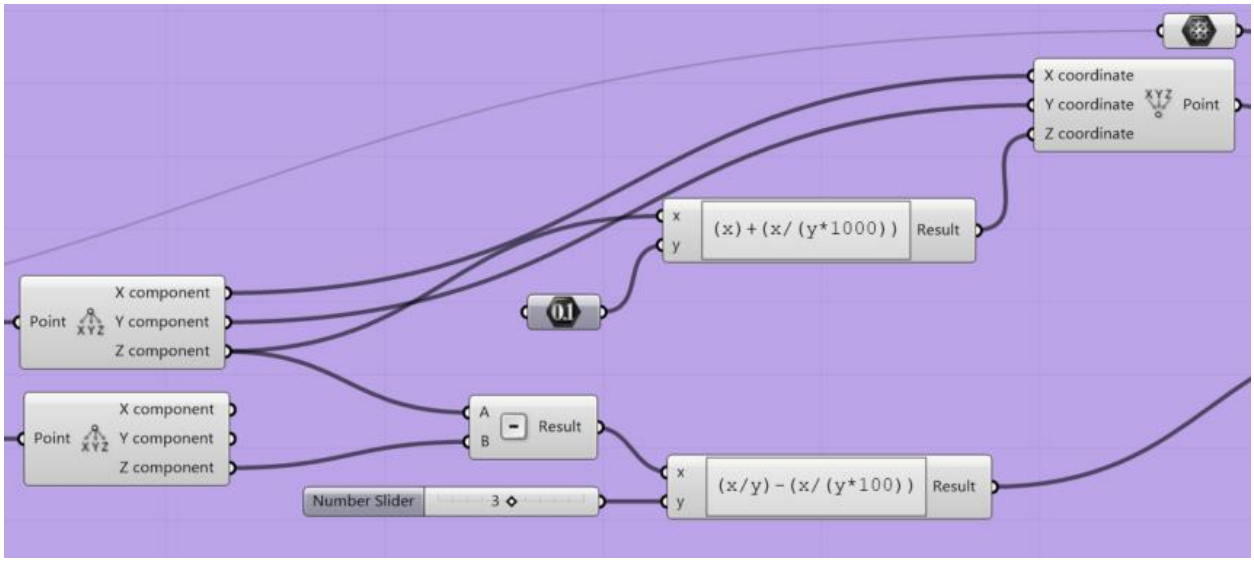


Figura 3.24 Grasshopper. Procesado de modelos 3D. Zoom 2

Ahora necesitamos crear las capas de puntos intermedias del modelo, para hacer esto calcularemos la altura del objeto a imprimir restando a la componente z de la capa superior la de la capa inferior (que no tiene por qué estar situada en el suelo a z=0). Una vez sabemos la altura dividimos la altura entre el número de capas para obtener la distancia intermedia.

En el caso de la capa superior si seleccionamos la altura exacta del objeto (z de la capa superior) esa capa no se representará ya que Rhino la interpreta como fuera del sólido, por esto introducimos un valor de altura infinitesimalmente inferior que sí estará dentro del volumen.

El bloque pequeño superior, conectado por la izquierda al bloque inicial del objeto sirve para guardar como Malla nuestro solido ya que ahora vamos a crear los contornos que definirán las curvas de las capas intermedias.

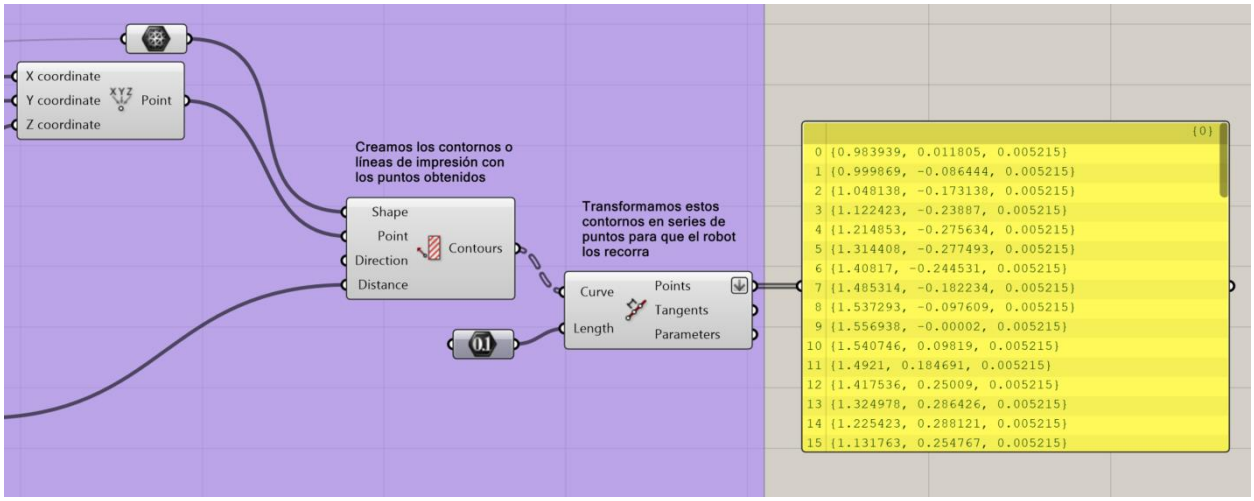


Figura 3.25 Grasshopper. Procesado de modelos 3D. Zoom 3

Para finalizar esta sección creamos los contornos con el bloque de Contornos, que tiene como entradas el sólido en formato Malla, el punto donde iniciará los contornos (siendo el punto donde comienza cada círculo, en el plano horizontal, se ha usado un punto de la capa superior pero no implica que la impresión vaya a comenzar por esta capa), la dirección de creación del contorno (horario o antihorario, en nuestro caso es irrelevante) y la distancia a crear los contornos.

Estos contornos se descomponen en una secuencia de puntos con el último bloque que separa los puntos una distancia fija, si queremos más precisión en la figura final podemos disminuir este valor (el robot viajará en línea recta de punto a punto). En la Figura 3.26, a la derecha, se pueden ver estos puntos.

El panel amarillo a la derecha de la sección se ha puesto de manera indicativa para mostrar cómo se obtienen los puntos, en el caso de querer imprimirlos deben ser conectados a la última sección del programa, que se detalla más adelante.

Estos puntos se crean en orden creciente, las capas con una altura inferior se procesan primero (como es lógico en el caso de impresiones 3D) y dentro de cada capa siempre comenzando y finalizando en el punto indicado en la entrada Point de forma que se evitan movimientos de desplazamiento del cabezal extrusor permitiendo una impresión continúa.

Figura 3.26 Modelo 3D. Cilindro con puntos de capas en verde

3.4.3. Procesado de modelos 3D. Método para esferas

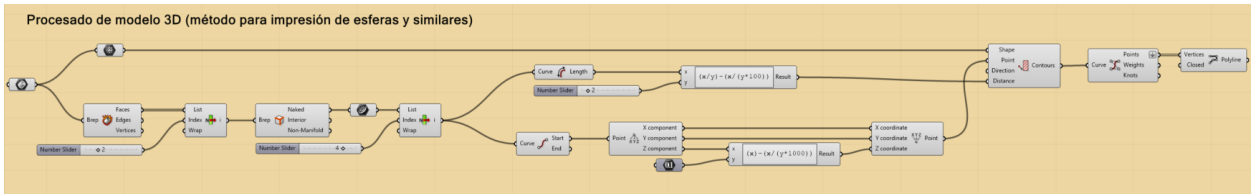


Figura 3.27 Grasshopper. Procesado de modelos esféricos

En el apartado anterior hemos visto que es necesario obtener la altura del objeto en Grasshopper de manera numérica pese que estemos utilizando un sólido completo y acotado de Rhino, esto se debe que la información sobre la altura y otras características no queda almacenada en el propio objeto y, por tanto, Grasshopper la desconoce.

Ahora vamos a emplear un método diferente para crear las capas intermedias de una esfera, y será necesario utilizar esta sección para cualquier solido que no tenga unas paredes lisas, más concretamente, para solidos cuya altura en el eje z difiera de la longitud de la curva que une el punto superior con el punto inferior en cualquier lugar de su revolución.

Uno de los hitos que se marcaron al inicio del proyecto fue poder crear una **Vasija de arcilla** con la estación de impresión, este sólido presenta una forma cilíndrica, pero con una pared irregular y que puede variar según el diseño concreto. De manera histórica este tipo de objetos junto a Ánforas, Botijos y multitud de otros recipientes se han fabricado de manera artesanal con arcilla en el barrio de Triana, estando presentes en la mayor parte de la historia andaluza.

Más allá del significado histórico y artístico de este tipo de objetos, se trata de superficies de revolución con cierta complejidad al estar sostenidos por una base menor que el diámetro máximo que se alcanza en la superficie. Para poder imprimirlo de manera correcta requiere que el software de procesado de modelos 3D sea capaz de analizar los objetos en base a sus superficies de revolución y no a sus características paramétricas como en el primer caso.

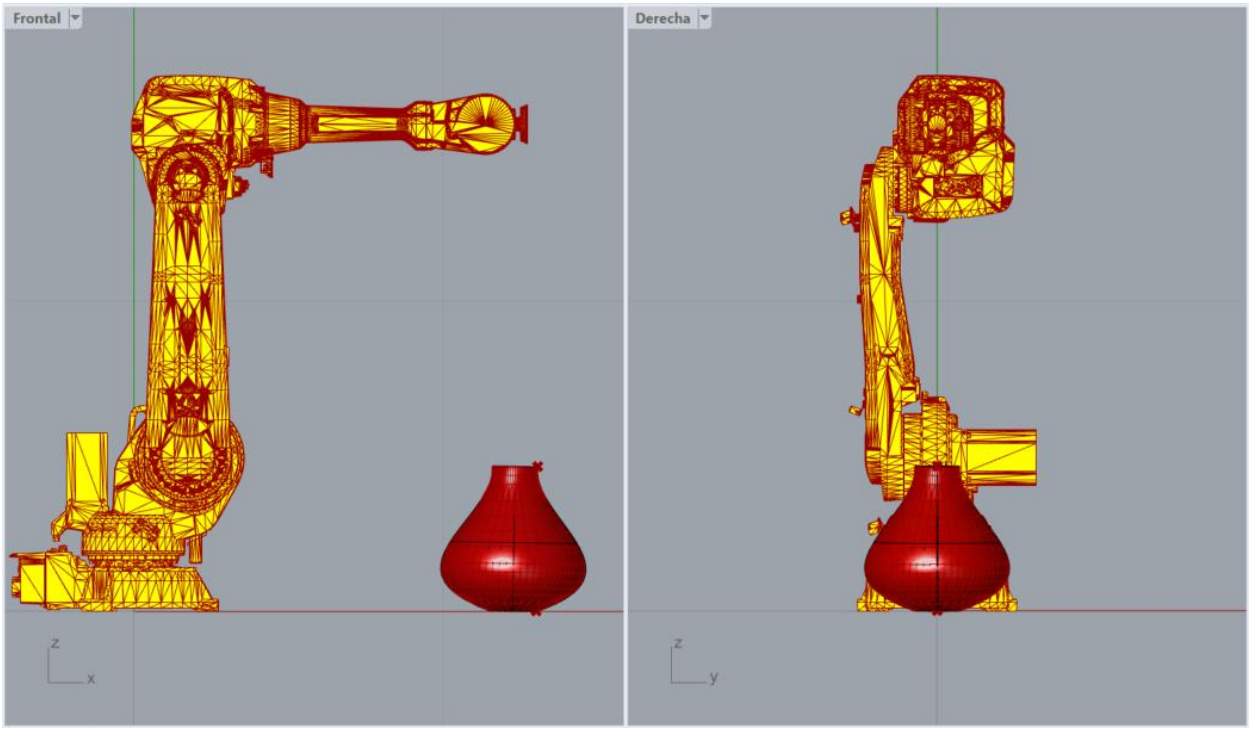


Figura 3.28 Ejemplo de modelo con jarrón en Rhino

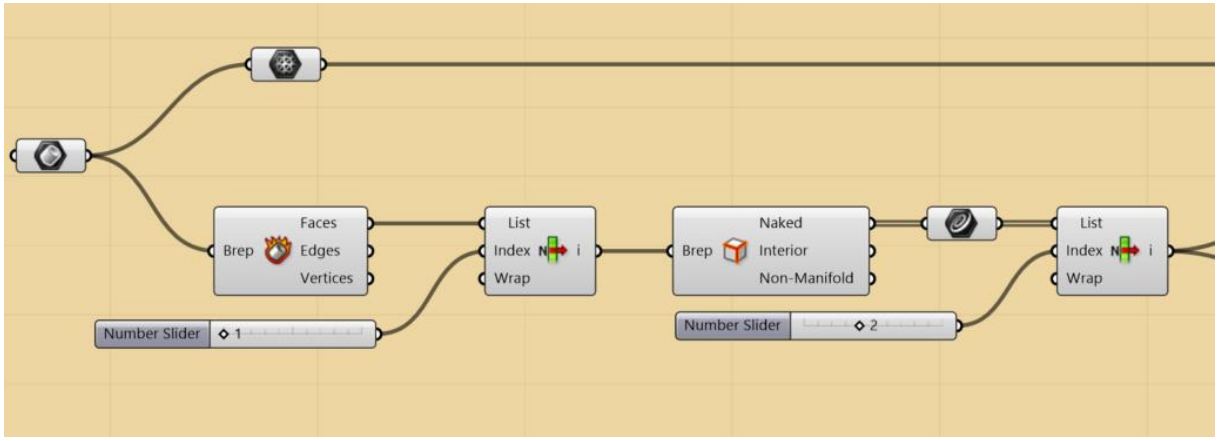


Figura 3.29 Grasshopper. Procesado de modelos esféricos. Zoom 1

La sección comienza obteniendo el modelo desde Rhino en formato Brep, esto es importante ya que nos permite extraer información de los bordes y superficies que lo conforman por separado, esto lo realizamos con el bloque Deconstruct Brep, a la derecha, obteniendo las Caras (o superficies), Bordes y Vértices que forman el modelo.

En nuestro modelo solo existe una superficie que crea al objeto ya que se trata de un sólido de revolución, esto implica que es simétrico respecto al eje de revolución (el eje Z en nuestro diseño). Si se tratase de modelos más complejos con varias superficies aparecerían aquí y con el bloque de lista y el Slider podríamos seleccionar la que deseemos.

Una vez hemos obtenido la superficie por separado utilizaremos el bloque Brep Edges para obtener los bordes de la superficie, marcados en verde en la Figura 3.30 a la derecha. Los bordes de la superficie están compuestos por 3 curvas, los anillos superior e inferior y la curva que se revolucionó.

Los bloques a la izquierda de la Figura 3.29 permiten seleccionar que curva de las extraídas queremos utilizar. Necesitamos una curva que recorra la figura completa por lo que seleccionamos la curva de revolución

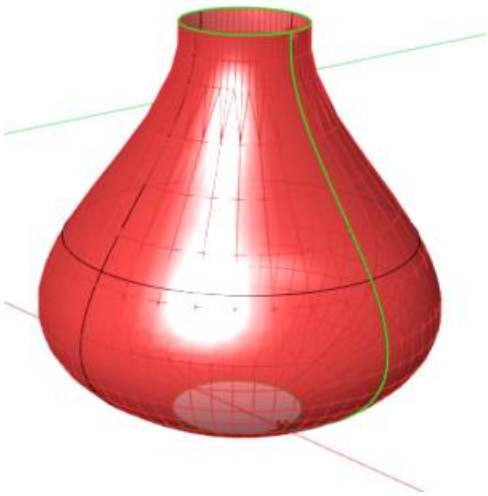


Figura 3.30 Modelo 3D. Superficie de revolución

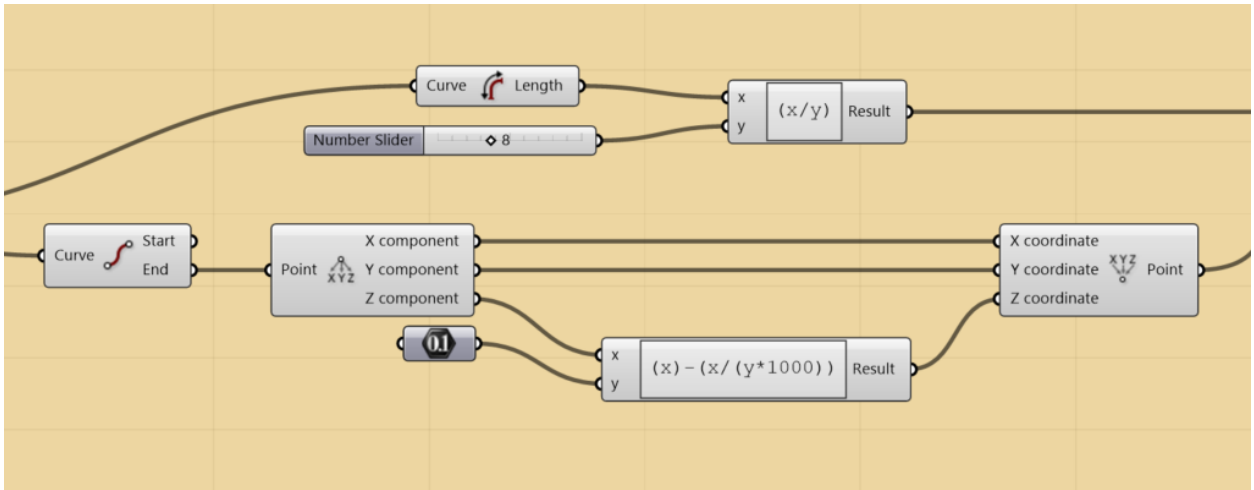


Figura 3.31 Grasshopper. Procesado de modelos esféricos. Zoom 2

Se obtienen 4 curvas del bloque Brep Edges, 2 curvas tipo arco (Grasshopper nombre así a las curvas circulares) y 2 curvas planas (ya que sus puntos inicial y final no están conectados). Las dos curvas arco viene de los bordes superiores e inferiores de la pieza y la doble curva plana es en realidad la misma, pero debido a que se ha generado la superficie con una curva de revolución existe técnicamente un inicio y un final.

La curva de revolución se conecta al bloque Curve Length para obtener su longitud y así poder calcular la distancia a la que se encontrarán las capa. En nuestra sección podemos elegir el número de capas que deseamos tener en el objeto, pero podría modificarse para introducir la distancia entre capa de manera numérica.

De igual forma que en la sección anterior debemos modificar ligeramente el punto final de la curva para que pertenezca al sólido, esto lo hacemos primero extrayendo los puntos inicial y final, y modificando la componente Z del punto final de la curva (parte superior).

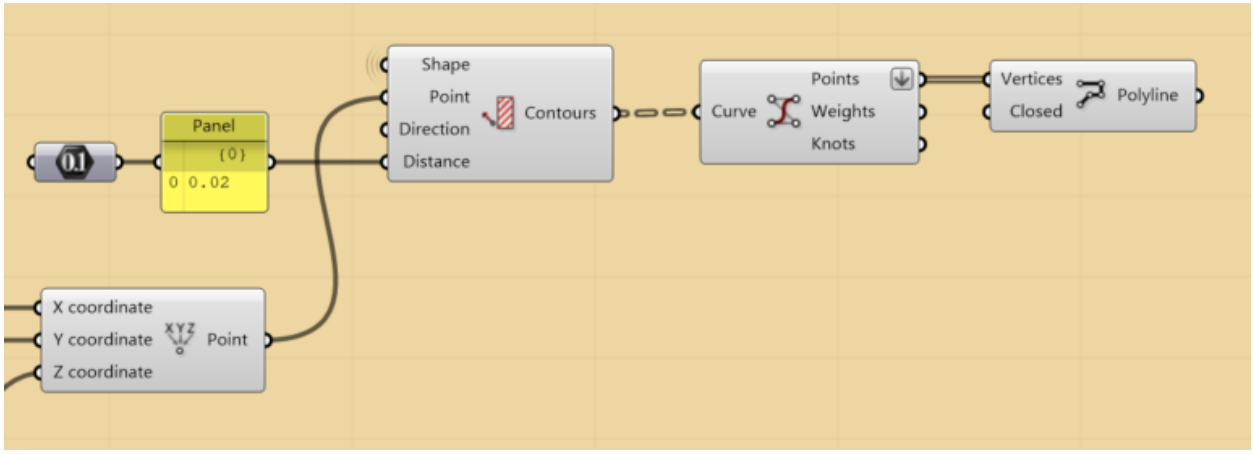


Figura 3.32 Grasshopper. Procesado de modelos esféricos. Zoom 3

Ahora, para mostrar el **modelo laminado**, se ha establecido una altura de capa de 0.02 metros o 20 milímetros. El valor de esta altura depende del diámetro del extrusor utilizado. En nuestra instalación, el extrusor ofrece opciones de diámetro de 4, 6 y 8 milímetros. Sin embargo, hemos seleccionado un valor de 20 milímetros para garantizar una visualización clara de las capas creadas.

El bloque Contorno es responsable de laminar la pieza, utilizando como punto de inicio el punto de la capa superior modificado y la distancia deseada. Las capas resultantes se pueden apreciar en la Figura 3.33. Estas capas deben convertirse en secuencias de puntos para que el control RRC del brazo pueda procesarlas, y los puntos resultantes se presentan en la Figura 3.34.

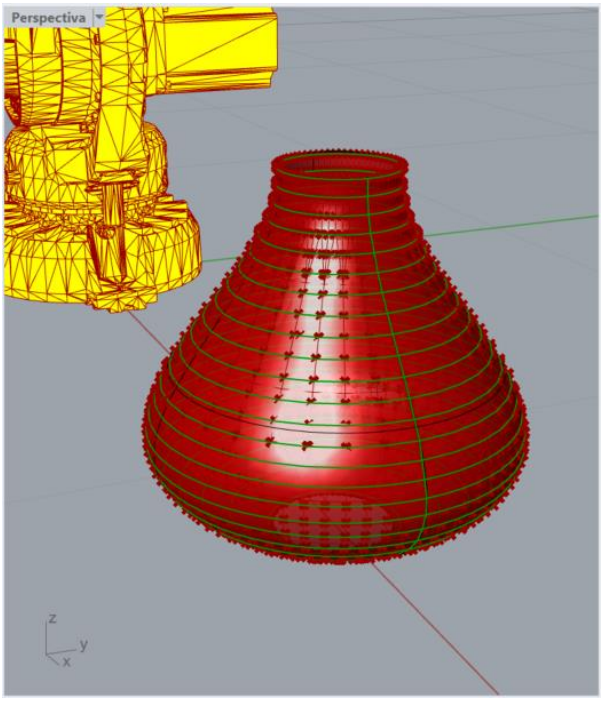


Figura 3.33 Procesado de modelos esféricos. Desplazamientos del cabezal de impresión

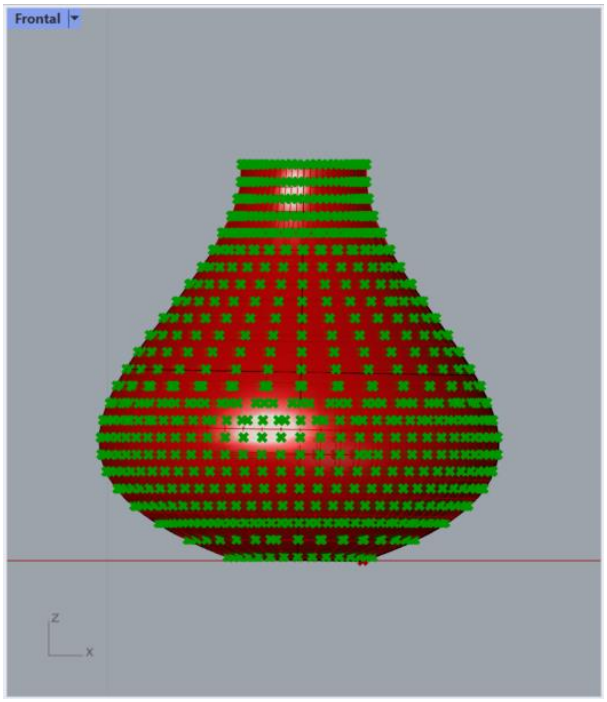


Figura 3.34 Procesado de modelos esféricos. Secuencia de puntos de cada capa

3.4.4. Procesado de puntos y envío a RobotStudio

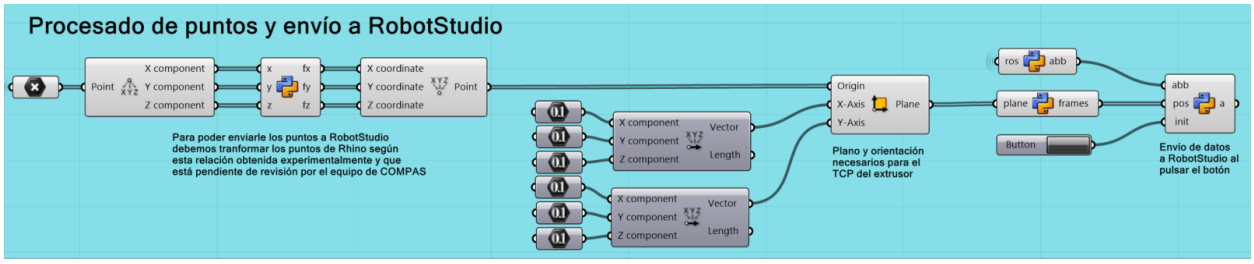


Figura 3.35 Grasshopper. Envío a RobotStudio

La última sección del programa tiene como objetivo enviar los puntos a controlador IRC5 o a la simulación de RobotStudio. Los puntos obtenidos desde las secciones anterior necesitan ser adaptados al sistema de referencia que utilizar RobotStudio y que varía del utilizado por Rhino y Grasshopper.

Para solucionar esto se ha desarrollado una transformada de manera que los puntos que ubiquemos en el espacio de trabajo de Rhino aparezcan en la misma posición relativa al brazo robótico en la estación de RobotStudio, pese a que se ha tratado de minimizar la necesidad de implementar una transformación debido a que puede generar imprecisiones en la impresión y es constituye una variable extra a la hora de realizar modificaciones a la estación, es fundamental en nuestro caso ya que los puntos originales de Grasshopper al ser enviados a RobotStudio tenían una gran desviación. También se contactó con los desarrolladores de COMPAS FAB y el CONTROL RRC para obtener más información al respecto y lo único que pudieron indicar es que se trata de problema conocido y en desarrollo ya que se desconoce la causa exacta de este cambio de coordenadas.

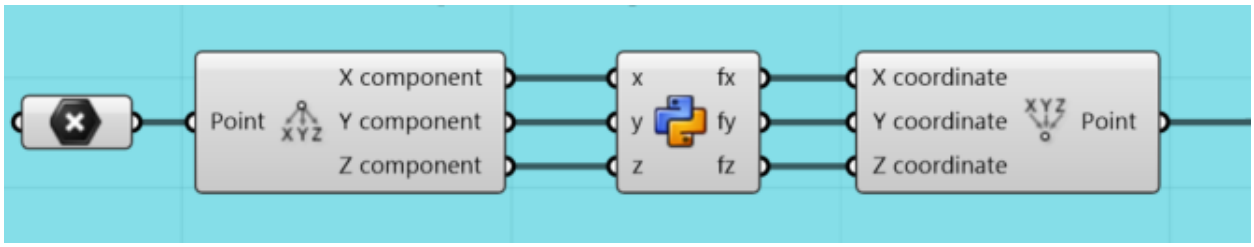


Figura 3.36 Grasshopper. Envío de puntos. Zoom 1

La transformada consiste, primero, en un cambio de escala para las tres componentes ya que Grasshopper opera en m y RobotStudio en mm y, segundo, un offset para cada una de las componentes. Quedando la siguiente expresión:

$$FX = -430 + 1000 * x$$

$$FY = 500 + 1000 * y$$

$$FZ = -500 + 170 + 1000 * z$$

Los offset necesarios han sido calculados de manera experimental para nuestra estación, en la componente Z se ha añadido un offset extra (170mm) para establecer el nivel Z=0 de Rhino justo a la altura de la mesa de impresión sobre la que trabajará el robot real siendo más cómodo para trabajar con Rhino ya que se puede utilizar el plano XY como base de los sólidos.

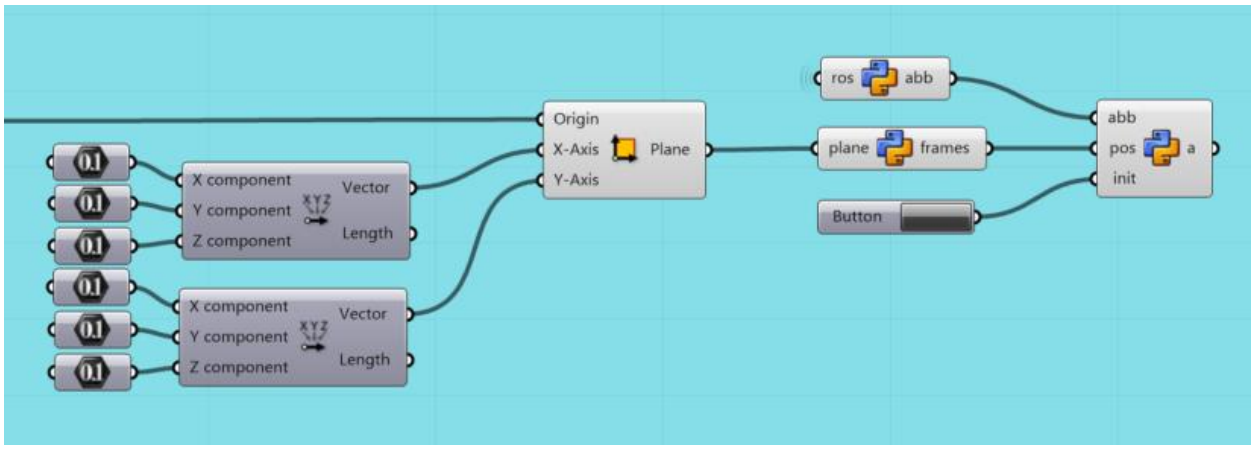


Figura 3.37 Grasshopper. Envío de puntos. Zoom 2

Después de ajustar las coordenadas de los puntos a enviar debemos crear los marcos de referencia que definan completamente como se debe aproximar el cabezal de impresión. Los marcos de referencia o "frame" definen la orientación de los elementos en el entorno de trabajo del robot. Estos marcos se pueden definir únicamente con un vector adicional a las tres coordenadas del punto ya que utiliza la convención de orientación de la "regla de la mano derecha".

La **orientación** con la que queremos que el cabezal alcance la mesa de impresión puede variar en función de la configuración particular, pero de manera general se situará el área de trabajo en la misma dirección que el brazo superior del robot ("enfrente" o con la articulación 5 en su posición inicial) como se muestra en la Figura 3.38. En cambio, en la Figura 3.39 se puede observar una aproximación posible pero no deseada ya que disminuye en gran medida el rango de movilidad del brazo.

El marco necesario para alcanzar la posición con la orientación anterior varía en función de la ubicación de la mesa de impresión, en nuestro caso de sitúa en sentido positivo del eje y, se puede observar en la Figura 3.39 siguiendo el sistema de referencia global de RobotStudio en la esquina inferior izquierda de la imagen.

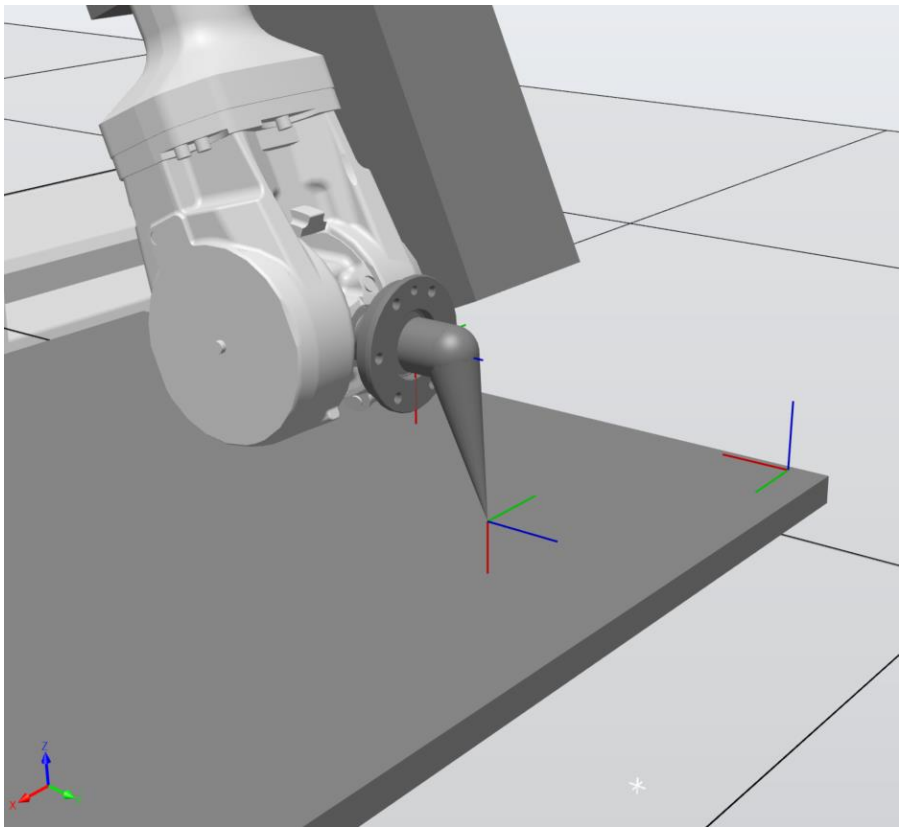


Figura 3.38. Orientación correcta de la herramienta en RobotStudio

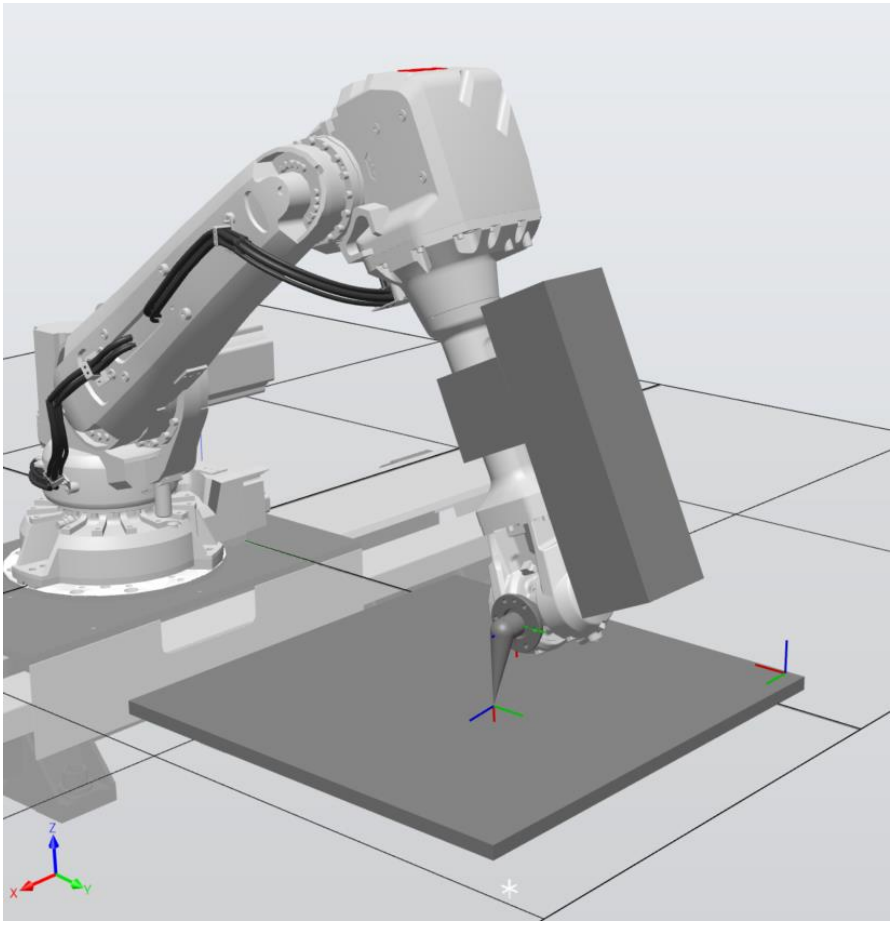


Figura 3.39 Orientación errónea de la herramienta en RobotStudio

En la Figura 3.38 se puede observar la posición de los ejes de la herramienta que necesitamos indicar en Grasshopper para alcanzar la orientación, estos vectores son los que introducimos en el bloque Plano XY:

Vector X (0,0,-1)

Vector Y (0,1,0)

Para finalizar la sección encontramos el bloque que envía estos puntos a ROS y de aquí al controlador IRC5 o RobotStudio. Este bloque es parte de la librería de CF y necesita el control RCC instalado ya sea en RobotStudio o en controlador real. Dentro de la librería se recogen las instrucciones de envío y recepción de frames entre el controlador y ROS utilizando **rrc.GetFrame()** para obtener el frame actual de robot, esto se hace a modo de comprobación para verificar la conexión; y **abb.send** para enviar órdenes de movimiento.

```
import compas_rrc as rrc
import math
instruction = rrc.GetFrame()
timeout = 10
frame = abb.send_and_wait(instruction, timeout)
print frame
print pos

if init:
  for i in pos:
    abb.send(rrc.MoveToFrame(pos,300,rrc.Zone.Z1, rrc.Motion.JOINT))
```

Los argumentos de la instrucción MoveToFrame son:

- **pos,** indica la variable que representa la posición a alcanzar por el brazo.
- **300,** indica la velocidad a la que se moverá el robot en mm/s. El rango permitido mecánicamente es desde 0 a 5000 mm/s, pero no es seguro operar a la máxima velocidad además de que puede deteriorar la impresión. Se puede tomar como un valor máximo seguro 500mm/s.
- **rrc.Zone.Z1,** indica el grado de tolerancia permitida al alcanzar la posición deseada. Tenemos disponible Zone.FINE, con la cual en robot tratará de alcanzar la posición con la mayor precisión posible. En nuestro caso podemos imponer una restricción un poco más ligera con ZoneZ1, alcanzando la posición con 1 mm de margen.
- **rrc.Motion.JOINT,** aquí se define el tipo de movimiento que seguirá el TCP para alcanzar el punto. Motion.JOINT controlará las articulaciones para alcanzar el objetivo sin tener en cuenta la trayectoria que realiza el TCP. también tenemos disponible Motion.LINEAR, que buscará una trayectoria lineal entre la ubicación actual del TCP y la deseada, esta restricción puede provocar que el programa se detenga al no encontrar una trayectoria lineal disponible dentro del rango de movimiento del brazo.

El botón que se encuentra en la sección habilita el envío de los puntos, estos puntos serán enviados como una secuencia en un solo paquete, no pudiendo alterar nada de la pieza una vez activados por lo que es necesario laminar correctamente el modelo hasta estar satisfecho antes de enviar la información ya que, si se trata del controlador IRC5 real, el robot comenzará la secuencia. Como se indicó anteriormente en este capítulo, es necesario revisar los movimientos de impresión y las trayectorias seleccionadas en la simulación de RobotStudio en busca de posibles colisiones del tanque de arcilla ya que el control IRC5 no dispone del paquete Collision Avoidance. Si se sitúa la mesa de impresión en la zona indica es muy poco probable que el tanque pueda colisionar con el brazo o la pieza impresa.

4. USO DE LA ESTACIÓN

Después de haber expuesto detalladamente el robot, el extrusor y la conexión física entre los equipos involucrados, así como de haber abordado en profundidad la programación de Grasshopper, Arduino y RobotStudio, se procede a mostrar la estación completa. En esta etapa discutiremos la configuración y **puesta en marcha** de la Estación de Impresión. Además, se comentan los desafíos y **errores** encontrados a lo largo del proceso con el fin de facilitar el arranque y primeros pasos.

4.1. SECUENCIA DE INICIO

Para asegurar la conexión entre los distintos componentes virtuales, debemos iniciar cada programa siguiendo un orden concreto.

El primer programa que debemos abrir es **Docker**, la aplicación de escritorio, ya que permitirá cargar y ejecutar el servidor de ROS. Si vamos a tener un equipo informático dedicada a esta aplicación es conveniente establecer a Docker como programa de arranque al inicio, de esta manera siempre se iniciará en segundo plano evitando cualquier problema. Esto se puede hacer en el Menú "Aplicaciones de Inicio" desde el buscador de Windows.

Una vez Docker se está ejecutando podemos cargar desde **VisualStudio** el código del Apartado 3.4.2, para hacer esto basta con seleccionar el archivo que contenga el código, abrirlo, y realizar Clic derecho > Compose Up. Esto creará los contenedores con los nodos de ROS en Docker. Cuando queramos desconectar el sistema, desde Docker podemos "eliminar" esos contenedore o, también, podemos hacer Compose Down desde el mismo archivo de VisualStudio utilizado.

El siguiente paso es abrir **RobotStudio** y cargar la estación que hemos desarrollado en el Apartado 3.1, el siguiente paso dependerá si estamos utilizando el robot real o la simulación.

4.1.1. Configuración para el robot real

El uso del robot IRB400 depende únicamente de su controlador IRC5 por lo que el primer paso es encenderlo situando el interruptor A de la Figura 4.1 en posición vertical.

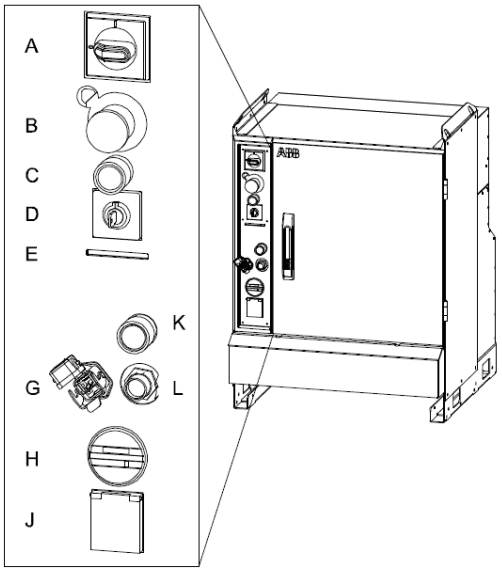


Figura 4.1. IRC5 con controles ampliados.

El proceso de arranque puede llevar varios minutos y se puede observar el en FlexPendant el avance del mismo.

Una vez correctamente iniciado debemos situar el **Seleccionador de modo** (D en la Figura 4.1) en la posición AUTO (posición C) de la Figura 4.2. El modo D es el modo manual, para operar con el FlexPendant. El modo E es el modo manual sin límite de velocidad, aquí quedan restringidas algunas opciones por seguridad.

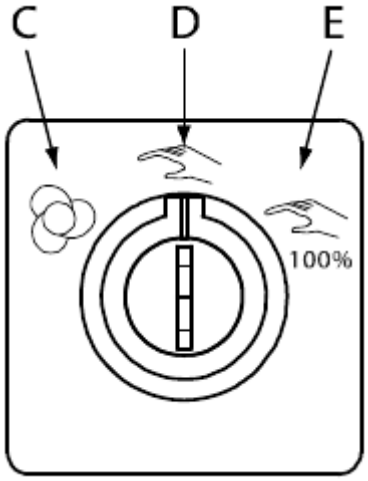


Figura 4.2. Selector de modo de tres opciones presente en el controlador IRC5 del FabLab

Situado el Selector en la posición AUTO aparecerá una notificación de confirmación en el FlexPendant que debemos aceptar.

Para finalizar el arranque del IRC5 debemos pulsar el **botón C** de la Figura 4.1, este es el **activador de Motores** (activa el estado Motores ON en el FlexPendant y RobotStudio), ahora el robot está listo para moverse. Antes de pulsarlo el indicador luminoso del botón debe estar parpadeando y una vez activo debe quedarse fijo (encendido)

POSIBLE ERROR: Al activar los motores, ya sea en el modo manual pulsando el botón de operador del FlexPendant o en modo AUTO con el botón de Motores ON puede **activarse las protecciones eléctricas del FabLab** relativas al robot. Para continuar la operación hay que subir estas protecciones y repetir el proceso con el IRC5. El equipo de mantenimiento está notificado de problema y debe solucionarse pronto.

El último paso es conectar un cable ethernet en el puerto G de la Figura 4.1, con este cable podremos establecer la conexión entre el PC que vaya a ejecutar el programa.

Volviendo a **RobotStudio,** debemos añadir el controlador mediante la conexión con un clic disponible en la pestaña Controlador de la barra de pestañas superior, en la Figura 4.3 se muestra el menú indicado. El IRC5 se mostrará entonces disponible en nuestra estación de RobotStudio pudiendo controlar la ejecución de programas. Lamentablemente, la aplicación no muestra la posición del robot real ni sus movimientos y cualquier movimiento que gestiones fuera del programa a ejecutar ocurrirá en la simulación y no se trasladará al robot.

De igual forma, el tanque de arcilla añadido no lo tendrá en cuenta por lo que es MUY IMPORTANTE realizar la impresión en la simulación para asegurar que no existen colisiones.

Si queremos manipular la posición del robot para llevarlo a un área de trabajo diferente o a la posición de reposo debemos hacerlo en MODO MANUAL con el FlexPendant. Notar que esto interrumpirá la conexión con RobotStudio y cualquier proyecto que esté realizando.

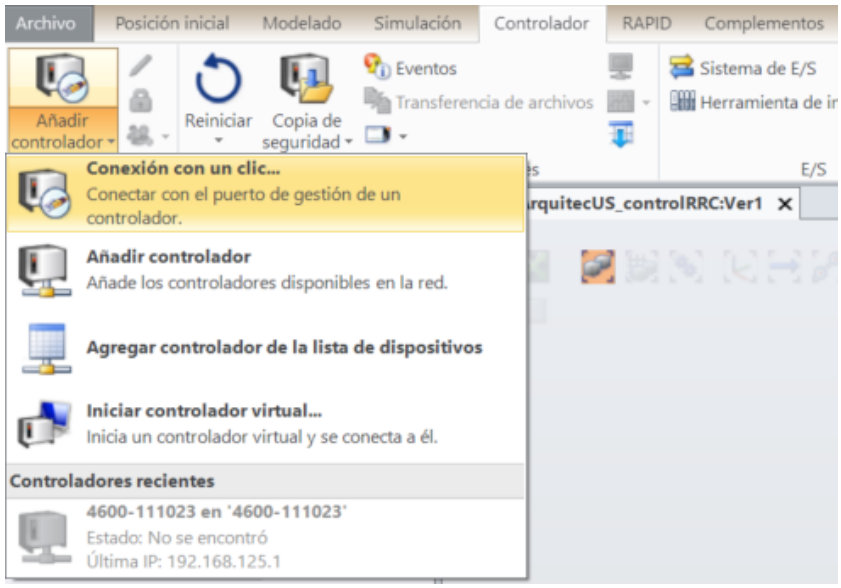


Figura 4.3. RobotStudio. Conexión con un clic

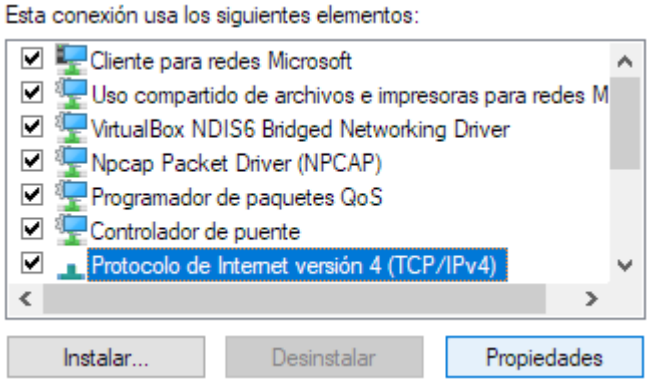
El controlador del FabLab se muestra en la Figura anterior en controladores reciente como referencia, la IP puede variar.

POSIBLE ERROR: Durante la conexión con el ordenador y el IRC5 puede que RobotStudio no detecte al controlador. Ante esto debemos asegurar que el ordenador tiene seleccionada la configuración adecuada para su controlador de red. Para hacer esto entramos en Ajustes > Red e Internet > Estado **> Cambiar las opciones del adaptador.** (Windows 10).

Dentro de este menú localizamos el adaptador de ETHERNET (no el vEthernet (WSL) ya que se trata de un puerto virtual). Hacemos Clic Derecho > Propiedades y seleccionemos:

Dentro del menú de propiedades seleccionamos Obtener una dirección de IP automáticamente.

Ahora podremos consultar la dirección IP a la que está conectada el PC en Estado (Red e Internet)



Una vez esté el controlador iniciado correctamente en RobotStudio podemos ver los controles disponibles en la barra de pestañas superior RAPID. Para iniciar el programa debemos situar el puntero de programa en Main para todas las tareas, esto puede no ser obligatorio si el controlador estaba apagado, pero es una buena práctica ya que si el controlador provenía de una detención puede reanudar el programa anterior sin previo aviso. En la Figura 4.4 se muestra la ubicación.

Finalmente, pulsamos el botón Inicio (situado a la en medio de la barra de RAPID). Ahora el controlador queda a la espera de recibir los puntos objetivo desde Grasshopper y ROS.

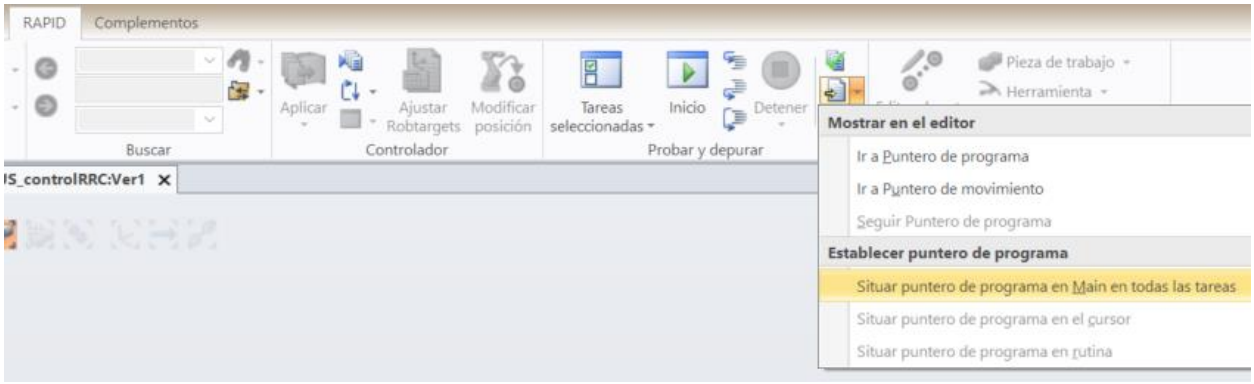


Figura 4.4 RobotStudio. Situar puntero de programa en Main

En la siguiente imagen se muestra la salida que se obtiene por la terminal de RobotStudio en el caso de un inicio exitoso. Los valores de Puerto e IP pueden variar.

Figura 4.5 Log Output de RobotStudio (parte inferior de la estación)

4.1.2. Configuración para la simulación

Este caso es más sencillo ya que únicamente necesitamos abrir RobotStudio, cargar la estación del FabLab creada anteriormente e iniciar la simulación, esto se hace desde la barra de pestañas superior en la pestaña Simulación, donde tenemos disponibles los controles de simulación (Reproducir para iniciar el programa, no la impresión, eso viene después). Aquí podemos pausar la ejecución en cualquier momento y reanudarla posteriormente desde el punto de pausa, cosa que no se garantiza con el robot real.

Figura 4.6 RobotStudio. Controles de simulación

Teniendo listo el robot real o la simulación y el servidor de ROS, podemos abrir Rhino 7 y desde su línea de comandos abrir Grasshopper.

La configuración necesaria en el programa desarrollado para imprimir en Grasshopper variará en función del modelo, pero para comenzar debemos seleccionar el sólido a laminar y enlazarlo al bloque inicial de alguna de las secciones de Procesado de modelos 3D tal y como se explicó en el capítulo anterior.

Al pulsar los bloques de Grasshopper se resalta en verde en Rhino los elementos que ese bloque esté creando, como las capas de impresión, o los puntos del recorrido; de esta manera es más sencillo apreciar los cambios.

Una vez está configurado correctamente podemos realizar el envío de la pieza. Este envío se hará de manera directa y en una única vez. Así el robot comenzará a imprimir el sólido. Durante el proceso de impresión será necesario que el ordenador permanezca conectado en todo momento para poder controlar desde RobotStudio al brazo.

POSIBLE ERROR: Pese a que la plataforma de Rhino y Grasshopper sea bastante estable y no hayamos sufrido abundantes complicaciones a lo largo del proyecto cabe recalcar la importancia de la correcta instalación de la librería COMPAS FAB siguiendo la guía proporcionada por los desarrolladores ya que al utilizar bloques de diferentes versiones puede ocasionarse incompatibilidad, para nosotros la versión que mejor ha funcionado es la 0.19.1.

Estos errores pueden ser difíciles de detectar ya que surgen durante la programación en Grasshopper ya que hacen que los bloques no compilen y parece un error del código. Realmente lo que ocurre es que no puede acceder a la librería o a cierta parte de ella. La solución más fiable es desinstalar la librería e instalar la versión adecuada de nuevo.

---

5. CONCLUSIONES

Finalizando este proyecto de investigación y desarrollo, es el momento para reflexionar sobre los logros alcanzados y las contribuciones significativas que se han realizado para el desarrollo de una Estación de Prototipado con Arcilla en el FabLab de la Universidad de Sevilla.

A lo largo de este trabajo, hemos explorado en detalle los componentes necesarios para poder realizar la Estación, la **adaptación** de equipos y su **interconexión** y el **desarrollo** software realizado. Ahora vamos a discutir los avances más significativos, las aplicaciones de nuestra investigación y delinearemos las posibles direcciones para la continuidad del proyecto.

Como se explica en la Introducción, este proyecto nace de la necesidad de ampliar las capacidades de fabricación del FabLab al introducir un Robot capaz de imprimir en 3D con Arcilla. El punto de partida fue un Robot ABB IRB4600 prácticamente en blanco, únicamente con la configuración mínima para su funcionamiento; y las piezas que conforman el extrusor y tanque de arcilla de WASP3D. Así, **el objetivo superior del trabajo es desarrollar la plataforma capaz de permitir la impresión 3D con Arcilla y la investigación de técnicas constructivas.** Esto engloba la configuración del robot, investigación y solicitud de los componentes, elecciones de software aplicable, ajuste del hardware y programación necesaria para convertir a un equipo de ABB en un sistema capaz de entender y ejecutar las ordenes de impresión.

Detallando objetivos más concretos tenemos:

- **Integración del extrusor al sistema del ABB IRB4600.** Gracias a la investigación realizada sobre el conexionado del brazo y su controlador se ha sido capaz de desarrollar un cable personalizado para esta configuración e implementar un Arduino que actúa de puente entre con IRC5 y el extrusor controlando su funcionamiento. Esto permite el control del extrusor desde el mismo entorno que el robot ABB.
- **Comunicación mediante ROS entre Rhino y el robot.** En la documentación a la que tuvimos acceso se describía el desarrollo de una librería por parte de ETH Zúrich diseñada para establecer la comunicación entre equipos ABB y PC a través de ROS. Sin embargo, esta documentación se enfocaba exclusivamente en la implementación en un entorno de simulación, lo que nos llevó a la necesidad de personalizar y adaptar este proceso para su ejecución en el entorno del robot real.
- **Diseño del programa de procesado de modelos 3D.** Además de realizar la configuración e instalación del hardware necesario para que la estación fuera utilizable, se necesita desarrollar el programa que permita procesar las piezas CAD. Este programa ha sido desarrollado específicamente para esta aplicación siendo una de las pocas ocasiones que se ha controlado desde Grasshopper un robot real con éxito.
- Como último objetivo se tiene en cuenta la necesidad de **recopilar** todo el trabajo realizado en el proyecto y como reproducir parte de estos avances para facilitar la continuidad de la investigación en las sucesivas mejoras que se acometerán, facilitando todo lo posible a las personas que tengan este objetivo su inicio en el proyecto.

Ahora mismo el entorno de impresión está **listo para su puesta en marcha** habiendo superado las pruebas de movimiento y comunicación realizadas. Se completado el desarrollo de software necesario teniendo la aplicación de Grasshopper en funcionamiento, la adaptación del extrusor a la anilla del robot, su cableado con el controlador y control con Arduino.

Cabe destacar algunas de las limitaciones existentes en la plataforma actual que pueden marcar el camino para la continuación del proyecto:

- **Control del extrusor.** Pese a tener el extrusor conectado y respondiendo a las señales del control IRC5 el software de impresión en Grasshopper es bastante simple en este caso, no permitiendo retracciones, cambios de velocidad o desplazamientos fuera del área de impresión como sí se permiten en las impresoras convencionales. La solución a este problema puede estar dentro de Grasshopper, creando

una capa más de información que permita distinguir los movimientos que se van a ejecutar.

- **Control RRC.** Pese a haber mostrado buenos resultados tiene limitaciones por la forma en la que está diseñado, el cliente de RRC en el IRC5 recibe los puntos desde ROS en el PC y ejecuta las órdenes de RAPID necesarias para alcanzarlos. Este proceso lo realiza de manera independiente a cada punto en lugar de generar trayectorias o "Path" que permitan un movimiento más fluido al robot. Este problema es intrínseco a la arquitectura de su diseño y sería necesario una programación adicional en RAPID para modificarlo además del permiso de COMPAS FAB ya que su uso está sujeto a licencia.

El equipo del FabLab de la US tiene a su disposición la herramienta necesaria para comenzar su investigación de impresión con Arcilla. Esta investigación requerirá de un proceso previo de ensayo y error para conocer los ajustes necesarios que introducir a la hora de imprimir con el material como puede ser la velocidad de impresión, la altura de capa requerida o la presión del sistema neumático.

Como punto clave del proyecto se remarca la **integración**, desarrollar un sistema de estas capacidades requiere de un trabajo multidisciplinar en el que se enfoca el Grado de Electrónica, Robótica y Mecatrónica de la ETSI, pese a no ser los primeros en España en realizar una estación capaz de imprimir con Arcilla a través de un brazo robótico, se han investigado los caminos a seguir desde cero en un campo todavía en desarrollo y se han utilizado únicamente los recursos y estudios disponibles en el FabLab y en la US, haciéndola ser una de las pocas universidades españolas y europeas con esta herramienta disponible para la investigación y, gracias a FabLab, accesible a los estudiantes.

REFERENCIAS

[1] FabLab, «Universidad de Sevilla,» [En línea]. Available: https://fablabsevilla.us.es. [Último acceso: 19 Abril 2023].

[2] Stratasys, [En línea]. Available: https://excelencia-tech.com/impresoras-3d-stratasys/impresoras-3d-stratasys-fdm/. [Último acceso: 5 Mayo 2023].

[3] FormLabs, «FormLabs Blog,» [En línea]. Available: https://formlabs.com/blog/3d-printing-materials/. [Último acceso: 20 Abril 2023].

[4] FormLabs, «FormLabs Blog SLS,» [En línea]. Available: https://formlabs.com/blog/what-is-selective-laser-sintering/. [Último acceso: 20 Abril 2023].

[5] Smartfil, «Smart Materials 3d,» [En línea]. Available: https://www.smartmaterials3d.com/en/antibacterial. [Último acceso: 20 Abril 2023].

[6] Prusa3d, «Prusa Research,» Prusa3d, 2020. [En línea]. Available: https://help.prusa3d.com/article/max-volumetric-speed_127176. [Último acceso: 20 Abril 2023].

[7] UNSW Sidney, «UNSW Making,» UNSW, [En línea]. Available: https://www.making.unsw.edu.au/learn/clay_printing/. [Último acceso: 20 Abril 2023].

[8] 3d sourced, «The Complete Clay & Ceramic 3D Printer Buyer's Guide,» 3DSOURCED, 30 Marzo 2023. [En línea]. Available: https://www.3dsourced.com/3d-printers/clay-ceramic-3d-printer-buyers-guide/. [Último acceso: 21 Abril 2023].

[9] B. O'Nell, «Guide to Robotic Arm 3D Printing,» Aniwaa, 16 Diciembre 2021. [En línea]. Available: https://www.aniwaa.com/guide/3d-printers/robotic-arm-3d-printing-guide/. [Último acceso: 21 Abril 2023].

[10] I. Kauppila, «Robotic Arm 3D Printing – The Ultimate Guide,» ALL3DP, 22 Julio 2022. [En línea]. Available: https://all3dp.com/1/robotic-arm-3d-printing-platforms-software/#section-manufacturers-of-robot-3d-printers. [Último acceso: 21 Abril 2023].

[11] Massive Dimension, «Demostration at IMTS 2022,» 16 Septiembre 2022. [En línea]. Available: https://massivedimension.com/blogs/news/massive-dimension-and-abb-additive-manufacturing-demonstration-at-imts-2022. [Último acceso: 16 Abril 2023].

[12] CEAD, «Cead Group,» 2023. [En línea]. Available: https://ceadgroup.com/solutions/robot-based-solutions/. [Último acceso: 30 Abril 2023].

[13] IAAC, «IAAC,» [En línea]. Available: https://iaac.net/project/pylos/. [Último acceso: 5 Mayo 2023].

[14] Gramazio Kohler, «Gramazio Kohler Research,» ETH Zürich, 2023. [En línea]. Available: https://gramaziokohler.arch.ethz.ch. [Último acceso: 1 Mayo 2023].

[15] CEAD, «CEAD Group,» 2022. [En línea]. Available: https://ceadgroup.com/portfolio-items/ribb3d-sustainable-ribbed-concrete-floors-with-3d-printed-formwork/. [Último acceso: 1 Mayo 2023].

[16] Gramazio Kohler, «Gramazio Kohler Research,» ETH zürich, 2022. [En línea]. Available: https://gramaziokohler.arch.ethz.ch/web/e/projekte/393.html. [Último acceso: 1 Mayo 2023].

[17] ABB, «Manual 3HAC032885-005,» ABB, 2022.

[18] ABB, «Manual 3HAC051131-005,» 2019.

[19] ABB, «Manual 3HAC047136-005,» 2015.

[20] 3DWASP, [En línea]. Available: https://www.3dwasp.com. [Último acceso: 2023 Agosto 10].

[21] Gramazio Kohler Research, «Gramazio Kohler Research,» ETH Zurich, 2018. [En línea]. Available: https://gramaziokohler.arch.ethz.ch/web/e/forschung/391.html. [Último acceso: 2023 Agosto 15].

[22] COMPAS FAB, «Robotic Fabrication in Rhino with the COMPAS framework,» Zúrich, 2022.

[23] Gramazio Kolher Research, «GitHub CompasFab,» [En línea]. Available: https://github.com/compas-dev/compas_fab. [Último acceso: 2023 Agosto 20].