ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA EN CONTROL Y REDES INDUSTRIALES

DISEÑO DE UNA INTERFAZ DIGITAL TWIN DE UN BRAZO ROBÓTICO UR3 QUE REALIZA MARCADO PARA CORTE DE SUPERFICIES USANDO EXPERIOR

Trabajo de titulación

Tipo: Propuesta Tecnológica

Presentado para optar al grado académico de:

INGENIERO EN ELECTRÓNICA, CONTROL Y REDES INDUSTRIALES

AUTORES: ALEXIS FERNANDO MARIÑO SALGUERO

SOFÍA JUDITH SÁNCHEZ URRESTA

DIRECTOR: PhD. JORGE LUIS HERNÁNDEZ AMBATO

Riobamba - Ecuador

2021

Nosotros, Alexis Fernando Mariño Salguero y Sofía Judith Sánchez Urresta, declaramos que el presente trabajo de titulación es de nuestra autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autores asumimos la responsabilidad legal y académica de los contenidos de este trabajo de titulación; El patrimonio intelectual pertenece a la Escuela Politécnica de Chimborazo.

Riobamba, 8 de enero de 2021.

Alexis Fernando Mariño Salguero
0605458926

Sofía Judith Sánchez Urresta
1724145600

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA EN CONTROL Y REDES INDUSTRIALES

El Tribunal del trabajo de titulación certifica que: El trabajo de titulación: Tipo: Propuesta Tecnológica, **"DISEÑO DE UNA INTERFAZ DIGITAL TWIN DE UN BRAZO ROBÓTICO UR3 QUE REALIZA MARCADO PARA CORTE DE SUPERFICIES USANDO EXPERIOR"**, realizado por el señor **ALEXIS FERNANDO MARIÑO SALGUERO** y la señorita **SOFÍA JUDITH SÁNCHEZ URRESTA**, ha sido minuciosamente revisado por los Miembros del Tribunal del trabajo de titulación, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal Autoriza su presentación.

NOMBRE	**FIRMA**	**FECHA**
Ing. Jorge Luis Paucar Samaniego.		
PRESIDENTE DEL TRIBUNAL.	_____	_____
Ing. Jorge Luis Hernández Ambato, PhD.		
DIRECTOR DEL TRABAJO DE TITULACIÓN	_____	_____
Ing.		
MIEMBRO DEL TRIBUNAL.	_____	_____

DEDICATORIA

En primer lugar, dedico este trabajo de titulación a mis padres, quienes han sido un pilar fundamental en mi desarrollo como persona y día a día me enseñaron el valor del esfuerzo, dedicación y sobre todo a nunca rendirme, a mis hermanas que siempre me apoyan en cada etapa de mi vida. Finalmente, a todas las personas que cursaron por mi vida politécnica y ahora su esencia es parte de mi personalidad y con gran cariño puedo llamarlos amigos.

Alexis

Este trabajo es dedicado a mis padres que, con amor, dedicación, paciencia, apoyo incondicional, esfuerzo y disciplina, me han forjado como la persona que soy ahora. Son el pilar fundamental en mi vida y mi ejemplo a seguir tanto en mi formación personal como académica y sin ellos este logro hubiera sido imposible. A mis hermanos, quienes me han dado soporte, apoyo y complicidad durante toda mi vida. A Alexis Mariño, por ser una de las mejores personas que ha llegado a mi vida, por brindarme una amistad sincera, por todos los buenos momentos pasados, por alegrarse de mis logros, por estar para mí siempre y por abrirme las puertas de su casa junto a sus padres, Pilar y Luis, ayudándome a cumplir esta meta lejos de mi hogar. A mis amigos, en especial a Kathy, quienes han sido un cimiento para poder culminar con éxito esta etapa en mi vida. Para todos ustedes, con mucho amor, dedicación y esfuerzo.

Sofia

AGRADECIMIENTO

A mis padres por todo el amor, cariño y apoyo que recibo cada día.

A mis hermanas Jessica y Elizabeth que siempre están apoyándome y guiándome en mi camino.

A mi amiga Sofia por todo el esfuerzo, dedicación y sobre todo gran calidad humana que ha demostrado a lo largo de nuestra carrera universitaria y en el desarrollo de este trabajo de titulación .

A la Corporación Ecuatoriana para el Desarrollo de la Investigación y Academia- CEDIA por su contribución en innovación, a través de los proyectos CEPRA, en especial el proyecto CEPRA-XIV-2020-08-RVA "Tecnologías Inmersivas Multi-Usuario Orientadas a Sistemas Sinérgicos de Enseñanza-Aprendizaje "; también a la Escuela Superior Politécnica de Chimborazo - ESPOCH y al Grupo de Investigación GITEA, por el apoyo para el desarrollo de este trabajo.

Alexis

A mis padres, Carlos y Judith, por su amor, por su paciencia, por su esfuerzo, por su apoyo incondicional, por sus invaluables enseñanzas, por la confianza depositada en mí y por su constancia durante todo este proceso.

A mi compañero y amigo, Alexis, por la amistad y cariño sinceros brindados, por su apoyo, por su esfuerzo y trabajo constantes para la culminación de esta tesis.

A la Corporación Ecuatoriana para el Desarrollo de la Investigación y Academia- CEDIA por su contribución en innovación, a través de los proyectos CEPRA, en especial el proyecto CEPRA-XIV-2020-08-RVA "Tecnologías Inmersivas Multi-Usuario Orientadas a Sistemas Sinérgicos de Enseñanza-Aprendizaje "; también a la Escuela Superior Politécnica de Chimborazo - ESPOCH y al Grupo de Investigación GITEA, por el apoyo para el desarrollo de este trabajo.

Sofia

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ECUACIONES

ÍNDICE DE ANEXOS

ÍNDICE DE ABREVIATURAS

MBSE: Model Based Systems Engineering

HCMI: human computer machine interaction.

TCP/IP: Transmission Control Protocol/Internet Protocol.

MBSE: Model Based Systems Engineering.

PDM: Product Data Management.

PLM: Product Lifecycle Management.

SCADA: Supervisory Control and Data Acquisition.

IGU: Interfaz Gráfica de Usuario.

PLC: Controlador Lógico Programable.

WCS: Web Coverage Service.

WMS: Web Map Service.

OPC: OLE for Process Control.

PIP: Package Installer for Python.

UDP: User Datagram Protocol.

RESUMEN

El presente trabajo de titulación tuvo como objetivo el diseño de una interfaz *Digital Twin* de un brazo robótico UR3 que realiza el marcado para corte de superficies en el contexto de la producción flexible. La interfaz se logró a través de una interacción Humano-Computador-Máquina (HCMI). El proceso se desarrolló inicialmente en PolyScope con la adquisición de piezas rectangulares como ordenes de producción personalizadas, luego mediante un algoritmo de *cut and packing*, en este caso usando el algoritmo de guillotina programado en Python, se organizó y ubicó las piezas en la mejor disposición basado en el tamaño de la plancha la cual es el recurso disponible, la trasmisión y recepción de datos entre PolyScope y Python se basó en el protocolo TCP/IP. A continuación, el diseño del *Digital Twin* se lo realizó dentro del software Experior donde se usó el brazo robótico UR3 proporcionado por la misma compañía y se modeló el entorno en el que trabaja este. Los datos del entrono real fueron recabados por una adaptación en el efector final del brazo robótico UR3 llamada *Wrist Camera* y enviados hacia Experior usando protocolo MODBUS para que de esta manera el entorno virtual refleje una réplica exacta del entorno real. Finalmente, se realizó una implementación de todo el sistema de forma virtual usando Ursim como una copia exacta del brazo robótico UR3 distribuido por Universal Robots. Para la evaluación del sistema se obtuvo un retardo medio de 3.64 ms considerando que el sistema se puso a prueba dentro de un entorno virtual y consiguiendo una fidelidad de movimiento de 99,758% en relación con el brazo robótico UR3 y su *Digital Twin.*

Palabras Claves: <AUTOMATIZACION DE PROCESOS ROBÓTICOS>, <DESARROLLO DE INTERFACES>, <FÁBRICAS FLEXIBLES>, <GEMELOS DIGITALES>, <BRAZO ROBÓTICO UR3>, <WRIST CAMERA (HARDWARE)>, <EXPERIOR (SOFTWARE), <ALGORITMO GUILLOTINA>.

ABSTRACT

Keywords: <POULTRY FARMING>,

INTRODUCCIÓN

La industria, en especial la manufacturera, en la actualidad enfoca su desarrollo hacia la cuarta etapa de industrialización. Esta etapa conocida como *Industria 4.0* se define como una combinación de tecnologías modernas (la digitalización, internet de las cosas, los avances en inteligencia artificial, la gestión de Big Data y las fábricas inteligentes) y enfoques metodológicos novedosos para la resolución de desafíos actuales y proyectados hacia el futuro (Rosen et al. 2015). El paradigma de la Industria 4.0 está formado por tres factores mutuamente interconectados: los sistemas ciberfísicos, el Internet de las cosas y la nube (Zhong et al. 2017). Al tener acceso a estos elementos se pueden modelar, mediante ordenador, espacios virtuales tridimensionales los cuales son reales o imaginarios con los que el usuario puede interactuar (Carmen et al. 2007). La interacción de estos debe ser lo más natural posible, por esto se usa medios electrónicos que permiten al usuario la manipulación del ambiente. Los dispositivos pueden brindar una experiencia visual (no inmersiva) o multisensorial (inmersiva) de una simulación computarizada en tiempo casi real. Las interfaces y simulaciones estrechamente vinculadas con el espacio de trabajo son un claro ejemplo de un ambiente virtual no inmersivo (Matteo y Coto 2000). Los beneficios al utilizar este tipo de ambientes son: la libertad y amplitud de movimiento en la escena virtualmente generada, la visualización, retroalimentación táctil, los detalles y escalabilidad. En el caso de la manufactura, el tener la capacidad de analizar los objetos o sistemas en escala real, faculta al operador observar el espacio de trabajo para poder tomar decisiones y realizar cambios acertados (Enríquez et al. 2017).

Estas innovaciones de fabricación y servicio basadas en ambientes inmersivos y no inmersivos son tendencias y desafíos inevitables para las industrias manufactureras ya que, hoy en día, las cambiantes demandas del mercado, los lotes pequeños de productos personalizados producidos en masa con requerimientos de entrega rápida se consideran uno de los principales desafíos que se presentan ante sus sistemas de producción y fabricación (Wallace J. Hopp 2011). Este cambio requiere la utilización de herramientas informáticas predictivas inteligentes, de modo que los datos puedan procesarse sistemáticamente en información para explicar las incertidumbres y, por lo tanto, tomar decisiones más informadas y acertadas al momento de la producción (Lee, Kao y Yang 2014). Ya que, los futuros sistemas de fabricación necesitarán ser más autónomos estas herramientas inteligentes deberán ejecutar tareas de alto nivel sin control humano, capaces de decidir entre un conjunto de alternativas para tomar decisiones y ejecutar habilidades. Para que esto suceda los sistemas autónomos necesitan tener acceso a modelos de simulación realistas del estado actual del proceso y como este interactúa con su entorno y el mundo real así llamado Digital Twin (Rosen et al. 2015).

En una fábrica que pertenece a la Industria 4.0, las máquinas están conectadas como una comunidad colaborativa, por lo que los resultados de los procesos de producción se pueden analizar manualmente o con ayuda de simulaciones. Por lo tanto, para su mejora se requiere de investigaciones en adquisición de datos digitales y automatizados y del concepto de Digital Twin (Lee, Kao y Yang 2014). Así, el proceso de producción permitirá una interacción ideal del sistema de producción con su equivalente digital con un retraso mínimo entre el momento de la adquisición de datos y la creación de la Digital Twin. Como consecuencia un sistema de producción-cibernética-física puede ser generada y así asegurar una concordancia máxima del proceso ciberfísico con su modelo de la vida real convirtiendo a un Digital Twin en un componente básico en la industria 4.0 (Koulamas y Kalogeras 2018).

En una investigación, previa a esta, se propuso una interacción humana-computadora-máquina para la supervisión de aplicaciones de robótica. La arquitectura que se había propuesto era reactiva a los cambios de producción, ya que la lógica de control del robot se adaptaba automáticamente para cumplir con la nueva orden de producción y con los recursos disponibles. Además, una simulación apoyó al operador en la supervisión del proceso. Sin embargo, esa arquitectura no reaccionó a los cambios ambientales, ya que la simulación no utilizó ninguna forma de intercambio de datos pata automatizar la planta física y el modelo digital (Daniel et al. 2018). En este trabajo, se propone un sistema virtual no inmersivo (Digital Twin) de un brazo robótico UR3 para apoyar la actividad de supervisión del operador en el contexto de la producción flexible, además el uso de una Wrist Camera que permite que nuestra arquitectura se autoadapte tanto a la producción como a los cambios ambientales.

El trabajo de investigación está estructurado de la siguiente manera: el Capítulo 1 muestra el concepto de Digital Twin y el marco teórico necesario para la realización de nuestra interfaz. En el Capítulo 2 se describe la metodología utilizada para implementar el ambiente virtual no inmersivo y su prueba en un estudio de caso de laboratorio. Los resultados obtenidos se discuten en detalle en el Capítulo 3 y, finalmente, en el Capítulo 4 se presenta las conclusiones y recomendaciones.

ANTECEDENTES

El concepto de gemelo digital aparece en el año 2002 en la universidad de Michigan en una presentación realizada por Michael Grieves que trataba sobre el ciclo de vida de un producto. En esta se presentó los tres elementos que definen a este concepto: mundo real, mundo virtual y el enlace entre estos (Salvador 2017).

Ya que el gemelo digital está enfocado a ayudarnos a mejorar la productividad y la calidad, a reducir costos y tiempo de producción, y a aumentar la seguridad, este concepto ha sido aplicado en varios ámbitos.

En el 2011 en el artículo "Reengineering Aircraft Structural Life Prediction Using a Digital Twin" de la revista "International Journal of Aerospace Engineering" se propuso la reingeniería del proceso de predicción de la vida estructural de una aeronave utilizando para esto un gemelo digital el cual permitió predecir la vida de la estructura de la aeronave y asegurar la integridad de esta (Tuegel 2012). El uso de un gemelo digital en el modelamiento de la estructura de un avión siguió siendo investigada en el año 2012. Esta vez el gemelo digital fue utilizado en todo el proceso de ensamblaje de la estructura para así poder evaluar la capacidad de la aeronave para cumplir los requisitos adecuados para su vuelo. Eric Tuegel publicó los resultados de la investigación "The Airframe Digital Twin: Some Challenges to Realization". En el 2013 este tipo de sistemas no solo fue utilizado en la fabricación y modelamiento de estructuras para aeronaves sino también en el sector manufacturero. El sector industrial estaba empezando una gran transformación, la fabricación predictiva, como Jay Lee explica en su artículo "Recent advances and trends in predictive manufacturing systems in big data environment". Se adoptaron tecnologías emergentes como sistemas ciberfísicos o gemelos digitales que mejoraron la eficiencia y productividad. Lee estableció que se requiere el enfoque y las herramientas adecuadas para convertir los datos en información útil y procesable (Lee et al. 2013).

A pesar de que en el 2010 el concepto de industria 4.0 ya había aparecido, el mismo tomó varios años para que algunos investigadores hablaran sobre el tema, su importancia y como un gemelo digital puede complementar este tipo de industrias. En 2015 la industria 4.0 empezó su expansión a nivel mundial. Roland público "About The Importance of Autonomy and Digital Twins for the Future of Manufacturing" donde explicó que una fábrica para poder responder rápidamente a eventos inesperados necesita sistemas mucho más autónomos los cuales puedan conocer su comportamiento en el proceso y su interacción con el mundo real para así tener la capacidad de tomar decisiones acertadas para mejorar el proceso de producción (Rosen et al. 2015). Paralela a esta investigación también el departamento de Ingenieria Mecanica de la Universidad Politecnica de

Madrid en España dio a conocer su trabajo "Product Avatar as Digital Counterpart of a Physical Individual Product: Literature Review and Implications in an Aircraft" donde al contrario de investigaciones previas esta se enfocó en un gemelo digital o un avatar para los productos que fabrican y la relevancia de tener este equivalente digital al permitir definir, simular, predecir, optimizar y verificar el producto a lo largo de su ciclo de vida (Ríos et al. 2015).

A partir del 2016 el estudio de los gemelos digitales desde el punto de vista de los sistemas ciberfísicos creció en conjunto con la industria 4.0 y el querer predecir el comportamiento de un proceso productivo. Entre algunas de las investigaciones más destacadas a partir de este año está "From simulation to experimentable digital twins: Simulation-based development and operation of complex technical systems" en la cual, Schluse Michael, su autor explica dos conceptos importantes. El primero son los bancos de pruebas virtuales que son sistemas complejos que interaccionan con los posibles entornos de trabajo para poder diseñar, programar, controlar y optimizar en simulación antes de la puesta en servicio del sistema real. El segundo concepto que propone es el de gemelo digital, que son sustitutos virtuales de objetos del mundo real es decir son representaciones virtuales con capacidades de comunicación que forman objetos inteligentes que actúan como nodos inteligentes dentro de Internet de cosas y servicios. Este autor propone la combinación de estos dos conceptos generando así un gemelo digital experimental el cual puede actuar como el núcleo de los procesos de producción, permitiendo al usuario visualizar simulaciones detalladas a nivel de sistema (Michael Schluse y Juergen Rossmann [sin fecha]).

Grieves Michael, quien dio por primera vez un concepto de digital twin, publicó su investigación "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems" donde propone que un gemelo digital que vincula el sistema físico con su equivalente virtual puede mitigar problemas que afectan el ciclo de producción abordando las interacciones humanas que conducen a accidentes normales (Grieves y Vickers 2016).

En el 2018 los sistemas ciberfísicos y gemelos digitales se complementaron con el internet de las cosas (IoT) como se explica en "Cyber-Physical Systems and Digital Twins in the Industrial Internet of Things [Cyber-Physical Systems]". La evolución de IoT permite interacciones más complejas entre el usuario, el proceso de producción y el gemelo digital (Koulamas y Kalogeras 2018).

Hoy en día se sigue trabajando sobre el tema ya que las industrias deben generar estrategias para situarse en un nivel óptimo para poder competir a nivel mundial. La innovación de la producción es un elemento clave para lograr el uso óptimo de tiempo y recursos. Con el desarrollo tecnológico e industrial, las empresas han tenido grandes cambios en la forma que realizan sus procesos de producción y con la ayuda de gemelos digitales se mejora la productividad y la calidad, se reduce costos y se aumenta la seguridad por lo que su estudio y desarrollo es muy importante dentro de la industria (Val Román 2012).

OBJETIVOS

Objetivo General:

Desarrollar una interfaz Digital Twin de un brazo robótico UR3 que realiza marcado para corte de superficies usando EXPERIOR

Objetivos Específicos:

- Investigar las tecnologías y estudios realizados referentes a las interfaces Digital Twin y el uso del software EXPERIOR.
- Determinar el hardware y software necesarios para la creación de un Digital Twin aplicado a un brazo robótico UR3 el cual realiza el marcado de piezas para el corte de superficies.
- Diseñar la interfaz Digital Twin que pueda replicar virtualmente el movimiento del brazo robótico UR3 en el marcado de piezas para el corte de superficies.
- Implementar la interfaz Digital Twin en el proceso de marcado de piezas para el corte de superficies que realiza el brazo robótico UR3.
- Evaluar la fidelidad y efectividad de la interfaz Digital Twin al momento de replicar los movimientos del brazo robótico UR3 en el marcado de piezas para el corte de superficies.

CAPÍTULO I

1 MARCO TEÓRICO

En el presente capítulo se realiza una breve descripción acerca de los conceptos de: Digital Twin, robots manipuladores, brazo robótico UR3, máquina virtual Ursim, algoritmos de corte, Experior Python y la metodología utilizada para la investigación.

1.1 Digital Twin

1.1.1 Definición

"Digital twin es una frase utilizada para describir una versión computariza o digital de un activo y/o proceso físico" (Parris 2017). Se refiere a una descripción física y funcional de un componente, producto o sistema llevada hacia el mundo digital, que mediante recolección y análisis de datos enlazada un sistema real a un modelo de software que puede mostrar el estado actual, forma, condición, velocidad u otra información de relevancia, creando un sistema de monitoreo, predicción, prueba y validación inteligente (Rosen et al. 2015).

Hoy en día, la simulación y el modelado es un proceso clave en el desarrollo de sistemas el cual permite validar sus propiedades y apoyar en tareas de diseño durante la operación y servicio. Las simulaciones son realizadas para optimizar operaciones y predecir fallas, de este modo se relaciona el mundo físico con entornos virtuales en todas las fases del ciclo de vida del componente, producto o sistema (Rosen et al. 2015).

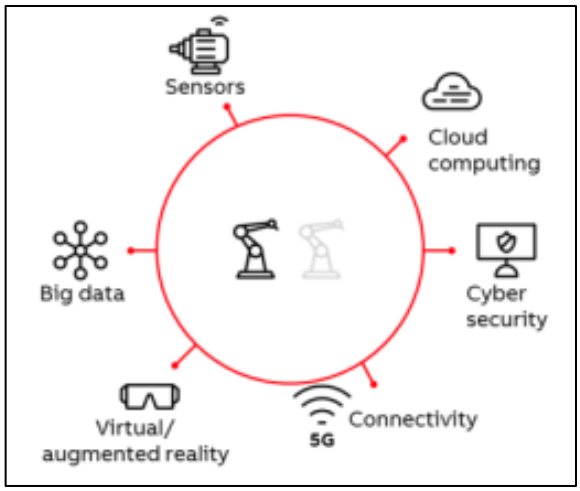


Figura 1-1. Combinación de tecnologías para generar un Digital Twin.

Fuente: Erick, H.; Christopher, G. 2019

A medida que se desarrolla las nuevas tecnologías como lo son la industria 4.0, la virtualización, el internet de las cosas, la big data, la inteligencia artificial, la conectividad, entre otros, los sistemas mecatrónicos necesitan un enfoque de simulación multidominio y multinivel estos enfoques tiene que integrarse al proceso de ingeniería y desarrollo del sistema, así como en todas las fases del ciclo de vida lo que describe el funcionamiento de un *Digital Twin.*

1.1.2 Características

Las características que presenta un *Digital Twin* son:

- Es un conjunto de datos que incluye: datos de operación, ingeniería y descripciones del comportamiento mediante modelos de simulación, dichos modelos deben ser de gran fidelidad para aplicarlos a procesos reales (Rosen et al. 2015).
- Tiene la capacidad de evolucionar junto al sistema real durante todo el periodo de vida y adaptarse a los conocimientos actualmente disponibles del mismo.
- Describe el comportamiento de un bien o servicio, pero también es utilizado para derivar soluciones relevantes para el sistema real obteniendo mejoras en el rendimiento y el servicio de los sistemas, así un *Digital Twin* amplia el concepto de MBSE desde el proceso de ingeniería y fabricación hasta la operación y servicio (Rosen et al. 2015).

1.1.2.1 Aspectos principales

Los sistemas informáticos como PLM, PDM, SCADA almacenan y proporcionan grandes cantidades de información. Por ejemplo, datos de operación, requisitos de los usuarios y datos de aplicaciones CAD que pueden ser utilizados por un *Digital Twin* para crear modelos de datos y simulación. Por lo que, se crean modelos de simulación específicos para cada fase utilizando solo la información requerida y almacenando la información optima y necesaria para la siguiente fase, de esta manera, se convierte en un modelo inteligente capaz de aumentar la productividad y desarrollar nuevas ofertas más eficaces de producción.

Un *Digital Twin* es desarrollado a la par con la construcción del modelo real, así como fue explicado anteriormente, un *Digital Twin* debe estar presente en todo el ciclo de vida de su gemelo real a medida que el sistema se desarrolla trasfiere datos reales del modelo asociado convirtiéndose en parte del producto físico (Rosen et al. 2015).

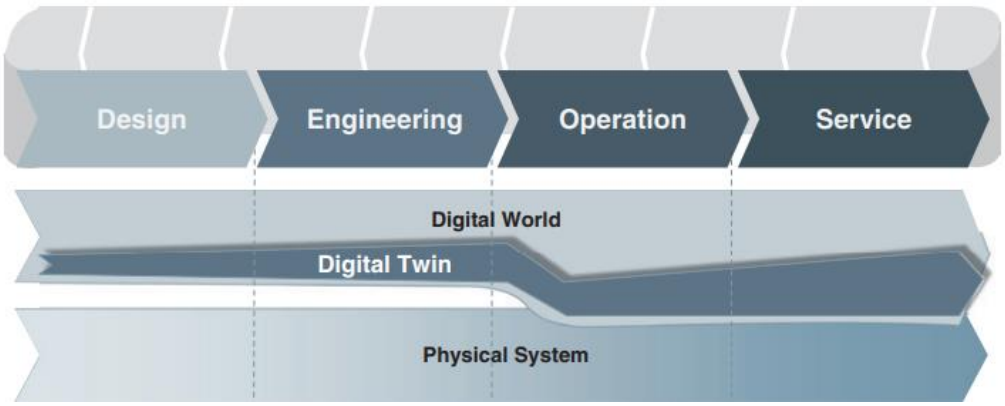


Figura 2-1. Evolución del *Digital Twin* en el ciclo de vida de un sistema.

Fuente: Rosen, R.; Boschert, S. 2016

La figura 2-1 muestra como un *Digital Twin* es parte del mundo digital, nace en la fase de diseño y va aumentando mientas de desarrolla la ingeniería. Para el paso hacia la operación es posible que no se trasfieran todos los datos, sin embargo, recopila más datos mientas se encuentra en la fase de operación y servicio. Una característica importante es que parte de su contenido se convierte en una parte del sistema real (Rosen et al. 2015).

1.1.3 Digital Twin integrado al internet de las cosas

El objetivo principal por el cual el internet de las cosas se encuentra en auge es mejorar el mantenimiento predictivo. Una gran parte de este desarrollo se ha dado con las réplicas virtuales de aparatos, dispositivos y cualquier proceso físico, llamados *Digital Twin.*

Actualmente en el mercado la mayor parte de dispositivos o activos tienen un modelo virtual que demuestra la funcionabilidad del este. Dicho modelo es una réplica estética y funcional, pero con la deficiencia de que este no cambia con el paso del tiempo y cuando el activo real presenta problemas estos no se ven reflejados en su réplica virtual lo que conlleva problemas.

Se presagia que en el año 2021 existirán más de 50 mil millones de activos relacionados mediante el internet de las cosas para así crear un ambiente de intercomunicación entre estos, otras máquinas y los humanos, esto permitirá que se generen datos de todo el proceso que llevan a cabo dejando a criterio la posibilidad de crear replicas vivas de dichos activos o sistemas para evaluar posibles fallos que se podrán suscitar.

La incorporación de un *Digital Twin*, dentro del mantenimiento de una línea de producción, dará la capacidad a los operadores de tomar decisiones sobre dicho proceso, es decir, se puede resolver cuando dar mantenimiento o reemplazar un equipo o sistema dependiendo la información que sea brindada por su gemelo (Parris 2017).

1.1.4 Industria 4.0 y Digital Twin

La industria 4.0 consiste en la interconectividad de todas las partes de la empresa dando lugar a una automatización efectiva y una empresa más inteligente. Es decir, que sea capaz de adaptarse a los medios los recursos y requerimientos de producción para que estos sean aprovechados al máximo.

Por lo tanto, se infiere que la industria 4.0 basa su funcionamiento en la digitalización de la industria y todos los servicios y sistemas que la integran así creando un enlace entre el mundo real y virtual, es decir, se necesitan de todas las tecnologías que hoy en día están en desarrollo incluyendo los procesos de producción, consiguiendo así que las fábricas sean capaces de adaptarse y gestionarse de manera autónoma

Entre las tecnologías con mayor impacto dentro del desarrollo de una interfaz *Digital Twin* se encuentran: el internet de las cosas, por su facultad de conectividad; inteligencia artificial y *machine learning*, desarrollando sistemas autónomos; *big data*, ya que tiene la facultad de procesar grandes cantidades de datos; entre otros. La figura 3-1 muestra la representación de estas tecnologías (Parrott y Warshaw 2017).

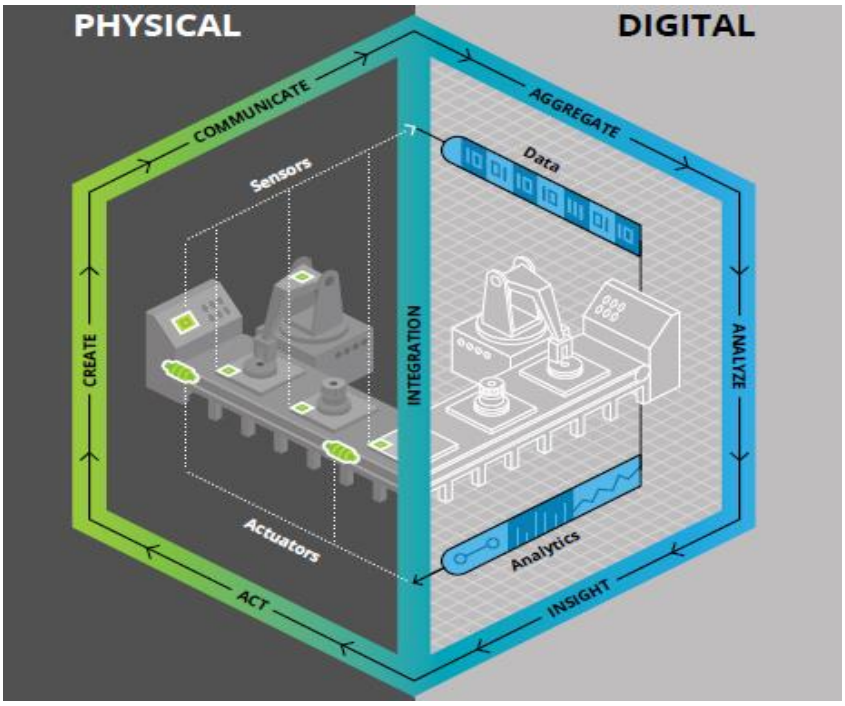


Figura 3-1. Representación de un sistema real interconectado con uno virtual.

Fuente: Parrott, A.; Warshaw, L. 2017

Las contribuciones que genera un *Digital Twin* a la industria son:

- Conseguir un entorno protegido y seguro para experimentar proyectos futuros.
- Examinar los cambios de nuevos escenarios.

- Planificar mantenimientos preventivos.
- Reducir margen de error y fallos.
- Predecir resultados.
- Optimizar el control de los parámetros de producción.
- Abaratamiento del diseño de los productos.

1.2 Robots manipuladores

Esencialmente cuando se habla de robots manipuladores de refiere a brazos articulados, mientas que técnicamente se define "un manipulador industrial convencional es una cadena cinemática abierta formada por un grupo de eslabones o elementos de la cadena interrelacionados mediante articulaciones o pares cinemáticos" (Ollero 2001). Como describe la figura 4-1 las articulaciones dan lugar al movimiento entre los sucesivos eslabones.

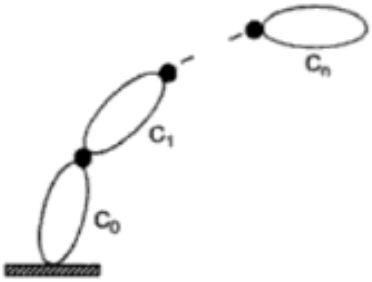


Figura 4-1. Cadena cinemática abierta.

Fuente: Ollero, A. 2001

La configuración de estos robots de caracteriza por tener uno de sus extremos fijo y el otro libre permitiendo así que este pueda moverse para realizar tareas que para el operador son repetitivas o peligrosas, sin embargo, hay que tomar en cuenta que para un correcto funcionamiento y operación serán necesarios los modelos matemáticos, cinemáticos, dinámicos y adicionalmente un análisis teórico de su funcionamiento de acuerdo a sus características físicas, arquitectura, configuración, grados de libertad, tipo de control, etc. (Abdalá y Caberta 2016).

Hoy en día, la industria ha convertido a los robots en un recurso indispensable dentro de sus procesos de producción, generando así una solución óptima y mucho más rentable para varios procesos.

La Federación Internacional de la Robótica estableció una clasificación de las aplicaciones de la Robótica en el sector manufacturero.

- Manipulación en fundición.
- Manipulación en moldeado de plásticos.
- Manipulación en tratamientos térmicos.

- Manipulación en la forja y estampación.
- Soldadura.
- Aplicación de materiales.
- Mecanización.
- Corte.
- Montaje.
- Paletización.
- Formación, enseñanza e investigación.

1.2.1 Robots manipuladores utilizados en corte

El sector industrial manufacturero ha evolucionado y junto a este se requieren de sistemas que sean extremadamente precisos. Los sistemas de corte realizados por robots manipuladores presentan grandes beneficios por algunas de sus cualidades como: capacidad de reprogramación, precisión y exactitud nanométrica, capacidad de integrarse a determinados sistemas y la facilidad para trasportar varias herramientas de corte.

Los métodos de corte más comunes realizados por robots dependen del tipo de material y se clasifican en: oxicorte, plasma, láser y chorro de agua, en todos estos métodos el robot sigue una trayectoria determinada por encima del material a cortar mientas su efector final expulsa el material de corte (Abdalá & Caberta, 2016).

Uno de los sectores industriales con mayor auge en esta tecnología es el automovilístico, citando un caso específico se encuentra la empresa Zhengzhou Yutong Bus Co. Ltd. (Yutong Bus) la cual es una de las compañías constructoras de autobuses con mayor importancia en el mundo. El gigante chino confía en los brazos robóticos Stäubli para la delicada labor del corte. En el año 2011 la empresa empezó a integrar los robots Stäubli RX160L en sus líneas de producción que funcionan a partir de corte por láser. El robot conformado por seis ejes se encuentra instalado en el techo y trabaja cortando puertas, parachoques y otras partes de metal que pasan por una mesa giratoria, produciendo desde los recortes circulares más pequeños hasta las superficies de forma libre 3D más complejas como muestra la figura 5-1 (Ágora 2018).

Figura 5-1. Brazo robótico Stäubli RX160L acoplado a una línea de producción.

Fuente: Ágora, G. 2018.

1.2.2 Brazo robótico UR3

La compañía Universal robots ha destacado en el desarrollo de robots colaborativos entre sus principales diseños se encuentra el brazo botico UR3 el cual mediante sus 6 articulaciones está diseñado para imitar la libertad de movimiento del brazo humano. Sus 360° grados de rotación en cada articulación y su rotación infinita en el efector final ha sido desarrollada para trabajar con en colaboración con operarios humanos o de manera autónoma. Adicionalmente, posee un sistema de seguridad que controla la posición, velocidad y fuerza tanto en las articulaciones como en el efector final (Universal Robots 2015). La siguiente tabla detalla los datos más relevantes del robot.

Tabla 1-1. Parámetros importantes del brazo robot colaborativo UR3

Brazo Robot Colaborativo UR3	
Consumo de energía	Aprox. 100 W para un programa típico
Temperatura ambiente	0-50°C
Grados de libertad	6 articulaciones giratorias
Programación	Interfaz gráfica del usuario PolyScope
Radio de acción de todas las articulaciones	± 360°

Carga útil	3 kg / 6,6 lbs
Ruido	Menos de 60 dB(A)
Certificación IP	IP54
Alimentación	12 V/24 V 600mA continuos
Materiales	Aluminio, plástico de PP, acero
Peso	11.2 kg / 24.7 lbs

Fuente: Universal Robots, 2015

Figura 6-1. Brazo robot colaborativo UR3

Fuente: Universal Robots, 2015

Entre las principales aplicaciones del brazo robótico UR3 se encuentra paletización, pulido, montaje, atornillado y encolado las cuales se caracterizan por tener un estándar de producción uniforme con alta calidad. El robot se instala sobre un espacio de trabajo firme y gracias a la variedad de herramientas que se pueden ubicar en el efector final más su entorno de programación amigable lo convierten en un implemento ideal para tareas de agarre, montaje y colocación de piezas (Universal Robots 2015).

1.2.3 Control del robot UR3

El control de potencia del robot se da dentro de la caja de control la cual administra la energía necesaria para que el robot pueda mover sus articulaciones. Para trabajos en colaboración con otros dispositivos el robot posee puertos e interfaces de comunicación ubicados en la consola de programación (Ferriz 2018).

Los puertos que dispone el robot son:

- 16 entradas digitales.
- 16 salidas digitales.

- 2 entradas analógicas.
- 2 salidas analógicas.

Y comunicación mediante

- TCP/IP 100 Mbit.
- MODBUS TCP.
- PROFINET.
- Ethernet IP.

El control del robot se da mediante el software de programación *PolyScope* el cual está contenido dentro de la consola de programación la misma que es una pantalla táctil de 12” instalada junto a la caja de control.

Figura 7-1. Caja de control y consola de programación del brazo robot UR3

Fuente: Universal Robots, 2015

1.2.3.1 PolyScope

PolyScope está definido como la interfaz gráfica de usuario (IGU). Este es un software desarrollado por la compañía Universal Robots y distribuido junto a los modelos de brazos robóticos. El software permite controlar el robot y realizar programas con una interfaz gráfica sencilla por lo que no es necesario tener alto grado de conocimientos en robótica o programación (Universal Robots 2015) la figura 8-1 muestra la pantalla principal con las funciones que se puede configurar dentro de PolyScope.

Figura 8-1. Interfaz de usuario de robot PolyScope.

Fuente: Universal Robots, 2015

Para poner en marcha al robot. Inicialmente, se ingresa a la interfaz y se da inicio al mismo esto permite liberar los frenos de las juntas y habilita todas funciones disponibles, luego la pantalla de inicio muestra un menú desplegable con las configuraciones del robot.

Entre las funciones básicas se encuentra comandos como: mover, crear punto, esperar, línea entre otros, mientras que la opción avanzada permite crear código estructurado ya que cuenta con bucles como *if, while* y *for* además de funciones para ingresar datos crear variables e incluso desarrollar programas en *Scripts programing*. Todos los cambios realizados en el programa se pueden observar en la pantalla de escritura que muestra la estructura línea por línea (Universal Robots 2015).

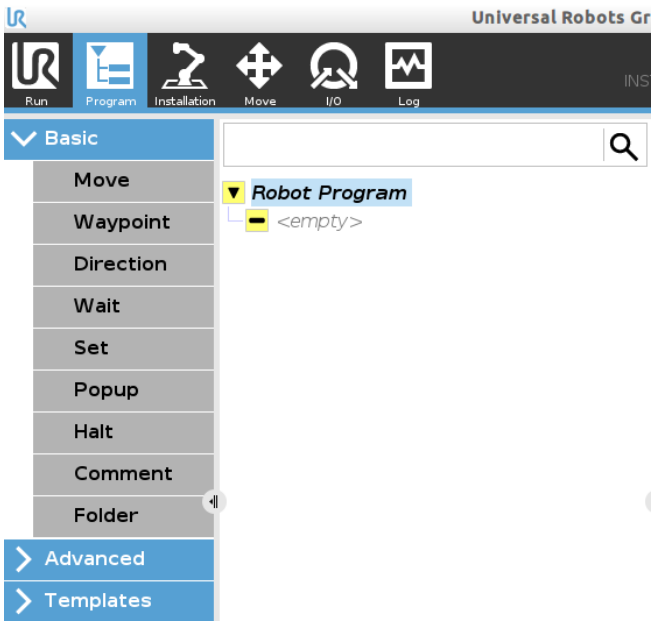


Figura 9-1. Interfaz de PolyScope para programar el robot.

Fuente: Universal Robots, 2015

PolyScope también permite mover el robot sin la necesidad de realizar un programa ya que cuenta con la función mover, la cual desactiva los frenos de las juntas y permite ubicar al robot de forma manual en una posición arbitraria, esto acorta y facilita la programación.

Finalmente, PolyScope cuenta con un software de simulación llamado Ursim el cual replica con exactitud el movimiento del robot real o incluso se puede configurar para que el movimiento sea realizado primero por la simulación y comprobar que no existan fallas en la programación (Universal Robots 2015).

1.2.4 Wrist Camera

Universal Robots posee líneas de investigación que van a la par con el desarrollo de sus productos entre ellas se encuentra la empresa Robotiq la cual es encargada de desarrollar adaptaciones para los brazos robóticos como son: herramientas para el efector final, sensores, grippers y cámaras.

Wrist Camera es una adaptación que se instala entre el brazo botico y el efector final, esta junto al software *Camera Locate* crean un sistema de visión artificial que asiste al robot para que pueda reconocer la posición y rotación de las piezas que se encuentran en su espacio de trabajo del mismo (Universal Robots 2018).

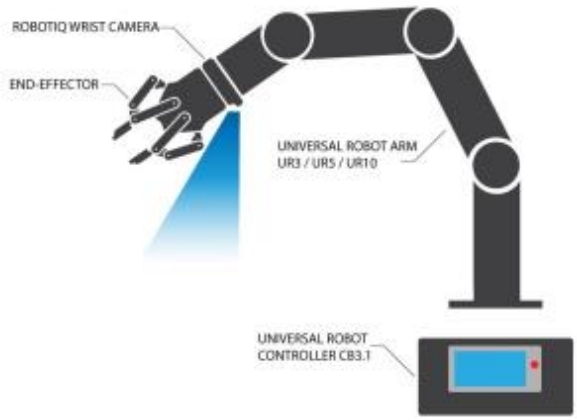


Figura 10-1. *Wrist Camera* ubicada en el brazo robot.

Fuente: Universal Robots, 2018

La interfaz permite configurar el sistema de visión para que pueda reconocer las piezas ubicadas y determine la ubicación y orientación de los objetos automáticamente.

Tabla 2-1. Especificaciones de la Wrist Camera de Robotiq

ESPECIFICACIONES	
Campo visual mínimo (cm)	10 x 7.5
Campo visual máximo (cm)	36 x 27
Tamaño mínimo de la pieza (% de campo visual)	10%
Tamaño máximo de la pieza (% de campo visual)	60%
Iluminación integrada	6 luces LED blancas y difusas
Rango de enfoque	70 mm hasta infinito

Fuente: Universal Robots, 2018

1.2.4.1 Instalación y calibración

La cámara se instala alineando las clavijas de indexación correctamente en el patrón de pernos en la última articulación del robot, luego se coloca la placa de herramienta en la cámara de forma que la espiga quede correcta. Finalmente, se ubica el efector final deseado como muestra la figura 11-1.

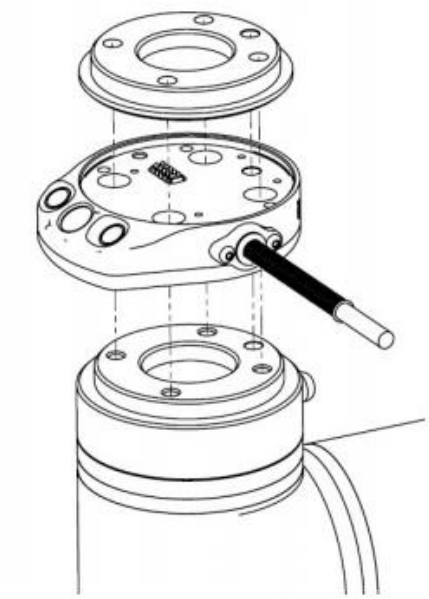


Figura 11-1. Instalación mecánica de la *Wrist Camera.*

Fuente: Universal Robots, 2018

Para un correcto funcionamiento la cámara necesita ser calibrada. Esto se consigue ubicando la placa cuadriculada en el espacio de trabajo del robot. A continuación, el robot se ubica de manera perpendicular a la placa y esta será la posición en la cual el robot tomará las fotos a las piezas, esa posición determinará el campo de visión de la cámara y, por lo tanto, el espacio de trabajo.

Finalmente, se ubica la placa colorida en el espacio de trabajo del robot que permitirá configurara resolución de la cámara (Universal Robots 2018).

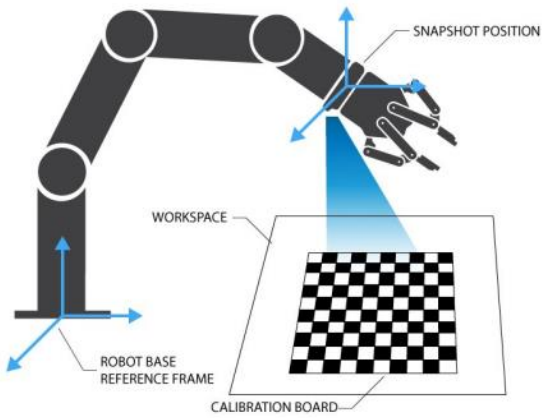


Figura 12-1. Calibración de la *Wrist Camera*.

Fuente: Universal Robots, 2018

1.3 Ursim

Ursim es un software de simulación desarrollado por Universal Robots que proporciona una interfaz idéntica a un brazo robótico tanto de forma física como funcional y consta de simulaciones para los robots UR3, UR5 y UR10. El programa se instala sobre la plataforma Java 6 ya que este software posee una gran variedad de bibliotecas para trabajar en entornos tridimensionales y producir gráficos de simulaciones.

Ursim es desarrollado como un software de acceso libre por lo que es de código abierto así es necesario ponerlo en funcionamiento dentro de Ubuntu, para usuarios que desean abrir Ursim en otro sistema operativo se lo implementa a través de una máquina virtual usando programas como lo son Vmware, VirtualBox, entre otros, como se muestra en la figura 13-1 (Hietanen y tarkastettavaksi 2017).

Ursim es programado mediante el lenguaje Urscript y los datos pueden enviarse hacia o desde el simulador externamente a través del zócalo TCP/IP. Si Ursim es utilizado como máquina virtual dentro de otro sistema operativo es necesario crear un puente de red para poder vincular segmentos de red, transfiriendo los datos de un software a otro en base a la dirección física de destino que se encuentra en cada uno de los paquetes.

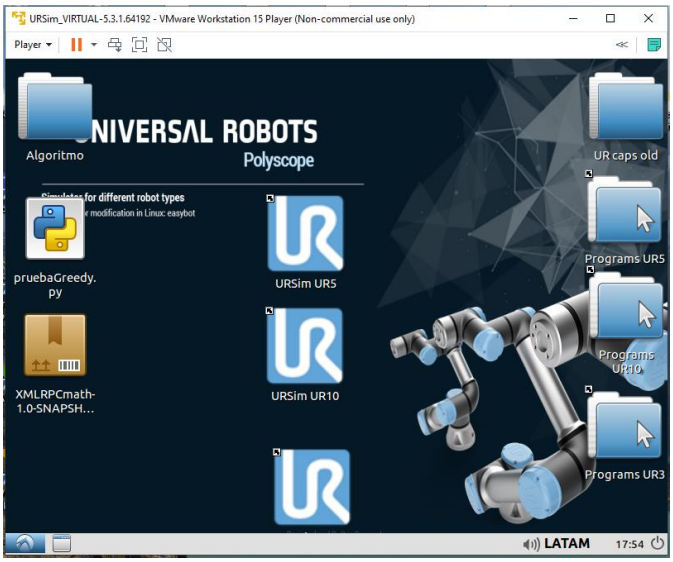


Figura 13-1. Máquina virtual Ursim dentro de Vmware.

Fuente: Robot, U. 2018

El lenguaje Urscript proporciona la función llamada socket_open la cual abre una conexión a un socket previamente creado cuyas entradas son la dirección IP, el puerto, y el nombre del programa al cual se desea conectar. Una vez que se establece la conexión Ursim tiene la capacidad de enviar y recibir datos desde la salida y entrada estándar del programa, ya que estos datos son enrutados al zócalo TCP/IP que es utilizando como medio de comunicación (Zhang 1999).

1.4 Algoritmos de Optimización de Corte

La función principal de los algoritmos de corte es generar patrones que aprovechen de mejor manera la materia prima a ser cortada, determinando la forma y la mejor distribución en que la(s) pieza(s) serán cortadas.

1.4.1 Corte en dos dimensiones

Los algoritmos de corte y empaquetamiento bidimensional son considerados un problema típico en la optimización combinatoria. El objetivo de este es ordenar piezas pequeñas dentro de una pieza de mayor tamaño en caso de piezas regulares como cuadrados y rectángulos los bordes de cada pieza deben ser paralelos a los bordes de la pieza mayor.

En esta clase de algoritmos es necesario realizar una distribución del problema por lo que se tiene en consideración tres factores: Las dimensiones, la orientación y la forma de la pieza a ser cortada. El primer factor, dimensionalidad, está relacionado con el tamaño de las piezas que serán ubicadas para ser cortadas. La orientación hace referencia a si la pieza puede realizar giros para que se ubique de mejor manera, mientras que la forma de las piezas está determinada por si estas son de forma regular o irregular (Breuel 2002).

Existen cuatro principales algoritmos de corte los cuales son: *Shelf Algorithm, Guillotine Algorithm, Maximal Rectangles Algorithm* y *Skyline Algorithm.*

***Shelf Algorithm*:**

Comúnmente conocido como algoritmos de nivel, este es el método más simple que se puede usar para ordenar piezas y producir cortes. Su funcionamiento se basa en una línea que corta a la plancha de lado a lado generando sub-rectángulos dentro de la plancha a ser cortada como se muestra en la figura 14-1. A estos sub-rectángulos se los denomina estantes. A medida que se desarrolla el algoritmo, el área total de la pieza grande está dividida en varios estantes en los cuales las piezas pequeñas se van ubicando de izquierda a derecha. El estante más alto se llama el estante abierto. Ya que todas las piezas se colocan de abajo hacia arriba, el espacio sobre el estante abierto siempre está sin usar. Permitiendo así que la altura del estante abierto se pueda ajustar a la altura de la pieza más alta que sea ubicada en este. Los estantes que se encuentran debajo del estante abierto no tiene la libertar de ajustar su altura por esto se les llama estantes cerrados (Baker y Schwarz 1983).

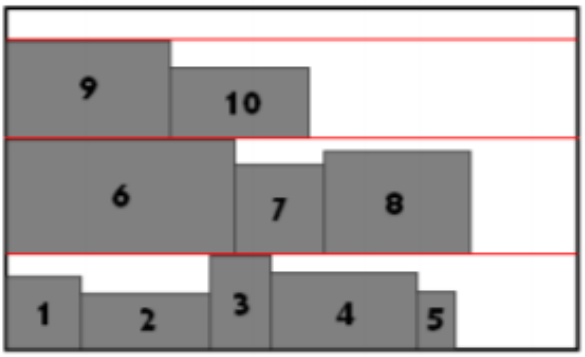


Figura 14-1. Ejemplo de corte producido un Shelf Algorithm.

Fuente: Jukaj, Y. 2010.

Guillotine Algorithm:

La metodología de este algoritmo se basa en los cortes realizados por una guillotina física de ahí su nombre. Principalmente se coloca una pieza en una esquina de la plancha más grande, después se forman dos rectángulos libres donde colocar las demás piezas como muestra la figura 15-1.

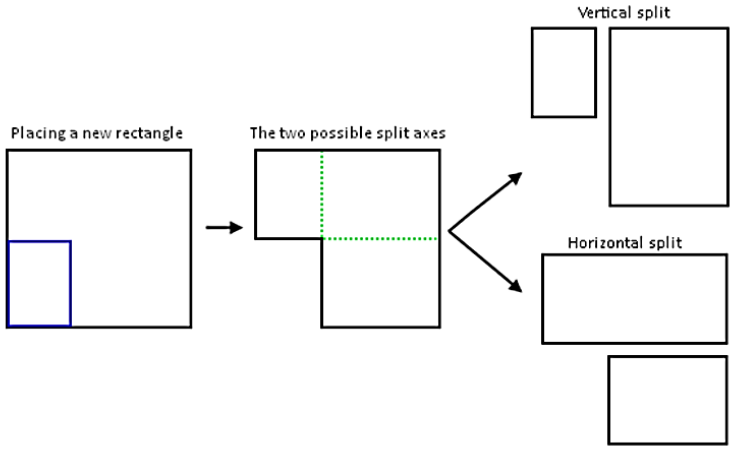


Figura 15-1. Proceso de división del espacio en el Guillotine Algorithm.

Fuente: Jukaj, Y. 2010

La siguiente pieza se localiza en cualquiera de los rectángulos libres y hará el mismo proceso. Por lo tanto, se irán formando múltiples opciones de corte. El algoritmo escogerá, dependiendo su finalidad y heurística, la opción de empacado más optima (Dyckhoff, Scheithauer y Terno 1997).

***Maximal Rectangles Algorithm*:**

Este algoritmo tiene una metodología de trabajo similar al algoritmo de guillotina, al ubicar una pieza se crea una lista de posibilidades en donde se puede colocar las piezas de menor dimensión. A diferencia del algoritmo de guillotina que se divide individualmente, el algoritmo de rectángulos máximos elige ambos ejes divididos al mismo tiempo para generar las diferentes posibilidades donde pueda ubicarse la pieza como se muestra en la figura 16-1 (Dyckhoff, Scheithauer y Terno 1997).

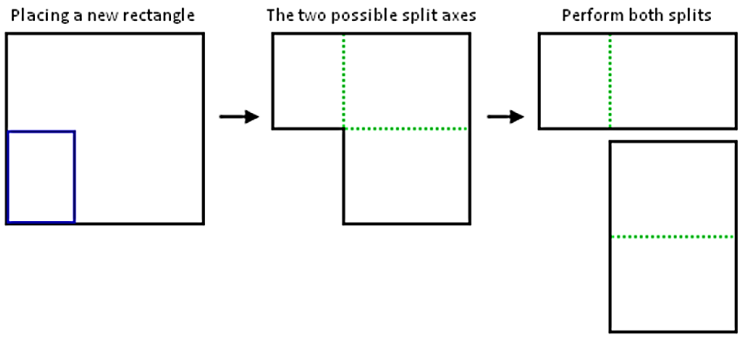


Figura 16-1. Proceso de división del espacio en el algoritmo *Maximal Rectangles*.

Fuente: Jukaj, Y. 2010.

***Skyline Algorithm*:**

Ya que, el algoritmo de rectángulos máximos genera una lista de rectángulos libres su programación es mucho más complicada que los otros algoritmos y genera un costo computacional mayor. El algoritmo de *Skyline* funciona de forma similar solo que mantiene una lista de todos los bordes horizontales, lo que se crean cada vez que se posiciona una pieza dentro de la plancha contenedora. Esta lista crece linealmente según el número de rectángulos ya ubicados y es muy fácil de administrar, de esta forma se crea un vector de posiciones para distribuir el espacio disponible y ubicar nuevas piezas (Dyckhoff, Scheithauer y Terno 1997).

Esta práctica, al igual que todos los métodos anteriores, tiene perdidas. Y escoger el algoritmo adecuado dependerá de la finalidad y recursos que se disponga.

1.5 Experior

Existen varios softwares utilizados para el modelado de un *Digital Twin* y *Experior* es uno de estos, teniendo la capacidad de replicar, simular, visualizar, emular, optimizar y poner en servicio virtual un sistema de automatización.

Xcelgo es la compañía que desarrollo este software y que durante estos últimos 15 años ha venido trabajando sobre este para poder perfeccionarlo razón por la cual, hoy en día, Experior puede trabajar con gemelos digitales a través de interfases gráficas en tres dimensiones de última generación que tienen interactividad total con los usuarios y que replican el funcionamiento de sistemas de automatización realizando así pruebas altamente realistas, mejorando resultados y rendimiento.

Experior posee una gran variedad de herramientas para facilitar el modelado en 3D, tales como: sensores, transportadores, unidades de control, entre otros. Por tanto, al momento que un usuario construye un modelo puede realizarlo con facilidad ya que puede seleccionar, arrastrar y soltar entre los modelos que ya están prediseñados y disponibles en un catálogo. Además, se puede desarrollar y crear cualquier objeto en cualquier herramienta CAD.

Tester, Commissioner, Builder y *Developer* son las cuatro licencias de Experior que brinda Xcelgo, cada una tiene diferentes permisos para realizar modificaciones en el software siendo la licencia de desarrollador la que permite modificar los componentes existentes, desarrollar sus propios componentes, administrar el contenido del catálogo y modificar la aplicación.

Entre los clientes principales del software podemos encontrar a Siemens, ABB, Rockwell automation, EWAB engineering, VOLVO, etc. (Xcelgo 2020).

1.5.1 Característica y aplicaciones

Las características principales del software son:

- Permite simulación física.

- Permite simulación discreta de eventos.
- Gráficos 3D modernos.
- Una amplia gama de medios de comunicación industrial, por ejemplo, control PLC y WCS / WMS.
- Integración con el flujo de trabajo de la empresa y los datos del proyecto.
- Implementación de código de software propio como complementos.

1.5.1.1 Comisionamiento virtual

Consiste en iniciar sistemas virtuales de bajo nivel. Es decir, se habla de un modelado en tres dimensiones que está conectado a un controlador físico, generalmente un PLC, que controla el modelo virtual de la misma manera en la que lo haría si fuera un modelo real por esta razón se pueden realizar varias pruebas para poder reducir el tiempo de programación, mejorar la calidad del software de control y producir una implementación segura en el sistema real (Xcelgo 2020).

1.5.1.2 Simulación

Xcelgo brinda un enfoque moderno simulación y de fácil interacción con el usuario lo que ayuda a acortar los tiempos de desarrollo y la falta de necesidad de ingenieros expertos en el tema al momento de crear modelos virtuales, entre las aplicaciones más importantes de Experior se pueden resaltar: simulación de diseño de sistemas, modelado físico, simulación multicuerpo y simulación matemática (Xcelgo 2020).

Beneficios:

- Es un apoyo al momento de tomar decisiones.
- Nuevas ideas y conceptos pueden ser analizados y diseñados.
- Eliminar cuellos de botella.
- Analizar diferentes configuraciones de sistemas o soluciones.
- Analizar y verificar el diseño específico del sistema con respecto a, por ejemplo, la capacidad, el rendimiento, la capacidad.

1.5.1.3 Sistemas de entrenamiento del operador

Ya que los modelos en tres dimensiones representan uno real estos pueden ser utilizados para poder capacitar a operadores, generando de esta manera una herramienta educativa que ayudará en la enseñanza para los trabajadores de la manera correcta al momento de utilizar una maquinaria gestionando los diferentes escenarios y evaluando las posibles actualizaciones de este (Xcelgo 2020).

Beneficios:

- Capacitar a los operadores y supervisores para un aumento rápido y seguro de la operación.
- Capacitar a nuevos operadores y supervisores.
- Capacite a los operadores y supervisores antes de la transición a nuevos modos de operación y situaciones que desafíen la operación.
- Pruebe la optimización del sistema: aumente el rendimiento, implemente nuevas capacidades.

1.5.1.4 Digital Twins

Un *Digital Twin* es capaz de monitorear y presentar una visualización en tres dimensiones de un sistema real al momento en el que este se encuentra en funcionamiento, así el usuario puede obtener una vista general del rendimiento para un análisis detallado del proceso de producción y de esta manera realizar la toma de decisiones acertadas sobre los sistemas (Xcelgo 2020).

Beneficios:

- Monitorea y muestra el estado en vivo de una instalación en funcionamiento.
- Obtener información detallada en áreas cruciales, como los cuellos de botella.
- Detectar parámetros para optimizar, mejorar el rendimiento y reducir el tiempo de inactividad, el tiempo de espera y los efectos de integración inesperados.

1.5.1.5 Espectador

Cuando un sistema es representado en tres dimensiones este se convierte en una importante herramienta de comunicación para el usuario final. Hoy en día, la tecnología nos permite recrear ambientes inmersivos donde un usuario al utilizar un visor puede recorrer la extensión total de una fábrica (Xcelgo 2020).

Beneficios:

- Ilustrar nuevas instalaciones para diferentes partes como futuros operadores o visitantes durante las presentaciones.
- Servicio de soporte, mantenimiento, reparación, etc. Con instrucciones y documentación ilustrativas.
- Mostrar rutas de emergencia.

1.5.2 Entorno

Xcelgo ofrece un entorno de programación fácil para Experior con un sistema de escoger, arrastrar y soltar para ensamblar modelos en 3D, esta interfaz posee cuatro principales pestañas que son: carpetas, importar, exportar e información.

En la ventana Modelo se visualiza el entorno grafico mientras que en la ventana Script se encuentra la programación asignada para este, figura 16-1.

Es posible importar catálogos que contengan nuevos accesorios de cualquier diseño en 3D, así como catálogos de robots o procesos ya existentes por ejemplo la serie de robots de Universal robots (Xcelgo 2020).

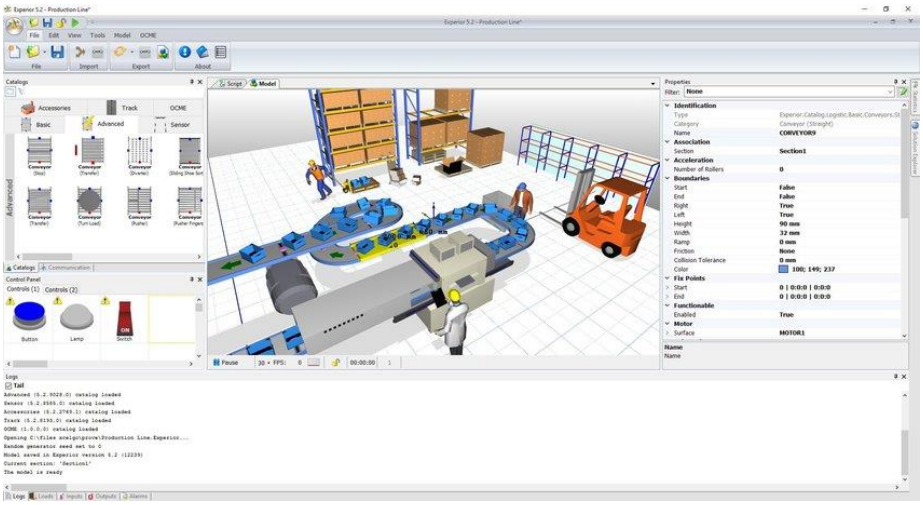


Figura 17-1. Interfaz gráfica del software Experior.

Fuente: Xcelgo, 2020

La construcción del diseño se realiza en el área de trabajo, también conocida como piso, con un tamaño y textura predeterminados de 30x30 [m] y cemento. Características que pueden ser modificadas.

El sistema de referencia se encuentra en el centro del plano de trabajo y todos los objetos que sean ubicados en este plano serán referenciados a este.

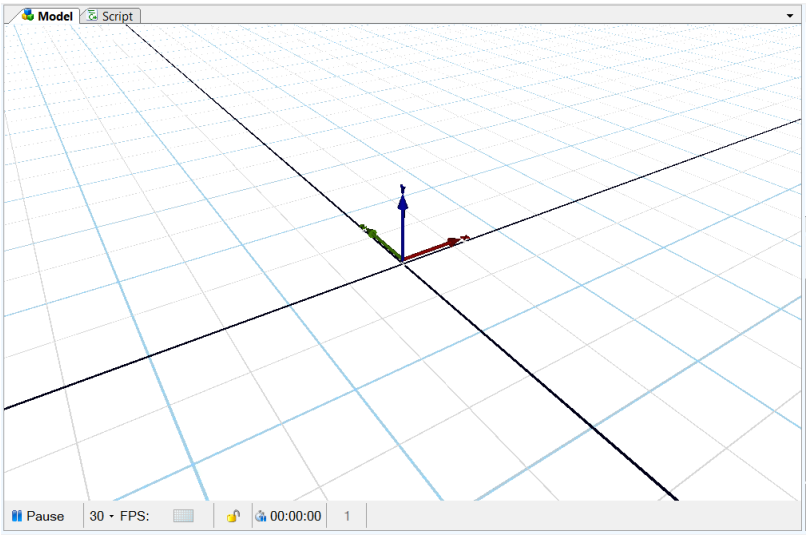


Figura 18-1. Plano de trabajo de Experior.

Fuente: Xcelgo, 2020

1.5.3 Comunicación

Dependiendo el tipo de licencia que se adquiera, Experior poseerá algunos protocolos de comunicación, algunos de los principales son:

- Siemens (que a su vez se divide en dos tipos de protocolos diferentes, Fetch & Write, que generalmente se usan juntos y Funciones S7)
- Ethernet / IP – CIP
- OPC
- Serie (RAW)
- ADS Beckhoff
- STX / ETX (contiene opciones para configurar conexiones en serie y TCP / IP)
- 3964R (contiene opciones para configurar conexiones en serie y TCP / IP)
- RFC1006

Los cinco primeros generalmente se usan para conexiones directas a PLC permitiendo a Experior usar y responder a PLC, código de lógica de escalera, etc.

Los siguientes protocolos son usados para configurar sistemas que permitir a Experior enviar y recibir telegramas y mensajes del sistema (Xcelgo 2020).

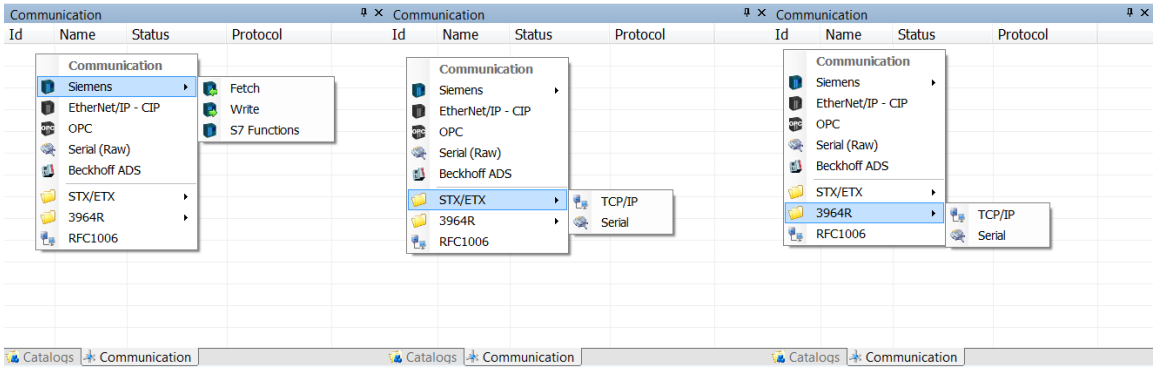


Figura 19-1. Protocolo de comunicación existentes en Experior.

Fuente: Xcelgo, 2020

Cuando la opción de autoconectar (Ubicada debajo de los protocolos de comunicación) está marcada como *True* el protocolo intentara conectarse al modelo automáticamente.

El usuario puede definir un tiempo de espera, antes que el protocolo intente conectarse automáticamente, utilizando la propiedad *Delay* como muestra la figura 20-1.

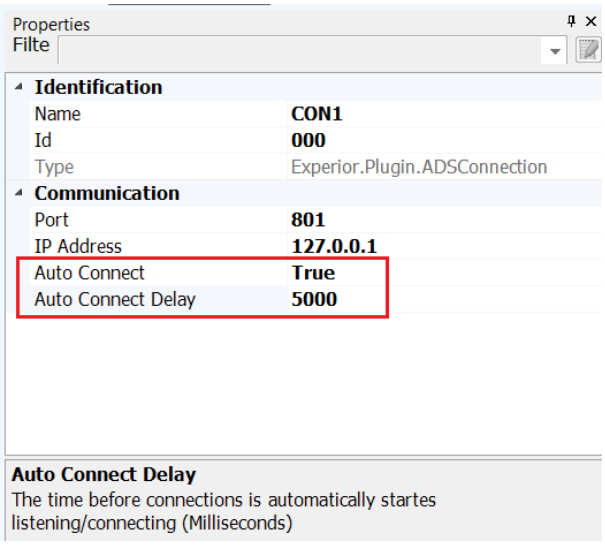


Figura 20-1. Auto conexión de los protocolos de comunicación en Experior.

Fuente: Xcelgo, 2020

1.5.4 Catálogos

Se puede crear uno o varios catálogos con diseños desarrollados por el usuario con el propósito de permitir a los usuarios de Experior importar objetos personalizados para su uso dentro del modelo.

Los archivos permitidos para ser importados son:

- Microsft DirectX(*.x).

- COLLADA (*dae).
- Wavefront(*.obj).
- Autodesk 3D Studio (*.3ds).
- Autodesk 3D Studio Max ASCII (*ase).
- Blender(*.blend).
- LightWave(*.Iwo).
- Stereolithography(*.stl).
- AutoCAD(*.dxf).

1.6 Python

Python, creado por Guido van Rossum y lanzado por primera vez en 1991, es un lenguaje de programación de alto nivel, multiparadigma y disponible en varias plataformas (Becerra 2009).

Python brinda al programador a realizar muchas tareas sin tener que empezar una programación desde cero ya que contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en su propio lenguaje (Chun 2001).

1.6.1 PIP (Python Package Index)

PIP es un sistema que gestiona paquetes, es decir, este puede instalar y administrar paquetes dentro de Python. En la actualidad, se puede encontrar muchos paquetes a disposición del programador.

Aunque en la mayoría de las versiones de Python PIP se encuentra preinstalado se puede utilizar el comando pyinstall para hacerlo (Chun 2001).

1.6.1.1 Librería MATH

Es una librería brindada por Python que forma parte de su "librería estándar", ofreciendo la posibilidad a los programas en los que se utiliza poder realizar operaciones matemáticas de números reales. Algunas de las funciones ofrecidas son:

- Funciones numéricas
- Funciones de potencial y algorítmicas
- Funciones trigonométricas y conversión de ángulos
- Constantes

Esta librería es importante para programar tareas específicas como: estudiar movimientos, rotaciones, simulación de circuitos. En fin, realizar tareas que contengas problemas matemáticos complejos que requieran de funciones específicas (Chun 2001).

1.6.1.2 Librería GREEDYPACKER

Esta librería realiza una distribución de contenedores rectangulares bidimensionales a un área específica. La librería incluye todos los algoritmos heurísticos y optimizaciones como lo son:

- Estante.
- Guillotina.
- Rectángulos máximos.
- Horizonte.

Las heurísticas que contiene la librería son:

- ASCA: Ordenar por área ascendente.
- DESCA: Ordenar por área descendente (esta es la configuración predeterminada).
- ASCSS: ordenar por lado más corto ascendente.
- DESCSS: Ordenar por lado descendente más corto.
- ASCLS: ordenar por lado más largo ascendente.
- DESCLS: ordenar por lado más largo descendente.
- ASCPERIM: Ordenar por perímetro ascendente.
- DESCPERIM: Ordenar por perímetro descendente.
- ASCDIFF: Ordenar por la diferencia de ABS entre lados ascendentes.
- DESCDIFF: Ordenar por la diferencia de ABS entre lados descendentes.
- ASCRATIO: ordenar por la proporción de los lados ascendentes.
- DESCRATIO: ordenar por la proporción de los lados descendentes.
- Falso: Empaque en el orden agregado al binmanager.

La librería configura todas sus opciones por medio de palabras reservadas que pueden ser activadas a lo largo del programa (Bothwell Solomon 2019).

1.6.2 Comunicación

La comunicación permite conectar dos equipos a través de la red, esto es normalmente un modelo cliente-servidor el cual es un Framework de comunicación distribuida de procesos de redes entre solicitantes, clientes y proveedores de servicio. Una conexión cliente-servidor es normalmente establecida a través de una red de Internet.

Un socket es un módulo dentro de Python que permite realizar conexiones TCP/IP y UDP. Los sockets actúan como el receptor y emisor dentro de un canal de comunicación bidireccional. Cuando se trabaja con un servidor se puede usar el módulo socketserver dentro de Python y cuando se abra la conexión cliente-servidor, ambos sockets pueden emitir y recibir mensajes (Becerra 2009).

1.6.2.1 Funciones importantes

- Socket ( )

La función socket ( ) es la encargada de retornar un descriptor de socket, el cual se puede usar para llamadas al sistema. Si nos retorna -1, se ha producido un error esto podría resultar útil para verificar errores.

- Bind ( )

La llamada Bind () se usa cuando los puertos locales de nuestra máquina están en nuestros programas como opciones de uso, cuando utilizamos la llamada listen(). Es utilizado para asociar un socket con un puerto (de nuestra máquina).

- Connect ( )

La función connect() se usa para conectar nuestra maquina a un puerto definido en una dirección IP. Esta función devuelve como resultado -1 si algún error llega a ocurrir.

- Listen ( )

La función listen() se usa si cuando se esperan conexiones entrantes, es decir, si se requiere una conexión.

- Send ( )

Esta función es la encargada de enviar datos usando sockets de flujo o sockets conectados de datagramas, pero si se desea enviar datos usando sockets no conectados de datagramas se debe usar la función sendto(). Al igual que el caso anterior, estas dos funciones devuelven -1 en caso de error, o el número de bytes enviados en caso de éxito.

Python permite generar graficas las cuales son de utilidad para comprobación de resultados, las gráficas en dos dimensiones requieren de librerías como lo son:

- Matplotlib
- Numpy

Las funciones contenidas en estas librerías procesan los datos devolviendo como resultado graficas en un plano cartesiano (Becerra 2009).

1.7 Multi-Robot Manufacturing Cell Commissioning

La *Smart Industry* está en auge actualmente en el proceso de fabricación, dentro del contexto de fabrica 4.0 los sistemas ciber-físicos controlan la producción donde los robots colaborativos

cumplen un roll importante para algunas tareas repetibles, generales y estratégicamente importantes.

El paso hacia la *Smart Industry* es un proceso gradual ya que no todas las tecnologías tienen el grado de flexibilidad requerido para que esto suceda. La incorporación de robots colaborativos permite una combinación eficaz de la producción manual y automatizada. En la mayoría de las industrias se requiere más de un robot de diferentes fabricantes por lo que se necesita de un diseño de sistemas multi-robot para priorizar la calidad y eficiencia.

Multi-robot Manufacturing Cell Commissioning es una metodología diseñada por la fundación IDONIAL que es la fusión de las fundaciones ITMA y PRODINTEC que cuenta con más de 28 años de experiencia en el desarrollo de materiales, la fabricación avanzada y la industria digital generando metodologías de desarrollo para dichos campos. Esta metodología es utilizada en el diseño de automatización, implementación mejorada y monitoreo en tiempo de un proceso basado en la creación de un gemelo digital con una interfaz inmersiva que se utiliza en el banco de pruebas virtual antes de la implementación física. Además, Se puede utilizar de forma eficiente para la formación de operadores, el seguimiento en tiempo real y los estudios de viabilidad de optimizaciones futuras (Pérez et al. 2020).

La metodología cuenta con cuatro sub-procesos claves en el desarrollo de un Digital Twin los cuales val desde el diseño hasta la puesta en funcionamiento y son:

- *Desing*
- *VR molde*
- *Implementation*
- *Digital Twin*

Cada etapa cuenta con pasos a seguir y retroalimentación para un correcto funcionamiento las cuales se detallan en la figura 21-2.

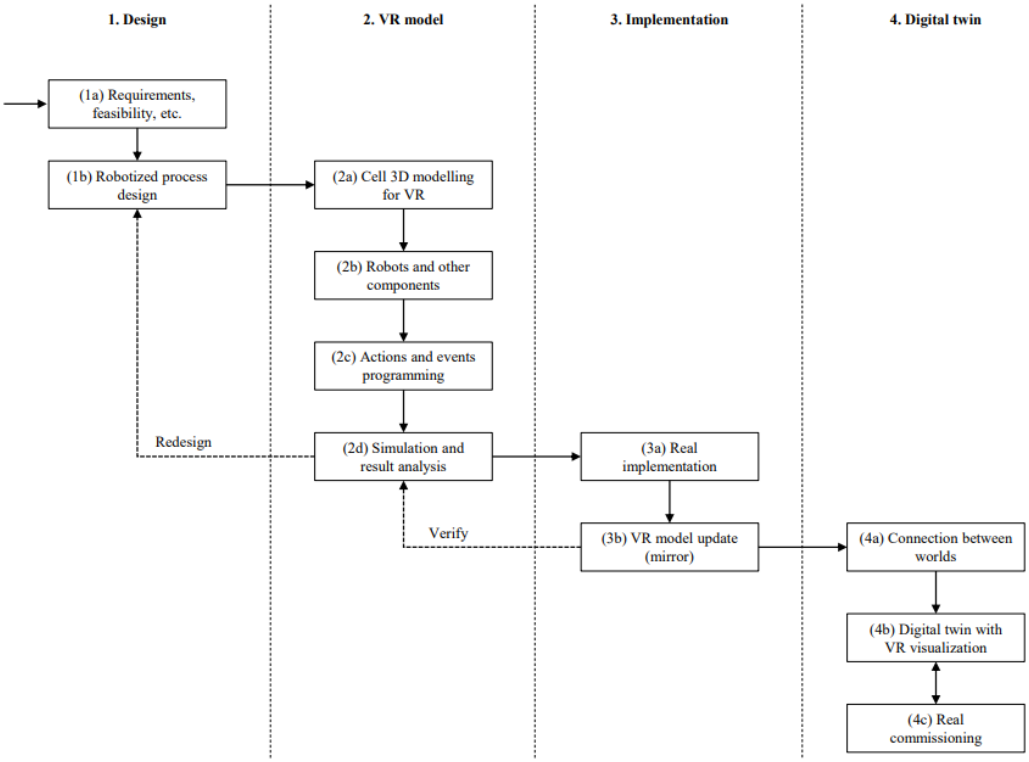


Figura 21-2. Metodología secuencial con retroalimentación para crear un Digital Twin.

Fuente: Pérez.; et al. 2020

CAPÍTULO II

2 MARCO METODOLÓGICO

2.1 Introducción

En el presente capítulo se describe el desarrollo metodológico del Diseño de una Interfaz Digital Twin de un brazo robótico UR3, que realiza el marcado para corte de superficies usando Experior y tiene como objetivo principal replicar el movimiento del brazo robótico UR3 mientras realiza dicho proceso. En la presente investigación se detalla la metodología de investigación científica y tecnológica que permiten cumplir con el ciclo de vida del proyecto de ingeniería y sus objetivos.

Methodology for Multi-Robot Manufacturing Cell Commissioning, es una metodología que basa su estudio en cuatro fases con el fin de determinar un proceso de virtualización que puede ser migrado a un proceso real, esta metodología servirá de referencia para el diseño de la interfaz Digital Twin y consta de cuatro fases: Diseño, Modelo VR, Implementación y Digital Twin, las mismas que son detalladas a continuación:

La fase de Diseño conlleva el análisis de los requisitos del nuevo proceso a ser modelado en este caso el brazo robótico UR3, en el mismo sentido en esta fase se diseña el proceso para el marcado de corte de superficies. La fase Modelo VR proporciona las pautas y directrices necesarias para el modelado 3D del brazo robótico UR3 y del entorno de trabajo, para este modelado se utiliza tanto el software Experior como el software Ursim que simula exactamente el brazo robótico UR3 y demás aplicaciones que servirán para la programación, acción de eventos, simulación y análisis de resultados. Una vez que el entorno virtual se ha modelado se procede con la fase de implementación en la cual se integra el modelado virtual hacia un escenario real, en esta fase se aprovecha la realización de algún cambio que puede surgir en el modelado original VR con el fin de mantener un reflejo sin alteración del brazo robot UR3 virtual. Finalmente, la fase Digital twin permite conjugar las conexiones y puesta en servicio de los entornos virtuales y reales.

La metodología de investigación para el diseño de una interfaz digital twin de un brazo robótico UR3 que realiza marcado para corte de superficies usando Experior se detalla a continuación:

El método descriptivo y exploratorio permite definir algunas tecnologías y estudios realizados referentes a las interfaces Digital Twin y el uso de Experior, que servirá de guía para el desarrollo de la interfaz. La metodología descriptiva, exploratoria y no-experimental determina el hardware y software adicional y necesario para el marcado y corte de superficies en entornos tanto virtual

como real. Para el diseño de la interfaz Digital Twin se utiliza el método aplicativo, no experimental y de laboratorio, con el fin de replicar virtualmente el movimiento del brazo robótico UR3 en el marcado de piezas para el corte de superficies, se generan diferentes escenarios de prueba hasta lograr un entorno de virtualización óptimo.

Finalmente, la interfaz Digital Twin se implementa aplicando la metodología analítica y de laboratorio para el proceso de marcado de piezas para el corte de superficies que realiza el brazo robótico UR3, con esta implementación se procede a la validación de la fidelidad y efectividad de la interfaz Digital Twin, con escenarios de prueba y error al replicar los movimientos del brazo robótico UR3 en el marcado de piezas para el corte de superficies.

2.2 Metodología para el diseño de una interfaz *Digital Twin* de un brazo robótico UR3

2.2.1 Tecnologías y estudios previos

Los estudios realizados anteriormente referentes a interfaces *Digital Twin* y al uso de software Experior son tratados en el capítulo anterior como también en los antecedentes, donde se especifica casos de estudio, casos de éxito y la interfaz de trabajo del software Experior. Sin embargo, se tratará a fondo la investigación previa que dio paso a la presente tesis.

La presente investigación surge con el ánimo de mejorar el estudio realizado por la universidad Autónoma de Los Andes de Bogotá Colombia denominada "*Human-Computer-Machine Interaction for the Supervision of Flexible Manufacturing Systems: A Case Study"* la cual fue publicada en el año 2018 y tiene como objetivo una interacción humano-computador-máquina perfecta, que ayuda en la supervisión de un operador para la validación de un caso de estudio en el contexto de sistema de fabricación flexible. La arquitectura propuesta para el cumplimiento del objetivo está representada en el gráfico 1-2.

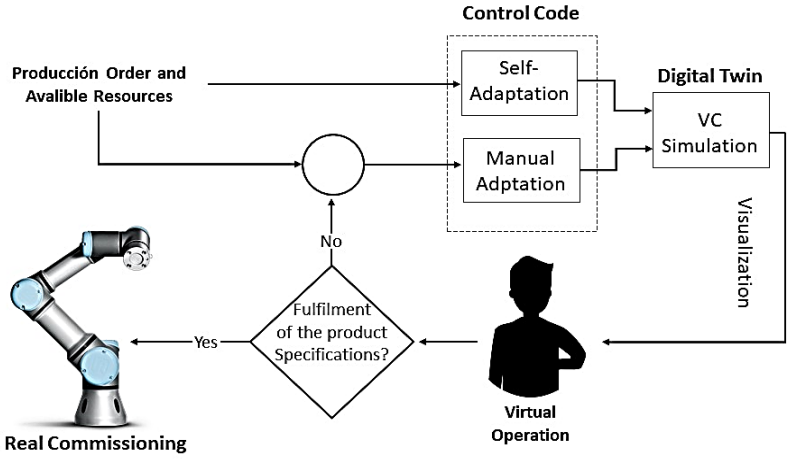


Gráfico 1-2. Arquitectura de la investigación previa.

Fuente: José, H.; et al. 2018

La figura 1-2 representa la arquitectura llevada hacia un modelo real donde se denota varias deficiencias las cuales son tratadas y resultas en esta investigación.

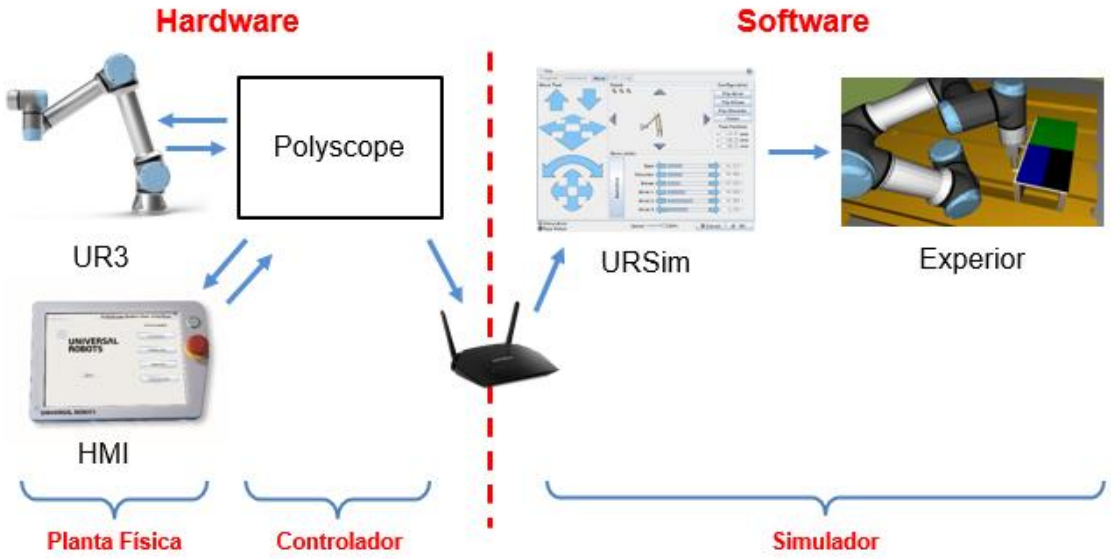


Figura 1-2. Arquitectura implementada y funcional.

Realizado por: Alexis, M.; Sofía, S. 2020

La investigación destaca los siguientes problemas:

La comunicación entre PolyScope y el algoritmo que selecciona las trayectorias es no es óptima. El algoritmo se encuentra y compila sobre un servidor privado perteneciente a la Universidad Autónoma de los Andes por lo que no puede ser utilizado fuera de la universidad, a la par presenta problemas de conexión con PolyScope.

El sistema propuesto no es robusto en cuanto al entorno de trabajo debido a que no posee una retroalimentación que indique el estado actual del sistema como: posición y orientación de las piezas a ser cortadas. Por tanto, el sistema es monótono e incapaz de adaptarse a cambios en el ambiente de trabajo.

El algoritmo de corte no proporciona una buena distribución de los cortes dentro de la plancha por lo que es esencial mejóralo, teniendo en cuenta la función objetivo que se desea alcanzar junto con las variables de decisión que determinaran un algoritmo de corte adecuado en concordancia con nuestra necesidad y disponibilidad de recursos.

Las soluciones se presentan como el objeto de estudio del presente trabajo de titulación y son desarrolladas a lo largo de este capítulo.

2.2.2 Fase diseño

2.2.2.1 Selección de hardware y software

Los problemas antes mencionados requieren de soluciones tanto en software como hardware los que se describen a continuación.

La necesidad de suprimir el servidor en el cual se está compilando el algoritmo de corte converge en la búsqueda de un nuevo lenguaje de programación donde se pueda desarrollar el mismo. Una de las opciones más viables fue generar un script, función subyacente a PolyScope, el cual contenga el algoritmo de corte de esta forma el problema de comunicación es inexistente como se observa en la figura 2-2

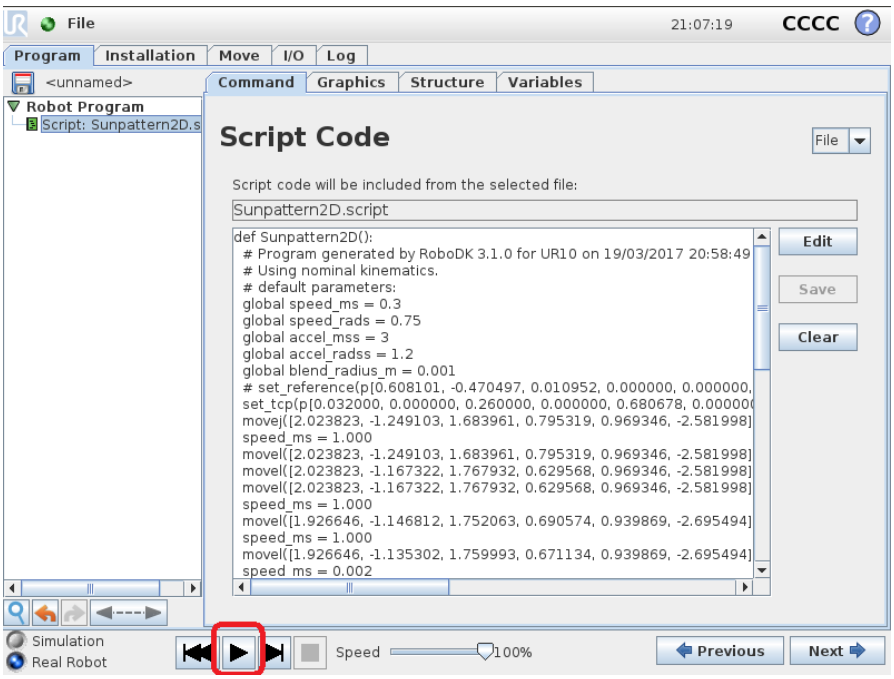


Figura 2-1. Algoritmo de corte dentro de un script en PolyScope.

Realizado por: Alexis, M.; Sofía, S. 2020

El desarrollo del algoritmo dentro de un script desemboco en un complejo problema de sintaxis ya que el formato de instrucciones, como funciones que procesa PolyScope, es pensado únicamente para programar el movimiento del robot, configurar entradas y salidas, agregar tiempos, generar y modificar variables, en general programar el robot de una forma ordenada, extensiva y con más versatilidad de la que se obtiene al programar con la interfaz principal que ofrece PolyScope. Debido a esto, esta solución no posee la capacidad de procesar algoritmos complejos que no estén enfocados con la programación del robot. Otra gran deficiencia es que no puede ejecutar código interpretado como el que se genera en Python. Por último, no existen librerías que se puedan importar a un script de PolyScope las cuales sean capaces de ayudar en el caso de estudio.

Debido a lo mencionado anteriormente, para la programación del algoritmo se escogió el lenguaje Python ya que destaca múltiples beneficios como:

- Al ser un leguaje interpretado, Python permite que el código fuente pueda ser ejecutado en cualquier plataforma siempre que este disponga del interprete. En otras palabras, es un leguaje multiplataforma y portable, compatible con diferentes sistemas operativos (Windows, Linux y MacOS) y con múltiples tarjetas de desarrollo. Además, ya que no necesita ser compilado reduce el tiempo de desarrollo y prueba lo que hace de este lenguaje una herramienta ideal para trabajar con volúmenes grandes de datos. Al poseer estas características, Python, favorece al programador no solo en el procesamiento de datos si no también en su extracción.

- Python posee una gran biblioteca de recursos y soporta varios modelos de desarrollo haciéndolo un lenguaje multiparadigma, tiene una amplia capacidad de combinar librerías lo que lo convierte en un lenguaje con gran rendimiento para proyectos de machine learning, inteligencia artificial, Big Data, automatización de procesos y tecnologías de vanguardia.

- Este lenguaje es pensado para vincularse con las maquinas mediante protocolos como lo son TCP/IP o comunicación serial, permitiendo una fácil interacción con dispositivos robóticos y controladores lógicos programables. A su vez, Python es ampliamente usado en entornos digitales como lo es Ursim y RoboDK.

- Por último, Python es *Open Source* por lo que cuenta con un gran número de usuarios que participan activamente en el avance de este lenguaje, se puede descargar y modificar una gran cantidad de contenido útil para el desarrollo de proyectos, así como publicar códigos creados en el mismo.

Además, es importante mencionar que el algoritmo a desarrollar tiene que comunicarse con el robot físico, pero, por motivos de fuerza mayor debido a la situación de emergencia sanitaria por el COVD-19, el acceso al brazo robótico UR3 no fue posible. La solución de esta problemática fue descargar el software de simulación que ofrece la misma compañía Universal Robots. Este software se denomina “Ursim” y contiene una réplica virtual de la interfaz de programación y el modelo para una simulación exacta del robot físico. Además, Ursim es de código abierto y desarrollado en el sistema operativo Ubuntu por lo que también fue necesario el software “Vmware Workstation 15 Player” para implementar máquinas virtuales.

La adaptación al entorno de trabajo requiere de un dispositivo que genere una retroalimentación del plano de trabajo. La empresa Robotiq brinda la herramienta necesaria para este propósito. La *Wrist Camera* es la opción perfecta ya que al ser un dispositivo instalado como un componente del brazo robótico UR3 interactúa con el mismo. De esta forma, los datos que genera la cámara se utilizan directamente en Polyscope y así se evita la problemática de extracción y análisis que un dispositivo externo puede generar.

2.2.2.2 Diseño de la arquitectura para la interfaz digital twin en el proceso de corte de piezas

Se propone una correlación optima entre el operador, el *Digital Twin*, la planta de producción y el entorno de trabajo, es decir, se tiene una interacción humano-computadora-máquina (HCMI) que puede aplicarse a cualquier entorno de fabricación donde pequeños lotes de productos personalizados son una de sus características.

El gráfico 2-2 muestra la arquitectura HCMI propuesta para ayudar al operador en la "actividad de supervisión" dentro del contexto de la producción flexible.

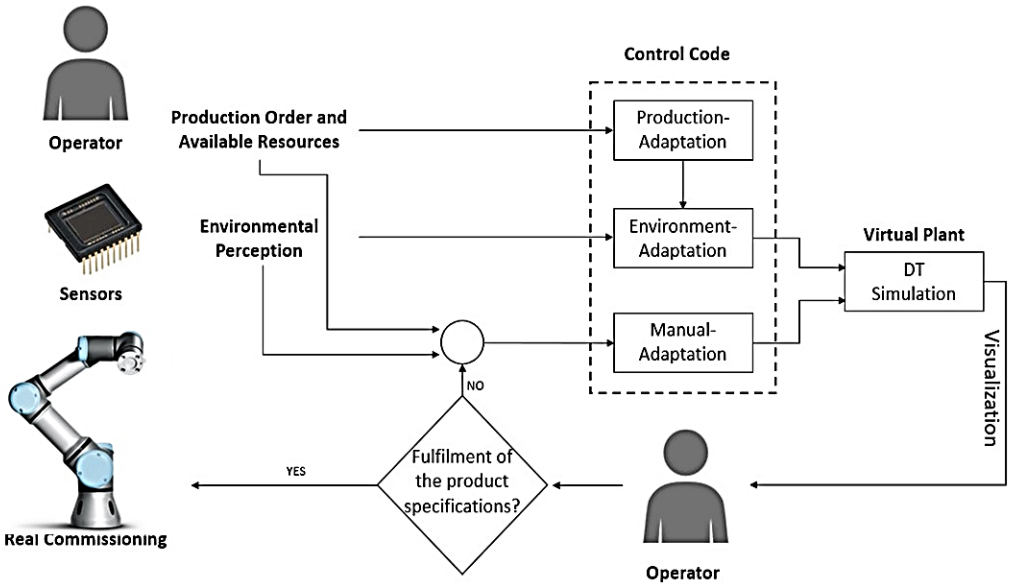


Gráfico 2-1. Arquitectura para la interfaz *Digital Twin* en el proceso de corte de piezas

Realizado por: Alexis, M.; Sofia, S. 2020

El proceso da inicio con los requerimientos de corte que son ingresados por el operador, estos especifican los recursos disponibles y la orden de producción que da paso a la operación de fabricación. Con la orden de producción definida, la percepción y adaptación del entorno es realizada mediante la *Wrist Camera*. Los datos generados por el operador y la cámara son enviados hacia el controlador que mediante bucles de ajuste auto-adapta la lógica de su control

para cumplir la orden de producción con los recursos a disposición. Terminada esta acción, el *Digital Twin* inicia una réplica de las configuraciones de la actividad de fabricación. Dicha acción es supervisada por un operador que valida la factibilidad de la operación (por ejemplo, movimientos bruscos, colisiones, adaptación inadecuada al entorno, etc.).

La supervisión visual del operador llevará a dos posibles resultados. En el primer resultado, la operación de fabricación se puede realizar con éxito y los productos satisfacen las especificaciones de la orden de producción. En este caso, el supervisor da paso al *Real Commissioning* y el proceso termina. En el segundo resultado, existe una falla en la operación de autoadaptación debido a que la orden de producción no es factible o el producto no cumple con las especificaciones requeridas. Esto crea la necesidad de adaptar la lógica de control de manera manual. A continuación, los datos generados (el movimiento del *Digital Twin*, el código lógico de control, la orden de producción, los recursos disponibles y la percepción del entorno) serán utilizados como base para que un ingeniero en software pueda diseñar una nueva lógica de control que sea viable y comenzar un nuevo ciclo de validación. Sin embargo, la adaptación manual se plantea como trabajo futuro.

2.2.3 Fase Modelo VR

2.2.3.1 Algoritmos de corte

Se selecciona el algoritmo de corte tratándolo como un problema bidimensional para piezas rectangulares que busca maximizar la salida logrando así una solución viable, convergente rápida con los recursos disponibles.

La elección de la heurística es un factor fundamental en la determinación del algoritmo. En la tabla 1-2 se presentan los algoritmos de corte más conocidos junto a los factores determinantes de selección. A la luz de la comparación de dichos factores se decantó por el *Corte Guillotina* que presenta las mejores características dentro del tema de estudio y en comparación con sus competidores.

Tabla 1-2. Comparación entre algoritmos de corte

ALGORITMOS VS FACTORES DETERMINANTES DE SELÉCCIÓN	Maximización salida	Minimización entrada	Piezas rectangulares	Generación de columnas	Patrón de corte libre	Fácil implementación
Algoritmo Grasp		X	X		X	
Algoritmo Gloton		X	X	X		X
Algoritmo Viswanathan y Bagchi		X			X	
Algoritmo Genético	X			X	X	

Algoritmo corte Guillotina	X		X	X		X
Heurísticos	X				X	
Branch And Bound	X			X	X	X

Realizado por: Alexis, M.; Sofia, S.2020

El gráfico 3-2 muestra el diagrama de flujo *Corte Guillotina* utilizado para el problema de corte en 2D.

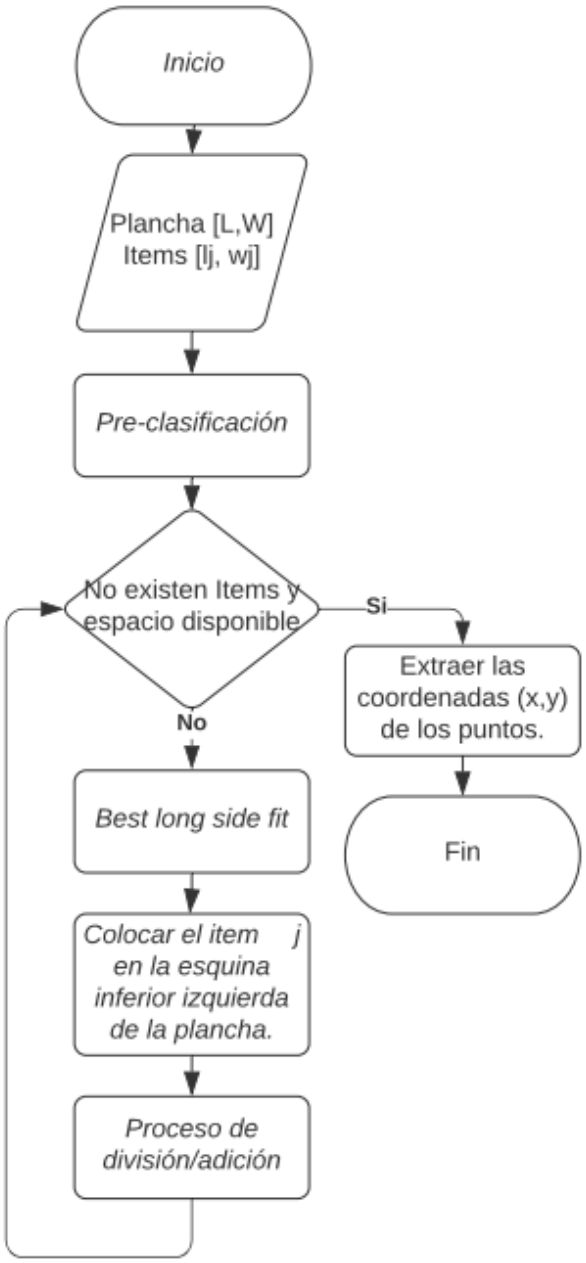


Gráfico 3-2. Diagrama de flujo del algoritmo utilizado para el problema de corte

Realizado por: Alexis, M.; Sofia, S.2020

Los datos de ingreso hacia el algoritmo son: los recursos disponibles y la orden de producción. En este caso, el recurso disponible es una plancha rectangular de ancho W y largo L mientras que la orden de producción es definida por las piezas que requiere el operador de dimensiones [wj, lj]. A continuación, se hace una preclasificación que enlista las piezas de manera descendente según su área y pasa a una condición en donde el algoritmo posiciona todas las piezas una por una o ya no existe espacio en la plancha, esto se realiza según la heurística que en esta aplicación es *best long side fit.* Entonces el algoritmo termina su ejecución generando como datos de salida las coordenadas cartesianas de las 4 esquinas de cada una de las piezas rectangulares finales. Los puntos están definidos con referencia al punto (0,0) ubicado en la esquina inferior izquierda de la plancha original como se muestra en la figura 3-2.

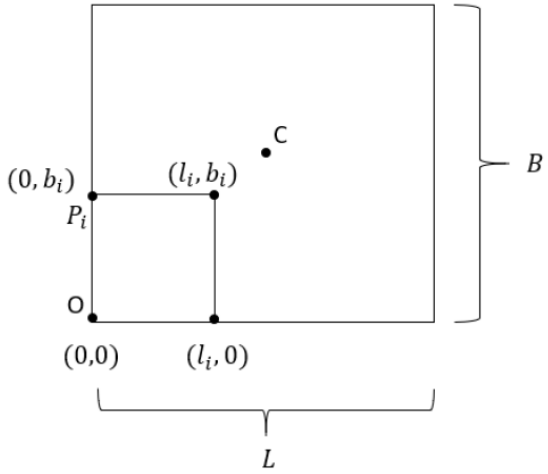


Figura 3-2. Ejemplo de los puntos cartesianos generados por el algoritmo.

Realizado por: Alexis, M.; Sofia, S.2020

2.2.3.2 Percepción del entorno de trabajo

La ubicación y orientación inicial de las planchas que llegan al espacio de trabajo del robot son captadas con la adaptación hardware llamada Wrist Camera que junto al software de visión Camera Locate generan una interfaz de aprendizaje, lo que permite configurar el sistema de visión para que pueda reconocer la ubicación y orientación de las planchas automáticamente.

La ubicación y orientación es definida por la variable *object_location* la cual posee 6 datos.

Ecuación 1-2. Variable obtenida por la cámara

$$object_{location} = [X_0, Y_0, Z_0, x_{rotación}, y_{rotación}, z_{rotación}\,] \quad (1)$$

Donde los tres primeros datos son la posición del centro del objeto con relación al sistema de referencia de la base del robot y los 3 siguientes son la orientación con relación al sistema de referencia de la base del robot donde los ejes x rotación e y rotación son paralelos al plano de

trabajo en el que se realizó la calibración y el eje z es normal al plano formado por los mismo y apunta hacia abajo. Los sistemas de referencia y sus rotaciones se muestran en la figura 4-2ª.

El plano formado por los ejes Y_W e X_W siempre será paralelo al plano formado por los ejes Y_P e X_P sin importar la orientación en la que llegue la plancha por lo que la rotación de esta se da sobre el eje Z_P como se muestra en la figura 4-2b. El dato brindado por la cámara nos muestra esta rotación menos 90 grados que representan la rotación entre el sistema de referencia del plano de trabajo y el de la plancha figura 4-2ª.

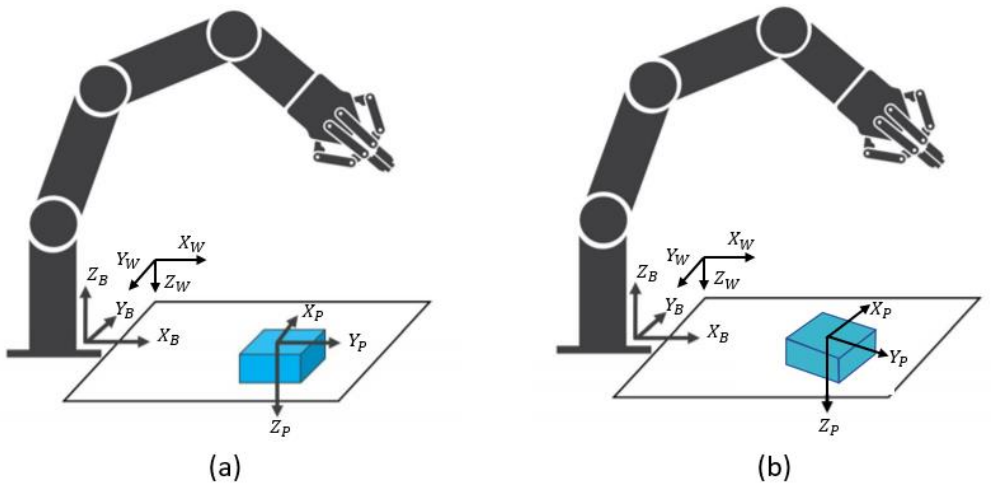


Figura 4-2. (a) Sistemas de referencia del robot y la pieza sin rotar. (b) Sistemas de referencia del robot y la pieza rotada.

Realizado por: Alexis, M.; Sofia, S.2020

Los datos que entrega el algoritmo de corte no tienen ninguna relación con el robot o la plancha. Lo que se busca con la percepción del entorno de trabajo es que el algoritmo funcione en un entorno cambiante y no solo como un proceso computacional ideal.

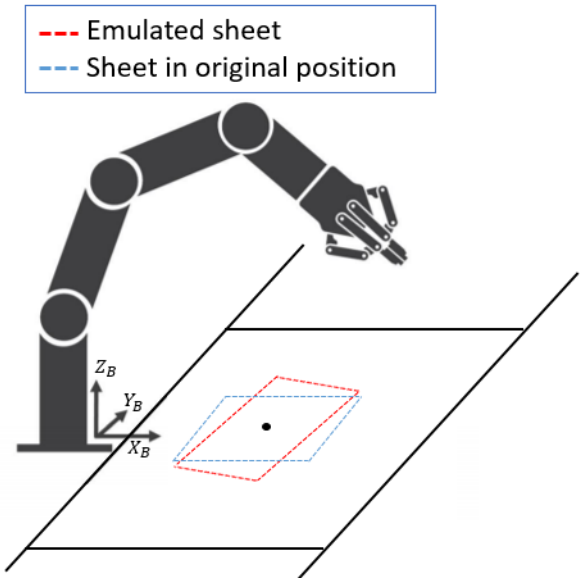


Figura 5-2. Robot UR3 adquiriendo datos mediante la cámara.

Realizado por: Alexis, M.; Sofia, S.2020

Para trasladar los puntos obtenidos por el algoritmo de corte hacia la plancha real, se asume una plancha de tamaño (L x W) perpendicular al sistema de referencia de la base del robot y ubicando su centro en la coordenada (x_0,y_0,z_0) obtenida por la cámara como muestra la figura 5-2.

A todos los puntos obtenidos por el algoritmo se les suma la coordenada de la esquina inferior izquierda de la plancha asumida (X_p, Y_p, Z_p) que se muestra en la figura 6-2, la cual se consigue con la ecuación 2-2.

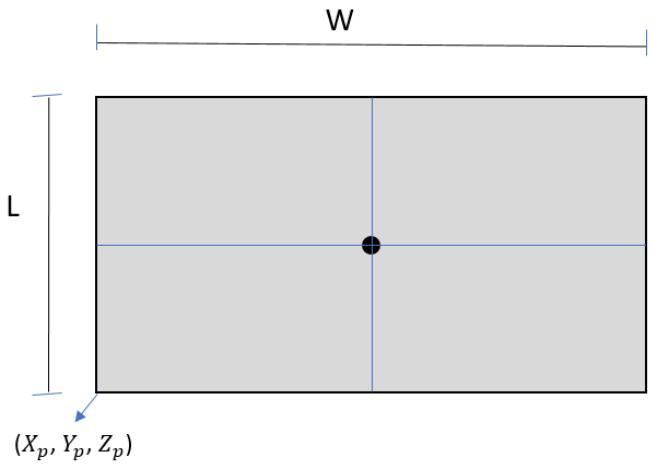


Figura 6-2. Representación de la plancha asumida.

Realizado por: Alexis, M.; Sofia, S.2020

Ecuación 2-1. Fórmula para hallar la esquina inferior izquierda.

$$(X_p, Y_p, Z_p) = \left(-\frac{W}{2}, -\frac{L}{2}, Z_0\right)$$

De esta forma, y como se observa en la figura 7-2, todos los puntos quedarán con respecto al sistema de referencia de la base del robot.

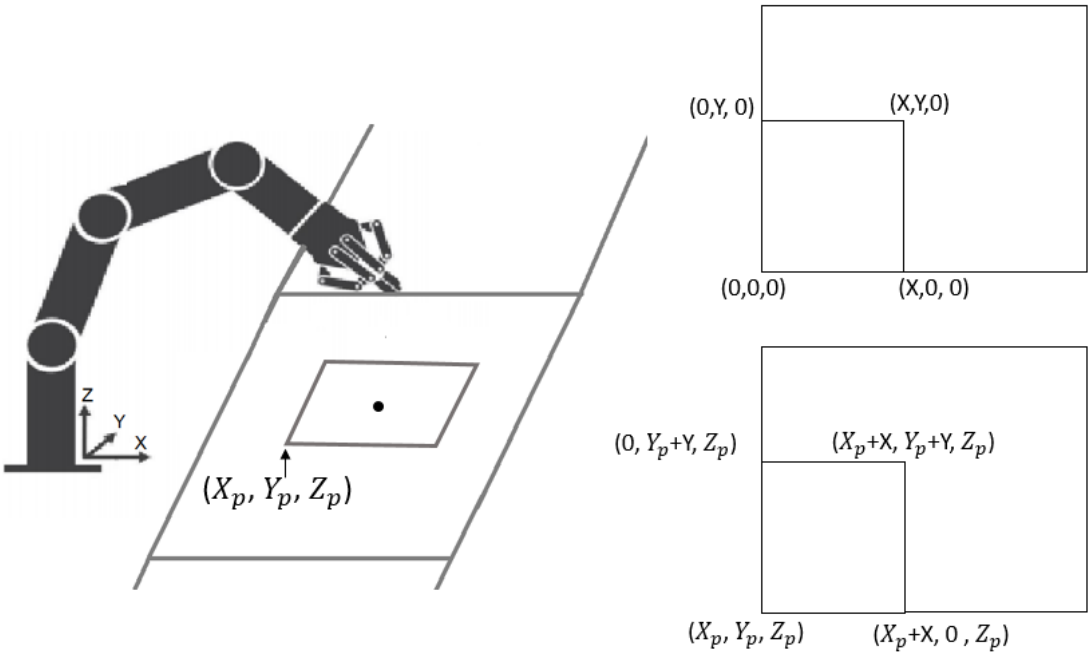


Figura 7-2. Puntos referentes a la base del robot.

Realizado por: Alexis, M.; Sofia, S.2020

Al ubicar los puntos en la plancha emulada quedan paralelos al sistema de referencia de la base del robot. A continuación, se debe rotar estos para llegar a la posición real.

Para realizar la rotación se toma como eje el centro de la plancha, por ende, todos los puntos deben estar referenciados con respecto al mismo, esto se lo consigue médiate una resta de vectores siguiendo la ecuación 2-3 a través de todos los puntos, como se muestra en la figura 8-2.

Ecuación 3-2. Vector posición.

$$\overrightarrow{V_{CP}} = \overrightarrow{V_{BP}} - \overrightarrow{V_{BC}}$$

$\overrightarrow{V_{CP}} = Vector\ desde\ el\ centro\ de\ la\ pieza\ al\ punto\ P.$
$\overrightarrow{V_{BP}} = Vector\ desde\ la\ base\ del\ robot\ hasta\ el\ punto\ P.$
$\overrightarrow{V_{BC}} = Vector\ desde\ las\ base\ del\ robot\ hasta\ en\ centro\ de\ la\ pieza\ \ (dato\ obtenido\ por\ cámara)$

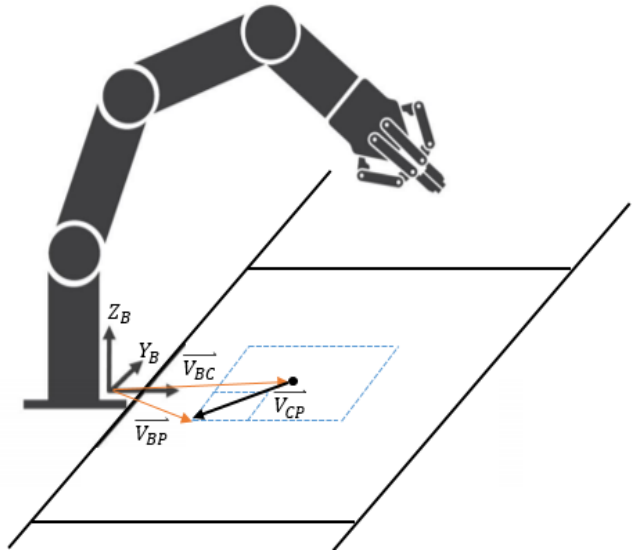


Figura 8-2. Resta de vectores para ubicar los puntos.

Realizado por: Alexis, M.; Sofia, S.2020

Con los puntos referenciados al centro de la plancha aplicamos la matriz de rotación mostrada en la ecuación 2-4, como se observa en la figura 9-2.

Ecuación 4-2. Matriz de rotación en Z

$$\begin{bmatrix} X_R \\ Y_R \\ Z_R \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{x} \begin{bmatrix} X_n \\ Y_n \\ Z_n \end{bmatrix}$$

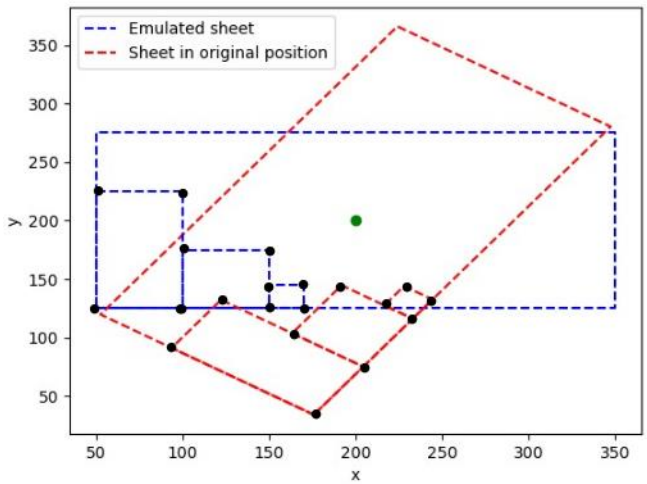


Figura 9-2. Puntos generados por el algoritmo de corte y ubicados en la plancha real

Realizado por: Alexis, M.; Sofia, S.2020

Por último, para poder utilizar los puntos previamente rotados tiene que ser nuevamente referenciados a la base del robot lo que se consigue utilizando una vez más la ecuación 2-3. Despejando esta vez $\overrightarrow{V_{BC}}$.

2.2.3.3 Virtualización del brazo robótico UR3 y entorno

La empresa desarrolladora del software *Experior* es un *partner* comercial de la compañía *Universal Robots*, por lo que posee un catálogo que contiene sus productos como son los brazos robóticos UR3, UR5 y UR10 los mismos que pueden ser configurados para trabajar como *Digital Twin*. La figura 10-2 muestra el brazo robótico UR3 en la interfaz de *Experior*.

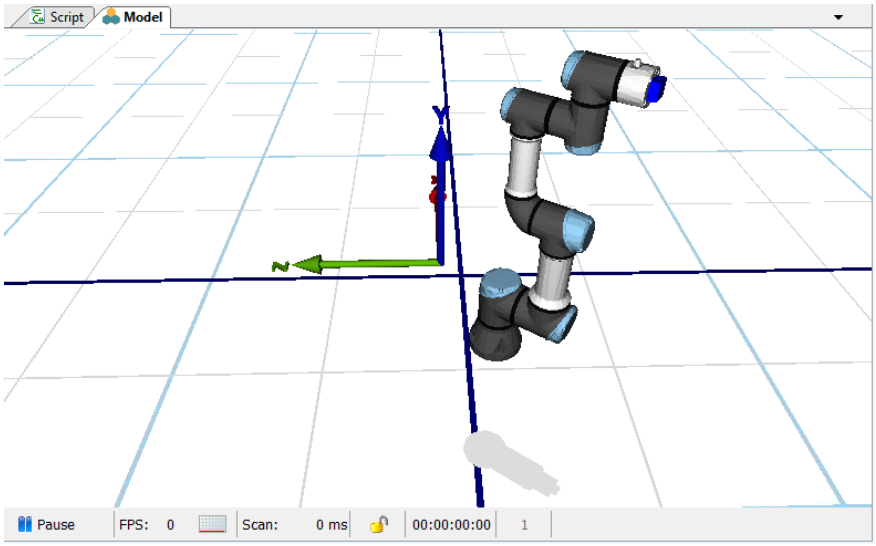


Figura 10-2. Brazo robótico UR3 dentro del software *Experior*.

Realizado por: Alexis, M.; Sofia, S.2020

Como se puede observar en la figura 10-2, el modelo 3D es una réplica exacta tanto física como funcional del robot UR3. La comunicación entre el robot físico y digital se da mediante protocolo Modbus TCP/IP donde los dos deben poseer la misma dirección.

Los diez campos que existen para ser configurados son:

- Identification.
- Association.
- Communication from the robot.
- Communication to the robot.
- Functionable.
- Position.
- Robot Connection.
- Scripts.
- Tool Settings.
- Visualization.

Donde los campos Identification, Communication from/to the robot, Position, Robot Connection y Tool Settings son modificados de acuerdo con las características del robot real y se muestra en la figura 11-2.

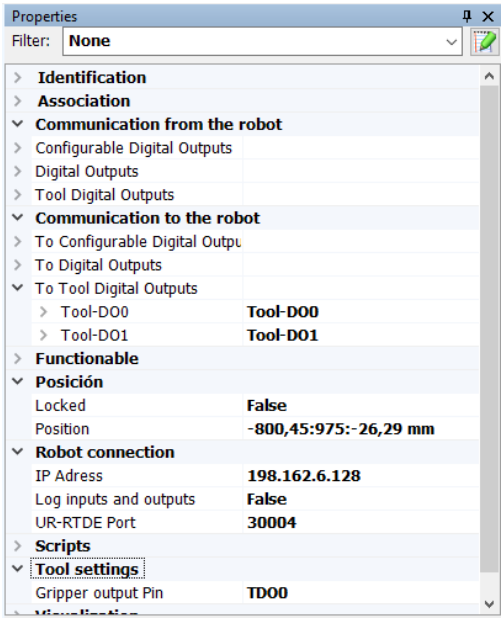
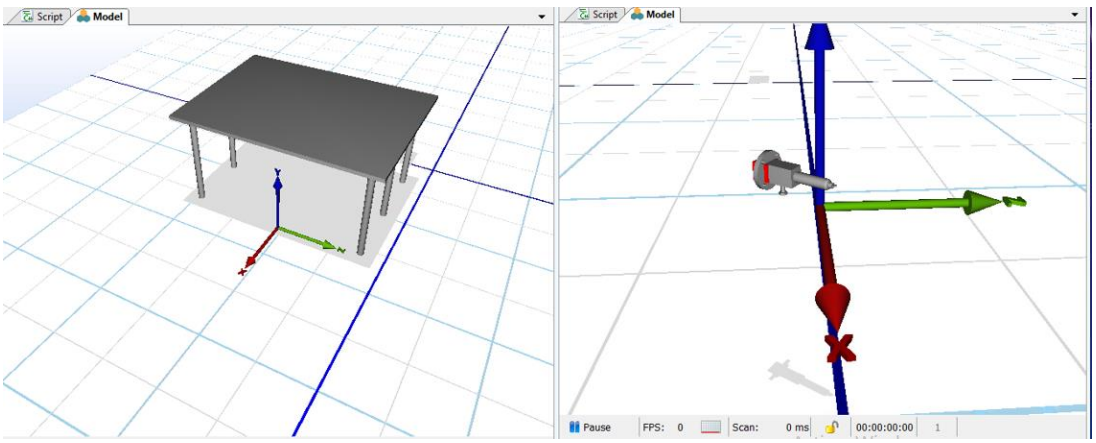


Figura 11-2. Configuraciones del robot.

Realizado por: Alexis, M.; Sofia, S.2020

El efector final "Gripper" como de la mesa fueron modelados en el software AutoCAD donde se asigna el material de construcción, aluminio, y sus dimensiones las cuales son a escala real. Los archivos .dxf, que genera AutoCAD, son exportados de forma directa hacia *Experior* como se observa en la figura 12-2 aquí se requiere configurar un nuevo catálogo que contenga este nuevo modelo, lo que se consigue ingresando a *My Catalog* y accediendo a la opción *Add New Tool.*



(a) (b)

Figura 12-2. (a) Mesa de trabajo. (b) Gripper para marcado de piezas.

Realizado por: Alexis, M; Sofia, S.2020

En el caso de la mesa de trabajo se asignarán sus propiedades mediante código C# subyacente a *Experior*, donde, al ser un objeto estático inanimado, las funciones que se asignan son las más básicas como se observa en la figura 13-2.

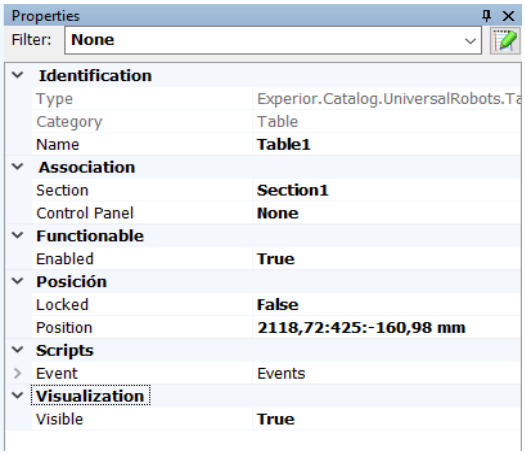


Figura 13-2. Propiedades de la mesa de trabajo dentro de Experior

Realizado por: Alexis, M.; Sofia, S.2020

El Gripper está relacionado al funcionamiento del robot por lo que es necesario asignar comportamiento al mismo. Una de las funciones necesarias en su programación es *Grabbing Properties* que permite un acople entre objetos. En este caso, es necesario aumentar un módulo de ubicación y orientación que contiene la posición z, x, e y en mm como los ángulos yaw, pitch y roll en grados que junto a la función mencionada trasmitirán el movimiento del efector final del robot hacia el Gripper. Para esto solo es necesario unir el robot con el Gripper y el acople se da de forma automática al igual que su movimiento. La figura 14-2 muestra las propiedades del Gripper.

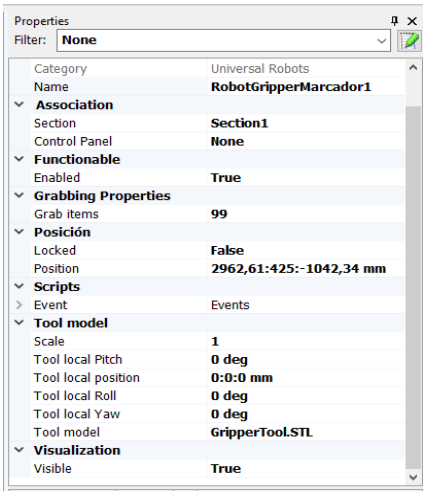


Figura 14-2. Propiedades del Gripper dentro de *Experior.*

Realizado por: Alexis, M.; Sofia, S.2020

Por último, al tratarse de un entorno cambiante la plancha es diseñada y programada en C# ya que sus dimensiones se modifican según los recursos disponibles, figura 15-2.

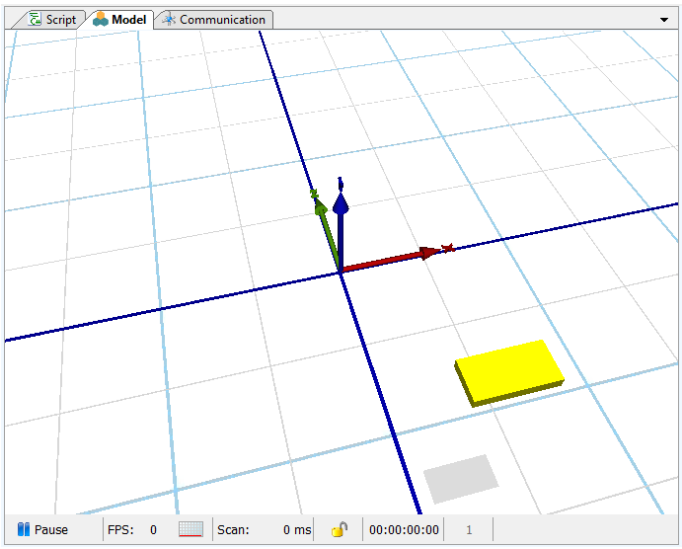


Figura 15-2. Diseño de la plancha en Experior.

Realizado por Alexis, M.; Sofia, S.2020

La plancha, como se denotó en párrafos anteriores, está ubicada y orientada con respecto al sistema de referencia del robot y localizada sobre el plano de trabajo (mesa). Por tanto, se asigna una posición alterna que contenga la ubicación del robot dentro de *Experior* para que la plancha este referenciada con respecto a la base del robot como se ve en la figura 16-2.

Posición	
Locked	**True**
Position	**221,1:945:-1269,27 mm**
Robot Info. [mm]	
Robot X	**-600**
Robot Y	**905**
Robot Z	**-600**

Figura 16-2. Posición de la plancha e información de la ubicación del robot.

Realizado por: Alexis, M.; Sofia, S.2020

La plancha en *Experior* tiene que ser simulada digitalmente y representar la realidad. Para esto, es necesario adquirir datos los cuales son obtenidos por la cámara y se envían a *Experior* mediante protocolo Modbus creado dentro de PolyScope, lo que se explica en la fase de implementación. Dentro de *Experior* se configura un módulo de entrada que contenga el ángulo, la posición x e y del centro de la plancha y dos variables booleanas que serán de utilidad para la programación.

Además, en *Experior* es necesario crear una comunicación Modbus, con la dirección IP del robot, con este protocolo se conecta el módulo de entradas con los datos enviados de PolyScope. A cada

entrada se asigna una dirección y el nombre de la variable a la cual se va a conectar como se observa en la figura 17-2.

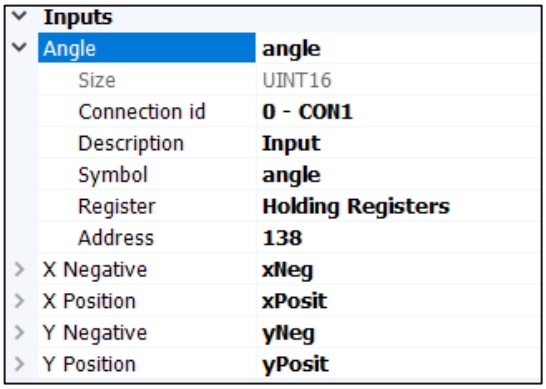


Figura 17-2. Módulo de comunicaciones para la plancha.

Realizado por: Alexis, M; Sofia, S.2020

2.2.4 Fase Implementación

2.2.4.1 Manejo de la cámara y el software Camera Locate

El software Camera Locate ofrece una interfaz muy intuitiva para poder configurar el sistema de visión artificial, a este se accede por la interfaz principal de PolyScope.

La función que se configura es la enseñanza de objetos paramétricos. Este método cosiste en un modelo basado en parámetros de una forma 2D básica (círculo, anillo, cuadrado y rectángulo) lo que se busca es que el sistema reconozca y ubique con gran precisión objetos que tienen características semejantes. Una vez que se define la posición de la fotografía el operador puede utilizar el nodo de la localización de la cámara dentro de un programa al igual que la variable *object_location* que contiene la información de ubicación y rotación.

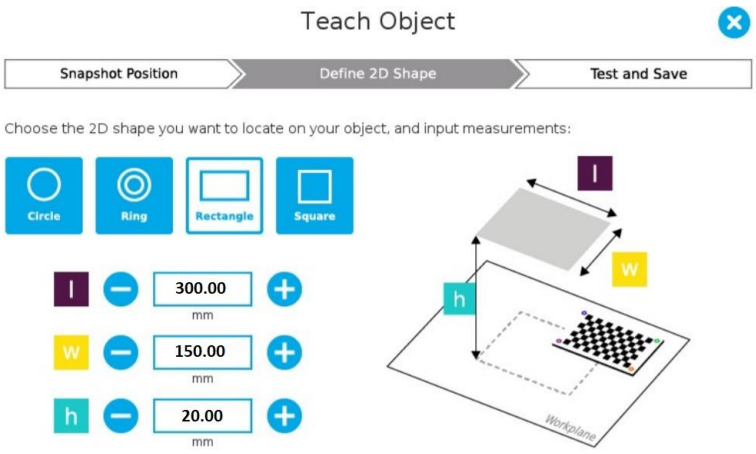


Figura 18-2. Configuración de las dimensiones de la plancha.

Realizado por: Alexis, M.; Sofia, S.2020

En nuestro caso de estudio se configura el sistema para que reconozca la plancha rectangular de dimensiones 300x150 [mm] como se muestra en la figura 18-2.

A continuación, se ubica la plancha en diferentes posiciones donde la cámara toma capturas y mediante un algoritmo de detención de bordes el software aprende a localizar la ubicación y rotación de la plancha. Luego se crea el nodo Camera Locate dentro del entorno de programación de PolyScope como se muestra en la figura 19-2 donde se creó un programa de prueba capaz de reconocer la ubicación y rotación de varias piezas y ponerlas en otro espacio de forma ordenada.

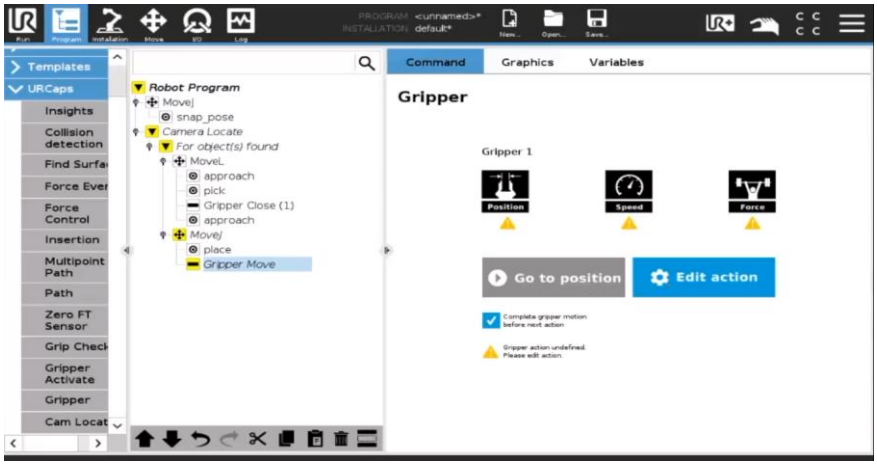


Figura 19-2. Programa de prueba con reconocimiento de posición y orientación.

Realizado por: Alexis, M.; Sofia, S.2020

2.2.4.2 Código en PolyScope

El gráfico 4-2 muestra el diagrama de flujo que describe la secuencia lógica para realizar el proceso de marcado de piezas.

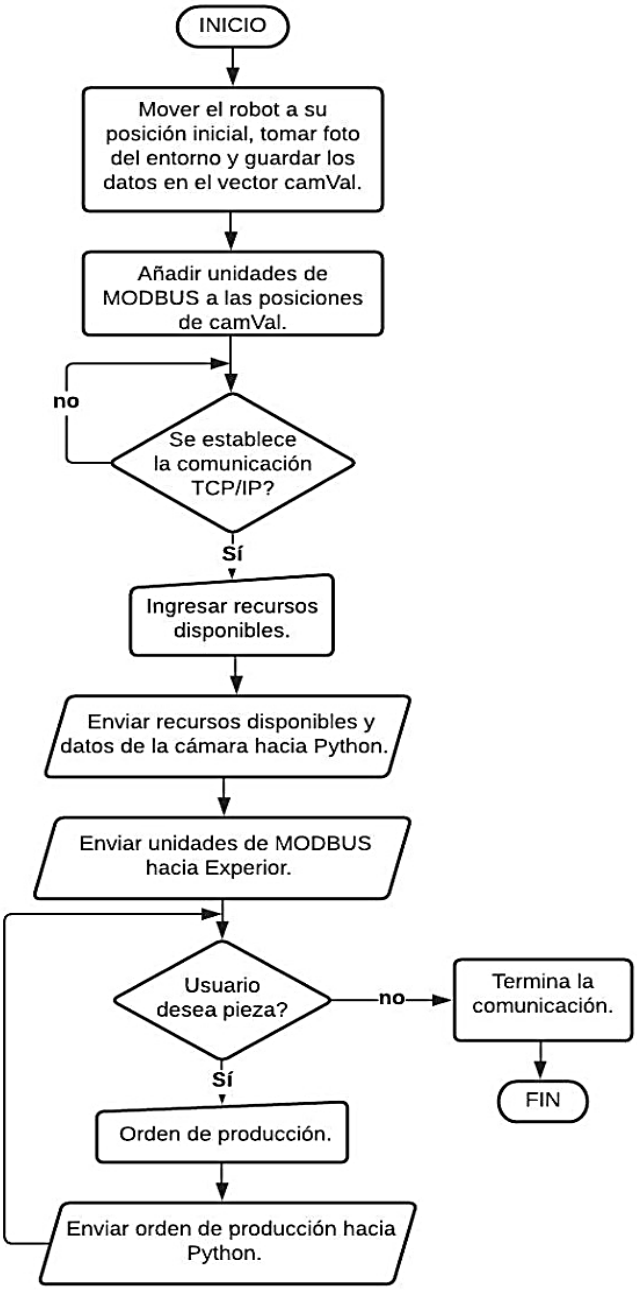


Gráfico 4-2. Diagrama de flujo del algoritmo en PolyScope.

Realizado por: Alexis, M.; Sofia, S.2020

El programa realizado en PolyScope es utilizado para la adquisición de datos como: requerimiento de corte, recursos disponibles y la percepción del entorno.

El programa comienza ubicando al robot en la posición establecida para que la cámara pueda captar el entono.

Para trabajar con los datos, estos deben ser enviados desde el robot hacia un ordenador para ello se utiliza una comunicación cliente-servidor TCP/IP usando el brazo robótico UR3 como cliente. Inicialmente se abre una conexión entre el robot y la computadora llamada “Socket”, dentro del

cual se agrega la dirección IP del robot, un puerto arbitrario que no esté siendo usado por el otro dispositivo y el nombre del programa que administrará el socket en el otro dispositivo (en este caso, dicho programa fue realizado también en Python) al cual se va a conectar el robot como se mira en la figura 20-2.

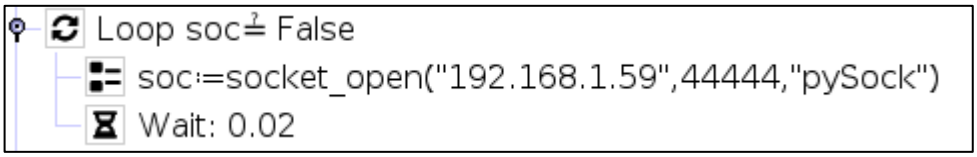


Figura 20-2. Creación de socket en PolyScope.

Realizado por: Alexis, M.; Sofia, S.2020

El entorno dentro *Experior* necesita los datos obtenidos por la cámara para que la plancha simulada represente fielmente la ubicación y orientación de la real. Los datos tienen que enviarse desde el robot hacia *Experior* lo cual se consigue usando protocolo MODBUS. Esta configuración se encuentra dentro del menú *Installation – Fieldbus – MODBUS* dentro de PolyScope. Para esta configuración, se asigna la misma dirección IP del robot, se crean tres *Register Output* para enviar datos numéricos que contengan las posiciones en los ejes X e Y y el ángulo en el que se encuentra orientada la plancha. Los datos deben ser tratados antes de ser enviados ya que *Experior* solo trabaja con datos enteros y positivos para lo cual se crean dos *Digital Output* que indiquen cuando las posiciones de la plancha sean negativas. A cada unidad de MODBUS creada se le asigna una dirección arbitraria como se muestra en la figura 21-2.

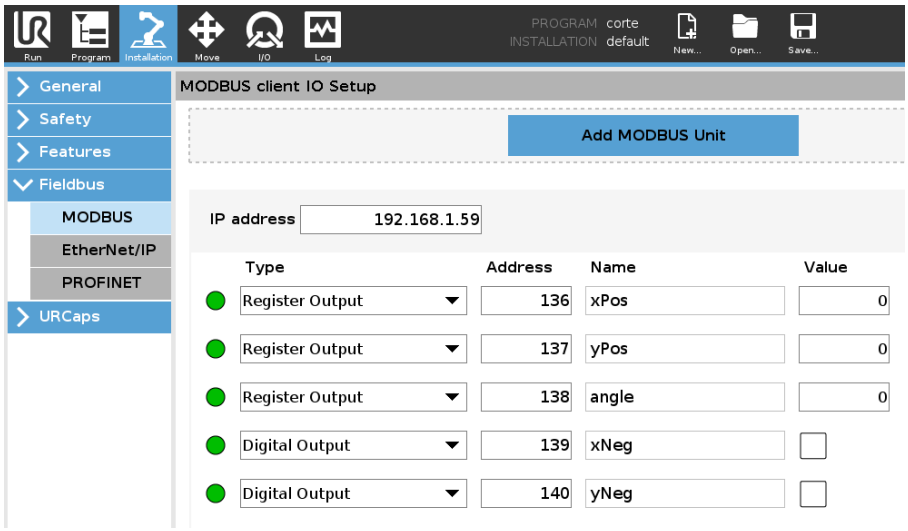


Figura 21-2. Configuración de las salidas MODBUS para *Experior*.

Realizado por: Alexis, M.; Sofia, S.2020

La entrada de los recursos disponibles y los requerimientos de corte se dan mediante *Assignment* configurados como *Operator*. Esto nos permite crear cuadros de entrada de datos y guardarlos como variables para luego ser utilizadas como se muestra en la figura 22-2.

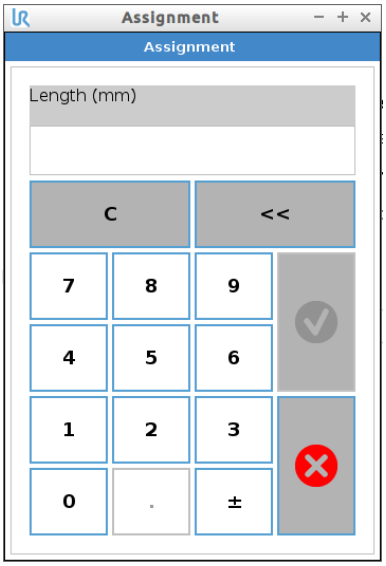


Figura 22-3. Cuadro de entrada de datos.

Realizado por: Alexis, M.; Sofia, S.2020

Para enviar los datos hacia Python se usa la función *Socket_sent_string*(), la cual envía los datos como una cadena de caracteres. Lo que se pretende es crear un vector que incluya los datos de la cámara, el tamaño de la plancha y todos los requerimientos de corte para que el algoritmo en Python devuelva como resultado el movimiento que tiene que realizar el robot.

Sin embargo, PolyScope no posee la capacidad de enviar una gran lista de datos concatenados por lo que se envía dato por dato separados por punto y coma como muestra la figura 23-2. Después, en Python se creará un vector concatenado con todas las variables.

```
socket_send_string(lenT, "pySock")
socket_send_string(";", "pySock")
socket_send_string(widT, "pySock")
socket_send_string(";", "pySock")
socket_send_string(camVal[0]*1000, "pySock")
socket_send_string(";", "pySock")
socket_send_string(camVal[1]*1000, "pySock")
socket_send_string(";", "pySock")
socket_send_string(camVal[2]*1000, "pySock")
socket_send_string(";", "pySock")
socket_send_string(floor(r2d(camVal[3]*(-1))), "pySock")
```

Figura 23-2. Envío de datos hacia Python

Realizado por: Alexis, M.; Sofia, S.2020

Finalmente, se usa la función socket_close( ) que termina la comunicación entre el brazo robótico y la computadora dando paso al algoritmo de corte desarrollado en Python que posteriormente controlará el robot enviando comandos de movimiento a PolyScope. El algoritmo está disponible en el anexo A.

2.2.4.3 Desarrollo del Código en Python

El diagrama de flujo mostrado en el gráfico 5-2, describe la secuencia lógica para realizar el proceso de marcado de piezas.

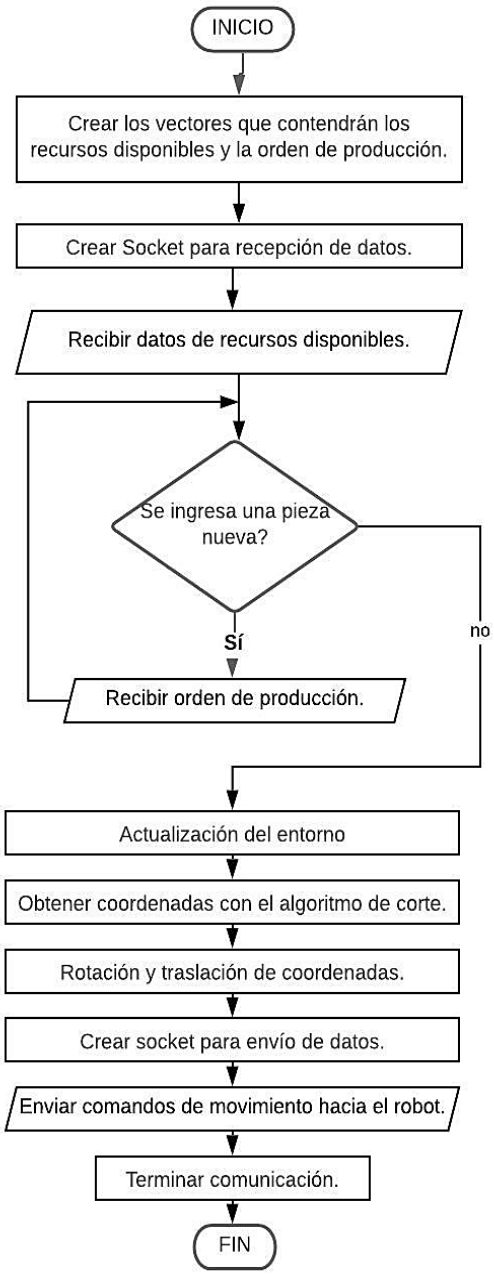


Gráfico 5-2. Diagrama de flujo del algoritmo en Python desarrollado para el marcado de piezas.

Realizado por: Alexis, M.; Sofia, S.2020

El programa realizado en Python complementa el proceso de marcado para corte de superficies, el mismo que es utilizado para la recepción de datos generados en PolyScope, tratar los datos mediante el algoritmo de corte, manipular los resultados del algoritmo para que se ajusten al ambiente de trabajo real y, finalmente, poner en funcionamiento al robot real.

Las librerías que requiere el programa son:

- Math()
- Time()
- Greedepacker()
- Socket()
- Matplotlib()
- Numpy()

Las cuales contiene funciones que serán de utilidad en el desarrollo del programa que será descrito a continuación:

Paso 1

El programa da inicio creando 4 vectores

- datT = [] Lista con los datos del tablero [Largo, Ancho, posX, posY, posZ, Ángulo]
- datP = [] Lista con los datos de las piezas [numPiezas, LargoPieza, AnchoPieza]
- 56untos = [] Lista con los puntos en X de las piezas
- puntosY = [] Lista con los puntos en Y de las piezas

En los dos primeros vectores se guarda los recursos disponibles y requerimientos de producción, en los siguientes se crean vectores que contenga las posiciones finales de las esquinas de cada uno de los cortes que tiene que realizar el brazo robótico UR3.

Paso 2

A continuación, se inicia la comunicación TCP/IP con Python como servidor y el robot como cliente. Se crea dos tuplas que representan la conexión tanto para la recepción y el envío de datos, la primera tupla contiene la dirección IP del cliente y el puerto de conexión que es “44444” y la segunda tiene la misma dirección IP con el puerto “30002” que está abierto para *Script Programming* en el robot. La figura 24-2 muestra la configuración de las funciones obtenidas de la librería socket() para realizar la comunicación.

```
robotIP = "192.168.1.59";   # Dirección IP del Robot UR
puertoEN = 30002;           # Puerto abierto del Robot para Script programing
puertoRE = 44444;           # Puerto arbritario para recibir información

conEnvi = (robotIP, puertoEN);  # Tupla que representa la conexión de envío
conRece = (robotIP, puertoRE);  # Tupla que representa la conexión de recepción

sRE = socket.socket(socket.AF_INET, socket.SOCK_STREAM);   # Crear socket para recepción
sRE.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1);
sRE.bind(conRece);          # Une el socket a la dirección IP
print("[Conectando] ....");
sRE.listen(5);              # Deja a la espera de conexión
c, add = sRE.accept();      # Conecta con el Robot a través del Socket creado de recepción
print("[Conectado] Conexión establecida con URobot en {} ".format(robotIP));
```

Figura 24-2. Configuración de las funciones para comunicación TCP/IP con el robot.

Realizado por: Alexis, M.; Sofia, S.2020

Paso 3

Una vez establecida la comunicación comienza la recepción de datos. En primera instancia, se debe ingresar los recursos disponibles y guardarlos en el vector *datT*. Luego, se crea un bucle de recepción donde se concatena todos los requerimientos de cada una de las piezas que se desea cortar y se guarda en el vector *datP* como muestra la figura 25-2.

```
print("[Esperando recepción de datos]");
count = 0;
while count < 150:
  try:
    msg = c.recv(1024).decode('utf-8');
    print("[Recivido] De {}: {}".format(robotIP, msg));
    if msg == FIN:
      break;
    elif count == 0:
      datT = list(map(int, msg.split(";")));
    else:
      datP.append(list(map(int, msg.split(";"))));
    time.sleep(0.5);
  except Exception as e:
    print(str(e));
  count += 1;

print("[FIN] Fin de la recepción");
c.close();
sRE.close();
```

Figura 25-2. Recepción de datos, orden de producción y recursos disponibles.

Realizado por: Alexis, M.; Sofia, S.2020

Posteriormente, se asignan los datos a variables, se busca la coordenada de la esquina inferior izquierda de la plancha para que el algoritmo de corte tenga referencia y se configura el ángulo

de giro para poder resolver las dos posibles orientaciones, horizontal o vertical, en las que se puede configurar la plancha.

Paso 4

Una vez este proceso se ha realizado se da paso al algoritmo de corte el cual da como resultado cinco pares ordenados para cada una de las piezas, los cuales representan las esquinas repitiéndose el primero ya que el robot tiene que regresar al punto desde el que parte. La función greedypacker.BinManager() permite elegir el tipo de algoritmo de corte en este caso "*guillotine*" y la heurística "*best_longside*" como se observa en el anexo B.

El conjunto de pares ordenados tiene que ser rotado y trasladado hacia la posición y orientación de la plancha real lo que se consigue programando las ecuaciones 3-2 y 4-2 previamente explicadas.

Paso 5

Para poder enviar los datos hacia el robot se crea un nuevo socket de envió el cual se configura con la tupla para *Script Programming* esto permite que Python se conecte directamente con el robot y pueda programar su movimiento sin la necesidad de la interfaz PolyScope. Para esto dentro de Python se programó dos comandos "movel" con los argumentos requeridos que harán líneas de corte rectas al momento de formar los rectángulos como muestra la figura 26-2. Por último, se cierra el socket para terminar la comunicación y el robot se pone en marcha.

```
print("[Enviando puntos] .... ");
timeS = 1.2;
num = 0;
for rectangulo in rect:
  pR = rotarTrasP(rectangulo[0]);
  cmd = "movel(p[{}, {}, {}, 0, 0, 0], a=1.4, v=1.1, t={})".format(pR[0]/1000, pR[1]/1000, pCz/1000 + 0.04, timeS+1) + "\n";
  sEN.send(cmd.encode());
  time.sleep(timeS*2);
  for p in rectangulo:
    p = rotarTrasP(p);
    cmd = "movel(p[{}, {}, {}, 0, 0, 0], a=1.4, v=1.1, t={})".format(p[0]/1000,p[1]/1000,pCz/1000,timeS) + "\n";
    sEN.send(cmd.encode());
    time.sleep(timeS+0.8);
    cmdT = "set_tool_digital_out(0,True)" + "\n";
    sEN.send(cmdT.encode());
    time.sleep(0.2);
    cmdT = "set_tool_digital_out(0,False)" + "\n";
    sEN.send(cmdT.encode());
    time.sleep(0.2);
  num += 1;
  print("[Completado] Rectangulo {} formado ...".format(num));

print("[Completado] Todos los rectangulos se han formado");
cmd = "movel(p[{}, {}, {}, 0, 0, 0], a=1.4, v=1.1, t={})".format(0.15,0.1,0.25,timeS) + "\n";
sEN.send(cmd.encode());
```

Figura 26-2. Programación en Python para controlar el robot.

Realizado por: Alexis, M.; Sofia, S.2020

2.2.5 Fase Digital Twin (Puesta en funcionamiento)

Para poner en funcionamiento el *Digital Twin* del brazo robótico UR3 en el marcado de piezas para posterior corte se configura el entorno virtual dentro de *Experior* y la máquina virtual Ursim.

En primer lugar, se ubica el brazo robótico UR3 en el espacio de trabajo. Para configurar el mismo en la pestaña *Propierties* en la sección *Position* se asigna la coordenada (-600;905;-600) [mm] que representa el lugar espacial donde está ubicada la base del robot y en la sección *Robot Communication* se asigna la dirección IP del robot real. Para terminar la configuración del brazo robótico se selecciona el *Gripper* y se conecta al efector final del robot.

Para configurar la plancha se ingresa a la pestaña *Communication* y se crea una unidad de MODBUS con la dirección IP del robot de donde se va a recibir la información (posición y rotación). Una vez creada esta unidad, dentro de las configuraciones de la plancha se forma un inciso donde se ingresan las direcciones de MODBUS creadas previamente en PolyScope. A continuación, se ingresan todas las configuraciones de acuerdo con lo explicado en la sección 2.2.3.3. En la figura 27-2 se presenta el entorno de *Experior* configurado.

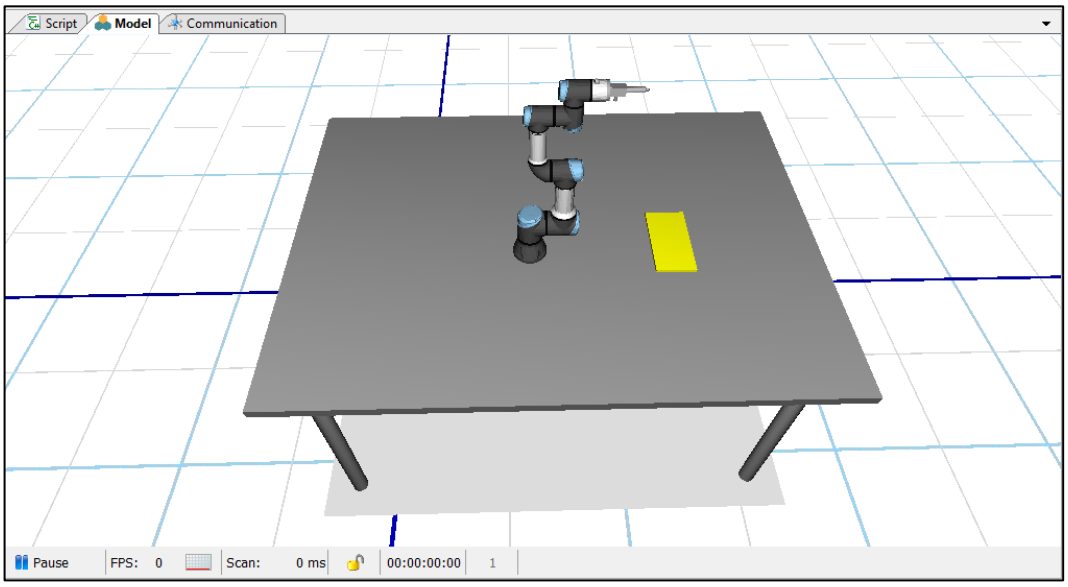


Figura 27-2. Entorno configurado dentro de Experior

Realizado por: Alexis, M.; Sofia, S.2020

Para establecer la comunicación y trasmitir la información necesaria entre Ursim (ubicada en el sistema operativo Ubuntu, sobre una máquina virtual) y *Experior* (instalado en Windows) es necesario relacionar las dos redes mediante un *Puente de Red*.

Una vez que inicia la máquina virtual en PolyScope se ejecuta el programa, el cual espera hasta que exista comunicación con el servidor por lo que es necesario dar inicio al programa desarrollado en Python. A continuación, PolyScope muestra los cuadros de entrada donde se ingresa los diferentes datos como: el tamaño de la plancha, número de piezas a cortar y las dimensiones de las piezas. Terminado el ingreso el algoritmo de corte genera los puntos y envía comandos de movimiento que pondrán en funcionamiento al robot junto con el *Digital Twin* hasta finalizar los cortes y así terminar con todo el proceso.

CAPÍTULO III

3 GESTIÓN DEL PROYECTO Y EVALUACION

3.1 Gestión de Proyecto

3.1.1 Cronograma

El cronograma de la tabla 1-3 muestra la relación actividad vs tiempo de las actividades que se realizaron durante el desarrollo de la interfaz, se especifica el tiempo de duración de cada una de las etapas. Se utilizó como herramienta un Diagrama de Gantt.

Tabla 1-3. Cronograma de actividades.

ACTIVIDADES		MES 1				MES 2				MES 3				MES 4				MES 5				MES 6			
		S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
Análisis, selección y organización del tema.	P																								
	R																								
Identificación de los requerimientos del sistema.	P																								
	R																								
Diseño de la interfaz Digital Twin.	P																								
	R																								
	P																								

Programación de la interfaz.	R																									
Puesta en marcha.	P																									
	R																									
Pruebas y Evaluación de la interfaz.	P																									
	R																									
Entrega del proyecto	P																									
	R																									

Realizado por: Alexis, M.; Sofia, S.2020.

3.1.2 Recursos y Materiales

El presupuesto para el desarrollo del proyecto es detallado en la tabla 2-3 tomando en cuenta todos los equipos, materiales y todo lo que tuvo que ver con la erogación de recursos económicos; se consideró un 10% del valor total para gastos extras.

Tabla 2-3. Tabla de costos.

Cantidad	Material	Precio Unitario	Precio Total
1	Brazo Robótico UR3	23000	23000
1	Wrist Camera	265	265
2	Licencias del software Experior	75	150
	Otros Gastos (10% gasto total)	15	15
		TOTAL	23430

Realizado por: Alexis, M.; Sofia, S.2020.

El financiamiento fue cubierto por los proponentes del trabajo de investigación, el brazo robótico UR3 y la Wrist Camera son parte de los bienes de las Facultad de Informática y Electrónica.

3.2 Evaluación

En la presente sección se describen los resultados obtenidos en la investigación comenzando por el algoritmo de corte implementado evaluado en diferentes escenarios, se detallan también los resultados del giro y adaptación al ambiente de trabajo para comprobar la fidelidad de las coordenadas enviadas al robot para su puesta en funcionamiento. Por último, se obtiene datos de tiempo de latencia para calcular el retardo existente entre el *Digital Twin* y el robot.

Cabe recalcar que por motivos de la emergencia sanitaria por el Covid 19 iniciada en el país el 12 de marzo del 2020 y presente hasta la fecha el ingreso a la Escuela Superior Politécnica de Chimborazo fue restringido, por ende, los resultados fueron obtenidos a través de simulaciones del brazo robótico UR3. Utilizando Ursim se resguardo la veracidad de los resultados ya que es una fiel copia del robot real avalada y distribuida por *Universal Robots*.

3.2.1 Resultados del algoritmo de corte con diferentes medidas

El código y la heurística implementados son el algoritmo *de Guillotine* y *Best long side* respectivamente. Para poder evaluar el algoritmo implementado en primera instancia se ingresan una determinada cantidad de piezas sin sobrepasar el tamaño de la plancha que las contendrá.

La tabla 3-3 muestra los datos ingresados y los resultados obtenidos en milímetros. Para mostrar la gráfica de los resultados dentro de Python se utiliza la librería Matplotlib y la programación que muestra en anexo C.

Tabla 3-3. Datos de ingreso y respuesta del algoritmo de corte

Datos ingresados			Resultado del algoritmo			
Ítem	Dimensiones (mm)		Coordenadas (mm)			
	Width	Length	Esquina 1	Esquina 2	Esquina 3	Esquina 4
Plancha	300	150	(0, 0)	(0, 150)	(300, 150)	(300, 0)
Pieza 1	100	50	(100, 0)	(200, 0)	(200, 50)	(100, 50)
Pieza 2	100	100	(0, 0)	(100, 0)	(100, 100)	(0, 100)
Pieza 3	50	50	(200, 0)	(250, 0)	(250, 50)	(200, 50)
Pieza 4	50	25	(250, 0)	(300, 0)	(300, 25)	(250, 25)

Realizado por: Alexis, M.; Sofia, S.2020.

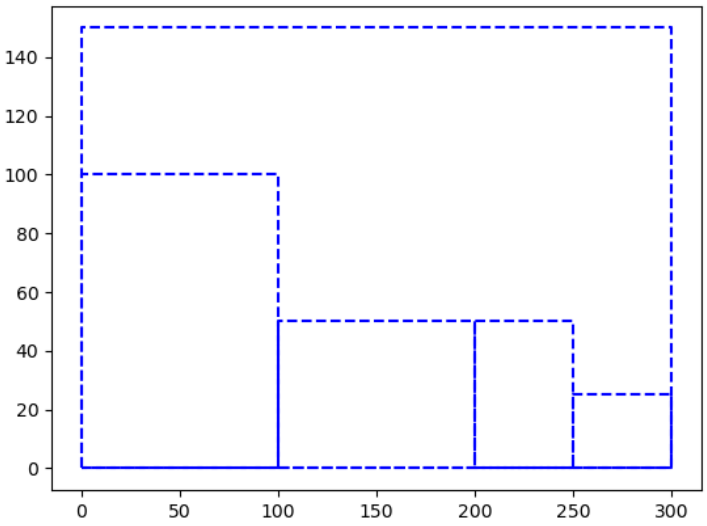


Figura 1-3. Gráfica generada por el algoritmo de corte implementado.

Realizado por: Alexis, M.; Sofia, S.2020

Como se observa en los resultados numéricos y gráficos el algoritmo convergió en una solución óptima para el problema propuesto utilizando de manera adecuada el recurso a su disposición respondiendo así a uno de los objetivos principales del algoritmo que es maximizar la salida.

A continuación, se plantea un problema en el cual la suma del área de las piezas sea igual al de la plancha, luego se compara los resultados obtenidos por el algoritmo implementado con otros algoritmos disponibles.

La tabla 4-3 muestra los datos de ingreso a los algoritmos mientas que la figura 2-3 representa una solución gráfica, diseñada de forma manual, que muestra el perfecto acople de las piezas a la plancha.

Tabla 4-3. Datos de ingreso hacia el algoritmo

Datos ingresados		
Ítem	Dimensiones (mm)	
	Width	Length
Plancha	300	150
Pieza 1	100	100
Pieza 2	100	50
Pieza 3	50	150
Pieza 4	50	50
Pieza 5	50	100
Pieza 6	50	150
Pieza 7	50	75
Pieza 8	50	75

Realizado por: Alexis, M.; Sofia, S.2020

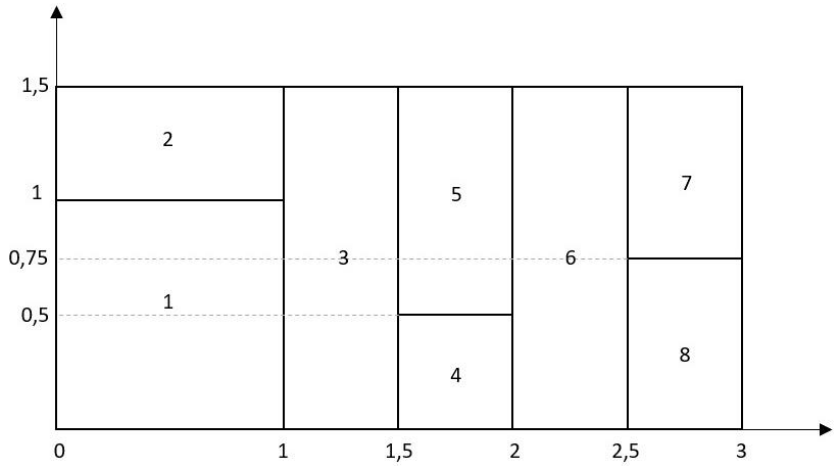


Figura 2-3. Distribución de las piezas dentro de la plancha de forma manual.

Realizado por: Alexis, M.; Sofia, S.2020

El algoritmo de corte guillotina resuelve el problema ubicando todas las piezas dentro de la plancha como muestran los resultados numéricos y gráficos mostrados en la figura 3-3 y la tabla 5-3, esto representa una solución adecuada para nuestro caso de estudio.

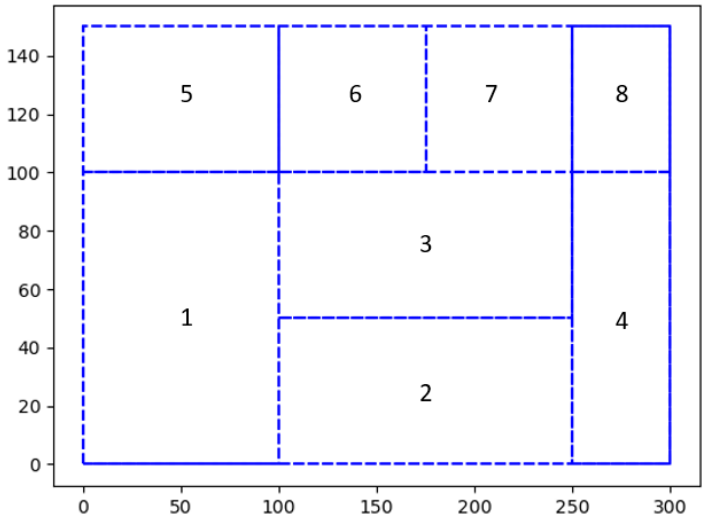


Figura 3-1. Respuesta grafica del algoritmo implementado.

Realizado por: Alexis, M.; Sofia, S.2020

Tabla 5-1. Resultado numérico del problema propuesto.

Resultado del algoritmo guillotina				
Ítem	**Coordenadas (mm)**			
	Esquina 1	**Esquina 2**	**Esquina 3**	**Esquina 4**
Plancha	(0, 0)	(300, 0)	(300, 150)	(0, 150)
Pieza 1	(0, 0)	(100, 0)	(100, 100)	(0, 100)
Pieza 2	(100, 0)	(250, 0)	(250, 50)	(100, 50)
Pieza 3	(100, 50)	(250, 50)	(250, 100)	(100, 100)
Pieza 4	(250, 0)	(300, 0)	(300, 100)	(250, 100)

Pieza 5	(0, 100)	(100, 100)	(100, 150)	(0, 150)
Pieza 6	(100, 100)	(175, 100)	(175, 150)	(100, 150)
Pieza 7	(175, 100)	(250, 100)	(250, 150)	(175, 150)
Pieza 8	(250, 100)	(300, 100)	(300, 150)	(250, 150)

Realizado por: Alexis, M.; Sofia, S.2020

Los resultados gráficos mostrados a continuación en la figura 4-3 corresponden a la solución del mismo problema planteado para tres diferentes algoritmos, donde se puede inferir que el algoritmo *Shelf* y *Skyline* no logran ubicar todas las piezas debido a su lógica de programación mientras que el algoritmo de *Maximal Rectangles* llega hacia una solución óptima ya que se trata de un proceso complejo al igual que su programación por ende tiene un costo computacional alto razón por la cual el algoritmo elegido fue Guillotine el cual nos da buenos resultados a un costo computacional bajo. Estos resultados se obtuvieron del sistema de gestión de proyectos GitHub en el apartado de pruebas para algoritmos de corte que se lo puede encontrar en la siguiente referencia (Bothwell Solomon 2019).

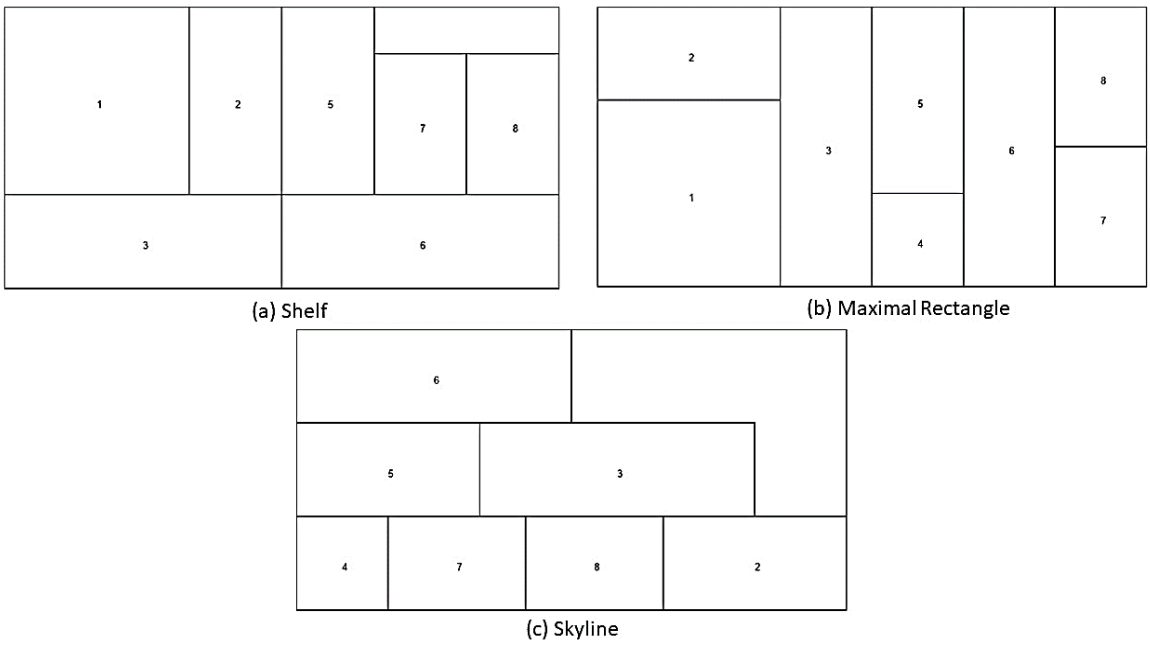


Figura 4-3. (a) Resultado gráfico del algoritmo *Shelf*. (b) Resultado gráfico del algoritmo *Maximal Rectangle*. © Resultado gráfico del algoritmo *Skyline.*

Realizado por: Alexis, M.; Sofia, S.2020

3.2.2 Pruebas de rotación y traslación

La adaptación al medio es un punto clave en el desarrollo de la investigación, este factor determina el éxito o fracaso del corte de las piezas. Las pruebas desarrolladas a continuación evalúan las ecuaciones planteadas en la sección 2.2.3.2 de este documento.

Al espacio de trabajo llega la plancha en una posición y orientación arbitraria, la cámara es la encargada de recopilar esta información. Es importante tener en consideración como se realiza este proceso para posteriormente tratar los datos, el video explicativo está disponible en la siguiente referencia (Mariño y Sanchez 2020).

Una vez el usuario ingresa la orden de producción el algoritmo de corte inicia su proceso hallando así la distribución de piezas más adecuada. Continuando con el último ejemplo las coordenadas generadas por el algoritmo de guillotina tienen que ser ubicadas conforme se encuentra la pieza. Para probar el algoritmo se plantea dos escenarios.

Primero, la ubicación del centro de la pieza es (200,200) mm correspondientes al X e Y del sistema de referencia de la base del robot mientras que el eje Z se estima dependiendo del grosor de la plancha a la cual se le asigna 4 mm, la rotación es $\pi/4$ [rad] y para las dimensiones de la plancha se considera que el ancho es mayor que el largo. De esta forma la variable que guarda la cámara es:

$$object\_location = [200, 200, 4, 0.785, 0.785, 0.000\ ]$$

Lo resultados mostrados a continuación, en la tabla 6-3, son los devueltos por el algoritmo utilizando los datos del ambiente descritos.

Tabla 6-3. Resultados de la adaptación al entorno con un giro de $\boldsymbol{\pi/4}$ [rad].

Resultado de la adaptación al entorno				
Ítem	**Coordenadas (mm)**			
	Esquina 1	**Esquina 2**	**Esquina 3**	**Esquina 4**
Plancha	(146.966, 40.900)	(359.099, 253.033)	(253.033, 359.099)	(40.900, 146.966)
Pieza 1	(146.966, 40.900)	(217.677, 111.611)	(146.966, 182.322)	(76.256, 111.611)
Pieza 2	(217.677, 111.611)	(323.743, 217.677)	(288.388, 253.033)	(182.322, 146.966)
Pieza 3	(182.322, 146.966)	(288.388, 253.033)	(253.033, 288.388)	(146.966, 182.322)
Pieza 4	(323.743, 217.677)	(359.099, 253.033)	(288.388, 323.743)	(253.033, 288.388)
Pieza 5	(76.256, 111.611)	(146.966, 182.322)	(111.611, 217.677)	(40.900, 146.966)
Pieza 6	(146.966, 182.322)	(200.0, 235.355)	(164.644, 270.710)	(111.611, 217.607)
Pieza 7	(200.0, 235.355)	(253.033, 288.388)	(217.677, 323.743)	(164.644, 270.710)
Pieza 8	(253.033, 288.388)	(288.388, 323.743)	(253.033, 359.099)	(217.677, 323.743)

Realizado por: Alexis, M.; Sofia, S.2020.

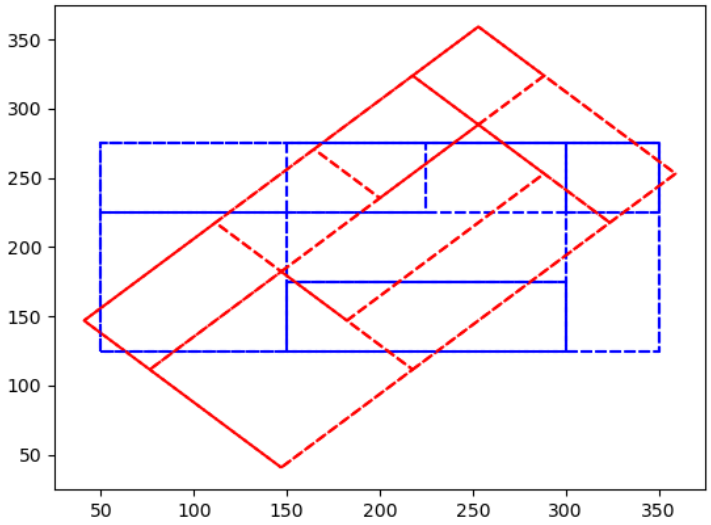


Figura 5-3. Resultado gráfico de la ubicación de las piezas respecto al entorno con un giro de $\pi/4$ [rad].

Realizado por: Alexis, M.; Sofia, S.2020

Segundo, se evalúa el mismo ejemplo, pero esta vez se asume que el largo es mayor que el ancho dando como respuesta gráfica la figura 6-3.

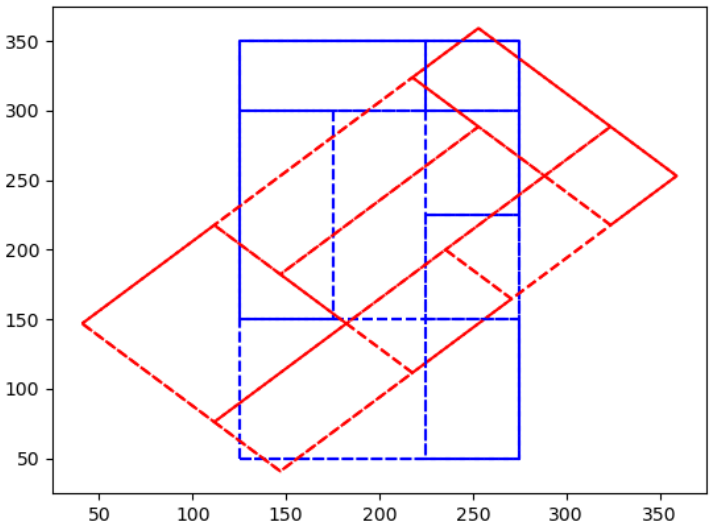


Figura 6-3. Resultado grafico de las piezas respecto al entorno con otra distubucion del tamaño con un giro de $\pi/4$ [rad].

Realizado por: Alexis, M.; Sofia, S.2020

3.2.3 Resultado de la adaptación al entorno dentro de Experior.

3.2.3.1 Comunicación entre PolyScope y Experior

Primero se verifica una comunicación exitosa entre PolyScope y Experior. Si el robot y la computadora se encuentran en la misma red Experior no presenta ningún problema al momento de conectar el módulo MODBUS programado.

El tráfico existente se comprueba en la ventana comunicaciones la cual muestra los paquetes de Holding Registers que son las entradas de las configuraciones de la pieza que corresponden a las salidas MODBUS en PolyScope, la figura 7-3 muestra en código binario el valor del centro de la pieza y el ángulo de giro enviado desde PolyScope, comprobando así la correcta comunicación entre estas dos plataformas.

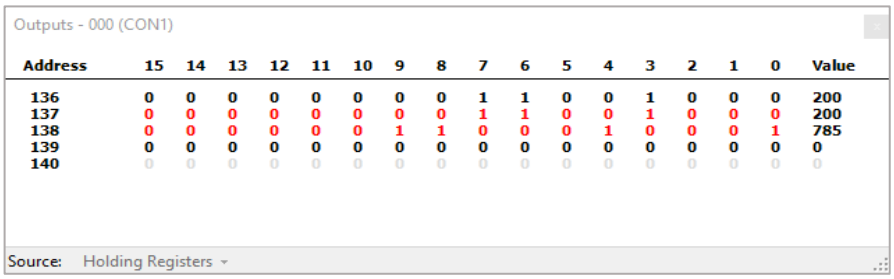


Figura 7-3. Datos de conectividad del ángulo y la ubicación del centro de la pieza.

Realizado por: Alexis, M.; Sofia, S.2020

Para calcular el tiempo de latencia presente en la comunicación tanto en la adaptación del entorno como en el movimiento del *Digital Twin* se realiza una prueba enviando paquetes de datos desde PolyScope hacia Experior y almacenando el tiempo que tarda el mismo en responder.

La figura 8-3 muestra el histograma de los datos de tiempo de latencia recaudados teniendo una muestra de cuarenta datos divididos en seis intervalos igualmente espaciados.

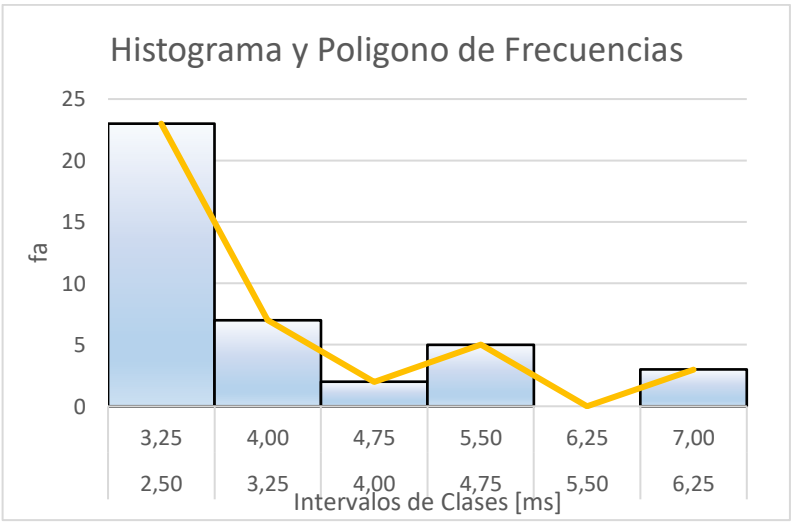


Figura 8-3. Histograma de frecuencias de tiempo.

Realizado por: Alexis, M.; Sofia, S.2020.

Al realizar el estudio se consideró una muestra de 40 datos la cual se estima suficiente para representar el comportamiento del sistema. Se requiere conocer la tendencia central y la dispersión de los datos obtenidos por lo que se utiliza estadística descriptiva, como se muestra en la tabla 7-3.

Tabla 7-3. Estadística descriptiva de los datos de tiempo.

Análisis de datos	
Tamaño de la muestra	40
Retardo Mínimo [ms]	2,72
Retardo Máximo [ms]	6,9
Amplitud [ms]	4.18
Media [ms]	3,64
Moda [ms]	2,83
Desviación Estándar [ms]	1,1434

Realizado por: Alexis, M.; Sofia, S.2020.

Después del análisis se obtuvo una media de 3.64 [ms] que muestra la tendencia central a la que convergen los datos de la muestra que va desde el tiempo de retardo mínimo, 2.72 [ms], hasta el máximo, 6.9 [ms]. Se observa también la dispersión de estos datos con respecto a dicha media, que, en el caso planteado, es de 1.1434 [ms] con estos datos se concluye que el retardo existente entre la interfaz gráfica Ursim y la réplica de movimientos del *Digital Twin* es imperceptible visualmente. Cabe recalcar que se trabaja únicamente en entornos virtuales por lo que el tiempo de latencia es solo el que tarda en enviar y recibir datos Experior.

3.2.3.2 Resultado grafico de la adaptación al entorno

Al comprobar la correcta comunicación y un fidedigno envió como recepción de registros. Dentro de Experior debe reflejarse una correcta adaptación al medio, la figura 9-3 muestra el resultado grafico que genera Experior para el último ejemplo planteado el cual ubica la pieza en la coordenada (200,200) [mm] desde el sistema de referencia de la base del robot y rotada un ángulo de 45° donde visualmente se observa que la configuración es correcta.

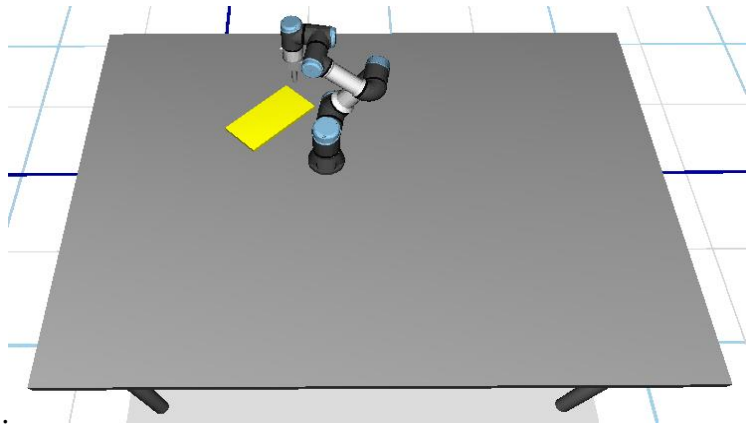


Figura 9-3. Respuesta grafica de Experior ubicando el centro en (200, 200) [mm] y rotado 45°.

Realizado por: Alexis, M.; Sofia, S.2020

3.2.4 Efectividad

En principio el diseño de la interfaz Digital Twin se esperaba realizarlo con el brazo robótico UR3 real, bajo esas consideraciones las datas recabadas de manera real debían presentar niveles de imprecisión siendo así susceptibles a ser tratadas con un criterio de efectividad. Dada la emergencia sanitaria todo es modelado por computadora por lo que la incertidumbre numérica queda inexistente. Entonces, un estudio enfocado en eficacia, eficiencia y efectividad ya no aplica o no tendría sentido, ya que al hablar de una eficacia eficiente se hace referencia a contrastar una realidad de naturaleza estocástica que en este análisis de resultados ya no está presente.

Por lo tanto, se argumenta ya que todo ha sido hecho por computador utilizando software no es posible cumplir con este criterio o a su vez determinar que la efectividad existente entre el entorno de simulación Ursim y el Digital Twin es del cien por ciento.

3.2.5 Fidelidad del Digital Twin en el proceso de marcado para corte de superficies

El Digital Twin del brazo robótico realiza su movimiento como un proceso independiente es decir desvinculado de los datos del ambiente de trabajo por lo que su fidelidad se verá reflejada en la similitud con la que el brazo robótico realiza los cortes configurados por el algoritmo de corte sobre la plancha.

La fidelidad va a ser probada con un caso de estudio donde se ubican tres piezas que cubran el área total de la plancha, la tabla 8-3 muestra los recursos disponibles y la orden de producción utilizadas, los datos del ambiente son los siguientes:

$$object\_location = [150, 300, 4, -0.523, -0.523, 0.000]$$

Tabla 8-3. Datos de ingreso para prueba de fidelidad.

Datos ingresados		
Ítem	Dimensiones (mm)	
	Width	Length
Plancha	300	150
Pieza 1	75	150
Pieza 2	75	150
Pieza 3	150	150

Realizado por: Alexis, M.; Sofia, S.2020.

Para la respuesta gráfica en Experior se configura el entorno de tal manera que las piezas marcadas tomen un color diferente al de la plancha así se comprueba gráficamente que el robot se mueva de forma adecuada con las configuraciones del ambiente.

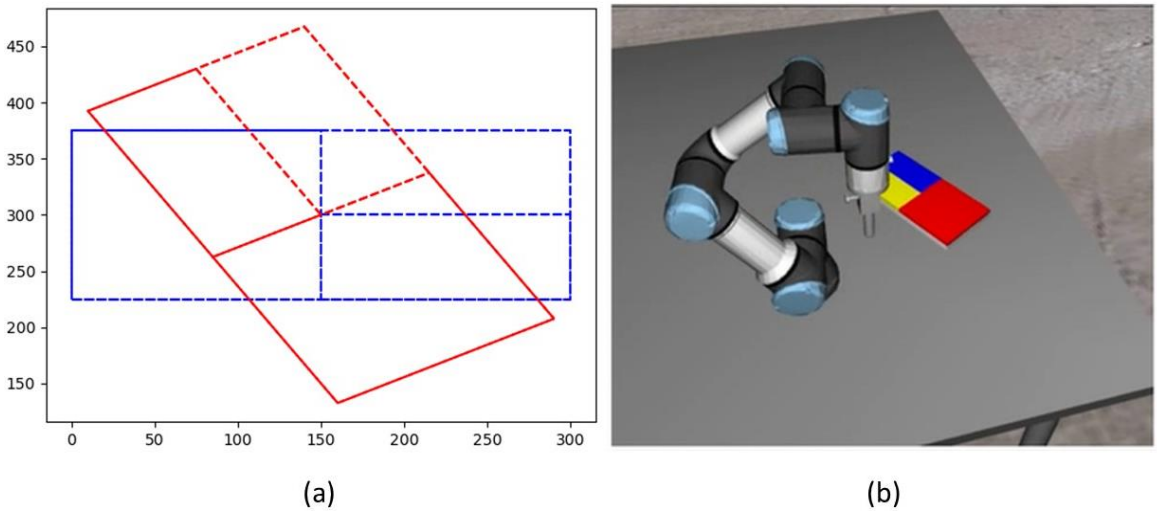


Figura 10-3. (a) Respuesta gráfica del algoritmo de corte con giro de -30°. (b) Resultado gráfico de Experior después del corte.

Realizado por: Alexis, M.; Sofia, S.2020

La tabla 9-3 muestra los resultados numéricos de las coordenadas de cada pieza obtenidos por el algoritmo de corte generados en Python y los datos reales de la ubicación del robot en las coordenadas que realiza el corte que están limitados por la exactitud y precisión mecánicas del robot. Los datos reales de la tabla presentan un error respecto a los calculados de $RMSE = 0{,}242\%$ el cual se halla empleando la ecuación 1-3 para el error cuadrático medio, cabe recalcar que estos datos son recaudados de la plataforma de simulación Ursim, aunque reflejan la misma configuración que se realiza en el robot real.

Ecuación 1-3. Error cuadrático medio

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(C_i - R_i)^2}{n}}$$

$C_i = datos\ calculados$

$R_i = datos\ reales$

Tabla 9-3. Datos calculados y datos reales de ubicación.

Resultados				
ITEM	Calculados		Reales	
	X	Y	X	Y
Plancha	289,951905283832	207,596189432334	289,95	207,60
	139,951905283832	467,403810567665	139,95	467,40
	10,048094716167	392,403810567665	10,05	392,40
	160,048094716167	132,596189432334	160,05	132,60
	214,951905283832	337,500000000000	214,95	337,50
	139,951905283832	467,403810567665	139,95	467,40

Pieza 1	75,000000000030	429,903810567665	75,00	429,90
	150,000000000000	300,000000000000	150,00	300,00
Pieza 2	150,000000000000	300,000000000000	150,00	300,00
	75,000000000030	429,903810567665	75,00	429,90
	10,048094716167	392,403810567665	10,05	392,40
	85,048094716167	262,500000000000	85,05	262,50
Pieza 3	289,951905283832	207,596189432334	289,95	207,60
	214,951905283832	337,500000000000	214,95	337,50
	85,048094716167	262,500000000000	85,05	262,50
	160,048094716167	132,596189432334	160,05	132,60

Realizado por: Alexis, M.; Sofia, S.2020.

CONCLUSIONES

- La interfaz *Digital Twin* apoya la actividad de supervisión del operador de acuerdo con los requisitos de los sistemas de fabricación flexible en el proceso de corte de superficies con una arquitectura HCMI que permite la interacción entre máquina-computador-humano.
- El uso de la *Wrist Camera* y el software *Camera Locate* como una adaptación para el brazo robótico UR3 permitió una percepción del ambiente de trabajo adecuada obteniendo la variable *object_location* que resulto de gran ayuda para la programación dentro de PolyScope y Python.
- La elección del algoritmo Guillotina tubo resultados favorables frente a varios casos de estudio ubicando las piezas con mejor disposición que los algoritmos Shelf y Skyline, y presenta resultados similares con el algoritmo Maximal Rectangles. Además, el algoritmo elegido posee una programación menos compleja por lo que su costo computacional es reducido.
- La implementación del *Digital Twin* ayuda al operador a verificar si la lógica de control del robot cumple con los requisitos de producción y tiene una adecuada adaptación al ambiente de trabajo.
- Los resultados obtenidos confirmaron como la interfaz *Digital Twin* permite una supervisión casi en tiempo real con una media de retardo de 3.64 [ms] y una fidelidad al momento de seguir los segmentos de corte dados por el algoritmo de 99,758%

RECOMENDACIONES

- Se recomienda verificar la efectividad del proceso en un futuro cuando se tenga acceso de manera física al robot al concluir de la pandemia.
- Cuando un error ocurre en la autoadaptación los datos sobre el modelo de simulación *Digital Twin*, la lógica de control, la orden de producción, los recursos disponibles, la adaptación al entorno y la información de la falla deben ser enviados hacia el operador para ser resulto. El envío de esta información deberá ser estudiado como un trabajo futuro.
- Por ahora la interfaz *Digital Twin* ha sido aplicada a escala de laboratorio universitario por lo que se recomienda realizar pruebas con operadores reales para evaluar su posible escalabilidad industrial.
- Como trabajo posterior se recomienda desarrollar un modelo de simulación del robot propio con el cual se pueda contrastar los datos adquiridos del modelo de simulación distribuido por el mismo fabricante.

GLOSARIO

Compilar: Convertir un programa en lenguaje máquina a partir de otro programa de computadora escrito en otro lenguaje (Diccionario de la lengua española 2020a).

Digitalización: Acción y efecto de digitalizar (Diccionario de la lengua española 2020b).

Estocástico: Teoría de estadística de los procesos cuya evolución en el tiempo es aleatoria, tal como la secuencia de las tiradas de un lado (Diccionario de la lengua española 2020c).

Fabricas inteligentes:

Gripper: Adaptación para brazos robóticos colaborativos (Universal Robots 2015).

Heurística: En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas (Diccionario de la lengua española 2020d).

Interfaz: Conexión, física o lógica, entre una computadora y el usuario, un dispositivo periférico o un enlace de comunicaciones (Diccionario de la lengua española 2020e).

Latencia: Tiempo que transcurre entre un estímulo y la respuesta que produce, y, en particular, lapso entre el momento en que se contrae una enfermedad y la aparición de los primeros síntomas (Diccionario de la lengua española 2020f).

Modelado: Acción y efecto de modelar (Diccionario de la lengua española 2020g).

Productividad: Capacidad o grado de producción por unidad de trabajo, superficie de tierra cultivada, equipo industrial (Diccionario de la lengua española 2020h).

Simulación: Acción y efecto de simular (Diccionario de la lengua española 2020i).

Sintaxis: Conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje de programación (Diccionario de la lengua española 2020j).

Tupla: concatenación de dos elementos en un solo registro dentro de Python (Chun 2001).

BIBLIOGRAFÍA

ABDALÁ, C.; & CABERTA, Ñ. "Caracterización de un robot". *MECAMEX*, vol. 12, no. 1 (2016), (Mexico) pp. 1-6.

ÁGORA, N. *La industria automovilística con robots de corte por láser - Automatización en la industria 4.0*. [en línea]. [Consulta: 3 junio 2020]. Disponible en: https://www.interempresas.net/Robotica/Articulos/216809-Ganando-en-la-industria-automovilistica-con-robots-de-corte-por-laser.html.

BAKER, B.S.; & SCHWARZ, J.S. "Shelf Algorithms for Two-Dimensional Packing Problems". SIAM Journal on Computing [en línea], 1983, (Nueva Zelanda) 12(3), pp. 508-525. [Consulta: 31 agosto 2020]. ISSN 0097-5397. Disponible en: http://epubs.siam.org/doi/10.1137/0212033.

BECERRA, A. *Introduccíon a la programacíon con Python* [en línea]. [Consulta: 28 agosto 2020]. Disponible en: http://myriadicity.net.

BOTHWELL, S. *GitHub - greedypacker: Algoritmos de empaquetado de contenedores 2D.* [en línea]. [Consulta: 18 agosto 2020]. Disponible en: https://github.com/ssbothwell/greedypacker#tests.

BREUEL, T. "Two Geometric Algorithms for Layout Analysis". Xerox Palo Alto Research Center [en linea], 2002, (California) 1(1), pp. 2-6. [Consulta: 31 de agosto 2020]. DOI: 10.1007/3-540-45869-7_23. Disponible en: https://www.researchgate.net/publication/2504221_Two_Geometric_Algorithms_for_Layout_Analysis

CARMEN, R.; et al. "Creación de ambientes virtuales inmersivos con software libre". Revista Digital Universitaria [en línea], 2007, (Mexico) 8(6), pp. 3-7. [Consulta: 4 de junio 2020]. ISSN

1067-6079. Disponible en: http://ru.tic.unam.mx:8080/tic/handle/123456789/1280

CHUN, W. *Core Python Programming. PH-PTR* [en línea]. Taiwan: PHPTR, 2001. [Consulta: 18 agosto 2020]. Disponible en: https://books.google.com.ec/books?id=mh0bU6NXrBgC&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false.

DANIEL, J.; et al. "Human-Computer-Machine Interaction for the Supervision of Flexible Manufacturing Systems : A Case Study". IFAC-PapersOnLineC [en línea], 2028, (Colombia) 7(1), pp. 1-6. [Consulta: 18 agosto 2020]. Disponible en: https://www.researchgate.net/publication/340952452_Human-Computer-Machine_Interaction_for_the_Supervision_of_Flexible_Manufacturing_Systems_A_Case_Study

DICCIONARIO DE LA LENGUA ESPAÑOLA. *Compilar*. [en línea]. 23(3). Madrid-España: Real Academia Española, 2020. [Consulta: 10 noviembre 2020]. Disponible en: https://dle.rae.es/compilar?m=form.

DICCIONARIO DE LA LENGUA ESPAÑOLA. *Digitalización*. [en línea]. 23(3). Madrid-España: Real Academia Española, 2020. [Consulta: 10 noviembre 2020]. Disponible en: https://dle.rae.es/digitalización?m=form.

DICCIONARIO DE LA LENGUA ESPAÑOLA. *Estocástico, estocástica*. [en línea]. 23(3). Madrid-España: Real Academia Española, 2020. [Consulta: 10 noviembre 2020]. Disponible en: https://dle.rae.es/estocástico?m=form.

DICCIONARIO DE LA LENGUA ESPAÑOLA. *Heurístico, heurística.* [en línea]. 23(3). Madrid-España: Real Academia Española, 2020. [Consulta: 10 noviembre 2020]. Disponible en: https://dle.rae.es/heurístico#KHdGTfC.

DICCIONARIO DE LA LENGUA ESPAÑOLA. *Interfaz.* [en línea]. 23(3). Madrid-España: Real Academia Española, 2020. [Consulta: 10 noviembre 2020]. Disponible en:

https://dle.rae.es/interfaz?m=form.

DICCIONARIO DE LA LENGUA ESPAÑOLA. *Latencia.* [en línea]. 23(3). Madrid-España: Real Academia Española, 2020. [Consulta: 10 noviembre 2020]. Disponible en: https://dle.rae.es/latencia?m=form.

DICCIONARIO DE LA LENGUA ESPAÑOLA. *Modelado*. [en línea]. 23(3). Madrid-España: Real Academia Española, 2020. [Consulta: 10 noviembre 2020]. Disponible en: https://dle.rae.es/modelado?m=form.

DICCIONARIO DE LA LENGUA ESPAÑOLA. *Productividad.* [en línea]. 23(3). Madrid-España: Real Academia Española, 2020. [Consulta: 10 noviembre 2020]. Disponible en: https://dle.rae.es/productividad?m=form.

DICCIONARIO DE LA LENGUA ESPAÑOLA. *Simulación.* [en línea]. 23(3). Madrid-España: Real Academia Española, 2020. [Consulta: 10 noviembre 2020]. Disponible en: https://dle.rae.es/simulación?m=form.

DICCIONARIO DE LA LENGUA ESPAÑOLA. *Sintaxis.* [en línea]. 23(3). Madrid-España: Real Academia Española, 2020. [Consulta: 10 noviembre 2020]. Disponible en: https://dle.rae.es/sintaxis?m=form.

DYCKHOFF, H.; et al. *Cutting and Packing. Physica-Verlag Heidelberg* [en línea]. Alemania: Physica-Verlag Heidelberg, 1997. [Consulta: 31 agosto 2020]. Disponible en: https://www.springer.com/gp/book/9783642634871.

ENRÍQUEZ, D.; et al. "Didactic use of immersive virtual reality with NUI focused on the inspection of wind turbines". Opening [en línea], 2017, Guadalajara 9(2), pp. 8-23. [Consulta: 4 junio 2020] . ISSN: 16656180. Disponible en: https://www.academia.edu/34717004/Didactic_use_of_immersive_virtual_reality_with_NUI_focused_on_the_inspection_of_wind_turbines.

FERRIZ, J. Control de fuerza con el brazo robot colaborativo UR3 [en línea] (trabajo de titulación) (Pre-grado). Poltecnica de Valencia, Ingeniería Electrónica Industrial y Automática. Valencia-España. 2018. pp. 30-45. [Consulta: 31 agosto 2020]. Disponible en: https://m.riunet.upv.es/bitstream/handle/10251/112457/Ferriz%20-%20Control%20de%20fuerza%20con%20el%20brazo%20robot%20colaborativo%20UR3.pdf?sequence=3&isAllowed=y

GRIEVES, M.; & VICKERS, J. "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems". *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches*, 1(1) (2016), (United State of America) pp. 105-122.

KOULAMAS, C.; & KALOGERAS, A. "Cyber-Physical Systems and Digital Twins in the Industrial Internet of Things". IEEE Xplore [en línea], 2018, (Venezuela) 51(11), pp. 95-98. [Consulta: 4 junio 2020]. DOI: 10.1109/MC.2018.2876181. Disponible en: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8625931.

LEE, J; et al. "ScienceDirect Service innovation and smart analytics for Industry 4.0 and big data environment". Procedia CIRP [en línea], 2014, (Cincinnati) 16(1), pp. 3-8. [Consulta: 4 junio 2020]. DOI: 10.1016/j.procir.2014.02.001. Disponible en: www.sciencedirect.com.

LEE, J.; et al. "Recent advances and trends in predictive manufacturing systems in big data environment". Manufacturing Letters [en línea], 2013, (Cincinnati) 1(1), pp. 38-41. ISSN 22138463. DOI 10.1016/j.mfglet.2013.09.005. [Consulta: 4 junio 2020] Disponible en: http://dx.doi.org/10.1016/j.mfglet.2013.09.005.

MARIÑO, A.; & SANCHEZ, S. *Percepción del ambiente por la Wrist Camera del brazo robótico UR3.* [en línea]. Riobamba-Ecuador: 2020.[Consulta: 12 septiembre 2020]. Disponible en: https://www.youtube.com/watch?v=339BbZryJYM&feature=youtu.be.

MATTEO, A.J.; & COTO, E. *Una herramienta para generar Mundos Virtuales Inmersivos*. *View project SOS Telemedicina View project.* [en línea]. Caracas: 2000 .[Consulta: 4 junio 2020]. Disponible en: https://www.researchgate.net/publication/228858236.

SCHLUSE, M.; & ROSSMANN, J. "From Simulation to Experimentable Digital Twins Simulation-based Development and Operation of Complex Technical Systems". IEEE International Symposium on Systems Engineering (ISSE) [en línea], 2016, (Edinburgh) 5(1), pp. 1-6. [Consulta: 10 junio 2020]. DOI: 10.1109/SysEng.2016.7753162.

OLLERO, A. *Robótica: Manipuladores y Robots Móviles* [en línea]. S.l. United State of America: Marcombo, 2001. [Consulta: 3 junio 2020]. Disponible en: https://books.google.es/books?hl=es&lr=&id=TtMfuy6FNCcC&oi=fnd&pg=PR13&dq=robots+manipuladores&ots=32PUD3v93M&sig=z9FpPd4YZ1EKhJqWgg6gfsR0WrM#v=onepage&q=robots manipuladores&f=false.

PARRIS, C. *Minds Machines: Meet A Digital Twin. Minds Machines: Meet A Digital Twin* [en línea]. [Consulta: 1 junio 2020]. Disponible en: https://www.youtube.com/watch?v=2dCz3oL2rTw.

PARROTT, A.; & WARSHAW, L. "Industry 4.0 and the digital twin". Deloitte University Press [en línea], 2017, (United State of America) 5(2), pp. 1-17. Disponible en: https://dupress.deloitte.com/dup-us-en/focus/industry-4-0/digital-twin-technology-smart-factory.html.

PÉREZ, L.; et al. "Digital twin and virtual reality based methodology for multi-robot manufacturing cell commissioning". Applied Sciences [en línea], 2020, (España) 10(10), pp 10-15. [Consulta: 19 junio 2020]. ISSN: 20763417. DOI: 10.3390/app10103633. Disponible en: https://www.researchgate.net/publication/341619699_Digital_Twin_and_Virtual_Reality_Based_Methodology_for_Multi-Robot_Manufacturing_Cell_Commissioning.

RÍOS, J.; et al. "Product avatar as digital counterpart of a physical individual product: Literature review and implications in an aircraft". Advances in Transdisciplinary Engineering [en línea], 2015, (España) 10(2), 2015, pp. 657-666. [Consulta: 12 junio 2020] DOI 10.3233/978-1-61499-544-9-657. Disponible en: https://www.mdpi.com/2076-3417/10/10/3633.

ROSEN, R.; et al. "About the importance of autonomy and digital twins for the future of

manufacturing". IFAC-PapersOnLine [en línea], 2015, (Alemania) 28(3), pp. 567-572. [Consulta: 29 mayo 2020]. ISSN 24058963. DOI 10.1016/j.ifacol.2015.06.141. Disponible en: https://linkinghub.elsevier.com/retrieve/pii/S2405896315003808.

SALVADOR, S. "Estado del arte y análisis de las técnicas Big Data en Gemelos Digitales". *ITI-Instituto Tecnologico de Información*, vol. 5, no. 1 (2017), pp. 6-15.

TUEGEL, E. "The airframe digital twin: Some challenges to realization". Collection of Technical Papers [en línea], 2012, (United State of America) 3(1), pp. 1-8. [Consulta: 29 mayo 2020]. ISSN 02734508. DOI 10.2514/6.2012-1812. Disponible en: https://www.researchgate.net/publication/268478541_The_Airframe_Digital_Twin_Some_Challenges_to_Realization

UNIVERSAL ROBOTS. *Manual de usuario UR3/CB3* [en línea]. 3(1). United State of America: Universal robots, 2015. [Consulta: 3 junio 2020]. Disponible en: https://www.cfzcobots.com/wp-content/uploads/2017/03/ur3_user_manual_es_global.pdf

UNIVERSAL ROBOTS. *Wrist Camera Vision System for e-Series Universal Robots Instruction Manual* [en línea]. 3(1). United State of America: Universal robots, 2018. [Consulta: 3 junio 2020]. Disponible en: https://assets.robotiq.com/website-assets/support_documents/document/Vision_System_e-Series_PDF_20190116.pdf

VAL, J. "Industria 4.0. La Transformación Digital de la Industria Española". Coddiinforme [en línea], 2012, (España) 3(1), pp. 120. [Consulta: 13 mayo 2020] Disponible en: http://coddii.org/wp-content/uploads/2016/10/Informe-CODDII-Industria-4.0.pdf.

WALLACE J. HOPP, M.L.S. *Factory Physics* [en línea]. Tercera edición. Waveland - United State of America: Waveland Press, 2011. [Consulta: 29 mayo 2020]. Disponible en: https://books.google.com.ec/books?hl=es&lr=&id=TfcWAAAAQBAJ&oi=fnd&pg=PP2&dq=.+Hopp,+W.+J.,+%26+Spearman,+M.+L.+(2011).+Factory+physics.+Waveland+Press.&ots=w__EP8h1cP&sig=2vqw9_D10Gh8vpBQUKDb2GDF6QU#v=onepage&q=. Hopp%2C W. J.%2C %26 Spearman%2C M. L. (2.

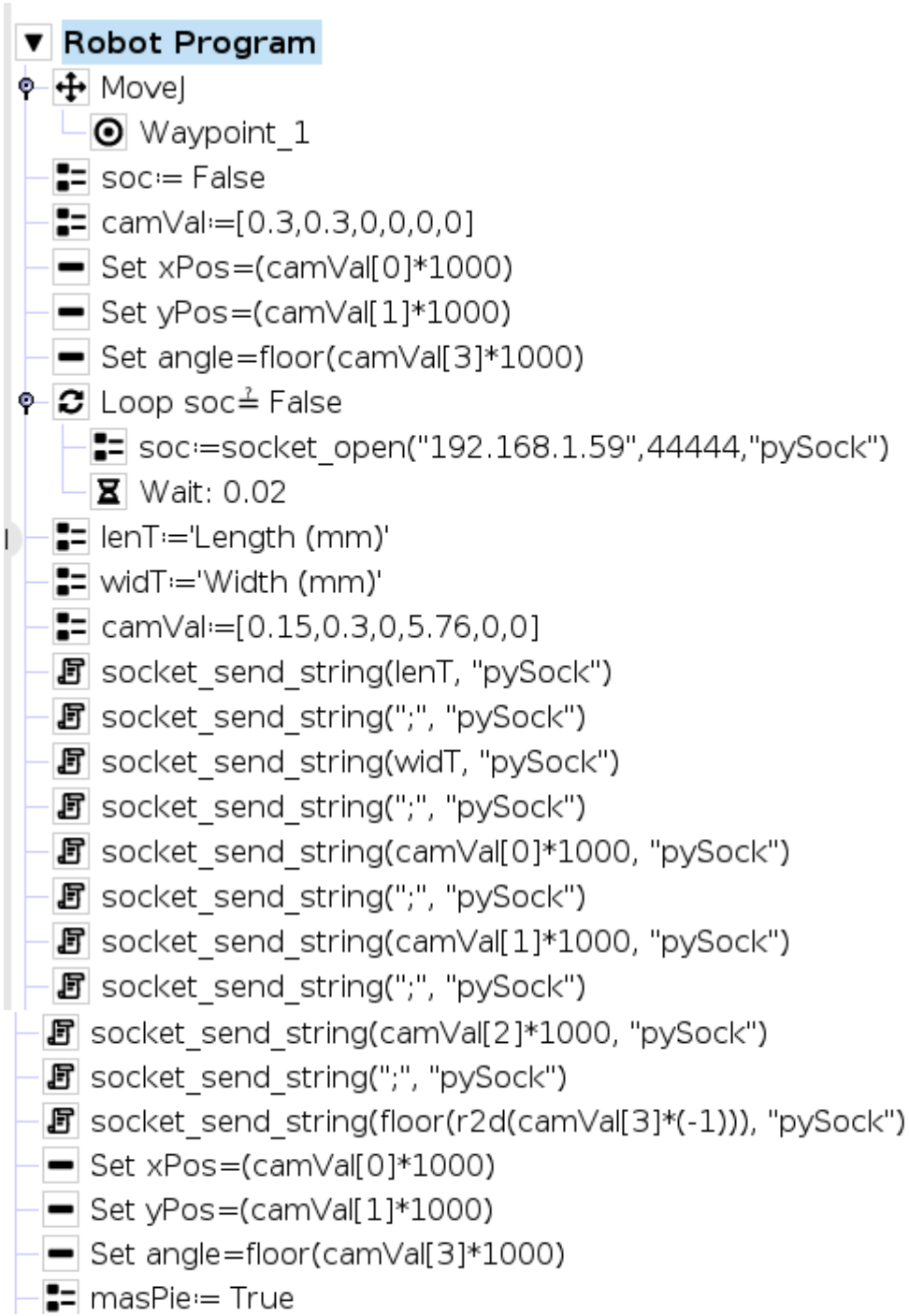
XCELGO. *Xcelgo | Software de modelado digital doble y puesta en marcha virtual* [en línea]. [Consulta: 17 junio 2020]. Disponible en: https://xcelgo.com/about-us/.

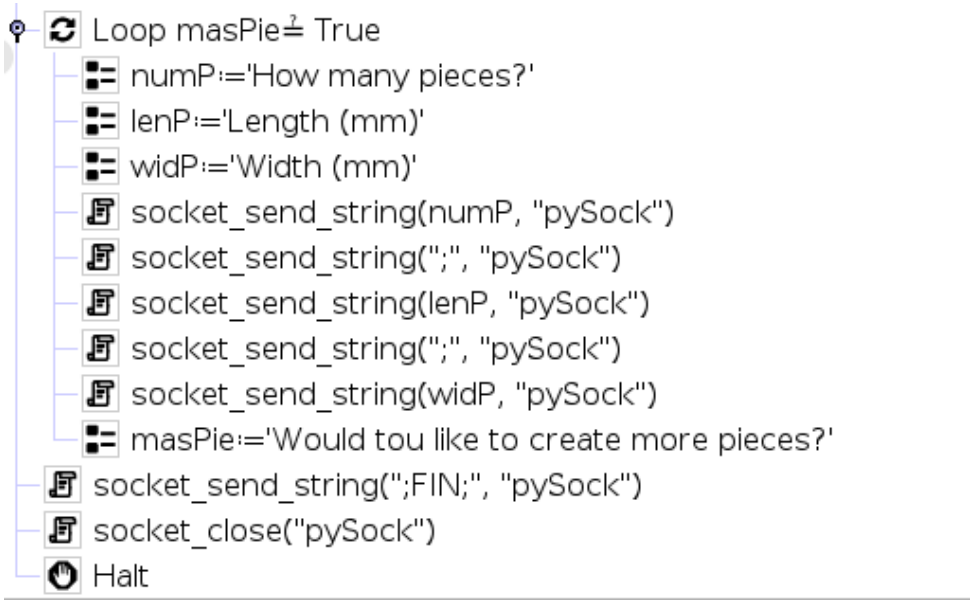
ZHANG, L. *URSIM Reference Manual* [en línea]. Tercera edición. United State of America: Universal robots, 1999. [Consulta: 3 junio 2020]. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.467.6430&rep=rep1&type=pdf

ZHONG, R.; et al. "Intelligent Manufacturing in the Context of Industry 4.0". Engineering [en linea], 2017, (United State of America) 3(5), pp. 616-630. [Consulta: 4 junio 2020]. ISSN: 20958099. DOI: 10.1016/J.ENG.20 17.05.015. Disponible en: https://researchportal.bath.ac.uk/en/publications/intelligent-manufacturing-in-the-context-of-industry-40-a-review

ANEXOS

Anexo A: código programado en PolyScope





Anexo B: Código programado en Python.

```
import socket;
import time;
import greedypacker;

FIN = ";FIN;";  # Comando de fin de comunicación
SIG = ";SIG;";  # Comando de siguiente punto

datT = [];     # Lista con los datos del tablero [Largo, Ancho, Ángulo]
datP = [];     # Lista con los datos de las piezas [numPiezas, LargoPieza, AnchoPieza]
puntosX = [];  # Lista con los puntos en X de las piezas
puntosY = [];  # Lista con los  puntos en Y de las piezas
pInX = 0.2;
pInY = 0.2;

robotIP = "192.168.6.128";    # Dirección IP del Robot UR
#robotIP = socket.gethostbyname(socket.gethostname());  # Dirección IP del Robot UR
puertoEN = 30002;             # Puerto abierto del Robot para Script programing
puertoRE = 44444;             # Puerto arbritario para recibir información

conEnvi = (robotIP, puertoEN);  # Tupla que representa la conexión de envío
conRece = (robotIP, puertoRE);  # Tupla que representa la conexión de recepción

sRE = socket.socket(socket.AF_INET, socket.SOCK_STREAM);    # Crear socket para recepción
sRE.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1);
sRE.bind(conRece);            # Une el socket a la dirección IP
print("[Conectando] ....");
sRE.listen(5);                # Deja a la espera de conexión
c, add = sRE.accept();        # Conecta con el Robot a través del Socket creado de recepción
print("[Conectado] Conexión establecida con URobot en {} ".format(robotIP));

## ----------------------------------------
#   Recepción de información del Robot
## ----------------------------------------

print("[Esperando recepción de datos]");
count = 0;
```

```
while count < 150:
   try:
      msg = c.recv(1024).decode('utf-8');
      print("[Recivido] De {}: {}".format(robotIP, msg));
      if msg == FIN:
         break;
      elif count == 0:
         datT = list(map(int, msg.split(";")));
      else:
         datP.append(list(map(int, msg.split(";"))));
      time.sleep(0.5);
   except Exception as e:
      print(msg);
   count += 1;

print("[FIN] Fin de la recepción");
c.close();
sRE.close();

## ---------------------------------------
#   Algoritmo de optimización
## ---------------------------------------

Tablero = greedypacker.BinManager(datT[0], datT[1], pack_algo='guillotine',
heuristic='best_longside');
num = 0;
for pAct in datP:
   num += pAct[0];
   for numP in range(pAct[0]):
      Tablero.add_items(greedypacker.Item(pAct[1], pAct[2]));
Tablero.execute();
pX = [(int(Tablero.bins[0].__repr__().split("x=")[i].split(",")[0])/1000)+pInX for i in range(1,
num+1)];
pY = [(int(Tablero.bins[0].__repr__().split("y=")[i].split(")")[0])/1000)+pInY for i in range(1,
num+1)];
wS = [int(Tablero.bins[0].__repr__().split("width=")[i].split(",")[0])/1000 for i in range(1, num+1)];
hS = [int(Tablero.bins[0].__repr__().split("height=")[i].split(",")[0])/1000 for i in range(1, num+1)];

rect = [[(pX[n], pY[n]), (pX[n]+wS[n], pY[n]), (pX[n]+wS[n], pY[n]+hS[n]),
      (pX[n], pY[n]+hS[n]), (pX[n], pY[n])] for n in range(num)];

## ---------------------------------------
#   Rotar y trasladar puntos
## ---------------------------------------




## ---------------------------------------
#   Crear Socket para envío de puntos
## ---------------------------------------

sEN = socket.socket(socket.AF_INET, socket.SOCK_STREAM);    # Crear socket para envío
sEN.connect(conEnvi);          # Conecta con el Robot a través del Socket creado de envío
print("[Conectado] Conexión establecida con robot para envío");

## ---------------------------------------
#   Enviar datos de movimiento
## ---------------------------------------
```

```
print("[Enviando puntos] .... ");
timeS = 0.5;
num = 0;
for rectangulo in rect:
   cmd = "movel(p[{}, {}, 0.115, 0, 0, 0], a=1.4, v=1.1,
t={})".format(rectangulo[0][0],rectangulo[0][1],timeS) + "\n";
   sEN.send(cmd.encode());
   time.sleep(timeS*2);
   for p in rectangulo:
      cmd = "movel(p[{}, {}, 0.104, 0, 0, 0], a=1.4, v=1.1, t={})".format(p[0],p[1], timeS) + "\n";
      sEN.send(cmd.encode());
      time.sleep(timeS*2);
   num += 1;
   print("[Completado] Pieza {} formada ...".format(num));

print("[Completado] Todos las piezas se han formado");
cmd = "movel(p[{}, {}, 0.154, 0, 0, 0], a=1.4, v=1.1, t={})".format(0.1,0.3,timeS) + "\n";
sEN.send(cmd.encode());

## ---------------------------------------
#   Cerrar sockets de comunicación
## ---------------------------------------

sEN.close();
print("[{}]".format(FIN));
```

Anexo C: código implementado para los resultados gráficos en Python.

```
import greedypacker
import math
import numpy as np
import matplotlib.pyplot as plt
pCx = 200;
pCy = 200;
wid = 150;
hei = 300;
pInX = pCx - wid / 2;
pInY = pCy - hei / 2;
if ( wid  >  hei) :th = math.radians((-124.84+90)+90);
else: th = math.radians((-124.84+90));
dx = [-wid/2,-wid/2,wid/2,wid/2,-wid/2]
dy = [-hei/2,hei/2,hei/2,-hei/2,-hei/2]

#rectangulo sin rotar
datosx = [[dx[n]+pCx] for n in range(5)]
print('Puntos en x de rectangulo sin rotar')
print(datosx)
datosy = [[dy[n]+pCy] for n in range(5)]
print('Puntos en y de rectangulo sin rotar')
print(datosy)
plt.plot(datosx,datosy,"b--")


#rectangulo rotado
girox=[[(dx[n]*np.cos(th)-dy[n]*np.sin(th))+pCx] for n in range(5)];
print('Puntos en x de rectangulo rotado')
print(girox)
giroy=[[(dx[n]*np.sin(th)+dy[n]*np.cos(th))+pCy] for n in range(5)];
```

```
print('Puntos en y de rectangulo rotado')
print(giroy)
plt.plot(girox,giroy,"b--")

#Algoritmo de corte

num = 2;
M = greedypacker.BinManager(wid, hei, pack_algo='guillotine', heuristic='best_longside')
#M.add_items(greedypacker.Item(50, 100))
M.add_items(greedypacker.Item(50, 50))
M.add_items(greedypacker.Item(20, 20))
# M.add_items(greedypacker.Item(60, 40))
M.execute()
pX = [(int(M.bins[0].__repr__().split("x=")[i].split(",")[0]))+pInX  for i in range(1, num + 1)]
pY = [(int(M.bins[0].__repr__().split("y=")[i].split(")")[0]))+pInY   for i in range(1, num + 1)]
wS = [int(M.bins[0].__repr__().split("width=")[i].split(",")[0])  for i in range(1, num + 1)]
hS = [int(M.bins[0].__repr__().split("height=")[i].split(",")[0])  for i in range(1, num + 1)]

rect = [[[pX[n], pY[n]], [pX[n]+wS[n], pY[n]], [pX[n]+wS[n], pY[n]+hS[n]],
         [pX[n], pY[n]+hS[n]], [pX[n], pY[n]]] for n in range(num)];


print('Puntos dados por el algoritmo sin rotacion')
print(rect)
#Rotacion

for i in range(num):alx=[rect[i][0][0],rect[i][1][0],rect[i][2][0],rect[i][3][0],rect[i][4][0]];
aly=[rect[i][0][1],rect[i][1][1],rect[i][2][1],rect[i][3][1],rect[i][4][1]];plt.plot(alx,aly,"y--")
def rotarTrasP(punto, pCx, pCy):
    puntoR = [];
    punto = [punto[0] - pCx, punto[1] - pCy];
    #print(punto[0], punto[1])
    puntoR.append((punto[0] * math.cos(th) - punto[1] * math.sin(th)) + pCx);
    puntoR.append((punto[0] * math.sin(th) + punto[1] * math.cos(th)) + pCy);
    return puntoR;
plt.plot(pCx,pCy,"r")
dr=[[rotarTrasP(rect[i][0],pCx,pCy),rotarTrasP(rect[i][1], pCx, pCy),rotarTrasP(rect[i][2], pCx,
pCy),rotarTrasP(rect[i][3], pCx, pCy),rotarTrasP(rect[i][4], pCx, pCy)]for i in range(num)]
print('Puntos dados por el algoritmo rotados')
print(dr)
for i in
range(num):drtx=[dr[i][0][0],dr[i][1][0],dr[i][2][0],dr[i][3][0],dr[i][4][0]];drty=[dr[i][0][1],dr[i][1]
[1],dr[i][2][1],dr[i][3][1],dr[i][4][1]];plt.plot(drtx,drty,"r--")

plt.show()
```