



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



ESCUELA TÉCNICA
SUPERIOR INGENIEROS
INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

**DESARROLLO DE UN SISTEMA PARA LA
TELEOPERACIÓN DE UN ROBOT
MANIPULADOR CILÍNDRICO MEDIANTE
RECONOCIMIENTO DE GESTOS A TRAVÉS
DEL SENSOR KINECT.**

AUTORA: RODRÍGUEZ PÉREZ, M^ªDOLORES

TUTOR: HERRERO DURÁ, JUAN MANUEL

COTUTOR: SIMARRO FERNANDEZ, RAÚL

Curso Académico: 2014-15

CONTENIDO DEL PROYECTO

RESUMEN

DOCUMENTO Nº1: MEMORIA DESCRIPTIVA

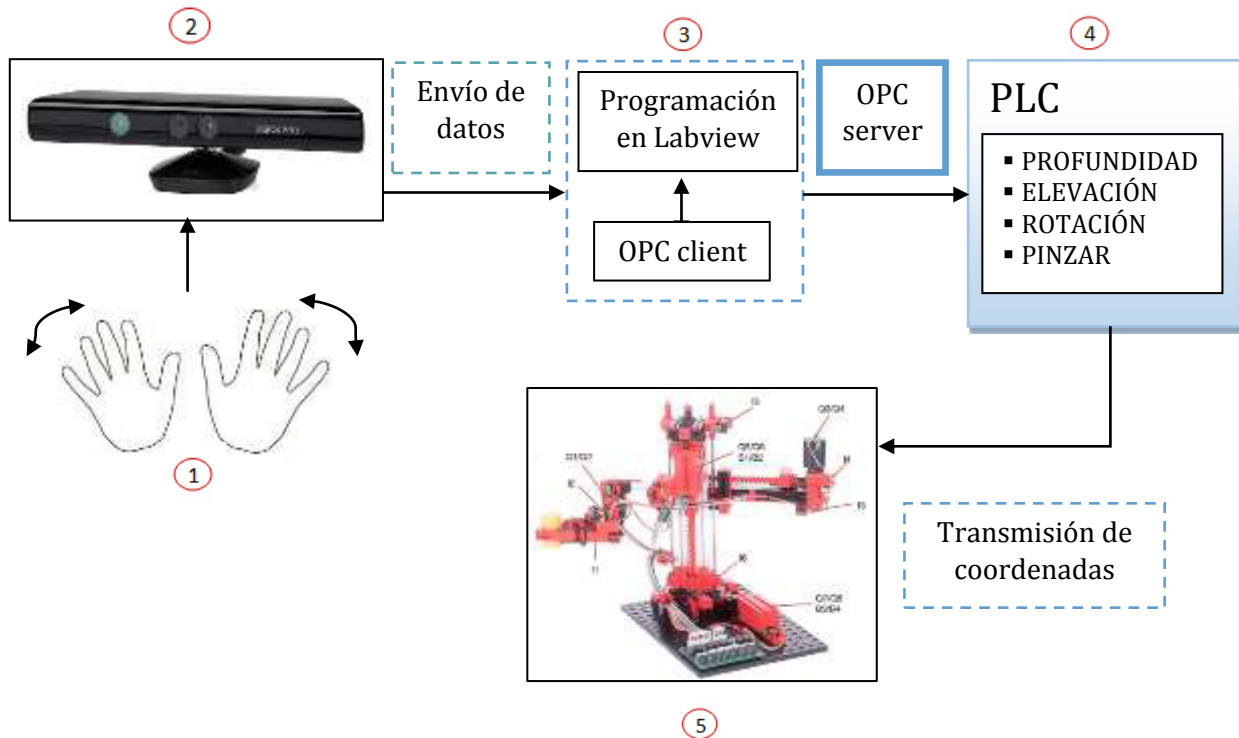
DOCUMENTO Nº2: PRESUPUESTO

DOCUMENTO Nº3: MANUAL DE PROGRAMACIÓN

DOCUMENTO Nº4: MANUAL DE USUARIO

RESUMEN

En este trabajo se busca diseñar e implementar un Software que permita dirigir los movimientos de un brazo robótico cilíndrico a distancia haciendo uso de Kinect® de Microsoft®, tal y como se muestra en la figura 1.



Desde kinect de Microsoft son leídas las posiciones de la mano derecha e izquierda del usuario. Estas posiciones son enviadas a LabVIEW (programa de lenguaje gráfico elegido para el tratado y filtrado de datos procedentes de kinect).

Se definen cuatro variables de interés: profundidad, elevación, rotación y pinzado, las cuales serán transferidas, mediante la comunicación OPC, hacia el Autómata TSX o controlador lógico programable (PLC) que es el que se encarga de controlar la posición del robot.

Para el mejor uso del programa, se ha creado una interfaz gráfica en LabVIEW que permite al usuario conocer en qué paso del proceso se encuentra. A su vez, se ha creado un modelo en 3D que permite ver en tiempo real cómo las variables enviadas en la lectura, son transferidas e interpretadas por el robot.

Palabras clave: Sensor Kinect, LabVIEW, servidor OPC, Autómata TSX o PLC.



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



ESCUELA TÉCNICA
SUPERIOR INGENIEROS
INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UN SISTEMA PARA LA TELEOPERACIÓN DE UN ROBOT MANIPULADOR CILÍNDRICO MEDIANTE RECONOCIMIENTO DE GESTOS A TRAVÉS DEL SENSOR KINECT.

DOCUMENTO Nº1: MEMORIA DESCRIPTIVA

AUTORA: RODRÍGUEZ PÉREZ, M^ªDOLORES

TUTOR: HERRERO DURÁ, JUAN MANUEL

COTUTOR: SIMARRO FERNANDEZ, RAÚL

Curso Académico: 2014-15

CONTENIDO

| | |
|---|-----------|
| INDICE DE FIGURAS | 2 |
| INDICE DE TABLAS | 2 |
| 1. OBJETO DEL PROYECTO | 4 |
| 2. MOTIVACIÓN | 5 |
| 3. DESCRIPCIÓN DEL PROCESO..... | 6 |
| 3.1. <i>BRAZO ROBOT</i> | <i>6</i> |
| 3.2. <i>AUTÓMATA.....</i> | <i>7</i> |
| 3.3. <i>KINECT DE MICROSOFT.....</i> | <i>8</i> |
| 4. DESCRIPCIÓN DE LA SOLUCIÓN | 9 |
| 4.1. <i>ESTRUCTURA DE LA SOLUCIÓN.....</i> | <i>9</i> |
| 4.2. <i>MANEJO DE KINECT.....</i> | <i>15</i> |
| 4.3. <i>APLICACIÓN DESARROLLADA EN LABVIEW</i> | <i>18</i> |
| 5. CONCLUSIÓN..... | 25 |
| 6. BIBLIOGRAFIA..... | 26 |
| I. <i>DESCARGAS.....</i> | <i>26</i> |
| II. <i>CONSULTAS.....</i> | <i>27</i> |
| 7. ANEXO DE ENTRADAS / SALIDAS DEL BRAZO ROBOT | 28 |

INDICE DE FIGURAS

| | |
|--|----|
| Figura 1. Esquema del proyecto..... | 4 |
| Figura 2. Brazo Robot (Imagen obtenida del catálogo de FischerTechnik)..... | 6 |
| Figura 3. Autómata TSX Premium. | 7 |
| Figura 4. Partes de Kinect (Imagen obtenida de http://blog.robotiq.com/) | 8 |
| Figura 5. Kinect joints (Imagen obtenida de decibel.ni.com)..... | 8 |
| Figura 6. Esquema principal del proceso. | 9 |
| Figura 7. Proyecto en LabVIEW. | 10 |
| Figura 8. Contador..... | 11 |
| Figura 9. Variables del servidor KepServeEX..... | 13 |
| Figura 10. Bucle principal. | 18 |
| Figura 11. Captura de coordenadas de las manos en las 3 dimensiones del espacio..... | 19 |
| Figura 12. Ilustración de los pasos a seguir en el panel frontal. | 20 |
| Figura 13. Controles numéricos para las coordenadas transformadas a valores porcentuales. | 20 |
| Figura 14. Transformación al lenguaje del brazo robot. | 21 |
| Figura 15. Creación del modelo en 3D de brazo robot | 22 |
| Figura 16. Posicionamiento de algunas de las geometrías | 23 |
| Figura 17. Modelo en 3D del brazo robot | 24 |

INDICE DE TABLAS

| | |
|---|----|
| Tabla 1. Direcciones de la memoria del autómata. | 12 |
| Tabla 2. Representación gráfica de los movimientos | 15 |
| Tabla 3. Direcciones de entrada de 3D robot TX Fischer Technik..... | 28 |
| Tabla 4. Direcciones de salida de 3D robot TX Fischer Technik. | 28 |

1. OBJETO DEL PROYECTO

El objeto de este proyecto es controlar un dispositivo mecánico mediante el reconocimiento de gestos, es decir, lograr que un usuario sin conocimientos técnicos pueda controlar un brazo robot mediante el movimiento de las articulaciones de su cuerpo.

Son especificaciones del cliente:

- 1) El uso del sensor Kinect como elemento de reconocimiento de usuario. Éste presenta una gran versatilidad de control y existen librerías para su manejo –con diversos lenguajes informáticos (LabVIEW, C#...).
- 2) Desarrollo de la aplicación de interacción con el usuario mediante LabVIEW. LabVIEW presenta una interfaz visual e intuitiva, dado que permite combinar la programación gráfica con hardware para simplificar y acelerar el desarrollo de aplicaciones de control y medida.
- 3) Llevar a cabo el control de brazo robot mediante un PLC modelo TSX Premium. El PLC ya dispone de los módulos de entradas/salidas adecuadas para interactuar con el robot.
- 4) El brazo robot con el que se ha de trabajar es 3-D-Robot TX FischerTechnik. Presenta cuatro grados de libertad (puede desplazarse sobre el eje vertical, eje horizontal, girar y pinzar).

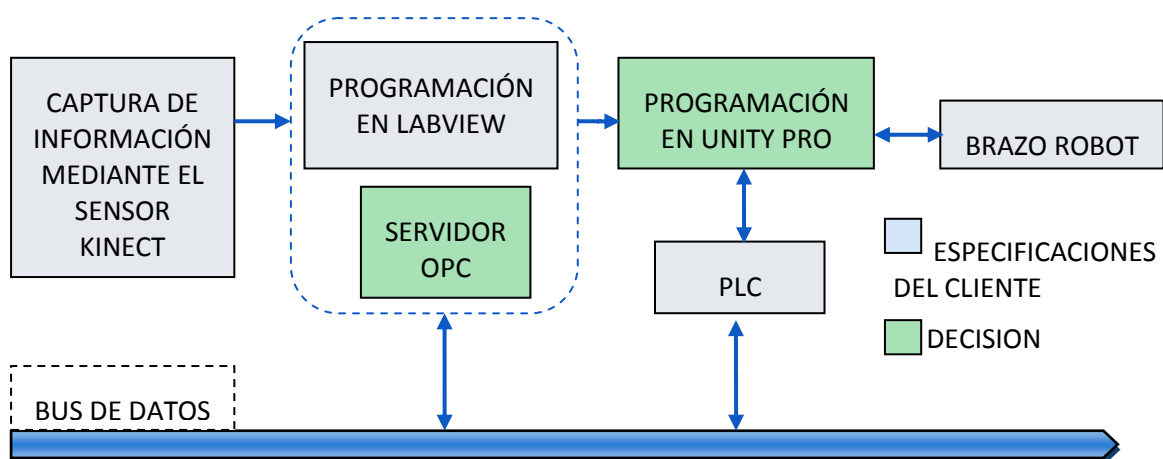


Figura 1. Esquema del proyecto.

2. MOTIVACIÓN

El alumno busca afianzar las bases de programación que ha obtenido durante el Grado, aplicándolas a un caso real con un nivel de abstracción superior a los estudiados hasta el momento. A su vez, potenciará sus conocimientos de robótica y de control, estudiados en asignaturas previas.

La motivación asociada al presente documento, se divide en dos vertientes principales:

- La primera, queda referida al interés personal del alumno por el aprendizaje de la temática que se desarrolla en el trabajo.
- La segunda, se sustenta en los múltiples beneficios de aplicación práctica y de utilidad real que pueden derivar del mismo.
 - ✓ Aplicaciones tecnológicas e industriales.
 - ✓ Aplicaciones para bioingeniería.
 - ✓ Aplicaciones de servicio especial.
 - ✓ Aplicaciones de control a distancia.

3. DESCRIPCIÓN DEL PROCESO

3.1. BRAZO ROBOT

El robot empleado para este trabajo es un 3-D-Robot TX FischerTechnik (a partir de ahora, se le llamará brazo robot).

Dispone de tres ejes capaces de realizar cuatro movimientos. Todos los movimientos disponen de dos sentidos, por lo que el brazo robot está conformado por cuatro motores de corriente continua cuya máxima frecuencia es de 1kHz. Además posee cuatro finales de carrera y dos sensores de pulso para la medición de la ruta del movimiento (marcados en la figura 2 y definidos en el punto 6 de este documento).

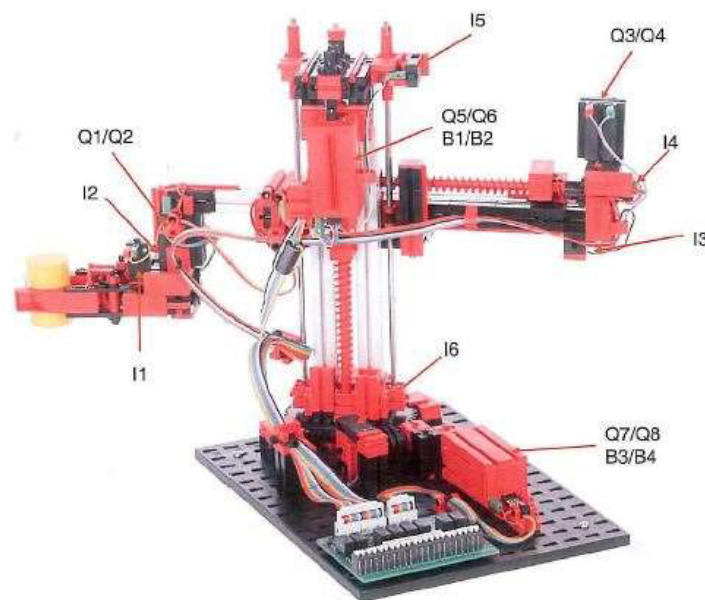


Figura 2. Brazo Robot (Imagen obtenida del catálogo de FischerTechnik).

Cada eje nombrado con anterioridad, está asociado a un movimiento en dos sentidos que corresponde con la rotación en sentido horario y antihorario, movimiento delantero y trasero en el eje horizontal y movimiento de subida y bajada en el eje vertical.

El cuarto movimiento corresponde con una pinza cuya función es coger o soltar objetos.

Los finales de carrera, de tipo booleano, están formados por pulsadores que se accionan cuando una coordenada del brazo robot ha llegado al final de su recorrido. El giro se encuentra limitado a 180°, mientras que el eje vertical a 150 mm y el horizontal a 90 mm.

Para llevar el brazo robot a una determinada posición, se ha empleado un contador de pulsos, el cual nos permite controlar la posición del robot en cada instante. Es importante

inicializar el brazo robot a la posición cero (todos los finales de carrera activados) antes de la transmisión de datos desde kinect, con objeto de evitar errores de posicionamiento debido a pérdidas de paso anteriores. Ésta será la posición de partida del brazo robot. Se profundizará sobre este tema en el “*Documento número 3. Manual de programación*”.

3.2. AUTÓMATA

El autómata, o controlador lógico programable (PLC) es el modelo TSX Premium. Posee una estructura de tipo modular en Entradas/ Salidas digitales DMY 28FK y dos nodos CAN (CAN es un protocolo de comunicaciones basado en la topología bus para la transmisión de información) de entradas y salidas digitales (los dos nodos) y de entradas y salidas analógicas (sólo el nodo 1 CAN), tal y como muestra la figura 3.

Para realizar la configuración del PLC se han seguido los pasos de la sesión 3 y 4 de las prácticas de la asignatura de Tecnología automática y se ha realizado con el programa Unity PRO.

La configuración de la conexión Ethernet entre el ordenador del laboratorio y el PLC es un paso muy importante, dado que si se realiza de manera errónea, los autómatas podrían llegar a perder su funcionalidad. Dentro del apartado “comunicaciones” (en la pantalla “explorador de proyectos” de Unity Pro) se selecciona “Redes”. Se hace click en el botón derecho del ratón y se elige la opción “nueva red”. Seguidamente se crea un elemento de red con el nombre de Ethernet. Se hace doble click sobre él y en la ventana emergente, dentro del apartado “configuración IP”, se elige la opción “Desde un servidor”. Si se sigue este procedimiento, no habrá problema a la hora de realizar la comunicación.

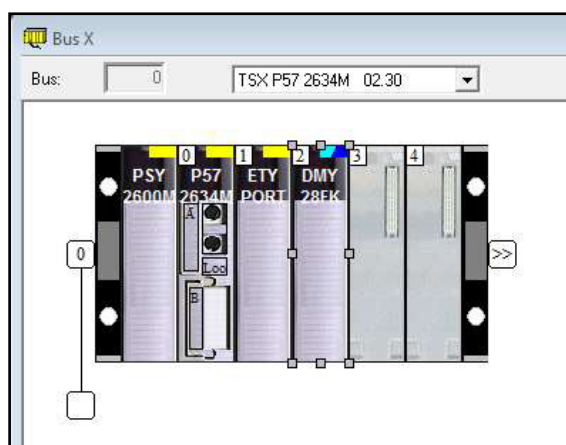


Figura 3. Autómata TSX Premium.

3.3. KINECT DE MICROSOFT

En sus inicios kinect nació como un controlador de juegos que permite a los usuarios interactuar con la consola sin necesidad de mandos o controladores físicos. Posteriormente Microsoft desarrolló la tecnología kinect para que pudiese ser utilizada en PC. Desde entonces se han desarrollado múltiples programas encaminados al control y manejo de dispositivos mecánicos o reconocimiento facial y de voz, gracias a que cuenta con una cámara RGB, un sensor de profundidad 3D (proyector de infrarrojos combinado con un sensor CMOS) y micrófonos (Figura 4).



Figura 4. Partes de Kinect (Imagen obtenida de <http://blog.robotiq.com/>)

Kinect viene equipado con un sistema de captura de movimientos (librerías SDK (Software Development Kit) de Windows) capaz de identificar el cuerpo y dibujar un esqueleto compuesto por 20 nodos (figura 5). En este trabajo se va a realizar un estudio centrado en las coordenadas de la mano derecha e izquierda que son los puntos de interés.

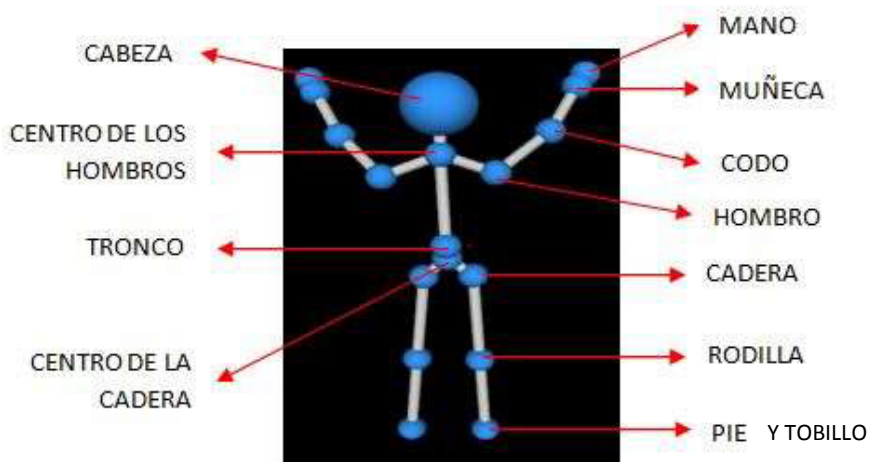


Figura 5. Kinect joints (Imagen obtenida de decibel.ni.com)

4. DESCRIPCIÓN DE LA SOLUCIÓN

4.1. ESTRUCTURA DE LA SOLUCIÓN

Este apartado será desarrollado en base a la figura mostrada en el resumen y también mostrado a continuación.

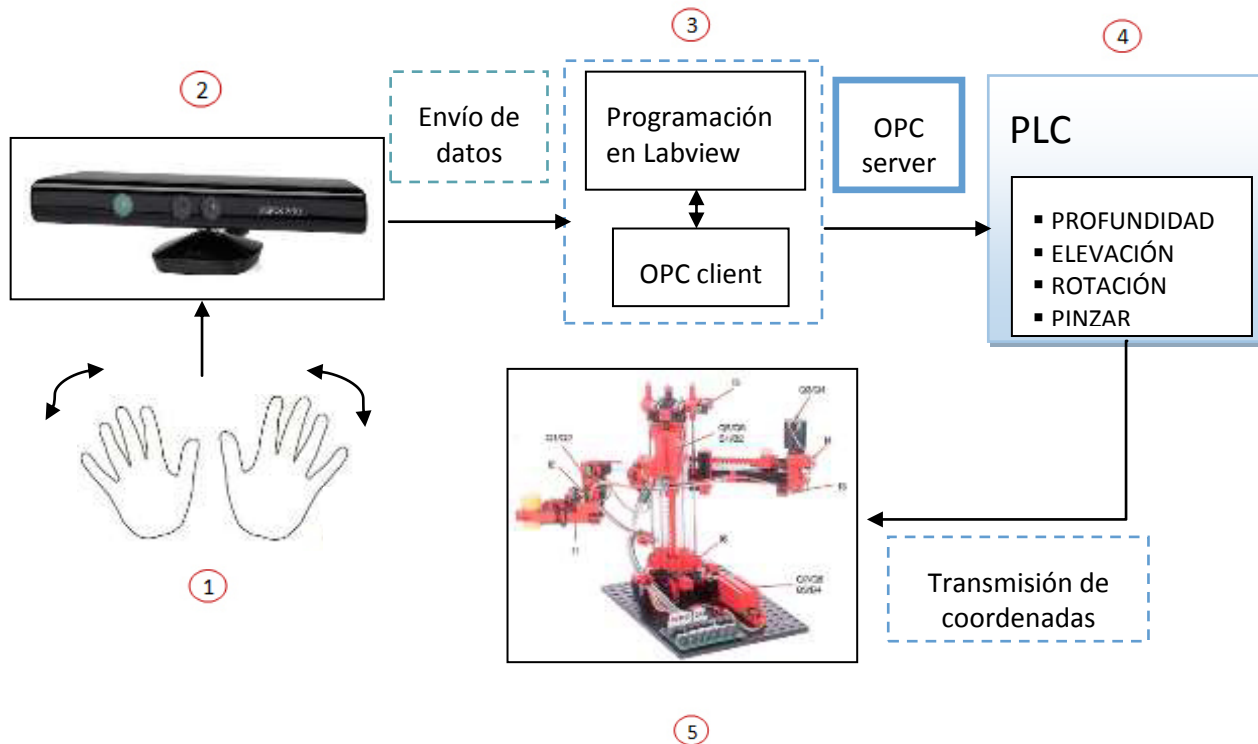


Figura 6. Esquema principal del proceso.

Desde Kinect de Microsoft, son leídas las posiciones de las dos manos del usuario.

Éstas han sido tomadas como referencia dado que con ellas se puede manejar el brazo robot de manera más precisa que con cualquier otra coordenada o nodo disponible.

La información captada desde Kinect es tratada con las funciones obtenidas de kinesthesia¹

Kinesthesia toolkit² proporciona un acceso inmediato a funciones de cámara RGB, cámara de profundidad y rastreo esquelético de Kinect.

¹ Funciones descargadas de http://www.ni.com/gate/gb/GB_EVALTLKTKINECTLV/US que fueron creadas por estudiantes de ingeniería mecánica de la Universidad de Leeds en 2012.

² Información obtenida de <http://sine.ni.com/nips/cds/view/p/lang/es/nid/210938>

Gracias a ellas, se ha podido manejar y filtrar las coordenadas de los nodos de interés en LabVIEW.

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) es una plataforma de programación gráfica e intuitiva, que emplea un lenguaje sencillo y de fácil implementación.

Los programas en LabVIEW son llamados instrumentos virtuales (VIs). Para agrupar varios VIs y permitir estos interactúen entre sí, se crean proyectos con extensión 'lvproj' con objeto de tener la información compactada.

El caso que el proyecto abarca, se ha creado un 'lvproj' titulado "PRINCIPAL" (Figura 7).

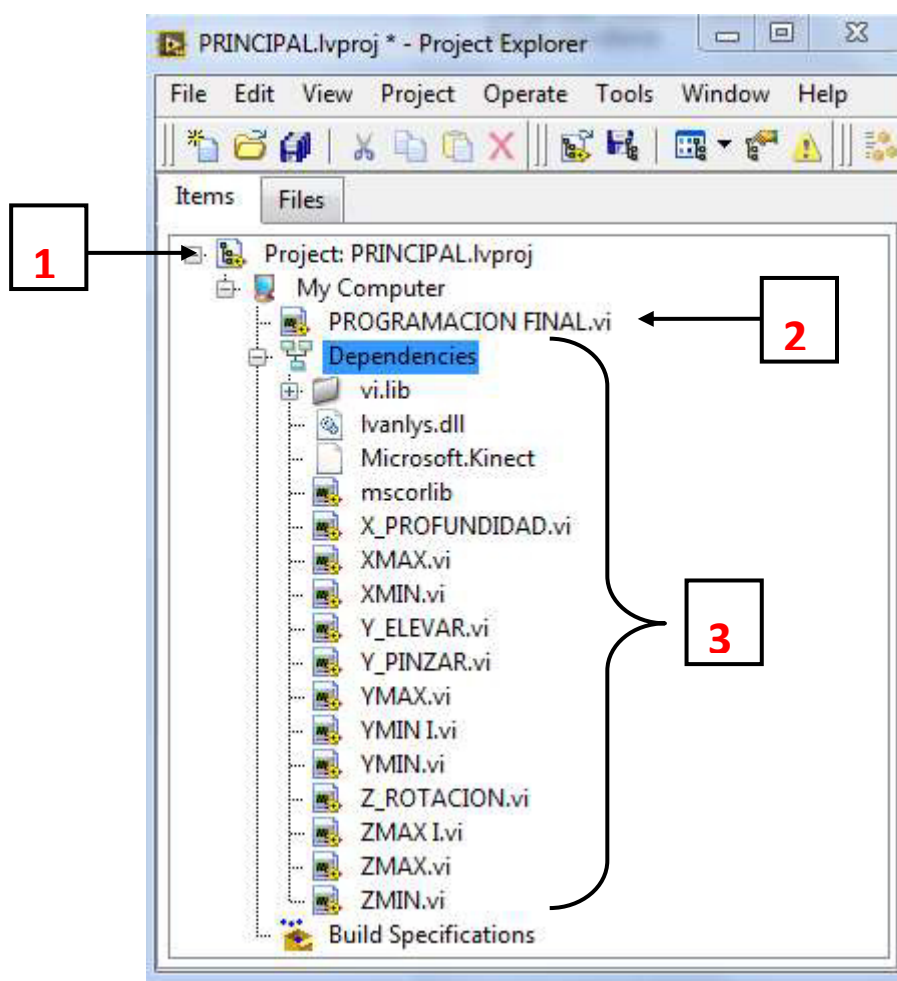


Figura 7. Proyecto en LabVIEW.

El número 1 hace referencia al nombre del proyecto, el número 2 indica el nombre del programa base sobre el que se ha realizado el trabajo y el número 3, hace referencia a todas las los VIs contenidos dentro de "Dependencies". Estos VIs forman parte de "PROGRAMACIÓN FINAL" e interactúan en su proceso, cada uno de los cuales, realiza una función concreta (esta parte será explicada en el apartado "4.3. Aplicación desarrollada en LabVIEW" localizado en este documento).

Con la interacción entre todos los Vis, expuestos en la figura anterior, se consigue realizar una serie de operaciones que trabajan con las coordenadas capturadas por Kinect para transformarlas al lenguaje del brazo robot.

Se debe de aclarar que cuando se habla de lenguaje del brazo robot, se refiere a que dicho dispositivo tiene sus desplazamientos limitados por sus características físicas y que sólo se pueden mover hasta una posición de pulsos determinada. Para cuantificar el valor de dichos pulsos, se ha creado un contador en Unity PRO el cual permite conocer la posición real del brazo robótico.

Una vez transformadas las variables deseadas a lenguaje del brazo robot se procede a la comunicación con el autómata.

LabVIEW envía, al servidor OPC, un total de cinco variables correspondientes al movimiento en el eje vertical, horizontal, giro, pinzas y botón “marcha”, el cual, al ser pulsado inicializa el brazo robot a su posición cero y comienza el proceso de lectura de información.

En paralelo, se ha utilizado Unity PRO para la realización de los diagramas de flujo (grafcet). Se han creado tres bloques funcionales:

- Primer bloque: contiene cuatro contadores empleados para determinar la posición de cada dimensión del brazo robot en cada momento de tiempo.

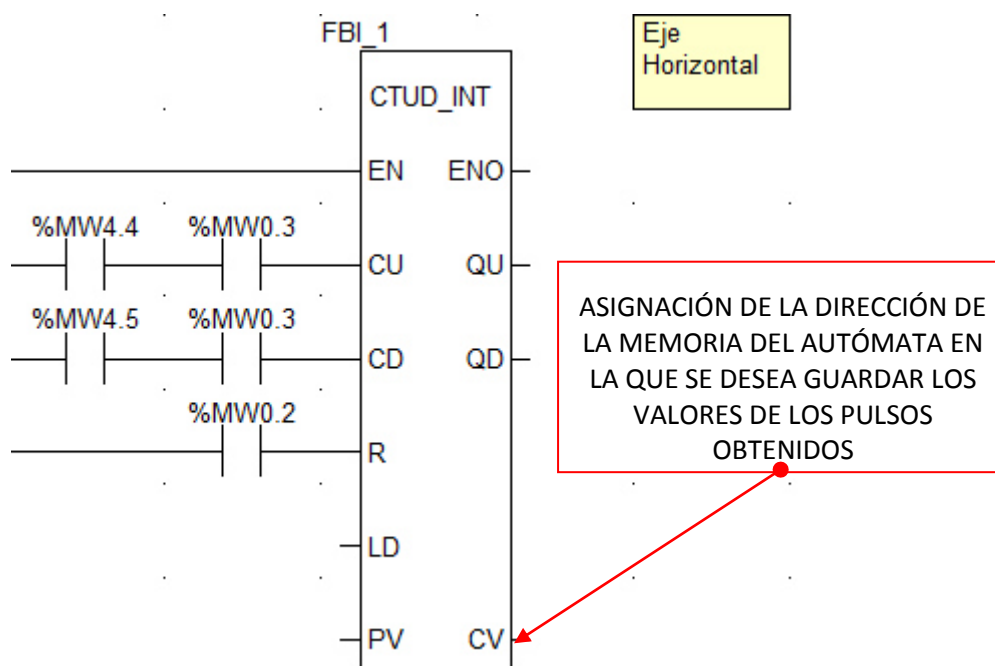


Figura 8. Contador

La figura 8 corresponde con el contador utilizado para el eje horizontal. A la salida “CV” se le asigna una dirección de la memoria del autómata, de forma que los pulsos leídos, se vayan almacenando en ella.

En este apartado no se detallará el significado de las demás entradas y salidas. Si se desea conocer más información sobre este contador, mirar el “documento n°3: Manual de programación”

- Segundo bloque: contiene el graficet de inicialización del brazo robot. Al ser pulsado el botón de marcha desde LabVIEW (como se ha comentado con anterioridad), este graficet es el encargado de llevar a la posición de inicio todas las coordenadas del brazo robot. Se ha configurado para que la posición inicial corresponda con la posición en la que todos los finales de carrera se encuentran accionados.
- Tercer bloque: contiene los graficets encargados de comparar la información recibida desde LabVIEW con la información de las posiciones reales del brazo robot y realizar los movimientos oportunos. Cada posición (tanto de las variables enviadas desde LabVIEW como las que dependen de los contadores de Unity PRO), está ligada a una dirección de la memoria del autómeta (Tabla 1).

| VARIABLES QUE DEPENDEN DE LABVIEW | | | |
|-------------------------------------|-----------|---------|-----------|
| NOMBRE | DIRECCIÓN | NOMBRE | DIRECCIÓN |
| REF_Y | %MW20 | TRANS_Y | %MW40.1 |
| REF_X | %MW21 | TRANS_X | %MW41.1 |
| REF_G | %MW22 | TRANS_G | %MW42.1 |
| REF_P | %MW23 | TRANS_P | %MW43.1 |
| MARCHA | %MW30.1 | | |
| VARIABLES QUE DEPENDEN DE UNITY PRO | | | |
| NOMBRE | DIRECCIÓN | NOMBRE | DIRECCIÓN |
| VERTICAL | %MW10 | GIRO | %MW12 |
| HORIZONTAL | %MW11 | PINZAR | %MW13 |

Tabla 1. Direcciones de la memoria del autómeta.

Para la comunicación se ha utilizado KepServerEx. En este servidor se han creado las variables definidas en la tabla 1.

En este servidor, se ha creado el canal “BRAZO ROBOT”.

En este canal, se ha creado un “device” titulado “VARIABLES”. Mediante “tags”, han sido definidas las variables de la tabla 1, como queda reflejado en la figura 9:

| Tag Name | Address | Data Type |
|----------|------------|-----------|
| TRANS_Y | %MW00040.1 | Boolean |
| TRANS_X | %MW00041.1 | Boolean |
| TRANS_P | %MW00043.1 | Boolean |
| TRANS_G | %MW00042.1 | Boolean |
| REF_Y | %MW00020 | Word |
| REF_X | %MW00021 | Word |
| REF_P | %MW00023 | Word |
| REF_G | %MW00022 | Word |
| MARCHA | %MW00030.1 | Boolean |
| CONT_Y | %MW00010 | Word |
| CONT_X | %MW00011 | Word |
| CONT_P | %MW00013 | Word |
| CONT_G | %MW00012 | Word |

Figura 9. Variables del servidor KepServeEX.

Para crear la conexión del OPC con LabVIEW, se hace click en el botón derecho del ratón sobre “My computer” (situado en la pantalla de “PRINCIPAL.lvproj”):

“My computer” → I/O Server → OPC Client → Continue → Browse → KepServer → OK.

A continuación se crea una librería en la cual se almacenarán todas las variables compartidas. Para ello, se vuelve a hacer click en el botón derecho del ratón sobre “My Computer”:

“My Computer” → New library → Create Bound Variables → Se selecciona del OPC las variables definidas en la tabla 1 y se pulsa “Add”.

Para utilizar dichas variables, sería suficiente con seleccionarlasy arrastrarlas hasta el lugar deseado. Por defecto, se encuentran en modo lectura. Si se desea, puede ser cambiada a modo escritura.

Una vez definidos todos los elementos que influyen en el proceso, se ejecutan ambos programas y se van siguiendo el procedimiento indicado en el panel frontal de LabVIEW.

Dicho panel, se ha organizado mediante un *'tag control'*, dividido en varias pestañas.

- La primera de ellas corresponde con el accionamiento del botón “Marcha” para comenzar el proceso.

- La segunda, se corresponde con la toma de posiciones máximas y mínimas que el usuario proporciona por cada movimiento que realiza.

Una vez personalizado el programa al usuario, se realizan una serie de operaciones para transformar las coordenadas en datos útiles para ser traspasado al robot.

De esta forma, se consigue tele-controlar el dispositivo mecánico.


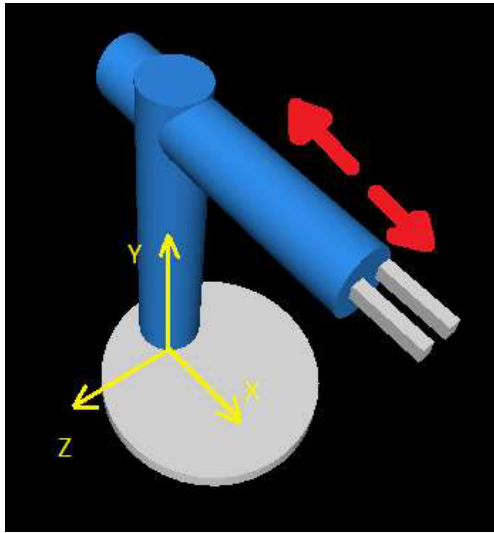
4.2. MANEJO DE KINECT


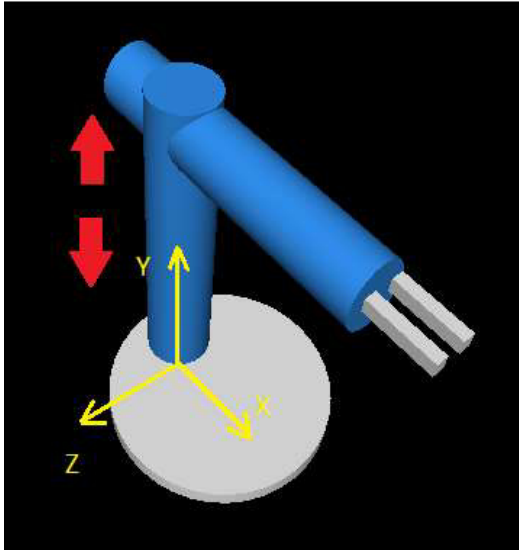

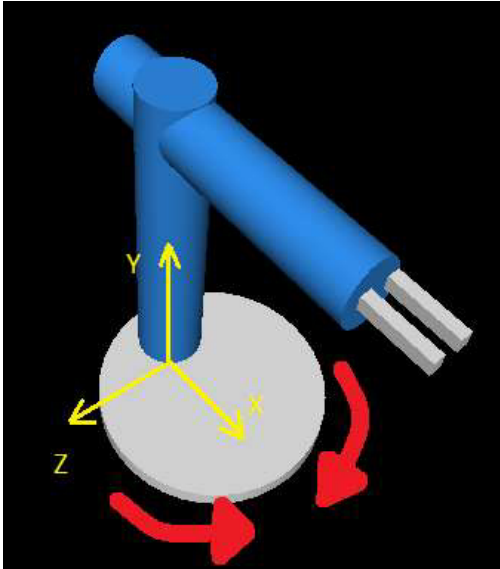
Como se ha explicado con anterioridad, se ha utilizado el dispositivo Kinect para la creación de una aplicación capaz de medir diferentes articulaciones del cuerpo, un total de 20 nodos, de entre los cuales sólo este trabajo se centrará en dos, correspondiente con las manos derecha e izquierda.

Gracias a su base monitorizada, encargada de ajustar la inclinación con un campo de visión en torno a 57 grados en el plano horizontal y 43 grados en el plano vertical, y al conjunto emisor-cámara infrarroja, situados en un eje alineado horizontalmente y separados a una distancia base de 75mm, se consigue enviar el movimiento que el usuario realiza en tiempo real al ordenador. Esta captura se hace en cuatro dimensiones, tres dimensiones espaciales (x , y , z) y la dimensión de color, que en este caso no tiene mayor importancia ya que esta dimensión no es objeto de este proyecto.

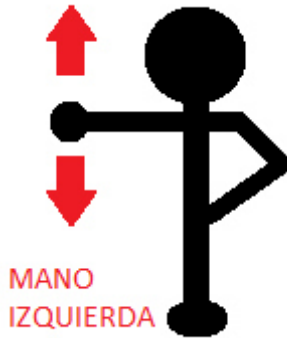
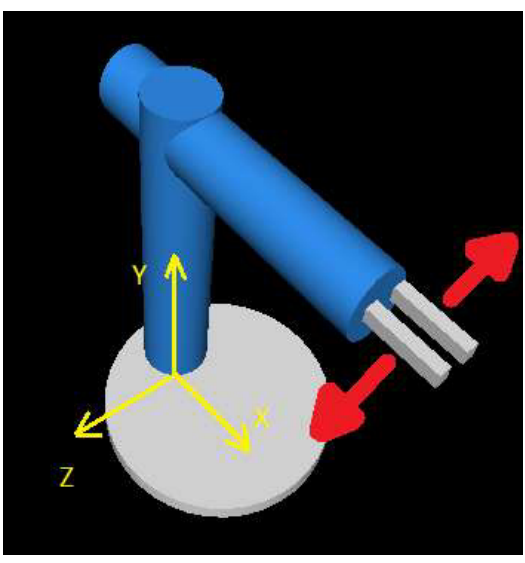
Los movimientos que el usuario debe realizar para gobernar el desplazamiento del brazo robot han sido elegidos para que las dos manos sean utilizadas. Si se levanta y baja la mano derecha, se controla el desplazamiento en el eje vertical del robot. Si se adelanta y se retrasa la mano derecha, el robot gira. Si se estira y se contrae la mano derecha en sentido horizontal, se consigue controlar el desplazamiento del robot en el eje horizontal. Y si se levanta y baja la mano izquierda, se controlan las pinzas (*Tabla 1*).

Tabla 2. Representación gráfica de los movimientos

| MOVIMIENTO DEL USUARIO | MOVIMIENTO REAL EQUIVALENTE DEL BRAZO |
|---|--|
| MOVIMIENTO HORIZONTAL | DESPLAZAMIENTO DEL BRAZO ROBOT |
|  |  |

| MOVIMIENTO VERTICAL | DESPLAZAMIENTO DEL BRAZO ROBOT |
|---|---|
|  <p>A stick figure with its right arm extended horizontally. Two red arrows, one pointing up and one pointing down, are positioned next to the hand. The text "MANO DERECHA" is written in red below the hand.</p> |  <p>A 3D rendering of a blue robot arm with two joints. A red arrow pointing up and a red arrow pointing down are shown next to the vertical joint. A yellow 3D coordinate system with axes labeled X, Y, and Z is shown at the base of the arm.</p> |
| GIRO | DESPLAZAMIENTO DEL BRAZO ROBOT |
|  <p>A stick figure with its right arm extended horizontally. Two red arrows, one pointing up and one pointing down, are positioned next to the hand. The text "MANO DERECHA" is written in red below the hand.</p> |  <p>A 3D rendering of a blue robot arm with two joints. A red arrow pointing up and a red arrow pointing down are shown next to the vertical joint. A yellow 3D coordinate system with axes labeled X, Y, and Z is shown at the base of the arm. Red curved arrows at the base indicate rotation around the Z-axis.</p> |

DESARROLLO DE UN SISTEMA PARA LA TELEOPERACIÓN DE UN ROBOT MANIPULADOR CILÍNDRICO MEDIANTE RECONOCIMIENTO DE GESTOS A TRAVÉS DEL SENSOR KINECT.

| PINZAR | DESPLAZAMIENTO DEL BRAZO ROBOT |
|--|---|
|  <p>MANO IZQUIERDA</p> <p>A stick figure with a black body and a black head. The left arm is extended horizontally to the left. Two red arrows are positioned vertically above and below the hand, pointing up and down respectively. The text 'MANO IZQUIERDA' is written in red below the figure.</p> |  <p>A 3D rendering of a blue robot arm with a grey base. A yellow coordinate system with X, Y, and Z axes is centered on the base. Two red arrows point away from the end effector in opposite directions, indicating movement.</p> |

4.3. APLICACIÓN DESARROLLADA EN LABVIEW

La pantalla de diagrama de bloques de este programa, ha sido dividida en tres sectores diferenciados:

- 1) **Primer sector:** corresponde con la lectura continuada de las coordenadas de las manos del usuario mediante el dispositivo Kinect. Para ello se utiliza un *bucle while* y las funciones obtenidas de kinesthesia (*Figura 10*).

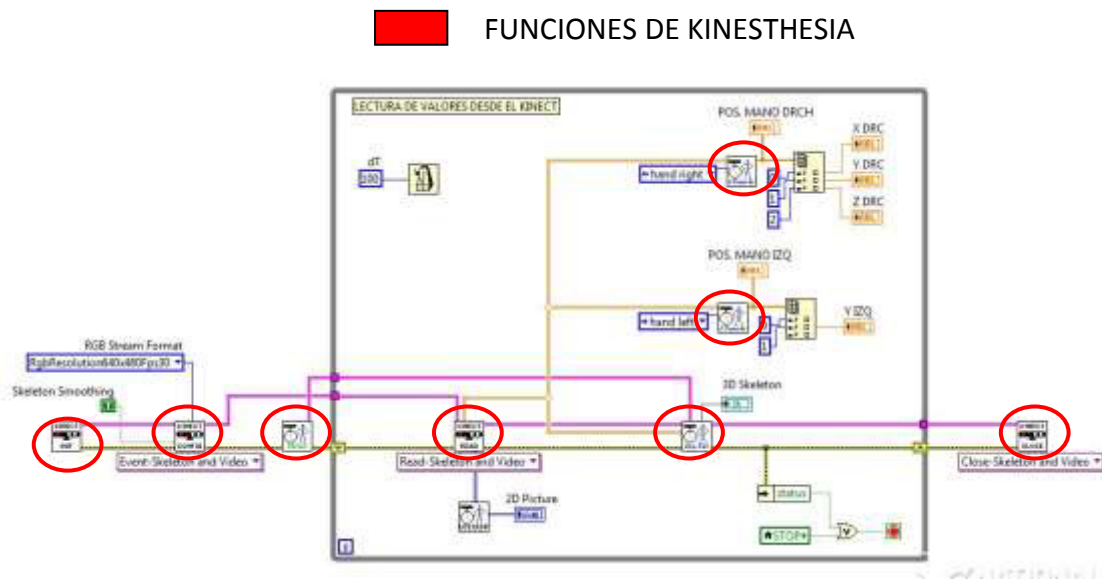


Figura 10. Bucle principal.

Éstas son empleadas para inicializar Kinect, configurar la parte que se desea leer y graficar un modelo en 3D del esqueleto del cuerpo del usuario.

La función '*Joint coordinates*' es la encargada de localizar los nodos de las manos y mediante un '*Index array*', volcar la información en variables (*Figura 11*).

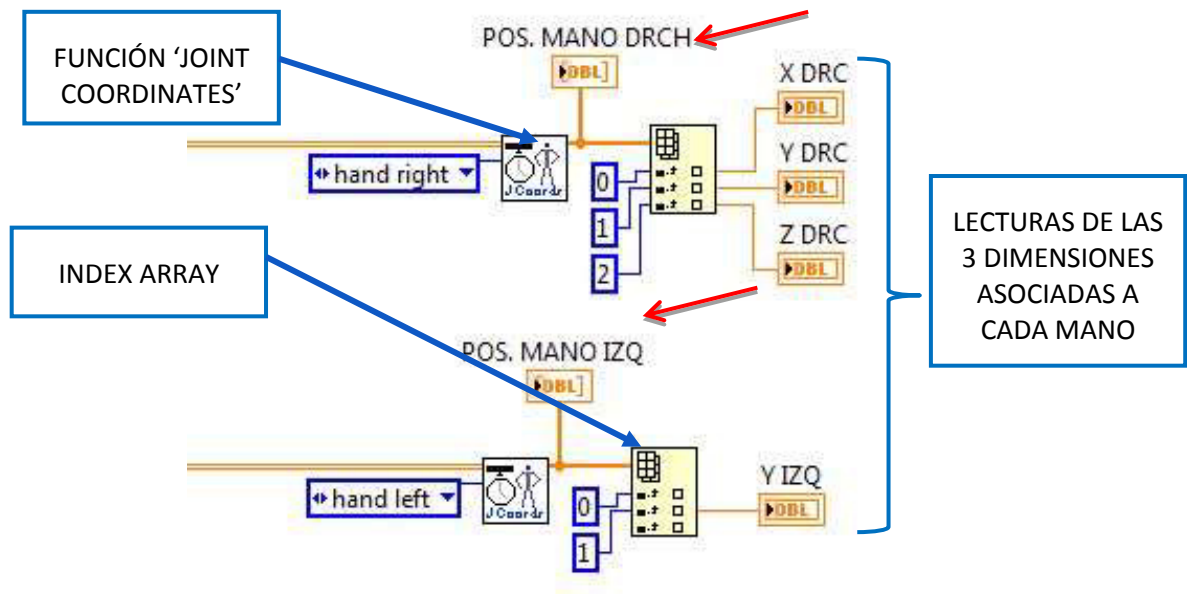


Figura 11. Captura de coordenadas de las manos en las 3 dimensiones del espacio.

- 2) Segundo sector: contiene el bucle principal encargado de procesar todos los datos para que la transmisión al brazo robot sea correcta.

Se ha utilizado una estructura de tipo secuencia que produce que las operaciones y funciones se accionen en serie.

Dentro de este sector se pueden diferenciar cuatro fases.

- La primera de ellas, se corresponde con la fase de toma de valores. Se van leyendo los valores máximos y mínimos para cada movimiento asociado a las manos derecha e izquierda, de modo que se crea un rango de valores personalizados para cada usuario.

Una vez que dichos máximos y mínimos han sido leídos, se enciende un LED y suena un 'beep' para indicar que la operación ha sido realizada con éxito (Figura 12).



Figura 12. Ilustración de los pasos a seguir en el panel frontal.

En la parte izquierda de la figura, se muestra un dibujo explicativo de los movimientos que se debe de realizar en cada paso. Adicionalmente, en la pantalla 'OPERACIÓN' se van mostrando los pasos a seguir en la toma de datos.

- La segunda fase, se corresponde con la normalización de los valores adquiridos en la fase anterior, a valores que pertenecen al rango de 0 a 100%.

Tras aplicar las operaciones adecuadas (apartado detallado en el "*documento n°3: Manual de programación*"), se muestran por pantalla dichos valores representados mediante controles numéricos de tipo tanque o 'knob' (Figura 13).

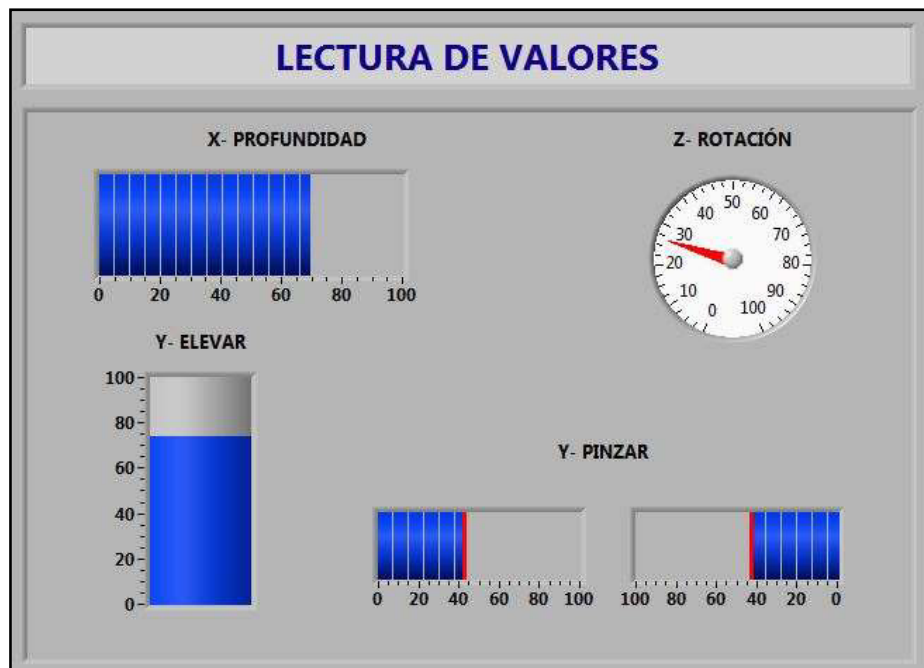


Figura 13. Controles numéricos para las coordenadas transformadas a valores porcentuales.

- En la tercera fase se utilizan las operaciones necesarias para transformar estos valores porcentuales a lenguaje del brazo robot (pulsos).
- La cuarta y última fase coloca un filtro con objeto de reducir las posibles oscilaciones y ruidos asociados a la transmisión de movimientos hacia el brazo robot.
Se ha empleado un “Median Filter”, cuya función es tomar los últimos 10 valores y hacer una media de ellos (Figura 14)

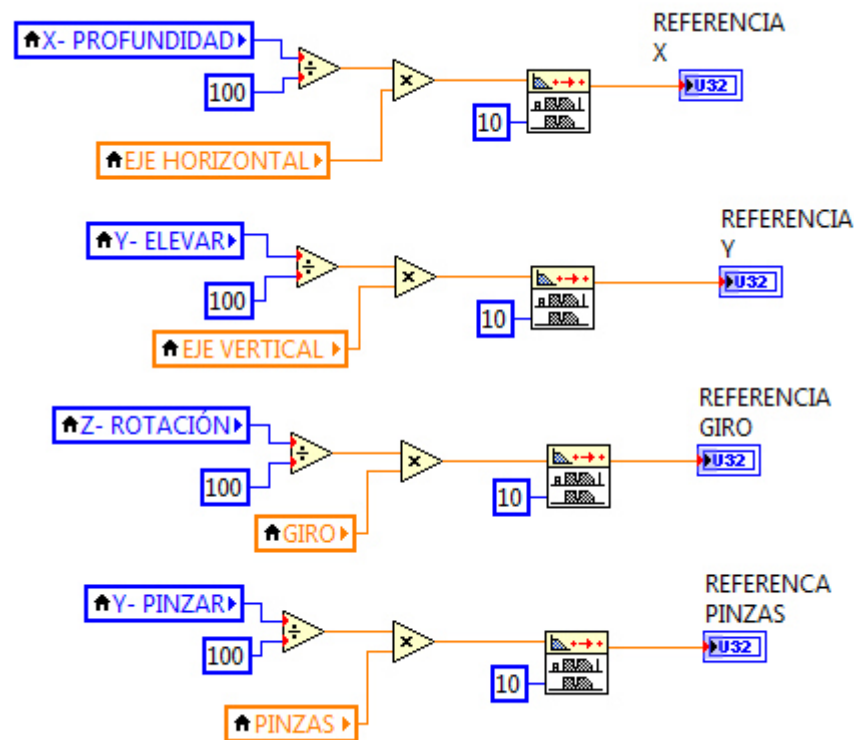


Figura 14. Transformación al lenguaje del brazo robot.

- 3) Tercer sector: Se trata de un bucle while encargado de formar una figura en 3D del brazo robot y desplazarse conforme a los movimientos que recibe desde el servidor de comunicación OPC client. Estos movimientos se corresponden con los desplazamientos reales del brazo robot.

Para formar una imagen en 3D, se elige un plano base sobre el cual, se van dibujando figuras primitivas referidas al primer elemento dibujado, esto es, componer escenarios a partir de objetos de primitivas (Figura 15).

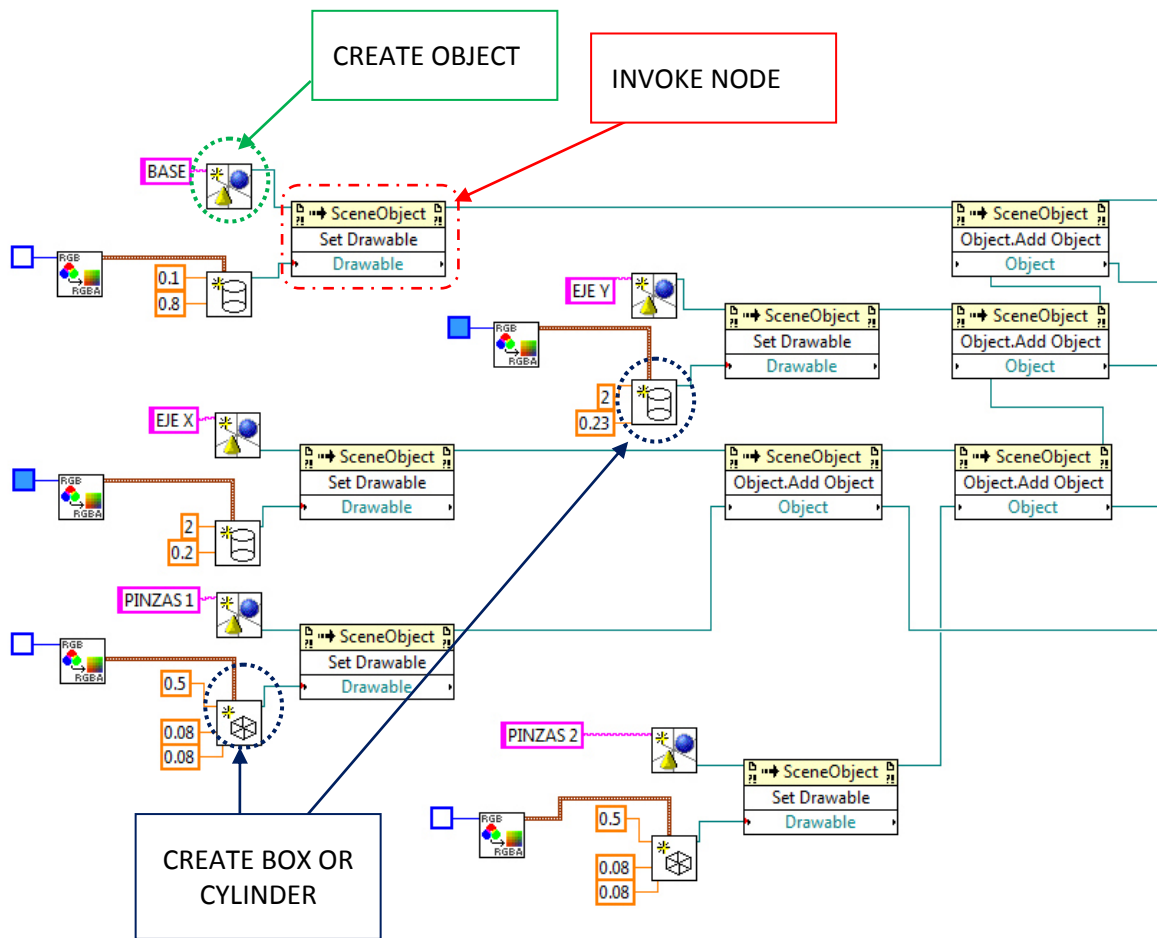


Figura 15. Creación del modelo en 3D de brazo robot

'Create object' es la función empleada para la creación de un nuevo objeto 3D. Se utiliza la función 'create box or cylinder' para formar las geometrías y el control de color para modificar el color (los colores elegidos para el brazo robot han sido gris y azul).

'Invoke node' permite realizar una acción sobre una referencia. Se han utilizado dos tipos de acciones: 'Add object' para añadir objetos a una referencia y 'Set Drawable' para dibujar objetos sobre una referencia.

Para posicionar las piezas ha sido necesario el uso de funciones de translación y rotación que tienen su origen en su centro geométrico. Estas operaciones son realizadas de forma secuencial y el orden en la que son aplicadas es muy importante (Figura 16).

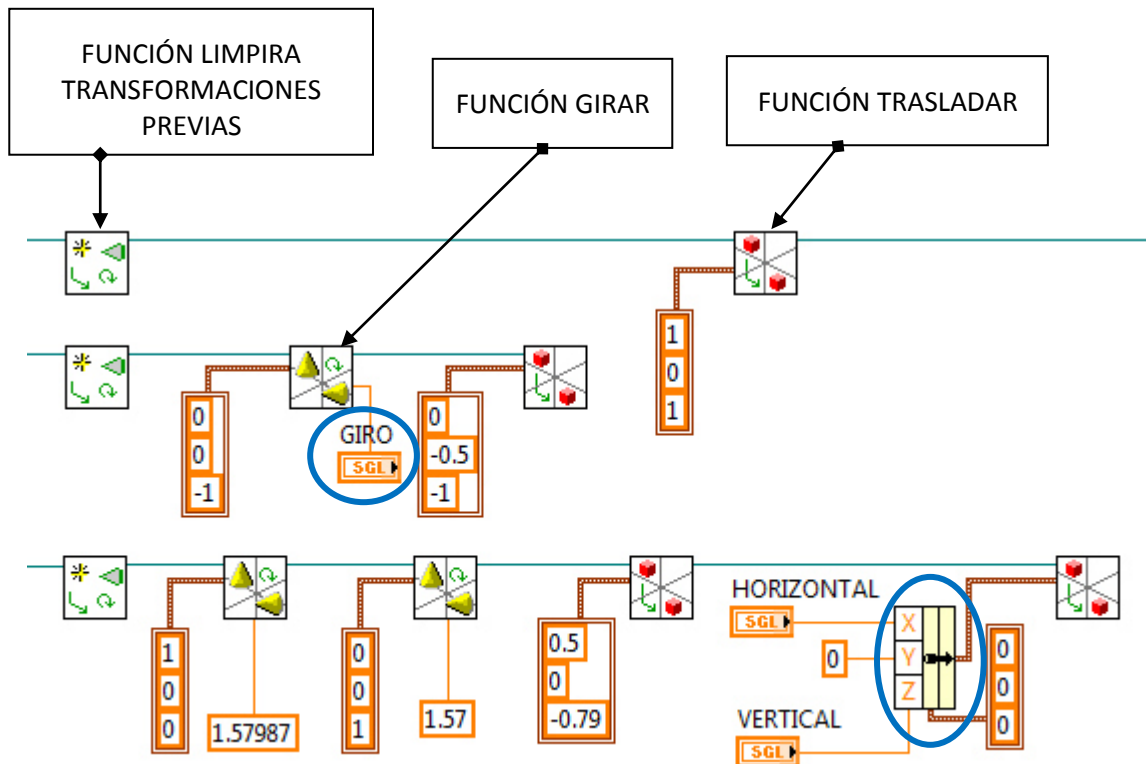


Figura 16. Posicionamiento de algunas de las geometrías

Los parámetros marcados con un círculo azul, son los parámetros que deben de modificarse para realizar el modelo en 3D (mostrado en la *Figura 17*).

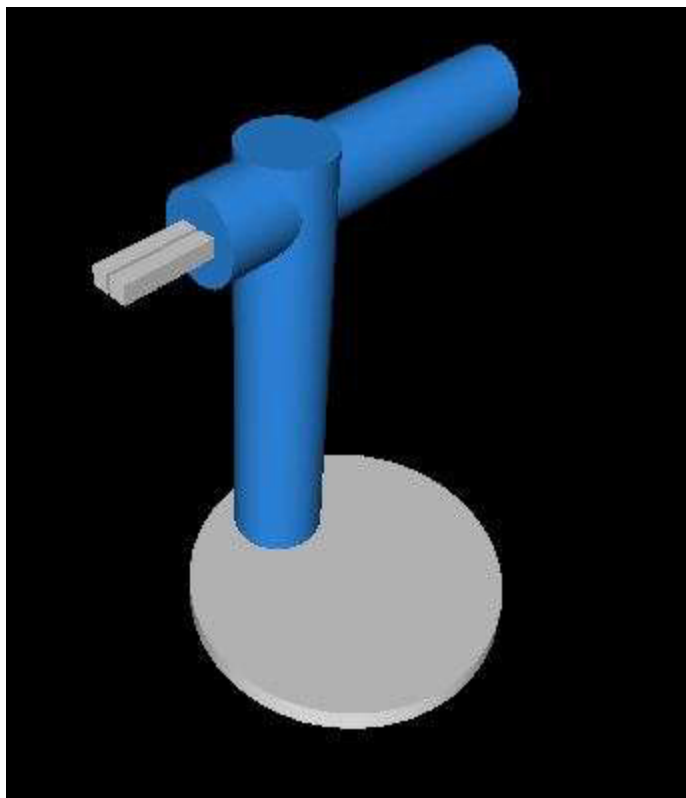


Figura 17. Modelo en 3D del brazo robot

5. CONCLUSIÓN

Durante el transcurso del trabajo y una vez finalizado el mismo, se han extraído las siguientes conclusiones:

1. Es importante limitar los movimientos físicos de los dispositivos mecánicos desde los sistemas de control para evitar daños o averías.
2. La transmisión correcta de la información de unos sistemas a otros, así como el correcto tratado de la misma, es crucial para el buen funcionamiento del sistema completo.
3. Pese a la complejidad de gestión de datos interna del programa de control, es importante simplificar el funcionamiento de cara a la interfaz gráfica del usuario. Para ello, ha sido necesario recurrir a la creación de un proyecto que agrupa todos los sub-Vis involucrados en la programación principal.
4. Como se ha observado, LabVIEW es una herramienta de gran envergadura que presenta, a su vez, una curva de aprendizaje muy pronunciada, al permitir la implementación del control de un dispositivo mecánico complejo partiendo de un nivel básico y en un corto período de tiempo.

Además, la realización del presente proyecto ha permitido al alumno afianzar y potenciar conocimientos previos:

- La implementación de Graficets, estudiados en las asignaturas de SAU y TAU, por medio del programa Unity Pro.
- Utilización de conocimientos de programación básica y LabVIEW para la gestión de un caso complejo, realizando a su vez una conexión con un dispositivo externo, Kinect.
- Matemática aplicada (traducción de variables, operaciones lógicas...).

Y también introducirse en otras áreas tecnológicas que le eran previamente desconocidas:

- Teleoperación y gestión de las variables medidas por Kinect.
- Comunicación Ethernet entre las diversas partes del sistema por medio del servidor OPC.
- Uso de Autómatas para el control del brazo robot.
- Modelado 3D mediante las herramientas de diseño gráfico que ofrece LabVIEW para representar el movimiento mecánico en tiempo real.

En definitiva, se puede afirmar que se ha cumplido con las especificaciones solicitadas: se ha logrado la teleoperación de un dispositivo mecánico gracias al sensor Kinect y se ha creado una interfaz gráfica en LabVIEW que permite al usuario controlar y manejar la información de forma sencilla y eficiente.

6. BIBLIOGRAFIA

I. DESCARGAS

1. Kinect for Windows SDK v1.7

MICROSOFT DOWNLOAD CENTRE. Kinect for Windows SDK v1. <<http://www.microsoft.com/en-us/download/confirmation.aspx?id=36996>> .

2. Kinect for Windows v1.7 Developer Toolkit

MICROSOFT DOWNLOAD CENTRE. Kinect for Windows v1.7 Developer Toolkit. <<http://www.microsoft.com/en-us/download/confirmation.aspx?id=36998>>.

3. Labview 2014

NATIONAL INSTRUMENTS. Descargue Labview. <<http://www.ni.com/download-labview/esa/>>

Labview es el programa elegido para realizar el control automático con la kinect.

4. Labview toolkit for Kinect

NATIONAL INSTRUMENTS. Kinesthesia LabVIEW Toolkit for Microsoft Kinect. <http://www.ni.com/gate/gb/GB_EVALTLTKKINECTLV/US>

II. CONSULTAS

1. ¿Qué es LabVIEW?

NATIONAL INSTRUMENTS. What is Labview. <<http://www.ni.com/newsletter/51141/en/>>

2. Brazo robot

FISCHERTECHNIK. Industry *Training model*.

<http://www.fischertechnik.de/en/desktopdefault.aspx/tabid-24/41_read-144/usetemplate-2_column_pano/>

3. Autómata

UNIVERSIDAD DE OVIEDO. Ingeniería de Sistemas y Automática. *Autómata TSX-Micro-PL7*.

<<http://www.isa.uniovi.es/docencia/iea/equipos/TSX-Micro-PL7V1-2.pdf>>

DIRECT INDUSTRY. Schneider- Electric. Autómata programable Modicon Premium TSX 57.

<<http://www.isa.uniovi.es/docencia/iea/equipos/TSX-Micro-PL7V1-2.pdf>>

4. Learning with Labview.

BISHOP, R.H. (1999). *Learning with Labview*. National Instruments.

5. Labview for everyone.

TRAVIS, J. (2001). *Labview for everyone*. National Instruments.

7. ANEXO DE ENTRADAS / SALIDAS DEL BRAZO ROBOT

| ENTRADA | DESCRIPCIÓN | DIRECCIÓN TSX |
|---------|--|---------------|
| I1 | Final de carrera referencia pinza | %MW0.0 |
| I2 | Contador de pulsos de la pinza | %MW0.1 |
| I3 | Final de carrera referencia brazo agarre | %MW0.2 |
| I4 | Contador de pulsos del brazo de agarre | %MW0.3 |
| I5 | Final de carrera referencia vertical | %MW0.4 |
| I6 | Final de carrera referencia de giro | %MW0.5 |
| B1 | Sentido movimiento vertical | %MW0.6 |
| B2 | Pulsos encoder movimiento vertical | %MW0.7 |
| B3 | Sentido movimiento de giro | %MW3.0 |
| B4 | Pulsos encoder movimiento giratorio | %MW3.1 |

Tabla 3. Direcciones de entrada de 3D robot TX Fischer Technik.

| SALIDA | DESCRIPCIÓN | DIRECCIÓN TSX |
|--------|--|---------------|
| Q1 | Motor de apertura de la pinza | %MW4.2 |
| Q2 | Motor de cierre de la pinza | %MW4.3 |
| Q3 | Motor del brazo de la pinza hacia adelante | %MW4.4 |
| Q4 | Motor del brazo de la pinza hacia atrás | %MW4.5 |
| Q5 | Motor movimiento vertical hacia abajo | %MW4.6 |
| Q6 | Motor movimiento vertical hacia arriba | %MW4.7 |
| Q7 | Motor movimiento giratorio horario | %MW6.2 |
| Q8 | Motor movimiento giratorio antihorario | %MW6.3 |

Tabla 4. Direcciones de salida de 3D robot TX Fischer Technik.



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



ESCUELA TÉCNICA
SUPERIOR INGENIEROS
INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

**DESARROLLO DE UN SISTEMA PARA LA
TELEOPERACIÓN DE UN ROBOT
MANIPULADOR CILÍNDRICO MEDIANTE
RECONOCIMIENTO DE GESTOS A TRAVÉS
DEL SENSOR KINECT.**

DOCUMENTO Nº2: PRESUPUESTO

AUTORA: RODRÍGUEZ PÉREZ, M^ªDOLORES

TUTOR: HERRERO DURÁ, JUAN MANUEL

COTUTOR: SIMARRO FERNANDEZ, RAÚL

Curso Académico: 2014-15

CONTENIDO

| | |
|---|----------|
| INDICE DE TABLAS | 2 |
| 1. INTRODUCCIÓN..... | 4 |
| 2. PRESUPUESTO..... | 4 |
| 2.1. <i>DEFINICIÓN DE LAS UNIDADES DE OBRA</i> | 4 |
| 2.2. <i>CUADRO DE PRECIOS.....</i> | 4 |
| 2.2.1. MANO DE OBRA | 4 |
| 2.2.2. MATERIALES | 5 |
| 2.2.3. PRECIOS UNITARIOS | 5 |
| 2.2.4. PRECIOS DESCOMPUESTOS | 6 |
| 3. PRESUPUESTO DE EJECUCIÓN MATERIALY POR CONTRATA | 8 |

INDICE DE TABLAS

| | |
|---|---|
| Tabla 1. Salario de los operarios..... | 4 |
| Tabla 2. Tabla del precio del software | 5 |
| Tabla 3. Tabla de precio del hardware..... | 5 |
| Tabla 4. Tabla de precios unitarios | 5 |
| Tabla 5. Tabla de precios descompuestos | 7 |

1. INTRODUCCIÓN

Como en todo proyecto en el ámbito de la ingeniería, es necesario cuantificar el valor económico del trabajo realizado, tanto a nivel de gasto en adquisición de materiales como en las horas empleadas por el personal involucrado.

En este caso, el proyecto está destinado al desarrollo informático de telecontrol de un brazo robot. Por tanto solo se tendrán en cuenta los conceptos relacionados al mismo, despreciando el gasto en autómatas

2. PRESUPUESTO

2.1. DEFINICIÓN DE LAS UNIDADES DE OBRA

Las unidades de obra que intervienen en este proyecto son:

- **UO1: Desarrollo de la aplicación de control del brazo robot.**
Incluye el estudio previo para el desarrollo de la aplicación, las horas de programación, desarrollo de las comunicaciones y pruebas de control. También se incluye la redacción de los documentos de manual de usuario y manual de programación.
- **UO2: Montaje y mantenimiento de los dispositivos.**
Incluye el montaje de los equipos que intervienen el proceso, así como de su inspección periódica para asegurar el correcto funcionamiento de éstos. Además se realizará un informe en el que se detallará el estado de los dispositivos tras la inspección.

2.2. CUADRO DE PRECIOS

2.2.1. MANO DE OBRA

Los recursos humanos que han intervenido en este proyecto son:

- Ingeniero Graduado en Ingeniería en Tecnologías Industriales, encargado de la realización del proyecto.
- Técnico de laboratorio, encargado del mantenimiento y del correcto funcionamiento de los recursos empleados.

El salario cobrado por hora queda reflejado en la siguiente tabla:

| OPERARIO | PRECIO (€/h) |
|--|--------------|
| Ingeniero Graduado en Ingeniería en Tecnologías Industriales | 25,00 |
| Técnico de laboratorio | 16,00 |

Tabla 1. Salario de los operarios

2.2.2. MATERIALES

Para el desarrollo de la aplicación, ha sido necesaria la utilización de algunos programas (Software) y equipos informáticos (Hardware).

Éstos quedan detallados en las tablas que se muestran a continuación:

| SOFTWARE | PRECIO LICENCIA (€/año)* | PRECIO A AMORTIZAR (€/hora)* |
|--------------|--------------------------|------------------------------|
| LabVIEW 2014 | 1429,00 | 0,82 |
| Unity Pro | 1500,00 | 0,85 |

Tabla 2. Tabla del precio del software

| HARDWARE | PRECIO LICENCIA (€/año)* | PRECIO A AMORTIZAR (€/hora)* |
|---------------------|--------------------------|------------------------------|
| Kinect de Microsoft | 200,00 | 0,114 |
| Ordeandor | 750,00 | 0,426 |

Tabla 3. Tabla de precio del hardware

*Para la determinación del precio por unidad de tiempo, se ha supuesto que el personal involucrado trabaja 220 días al año y realizan una jornada de 8 horas diarias.

2.2.3. PRECIOS UNITARIOS

En este apartado se muestra el precio unitario de cada unidad de obra:

| Nº ORDEN | DESCRIPCIÓN | MEDICIÓN | PRECIO | IMPORTE | |
|-----------------------------------|---|---|----------------|---------|------|
| 01 | DESARROLLO DE LA APLICACIÓN DE CONTROL DEL BRAZO ROBOT | | | | |
| 01.01 | Ud | DESARROLLO DE LA APLICACIÓN EN LABVIEW. | 1,000 | 2142,83 | 2143 |
| 01.02 | Ud | DESARROLLO DE LA APLICACIÓN EN UNITY PRO. | 1,00 | 804,05 | 804 |
| 01.03 | Ud | REDACCIÓN DE MANUALES. | 1,00 | 1296,73 | 1297 |
| Total de Unidad de Obra 01 | | | 4244,00 | | |
| 02 | MONTAJE Y MANTENIMIENTO DE LOS DISPOSITIVOS | | | | |
| 02.01 | Ud | MANTENIMIENTO. | 1,00 | 324,00 | 324 |
| 02.02 | Ud | MONTAJE Y PUESTA EN MARCHA DE LOS DISPOSITIVOS. | 1,00 | 362,04 | 362 |
| Total de Unidad de Obra 02 | | | 686,00 | | |

Tabla 4. Tabla de precios unitarios

DESARROLLO DE UN SISTEMA PARA LA TELEOPERACIÓN DE UN ROBOT MANIPULADOR CILÍNDRICO MEDIANTE RECONOCIMIENTO DE GESTOS A TRAVÉS DEL SENSOR KINECT.

2.2.4. PRECIOS DESCOMPUESTOS

| Nº ORDEN | Nº PRECIO | | DESCRIPCIÓN | RENDIMIENTO | PRECIO | IMPORTE |
|-----------------|---------------|----|--|-------------|---------|-----------------|
| 01 | | | DESARROLLO DE LA APLICACIÓN DE CONTROL DEL BRAZO ROBOT | | | |
| <i>01.01</i> | P01.01 | Ud | <i>DESARROLLO DE LA APLICACIÓN EN LABVIEW. Incluye programación y diseño.</i> | | | |
| | 001 | h | Ingeniero Graduado en Ingeniería en Tecnologías Industriales | 80,00 | 25,00 | 2000,00 |
| | 002 | h | Licencia LabVIEW 2014 | 80,00 | 0,82 | 65,60 |
| | 003 | h | Kinect de Microsoft | 10,00 | 0,114 | 1,14 |
| | 004 | h | Ordenador de sobremesa | 80,00 | 0,426 | 34,08 |
| | % | | Costes directos complementarios | 0,02 | 2100,82 | 42,02 |
| Costes directos | | | | | | 2142,83 |
| Coste total | | | | | | 2142,83€ |
| <i>01.02</i> | P01.02 | Ud | <i>DESARROLLO DE LA APLICACIÓN EN UNITY PRO. Incluye programación, diseño y comunicación con el autómeta</i> | | | |
| | 001 | h | Ingeniero Graduado en Ingeniería en Tecnologías Industriales | 30,00 | 25,00 | 750,00 |
| | 005 | h | Licencia Unity PRO | 30,00 | 0,85 | 25,50 |
| | 004 | h | Ordenador de sobremesa | 30,00 | 0,426 | 12,78 |
| | % | | Costes directos complementarios | 0,02 | 788,28 | 15,76 |
| Costes directos | | | | | | 804,05 |
| Coste total | | | | | | 804,05€ |
| <i>01.03</i> | P01.03 | Ud | <i>REDACCIÓN DE MANUALES. Incluye el manual de programación y el manual de usuario.</i> | | | |
| | 001 | h | Ingeniero Graduado en Ingeniería en Tecnologías Industriales | 50,00 | 25,00 | 1250,00 |
| | 004 | h | Ordenador de sobremesa | 50,00 | 0,426 | 21,30 |
| | % | | Costes directos complementarios | 0,02 | 1271,30 | 25,43 |
| Costes directos | | | | | | 1296,73 |
| Coste total | | | | | | 1296,73€ |

DESARROLLO DE UN SISTEMA PARA LA TELEOPERACIÓN DE UN ROBOT MANIPULADOR CILÍNDRICO MEDIANTE RECONOCIMIENTO DE GESTOS A TRAVÉS DEL SENSOR KINECT.

| 02 | | | MONTAJE Y MANTENIMIENTO DE LOS DISPOSITIVOS | | | |
|-----------------|---------------|----|---|----------------|--------|--------|
| 02.01 | P02.01 | Ud | <i>MANTENIMIENTO. Incluye la inspección periódica de los equipos, así como su mantenimiento y reparación en caso de avería.</i> | | | |
| | 006 | h | Técnico de laboratorio | 20,00 | 16,00 | 320,00 |
| | % | | Costes directos complementarios | 0,02 | 320,00 | 6,40 |
| Costes directos | | | | 324,00 | | |
| Coste total | | | | 324,00€ | | |
| 02.02 | P02.02 | Ud | <i>MONTAJE Y PUESTA EN MARCHA DE LOS DISPOSITIVOS. Incluye el montaje de los equipos y su calibración inicial.</i> | | | |
| | 001 | h | Ingeniero Graduado en Ingeniería en Tecnologías Industriales | 20,00 | 25,00 | 500,00 |
| | 006 | h | Técnico de laboratorio | 20,00 | 16,00 | 320,00 |
| | 002 | h | Licencia LabVIEW 2014 | 15,00 | 0,82 | 12,30 |
| | 003 | h | Kinect de Microsoft | 12,00 | 0,114 | 1,37 |
| | 005 | h | Licencia Unity PRO | 15,00 | 0,85 | 12,75 |
| | 004 | h | Ordenador de sobremesa | 20,00 | 0,426 | 8,52 |
| | % | | Costes directos complementarios | 0,02 | 354,94 | 7,10 |
| Costes directos | | | | 362,04 | | |
| Coste total | | | | 362,04€ | | |

Tabla 5. Tabla de precios descompuestos

3. PRESUPUESTO DE EJECUCIÓN MATERIAL Y POR CONTRATA

| RESUMEN | | | |
|---------------------------|--|--|---------|
| UNIDAD DE OBRA: 01 | DESARROLLO DE LA APLICACIÓN DE CONTROL DEL BRAZO ROBOT | | 4244,00 |
| UNIDAD DE OBRA: 02 | MONTAJE Y MANTENIMIENTO DE LOS DISPOSITIVOS | | 686,00 |
| | Suma Ejecución material | | 4930,00 |

Asciende el presupuesto de **Ejecución Material** a la expresada cantidad de EUROS: *CUATRO MIL NOVECIENTOS TREINTA*.

| | |
|---|---------|
| Total Presupuesto de Ejecución Material | 4930,00 |
| 15 % Gastos generales | 739,50 |
| 6 % Beneficio industrial | 295,80 |

| | |
|---|--------|
| Suma de Gastos Generales y Beneficio Industrial | 1035,3 |
|---|--------|

| | |
|--------------------------------|---------|
| Total Presupuesto de Inversión | 5965,30 |
|--------------------------------|---------|

| | |
|------------|---------|
| 21 % I.V.A | 1252,71 |
|------------|---------|

| | |
|--------------------------------------|------------------|
| Total Presupuesto Base de Licitación | 7218,013€ |
|--------------------------------------|------------------|

Asciende el presupuesto de **Base de licitación** a la expresada cantidad de EUROS: *SIETE MIL DOSCIENTOS DIECIOCHO CON CERO TRECE EUROS*.



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



ESCUELA TÉCNICA
SUPERIOR INGENIEROS
INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UN SISTEMA PARA LA TELEOPERACIÓN DE UN ROBOT MANIPULADOR CILÍNDRICO MEDIANTE RECONOCIMIENTO DE GESTOS A TRAVÉS DEL SENSOR KINECT.

DOCUMENTO Nº3: MANUAL DE PROGRAMACIÓN

AUTORA: RODRÍGUEZ PÉREZ, M^ªDOLORES

TUTOR: HERRERO DURÁ, JUAN MANUEL

COTUTOR: SIMARRO FERNANDEZ, RAÚL

Curso Académico: 2014-15

CONTENIDO

| | |
|--|-----------|
| INDICE DE FIGURAS | 2 |
| INDICE DE TABLAS | 2 |
| 1. INTRODUCCIÓN | 4 |
| 2. DESARROLLO DEL SOFTWARE EN LABVIEW PARA LA APLICACIÓN ROBÓTICAS | |
| 2.1. <i>INTRODUCCIÓN.</i> | 5 |
| 2.1.1. TIPOS DE DATOS | 5 |
| 2.1.2. Paleta de controles | 6 |
| 2.1.3. Paleta de funciones | 7 |
| 2.1.4. Variables locales y globales | 8 |
| I. Variables locales | 8 |
| II. Variables globales | 8 |
| 2.2. <i>ESTRUCTURA DEL PROGRAMA: 'BLOCK DIAGRAM'</i> | 9 |
| 2.2.1. Primer bloque | 9 |
| 2.2.2. Segundo bloque | 11 |
| I. Inicializar proceso y lectura de máximos/mínimos. | 11 |
| II. Convertir lecturas a valores de cero a cien por cien. | 14 |
| III. Transformación de coordenadas al lenguaje del brazo robot. Pulsos. | 18 |
| IV. Filtrado de la lectura obtenida de kinect. | 19 |
| 2.2.3. Tercer bloque | 20 |
| 2.3. <i>ESTRUCTURA DEL PANEL DE USUARIO: 'FRONT PANEL'</i> | 25 |
| 3. Programación en UNITY PRO | 26 |
| 3.1. <i>CONTADOR</i> | 27 |
| 3.2. <i>GRAFICET INICIAL</i> | 29 |
| 3.3. <i>GRAFICET DE MOVIMIENTOS</i> | 30 |
| 4. BIBLIOGRAFÍA | 32 |

INDICE DE FIGURAS

| | |
|---|----|
| Figura 1. Tipos de datos. Imagen obtenida de National Instruments..... | 6 |
| Figura 2. Paleta de funciones | 7 |
| Figura 3. Paleta de controles..... | 7 |
| Figura 4. Funciones de Kinesthesia | 9 |
| Figura 5. Additional VIs | 9 |
| Figura 6. Captura de coordenadas de las manos en las 3 dimensiones del espacio..... | 10 |
| Figura 7. Programa principal de lectura continua..... | 11 |
| Figura 8. Lectura de máximos y mínimos..... | 12 |
| Figura 9. Operaciones encargadas de encontrar máximos y mínimos. | 13 |
| Figura 10. Estructura para convertir lecturas de cero a cien por cien. | 14 |
| Figura 11. Normalización de valores a tanto por cien. | 15 |
| Figura 12. Recta que define la transformación para valores de mínimo positivo. | 16 |
| Figura 13. Ecuación de la recta trasladada a LabVIEW. | 16 |
| Figura 14. Recta que define la transformación para valores mínimos negativos. | 17 |
| Figura 15. Ecuación de la recta trasladada a LabVIEW. | 17 |
| Figura 16. Ecuación de la recta para transformación de coordenadas al lenguaje del brazo robot..... | 19 |
| Figura 17. Mejora de la lectura. Aplicación del filtro. | 20 |
| Figura 18. Estructura para crear una imagen en 3D a partir de figuras primitivas..... | 21 |
| Figura 19. Definición de un cilindro en 3D. | 21 |
| Figura 20. Modelado en 3D. | 22 |
| Figura 21. Ejes barra horizontal. | 23 |
| Figura 22. Modificación de los parámetros para que se mueva de acorde al brazo robot real. | 24 |
| Figura 23. Contador..... | 27 |
| Figura 24. Graficet de inicialización | 29 |
| Figura 25. Graficet de movimientos..... | 30 |

INDICE DE TABLAS

| | |
|---|----|
| Tabla 1. Descripción de los parámetros de entrada y salida de CTUD..... | 27 |
| Tabla 2. Variables en las direcciones de la memoria. | 31 |

1. INTRODUCCIÓN

En el presente documento se va a explicar con detalle la parte de programación tanto en LabVIEW como en Unity Pro. Para ello se especificarán algunas de las funciones más importantes utilizadas así como su implementación en el Software.

LabVIEW es programa elegido para la realización de la programación y transmisión de datos. La programación consta de tres partes cada una de las cuales posee una función diferenciada en la estructura global del programa. La primera parte corresponde con la lectura continua de datos de Kinect. La segunda corresponde a todo el proceso operatorio para el correcto funcionamiento del programa así como de confirmar el correcto envío de datos. Y la tercera y última parte, se corresponde con el modelado en 3D del brazo robot encargado de moverse según las referencias reales de este.

Unity Pro se utiliza como herramienta de creación de diagramas de flujo. Gracias a este programa, se podrá llevar al brazo robot a las posiciones deseadas.

2. DESARROLLO DEL SOFTWARE EN LABVIEW PARA LA APLICACIÓN ROBÓTICA

2.1. INTRODUCCIÓN.

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) es una plataforma de programación gráfica e intuitiva, que emplea un lenguaje sencillo y de fácil implementación. Fue creada por National Instruments (empresa fundada en 1976 que se dedica al desarrollo y venta de productos de software y hardware).

Los programas en LabVIEW son llamados instrumentos virtuales (VIs). Cada VI consta de un Panel frontal y un diagrama de bloques. El panel frontal es la interfaz con el usuario a partir de la cual, el usuario es capaz de manejar los datos. El diagrama de bloques es la parte donde se crea el programa y se define su funcionalidad.

En el panel frontal se pueden encontrar controles e indicadores genéricos de tipo booleano, numérico o cadena de caracteres y controladores e indicadores técnicos tales como gráficas, tablas o termómetros. Estos controles e indicadores tienen un elemento asignado en el diagrama de bloques, de modo que las entradas (controles) o salidas (indicadores) creadas por el usuario en el panel frontal, son tratadas en el diagrama de bloques para el correcto funcionamiento del programa.

LabVIEW permite la comunicación con múltiples interfaces (USB, GPIB, PXI...) entre las cuales se utilizará la comunicación OPC y dado que es un entorno de programación flexible, puede integrarse con dispositivos de medida y adquisición de datos, construir en sistemas de medida autónomos sobre una plataforma de entradas y salidas reconfigurables o incluso diseñar sistemas de monitoreo (entre muchas otras aplicaciones).

Para que distintos VIs se agrupen e interactúen entre sí, se crean proyectos con extensión 'lvproj.'

2.1.1. TIPOS DE DATOS

Esta parte es importante a la hora de entender la programación, así pues, se explicará algunos de los distintos tipos de datos que se puede encontrar en este lenguaje.

Como se ha dicho, LabVIEW es una plataforma de programación gráfica por lo que para definir tipos de datos, utiliza colores.

El color rosa representa la cadena de caracteres o '*string*'. Se utilizan para enviar y recibir mensajes de texto entre los diferentes módulos de una aplicación.

El color azul representa números enteros o '*integer*'. Dentro de éstos, podemos hacer una clasificación por su tamaño en bits, contando con enteros de 8 bits hasta 64 bits. Además

es posible definirlos tanto con signo (*'signed'*, con prefijo I), cuanto sin signo (*'unsigned'*, con prefijo U).

El color naranja representa número reales con decimales o vectores (*'arrays'*). Se emplea para procesar datos. Dentro de este dato hay muchas variantes dependiendo del número de bits de almacenamiento y dígitos decimales. En este trabajo el tipo de dato más empleado de esta clase es DBL (precisión doble de coma flotante) capaz de almacenar hasta 64 bits y posee 15 dígitos decimales.

Por último, el verde representa los datos de tipo booleano. Se utiliza para representar un 0 o 1 o un TRUE o FALSE, es decir, es utilizado para representar datos digitales y actuar como un conmutador en el panel frontal.

Dependiendo de si se tratan de un Scalar, un *'array'* 1D o 2D, el grosor del cable que une a las funciones aumentará en el sentido en que la complejidad de las dimensiones aumente.

En la siguiente figura (Figura 1), podemos observar todo lo comentado:






| Data Type | Scalar | 1D Array | 2D Array | Color |
|--------------------------|---|---|---|--------|
| Numeric - Floating Point |  |  |  | Orange |
| Numeric - Integer |  |  |  | Blue |
| Boolean |  |  |  | Green |
| String |  |  |  | Pink |

Figura 1. Tipos de datos. Imagen obtenida de National Instruments

Una vez comentado los tipos de datos, se detallará la programación.

2.1.2. PALETA DE CONTROLES

Se utiliza en el panel central y contiene los objetos necesarios para crear una interface de entradas y salidas de datos (controles e indicadores). *Figura 3*.

Como se puede observar, existen infinidad de sub-menús que contienen controles e indicadores respectivos de una clase de objeto.

En este trabajo, en gran parte, se ha utilizado los sub-menús:

- *'Numeric'*: se han creado constantes numéricas e indicadores gráficos de las coordenadas mostradas de 0 a 100 % (se explicará en el siguiente apartado).
- *'Boolean'*: se han creado indicadores, pulsadores 'STOP' y 'RESET'.
- *'String & Path'*: se ha creado un cuadro de diálogo en el que se indica el paso del proceso.

- *'Decorations'*: para decorar y organizar el panel frontal.

2.1.3. PALETA DE FUNCIONES

Se utiliza en el diagrama de bloques y contiene todos los objetos para crear y editar el programa (Figura 2).

Al igual que pasa con la paleta de controles, hay infinidad de funciones por explicar. Este trabajo se centrará en las más importantes:

- *'Structures'*: este sub-menú es muy importante, ya que es aquí de donde obtenemos las estructuras *'While loop'* (tan utilizadas) y *'sequence structure'* (empleada para la sucesión de procesos).
- *'Numeric', 'boolean', 'string' y 'comparison'*: estos sub-menús se utilizan para la transformación y comparación de datos, indicación de estado de controles por medio de leds y estado del proceso.
- *'Graphics & Sounds'*: este sub-menú, has sido utilizado para crear el modelado en 3D (en apartados posteriores, se explicará su programación).

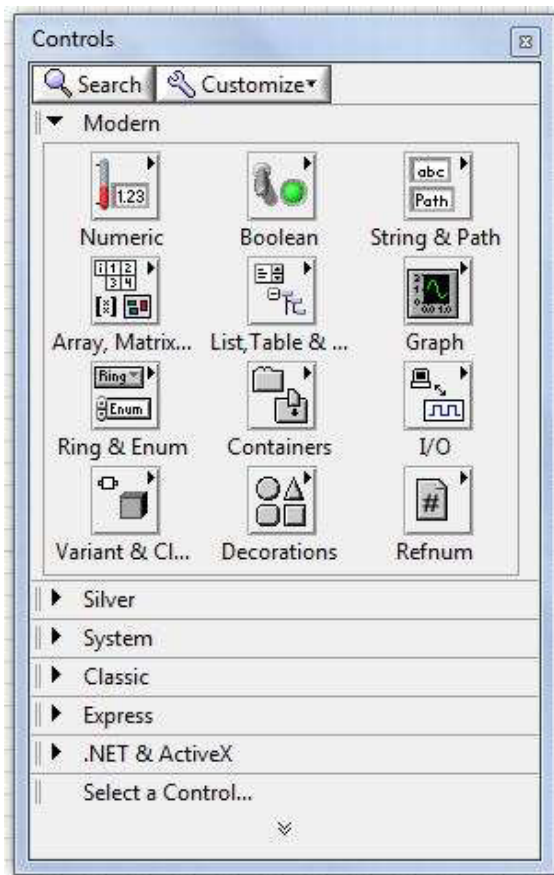


Figura 3. Paleta de controles.

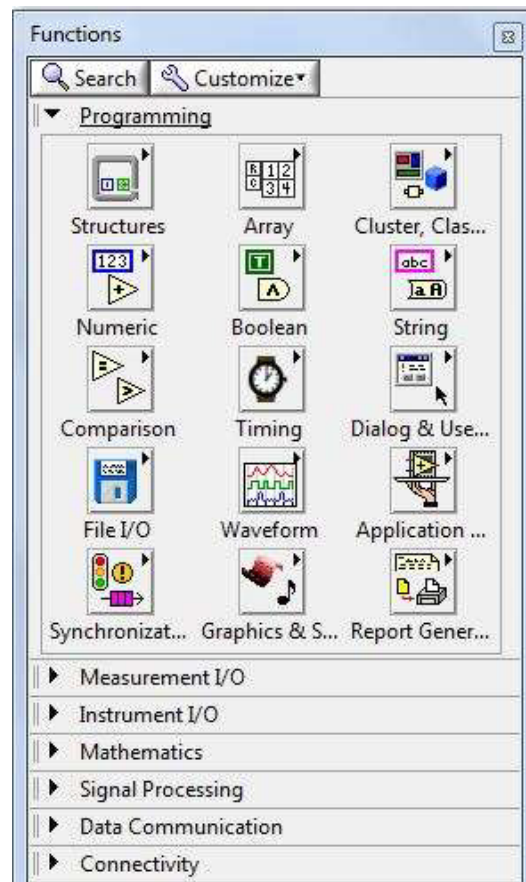


Figura 2. Paleta de funciones

2.1.4. VARIABLES LOCALES Y GLOBALES

I. VARIABLES LOCALES

Las variables locales son copia de un terminal de un control o indicador que se puede utilizar en cualquier lugar del diagrama. Su nombre se debe a que su alcance está limitado a un mismo VI (no puede utilizarse en otro distinto ni subVI asociado al propio VI).

Para crear una variable local, se debe hacer click en el botón derecho del ratón sobre el control o indicador que se desea copiar. Se debe buscar la opción Create → local variable. Esta copia de variable adopta el color y tipo de dato del control o indicador de origen.

Por defecto, estas variables se crean en modo escritura. Si se desea cambiar a lectura, se debe de hacer click en el botón derecho del ratón → 'Change to Read'.

En la programación de este trabajo, se recurre a variables de este tipo.

II. VARIABLES GLOBALES

Las variables globales son utilizadas para acceder y enviar datos entre varios VIs que se ejecutan simultáneamente.

Se pueden crear accediendo a la Paleta de funciones → 'structures' → global variable.

Tras crearla, se debe seleccionar el ítem que la define. Para ello se hace click en el botón derecho del ratón en la variable global → 'Select Item'. El aspecto dependerá del tipo de ítem seleccionado.

En este trabajo, este tipo de variables no son utilizadas ya que no ha sido necesario. Para conectar LabVIEW con el brazo robot se ha utilizado el servidor KepServer el cual ha permitido la compartición de datos.

2.2. ESTRUCTURA DEL PROGRAMA: 'BLOCK DIAGRAM'

El programa se divide principalmente en tres bloques:

2.2.1. PRIMER BLOQUE

Este bloque contiene un bucle while, encargado de la continua lectura de valores de posiciones del usuario hasta que éste pulse el botón 'STOP' gracias al cual, el programa se detiene.

El funcionamiento de este bucle se basa en la utilización de unos *tolkits*¹ para Kinect de Microsoft y su manejo para filtrar los datos de interés, que este caso corresponden con las posiciones de las manos derecha e izquierda.

Así pues, como ya se ha comentado, se obtuvieron las funciones base de la programación (de la web de Kinesthesia) mostradas en la *Figura 4*.

De la función 'Additional Vis' se despliega la *Figura 5*.

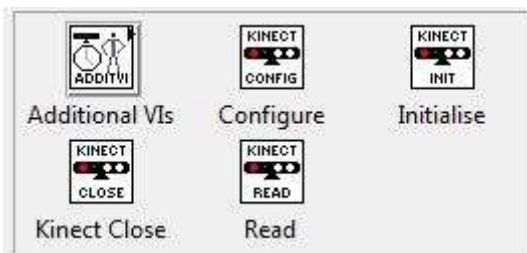


Figura 4. Funciones de Kinesthesia

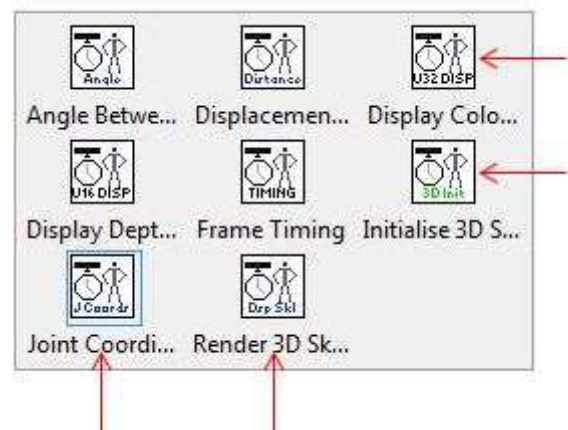


Figura 5. Additional VIs

A continuación se explicará brevemente el propósito de cada una de las funciones que han sido utilizadas:

- *Configure*: permite elegir entre las distintas opciones que ofrece el visionado directo con el kinect como por ejemplo video, resolución de imagen y profundidad.
- *Read*: Toma los valores de la lectura de kinect para tratarlos y organizarlos.
- *Display Colour data*: Convierte los datos leídos por la función 'Read' en una imagen 2D.
- *Joint coordinates*: Acepta valores numéricos entre 0-19 y devuelve la coordenada x, y o z, dependiendo del nodo seleccionado.

¹ Toolkit obtenidos de http://www.ni.com/gate/gb/GB_EVALTKTKINECTLV/US

- *Initialise 3D Skeleton*: establece la escena en la que se va a graficar el esqueleto en 3D.
- *Render 3D Skeleton*: Utiliza la imagen en 3D creada por LabVIEW para formarla y mostrarla en el espacio 3D del usuario (es decir, en el panel frontal).
- *Kinect close*: Se utiliza para cerrar todas las referencias del kinect.7
- *Initialise*: Ayuda a inicializar el Kinect.

Como se puede observar, la conexión de estas funciones, se realiza de forma secuencial y el tipo de dato que se transmite es una cadena de caracteres de 2 dimensiones.

Fuera del *'while loop'*, se ha utilizado *'initialise'* para realizar la llamada de encendido de Kinect. A continuación, se coloca la función *'configure'* para especificar la zona o parte de interés y que posteriormente, con la función *'initialise 3D skeleton'*, se cree el objeto a graficar en 3D. Seguidamente, se crea el *'while loop'* principal que engloba a toda la parte de lectura continua de datos.

Se utiliza la variable *'Read'* a la cual conectamos dos funciones *'Joint coordinates'* para realizar la lectura de datos. Es en este punto cuando seleccionamos, a la entrada de la función *'Joint coordinates'*, la parte del cuerpo a leer (como se nombró en el documento 1, Kinect es capaz de leer hasta 20 nodos, de entre los cuales, se elige los nodos correspondientes a ambas manos). A la salida, se conecta un *'Index Array'* para que devuelva una coordenada en cada una de las dimensiones (x, y, z) del nodo seleccionado.

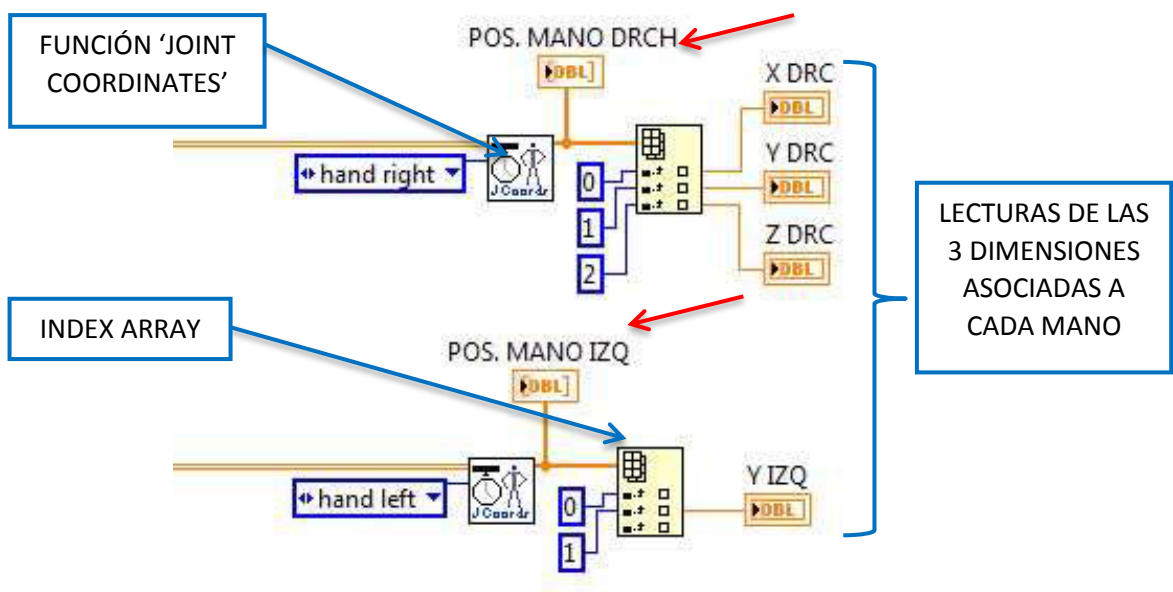


Figura 6. Captura de coordenadas de las manos en las 3 dimensiones del espacio

De esta forma, Kinect proporciona de forma continua los valores de las posiciones en las tres dimensiones del espacio de las manos del usuario (Figura 6).

Una vez leído los puntos de interés, se procede a la representación gráfica de todos los nodos del esqueleto y se cierra Kinect (para ello, en primer lugar se utiliza 'Render 3D Skeleton' y seguidamente, fuera del bucle, 'Kinect close').

Todo lo descrito, conforma el primer bucle (Figura 7).

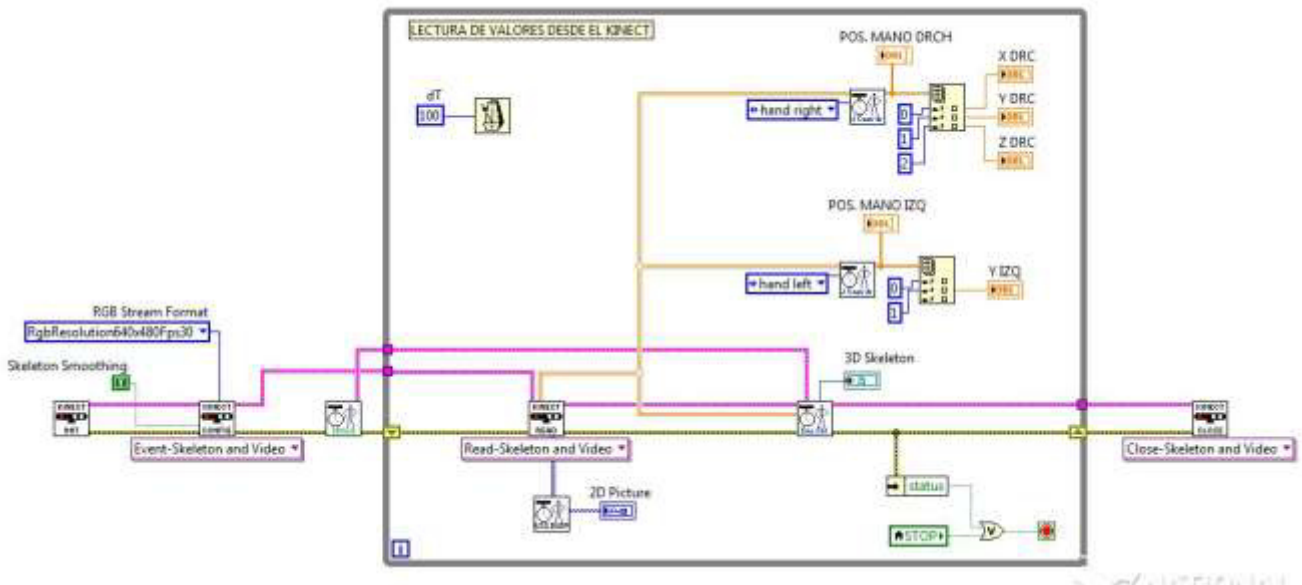


Figura 7. Programa principal de lectura continua

2.2.2. SEGUNDO BLOQUE

Este segundo bloque contiene el bucle de control principal, encargado de todo el procesamiento de datos para el correcto funcionamiento de la programación.

A grandes rasgos, está formada por una secuencia de fotogramas (frames) cada una de las cuales, realiza una función.

En los siguientes puntos a tratar se comentarán cada función.

I. INICIALIZAR PROCESO Y LECTURA DE MÁXIMOS/MÍNIMOS.

El primer 'frame' se encarga de inicializar el proceso. Esta inicialización está asociada a un pulsador llamado 'marcha'. Éste pulsador, por medio del servidor OPC, acciona un graficet (diagrama funcional creado en Unity PRO) que hace que el brazo robot vuelva a su posición inicial (finales de carrera accionados).

Una vez se ha inicializado el brazo robot y el botón marcha ha sido accionado, se pasa al siguiente *'frame'*. A partir de este punto, se ejecutan repetidamente cuatro estructuras para seleccionar los posibles valores máximos y mínimos de cada coordenada, es decir, se personaliza los rangos de captura de datos para cada usuario.

Por ejemplo, para la coordenada X se realiza la siguiente secuencia:

1. lectura máximo → 2. LED máximo en estado ON → 3. Lectura mínimo → 4. LED mínimo en estado ON.

En el panel frontal, mediante un indicador tipo cadena de caracteres y un dibujo explicativo, se le indica al usuario lo que debe de hacer. Por ejemplo, por pantalla aparece "PASO 1: estire mano derecha en dirección del eje X". A su lado, se muestra una imagen ilustrativa del procedimiento que debe de realizar. Cuando el máximo es encontrado, se acciona un LED que indica que el proceso ha sido completado y suena un *'beep'*. Seguidamente se realiza el mismo procedimiento para el mínimo valor en el eje X y el resto de coordenadas. En la *Figura 8* se ilustra el proceso.

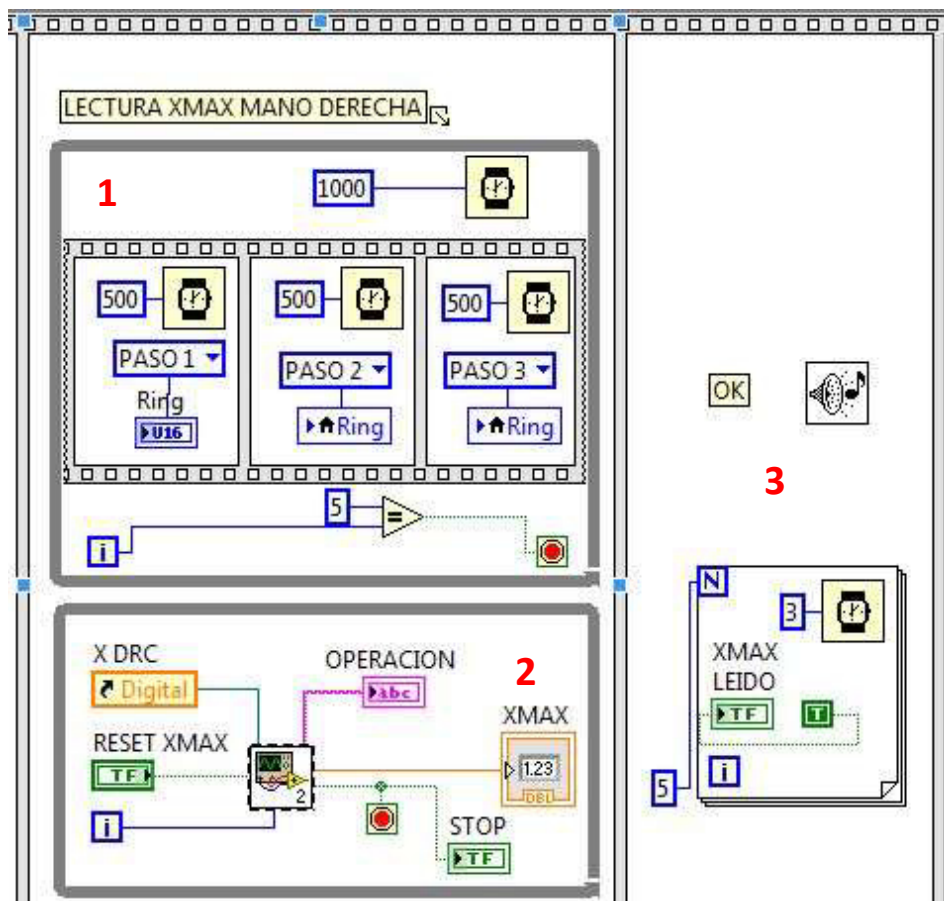


Figura 8. Lectura de máximos y mínimos.

Mediante tres números, se han indicado las partes de la estructura.

El número uno queda referido a la secuencia empleada para el cambio de imagen de ayuda al usuario que es mostrada en el panel frontal. Cada medio segundo se ilustra una imagen que, tras pasado un segundo y medio, forman un conjunto de imágenes imitando a una imagen animada. Esto ayudará a la comprensión del usuario.

El número dos marca el bucle de búsqueda del máximo o mínimo.

Dado que es un programa extenso, se ha creado SubVI (cuadrado centrado en el bucle con el símbolo de LabVIEW) encargado de compactar la información para evitar que la programación sea confusa.

El funcionamiento de cada SubVI es el mostrado en la *Figura 9* (se explicará el funcionamiento para la coordenada X de la mano derecha). Este procedimiento es también el empleado para el resto de coordenadas.

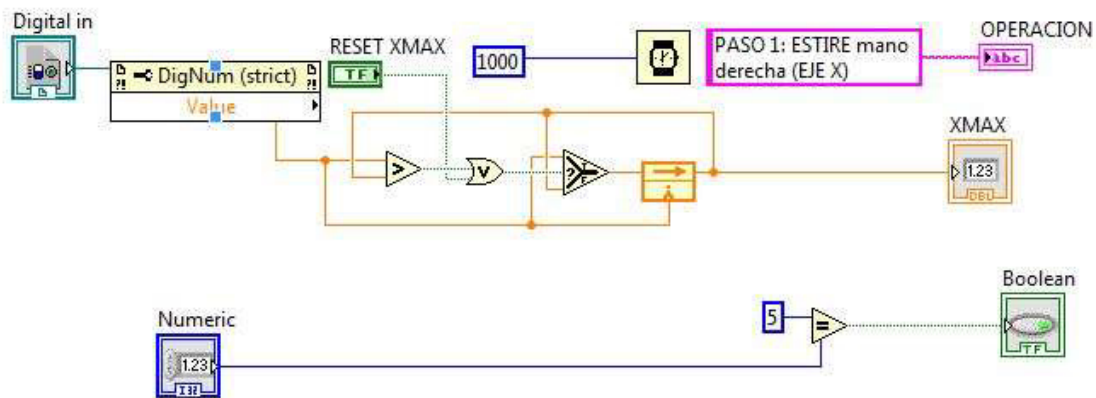


Figura 9. Operaciones encargadas de encontrar máximos y mínimos.

Se toma un primer valor de la coordenada X (obtenido en el primer bucle y copiado en una variable local). Se conecta a dos comparadores numéricos, una función de tipo booleano (puerta 'or') y un indicador. Se actualiza la coordenada X tomando un segundo valor. Si el segundo valor tomado es mayor que el valor anterior, el segundo valor pasa a ser el máximo. De lo contrario, el primer valor tomado es el más grande encontrado. Este procedimiento se realiza durante cinco segundos. Si durante ese período no se ha encontrado un valor más grande que el guardado como máximo, se toma como máximo el guardado y se pasa al siguiente 'frame'. Como se ha dicho en el párrafo anterior, este mismo procedimiento es el empleado para tomar el máximo valor del resto de coordenadas y cambiando mayor por menor y máximo por mínimo, se obtiene el procedimiento a seguir para la búsqueda del mínimo valor.

El botón verde llamado 'RESET', es empleado en caso de que los datos leídos sean erróneos. Pulsando este botón, se consigue que lo que se lea, sea lo mismo a lo que se compare, por lo que no se guarda máximo.

En último lugar, el número tres se refiere al proceso de encendido de los LEDs.

Si el proceso ha resultado exitoso, en el panel frontal aparecerán un total de ocho LEDs encendidos, correspondientes a cada coordenada (hay dos LEDs para el eje horizontal, dos para el eje vertical, dos para el giro y otros dos para la pinza).

La siguiente 'frame' apaga todos los LEDs para que otro usuario pueda utilizar el programa.

II. CONVERTIR LECTURAS A VALORES DE CERO A CIEN POR CIENTO.

Con la lectura de máximos y mínimos se ha obtenido un rango de valores para los cuales el brazo robot debe de moverse. Si se lee una posición de una coordenada mayor que la máxima, el brazo robot se debe de mover al máximo. Si se lee una posición de una coordenada menor que la mínima, el brazo robot se debe de mover al mínimo, de tal forma que no se puede sobrepasar ni el máximo ni el mínimo.

Para pasarle las coordenadas al brazo robot en primer lugar se convertirán a valores de cero a cien por cien (se normalizarán las lecturas) y en segundo lugar, se pasarán a lenguaje del brazo robot (se explicará en el apartado '3. Programación en Unity Pro').

Para obtener la normalización de datos (Figura 10), se ha procedido de la siguiente manera:

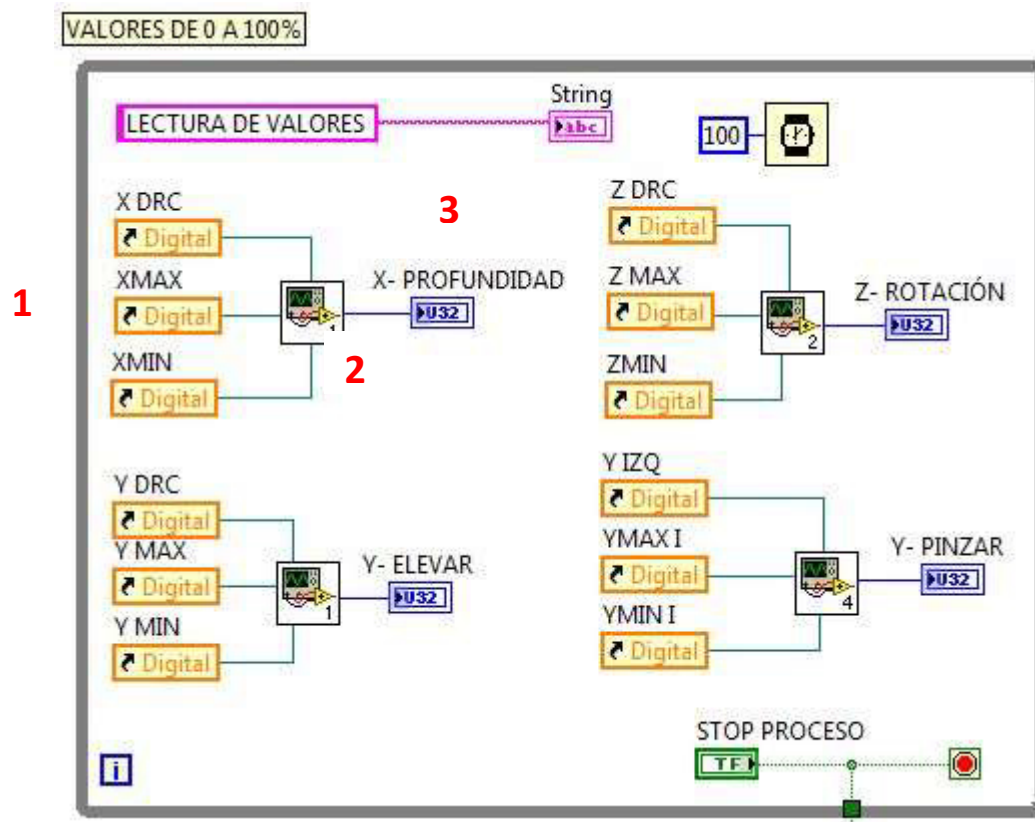


Figura 10. Estructura para convertir lecturas de cero a cien por cien.

Con el número uno se indica las variables que intervienen en este apartado. Como se puede observar, se han utilizado tres variables para cada transformación. La primera variable de cada SubVI (X DRC, Y DRC, Z DRC e Y IZQ) corresponde con el dato leído desde kinect. Las siguientes variables (definidas con sufijo MAX o MIN) corresponden a las obtenidas en el punto anterior.

El número dos se refiere a cada SubVI encargada de realizar la transformación. En la *Figura 11*, se muestra el método empleado.

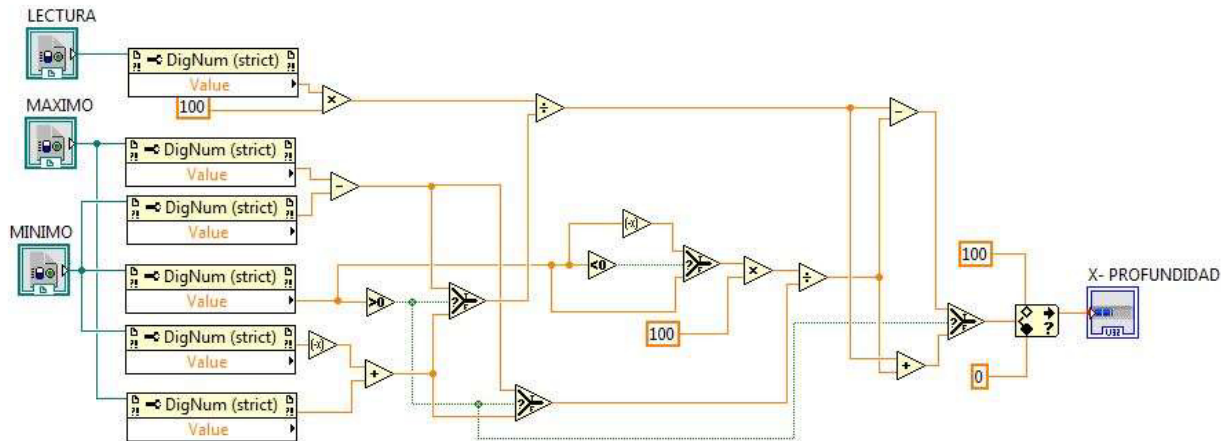


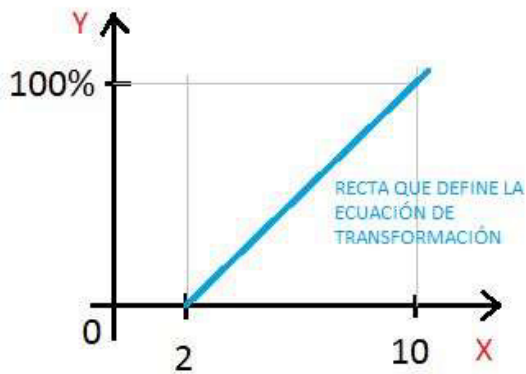
Figura 11. Normalización de valores a tanto por cien.

Numéricamente, lo que se pretende con este SubVI es calcular la ecuación de la recta que une los puntos con el matiz de que el valor mínimo puede ser negativo.

Para una mejor comprensión, se explicará esta imagen a partir de un ejemplo numérico.

➤ CASO A: MÍNIMO POSITIVO

Se define que el máximo valor de la posición de la mano derecha en el eje X para un usuario determinado es 10, mientras que el mínimo valor leído en las mismas condiciones, es 2. Si se dibuja unos ejes coordenados donde el eje de ordenadas queda limitado por 0% como valor mínimo y 100% como valor máximo y el eje de abscisas queda limitado por 10 y 2 (valores máximos y mínimos del usuario), se comprueba que uniendo puntos, se forma una recta que define la transformación (*Figura 12*).



La ecuación de la recta es:

$$Y = \frac{100}{MAX - MIN} * X - \frac{MIN * 100}{MAX - MIN}$$

La ecuación aplicada para este ejemplo es:

$$Y = \frac{100}{10 - 2} * X - \frac{2 * 100}{10 - 2}$$

Figura 12. Recta que define la transformación para valores de mínimo positivo.

Siendo X el nuevo valor que lee kinect e Y, la transformación a porcentaje de la lectura.

En la siguiente imagen (Figura 13), se resalta mediante líneas de colores, la ecuación implementada en LabVIEW

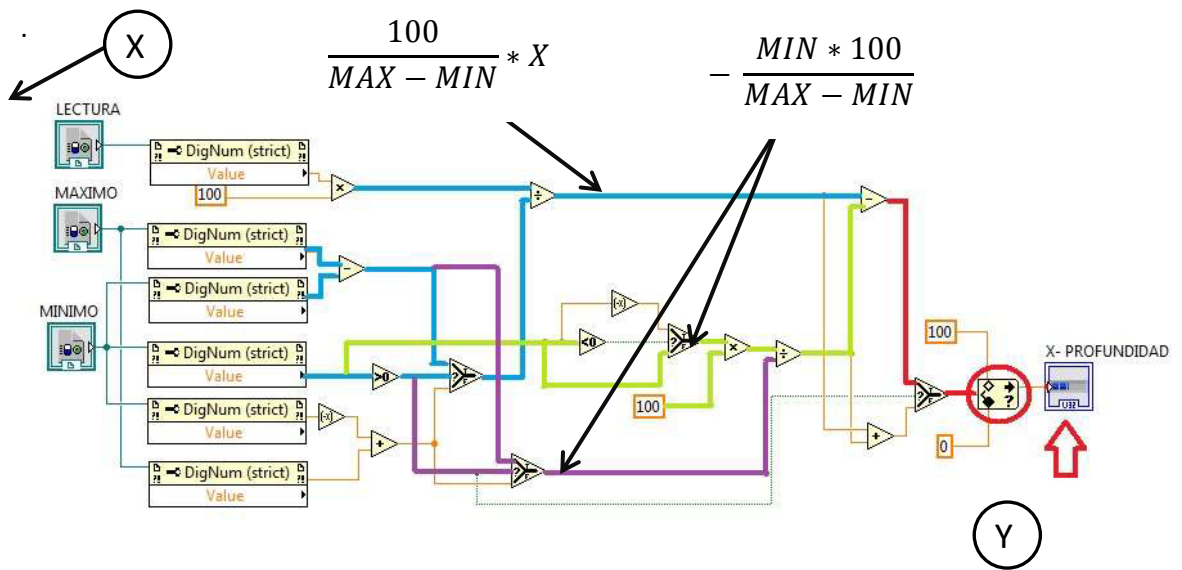
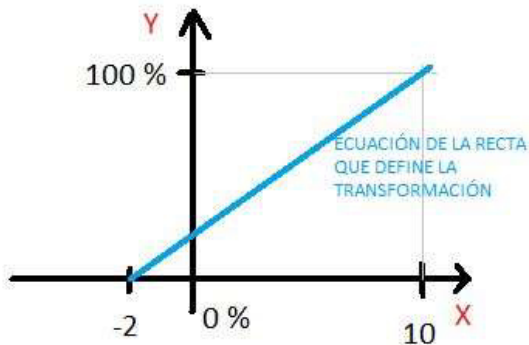


Figura 13. Ecuación de la recta trasladada a LabVIEW.

➤ CASO B: MÍNIMO NEGATIVO

En este caso, el valor mínimo es negativo y la ecuación de la recta cambia.

Igual que en el apartado anterior, se define que el máximo valor para la posición de la mano derecha es 10, mientras que ahora el mínimo valor se corresponde a -2. Si volvemos a dibujar los ejes coordenados, obtenemos la *Figura 14*.



La ecuación de la recta es:

$$Y = \frac{100}{MAX + MIN} * X + \frac{MIN * 100}{MAX + MIN}$$

La ecuación aplicada para este ejemplo es:

$$Y = \frac{100}{10 + 2} * X + \frac{2 * 100}{10 + 2}$$

Figura 14. Recta que define la transformación para valores mínimos negativos.

Siendo X el nuevo valor que lee Kinect e Y, la transformación a porcentaje de la lectura.

En la siguiente imagen (*Figura 15*), se resalta mediante líneas de colores, la ecuación implementada en LabVIEW.

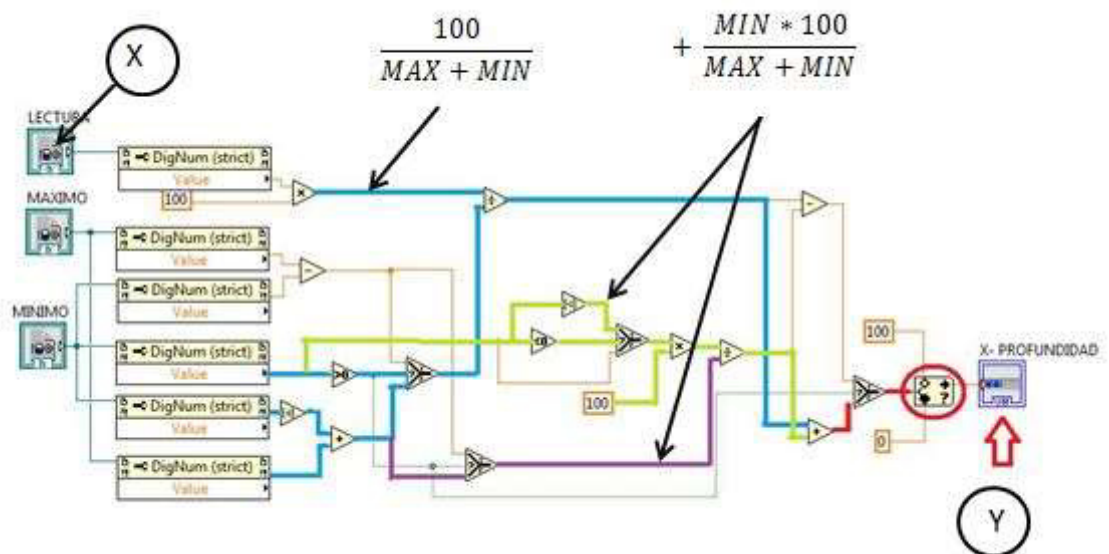


Figura 15. Ecuación de la recta trasladada a LabVIEW.

Así se ha conseguido obtener los valores adecuados en porcentajes.

La función rodeada con un círculo rojo se llama 'In Range and Coerce' cuya función es determinar si el dato leído se encuentra entre 100 y 0. En caso de que sea superior a 100, fuerza a X-PROFUNDIDAD sea 100. En caso de que sea menor que 0, fuerza a X-PROFUNDIDAD

sea 0. La variable X-PROFUNDIDAD es la que se utiliza para representar gráficamente en el panel frontal el eje horizontal.

Del mismo modo, se definen las variables Y-ELEVAR para representar el eje vertical, Z-ROTACIÓN para representar el giro e Y-PINZAR para representar las pinzas del brazo robot.

Una vez obtenidos y representados los valores normalizados y en porcentaje de cada coordenada, se procede a la transformación al lenguaje del brazo robot.

III. TRANSFORMACIÓN DE COORDENADAS AL LENGUAJE DEL BRAZO ROBOT. PULSOS.

En el apartado anterior, se han obtenido las coordenadas en tanto por ciento. En este apartado lo que se pretende es traducir estos porcentajes a valores máximos y mínimos de pulsos.

Como se explicará en el siguiente apartado, el desplazamiento de los ejes del brazo robot se han ido computando mediante un contador. Cada pulso que se cuenta, es un movimiento.

De esta forma, y aplicado en concreto al brazo robot 3-D-Robot TX FischerTechnik, se ha obtenido que el máximo pulso en el eje vertical es 67, en el eje horizontal 67, en el giro 70 y en las pinzas 15. Para todos los ejes, el mínimo pulso se corresponde con 0. El valor de pulso igual a 0 se da cuando los desplazamientos en los tres ejes coordenados, llegan a sus respectivos finales de carrera.

De nuevo, para realizar la transformación, hay que hallar la ecuación de la recta.

Este caso es mucho más sencillo, ya que la transformación sólo depende de la pendiente de la recta que pasa por el origen (se transforma valores de [0-100]% a valores de [0-máximo del eje coordenado]). *Figura 16.*

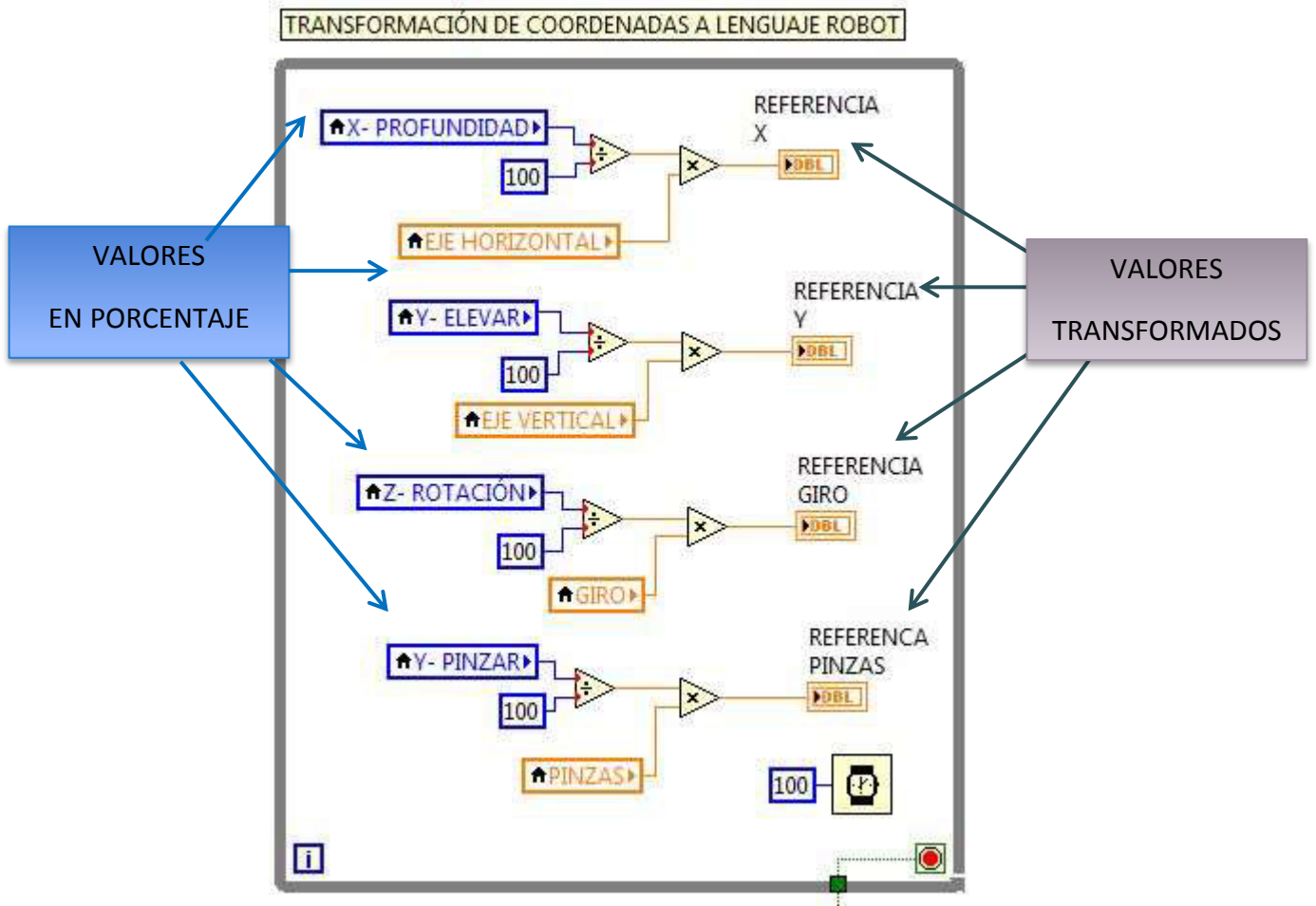


Figura 16. Ecuación de la recta para transformación de coordenadas al lenguaje del brazo robot.

IV. FILTRADO DE LA LECTURA OBTENIDA DE KINECT.

Este apartado se presenta como una posible mejora en la programación cuyo objetivo es filtrar las posibles oscilaciones que se realicen en transformación y la lectura de variables desde Kinect.

Cuando Kinect está leyendo valores de las posiciones relativas a las manos, no lee un valor fijo, sino que dicho valor oscila debido a que hay un ruido asociado a cada lectura.

Como respuesta a dicha oscilación, se ha colocado una función de filtrado de LabVIEW. Cada X rango de valores, aplica una media entre todos ellos (Figura 17).

En este caso, cada 10 valores, proporciona una media de los mismos para que puedan ser enviados.

Con objeto de evitar problemas de control del brazo robot, se ha optado por la sustitución del bucle *while loop* por una estructura tipo *timed lood* que es más estable y se puede manejar de manera precisa el tiempo.

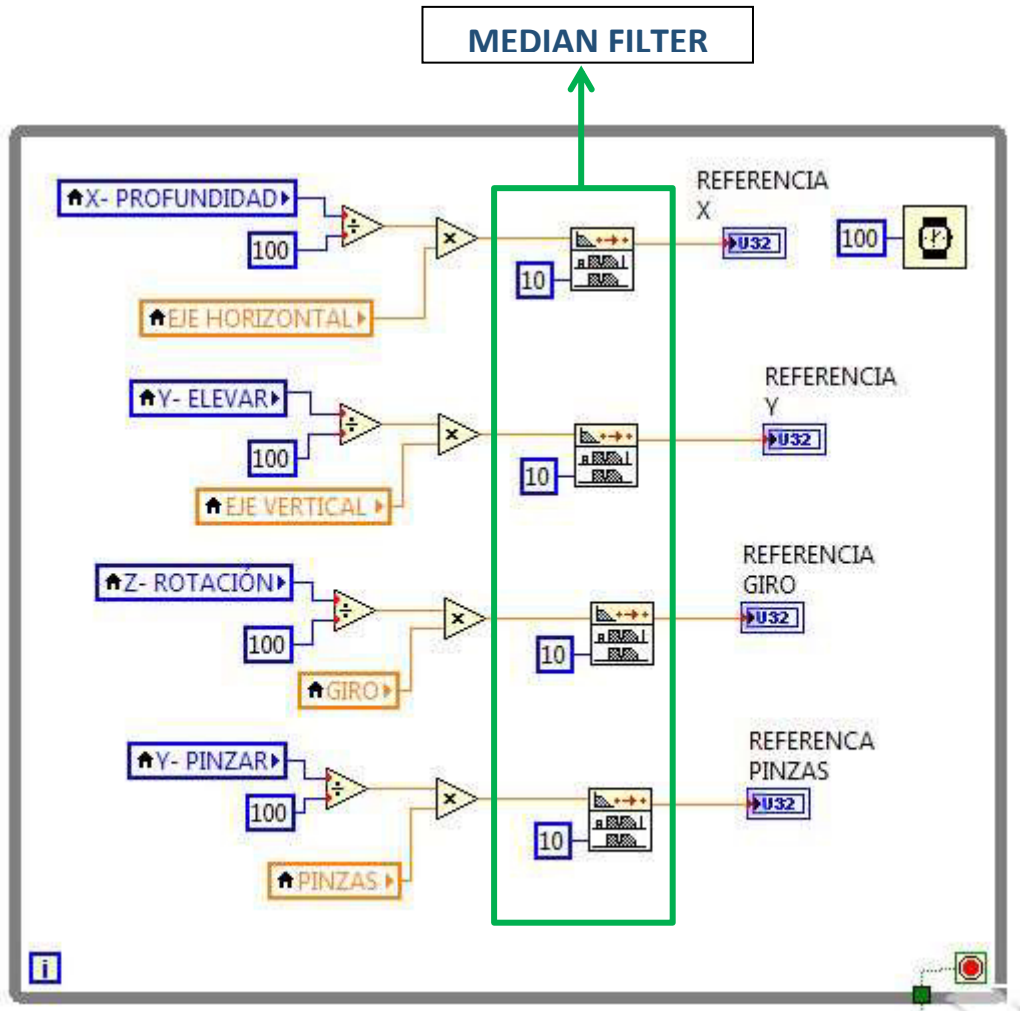


Figura 17. Mejora de la lectura. Aplicación del filtro.

2.2.3. TERCER BLOQUE

Ésta es la última parte que constituye a la programación. Se trata de un bucle *while* encargado de formar una figura en 3D del brazo robot y desplazarse conforme a los movimientos que recibe desde el servidor de comunicación OPC client. Estos movimientos se corresponden con los desplazamientos reales del brazo robot.

Para formar una imagen en 3D, se elige un plano base sobre el cual, se van dibujando figuras primitivas referidas al primer elemento dibujado, esto es, componer escenarios a partir de objetos de primitivas. En este caso, se ha elegido a la Base del brazo robot como elemento fijo (Figura18).

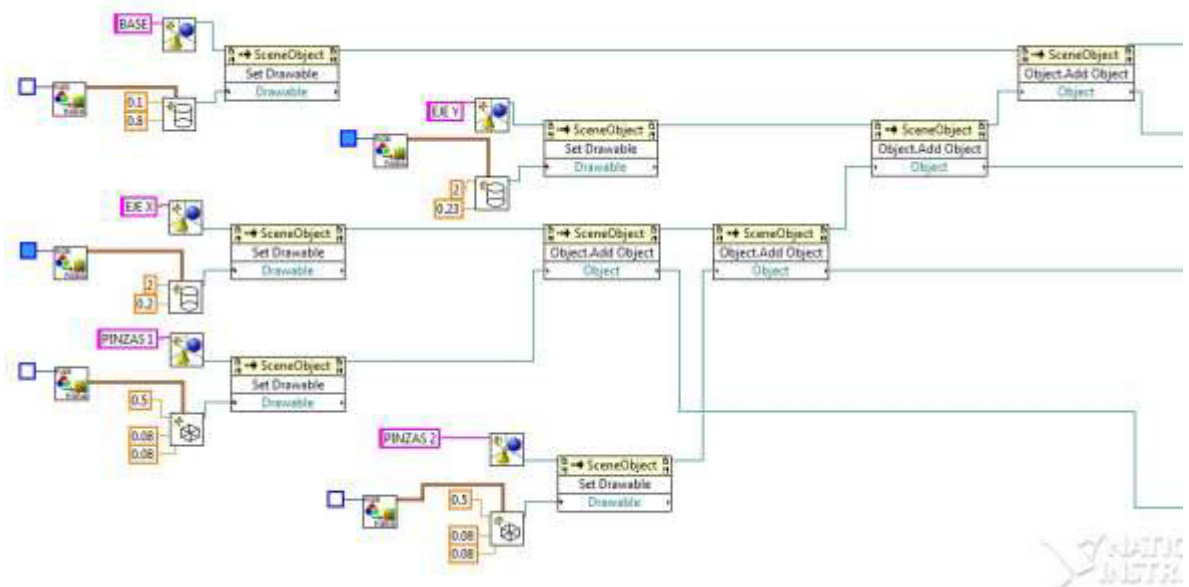


Figura 18. Estructura para crear una imagen en 3D a partir de figuras primitivas

Las funciones que se han utilizado para crear el modelado en 3D, se encuentran localizadas en Paleta de funciones → 'Graphics & Sounds' → 3D Picture Control.

En primer lugar, para definir cada una de las piezas que forman el modelo en 3D, ha sido necesario el uso de cuatro bloques (Figura 19).

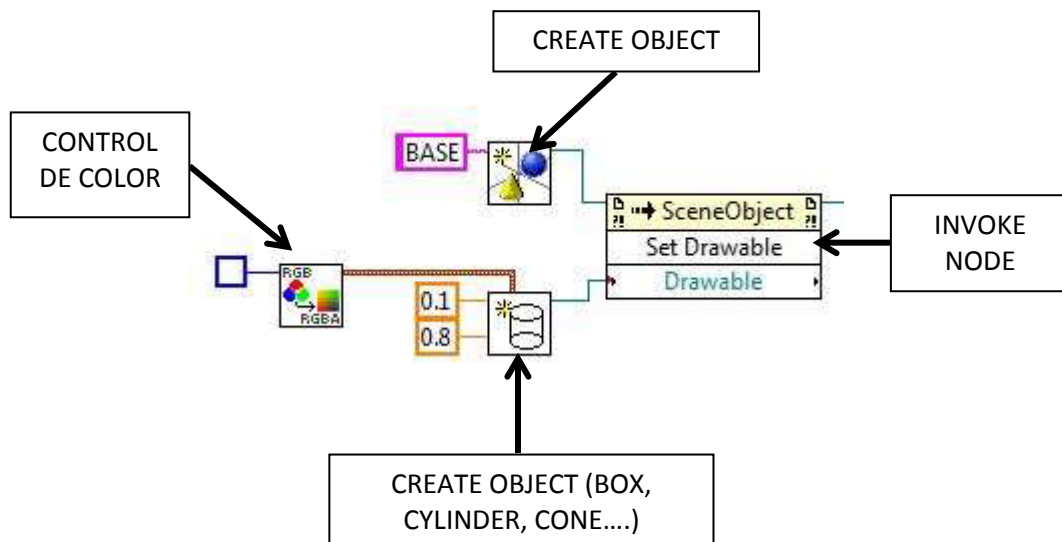


Figura 19. Definición de un cilindro en 3D.

'*Create object*' es la función empleada para la creación de un nuevo objeto 3D. Se utiliza la función '*create box or cylinder*' para formar las geometrías y el control de color para modificar el color (los colores elegidos para el brazo robot han sido gris y azul).

'*Invoke node*' permite realizar una acción sobre una referencia. Se han utilizado dos tipos de acciones: '*Add object*' para añadir objetos a una referencia y '*Set Drawable*' para dibujar objetos sobre una referencia.

Los números enmarcados con cuadrado naranja en la figura anterior, corresponden con las dimensiones de las geometrías (dimensión X, Y, Z respectivamente). Éstas son de importancia ya que posteriormente serán utilizadas para transformar los pulsos recibidos del modelo real al modelo gráfico.

Tanto la longitud del eje vertical como horizontal, corresponde con 2 unidades. 2 será el valor máximo que se pueden desplazar dichos ejes.

En segundo lugar, se han posicionado las geometrías primitivas sobre el plano de trabajo para que el conjunto de ellas formen la figura buscada.

Para posicionar las piezas ha sido necesario el uso de funciones de translación y rotación que tienen su origen en su centro geométrico. Estas operaciones son realizadas de forma secuencial y el orden en la que son aplicadas es muy importante.

Para limpiar cualquier operación, de translación o de giro, que se haya realizado a una geometría, se utiliza '*clear transformation*'. En caso de no utilizar esta función, las transformaciones serían acumulativas y se perderían las operaciones. Por ello, para evitar confusiones, se coloca dicha función antes de realizar ninguna acción.

A grandes rasgos se podría resumir que el proceso de modelado en 3D conlleva dos pasos. En primer lugar se crea el escenario virtual y en segundo lugar se aplican las transformaciones necesarias a cada geometría para formar el brazo robot.

En este caso, se procederá de modo que el brazo robot quede inicializado como lo haría el real (*Figura 20*). Además, dependiendo del orden de aplicación de las transformaciones, los ejes de referencia de cada pieza cambian.

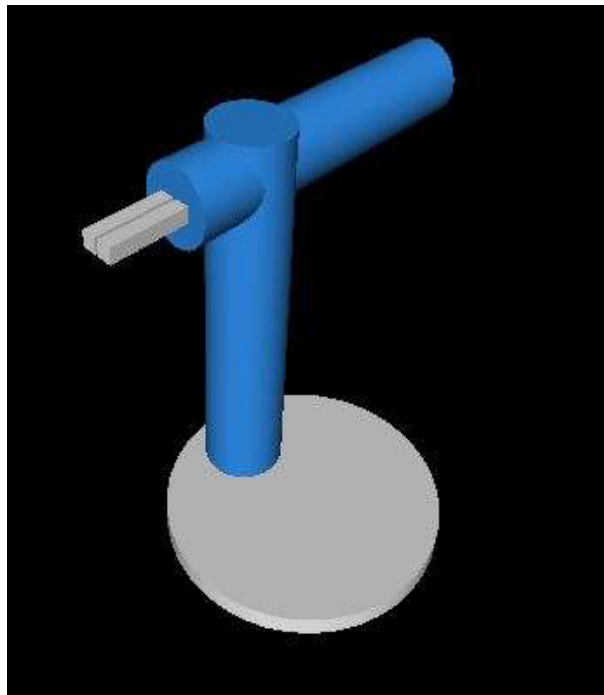


Figura 20. Modelado en 3D.

En la *Figura 21*, se muestra el cilindro horizontal con los ejes coordenados resultantes tras aplicar las transformaciones.

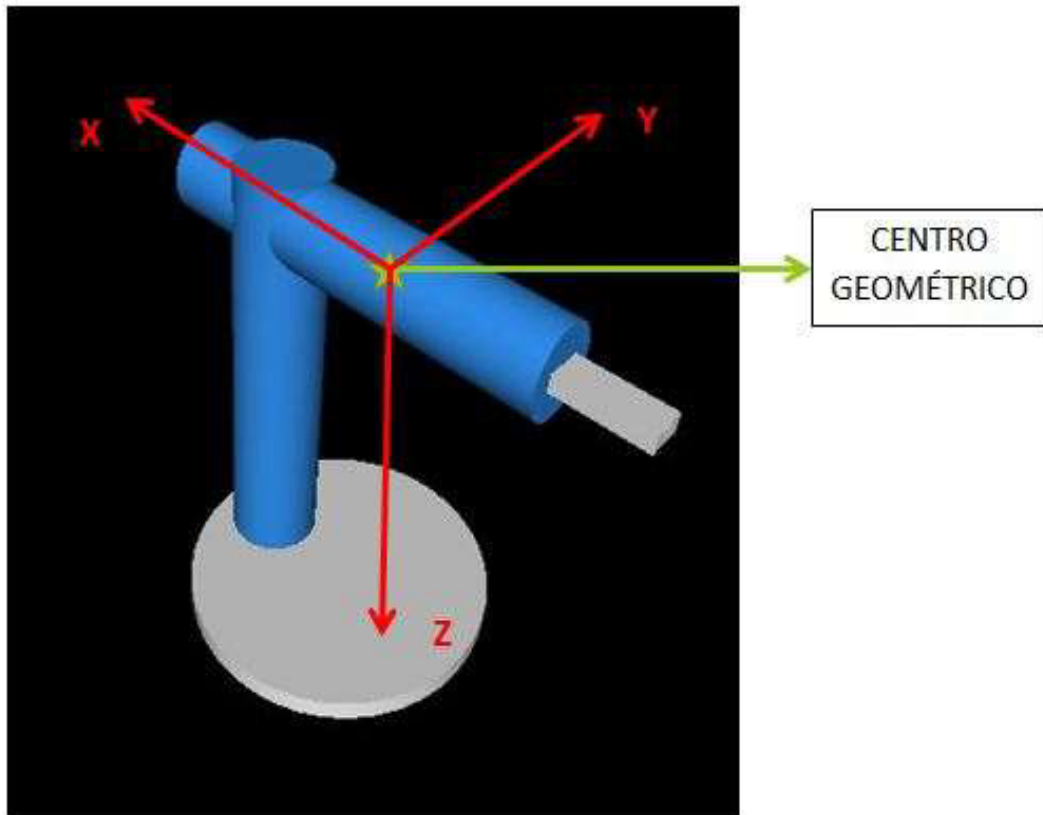


Figura 21. Ejes barra horizontal.

Conocer la orientación de los ejes coordenados es necesario para realizar un desplazamiento correcto del brazo robot real en el modelado. Por ejemplo, se modifica la coordenada X del cilindro horizontal para desplazarse sobre el eje horizontal, la coordenada Z del cilindro horizontal para desplazarse sobre el eje vertical, la coordenada Z del cilindro vertical para girar y las coordenadas Z y X de las pinzas para poder pinzar. Esto queda reflejado en la *Figura 22* en donde se han marcado con cuadrados azules la parte que debe de ser modificada para conseguir el movimiento deseado.

Las variables utilizadas para animar el modelo en 3D corresponden con las posiciones reales del brazo robot. Estas posiciones son conocidas gracias a los contadores de pulsos creados.

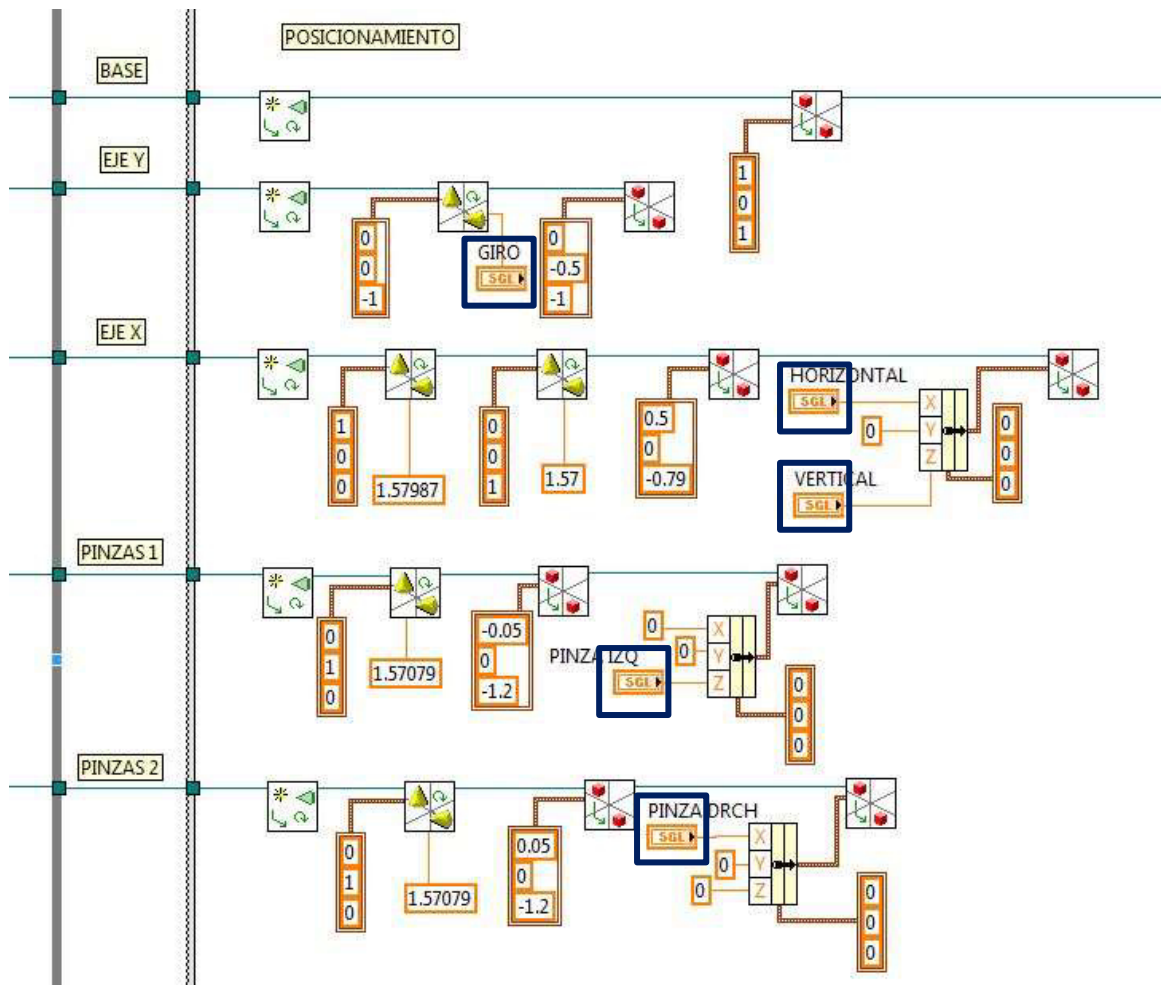


Figura 22. Modificación de los parámetros para que se mueva de acuerdo al brazo robot real.

Modificando las variables marcadas con un cuadrado azul se podrá visualizar en el panel frontal una imagen 3D del brazo robot que se moverá de acuerdo a la realidad.

2.3. ESTRUCTURA DEL PANEL DE USUARIO: 'FRONT PANEL'

Este apartado se comentará brevemente dado que se realizará una descripción más detallada en el "documento nº 4: Manual de usuario".

El enfoque que se va a mostrar es desde el punto de vista de la programación, es decir, se pretende definir qué funciones han sido utilizadas para organizar los datos o cómo se han representado las variables expuestas en los apartados anteriores.

La pantalla principal ha sido organizada mediante 'tag control' y éste a su vez, ha sido dividido en diferentes pestañas. Para cambiar de pestañas, en 'block diagram' se ha creado una variable local del 'tab control' en modo escritura, a la que se le ha conectado un seleccionador que permite elegir la pestaña a mostrar.

3. PROGRAMACIÓN EN UNITY PRO

Unity Pro ha sido el programa auxiliar empleado para la creación de los diagramas de flujo que ayudan a tratar los datos procedentes del servidor OPC para gobernar el funcionamiento del brazo robot.

La interfaz de usuario está formada por varias ventanas y barras de herramientas. Principalmente se trabajará con la ventana de explorador de proyectos y la ventana de editor.

A partir de la barra de estado (situada en la parte inferior de la ventana) se comprobará si el programa ha sido generado y éste ha sido transmitido al autómeta. Para ello, se hace click en PLC situado en la barra de herramientas. En primer lugar se comprueba que la dirección IP coincide con la del autómeta. A continuación se hace click en analizar proyectos y si no hay errores éste podrá generarse.

En el explorador de proyectos, dentro de 'Comunicaciones' → 'Red', se ha realizado la comunicación de red y dentro de 'Programas' → 'MAST', se han creado las siguientes secciones (son unidades de programa autónomas en las que se crea la lógica del proyecto):

3.1. CONTADOR

Se ha creado la sección “contador”, de tipo LD (lenguaje de esquema de contactos) el cual contiene cuatro contadores (Figura 23) con objeto de conocer máximos desplazamientos en el eje vertical, horizontal, giro y pinzas del robot, es decir, vamos a calibrar el robot contando pulsos para conocer sus limitaciones y llevarlo a una posición determinada.

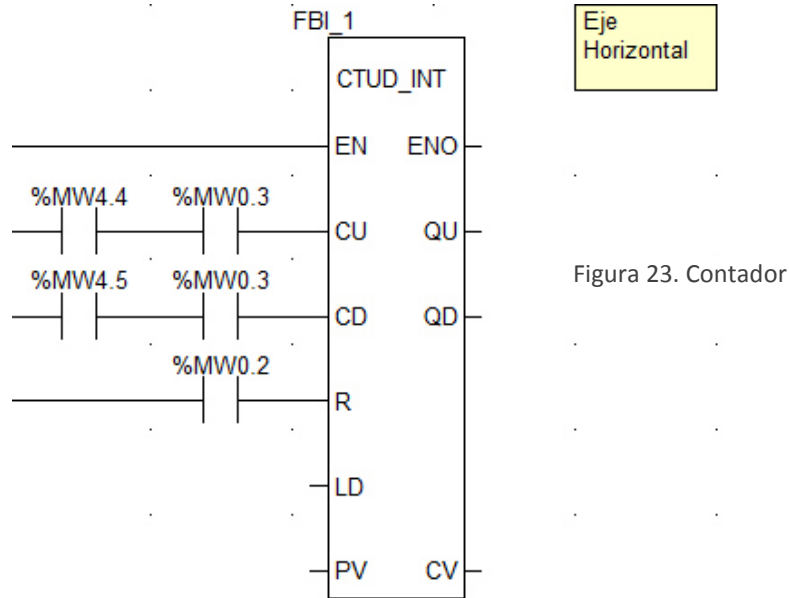


Figura 23. Contador

| PARÁMETRO | DESCRIPCIÓN |
|-----------|---|
| CU | Entrada de disparador de contador ascendente |
| CD | Entrada de disparador de contador descendente |
| R | Reset |
| LD | Carga de datos |
| PV | Valor preestablecido |
| QU | Indicador ascendente |
| QD | Indicador descendente |
| CV | Valor de cómputo (valor real) |

Tabla 1. Descripción de los parámetros de entrada y salida de CTUD

Se ha colocado en la salida CV la dirección a la que se quiere transmitir el conteo de pulsos (posteriormente, esta dirección será tratada en LabVIEW para el control del modelado en 3D):

- Dirección %MW 10 para controlar el movimiento vertical.
- Dirección %MW 11 para controlar el movimiento horizontal.
- Dirección %MW 12 para controlar el giro.
- Dirección %MW 13 para controlar la pinza.

3.2. GRAFCET INICIAL

Se ha creado la sección “INICIALIZACIÓN”, de tipo SFC (Lenguaje de ejecución secuencial), para llevar al brazo robot a su posición inicial, que como ya se ha explicado en el ‘Documento 1: Memoria Descriptiva’, corresponde con la posición en la que todos los finales de carrera se encuentran accionados (Figura 24):

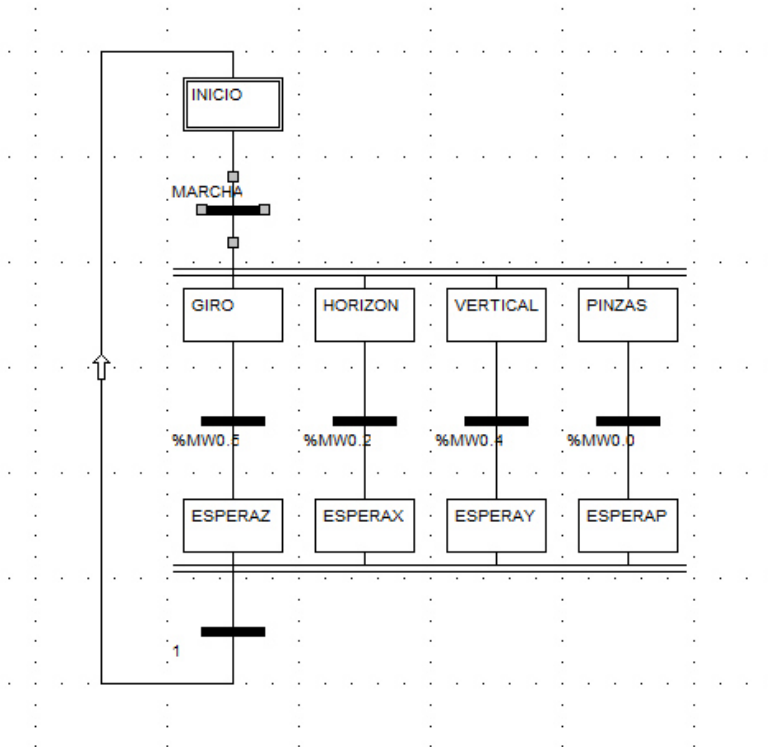


Figura 24. Grafcet de inicialización

‘MARCHA’ es la transición que se ejecuta cuando, desde el panel frontal de LabVIEW se acciona la dirección %MW30.1.

Una vez accionado ‘MARCHA’, se accionan todos los motores para que éstos lleven al robot a su posición de inicio.

3.3. GRAFCET DE MOVIMIENTOS

Del mismo modo que con “INICIALIZACIÓN” se ha creado una sección de tipo SFC para monitorizar los movimientos del brazo robot de acorde a los datos enviados desde Kinect (Figura 25):

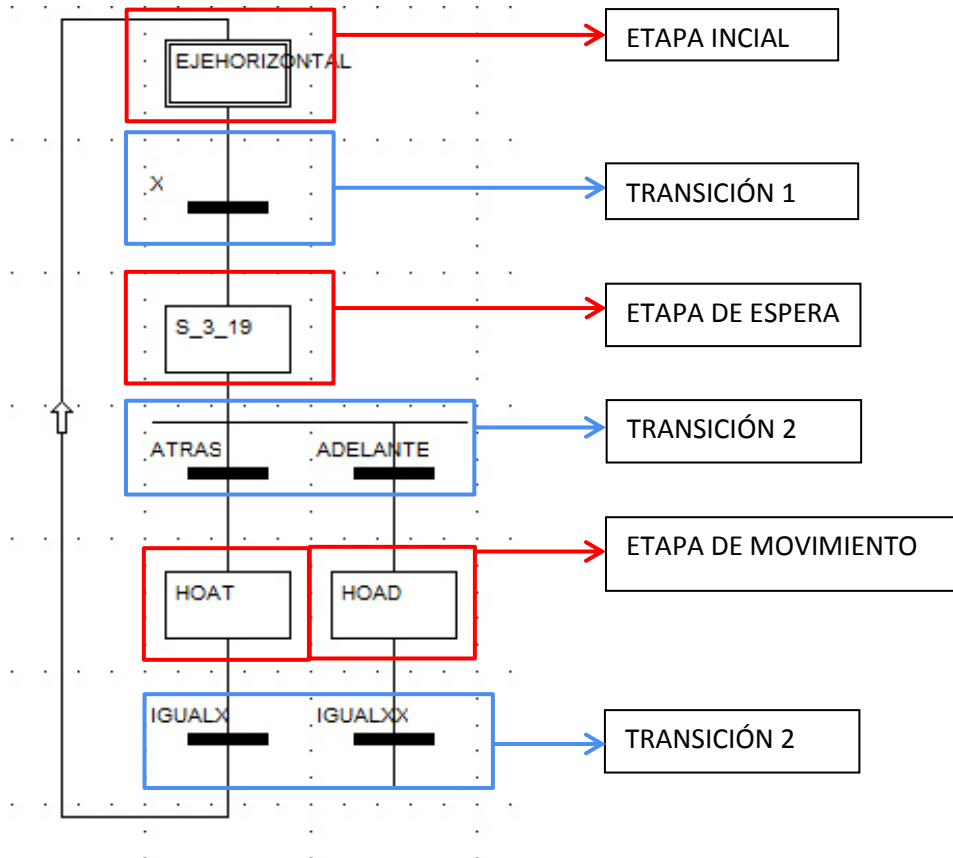


Figura 25. Grafcet de movimientos

Para que pueda ser explicado el proceso de etapas – transiciones, debe de especificarse las variables que se han compartido con el servidor OPC (Tabla 2).

| VARIABLES QUE DEPENDEN DE LABVIEW | | | | |
|-------------------------------------|-----------|---------|-----------|--|
| NOMBRE | DIRECCIÓN | NOMBRE | DIRECCIÓN | |
| REF_Y | %MW20 | TRANS_Y | %MW40.1 | |
| REF_X | %MW21 | TRANS_X | %MW41.1 | |
| REF_G | %MW22 | TRANS_G | %MW42.1 | |
| REF_P | %MW23 | TRANS_P | %MW43.1 | |
| MARCA | %MW30.1 | | | |
| VARIABLES QUE DEPENDEN DE UNITY PRO | | | | |
| NOMBRE | DIRECCIÓN | NOMBRE | DIRECCIÓN | |
| VERTICAL | %MW10 | GIRO | %MW12 | |
| HORIZONTAL | %MW11 | PINZAR | %MW13 | |

Tabla 2. Variables en las direcciones de la memoria.

La transición 1 se encuentra condicionada con la variable TRANS_X. Dicha transición no se cumplirá hasta que se accione la variable asociada.

Una se ha cumplido la transición, se pasa a una etapa de espera desde la que se dividen dos salidas.

La primera salida es empleada en caso de que la posición actual del robot fuese mayor que la posición que le envía el usuario, es decir, que el brazo robot se encuentre en una posición adelantada y el usuario quiera retroceder.

La segunda salida corresponde al caso inverso, en el que el robot se encuentre en una posición retrasada y el usuario quiera avanzar.

El robot se mueve dependiendo de qué parte de la transición se ha cumplido. Se mueve hasta que ocurra uno de los siguientes pasos:

- Que se mueva hasta su máximo pulso.
- Que se accione el final de carrera.
- Que los pulsos reales se igualen a los requeridos por el usuario.

Este proceso ha sido realizado con las tres posiciones restantes.

4. BIBLIOGRAFÍA

1. Tipos de datos y estructuras de LabVIEW

NATIONAL INSTRUMENTS. Tipos de datos y estructuras. <<http://www.ni.com/academic/students/learnlabview/esa/datatypes.htm>> [Consulta: 4 de abril].

NATIONAL INSTRUMENTS. Global variables. <http://zone.ni.com/reference/en-XX/help/371361H-01/lvconcepts/glob_variables/>

2. Manual de referencia de Unity PRO

SCHNEIDER- ELECTRIC. Unity Pro. *Lenguajes y estructura del programa. Manual de referencia.* <http://www2.schneiderelectric.com/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/31000/FA31761/es_ES/Unity%20v41%20-%20Manual%20de%20referencia.pdf> .



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



ESCUELA TÉCNICA
SUPERIOR INGENIEROS
INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DESARROLLO DE UN SISTEMA PARA LA TELEOPERACIÓN DE UN ROBOT MANIPULADOR CILÍNDRICO MEDIANTE RECONOCIMIENTO DE GESTOS A TRAVÉS DEL SENSOR KINECT.

DOCUMENTO Nº4: MANUAL DE USUARIO

AUTORA: RODRÍGUEZ PÉREZ, M^ªDOLORES

TUTOR: HERRERO DURÁ, JUAN MANUEL

COTUTOR: SIMARRO FERNANDEZ, RAÚL

Curso Académico: 2014-15

CONTENIDO

| | |
|--|----------|
| ÍNDICE DE FIGURAS | 2 |
| ÍNDICE DE tablas..... | 2 |
| 1. REQUISITOS DE HARDWARE Y SOFTWARE..... | 4 |
| 1.1. <i>KINECT</i> | 4 |
| 1.2. <i>LABVIEW</i> | 5 |
| 2. APLICACIONES..... | 6 |
| 2.1. <i>ESQUEMA DE INICIALIZACIÓN</i> | 6 |
| 2.2. <i>INICIO DE UNITY PRO</i> | 7 |
| 2.3. <i>INICIO DEL SERVIDOR OPC</i> | 7 |
| 2.4. <i>INICIO DE LABVIEW</i> | 7 |

ÍNDICE DE FIGURAS

| | |
|---|----|
| Figura 1. Esquema del proceso. | 6 |
| Figura 2. Panel frontal. Pestaña de inicio..... | 8 |
| Figura 3. Barra de herramientas | 8 |
| Figura 4. Pestaña de inicialización..... | 10 |
| Figura 5. Pestaña de coordenadas. | 11 |
| Figura 6. Pestaña de pulsos robot..... | 13 |

ÍNDICE DE TABLAS

| | |
|--|---|
| Tabla 1. Requerimientos de hardware de LabVIEW. | 5 |
|--|---|

1. REQUISITOS DE HARDWARE Y SOFTWARE

El objeto de este apartado es mostrar las especificaciones técnicas que requieren los distintos dispositivos que conforman el trabajo, así como los requerimientos de los programas utilizados para programarlos.

1.1. KINECT

Para el correcto funcionamiento de la cámara infrarroja (kinect), el PC debe de cumplir los siguientes requisitos mínimos:

I. HARDWARE¹

- ✓ Ordenador con doble núcleo de 3.1 GHz o superior.
- ✓ Procesador de 64-bit (x64).
- ✓ 4 GB de RAM.
- ✓ Tarjeta gráfica que soporte DirectX 11.

II. SOFTWARE (elementos descargados para el manejo de Kinect)

- ✓ Kinect for Windows SDK v1.7
- ✓ Kinect for Windows v1.7 Developer Toolkit

¹ Datos basados de la página oficial de Microsoft: https://www.microsoft.com/en-us/kinectforwindows/purchase/sensor_setup.aspx

1.2. LABVIEW

Los requerimientos mínimos de Hardware² para LabVIEW, son lo que se muestran en la tabla siguiente:

Tabla 1. Requerimientos de hardware de LabVIEW.

| ENTORNO DE DESARROLLO DE WINDOWS | |
|----------------------------------|---|
| PROCESADOR | Pentium 4/M o equivalente |
| RAM | 1GB |
| RESOLUCIÓN DE PANTALLA | 1024 x 768 píxeles |
| SO | Windows 8.1/8/7/Vista (32 bits y 64 bits) Windows XP SP3 (32 bits) Windows Server 2012 R2 (64 bits) Windows Server 2008 R2 (64 bits) Windows Server 2003 R2 (32 bits) |
| ESPACIO EN DISCO | 5 GB |

² Datos basados de la página web de NI: <http://www.ni.com/labview/requirements/esa/>

2. APLICACIONES

En este apartado se pretende explicar cómo debe el usuario manejar las aplicaciones para inicializar el proceso de telecontrol y revisar su correcto funcionamiento.

2.1. ESQUEMA DE INICIALIZACIÓN

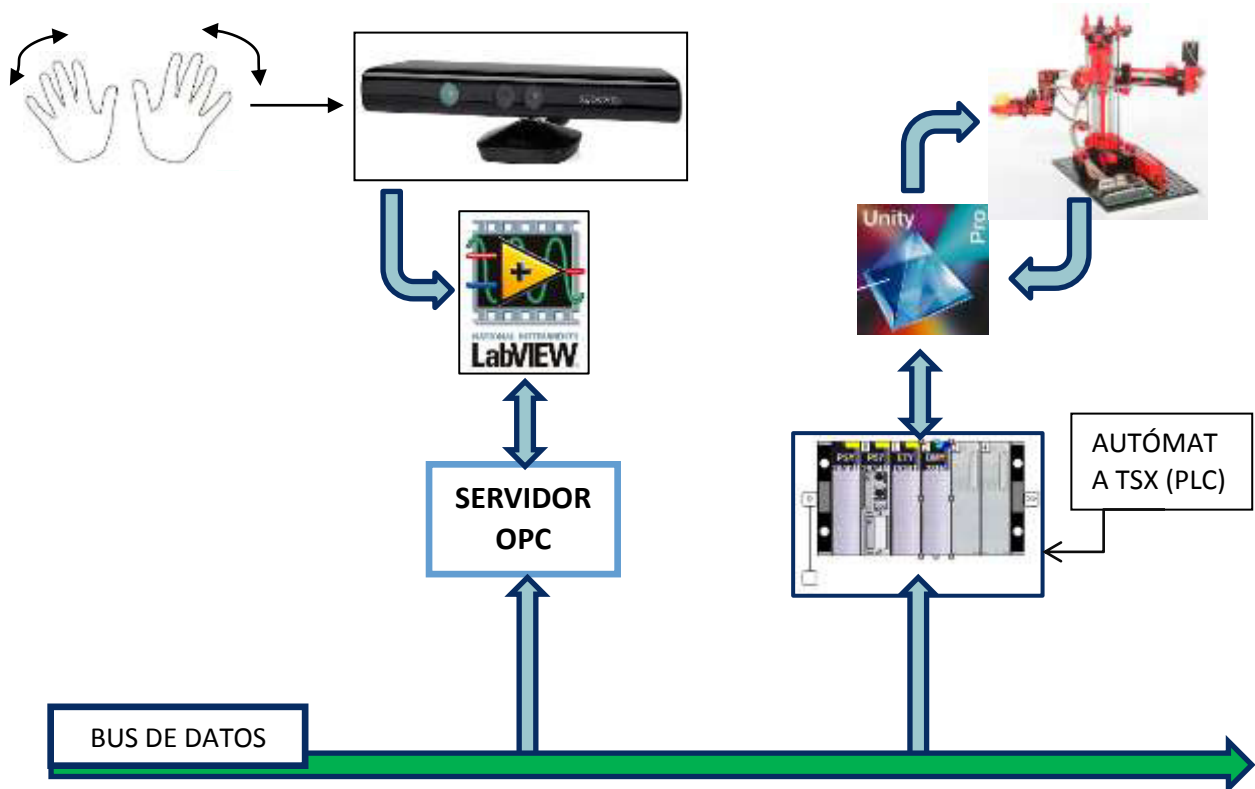


Figura 1. Esquema del proceso.

Como se observa en la figura, se deben de inicializar tres partes distintas del proceso, es decir, LabVIEW, Unity PRO y el servidor OPC deben de estar funcionando simultáneamente.

2.2. INICIO DE UNITY PRO

Se abre el programa referente del brazo robot desde Unity PRO. Seguidamente, se comprueba que la dirección IP es la correspondiente al autómata que se va a emplear. Para ello, en la barra de herramientas, se hace click en PLC y se elige la opción 'Establecer dirección'. De la pantalla emergente, se comprueba que la dirección IP es correcta y se pulsa 'aceptar'.

Se pulsa el botón de compilar para ejecutar el programa y transmitirlo al autómata.

2.3. INICIO DEL SERVIDOR OPC

Se abre el programa KEPServerEX. En la pantalla asociada a dicho servidor se debe de abrir el archivo "BRAZO ROBOT". Se debe verificar que dentro de dicho archivo se encuentran asociados dos 'Devices' llamados LABVIEW y UNITY PRO.

Antes de comenzar, se debe de comprobar que la dirección ID de ambos dispositivos coincide. Para ello se hace click derecho en "BRAZO ROBOT" → 'Properties' se accede al cuadro donde puede ser modificada dicha dirección.

2.4. INICIO DE LABVIEW

Se debe de abrir 'PROGRAMA PRINCIPAL' en LabVIEW. Aparecerá en el panel frontal un 'tag Control' dividido en diferentes pestañas (Figura 2).

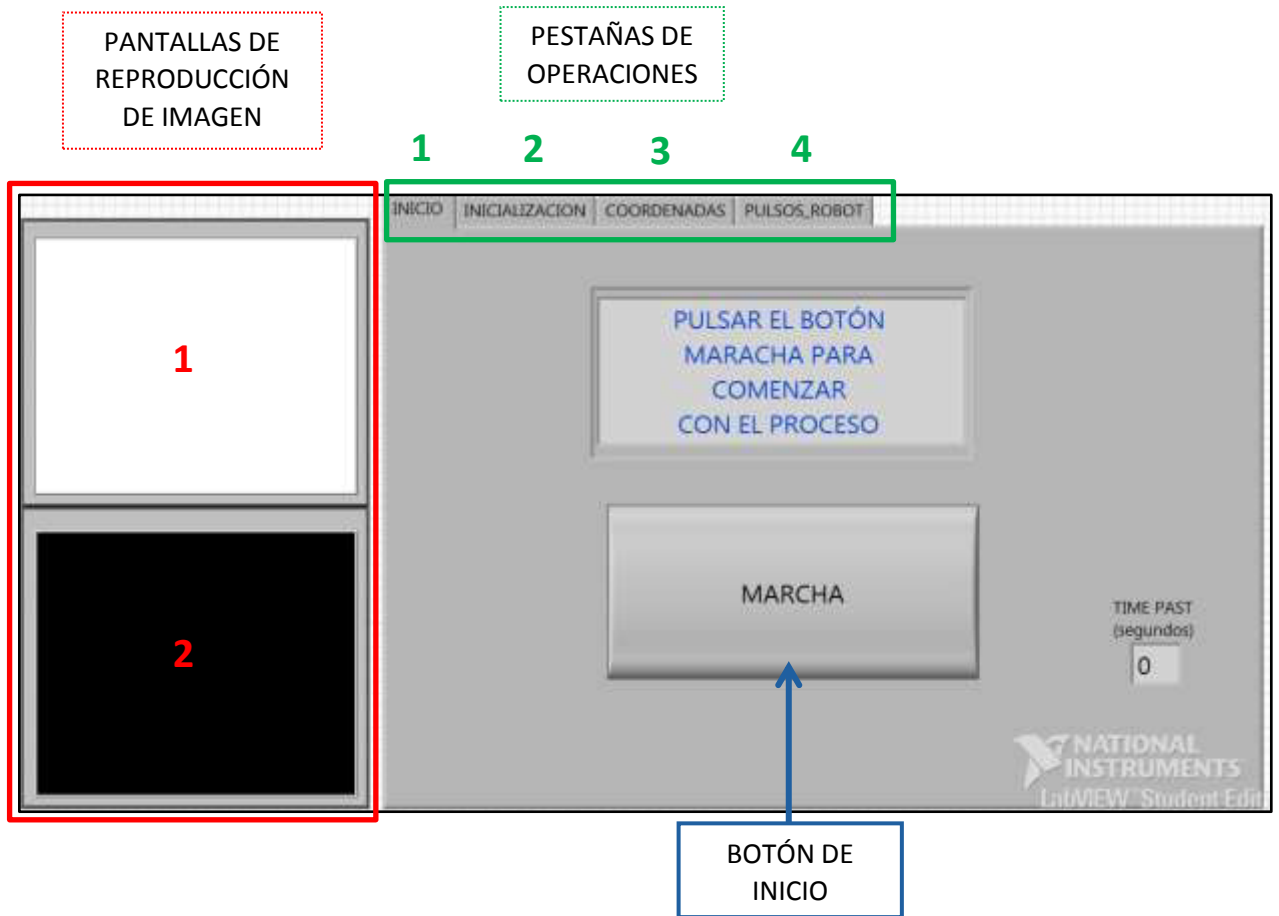


Figura 2. Panel frontal. Pestaña de inicio

En la barra de herramientas, se encuentran los comandos con los que se controla la programación (Figura 3).

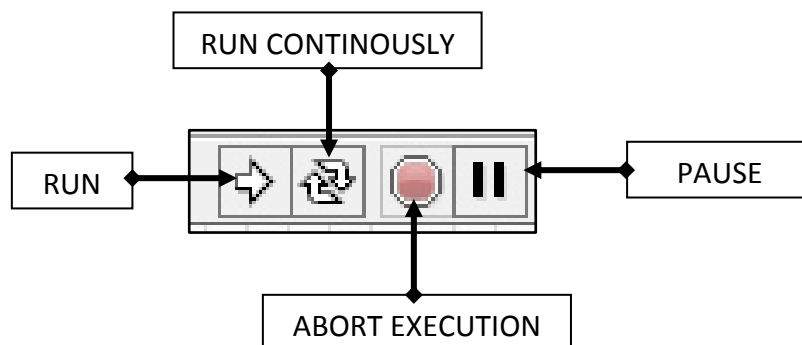


Figura 3. Barra de herramientas

Para comenzar, se debe de pulsar el botón “RUN”.

2.4.1. PANTALLAS DE REPRODUCCIÓN DE IMAGEN

La captura de video que lee kinect, queda marcada con el número 1, de forma que el usuario pueda visualizar la imagen que está recibiendo kinect en tiempo real.

En la pantalla marcada con el número 2, se muestra un modelado en 3D del esqueleto del usuario, en el que pueden diferenciar los distintos nodos que lo conforman (hasta un máximo de 20 nodos).

2.4.2. PESTAÑAS DE OPERACIONES

1. PESTAÑA DE INICIO

La función de esta pestaña es accionar el proceso de lectura de datos. Para ello se dispone de un botón llamado “MARCHA” que tras ser pulsado, se inicia el proceso.

En la esquina inferior derecha, se muestra un contador que indica el tiempo transcurrido desde que se inicia el programa hasta que el botón de “MARCHA” es accionado (*Figura 2*).

2. PESTAÑA DE INICIALIZACIÓN

Esta pestaña (*Figura 4*) es accionada automáticamente después de pulsar el botón de “MARCHA”.

Está dividida en dos secciones:

- En la parte superior se muestra los pasos a seguir. Para ello se ha utilizado una imagen explicativa en la que aparece las posiciones que debe de seguir el usuario. Además hay una pantalla de texto, en la que se especifican dichos pasos.

A medida que son completados, se ilumina un LED y suena un ‘beep’ para indicar que el proceso ha sido concluido.

- En la parte inferior se reflejan los valores de los máximos y los mínimos capturados según la posición de las manos del usuario.

En caso de que alguna lectura sea errónea o quiera modificarse algún valor, se debe pulsar el botón “Reset”.

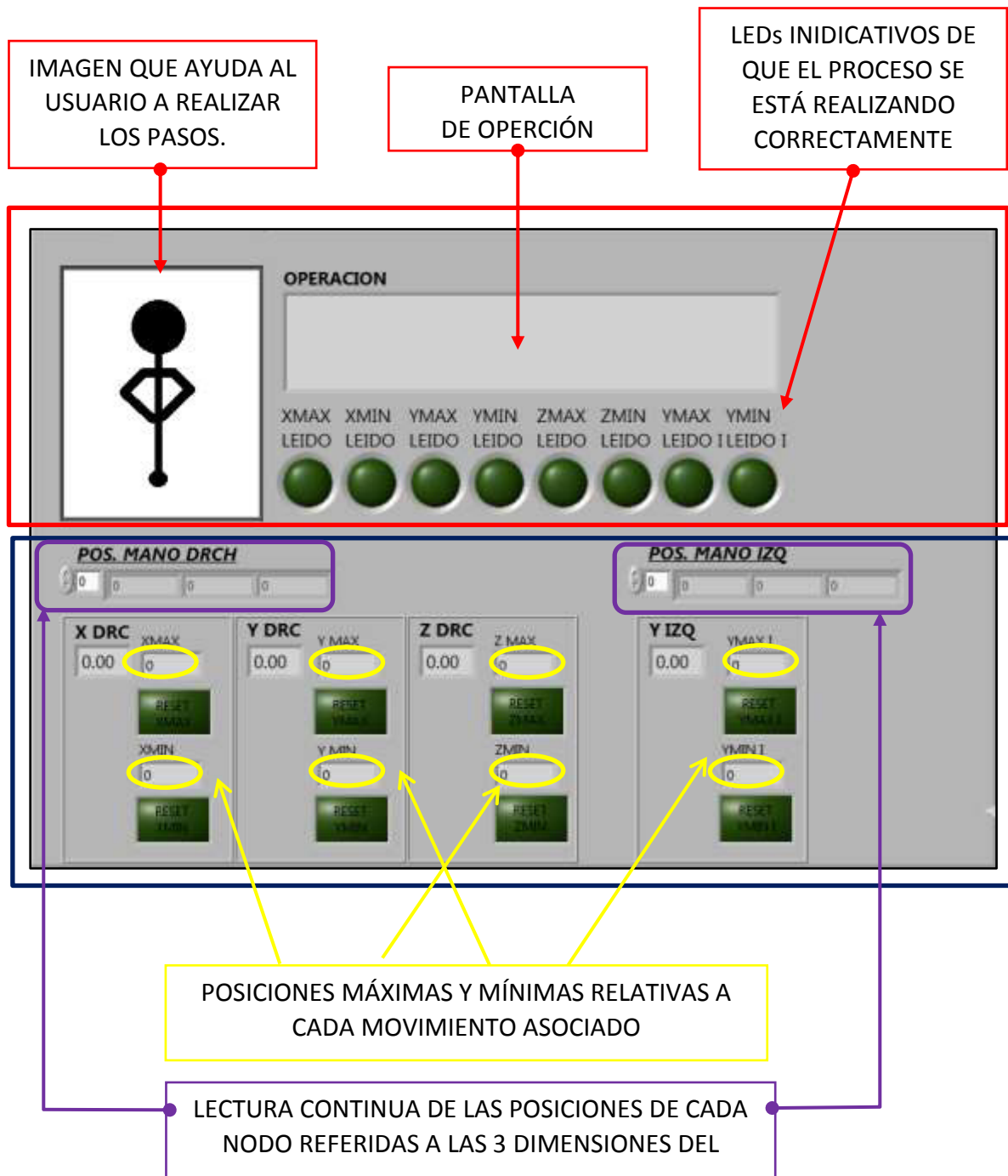


Figura 4. Pestaña de inicialización.

Una vez completado el proceso de lectura de datos y captura de rango de valores, personalizados para cada usuario, se sucede a la siguiente pestaña automáticamente.

3. PESTAÑA COORDENADAS

En la siguiente figura (Figura 5) se muestra la pestaña de coordenadas. En ésta se puede observar dos partes diferenciadas.

En primer lugar se distinguen una serie de geometrías encargadas de mostrar los valores de los movimientos del usuario dentro de un rango de 0 a 100 %. Se ha utilizado dos dispositivos de relleno horizontal para representar el movimiento de pinzado y otro del mismo tipo para representar el movimiento en el eje horizontal. Adicionalmente se ha utilizado un tanque para representar el movimiento vertical y un 'knob' para representar la rotación.

En segundo lugar, a la derecha de la pestaña, se muestra un modelado en 3D del brazo robot cuyo objetivo es representar los movimientos reales de éste.

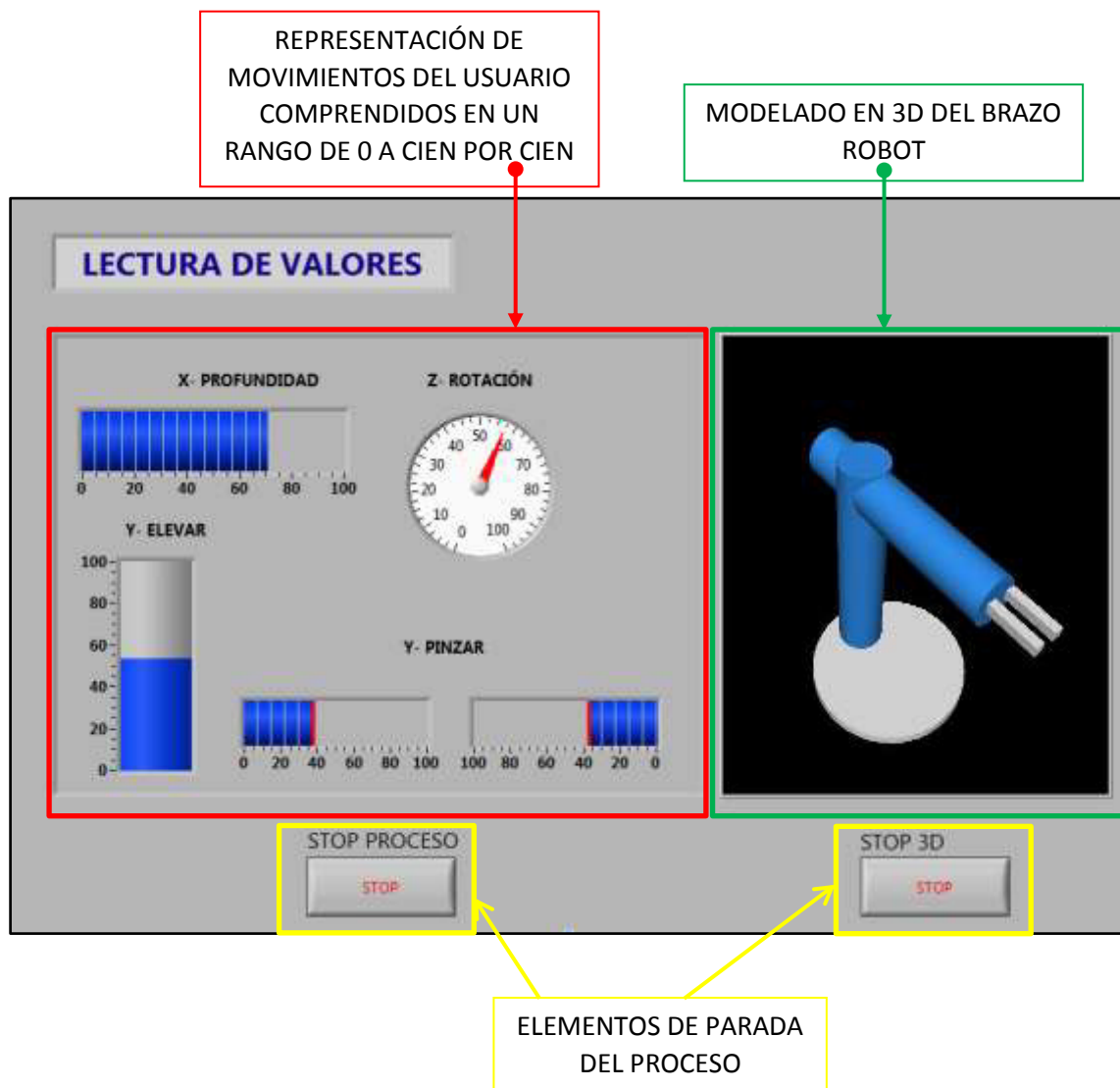


Figura 5. Pestaña de coordenadas.

Finalmente en la parte inferior, se han añadido dos botones de 'STOP' cuyo objetivo es detener el proceso en caso de fallo o errores inesperados.

4. PESTAÑA PULSOS ROBOT

Esta pestaña ha sido creada con objeto de definir los parámetros que rigen los movimientos del robot, concretamente, los pulsos (Figura 6).

Si se cambia el dispositivo mecánico, dichos pulsos (marcados en un cuadrado azul), deben de ser modificados de acorde a las nuevas especificaciones de éste.

Marcado en la figura adjunta con un cuadrado rojo, se encuentran los desplazamientos (en unidades de pulso) que son transmitidos al robot.

Una vez el programa ha sido puesto en marcha, debe de comprobarse que dichos pulsos pertenecen al intervalo establecido según las limitaciones del dispositivo mecánico. Por ejemplo: los valores mínimos para el desplazamiento corresponden con un pulso de cero, es decir, cuando los finales de carrera están activados. A su vez, los valores máximos quedan limitados por las especificaciones mecánicas del dispositivo. Así pues, los valores de referencia transmitidos deben de encontrarse entre el máximo y mínimo establecido.

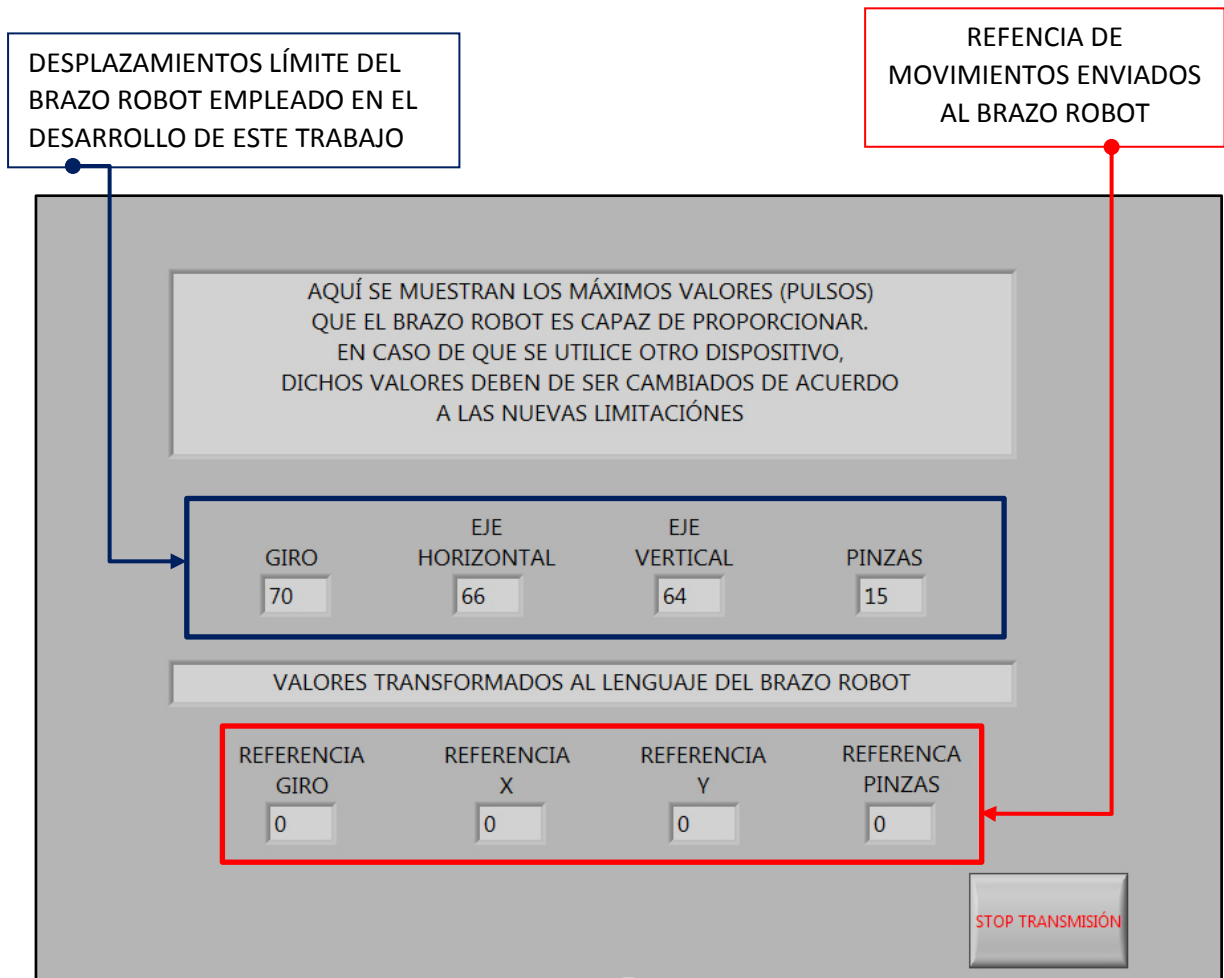


Figura 6. Pestaña de pulsos robot.

En la parte inferior derecha, está situado el botón para parar la transmisión de datos.

