UNIVERSITAT POLITÈCNICA DE VALÈNCIA	*Memoria del trabajo fin de máster: Ensamblaje de cámara de goteo utilizando una celda robotizada*	Escuela Técnica Superior de Ingeniería del Diseño

Memoria del trabajo fin de máster: Ensamblaje de cámara de goteo utilizando una celda robotizada

Realizado por:

Jatnna Amparo Torres Díaz

Dirigido por:

Carlos Ricolfe Viala

Septiembre 2019, Valencia

Máster en Ingeniería Mecatrónica

Índice

1 RESUMEN

Trabajo final del máster en Ingeniería Mecatrónica, realizado por la estudiante Jatnna Amparo Torres Díaz y dirigido por el Profesor Carlos Ricolfe Viala. Este trabajo consiste en la utilización de una celda robotizada para automatizar el ensamblaje manual del componente "cámara de goteo" (del inglés "drip chamber"). Las cámaras de goteo son ampliamente utilizadas en medicina durante el proceso de infusión intravenosa en pacientes. El ensamblaje de la cámara de goteo suele ser manual y cuenta con diversas desventajas las cuales se pretenden superar por medio de la automatización propuesta en este trabajo. Esta automatización está conformada principalmente por dos brazos robóticos, un sistema de visión y una cinta transportadora. Todos los equipos utilizados se encuentran disponibles en el laboratorio de Robótica de la Universidad Politécnica de Valencia.

2 INTRODUCCIÓN

Para el desarrollo de este proyecto, se parte de la idea de que una empresa dedicada a la industria médica está solicitando la automatización de uno de sus procesos de ensamblaje, específicamente el ensamblaje de su producto "cámara de goteo". Para ello, la empresa pone a disposición una celda robotizada que fue creada utilizando equipos de una máquina desmantelada. La máquina en cuestión fue desmantelada porque el producto que manufacturaba ha sido transferido a una planta en un país vecino. Si los equipos no se utilizan, la empresa podría verse en la obligación de enviarlos a un almacén externo para evitar gastos de ocupación de espacio en planta.

2.1 Objetivo y requerimientos del proyecto

El objetivo fundamental de este proyecto es, automatizar el proceso de ensamblaje de la cámara de goteo utilizando una celda robotizada. Además del objetivo, el cliente tiene los siguientes requerimientos específicos:

- Todos los equipos facilitados por la empresa deben ser utilizados en la implementación de la solución.
- El sistema de visión debe ser el controlador central o servidor del sistema dentro de la celda robotizada.
- Se debe ofrecer una explicación detallada del código fuente desarrollado para la implementación de la solución.

3 ESTADO DE LA CUESTIÓN Y FUNDAMENTACIÓN TEÓRICA

En esta sección se explica la información disponible respecto al proceso de ensamblaje actual con sus respectivos inconvenientes, soluciones similares a la sugerida en este proyecto o soluciones disponibles en el mercado, así como la fundamentación teórica (términos y/o conceptos) requeridos para la compresión del proyecto. Los conceptos y términos requeridos son los siguientes:

Concepto/término	Descripción
MSDS	Término de sus siglas en inglés: *Material Safety Data Sheet*. Es una hoja de datos de seguridad del material. Indica los riesgos de exposición al material y de las consideraciones para actuar al respecto [1].
Instrucción de trabajo (WI)	WI (Work instruction). Es un documento en donde se indican los pasos a seguir en la ejecución de una determinada tarea [2].
Ciclohexanona	La ciclohexanona es un líquido inflamable, de color entre incoloro y amarillo pálido con un olor similar al de la acetona. La ciclohexanona se emplea principalmente en la fabricación de nylon y como adhesivo en el sellado de objetos de PVC [3].
Robot	Máquina o ingenio electrónico programable que es capaz de manipular objetos y realizar diversas operaciones [4].
Robot colaborativo	Robot que permite la automatización de las tareas repetitivas y tediosas que no se pueden automatizar con robots tradicionales. El robot colaborativo hace el papel de asistente del trabajador [5].
Sistema de visión	Consiste en la incorporación de métodos usados para adquirir, procesar y analizar las imágenes del mundo real con la finalidad de producir una información que pueda ser tratada por una máquina [6].
Reglamento (CE) No 1272/2008	Reglamento sobre clasificación, etiquetado y envasado de sustancias y mezclas.
Infusión intravenosa	Método para incorporar líquidos, como medicamentos, en el torrente sanguíneo [7].

Tabla 1 - Conceptos y términos utilizados

3.1 La cámara de goteo

La cámara de goteo o cuentagotas (*drip chamber*) es un dispositivo médico utilizado en el proceso de infusión de soluciones líquidas a pacientes (infusión intravenosa). El cuentagotas tiene la utilidad de reducir la velocidad del líquido de infusión, así como evitar la formación de burbujas. Este dispositivo médico está conformado principalmente por dos componentes, uno es la tapa la

cual posee la entrada del líquido de infusión y el otro componente es la cámara (en este proyecto referenciada como "pieza base") en donde las gotas caen y luego salen a través del puerto de salida que también se encuentra en esta pieza [8].

Imagen 1. Cámara de goteo durante proceso de infusión intravenosa, disponible en: http://cpd.anmfvic.asn.au/events/viewitem/455

3.2 Proceso de ensamblaje actual

El proceso de ensamblaje actual se realiza manualmente por un operador. El operador toma la pieza base, la coloca en la boquilla de un dispensador de adhesivo y la mantiene dentro durante el tiempo especificado en la instrucción de trabajo (*Work instruction*). Al finalizar este tiempo, el operador toma la tapa y la coloca sobre la pieza base terminando de esta forma el ensamblaje.

La pieza ya ensamblada es colocada en una bandeja para que permanezca inmóvil y transcurra el tiempo de secado del adhesivo. A partir de este paso, la bandeja es enviada a las estaciones de trabajo siguientes para continuar con las inspecciones de calidad, y demás procesos.

Es necesario aclarar que tanto la pieza base como las tapas son recibidas dentro de bolsas. Como los componentes no se encuentran en posiciones específicas, el operador debe tomar cada componente y colocarlo en la posición que sea requerida para trabajar.

Un inconveniente importante del proceso manual es el riesgo de salud del operador al este encontrarse expuesto al adhesivo. También, existen riesgos de calidad para el producto debido al proceso ejecutado por el operador, esto se puede observar en la tabla siguiente:

Paso	**Descripción de la ejecución**	**Riesgo**	**Control**
Colocación de adhesivo en el punto de conexión de la pieza base	a) Tomar la pieza base de la bolsa y colocarla en la boquilla del dispensador de adhesivo. b) Esperar el tiempo indicado en el WI correspondiente c) Retirar la pieza base de la boquilla del dispensador de adhesivo.	a) Exceso o falta de adhesivo en la pieza debido al incumplimiento del tiempo de espera - **riesgo funcional.** b) Mancha de adhesivo por exceso de solvente o por exceso de movimiento de la pieza dentro de la boquilla del dispensador - **riesgo cosmético.** c) Fuga en pieza final debido a falta de adhesivo - **riesgo funcional.**	a) Instrucción de trabajo (WI) con el tiempo de espera claramente indicado b) Inspección visual en proceso posterior al ensamblaje c) Prueba de fuga en proceso posterior al ensamblaje.

Paso	Descripción de la ejecución	Riesgo	Control
Colocación de la tapa	a) Tomar la tapa y colocarla sobre la pieza base.	a) Incompleta colocación de la tapa en la pieza base **- riesgo funcional.**	a) Inspección visual en proceso posterior al ensamblaje b) Prueba de fuga en proceso posterior al ensamblaje.
Colocación del producto ensamblado en la bandeja de secado	a) Tomar el producto previamente ensamblado y colocarlo con el menor movimiento posible en la bandeja de secado.	a) Aparición de fuga/s en la pieza final debido a movimientos durante el tiempo de secado **- riesgo funcional.**	a) Prueba de fuga en proceso posterior al ensamblaje.

Tabla 2 - Riesgos y control del proceso manual

3.3 Análisis de los riesgos de la exposición al adhesivo

Para analizar el riesgo por parte del operador a la exposición del adhesivo, se revisa el MSDS del adhesivo (Ciclohexanona). Según el MSDS (*Material Safety Data Sheet*) los riesgos presentados son los siguientes:

Riesgo	Categoría	Código en reglamento (CE) No 1272/2008	Consejos de prudencia
Líquido y vapores inflamables	3	H226	a) Mantener alejado de fuentes de calor, chispas, llama abierta o superficies calientes. b) Llevar guantes/prendas/gafas/máscara de protección.
Nocivo en caso de ingestión	4	H302	a) No comer, beber ni fumar durante su utilización. b) Lavarse concienzudamente tras la manipulación.

Riesgo	Categoría	Código en reglamento (CE) No 1272/2008	Consejos de prudencia
Nocivo en caso de inhalación	4	H332	a) No comer, beber ni fumar durante su utilización. b) Lavarse concienzudamente tras la manipulación. c) Asegure ventilación adecuada d) Revise periódicamente acumulación de vapores en el área y nivel de exposición del personal
Nocivo en contacto con la piel	4	H312	a) No comer, beber ni fumar durante su utilización. b) Lavarse concienzudamente tras la manipulación. c) Llevar guantes/prendas/gafas/máscara de protección.
Provoca irritación cutánea	2	H315	a) Lavarse concienzudamente tras la manipulación. b) Llevar guantes/prendas/gafas/máscara de protección.
Provoca lesiones oculares graves	1	H318	a) Llevar guantes/prendas/gafas/máscara de protección.

Tabla 3 - Riesgos de la exposición al adhesivo

Como se puede apreciar en la tabla anterior, los riesgos con mayor impacto son el de lesión ocular grave (categoría 1) y de irritación cutánea (categoría 2). Ambos riesgos se encuentran presentes en el proceso actual, puesto que el operador debe manipular las piezas tanto para colocar el adhesivo como para realizar el ensamblaje. Por lo tanto, la aplicación automatizada debería ofrecer una solución para la mitigación de estos riesgos.

3.4 Proyectos similares y/o soluciones ofrecidas en el mercado

Se realizó una búsqueda de proyectos similares dentro del recolector de ciencia abierta "Recolecta" y no se encontraron coincidencias con este proyecto. Con relación a las soluciones disponibles en el mercado, se pueden mencionar las de las empresas: A UNO TEC [9], Yuhuan Zhengri Technology [10], y ENGEL [11]. Estas soluciones son máquinas especializadas en la producción de cámaras de goteo y la cantidad de unidades producidas por estos equipos puede ser

bastante elevada. No se encontraron soluciones automatizadas sencillas para producción a menor escala. De todas formas, no se descarta la posibilidad de que diversas empresas tengan sus soluciones hechas a medida, así como se pretende realizar en este proyecto.

4 PLAN DE TRABAJO

Este plan fue definido con el siguiente orden de prioridades y tareas a ser ejecutadas:

- Definición de la aplicación y el alcance de esta. Esto se realiza con el objetivo de delimitar las operaciones a realizar y considerar la complejidad de la programación.

- Definición de la forma de interacción entre los equipos. Este paso consiste en definir de qué manera se puede realizar la intercomunicación de los equipos para evitar que los robots sean llamados al mismo tiempo o que algún componente actuara cuando no le correspondiera. En este punto se decide tomar cada componente como si fuese una especie de "caja negra" y definir únicamente mensajes de entrada y salida suficientes para que cada componente funcione con un programa independiente.

- Obtención de los puntos en el sistema de coordenadas de cada robot. Esto se realiza para poder ingresar los valores en el programa del sistema de visión y que este pueda enviar las coordenadas correctamente.

- Programación del Sistema de visión y depuración inicial. La programación como tal se realizada en el Software Sherlock. Para hacer la depuración se requiere el uso del software "Hércules", para simular el intercambio de mensajes con los Robots.

- Programación del Robot ABB y depuración inicial. En esta etapa se realiza la programación del Robot ABB utilizando el software RobotStudio. Para probar la interacción con el Sistema de Visión, se le envían coordenadas provisionales y resto de mensajes requeridos utilizando el programa Hércules.

- Programación del Robot UR3 y depuración inicial. En esta etapa se realiza la programación del Robot UR3 utilizando el Software Polyscope. Para probar la interacción con el Sistema de Visión, se le envían coordenadas provisionales y los mensajes requeridos utilizando el programa Hércules.

- Integración y depuración final. En esta etapa se prueban todos los equipos funcionando al mismo tiempo y se realizan pruebas para hacer la depuración final de los programas.

5 DESARROLLO DEL PROYECTO Y RESULTADOS OBTENIDOS

5.1 Equipos utilizados

Los equipos utilizados para la realización de este proyecto son los siguientes:

Equipo	Modelo	Descripción de uso y/o información adicional
Cámara	CV-M77, de la compañía JAI	Se utiliza para adquirir la imagen a ser procesada.
Computador del sistema de visión	IPD VA 40 - Vision Appliance de la compañía Dalsa	Funciona como servidor y almacena el programa de análisis de la imagen.
Robot ABB	IRB 140 M2004, de la compañía ABB (No. de serie: 14M-51846)	Se utiliza para colocar el adhesivo en la pieza base. Este Robot cuenta con su respectivo controlador y mando (*Teach pendant*). La carga máxima que puede manipular el Robot es de 6 kg
Robot colaborativo	UR3, de la compañía Universal Robots (No. de serie: 2016331072)	Se utiliza para colocar la tapa sobre la pieza base. Este robot cuenta con su respectivo controlador y mando (*Teach pendant*). La carga máxima que puede manipular el Robot es de 3 kg
Cinta transportadora	No disponible	Se utiliza para desplazar las piezas bases hasta el punto en donde se realizaría la inspección por parte del sistema de visión y el ensamblaje por parte de los robots.

Tabla 4 - Equipos utilizados

5.2 Aplicaciones de software utilizadas

Las aplicaciones de software utilizadas para la implementación de este proyecto son:

Aplicación software	Versión	Descripción de uso y/o información adicional
Sherlock	7.2	Software utilizado para crear el programa del sistema de visión de la aplicación.
Robotstudio	V6.08	Software utilizado para desarrollar el programa del Robot ABB

Aplicación software	Versión	Descripción de uso y/o información adicional
Polyscope	URSoftware 3.7.2.40245	Software utilizado para desarrollar el programa del Robot colaborativo UR3
Hercules SETUP utility	3.2.3	Software utilizado para simular el intercambio de mensajes entre los diferentes equipos. Esto fue requerido en la etapa de depuración y pruebas.

Tabla 5 - Aplicaciones de software utilizadas

5.3 Definición del flujo de trabajo en la celda robotizada

El proceso automatizado utilizando la celda robotizada inicia con el operador tomando las piezas bases y colocándolas en la bandeja. Además, este también toma las tapas y las coloca en la plataforma fija. Esta plataforma estaría ubicada en una zona cercana a la entrada de la cinta transportadora para evitar desplazamientos innecesarios (en el laboratorio se aprovechó el espacio de la plataforma rotatoria ya que de todas formas no podía ser retirada de su respectivo lugar). Ver imagen 2 en donde se muestra la distribución de los equipos en el laboratorio de Robótica.

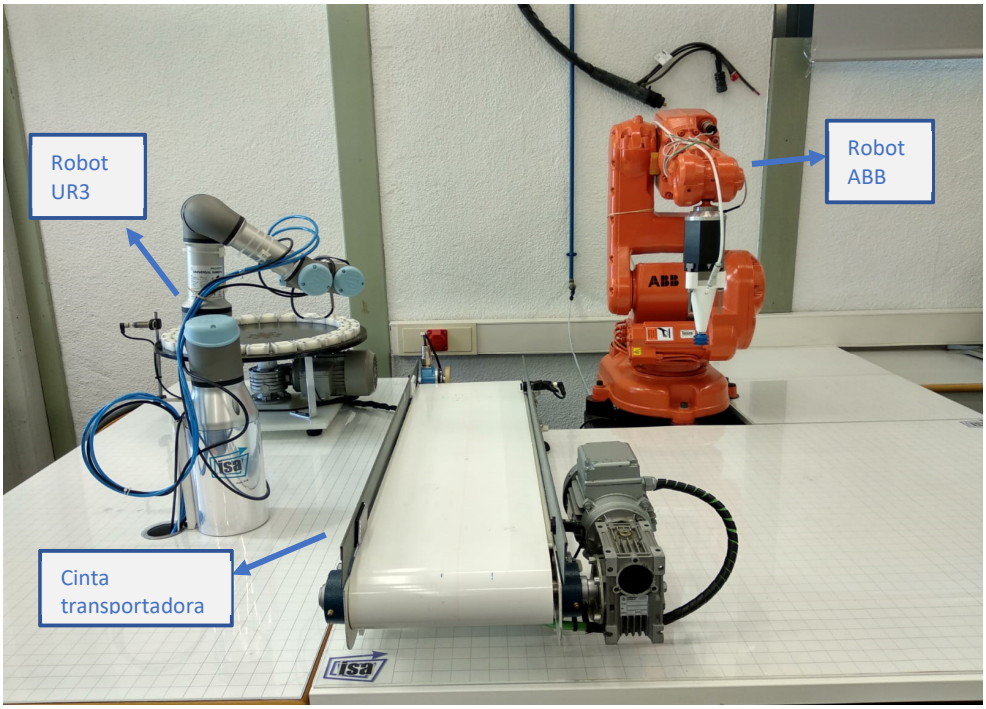


Imagen 2. Celda robotizada – laboratorio de Robótica

Se inicia la investigación de Sherlock la cual requiere la confirmación que el robot ABB y el UR3 están conectados, la cinta transportadora se mueve hasta que la bandeja (previamente colocada por el operador) activa el sensor al final del recorrido. Se adquiere la imagen de las piezas bases y el programa del sistema de visión determina las coordenadas centrales.

Una vez las coordenadas centrales de las piezas han sido determinadas, estas son convertidas al sistema de coordenadas de cada robot. Primero se envían las coordenadas para el Robot ABB, este Robot indica al Sistema de Visión (por medio de un mensaje) que está en movimiento y al finalizar la aplicación de adhesivo en ambas piezas, envía el mensaje indicando que su operación ha concluido.

Al finalizar la operación por parte del Robot ABB, se procede con el envío de las coordenadas centrales al robot UR3. Este robot toma las tapas y las coloca en sus respectivas piezas bases (ver imagen 3 en donde se aprecian los diferentes componentes del ensamblaje). De igual forma le indica al Sistema de Visión cuando está en movimiento y cuando ha concluido satisfactoriamente su operación.

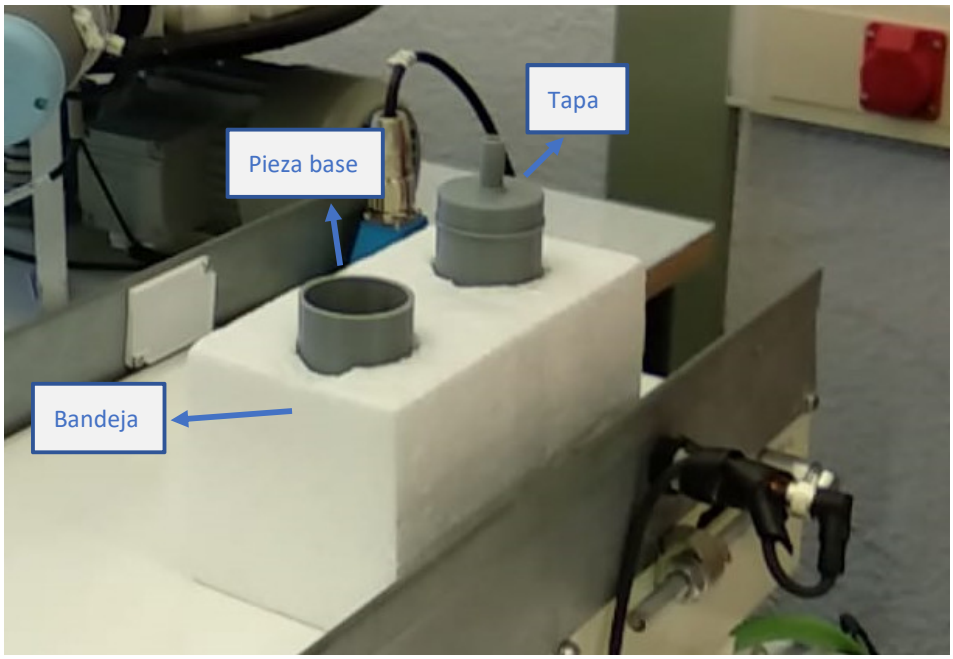


Imagen 3. Bandeja y componentes del ensamblaje

El diseño de la cámara de goteo fue realizado en SolidWorks y se imprimió en 3D. Ver en la imagen 4 el diseño en SolidWorks. El esquema de la cámara de goteo se puede observar en el anexo 11.5.

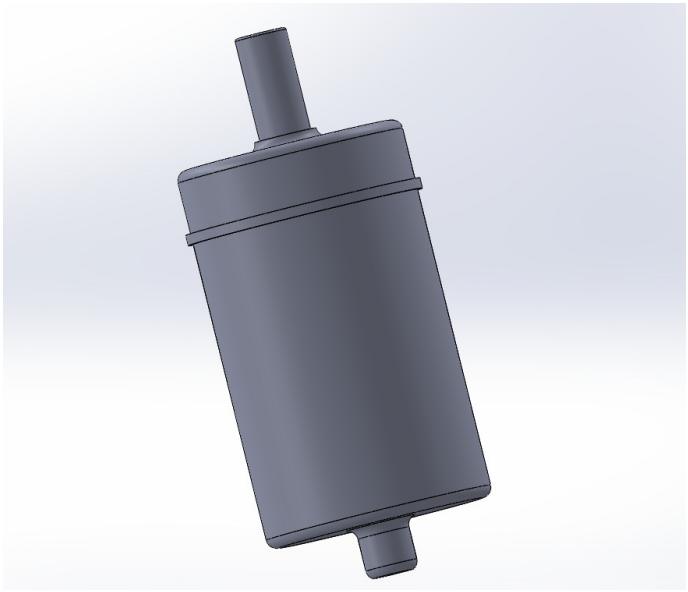


Imagen 4. Diseño en SolidWorks de la cámara de goteo

Como se puede apreciar en la explicación de las operaciones en la celda robotizada, el operador ya no se expondría a los riesgos de salud propios del adhesivo ya que el único momento en donde tocaría las piezas sería para alimentar al sistema. Nótese que, debido a la forma de recepción de los componentes, es necesario mantener al operador para la entrada de los componentes a la celda robotizada de forma que sea posible trabajar con los mismos. A continuación, se muestra una tabla con los riesgos previamente determinados en el proceso manual y la respectiva mitigación con el sistema automatizado:

Paso	Riesgo	Control en el proceso automatizado
Colocación de adhesivo en el punto de conexión de la pieza base	a) Exceso o falta de adhesivo b) Mancha de adhesivo por exceso de solvente c) Fuga en pieza final debido a falta de adhesivo	a) Activación y desactivación de la salida de adhesivo controlada por software.

Paso	Riesgo	Control en el proceso automatizado
Colocación de la tapa	a) Incompleta colocación de la tapa en la pieza base	a) Colocación de la pieza con precisión en la coordenada determinada por el sistema de visión. b) Permanencia de la pieza inmóvil en la bandeja después del ensamblaje.
Colocación del producto ensamblado en la bandeja de secado	a) Aparición de fuga/s en la pieza final debido a movimientos durante el tiempo de secado	a) Permanencia de la pieza inmóvil en la bandeja después del ensamblaje.

Tabla 6 - Control de riesgos en el proceso automatizado

Es importante mencionar, que el uso del proceso automatizado tiene otro tipo de riesgos inherentes. Normalmente estos riesgos son mitigados con la implementación de planes de mantenimiento preventivo considerando los requerimientos de cada uno de los equipos utilizados.

5.4 Calibración del sistema

Debido a que la aplicación requiere que dos robots (cada uno con su propio sistema de coordenadas) alcancen puntos en el espacio, determinados por el programa de visión, es necesario calibrar respecto al sistema de coordenadas correspondiente. Para ello, en el programa de visión se utiliza la función "*calibrate using points*".

La función "*Calibrate using points*" se encarga de asignar un sistema de calibración utilizando 4 puntos de referencia en la imagen (en formato pixeles) y sus puntos equivalentes en el mundo real. La equivalencia en el mundo real está dada por la coordenada determinada por el robot en uso según su propio sistema de coordenadas.

Como es necesario enviar las coordenadas para ambos robots, se crean dos sistemas de calibración distintos (ver Imagen 5).

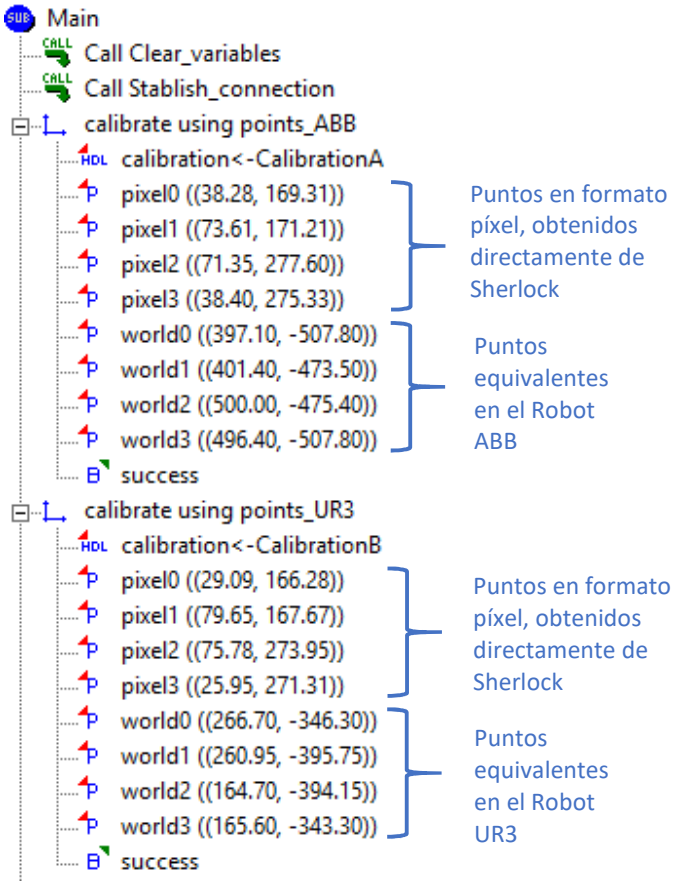


Imagen 5. Calibración Visión-Robots

Los puntos de calibración fueron determinados utilizando imágenes de la bandeja con las piezas reales para poder garantizar un resultado preciso. Si se utiliza una plantilla de calibración tradicional (plantilla plana con círculos impresos) colocada sobre la cinta transportadora, esto no permitiría recrear la perspectiva ni los efectos de sombra generados por las piezas reales (ver imagen 6).

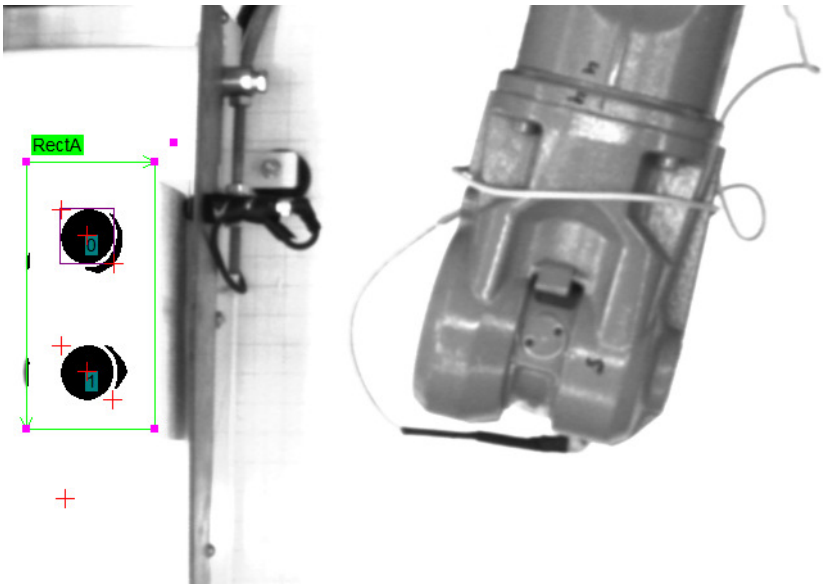


Imagen 6. Puntos de calibración en Sherlock

De igual forma, utilizando las piezas reales en la posición correcta, se movió de forma manual el brazo robótico deseado hasta el centro de cada pieza (ver imagen 7). Una vez posicionado en el centro de la pieza, se anotó la coordenada de la posición actual y esta fue ingresada dentro de los puntos equivalentes "*World*".

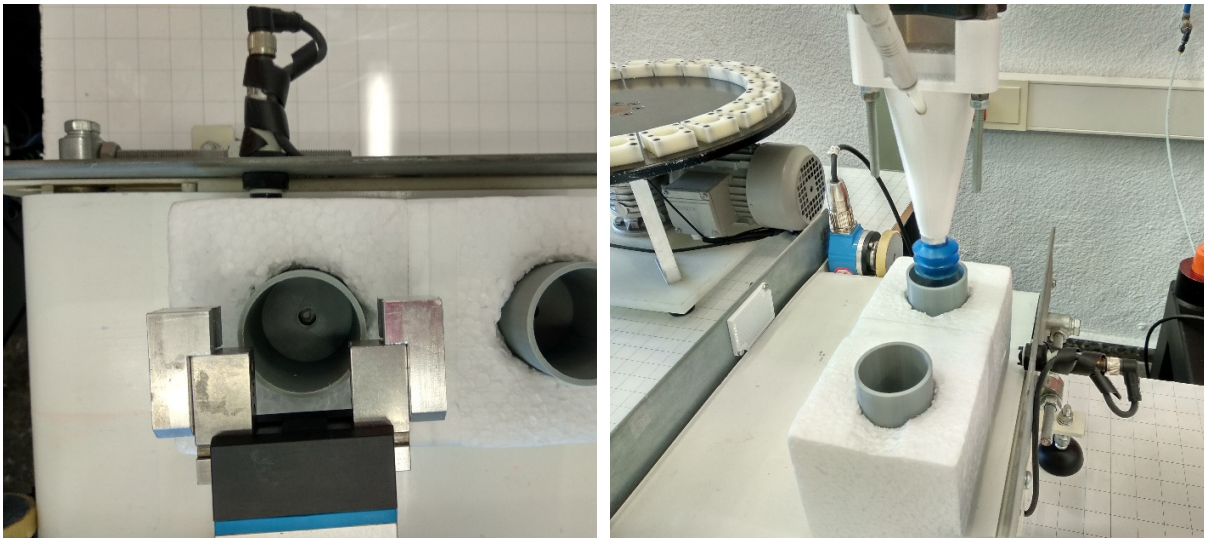


Imagen 7. Posicionamiento de los robots

Para obtener la coordenada actual del robot, una vez este ha sido ha posicionado donde se requiere, es necesario dirigirse a la pestaña "movimiento" en el "*Teach pendant*" (ver imagen 8).

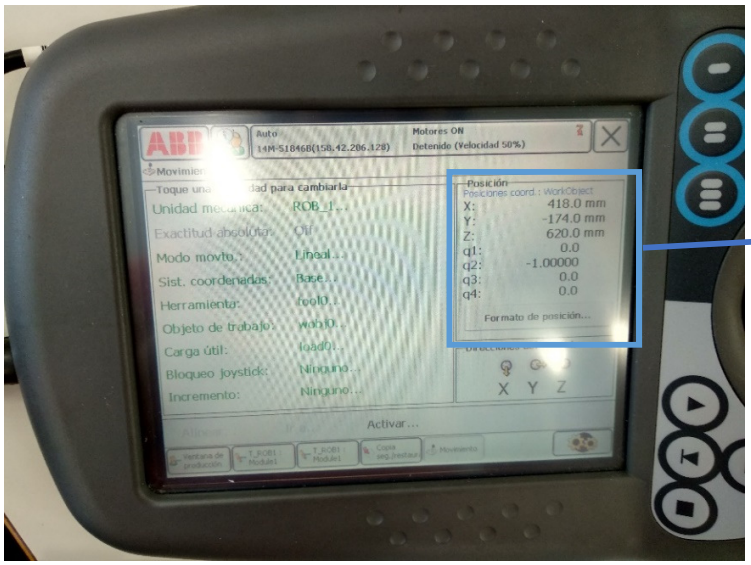


Imagen 8. Pestaña "movimiento"

Con respecto al Robot Colaborativo UR3, la coordenada actual se puede observar en su respectivo "*Teach Pendant*" dentro de la pestaña "Mover" (ver imagen 9).

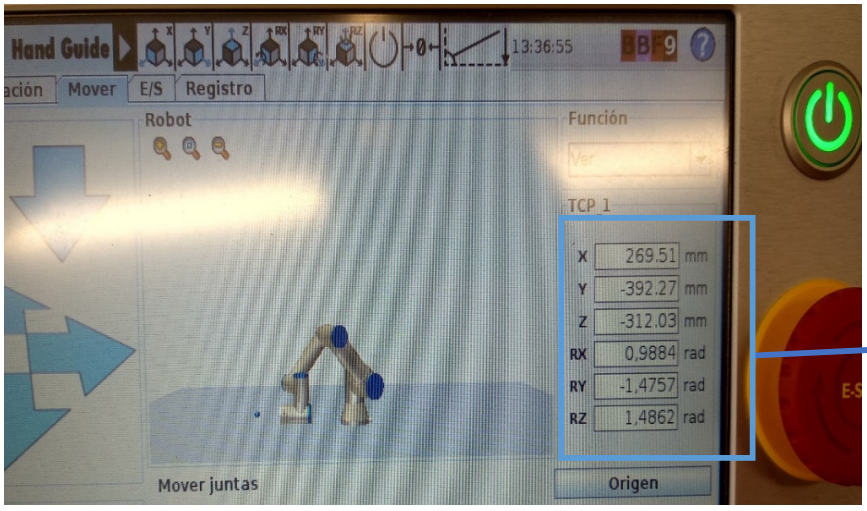


Imagen 9. Pestaña "Mover"

5.5 Programa del sistema de visión

Este programa funciona como el servidor de toda la aplicación, mientras que los robots (ABB y UR3) funcionan como clientes. El objetivo principal de este programa es, determinar la coordenada de cada pieza, para enviarla al robot requerido en el momento preciso. Los pasos principales que se realizan son los siguientes:

a. Desde "*Main*" se realiza la Llamada de la subrutina "*Clear_variables*" para inicializar las variables utilizadas en el programa. Para estos fines, se utiliza un script puesto que resulta ser una forma más sencilla sin tener que utilizar una función para cada una de las variables. Solo se utilizan dos funciones de "*SetString*" para inicializar las variables que no permitían ser editadas en el script (ver imagen 10).

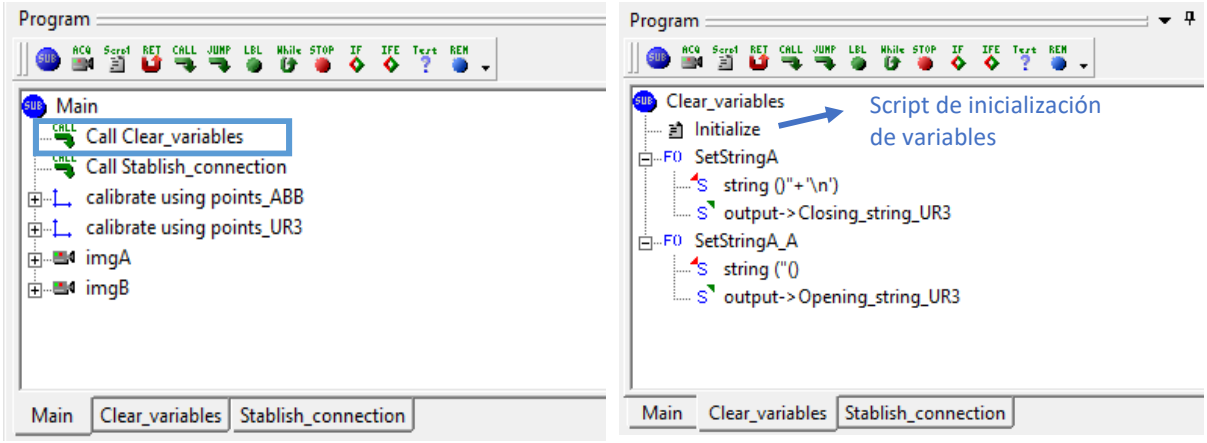


Imagen 10. Clear_variables

b. El siguiente paso consiste en establecer la conexión con los robots, por medio de la subrutina "*Stablish_connection*" (ver imagen 11). Esta subrutina realiza 3 intentos de conexión para cada robot. Si después de los 3 intentos no se confirma la conexión de alguno de los robots, el programa es interrumpido.

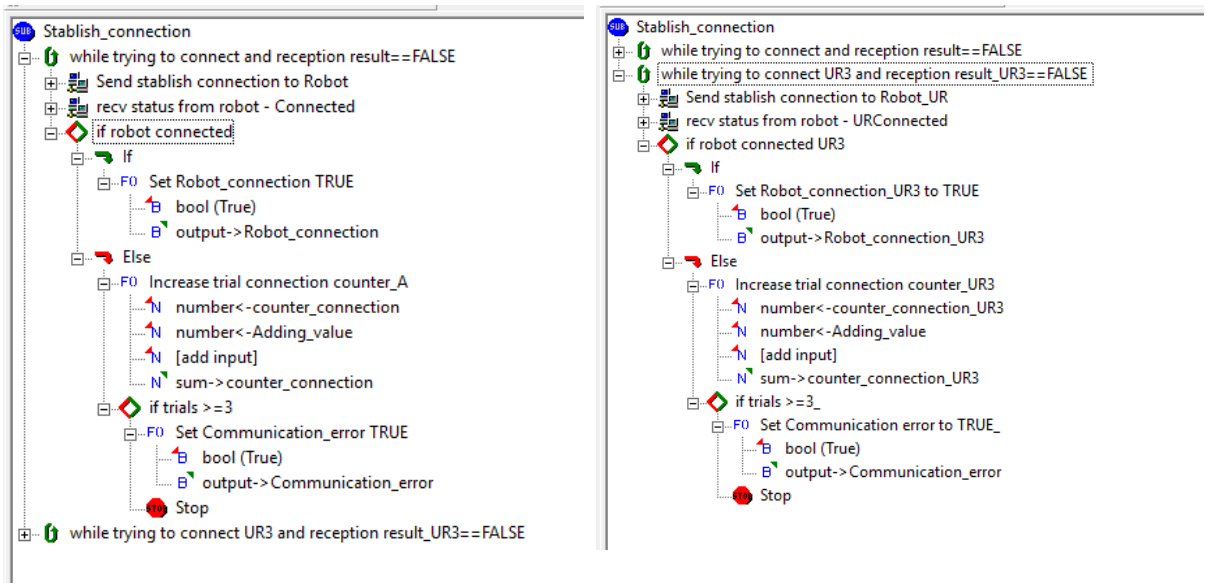


Imagen 11. Stablish_connection

Para que la conexión se considere satisfactoria se debe recibir desde el robot ABB un mensaje que diga "*Connected.*", mientras que del UR3, se debe recibir un mensaje que diga "*URConnected.*"

c. Se realiza la adquisición de la Imagen y se carga en la ventana "ImgA" que a su vez alimenta a "ImgB". Esta última ventana solo utiliza la imagen en modo monocromático de 8 bits, a causa de que la aplicación desarrollada no requiere que se identifiquen características a color.

d. Dentro de ImgB se realizan todas las operaciones requeridas para obtener las coordenadas de las piezas. Primero se utiliza un ROI para detectar las piezas con ayuda del preprocesador "*Threshold*" y el algoritmo "*Connectivity Binary*". Entendiéndose que un preprocesador es una herramienta que consiste en tomar una imagen como "entrada", realizar algún tipo de manipulación/operación sobre ella y dar esa imagen final como "salida". Mientras que el algoritmo, toma una imagen como "entrada", extrae información de esta y da como "salida" esa información [12]. La tabla 7, muestra los parámetros más relevantes utilizados para cada herramienta.

Herramienta	Parámetro	Descripción	Valor
Threshold	Threshold	Valor de referencia para comparar con el valor en escala de grises de los pixeles dentro del ROI.	130
	Below value	Valor adoptado por los pixeles seleccionados dentro del ROI cuando éstos originalmente tengan un valor menor al indicado dentro de "threshold"	0 (negro)
	Above value	Valor adoptado por los pixeles seleccionados dentro del ROI cuando éstos originalmente tengan un valor superior al indicado dentro de "threshold".	255 (blanco)
Connectivity Binary	Black blobs	Parámetro que indica si el algoritmo debe enfocarse en seleccionar los elementos o "blobs" de color negro dentro del ROI (una vez estos ya se encuentran con valores binarios).	True
	8 way	Cuando es verdadero (true), indica que el algoritmo funciona conectando 8 pixeles vecinos, y cuando es falso indica que funciona conectando 4 pixeles vecinos.	True
	Min area	Area mínina del elemento o "blob" detectado	1000
Connectivity Binary	Max area	Area máxima del elemento o "blob" detectado	1500
	Min width	Anchura mínima del elemento o "blob" detectado	35
	Max width	Anchura máxima del elemento o "blob" detectado	50
	Min height	Altura mínima del elemento o "blob" detectado	35
	Max height	Altura máxima del elemento o "blob" detectado	50

Tabla 7 - Herramientas en Sherlock [12]

e. Se confirman cuantas piezas bases fueron detectadas (puesto que puede que el operador solo haya colocado una sola pieza en la bandeja). Si se detecta que hay al menos una pieza la coordenada se carga en la variable "Point_part0" y se procede con la primera parte de

estructuración de la variable general de coordenadas, la cual consiste en convertir el primer punto, de variable tipo "point" a "String" (ver imagen 12). En caso de no detectarse ninguna coordenada se pasa la variable "Point _part0" a tipo "String" solo que tendría un valor de punto "0,0".

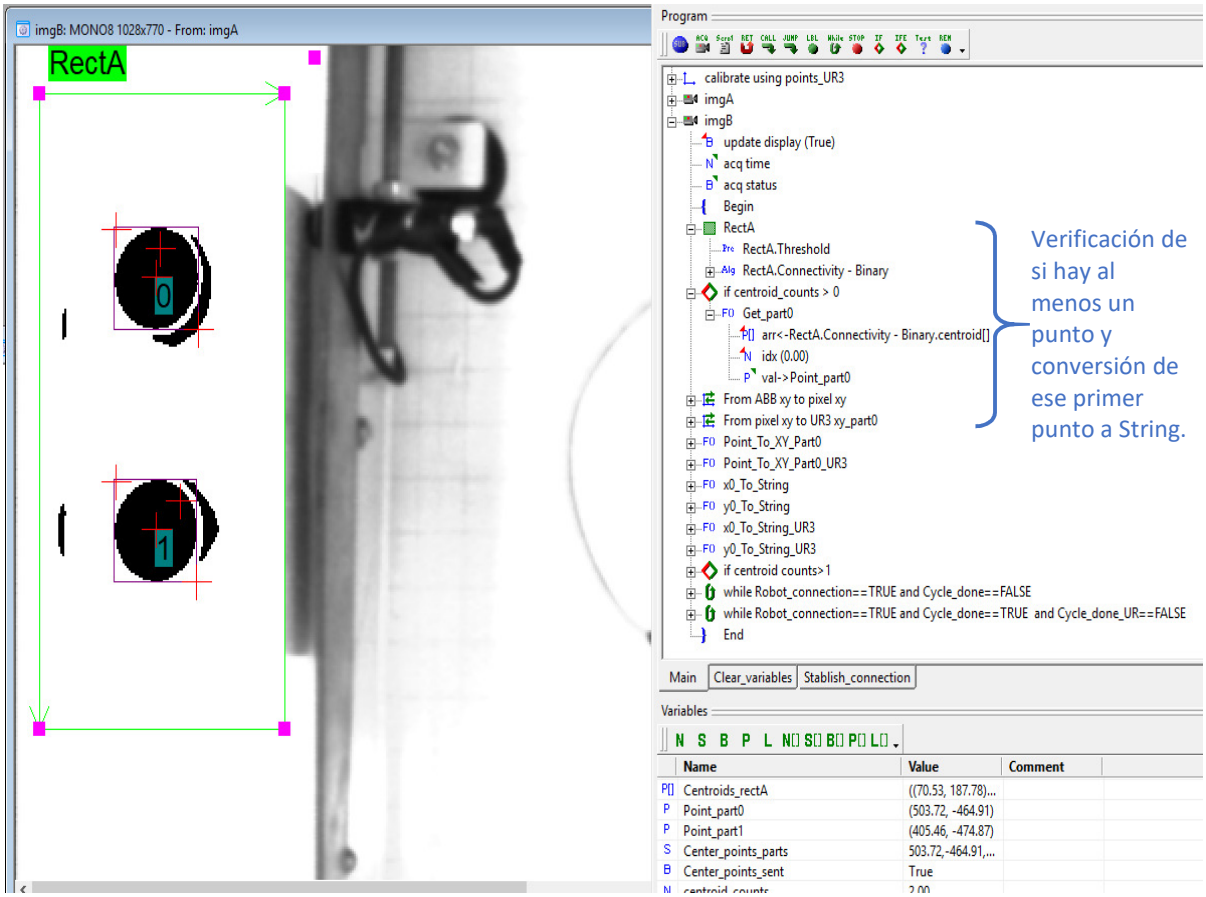


Imagen 12. Vista en Sherlock y Verificación de la primera pieza

f. Se verifica si se ha detectado más de una pieza. En caso de que esa condición sea verdadera, se carga la coordenada del segundo punto en su respectiva variable (Point_part1), se convierte en tipo String y se procede con la estructuración de la variable general de coordenadas (ver imagen 13).

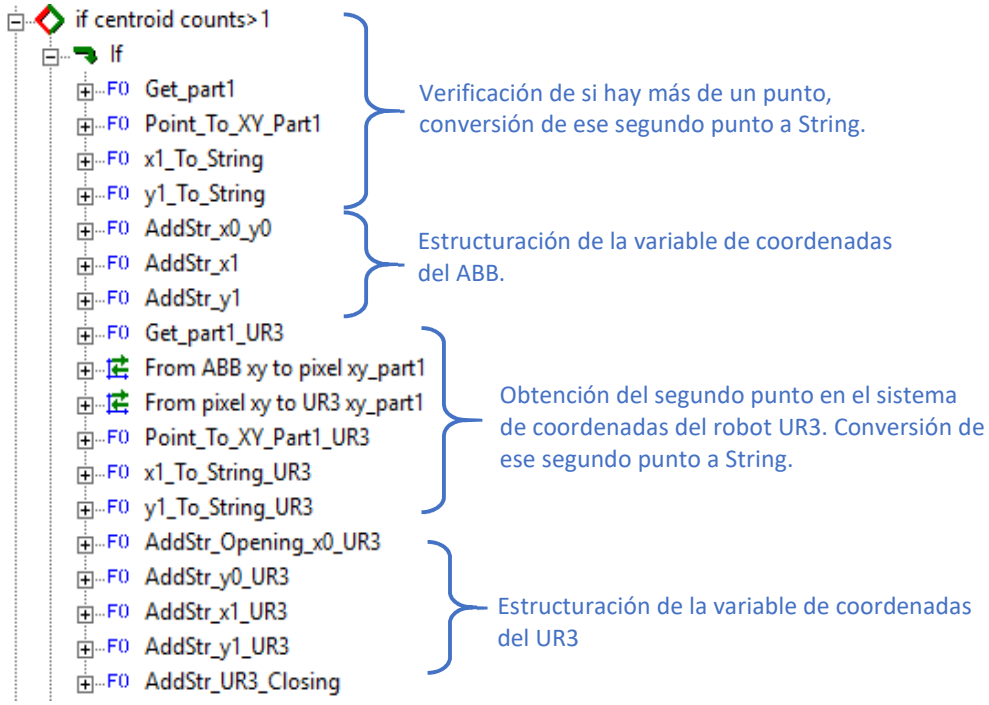


Imagen 13. Verificación de la segunda pieza

La variable general de coordenadas para el robot ABB recibe el nombre de "Center_points_parts", mientras que para el caso del UR3 recibe el nombre de "Center_points_parts_UR3".

Para realizar la estructuración de la variable de coordenadas, cada punto debe ser convertido en tipo String y configurar el formato necesario para que el respectivo robot pueda utilizar ese mensaje cuando sea recibido. El mensaje con las coordenadas enviadas al Robot ABB tiene la siguiente forma:

$$Center\ points\ parts = x_{0\ ABB}, y_{0\ ABB}, x_{1\ ABB}, y_{1\ ABB}, \quad (1)$$

Mientras que el mensaje con las coordenadas enviadas al Robot UR3 tiene la siguiente forma:

$$Center\ points\ parts\ UR3 = "(\mathrm{x}_{0\ \mathrm{UR3}}, \mathrm{y}_{0\ \mathrm{UR3}}, \mathrm{x}_{1\ \mathrm{UR3}}, \mathrm{y}_{1\ \mathrm{UR3}})" + '\backslash n' \quad (2)$$

g. En caso de que no se haya detectado más de una pieza (o ninguna pieza en lo absoluto), se completa la estructura de la variable de coordenadas con un punto de valor "0,0" para la segunda pieza.

h. Se procede con el envío de las coordenadas al Robot ABB y se espera que este envíe el mensaje "InCycle.". Cuando el Robot finaliza su ciclo, envía el mensaje "Done." Si el mensaje no se recibe satisfactoriamente en un tiempo determinado, la inspección en curso es finalizada y se coloca en "True" la variable "Communication_error" puesto que es probable que exista un problema de conexión (o un error mayor) con el Robot (ver imagen 14).

Imagen 14. Puntos de calibración en Sherlock

i. Inmediatamente se recibe el mensaje de finalización del ciclo del robot ABB, se procede con las operaciones del robot UR3. Se envía la variable general de coordenadas al robot

UR3, y se espera a que este envíe el mensaje “URInCycle.”. Cuando el Robot finaliza su ciclo, envía el mensaje “URDone.”

Si el mensaje “URDone.” no se recibe satisfactoriamente en un tiempo determinado, la inspección en curso es finalizada y se coloca en “True” la variable “Communication_error” (ver imagen 15).

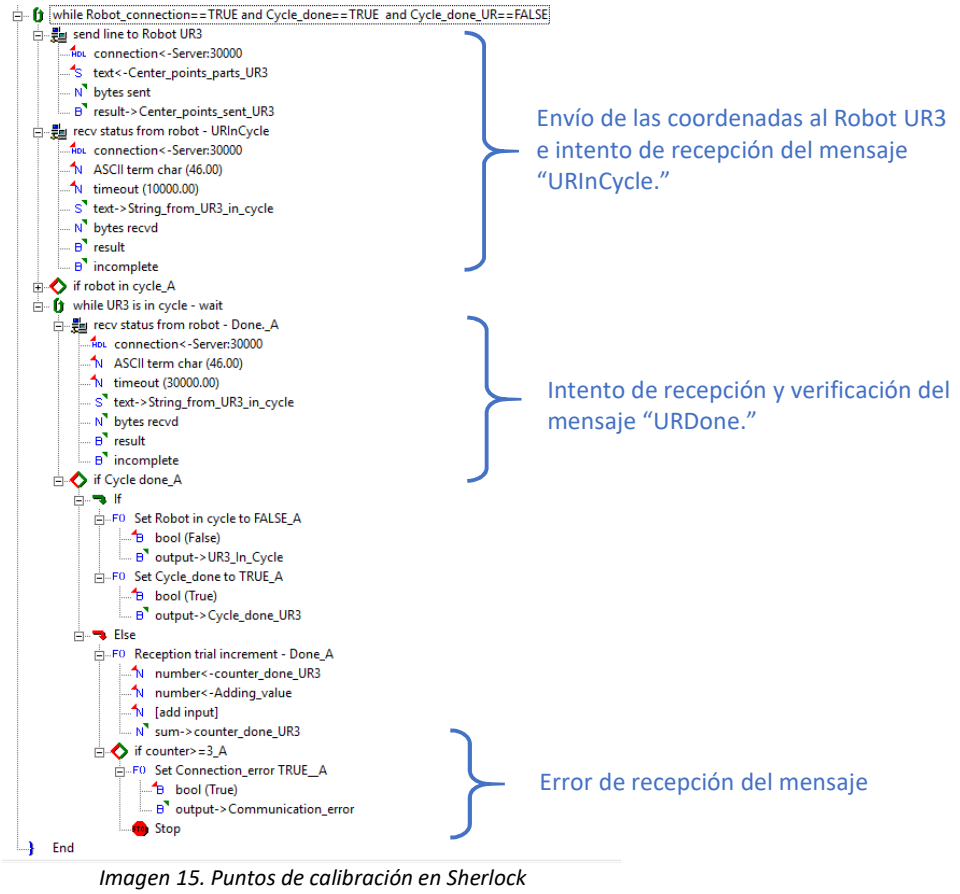


Imagen 15. Puntos de calibración en Sherlock

Diagrama de flujo: este diagrama se encuentra disponible en el anexo 11.1

5.6 Programa del Robot ABB

Este programa se encarga de controlar los movimientos que deben ser realizados por el Robot ABB, así como de manejar el intercambio de mensajes entre el Robot ABB y el sistema de visión. Está conformado por 3 subrutinas: "*Main*", "*Clear_variables*" y "*Update_variables*". Las instrucciones y funciones propias del lenguaje Rapid, que son utilizadas por este programa son las siguientes:

Nombre	Descripción del funcionamiento
StrLen	Se utiliza para obtener la longitud de una cadena de caracteres.
StrFind	Se utiliza para buscar en una cadena, a partir de una posición especificada, un carácter que se encuentra dentro de un conjunto determinado de caracteres. Una vez lo encuentra, devuelve la posición del primer carácter, ya sea en o después de la posición especificada.
StrPart	Se utiliza para encontrar una parte de una cadena y obtenerla como una cadena nueva.
StrToVal	Se utiliza para convertir una cadena de caracteres en un valor de cualquier tipo de dato.
Offs	Se utiliza para desplazar una posición dada en el Robot
SocketCreate	Se utiliza para crear un nuevo zócalo de conexión.
SocketConnect	Se utiliza para conectar el zócalo a un ordenador remoto. En este caso para conectarse con el ordenador del Sistema de visión.
SocketReceive	Se utiliza para recibir datos de un ordenador remoto. En este caso para recibir los datos específicamente del ordenador del Sistema de Visión.
SocketSend	Se utiliza para enviar datos a un ordenador remoto.
SocketClose	Se utiliza cuando ya no se va a utilizar una conexión de zócalo. Una vez cerrado un zócalo, no es posible utilizarlo en ninguna llamada a zócalo, excepto SocketCreate.
WaitDI	Se utiliza para esperar hasta que se activa una entrada digital.
Set	Se utiliza para poner a uno el valor de una señal digital de salida.
Reset	Se utiliza para poner a cero el valor de una señal digital de salida.
MoveL	Se utiliza para trasladar el punto central de la herramienta (TCP) en sentido lineal hacia un punto de destino determinado.
MoveC	Se utiliza para trasladar el punto central de la herramienta (TCP) en sentido circular hacia un punto de destino determinado.

Tabla 8 - Instrucciones/Funciones en RAPID [13]

Los pasos principales ejecutados por este programa son los siguientes:

a. Cuando se inicia el programa, una vez se han creado las variables (ver imagen 16), se salta a la subrutina "*Main*", que como primer paso carga a la subrutina "*Clear_variables*". Esta última se encarga de inicializar las variables del programa relacionadas con los puntos y la activación de la ejecución de los movimientos por el Robot.

```
VAR num x0:=0;
VAR num y0:=0;
VAR num x1:=0;
VAR num y1:=0;
VAR bool cycle:=FALSE;

VAR num var_pos;
VAR num counter;
VAR num trials;
VAR bool ok;

VAR string xy;
VAR num string_length;
VAR num end_position;

VAR string Robot_send:="";
VAR socketdev socket_Robot_PC;
VAR string received_string;
CONST robtarget Point_0:=[[0,0,576],[0,1,0,0],[-1,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Point_1:=[[0,0,576],[0,1,0,0],[-1,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Safety_point_mid:=[[464,-476,608],[0,1,0,0],[-1,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Safety_point_up:=[[418,-174,620],[0,1,0,0],[-1,-1,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

VAR robtarget p0_0;
VAR robtarget p0_1;
VAR robtarget p0_2;
VAR robtarget p0_3;
VAR robtarget p0_4;

VAR robtarget p1_0;
VAR robtarget p1_1;
VAR robtarget p1_2;
VAR robtarget p1_3;
VAR robtarget p1_4;
```

Puntos centrales genéricos y puntos de seguridad

Puntos para trazar los movimientos circulares

Imagen 16. Creación de las variables

Nótese que las variables "*Point_0*" y "*Point_1*", las cuales son los puntos de partida para las coordenadas centrales de las piezas, tienen ya definido el valor de z (en mm). Esto es porque el Sistema de Visión solo puede determinar los valores correspondientes a los ejes "x", y "y" puesto que se trata de una imagen (2D). También se han definido los valores de la orientación de la herramienta. Esta orientación fue generada automáticamente por el Robot al este ser posicionado manualmente en unos puntos cercanos a los puntos de funcionamiento.

b. Se establece la conexión con el sistema de visión y se inician los intentos de recepción. Si el mensaje es recibido satisfactoriamente se evalúa su contenido para definir las operaciones siguientes (ver imagen 17).

```
PROC main()
!Add your code here
  Clear_variables;
  TPErase;
  SocketCreate socket_Robot_PC;
        !Robot
     SocketConnect socket_Robot_PC,"158.42.16.207",1024;

WHILE trials<3 AND Robot_send<>"Done." DO
  SocketReceive socket_Robot_PC\Str:=received_string;
  TPWrite (received_string);

!-----------------------------------------------------

 TEST received_string

    CASE "Connection":
      TPWrite(received_string);
      Set CONVEYOR_FWD;
      WaitDI CONVEYOR_OBJ_SEN,1;
       Reset CONVEYOR_FWD;
        WaitTime 1;
        SocketSend socket_Robot_PC\Str:="Connected.";
        Robot_send:="Connected.";
    DEFAULT:
       IF (StrLen(received_string))>10 THEN
        Update_variables;
        cycle:=TRUE;
        SocketSend socket_Robot_PC\Str:="InCycle.";
         Robot_send:="InCycle.";
       ENDIF
    ENDTEST

!-----------------------------------------------------
```

Llamada de subrutina "Clear_variables" y establecimiento de la conexión con el servidor.

Recepción del mensaje enviado por el servidor y espera del fin de ciclo ("Done", o que los intentos de conexión se hayan finalizado).

Se evalúa el mensaje recibido para comprobar si el contenido es "Connection".

En su defecto se evalúa la longitud del mensaje.

Imagen 17. Puntos de calibración en Sherlock

Como se pudo observar en la imagen 17, se evalúa el mensaje recibido para comprobar si el contenido es "*Connection*".

Si el contenido del mensaje es "*Connection*", el Robot ABB inicia el movimiento de la cinta transportadora hasta que la bandeja llegue hasta el sensor final de carrera. Se detiene el movimiento de la cinta, se espera un segundo para garantizar que la pieza esté detenida y se envía al sistema de visión el mensaje "*Connected*.".

El tiempo de un (1) segundo es importante ya que inmediatamente el robot ABB envía su respectivo mensaje (*Connected.*), el sistema de visión realiza la adquisición de la imagen. Si para ese momento, las piezas no se encuentran estáticas, las coordenadas obtenidas no representarían la ubicación real de estas.

Por otro lado, por defecto, se evalúa la longitud del mensaje. Si este tiene más de 10 caracteres se presume que se ha recibido el mensaje de coordenadas de las piezas. Por tal razón, se llama a la subrutina "*Update_variables*" para que se encargue de convertir esas coordenadas a puntos que el Robot pueda utilizar. Una vez se ha hecho este paso, se coloca la variable "*cycle*" en "*true*" y el Robot indica que se encuentra en ciclo, enviando el mensaje "*InCycle*" al sistema de visión.

c. Dentro de la subrutina "*Update_variables*" se inicia el proceso de conversión de las coordenadas dentro del mensaje (tipo *String*) a puntos que sean útiles para el Robot. Como se puede observar en la imagen 18, se utiliza un bucle "*While*" para recorrer toda la cadena de caracteres recibida como mensaje y seccionarla punto a punto.

```
PROC Update_variables()
    var_pos:=1;
    counter:=0;
    string_length:=StrLen(received_string);

    WHILE (counter<5 AND var_pos<string_length) DO
    end_position:=StrFind(received_string,var_pos,",");
    xy:=StrPart(received_string,var_pos, end_position-var_pos);
    var_pos:=StrFind(received_string,var_pos,",")+1;
    counter:=counter+1;

    IF counter=1 THEN
         ok:=StrToVal(xy, x0);
    ENDIF
    IF counter=2 THEN
        ok:=StrToVal(xy, y0);
    ENDIF
    IF counter=3 THEN
         ok:=StrToVal(xy, x1);
    ENDIF
    IF counter=4 THEN
        ok:=StrToVal(xy, y1);
    ENDIF

 ENDWHILE
    p0_0:=Offs(Point_0,x0,y0,0);
    p0_1:=Offs(p0_0,-15,0,0);
    p0_2:=Offs(p0_0,0,15,0);
    p0_3:=Offs(p0_0,15,0,0);
    p0_4:=Offs(p0_0,0,-15,0);

    p1_0:=Offs(Point_1,x1,y1,0);
    p1_1:=Offs(p1_0,-15,0,0);
    p1_2:=Offs(p1_0,0,15,0);
    p1_3:=Offs(p1_0,15,0,0);
    p1_4:=Offs(p1_0,0,-15,0);

ENDPROC
```

Recorrido del mensaje recibido en búsqueda de los separadores (","). Una vez encuentra un separador se guarda su posición y se

Se actualiza el valor del contador de caracteres y se verifica el valor actual. Dependiendo dicho valor el punto encontrado es

Una vez obtenido el punto central para la pieza 0, se definen los puntos de la

Una vez obtenido el punto central para la pieza 1, se definen los puntos de la

Imagen 18. Subrutina "Update_Variables"

Nótese que la condicionante "*counter*<*5*" se utiliza por la cantidad de separadores (",") que se conoce tiene el mensaje (en total son 4 separadores). Este contador de los separadores es también el que se utiliza como referencia para determinar a cuál punto corresponde esa sección del mensaje.

d. Se retorna a "*Main*" y se procede con la realización de los movimientos del Robot (ver imagen 19).

```
IF cycle=TRUE THEN

    SingArea\Wrist;
    MoveL Safety_point_up,v50,z1,tool0\WObj:=wobj0;  !Punto de seguridad

     IF x0 <> 0 AND y0 <>0 THEN
    MoveL p0_0,v100,z0,tool0\WObj:=wobj0;
    MoveL p0_1,v100,z0,tool0\WObj:=wobj0;
    !activar la valvula del adhesivo
    Set GRIPPER_CLOSE;
    WaitTime 0.5;
    MoveC p0_2,p0_3,v50,fine,tool0\WObj:=wobj0;
    MoveC p0_4,p0_1,v50,fine,tool0\WObj:=wobj0;
    !desactivar la válvula del adhesivo
    Reset GRIPPER_CLOSE;
    WaitTime 0.5;
    MoveL Safety_point_mid,v100,z1,tool0\WObj:=wobj0;
     ENDIF

    IF x1<>0 AND y1<>0 THEN
    MoveL Safety_point_mid,v100,z1,tool0\WObj:=wobj0;
    MoveL p1_0,v100,z0,tool0\WObj:=wobj0;
    MoveL p1_1,v100,z0,tool0\WObj:=wobj0;
   Set GRIPPER_CLOSE;
    WaitTime 0.5;
    MoveC p1_2,p1_3,v50,fine,tool0\WObj:=wobj0;
    MoveC p1_4,p1_1,v50,fine,tool0\WObj:=wobj0;
   Reset GRIPPER_CLOSE;
    WaitTime 0.5;
    MoveL Safety_point_mid,v100,z1,tool0\WObj:=wobj0;
     ENDIF

     MoveL Safety_point_up,v100,z1,tool0\WObj:=wobj0;
    SocketSend socket_Robot_PC\Str:="Done.";
     Robot_send:="Done.";
    WaitTime 0.25;
ENDIF

!-----------------------------------------------------------------------
       WaitTime 1;
       trials:=trials+1;

ENDWHILE

     SocketClose socket_Robot_PC;
ENDPROC

ENDMODULE
```

Se verifica si las coordenadas recibidas para la pieza 0 son diferentes de cero. De ser así, se realizan los movimientos correspondientes para llegar a la

Se verifica si las coordenadas recibidas para la pieza 1 son diferentes de cero. De ser así, se realizan los movimientos correspondientes para llegar a la

Se envía el robot a la posición de seguridad y este le envía al sistema de visión el mensaje "Done."

Al finalizar el ciclo se cierra el zócalo de conexión.

Imagen 19. Gestión de los movimientos del Robot ABB

Diagrama de flujo: este diagrama se encuentra en el anexo 11.2

5.7 Programa del Robot UR3

Este programa se encarga de controlar los movimientos que deben ser realizados por el Robot UR3, así como de manejar el intercambio de mensajes entre el Robot UR3 y el sistema de visión. Las instrucciones y funciones propias del lenguaje de URScripting, que son utilizadas por este programa son las siguientes:

Nombre	Descripción del funcionamiento
Socket_open	Se utiliza para abrir un zócalo de conexión ethernet TCP/IP
Socket_send_string	Se utiliza para enviar un mensaje tipo "string" a través del zócalo abierto.
Socket_read_ascii_float	Se utiliza para leer un número determinado de "floats" en formato ascii que han sido recibidos a través del zócalo de conexión.
MoveJ	Se utiliza para mover el Robot al punto especificado sin necesidad de garantizar una trayectoria lineal.
MoveL	Se utiliza para mover el Robot al punto especificado garantizando una trayectoria lineal
Socket_close	Se utiliza para cerrar el zócalo de conexión previamente abierto.

Tabla 9 - Instrucciones/Funciones en Polyscope [14]

Los pasos principales ejecutados por este programa son los siguientes:

a. El primer paso es la creación de las variables requeridas por el programa (ver imagen 20).

```
Programa
  BeforeStart
    Connection= False
    Rcvd_parts=[0,0,0,0,0]
    Place_cap0=p[0,0,0,0,0,0]
    Place_up_cap0=p[0,0,0,0,0,0]
    cap0_back=p[0,0,0,0,0,0]
    Place_cap1=p[0,0,0,0,0,0]
    Place_up_cap1=p[0,0,0,0,0,0]
    cap1_back=p[0,0,0,0,0,0]
```

Imagen 20. Creación de variables programa UR3

b. Luego se procede con la apertura del zócalo de conexión y el envío del mensaje correspondiente por parte del UR3 (ver imagen 21)

```
Programa de robot
  Connection=socket_open("158.42.16.207",30000)
  Esperar: 0.5
  If Connection= True
    Bucle Rcvd_parts[0]=0
      socket_send_string("URConnected.")
      socket_send_string("URConnected.")
      Esperar: 0.2
      Rcvd_parts=socket_read_ascii_float(4)
      Esperar: 0.2
  If Rcvd_parts[1]≠0  and Rcvd_parts[2]≠0
    Place_cap0=p[Rcvd_parts[1]/1000,Rcvd_parts[2]/1000,95/1000,0.985,-1.4782,1.5024]
    Place_up_cap0=p[Rcvd_parts[1]/1000,Rcvd_parts[2]/1000,110/1000,0.985,-1.4782,1.5024]
    cap0_back=p[Rcvd_parts[1]/1000,-256.76/1000,118/1000,0.985,-1.4782,1.5024]
    socket_send_string("URInCycle.")
    Esperar: 0.2
    socket_send_string("URInCycle.")
    Esperar: 0.1
```

Apertura del zócalo de conexión y envío del mensaje "URConnected." al Sistema de visión si la conexión ha sido satisfactoria.
Luego se espera por las coordenadas centrales de las piezas

Imagen 21. Intercambio mensajes UR3

Si las coordenadas de las piezas se han recibido, la posición cero de la variable "*Rcvd_parts*" toma el valor del total de "*floats*" recibidos y por ende el bucle con la condición "*Rcvd_parts[0]=0*" finaliza. Las posiciones 1 y 2 de la variable "*Rcvd_parts*" almacenan la coordenada central de la primera pieza, mientras que las posiciones 3 y 4 almacenan la coordenada central de la segunda pieza. Nótese que esos valores son enviados por el Sistema de visión en milímetros, pero en este programa son convertidos a metros.

Una vez se actualizan las variables correspondientes (si las coordenadas se han recibido y son valores diferentes de cero), se envía al sistema de visión el mensaje "URInCycle."

c. Se procede con la realización de los movimientos por parte del Robot (ver imagen 22).

```
MoveJ
  safety_point
  go_to_cap0
  Esperar: 1.0
  Ajustar pinza=Encender
  Esperar: 1.5
  Pick_up_cap
  Place_up_cap0
MoveL
  Place_cap0
  Esperar: 1.0
  Ajustar pinza=Apagar
  Esperar: 2.0
  cap0_back
```

Se parte de la posición general de seguridad hasta la posición fija en donde está ubicada la tapa (cap 0 para este caso). Se toma la tapa y se coloca en una posición cercana a la pieza base

Desde ese punto se realiza el desplazamiento lineal hasta la pieza base 0 y de esta forma se coloca la tapa. Una vez ahí, se apaga la pinza para que el Robot de por finalizado ese ensamblaje.

Imagen 22. Ensamblaje primera pieza UR3

d. Se realiza el mismo proceso para la segunda pieza. Se verifica que las coordenadas son distintas de cero, se actualizan las variables requeridas para las posiciones a utilizar por el Robot y se procede con el ensamblaje (ver imagen 23).

```
If Rcvd_parts[3]≠0  and Rcvd_parts[4]≠0
  Place_cap1=p[Rcvd_parts[3]/1000,Rcvd_parts[4]/1000,95/1000,0.985,-1.4782,1.5024]
  cap1_back=p[Rcvd_parts[3]/1000,-256.76/1000,118/1000,0.985,-1.4782,1.5024]
  Place_up_cap1=p[Rcvd_parts[3]/1000,Rcvd_parts[4]/1000,110/1000,0.985,-1.4782,1.5024]
  MoveJ
    go_to_cap1
    Esperar: 0
    Ajustar pinza=Encender
    Esperar: 1.5
    Pick_up_cap
    Place_up_cap1
    Esperar: 0
  MoveL
    Place_cap1
    Esperar: 1.0
    Ajustar pinza=Apagar
    Esperar: 2.0
    cap1_back
    Esperar: 0
```

Se parte de la posición “cap0_back” hasta la posición fija en donde está ubicada la tapa (cap 1 para este caso). Se toma la tapa y se coloca en una posición cercana a la pieza base

Desde ese punto se realiza el desplazamiento lineal hasta la pieza base 1 y de esta forma se coloca la tapa. Una vez ahí, se apaga la pinza para que el Robot de por finalizado ese ensamblaje.

Imagen 23. Ensamblaje segunda pieza UR3

e. Finalmente, el Robot se coloca en la posición de seguridad, se envía el mensaje "*URDone.*" al Sistema de visión, se cierra el zócalo de conexión y se reinician las variables "Connection" y "*Rcvd_parts"* (ver imagen 24).

```
  MoveJ
    safety_point
socket_send_string("URDone.")
Esperar: 0.1
socket_close()
Connection= False
Rcvd_parts=[0,0,0,0,0]
```

Imagen 24. Finalización del ensamblaje

Diagrama de flujo: este diagrama se encuentra disponible en el anexo 11.3

5.8 Resultados obtenidos

Una vez fue finalizada la calibración y programación descrita en los apartados anteriores, la celda robotizada fue capaz de realizar satisfactoriamente el ensamblaje de la cámara de goteo. Como evidencia del resultado se puede observar el video adjunto con esta memoria **"Video_TFM_JT".**

6 ANÁLISIS DE PRESUPUESTO

El análisis de presupuesto se hace de acuerdo con las horas de trabajo para desarrollar la programación requerida por el cliente. En este análisis, aunque se menciona el precio de los Robots y resto de componentes ya existentes (ver punto 6.1), no se considerarían como parte del proyecto ya que se asume que la empresa ya los tenía disponibles. Es importante aclarar que las horas de trabajo colocadas en el presupuesto se han tomado de forma que tenga sentido con una empresa que cuente con experiencia con la instalación y programación de los equipos involucrados. Colocar la cantidad de horas en base al desarrollo como tal del Trabajo fin de máster no tendría sentido ya que no reflejan las horas dedicadas al estudio, investigación y experimentación necesarios antes de realizar correctamente la programación.

6.1 Presupuesto de los componentes previamente comprados por el cliente

Equipo	Descripción	Precio total
Cámara CV-M77	Incluye sensor, cables de conexión, y lentes	800 €
Computador IPD VA 40	Computador del sistema de visión. Incluye licencia de Sherlock	3.000 €
Fuente de alimentación 24 VDC	Fuente de alimentación para el sistema de visión	30 €
Robot ABB	Incluye brazo robótico, controlador, Teach-pendant, además de incluir todos los cables de alimentación y de conexión con el robot.	17.000 €
Luminaria para la cámara	Luminaria para que se pueda obtener una buena imagen por la cámara.	50 €
Robot UR3	Incluye brazo robótico, controlador, Teach-pendant, además de incluir todos los cables de alimentación y de conexión con el robot.	20.250 €
Cinta transportadora	Esto es todo el sistema de la cinta transportadora, es decir, fuente de alimentación, correa, motor, cables de conexión, estructura, controlador, entre otros.	900 €
Sensor fotoeléctrico	Sensor fotoeléctrico utilizado para detectar la bandeja con las piezas	62 €
Botón de paro de emergencia	Pulsador redondo de tipo sostenido, requiere giro para soltar la posición de activación.	30 €
Botón de arranque de la aplicación	Botón pulsador para indicar el arranque de la aplicación	5 €
	Total (solo para referencia):	42.127 €

Tabla 10 – Presupuesto de referencia

6.2 Presupuesto de los servicios de programación y depuración requeridos

Servicio	Precio por hora	Cantidad de horas	Precio total
Programación del sistema de visión y depuración inicial	30 €	60	1800 €
Programación del Robot ABB y depuración inicial	25 €	60	1500 €

Servicio	Precio por hora	Cantidad de horas	Precio total
Programación del Robot UR3 y depuración inicial	25 €	40	1000 €
Integración y depuración final	40 €	20	800 €
		Total:	**5.100** €

Tabla 11 – Presupuesto programación

6.3 Presupuesto de los componentes requeridos y el servicio de instalación

Equipo	Descripción	Precio total
Guardas de seguridad	Utilizadas para evitar que el operador se acerque al espacio de trabajo de los Robots. Estas guardas se colocarían fuera del alcance de los Robots.	1500 €
Sistema del Dispensador de adhesivo	Incluye la cabeza para el dispensador, junto con la válvula y el controlador de temperatura	1200 €
Válvula para actuador UR3	Válvula para la activación del actuador UR3	50 €
Actuador de la herramienta del UR3	Actuador que funciona como pinza para tomar las tapas	625 €
Mangueras y acopladores	Mangueras y acopladores para llevar desde el Robot UR3 hasta la válvula y salida principal del aire.	20 €
Instalación de los componentes anteriores	Incluye la instalación de las mangueras con sus respectivos acoples, el sistema de colocación de adhesivo, guardas de seguridad y válvula para el actuador UR3.	2000 €
	Total:	**5.395** €

Tabla 12 – Presupuesto componentes e instalación

- ***Precio total según el alcance de este proyecto: 10.495 €***

Nota: todos los precios descritos anteriormente incluyen IVA.

Es importante mencionar que en el presupuesto se incluyen las guardas de seguridad (vallado de seguridad). Estas guardas serían colocadas fuera del rango del movimiento de los robots con el objetivo de evitar que el operador se acerque. De esta forma ambos robots tienen libertad de movimiento, sin poner en riesgo la vida del operador.

7 PLIEGO DE CONDICIONES

En esta sección se presenta el pliego de prescripciones técnicas para la contratación de servicios de programación e instalación de componentes requeridos según el alcance del proyecto. Esta sección puede que presente información previamente documentada. Cuando sea necesario se indicará la sección del presente documento en donde está detallada la información requerida.

7.1 Objeto del contrato

Programación de los componentes requeridos por el proyecto "ensamblaje de cámara de goteo utilizando celda robotizada". También se incluye cualquier instalación adicional de componentes para finalmente realizar la puesta en marcha del proyecto.

7.2 Especificaciones técnicas

Para el desarrollo del proyecto, se requiere del adjudicado habilidades de programación en las siguientes plataformas: RobotStudio, Polyscope, y Sherlock. Se debe asegurar el funcionamiento de cada componente por lo que se requieren pruebas individuales antes de realizar la integración de todo el sistema.

7.3 Metodología de trabajo y seguimiento de tareas

El adjudicado deberá ofrecer un plan de trabajo el cual debe ser revisado y aprobado por el adjudicatario. El adjudicatario deberá dar seguimiento a la ejecución del proyecto junto con un representante del organismo adjudicado para asegurarse que se encuentran trabajando dentro del alcance estipulado.

Cuando se considere necesario cualquiera de las partes involucradas (adjudicatario y adjudicado) pueden solicitar una reunión de seguimiento. Esto es importante ya que durante el desarrollo del proyecto existe la posibilidad de encontrar inconvenientes cuya solución sea necesaria discutir. Las reuniones también se pueden requerir para fines informativos de los trabajos realizados y en planes de realización.

7.4 Fase de desarrollo y pruebas

En esta fase el adjudicado deberá trabajar de acuerdo con el plan aprobado. Se seguirá el orden estipulado de componente a programar y las pruebas a realizar. El seguimiento de las tareas en esta fase se hará de acuerdo con lo establecido en el apartado anterior.

7.5 Entregables

El adjudicado deberá entregar el código fuente de cada aplicación al adjudicatario de forma que este pueda tener acceso libre a estos. Por lo tanto, se deben entregar los derechos de modificación de código fuente al adjudicatario. También se deberá entregar la explicación del código fuente para que, en un futuro, en caso de requerirse alguna modificación por parte del adjudicatario, esto sea posible.

7.6 Duración de contrato

El contrato tendrá una vigencia anual desde el 30 de septiembre del 2018 hasta el 30 de septiembre del 2019.

7.7 Documentación final

El adjudicado deberá entregar una memoria en la que explique el trabajo realizado. Esta memoria deberá incluir la explicación del código fuente de cada aplicación utilizada.

7.8 Presupuesto máximo

El presupuesto máximo será de 11.500 € (IVA ya incluido). El detalle del presupuesto por parte del adjudicado se encontrará en la memoria final.

8 MEJORAS PROPUESTAS

Una mejora que podría implementarse es la de utilizar dos herramientas en un mismo robot. Esto permitiría colocar el adhesivo y también colocar la tapa en cada pieza utilizando un solo Robot, permitiendo que el robot restante pueda ser utilizado en otro proyecto. También se recomienda el rediseño de la bandeja para que en esta se puedan colocar las tapas además de las piezas bases. Esto, además de ser más cómodo para el operador, ahorraría tiempo de desplazamiento en el Robot.

También se propone aumentar la cantidad de unidades en la bandeja según la capacidad de colocación por parte del operador. Esta capacidad de colocación puede estar dada según los requerimientos de producción y el estudio de ergonomía referente a los movimientos periodicidad realizados. Finalmente se propone el uso de una cámara que funcione directamente en escala de grises, puesto que la opción de color no es necesaria en la aplicación. Esto permitiría que la cámara actual pueda ser utilizada en otros proyectos en donde sí sea necesaria.

9 CONCLUSIONES

Como conclusión, se puede afirmar que el objetivo del proyecto y los requerimientos específicos del cliente fueron cumplidos satisfactoriamente. Se partió de la idea de que un cliente reutilizó componentes de maquinaria desmantelada y solicitó un proyecto de automatización de un proceso utilizando los equipos facilitados. Todos los equipos facilitados por el cliente fueron utilizados.

El objetivo fundamental, fue automatizar el proceso de ensamblaje de cámara de goteo utilizando una celda robotizada. La cámara de goteo es un dispositivo médico utilizado principalmente en el proceso de infusión de soluciones líquidas a pacientes (infusión intravenosa). Este dispositivo está conformado principalmente por dos componentes; una pieza base o cuerpo, y la tapa.

El proceso de ensamblaje original era manual y presentaba diferentes inconvenientes. Dentro de los inconvenientes se destacan; posibles problemas de calidad en el producto y problemas de salud para el operador que ejecutaba el ensamblaje. Estos problemas fueron mitigados con la automatización realizada en este proyecto.

La automatización realizada consistió en utilizar el Robot ABB para colocar el adhesivo en las piezas bases para que posteriormente el Robot UR3 colocara la tapa a cada una de las piezas. La cinta transportadora se utilizó para desplazar las bandejas con las piezas bases hasta la zona de ensamblaje.

El sistema de visión funcionó como servidor para toda la aplicación. Este mantenía comunicación con cada uno de los robots. Los robots al ser clientes no tenían comunicación entre sí. El código fuente de cada aplicación fue desarrollado en su respectivo software y las explicaciones correspondientes fueron documentadas.

10 REFERENCIAS

[1] TRANSLINK [consulta: 14 de julio 2019]. Disponible en: https://www.translinknet.be/translation/msds/definition.html

[2] VKS, [consulta: 20 de julio 2019]. Disponible en: https://vksapp.com/blog/what-a-work-instruction-is-and-isn-t

[3] INSST, [consulta: 15 de julio 2019]. Disponible en: https://www.insst.es/InshtWeb/Contenidos/Documentacion/TextosOnline/Valores_Limite/Doc_Toxicologica/FicherosSerie2/DLEP%2024.pdf

[4] Real academia española, [consulta: 15 de julio 2019]. Disponible en: https://dle.rae.es/?id=WYRlhzm

[5] Blog universal Robots, [consulta: 15 de julio 2019]. Disponible en: https://blog.universal-robots.com/es/beneficios-robots-colaborativos

[6] Infaimon ©2017 INFAIMON S.L., [consulta: 15 de julio 2019]. Disponible en: https://blog.infaimon.com/sistemas-de-vision-artificial-tipos-aplicaciones/

[7] Instituto nacional del cáncer, [consulta: 15 de julio 2019]. Disponible en: https://www.cancer.gov/espanol/publicaciones/diccionario/def/infusion-intravenosa

[8] Hector Urrutia. Limited Flow rate drip chamber for intravenous fluid delivery system. U.S., Patent, No. 6099512, Ago. 08, 2000

[9] A UNO TEC, [consulta: 10 de julio 2019]. Disponible en: https://www.aunotec.it/en-gb/main-realizations/drip-chamber-automatic-assembly/31

[10] ZHENGRI, [consulta: 10 de julio 2019]. Disponible en: https://www.zhengri-machine.com/project/qz-017-drip-chamber-assembly-machine/

[11] ENGEL global, 15 Nov, 2013 [consulta: 10 de julio 2019]. Disponible en: [] ZHENGRI, [consulta: 10 de julio 2019]. Disponible en: https://www.engelglobal.com/zh/cn/news-press/news-press-releases/detail/news/detail/News/engelatswissplastics2014.html

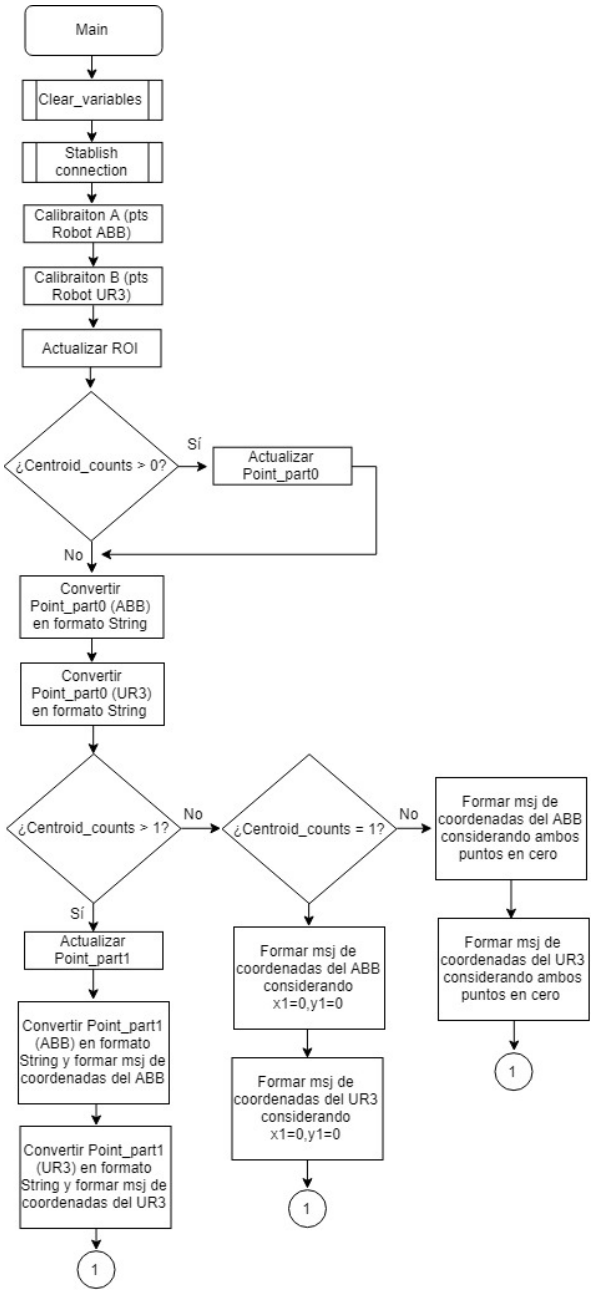
[12] Teledyne Dalsa, Sherlock User's reference Manual, 403-00007-00, September 10, 2014

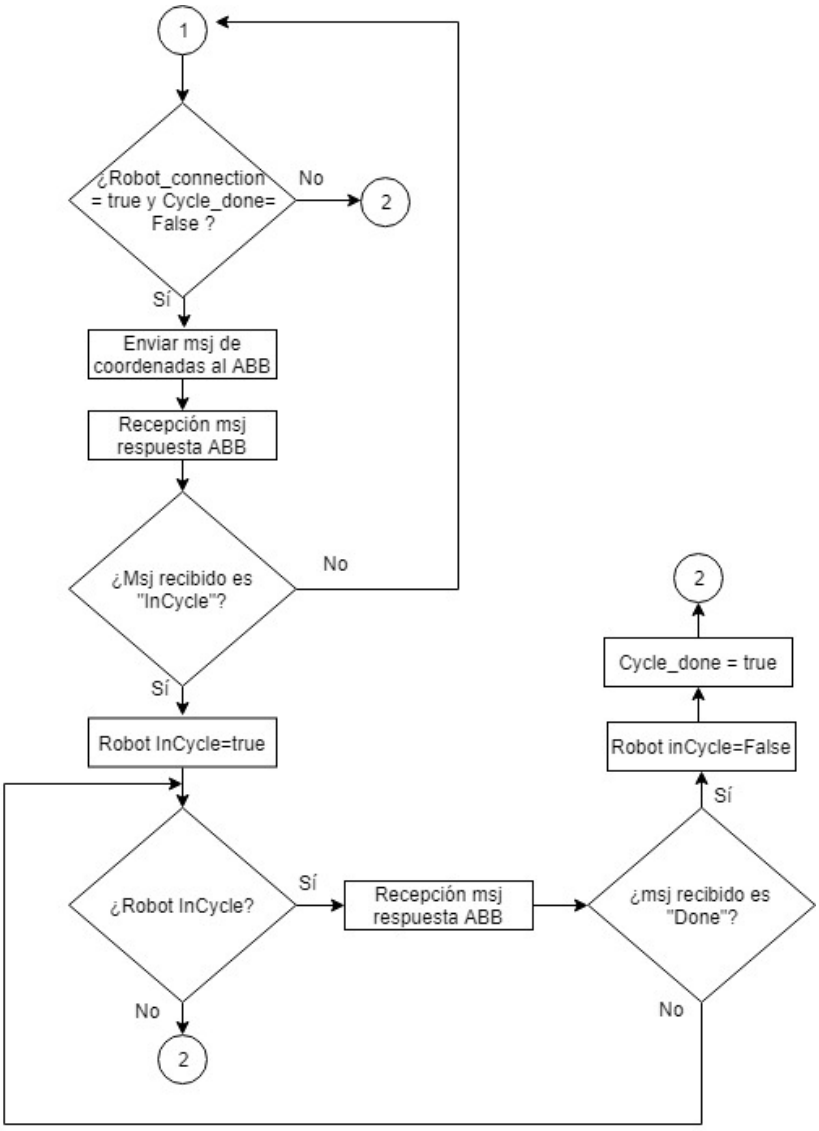
[13] ABB, Manual de referencia técnica instrucciones, funciones y tipos de datos RAPID, 3HAC16581-5, Rev. J

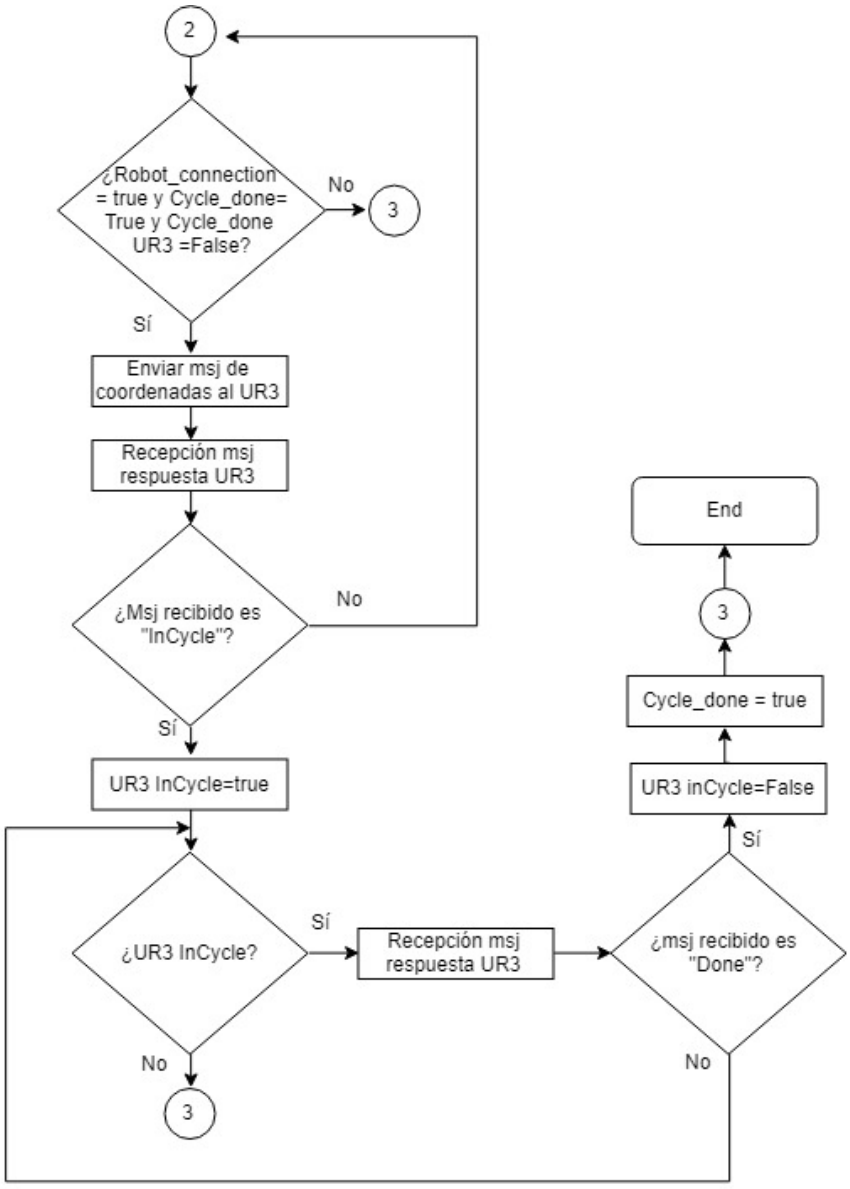
[14] Universal Robots, The URScript Programming Language version 3.5.4, April 12, 2018

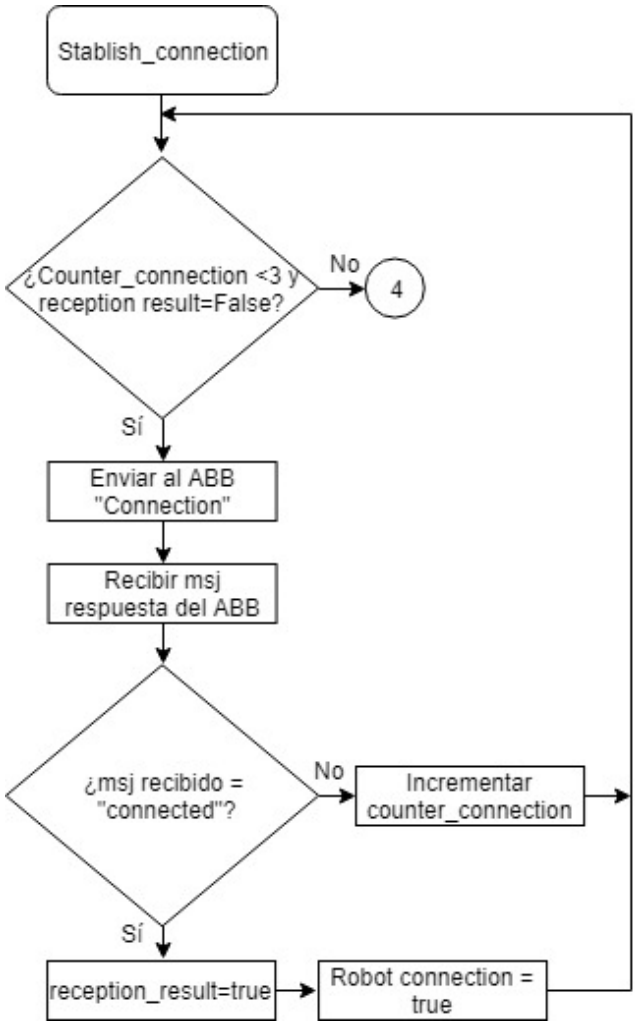
11 ANEXOS

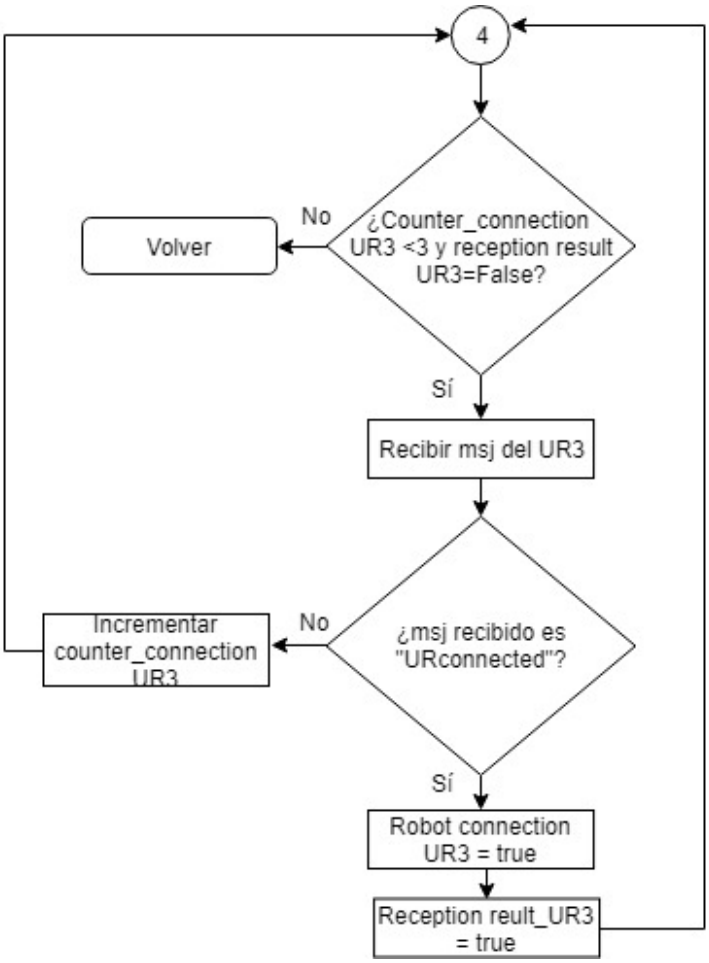
ANEXO 11.1: Diagrama de flujo del programa del sistema de visión

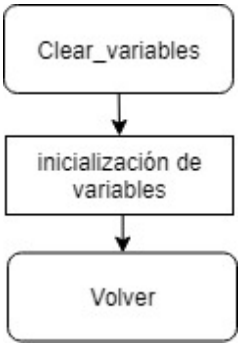



1
¿Robot_connection = true y Cycle_done= False ?
No
2
Sí
Enviar msj de coordenadas al ABB
Recepción msj respuesta ABB
¿Msj recibido es "InCycle"?
No
Sí
Robot InCycle=true
¿Robot InCycle?
Sí
Recepción msj respuesta ABB
¿msj recibido es "Done"?
Sí
No
Robot inCycle=False
Cycle_done = true
2
No
2

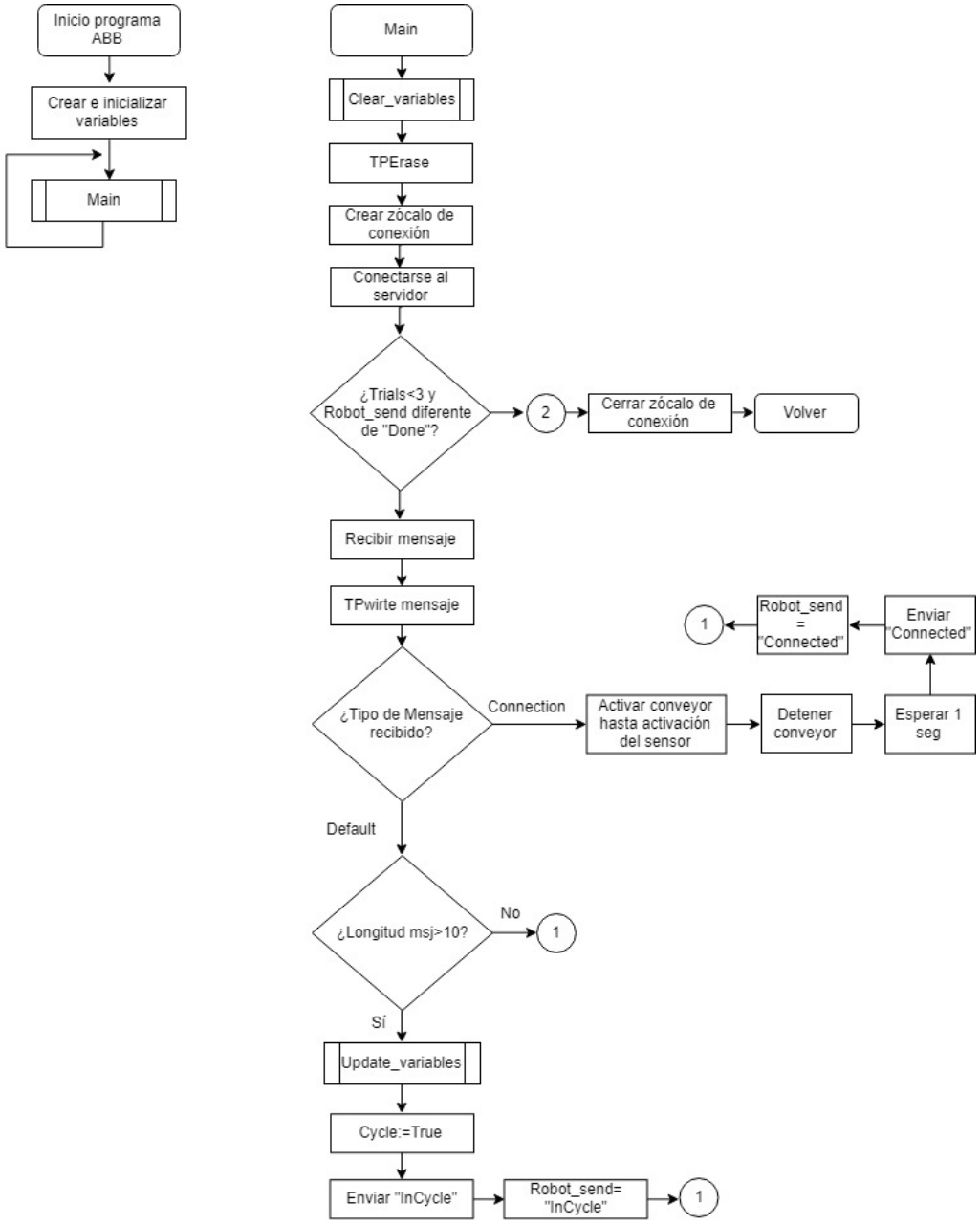

2
¿Robot_connection = true y Cycle_done= True y Cycle_done UR3 =False?
No
3
Sí
Enviar msj de coordenadas al UR3
Recepción msj respuesta UR3
¿Msj recibido es "InCycle"?
No
Sí
UR3 InCycle=true
¿UR3 InCycle?
Sí
Recepción msj respuesta UR3
¿msj recibido es "Done"?
Sí
No
No
3
UR3 inCycle=False
Cycle_done = true
3
End

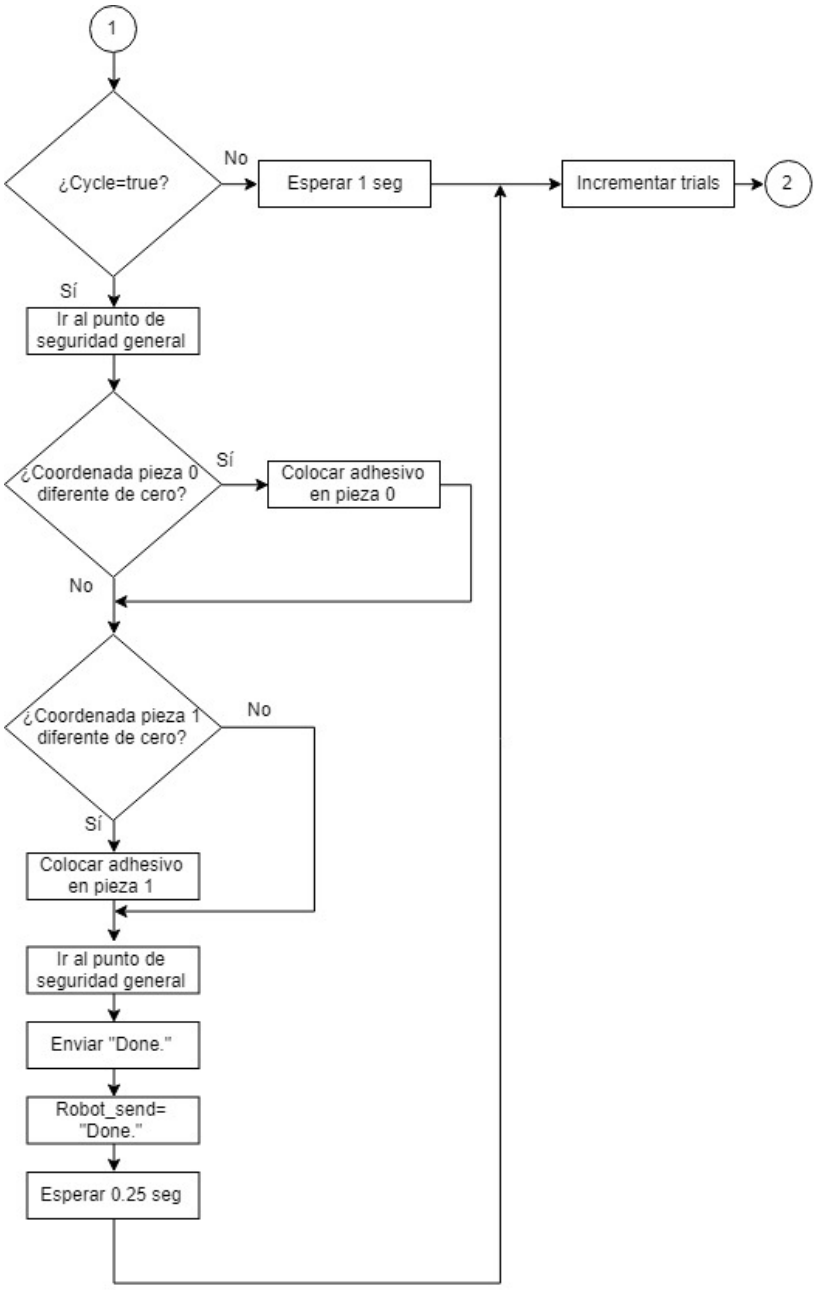

Stablish_connection
¿Counter_connection <3 y reception result=False?
No
4
Sí
Enviar al ABB "Connection"
Recibir msj respuesta del ABB
¿msj recibido = "connected"?
No
Incrementar counter_connection
Sí
reception_result=true
Robot connection = true

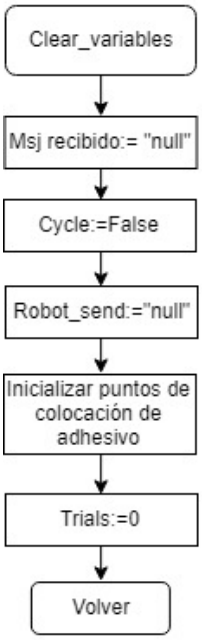

4
¿Counter_connection UR3 <3 y reception result UR3=False?
No
Volver
Sí
Recibir msj del UR3
¿msj recibido es "URconnected"?
No
Incrementar counter_connection UR3
Sí
Robot connection UR3 = true
Reception reult_UR3 = true

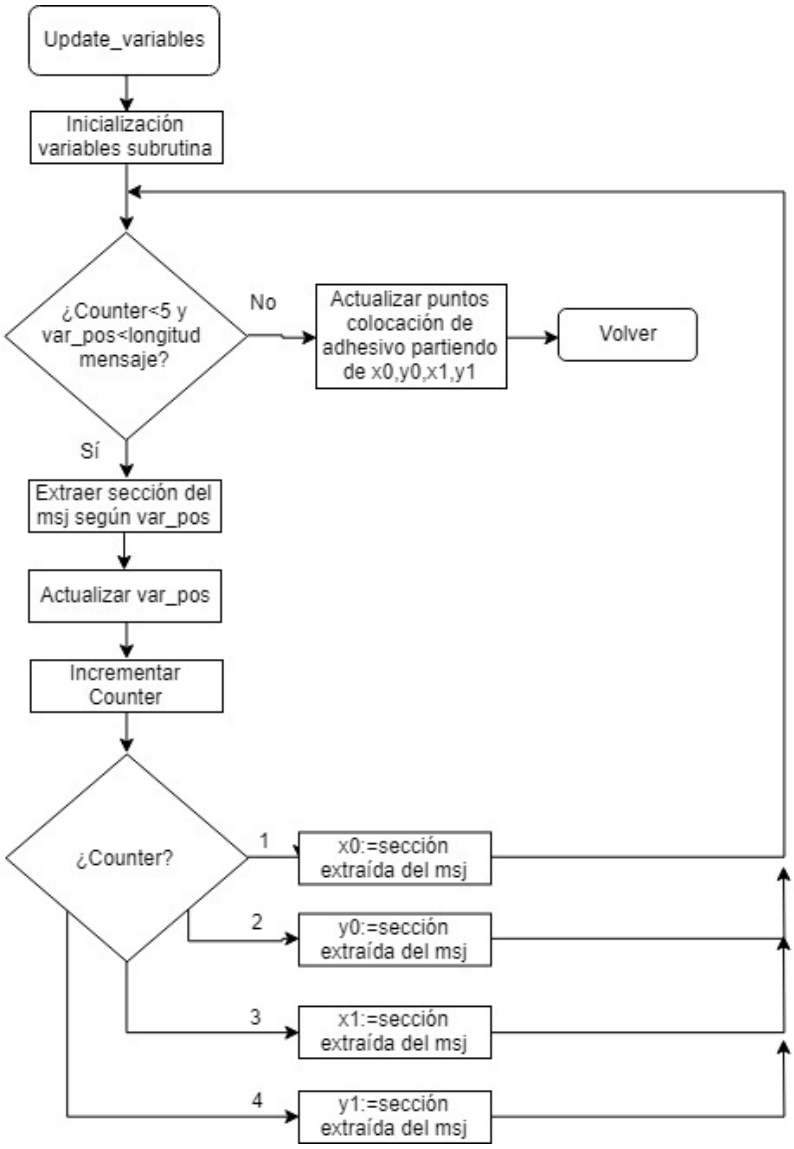

Clear_variables
inicialización de variables
Volver

ANEXO 11.2: Diagrama de flujo del programa del Robot ABB

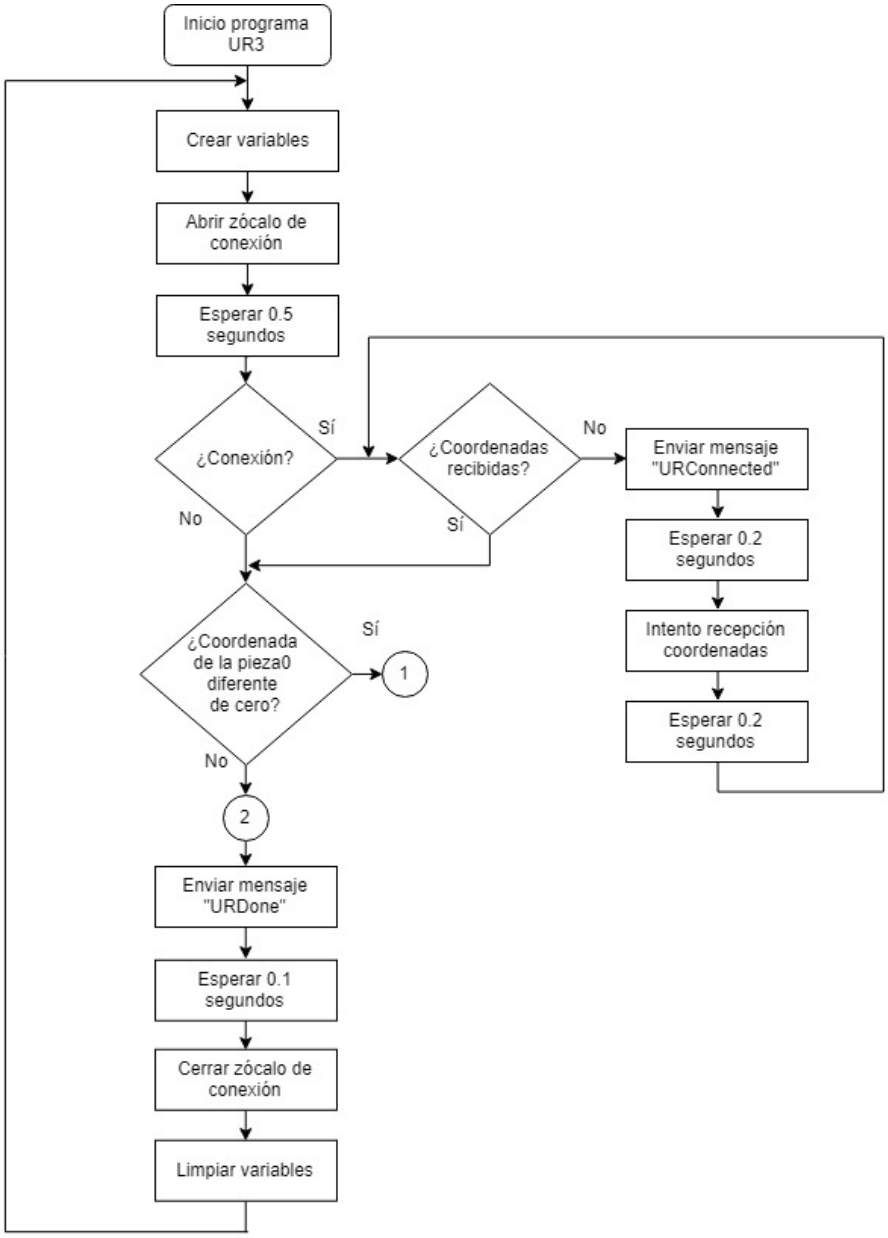


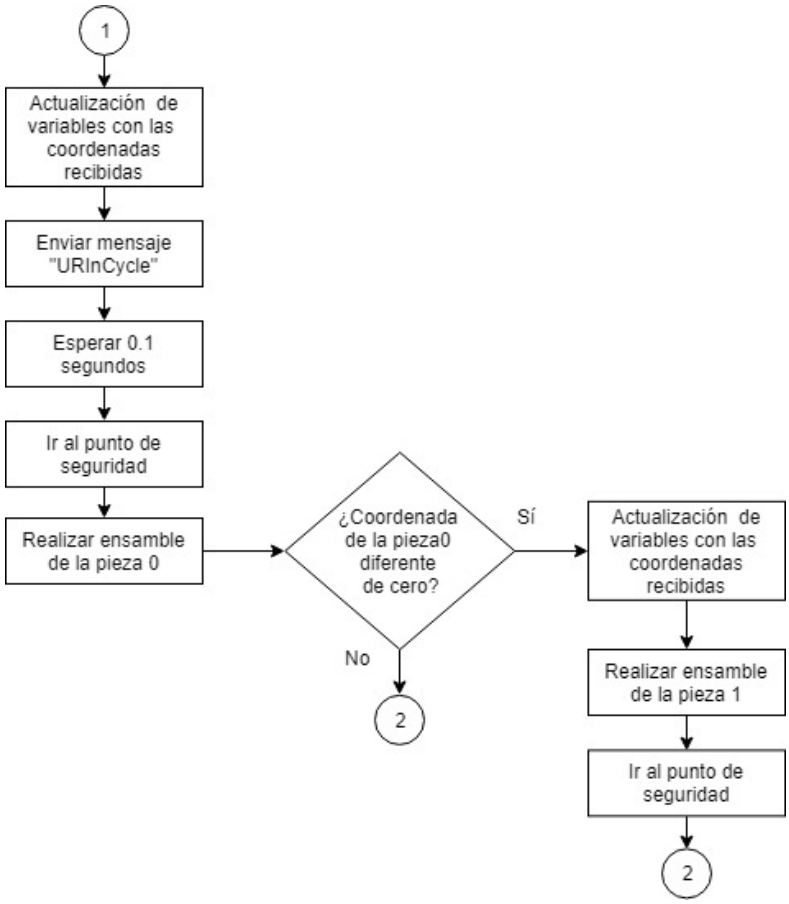

1
¿Cycle=true?
No
Esperar 1 seg
Incrementar trials
2
Sí
Ir al punto de seguridad general
¿Coordenada pieza 0 diferente de cero?
Sí
Colocar adhesivo en pieza 0
No
¿Coordenada pieza 1 diferente de cero?
No
Sí
Colocar adhesivo en pieza 1
Ir al punto de seguridad general
Enviar "Done."
Robot_send= "Done."
Esperar 0.25 seg


Clear_variables
Msj recibido:= "null"
Cycle:=False
Robot_send:="null"
Inicializar puntos de colocación de adhesivo
Trials:=0
Volver


Update_variables
Inicialización variables subrutina
¿Counter<5 y var_pos<longitud mensaje?
No
Actualizar puntos colocación de adhesivo partiendo de x0,y0,x1,y1
Volver
Sí
Extraer sección del msj según var_pos
Actualizar var_pos
Incrementar Counter
¿Counter?
1
x0:=sección extraída del msj
2
y0:=sección extraída del msj
3
x1:=sección extraída del msj
4
y1:=sección extraída del msj

ANEXO 11.3: Diagrama de flujo del programa del Robot UR3



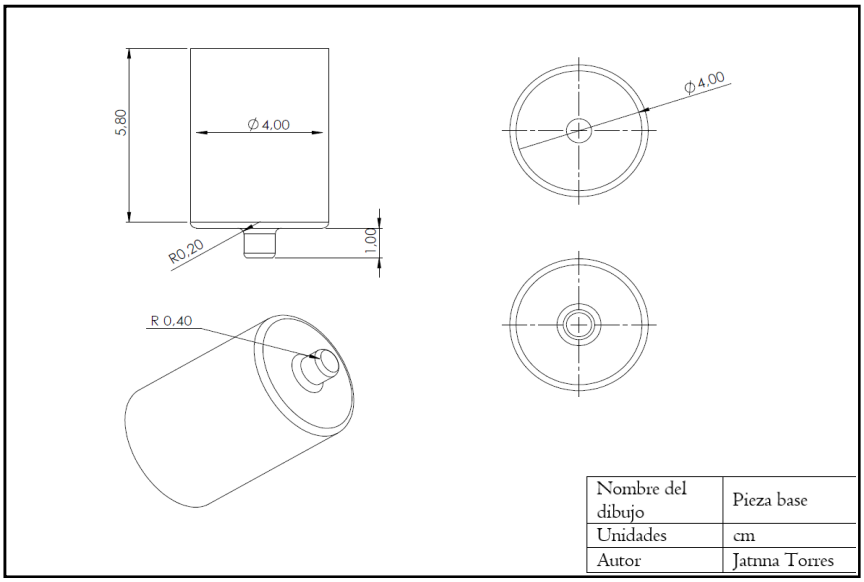

1
Actualización de variables con las coordenadas recibidas
Enviar mensaje "URInCycle"
Esperar 0.1 segundos
Ir al punto de seguridad
Realizar ensamble de la pieza 0
¿Coordenada de la pieza0 diferente de cero?
Sí
No
2
Actualización de variables con las coordenadas recibidas
Realizar ensamble de la pieza 1
Ir al punto de seguridad
2

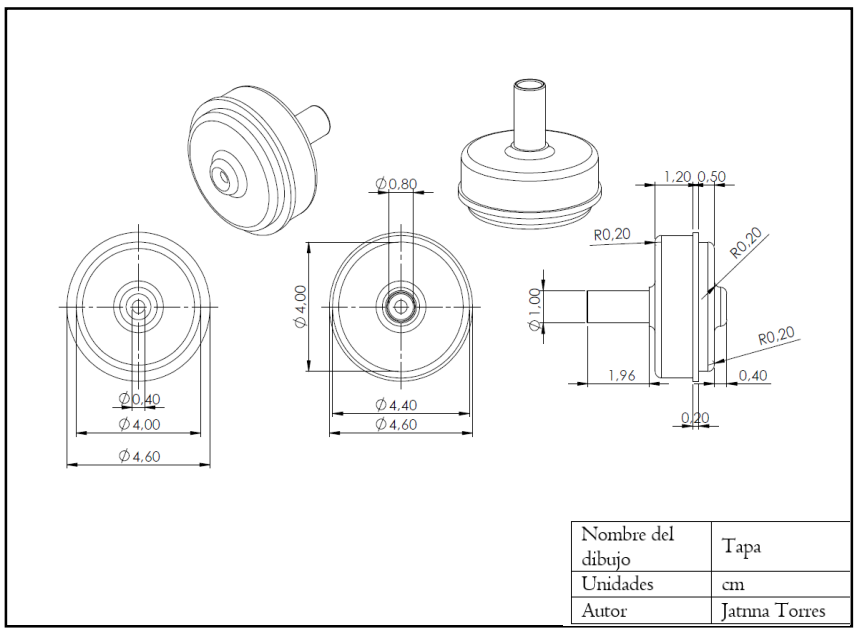
ANEXO 11.4: Script de limpieza de variables de Sherlock

```
Vars.Center_points_sent=0;
Vars.Center_points_sent_UR3=0;
Vars.Reception_result=0;
Vars.Center_points_parts="";
Vars.Center_points_parts_UR3="";

Vars.Point_part0.x=0;
Vars.Point_part0.y=0;
Vars.Point_part1.x=0;
Vars.Point_part1.y=0;

Vars.Pixels_Point_part0.x=0;
Vars.Pixels_Point_part0.y=0;
Vars.Pixels_Point_part1.x=0;
Vars.Pixels_Point_part1.y=0;


Vars.Point_part0_UR3.x=0;
Vars.Point_part0_UR3.y=0;
Vars.Point_part1_UR3.x=0;
Vars.Point_part1_UR3.y=0;

Vars.Centroids_rectA[0].x=0;
Vars.Centroids_rectA[0].y=0;
Vars.Centroids_rectA[1].x=0;
Vars.Centroids_rectA[1].y=0;

Vars.x0=0;
Vars.y0=0;
Vars.x1=0;
Vars.y1=0;
Vars.x0_UR3=0;
Vars.y0_UR3=0;
Vars.x1_UR3=0;
Vars.y1_UR3=0;

Vars.Cycle_done=0;
Vars.Communication_error=0;
Vars.connection_try_complete=0;
```

```
Vars.Robot_in_Cycle=0;
Vars.Robot_connection=0;
Vars.string_from_robot_in_cycle="";
Vars.String_from_UR3_in_cycle="";
Vars.string_from_robot_connection="";
Vars.String_from_UR3_Connection="";

Vars.counter_done=0;
Vars.Adding_value=1;
Vars.StablishConnection="Connection";
Vars.counter_connection=0;
Vars.centroid_counts=0;
Vars.counter_InCycle=0;
Vars.Cycle_done_UR3=0;
Vars.Reception_result_UR3=0;
Vars.Robot_connection_UR3=0;
Vars.counter_connection_UR3=0;
Vars.UR3_In_Cycle=0;
Vars.counter_InCycle_UR3=0;
Vars.counter_done_UR3=0;

Vars.connection_try_complete_UR=0;
```

ANEXO 11.5: Esquema de la cámara de goteo





ANEXO 11.6: MSDS del adhesivo

Cyclohexanone CAS No 108-94-1	MATERIAL SAFETY DATA SHEET SDS/MSDS

SECTION 1: Identification of the substance/mixture and of the company/undertaking

1.1 Product identifiers

Product name : Cyclohexanone

CAS-No. : 108-94-1

1.2 Relevant identified uses of the substance or mixture and uses advised against

Identified uses : Laboratory chemicals, Industrial & for professional use only.

1.3 Details of the supplier of the safety data sheet

Company : Central Drug House (P) Ltd
7/28 Vardaan House
New Delhi-10002
INDIA

Telephone : +91 11 49404040
Email : care@cdhfinechemical.com

1.4 Emergency telephone number

Emergency Phone # : +91 11 49404040 (9:00am - 6:00 pm) [Office hours]

SECTION 2: Hazards identification

2.1 Classification of the substance or mixture

Classification according to Regulation (EC) No 1272/2008
Flammable liquids (Category 3), H226
Acute toxicity, Oral (Category 4), H302
Acute toxicity, Inhalation (Category 4), H332
Acute toxicity, Dermal (Category 4), H312
Skin irritation (Category 2), H315
Serious eye damage (Category 1), H318

For the full text of the H-Statements mentioned in this Section, see Section 16.

Classification according to EU Directives 67/548/EEC or 1999/45/EC

		R10
Xn	Harmful	R20/21/22
Xi	Irritant	R38, R41

For the full text of the R-phrases mentioned in this Section, see Section 16.

2.2 Label elements

Labelling according Regulation (EC) No 1272/2008
Pictogram