UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

MÁSTER UNIVERSITARIO EN INGENIERÍA MECATRÓNICA

CONTROL DE ROBOT DE DOBLE BRAZO MEDIANTE ESTRUCTURA CORPORAL SENSORIZADA

DIRECTOR: CARLOS RICOLFE VIALA
CODIRECTOR: JUAN FRANCISCO BLANES NOGUERA

AUTOR: EDUARD CONESA GUERRERO

SEPTIEMBRE 2019

AGRADECIMIENTOS

En estas líneas querría mencionar la inestimable ayuda de todas las personas que han prestado apoyo a la realización de este proyecto. En primer lugar, a mis tutores Carlos Ricolfe y Juan Francisco Blanes por toda la ayuda y recursos prestados a lo largo del proyecto, a Ángel Valera por su asesoramiento en cuanto lo he necesitado, a Miguel Albero por toda la ayuda necesaria en cuanto a lo que supone el desarrollo de la electrónica del proyecto y demás compañeros de trabajo del Instituto de Automática e Informática Industrian (Ai2), entre ellos a Carlos Herrero y Jahel Carmona por la ayuda prestada en cuanto a programación.

También agradecer a mis compañeros de máster y de piso su apoyo, sean de donde sean, con especial mención a Carlos, Roi, Fernando, Eugenio, Jordi y Javi. Finalmente agradecer también a mi familia el apoyo prestado, especialmente a mi padre que siempre muestra un gran interés por los proyectos que he desempeñado.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. RESUMEN

A lo largo de este trabajo, se desarrollan los diferentes sistemas y componentes para un sistema de control de un robot de doble brazo, el IRB14000 YuMi de ABB, un robot colaborativo de alta precisión.

En primer lugar, se obtiene un modelo del funcionamiento de un brazo humano para posteriormente realizar la lectura de los grados de libertad que intervienen en su posicionamiento.

A continuación, se desarrolla un sensor de bajo coste y de precisión inferior a 10º para la lectura de cada articulación y se implementa el código correspondiente en el microcontrolador mediante una máquina de estados.

Posteriormente los valores son procesados siguiendo los dos modos de funcionamiento del posicionamiento, consistentes en posicionamiento por giro de articulación, donde se transmiten los valores de cada articulación a la articulación correspondiente en el robot, y el posicionamiento en ejes cartesianos, donde mediante la matriz de transformación de Denavit-Hartenberg se obtienen los valores de la situación de la palma de la mano en coordenadas X, Y y Z del sistema del robot.

Se diseña una estructura sensorizada que contenga toda la electrónica y se asemeje a la geometría de un brazo para su correcta integración

Finalmente se realiza una comunicación inalámbrica mediante sockets TCP/IP con el robot, facilitando la portabilidad del sistema sensorizado y facilitando la teleoperación.

Asimismo, también se realiza un simple control por fuerza de los manipuladores del robot, permitiendo una experiencia de teleoperación completa con el robot.

Todo el proyecto se implementará en el lenguaje C++ para el microcontrolador utilizado para sensorización y comunicación con sistema LINUX embebido y en RAPID para el controlador de ambos brazos del YUMI.

A lo largo de este proyecto también se diseñará la electrónica necesaria para facilitar la conexión de los componentes, así como también se manufacturarán las piezas de la estructura para verificar su correcta integración con el usuario.

Este dispositivo también es diseñado con la futura posibilidad de uso de este como herramienta de programación del propio robot.

La realización del proyecto se realiza de manera correcta ya que se comprueba cada sistema de manera individual, garantizando una correcta integración del funcionamiento colectivo de todos estos.

Palabras clave: robot de doble brazo, robot colaborativo, estructura sensorizada, teleoperación.

1.2. MOTIVACIÓN

Con la llegada del reciente 5G y la notable mejora en la latencia de los sistemas de transmisión, la teleoperación de sistemas ha recibido un notable impulso. El control de sistemas médicos para operar a distancia, control en situaciones de riesgo para personas como entornos nocivos de fabricas solo son el comienzo de muchas de las posibles aplicaciones que ahora están al alcance y posibilitan su uso.

También con la conocida llegada industria 4.0 se incrementa el uso de sistemas robotizados y de personal especializado en este sector.

Es por ello que un sistema que actúe como interfaz entre persona y robot adquiere una notable importancia, ya que además de permitir una fácil teleoperación de sistemas robotizados, también supone una herramienta de programación intuitiva para personal no especializado.

1.3. OBJETIVOS

Para el control del sistema robotizado propuesto por un IRB14000 de doble brazo mediante estructura sensorizada, se pretende desarrollar todos los sistemas necesarios que intervienen en dicha herramienta. Por tanto, los objetivos de este proyecto son:

- Obtener un modelo fidedigno del comportamiento de un brazo humano fácilmente sensorizable.
- Desarrollo y manufactura de un sensor de bajo coste y alta precisión, por debajo de los 10º.
- Elección del microcontrolador para realizar la lectura y comunicaciones de los sistemas.
- Implementación del código necesario para leer los sensores y realizar la comunicación, tanto en el microcontrolador como en el robot elegido.
- Desarrollo de toda la electrónica necesaria para la lectura de los sensores propuestos.
- Diseño y manufactura de una estructura adaptable a la fisonomía del brazo humano y que recoja los giros de las diferentes partes de este.
- Calculo de posicionamiento del TCP del robot en función de la configuración del brazo, ya sea por configuración de cada articulación del brazo o por la configuración del posicionamiento en ejes cartesianos.

2. ESTADO DEL ARTE

Desde los inicios, el ser humano ya poseía la necesidad de alcanzar lugares más lejanos de los que su cuerpo podía llegar, ya sea para alcanzar fuentes de comida como frutos u otros recursos. Los sistemas mas rudimentarios son considerados el uso de palos como extensión del brazo por tal de alcanzar dichos frutos. Es así como a lo largo de la historia se han ido diseñando dispositivos más complejos, pasando por la pértiga, así como también sistemas de pinzas, que permiten llegar a distancia más lejanas o bien interactuar con objetos sometidos al calor, hasta llegar finalmente a la actualidad en la que se puede definir una nueva herramienta conocida como robot [Nuño 08].

Todos estos mecanismos ofrecen un sistema de comportamiento basado en una arquitectura maestro-esclavo, en la que el esclavo realiza movimientos enviados por el maestro, que puede ser un controlador joystick, otro sistema robot sobre el que actúa un esfuerzo humano u otro mecanismo similar.

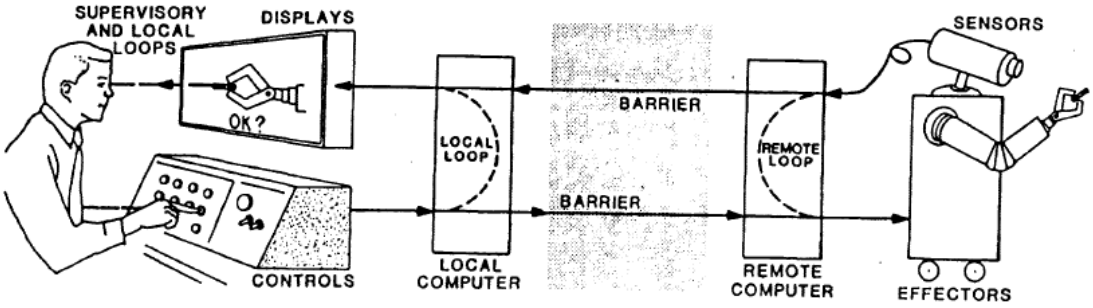


Ilustración 1. Sistema maestro-esclavo de teleoperación (Nuño 08)

La teleoperación cobra sentido en aquellos procesos en los que, debido a la complejidad de estos y al tipo de respuesta dinámico del sistema supone una difícil automatización. Es por ello por lo que la intervención de una persona es requerida por el sistema.

La telerrobótica es la aplicación de las tecnologías de teleoperación aplicadas al campo de la robótica, donde también es notable el uso de tecnologías de telepresencia por tal de tener una "sensación" de la interacción con el entorno. Para ello, estas tecnologías pretenden además monitorizar el comportamiento del robot y reproducir dichas interacciones en el sistema del control maestro. El sistema más simple de telepresencia puede consistir en una única cámara para visualizar el entorno de trabajo del robot. Posteriormente, con la llegada de la realidad virtual, se han añadido sistemas de mayor complejidad que permiten una experiencia más fidedigna en la teleoperación [Nuño 08]. Algunos de estos sistemas consisten en:

- Retroalimentación táctil: donde mediante actuadores se intenta obtener una sensación de contacto aplicada sobre la piel.
- Retroalimentación cinestésica o de fuerzas: se impone una resistencia al avance o un peso de referencia para simular las cargas o esfuerzos que actúan sobre el robot
- Realimentación háptica: retroalimentación de contacto ya sea por fuerzas o táctil.

Pero previamente a describir más profundamente el concepto y las modalidades de teleoperación existentes, es necesario introducir el concepto del robot, ya que es una de las herramientas fundamentales en las que se basa este proyecto.

La palabra robot no tiene su primera aparición hasta en 1920, donde nace el concepto de robota, es decir, trabajo tedioso. No es hasta 1938, que los americanos Willard Pollard y

Harold Roselund fabrican la primera máquina para pintar spray, que prescinde de la actuación humana mediante un sistema rudimentario de ejes sobre guías para proceder al movimiento de la pistola de pintura a lo largo de la carrocería del vehículo.

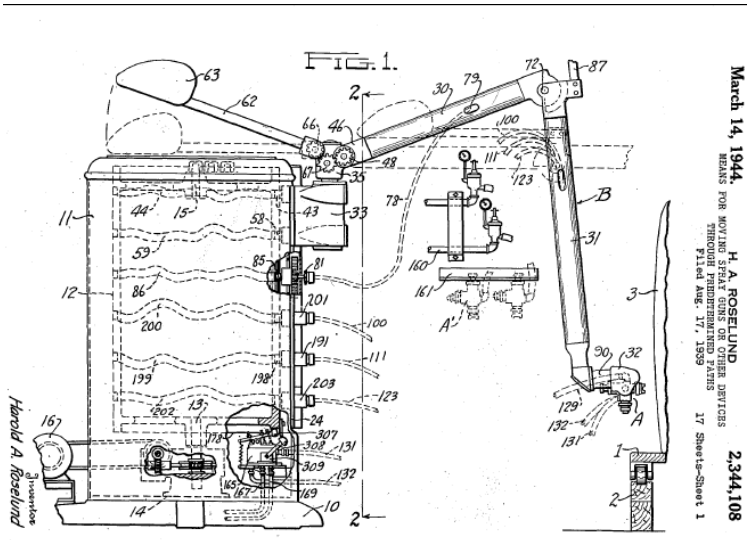


Ilustración 2.Diagrama de la máquina de pintar automatizada. (Spray-pain robot patents)

Sin embargo, los brazos robots no tienen aparición hasta 1951. Raymond Goertz había iniciado previamente investigaciones en Argonne National Laboratory, en Estados Unidos, para el desarrollo de un manipulador a distancia mediante el gobierno de otro manipulador exactamente idéntico, para actuaciones en ámbitos de radiación nuclear. Nace así el primer brazo mecánico controlado a distancia con arquitectura maestro-esclavo, con unas pinzas en su extremo que reproducían de manera casi idéntica el comportamiento de las pinzas del sistema maestro.

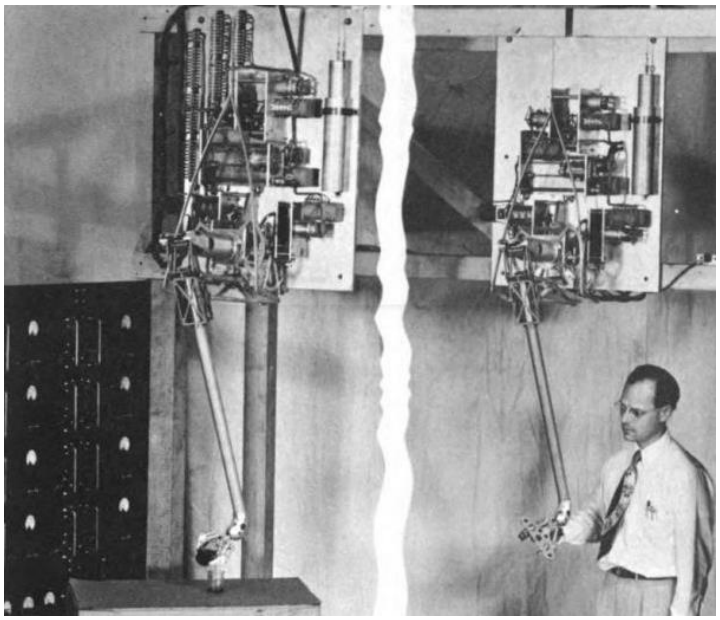


Ilustración 3. Sistema de maestro esclavo de Goertz (Argonne National Laboratory)

En 1954, George Devol diseña el primer robot programable comercial, que no se comercializa hasta en 1961, “Unimate”, instalado en General Motors, aunque previamente hace aparición el robot “Versatran”, desarrollado por American Machine & Foundry Company.

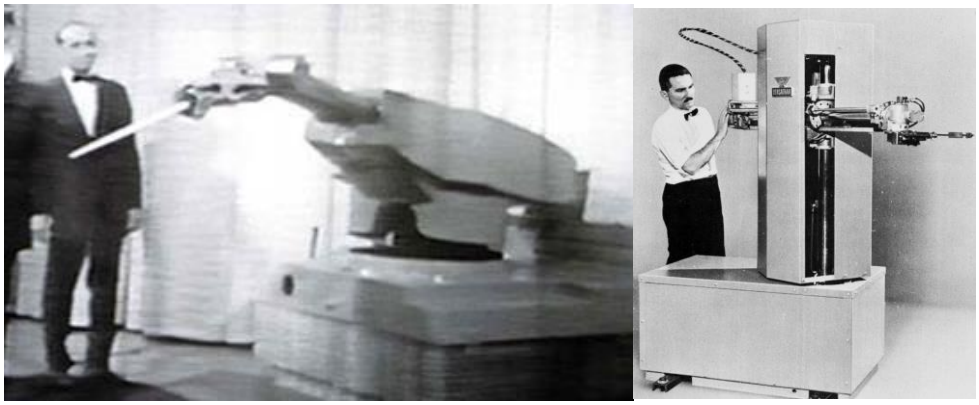


Ilustración 4. A la izquierda, el robot Unimate. A la derecha el robot Versatran. (robotics.org)

Posteriormente hacen aparición los primeros robots controlados por un miniordenador en 1973, como el “T3”, aunque cabe destacar que el peso del ordenador era de unos 30kg.

En 1976, hace aparición “Viking II” el robot enviado por la NASA a marte, que incluía un brazo articulado.

Es a partir del 1978, que se inicia una carrera del desarrollo de todo tipo de robots industriales. De manera simultánea también se inician investigaciones en robots humanoides, dando lugar a los primeros robots humanoides en 1997, donde el proyecto iniciado por HONDA del robot P3 en 1986 sale a la luz con unos resultados increíbles en comparación al del resto de sus competidores.

Ilustración 5. Robot P3 de Honda (Wikipedia)

A continuación, hace aparición “Qrio” de SONY en 2003, un robot bípedo capaz de reconocer voces y rostros, pero lo más novedoso, es que es el primer robot autónomo capaz de correr, llegando a una velocidad de 23 cm/s.

Ilustración 6. Robot Qryo (SONY)

Finalmente, en 2008, hacen aparición los cobots o robots colaborativos, robots capaces de realizar tareas en coordinación con humanos sin que estos padezcan peligro alguno. Es entonces cuando Universal robots, realiza la venta del primer robot colaborativo industrial de la historia. Actualmente es una de las empresas líder del sector.

Ilustración 7. Producción de los primeros UR (Universal Robots)

En este mismo contexto, aparece YUMI, el robot de ABB colaborativo que pose dos brazos capaces de levantar 0.5kg cada uno (sin contar el peso del propio manipulador) y en el cual se basa el desarrollo de este proyecto.

Ilustración 8. Robot colaborativo YUMI (ABB)

Es gracias a la necesidad de teleoperar este tipo de robots, que la teleoperación cobra una importancia vital, teniendo en cuenta también la aparición de la actual tecnología 5G, que permite tasas de transferencia de datos con una latencia ínfima en comparación con la tecnología 4G, que permite la teleoperación casi instantánea.

La posibilidad de disponer de este recurso de teleoperación permite el uso de esta tecnología para algunas aplicaciones como:

- Submarinas: en tareas de inspección, mantenimiento construcción de instalaciones submarinas, donde las profundidades o peligros que atañen al submarinista imposibilitan su participación.
- Espacio: la utilización de estos sistemas es necesaria en materia de seguridad, duración de las tareas y el coste necesario que conlleva la realización de estas misiones.
- Industria nuclear: ya con el suceso de la catástrofe de Chernobyl, se intento el uso de robots para estas tareas. Aunque en el caso anterior la misión no fue fructífera, el uso de esta tecnología ha ido progresando para realizar tareas de manipulación de sustancias radioactivas entre otras.
- Militares: en misiones de reconocimiento, por tal de no poner en peligro la vida de los soldados, se utilizan sistemas como los UAV, que son controlados a distancia por un operador.
- Medicas: las tareas de teleoperación no siempre implican una manipulación del sistema móvil desde grandes distancias. En la medicina, se hace uso de estas para tareas en las que es necesaria una precisión elevada.

3. EL ROBOT IRB 14000 YuMi

Para la realización de este proyecto se hace uso del modelo IRB14000 de ABB, un robot con dos brazos de 7 grados de libertad cada uno.

Cada brazo del robot es como un brazo independiente que posee su propio controlador, pudiendo realizar funciones totalmente independientes, aunque también pueden realizar tareas coordinadas de manera sencilla, ya que existen ordenes para sincronizar ambos controladores mediante simples instrucciones.

Además, este robot está capacitado según ABB para "manipular todo tipo de objetos, desde un reloj suizo hasta una tablet PC, y con un nivel de precisión capaz de ensartar una aguja" y "colaborar mano a mano con el ser humano en un entorno de fabricación normal, contribuyendo a que las empresas obtengan el mejor rendimiento de sus empleados y sus robots". A continuación, se muestran algunas de sus especificaciones más relevantes.

Caracteristica	Valor
Capacidad de manejo (kg)	0.5
Peso (kg)	38
Alcance(m)	0.559
Fuerza en x (N)	89
Fuerza en y (N)	147
Fuerza en y (N)	380
Par en x (Nm)	101
Par en y (Nm)	14
Par en z (Nm)	61
Distancia de apertura gripper (mm)	0-50
Velocidad máxima (mm/s)	25
Fuerza de agarre máxima (N)	20

Tabla 1.Especificaciones generals del IRB1400 (Especificaciones de producto ABB)

4. DISEÑO DE LA ESTRUCTURA SENSORIZADA

A continuación, se detalla el diseño de un prototipo de estructura capaz de recoger la posición en ejes cartesianos de la mano de una persona respecto de un sistema de coordenadas fijado situado inmediatamente debajo de su pecho, para lograr una similitud máxima a la del sistema de coordenadas base de robot IRB14000 de ABB (YUMI).

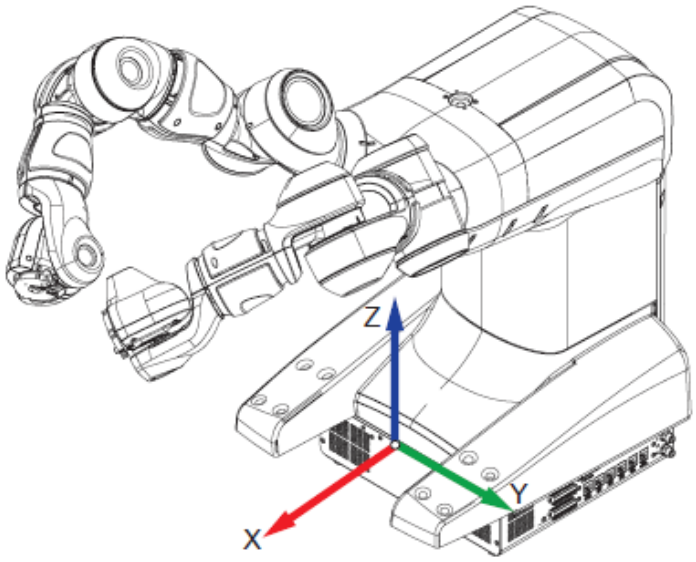


Ilustración 9. Sistema de coordenadas mundo del IRB14000 (Manual del operador IRB14000 ABB)

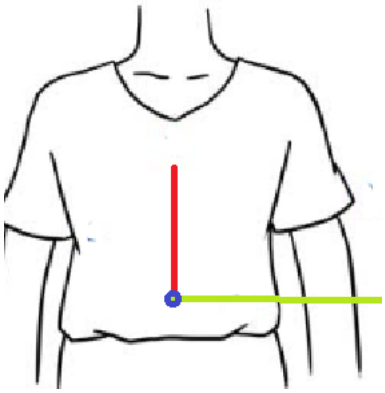


Ilustración 10. Sistema de coordenadas elegido para el usuario (propia)

Para ello se descompondrán las rotaciones de un brazo humano para posteriormente proceder a la elección de un sensor capaz de recoger dicha información y traducirla a un punto en el espacio en ejes cartesianos.

4.1. DESCOMPOSICIÓN DE ARTICULACIONES DE UN BRAZO HUMANO

Comúnmente, la mayoría de los robots disponen de 6 grados de libertad, 3 para la posición (X, Y, Z) y 3 para orientación según ángulos de Euler (RX, RY, RZ). Este funcionamiento permite alcanzar cualquier posicionamiento en el espacio de trabajo del robot con una orientación determinada. No obstante, esta posición es estática, con lo que no se puede realizar ningún cambio en la rotación de los elementos móviles del robot por tal de mantener dicha posición. Esto no sucede en el caso de un brazo humano, ya que dispone de más grados de libertad. El brazo humano se puede caracterizar como un modelo de 7 grados de libertad. [Díaz 14]

Asimismo, el IRB14000 dispone de dos brazos con 7 grados de libertad, lo que permite alcanzar posicionamientos dinámicos, es decir, que permiten una reorientación de los miembros del brazo sin cambiar la orientación.

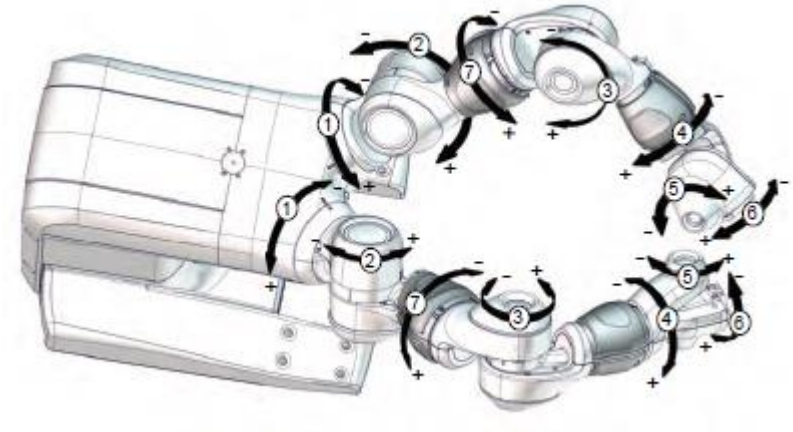


Ilustración 11. Ejes y direcciones de las articulaciones del YUMI (ABB)

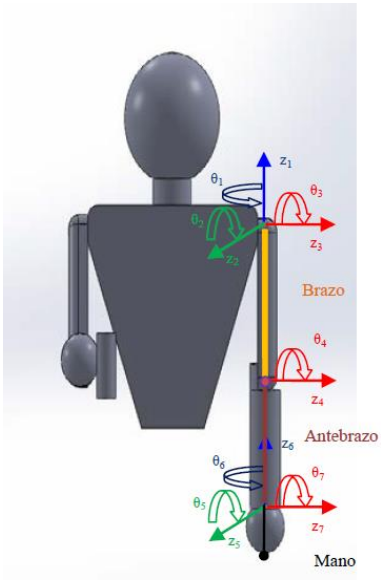


Ilustración 12. Ejes de un brazo humano (Díaz 14)

Como se muestra en la ilustración anterior, se ha diseñado un modelo del comportamiento de un brazo humano consistente en 2 grados de libertad en el hombre, 1 grado de libertad sobre el brazo, 1 grado de libertad sobre el codo, 1 grado de libertad sobre el antebrazo y 2 grados de libertad en la muñeca. Este modelo permite detectar la posición final de la mano mediante rotaciones a través de cada eje de cada articulación. Posteriormente se mostrará la selección de sistemas de coordenadas según el convenio de Denavit-Hartenberg y las matrices de transformación correspondientes para obtener el posicionamiento y la orientación de la mano.

4.2. DISEÑO Y ELECCIÓN DE LA SENSORIZACIÓN

Para conocer el posicionamiento de cada miembro del brazo es necesario conocer, como se ha dicho anteriormente, los grados de giro de cada articulación. más sencillos de utilizar y económicos son estos dos últimos.

Además, es necesario algún mecanismo para seleccionar los diferentes modos de funcionamiento del robot y realizar la apertura y cierre de los manipuladores del robot.

4.2.1. POSICIONAMIENTO

Hay gran cantidad de sensores que son capaces de realizar la lectura de posición angular de manera muy precisa pero su coste es elevado. Algunos de estos sensores son tales como resolvers, encoders y potenciómetros. Estos dos último son los más sencillos de leer y los más económicos, pero en primer lugar debemos conocer su precisión, ya que uno de los requerimientos es que la precisión de cada articulación este por debajo de los 10º.

En un potenciómetro se aprovecha el divisor de tensión generado por la variación de resistencia al girar el eje de este. El comportamiento del potenciómetro presenta una tendencia lineal ya que, según la ley de Ohm, la relación entre resistencia y tensión es lineal. Se realizan varias pruebas para verificar la precisión y repetitividad de este. Para ello se diseña una bancada de pruebas que contiene un nonio que nos da una precisión de 0.5º (consultar planos y código de la prueba en el anexo).

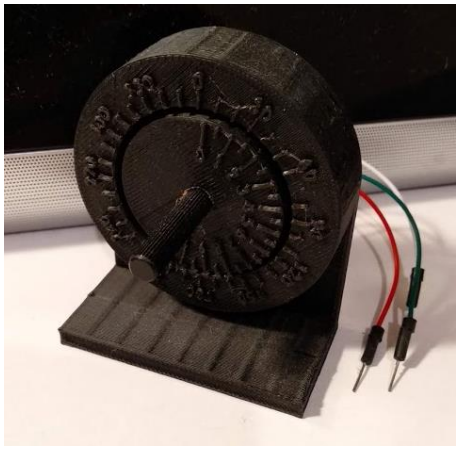


Ilustración 13. Nonio de prueba para resolución de potenciómetro (Fuente propia)

El microcontrolador seleccionado, del que se obtendrán más detalles posteriormente, dispone de 7 ADC para la lectura de valores analógicos de tensión entre 0 y 1.8V con una resolución de 12 bits, es decir, 4096 valores entre 0 y 1.8V. Idealmente se podría tener una resolución de 0.08º con este sistema, pero debido a imperfecciones mecánicas como holguras y la sensibilidad de este tipo de sensores al ruido eléctrico la precisión es bastante más baja. Se muestra a continuación los resultados obtenidos en los ensayos con el potenciómetro y la lectura del nonio mediante un modelo lineal.

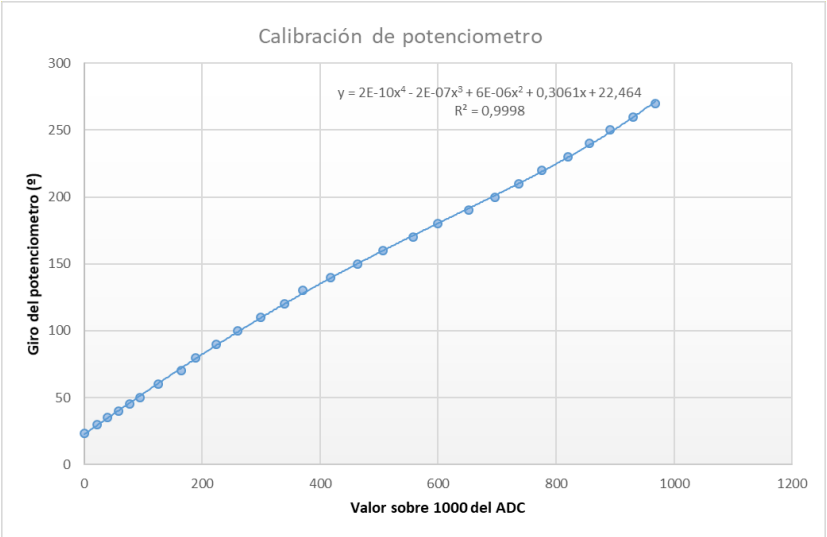


Ilustración 14. Ensayo de comportamiento de encoder

Esta precisión aun se puede ver incrementada si en los puntos intermedios se realizan interpolaciones lineales, pero este proceso incrementa el coste computacional y dado que la precisión es bastante elevada queda descartado.

A pesar del buen funcionamiento que se obtiene del potenciómetro en la prueba, se descarta el uso de este, ya que para realizar la lectura de ciertas articulaciones cuyo eje quedaría en el interior del propio brazo, sería necesario el diseño de complejos sistemas mecánicos. Además, sería necesario el uso de multiplexores para intercambiar la lectura de cada potenciómetro, al no disponer el controlador de más de 7 ADC.

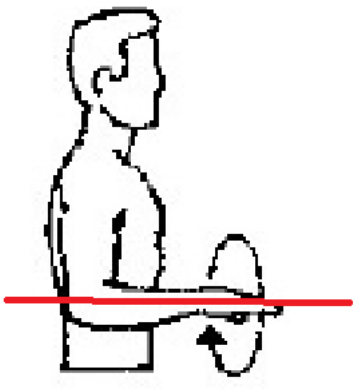


Ilustración 15. Ejemplo de eje contenido en el interior de un miembro (Fuente propia)

En el encoder se tienen pulsos digitales en diversos canales que permiten conocer el giro realizado y la dirección de su eje. No obstante, se busca una precisión igual o inferior a 10 º por articulación, lo que dificulta la adquisición de encoders comerciales, ya sea por su coste elevado, baja precisión en otros casos, o dimensiones excesivamente grandes. Es por ello por lo que se decide diseñar un encoder mediante el optoacoplador TCST2300, consistente en un diodo led como emisor y un fototransistor como receptor. Los cálculos de las resistencias necesarias para polarizar correctamente el emisor y el fototransistor se encuentran adjuntas en los anexos.

En primer lugar, se elige la tipología del encoder a diseñar, ya sea absoluto o incremental. El encoder absoluto requeriría de 6 optoacopladores para lograr una precisión inferior a 10º. Teniendo en cuenta las dimensiones de cada optoacoplador, se obtendría un disco demasiado voluminoso, por lo que se elige diseñar un encoder incremental con canal de puesta a 0.

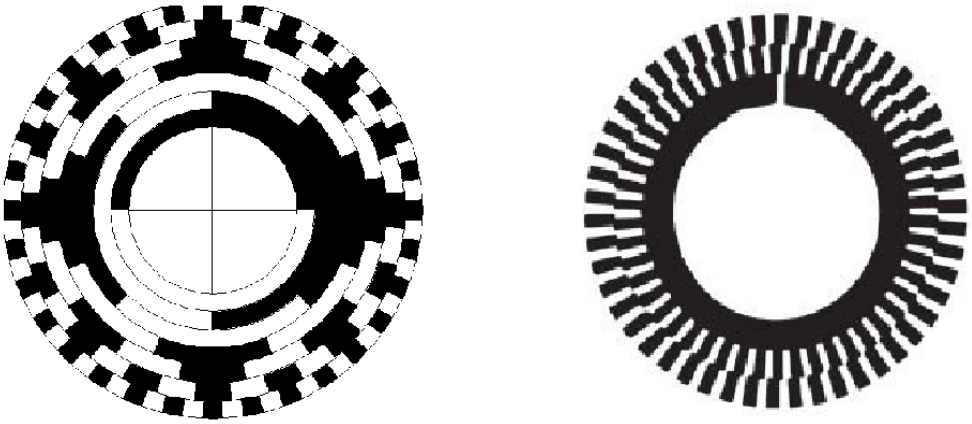


Ilustración 16. Comparación entre encoder absoluto (a la izquierda) y encoder incremental de 3 canales (a la derecha)(Fuente propia)

Se diseña un encoder de tipo incremental de tres canales. Es necesario el uso de tres canales, ya que con el canal A y el canal B se tienen las direcciones de y incrementos del encoder, mientras que el tercer canal se utiliza para realizar la puesta a 0 del encoder a un determinado valor. Se diseñan discos de diferentes graduaciones de orificios de 1,2,3,5 y 10 grados de precisión. Teniendo en cuenta las especificaciones del TCST2300, el tamaño del haz no es un problema para las aperturas de los orificios seleccionas. No obstante, dado que los discos de los encoders se manufacturarán en impresión 3D se realizan pruebas para conocer el acabado de estos y verificar su funcionamiento. Cabe destacar, que es necesario el uso de una imprimación de pintura para que el disco impreso en 3D sea capaz de obstaculizar el paso de la luz, que mediante este método de fabricación hay pequeñas imperfecciones en el disco que producen falsas lecturas.

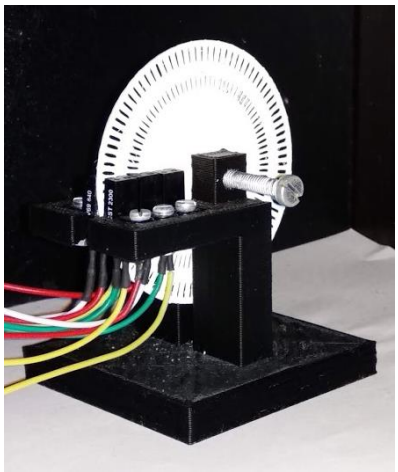


Ilustración 17. Prueba de encoder con disco impreso (Fuente propia)

Para verificar la calidad final de los discos de diferentes graduaciones se realizan pruebas en el montaje mostrado en la ilustración anterior cuyos planos están disponibles en el anexo.

0.0	0.1	1.1	1.0

Ilustración 18. Codificación gray para dos bits (Fuente propia)

La lectura del encoder se basa en una máquina de estados, facilitando así la posible pérdida de pasos del encoder. Se utiliza la codificación gray para los orificios del encoder, ya que al producir solamente cambios de un bit entre un estado y el siguiente proporciona mayor fiabilidad y detección de pérdidas de pasos.

El código utilizado para las pruebas esta disponible en los anexos. Se obtiene asimismo que la máxima precisión sin problemas de lectura en el encoder es de 3º.

La elección del sistema de encoder desarrollado permite la flexibilidad de su disposición en diferentes tipos de articulaciones a diferencia del uso de potenciómetros, facilitando mucho la detección de articulaciones cuyo eje está contenido en un miembro corporal, como la rotación del antebrazo respecto a su propio eje. Se hará uso de dos tipos diferentes de encoder: el encoder de disco y un encoder lineal curvo.

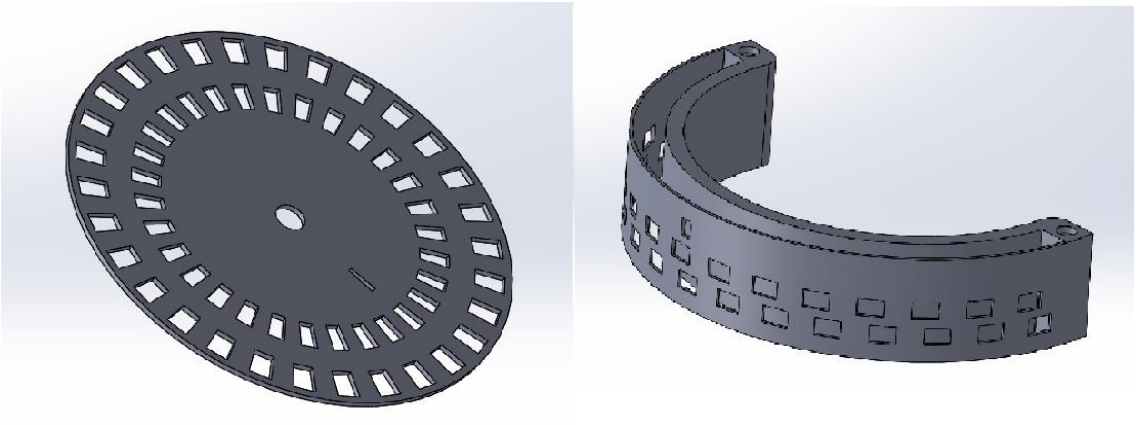


Ilustración 19.Encoder rotativo y encoder lineal curvo (Fuente propia)

El diseño de la bancada anterior permite utilizar el mismo tipo de cavidad donde se contienen la parte emisora y receptora del optoacoplador, siendo una parte fácilmente reproducible para cada articulación en la que la única variante es la banda o disco perforado. Cabe destacar que se ha mecanizado el encapsulado del encoder por tal de reducir sus dimensiones y mejorar su integración en la estructura que sensorizada

Foto cavidad que contiene optoacopladores

4.2.2. MANIPULADORES Y SELECCIÓN DE FUNCIONAMIENTO

El IRB14000 pude alojar gran multitud de herramientas en el extremo de sus dos brazos. En este proyecto se hará uso del IRB14000 Gripper, un manipulador que permite realizar agarres y aperturas por control de fuerza o por posición.

Ilustración 20. Gripper de IRB14000 (ABB)

El control de dicho Gripper se realiza con dos pulsadores situados en el extremo más cercano al pulgar de cada mano. Con una pulsación el manipulador cambia de cerrado a abierto y viceversa. Durante el cierre y la apertura de este, se realizará hasta cerrar la pinza con el valor de fuerza predefinido en cada manipulador mediante las ordenes g_GripIn y g_GripOut. Además, se dispone de un pulsador adicional en la mano derecha para controlar el comportamiento del movimiento si lo pulsamos durante 1-2 segundos, cambiando al funcionamiento de movimiento por articulación, donde cada eje transmite su valor al eje correspondiente en el IRB14000 o movimiento por posición en ejes cartesianos, donde se transmite la posición y orientación de cada brazo para que el robot la reproduzca independientemente de la orientación de cada articulación del brazo. Este pulsador también permite pausar el funcionamiento del robot y reanudarlo si se pulsa durante 5 segundos o más.

4.2.3. IMPLEMENTACIÓN DE CÓDIGO

Para la lectura de cada encoder se crea un hilo que implementa la máquina de estados comentada anteriormente. Las variables internas de cada hilo no pueden ser compartidas entre ellos. Por tal de poder realizar el envió de los valores leídos por los encoders, se creo un objeto compartido, al cual se accede para cambiar los valores de los contadores, activación de manipuladores y selección de modo y para realizar la lectura. Para evitar problemas de acceso a las variables compartidas entre hilos se utilizan la herramienta “MUTEX”, que bloquea el acceso a la variable para que no se corrompa ya sea por la lectura o escritura de un valor.

En primer lugar, se inicializan el hilo en tiempo real y los GPIO’s correspondiente a cada encoder como GPIO de entrada digital y activo a nivel bajo.

```
struct sched_param sp;
sp.sched_priority = 99;

GPIO pin_Home(GPIO::P8_36);
GPIO pin_ChA(GPIO::P8_37);
GPIO pin_ChB(GPIO::P8_38);
pin_ChA.setDirection(GPIO::INPUT);
pin_ChB.setDirection(GPIO::INPUT);
pin_Home.setDirection(GPIO::INPUT);
pin_ChA.setActiveLow();
pin_ChB.setActiveLow();
pin_Home.setActiveLow();
```

A continuación, se leen las entradas de cada GPIO y se realizan las comparaciones para seleccionar el estado y por tanto aumentar o disminuir el contador de posición una vez realizado previamente el homing del encoder. Al final del bucle se actualizan los valores de contador del objeto compartido.

```
while(1){
   estadoH = pin_Home.getValue();
   estadoA = pin_ChA.getValue();
   estadoB = pin_ChB.getValue();

  if ((estadoH == GPIO::HIGH && estadoA == GPIO::HIGH) && estadoB ==
GPIO::HIGH){
  homing = 1;
  pos_actual.home_7_der(true);
  estado = 0;
  cont = 0;
  }
  if (homing == 1){
    switch(estado){
        case 0:
            if (estadoA == GPIO::HIGH && estadoB == GPIO::LOW){
                cont=cont-3;
                estado = 3;
            }else if (estadoA == GPIO::LOW && estadoB == GPIO::HIGH){
                cont=cont+3;
                estado = 1;
            }else if((estadoA == GPIO::LOW && estadoB == GPIO::LOW)){
                homing = 0;
                pos_actual.home_7_der(false);
                cout << "Encoder 7 Derecho perdido, recalibra" << endl;
            }
            break;
        case 1:
            if (estadoA == GPIO::HIGH && estadoB == GPIO::HIGH){
                cont=cont-3;
                estado = 0;
            }else if (estadoA == GPIO::LOW && estadoB == GPIO::LOW){
                cont=cont+3;
                estado = 2;
            }else if((estadoA == GPIO::HIGH && estadoB == GPIO::LOW)){
                homing = 0;
                pos_actual.home_7_der(false);
                cout << "Encoder 7 Derecho perdido, recalibra" << endl;
                            }
            break;
        case 2:
            if (estadoA == GPIO::LOW && estadoB == GPIO::HIGH){
                cont=cont-3;
                estado = 1;
            }else if (estadoA == GPIO::HIGH && estadoB == GPIO::LOW){
                cont=cont+3;
                estado = 3;
```

```
                }else if((estadoA == GPIO::HIGH && estadoB == GPIO::HIGH)){
                        homing = 0;
                        pos_actual.home_7_der(false);
                        cout << "Encoder 7 Derecho perdido, recalibra" << endl;
                }
                break;
        case 3:
                if (estadoA == GPIO::LOW && estadoB == GPIO::LOW){
                        cont=cont-3;
                        estado = 2;
                }else if (estadoA == GPIO::HIGH && estadoB == GPIO::HIGH){
                        cont=cont+3;
                        estado = 0;
                }else if((estadoA == GPIO::LOW && estadoB == GPIO::HIGH)){
                        homing = 0;
                        pos_actual.home_7_der(false);
                        cout << "Encoder 7 Derecho perdido, recalibra" << endl;
                }
                break;
        }
        pos_actual.setCont_7_der(cont);
 }
  usleep(2500);
}
```

La implementación en máquina de estados permite la detección de salto de pasos, para recuperar el error volviendo a realizar el homing. La ejecución de cada hilo se realiza cada 2,5ms, es decir, con un muestreo de 400Hz, ya que se calcula que la velocidad de giro de una articulación es de unos 60Hz teniendo en cuenta que la velocidad máxima esperada a muestrear es de 180º/s y con una precisión de 3º por estado. Este valor de 400Hz permite muestrear perfectamente la señal de los encoders según el teorema de Nyquist, donde el mínimo para muestrear dicha señal debe de ser más del doble de frecuencia de esta. No se hace uso de interrupciones porque la BeagleBone no esta capacitada para tener 42 pines de interrupción, además el hecho de realizar un muestreo permite evitar el efecto rebote de la señal, ya que actúa como un filtro paso bajo.

Para verificar que los hilos en tiempo real se ejecutan correctamente, se realizan unas pruebas de código con el siguiente fragmento de código, donde se obtiene una estimación del tiempo de ejecución por vuelta.

```
struct timeval start, end;
unsigned int tact = 0;
gettimeofday(&start, NULL);

//Fragmento de código a medir

gettimeofday(&end, NULL);
tact = ((end.tv_sec  - start.tv_sec) * 1000000 + (end.tv_usec - start.tv_usec));
cout << "Tiempo de ejecución: " << tact << " us" << endl;
```

Para el control de los pulsadores se tienen diferentes hilos. De manera análoga al comportamiento de los encoders, se inicializan los GPIO como entrada digital activa a nivel bajo y el hilo en tiempo real. A continuación, se realiza la lectura de los pulsadores y se actualiza el valor de estos en el objeto compartido. Para activar el Gripper es necesario mantener pulsado el pulsador durante como mínimo 1s, ya que se muestrea el hilo cada 0.5s.

```
void *control_grippers(void *param){
        //Prioridad del hilo en tiempo real
        struct sched_param sp;
        sp.sched_priority = 99;
        GPIO pin_GripperR(GPIO::P8_39);
        pin_GripperR.setDirection(GPIO::INPUT);
        pin_GripperR.setActiveLow();
        GPIO pin_GripperL(GPIO::P8_41);
        pin_GripperL.setDirection(GPIO::INPUT);
        pin_GripperL.setActiveLow();

        GPIO::VALUE pulsador_R, pulsador_L;
        int actual = 0;
        cout << "++++++++++++++++++Grippers preparados++++++++++++++++" <<
endl;
        while(1){
                pulsador_R = pin_GripperR.getValue();
                pulsador_L = pin_GripperL.getValue();

                if (pulsador_R == GPIO::HIGH){
                        actual = pos_actual.getGripperR();
                        if (actual == 0){
                                actual = 1;
                        }else{
                                actual = 0;
                        }
                        pos_actual.setGripperR(actual);
                }

                if (pulsador_L == GPIO::HIGH){
                        actual = pos_actual.getGripperL();
                        if (actual == 0){
                                actual = 1;
                        }else{
                                actual = 0;
                        }
                        pos_actual.setGripperL(actual);
                }
        usleep(500000);
        }
}
```

El hilo del pulsador de selección de modo funciona de manera similar, cambiando de modo si se pulsa el pulsador entre 1 y 5s y pausando o reanudando la aplicación si se mantiene más de 5s.

```
void *seleccionar_modo(void *param){
	//Prioridad del hilo en tiempo real
	struct sched_param sp;
	sp.sched_priority = 99;
	GPIO pin_Modo(GPIO::P8_39);
	pin_Modo.setDirection(GPIO::INPUT);
	pin_Modo.setActiveLow();

	int modo = 0;
	GPIO::VALUE pulsador_Modo;
	struct timeval start, end;
	unsigned int tact = 0;

	while(1){
		pulsador_Modo = pin_Modo.getValue();
		if (pulsador_Modo == GPIO::HIGH){
			gettimeofday(&start, NULL);
			while(pulsador_Modo == GPIO::HIGH){
				pulsador_Modo = pin_Modo.getValue();
				usleep(500000);
			}
			gettimeofday(&end, NULL);
			tact = ((end.tv_sec  - start.tv_sec) * 1000000 + (end.tv_usec
- start.tv_usec));

			if (tact > 1000000 and tact < 5000000){
				if (modo == 0){
					modo = 1;
				}else{
					modo = 0;
				}
			}else if(tact > 5000000){
				modo = 2;
			}
			pos_actual.setModo(modo);
		}
		usleep(250000);
	}
}
```

4.3. DISEÑO Y ELECCIÓN DE LA ELECTRÓNICA

Para realizar toda la lectura de sensores y comunicaciones es necesario un microcontrolador con un mínimo de 42 entradas digitales y que permita una sencilla comunicación inalámbrica con el robot, además de una elevada capacidad de computo para realizar los cálculos de las matrices de transformación de sistemas de coordenadas. El controlador elegido y que cumple con las especificaciones es la BeagleBone Black, desarrollada por Texas Instruments. Dicho microcontrolador dispone de la versión ethernet y de una versión que sustituye dicho modulo por un módulo Wi-Fi. Este microprocesador contiene un sistema base LINUX para sistemas empotrados, lo que facilita mucho la implementación de los programas a desarrollar, tanto la comunicación como la lectura de las entradas digitales. Cabe destacar que la lectura de las entradas digitales se realiza mediante la lectura de un fichero en el que se van sobrescribiendo dichos valores, sin necesidad de acceder directamente a los registros específicos. Para configurar y acceder a las entradas digitales se hará uso de una librería para GPIO desarrollada por José Enrique Simó Ten. Dicha librería se encuentra adjunta en el anexo para más información de su funcionamiento.

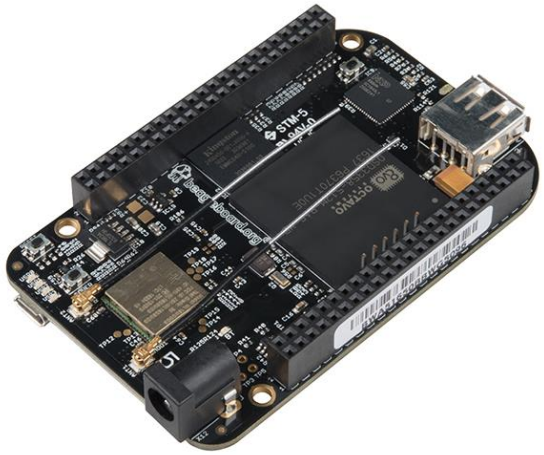


Ilustración 21. Microcontrolador Beaglebone Black Wireless (Sparkfun)

Se configuran los siguientes pines como entradas digitales para la lectura de todos los optoacopladores y los pulsadores de control de modalidad y control de los manipuladores.

Para simplificar e integrar todos los componentes necesarios para polarizar correctamente todos los sensores, se diseña una PCB que cumpla todas las necesidades requeridas.

Id	Identificador	Empaquetado	Cantidad	Identificación
1	J2	PinHeader_1x02_P2.54mm_Horizontal	1	Conn_01x02_Male
2	J4	PinHeader_2x11_P2.54mm_Vertical	1	Conn_02x11_Counter_Clockwise
3	R43-R84	R_1206_3216Metric_Pad1.42x1.75mm_HandSolder	42	Resistencia SMD 1/4W 100 Ohm
4	P2,P1	Socket_BeagleBone_Black	2	BeagleBone_Black_Header
5	J6,J3,J1	PinHeader_2x21_P2.54mm_Vertical	3	Conn_02x21_Counter_Clockwise
6	J5	PinHeader_2x10_P2.54mm_Vertical	1	Conn_02x10_Counter_Clockwise
7	J7	PinHeader_2x03_P2.54mm_Vertical	1	Conn_02x03_Counter_Clockwise
8	R1-R42,R85-R87	R_1206_3216Metric_Pad1.42x1.75mm_HandSolder	45	Resistencia SMD 1/4W 10 KOhm

Tabla 2. BOM de la PCB diseñada (Fuente propia)

Se realiza todo el esquema de conexiones, teniendo en cuenta que se disponen de 42 optoacopladores (3 por cada uno de las 14 articulaciones) y los pulsadores de control de las garras y la selección de modo. Todos estos sensores presentan el siguiente esquema eléctrico de funcionamiento.

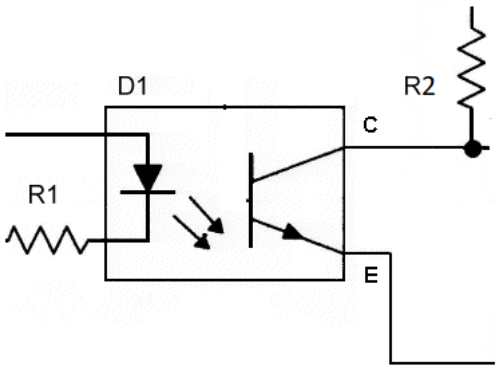


Ilustración 22. Esquema de funcionamiento de un optoacoplador de 4 patas (Fuente propia)

El pin digital de entrada de cada optoacoplador o pulsador funciona activamente a nivel bajo como se muestra en el esquema anterior. Como se ha mencionado anteriormente, los cálculos de los valores de las resistencias utilizadas se encuentran en los anexos.

A continuación, se realiza el calculo de los anchos de pista mínimo teniendo en cuenta los valores de espesor de cobre y las constantes necesarias según la fórmula:

$$Ancho(mils = (\frac{Intensidad\ (A)}{k1 \cdot \Delta T^{k2}})^{\frac{1}{K3}}$$

Donde k1, k2 y k3 son valores a seleccionar en función del tipo de PCB y ΔT es la variación máxima de temperatura entre la placa y el ambiente.

Finalmente se eligen los siguientes anchos para facilitar la manufactura de esta. Los cálculos se encuentran adjuntos en el anexo.

Pista	Intensidad(A)	Ancho(mils)	mm	Aprox	Recomend
General Conectores	0,01365	0,084815861	0,002078837	0,2	1
General Emisores	1,02	6,337888507	0,155341647	0,2	1
Colector	0,00065	0,004038851	9,89922E-05	0,2	0,2
Emisor	0,04875	0,302913789	0,007424417	0,2	0,2
Pin Digital	0,00065	0,004038851	9,89922E-05	0,2	0,2

Tabla 3. Valores de los anchos de pista calculados para a PCB

Se realiza entonces el “routing” de los componentes en dos capas.

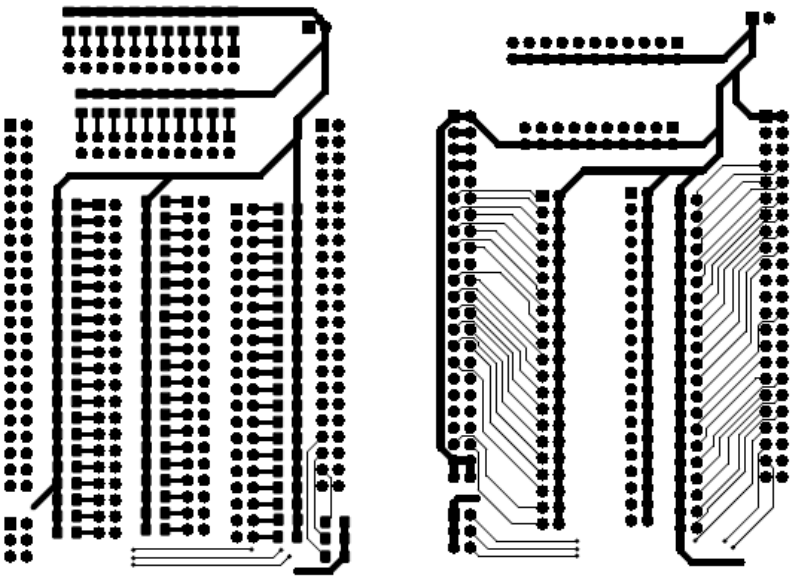


Ilustración 23. Routing de la PCB de la cara Top y la Bottom

Se generan los archivos Gerber necesarios para la manufactura de la PCB. Se puede observar a continuación un renderizado de la PCB final a manufacturar.

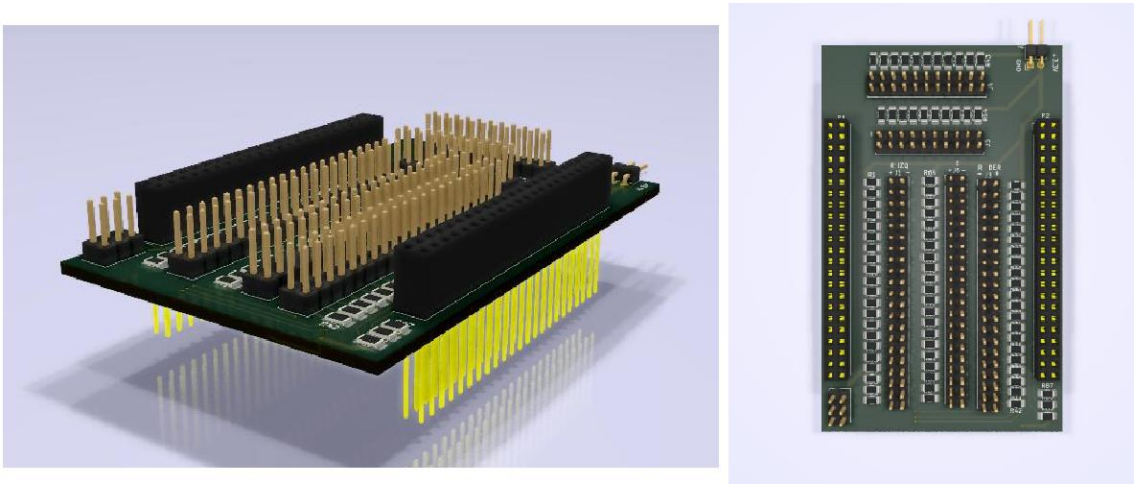


Ilustración 24. Renderizado de la PCB diseñada

Para la alimentación de la placa y del microcontrolador se hará uso de un sistema de pilas en serie que alcanza un valor de 4.5V, suficientes para alimentar al microcontrolador según la hoja de especificaciones de este. También se incorpora reductor de voltaje en paralelo a la alimentación del microcontrolador para obtener el voltaje de 3.3V necesario para polarizar los optoacopladores, ya que el voltaje máximo de entrada en los pines digitales es de 3.3V. Asimismo, se dispone también de un interruptor general para apagar y encender todo el sistema sin problemas.

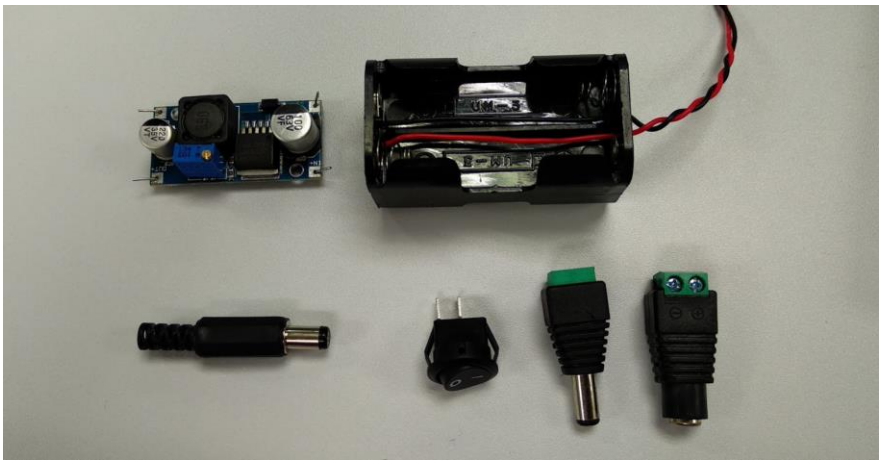


Ilustración 25. Componentes utilizados para la alimentación

4.4. DISEÑO MECÁNICO DE LA ESTRUCTURA

De acuerdo con la descomposición de las articulaciones de cada miembro de un brazo humano, se procede a diseñar los diferentes mecanismos para la correcta lectura de dichas rotaciones.

Todas las mediciones y distancias de cada articulación son tomadas de manera orientativa y posteriormente ajustadas de forma experimental.

En primer lugar, se tiene el siguiente conjunto de piezas que actúan de base fija y que se adaptan al hombro y pecho del usuario, fijándose completamente a este.

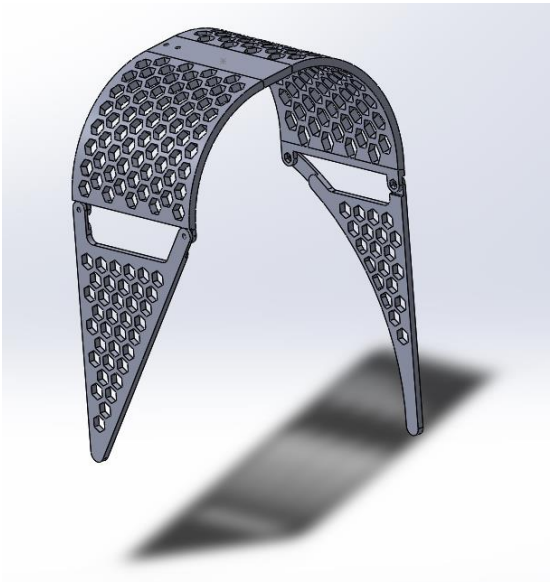


Ilustración 26.Hombrera de fijación corporal

Sobre esta hombrera se situarán enlazadas las siguientes estructuras una a una, conformando un sistema eslabón a eslabón.

Los primeros dos eslabones pertenecen al hombro y recogen los giros de la articulación de doble grado de libertad con un sistema de doble disco de encoder en dos ejes distintos pero coincidentes en el centro de la articulación.

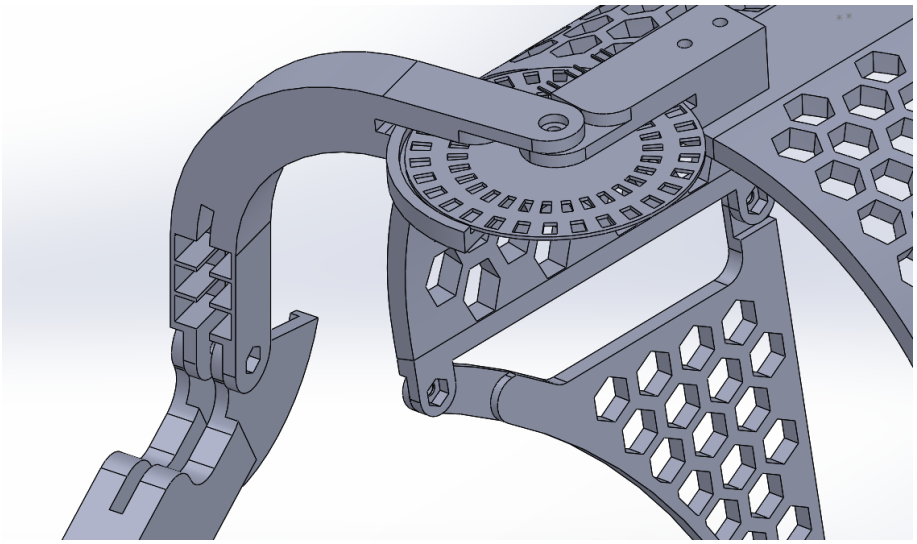


Ilustración 27. Lectura de los dos grados de libertad del hombro

El tercer eslabón contiene un encoder lineal, cuyo diámetro se ajusta y aproximo al de la parte inferior del brazo y que enlaza con la articulación del codo.

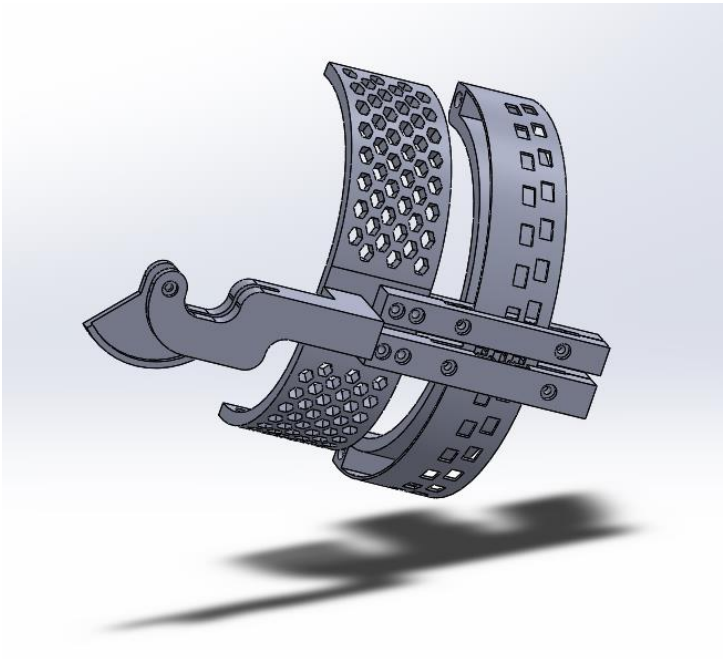


Ilustración 28. Lectura del grado de libertad restante del hombro

Dicho encoder lineal se compone de una banda curva perforada y de dos guias con ruedas que ajustan el desplazamiento de la banda. En una de ellas se dispone de los emisores y receptores del encoder.

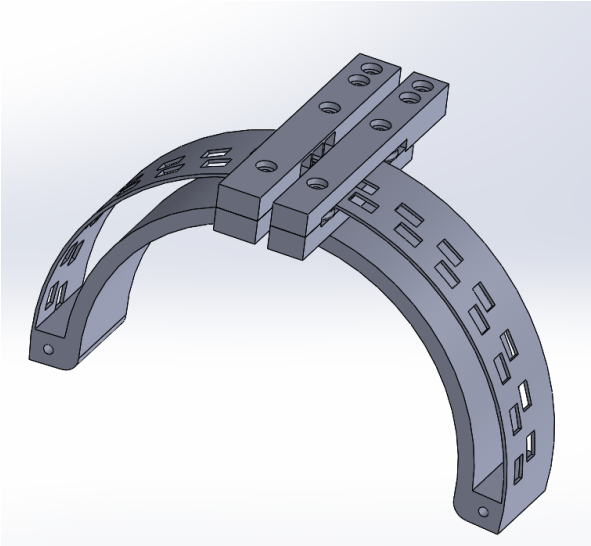


Ilustración 29. Encoder lineal curvo

El 4 pertenecen a la articulación del codo, que contiene un encoder rotativo de disco

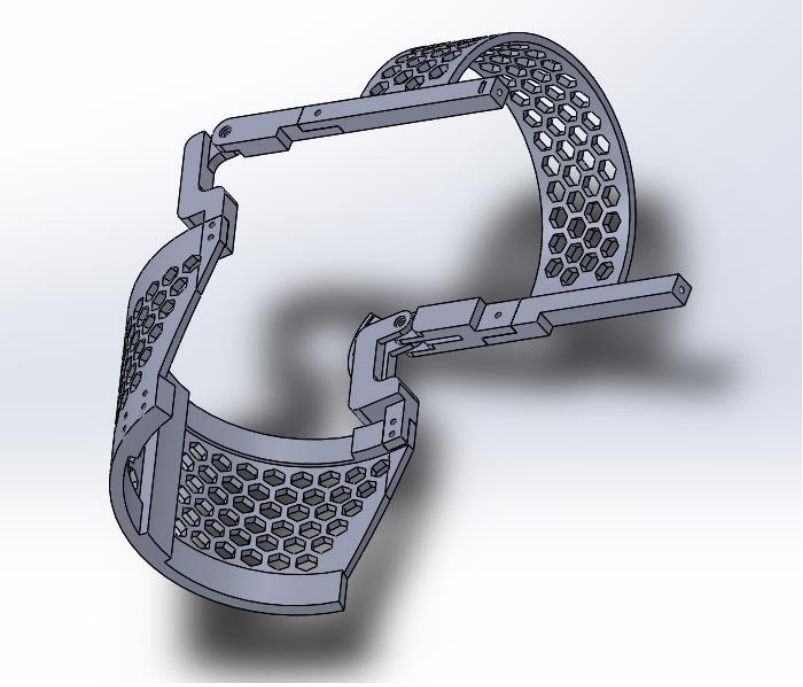


Ilustración 30. Sistema de lectura del grado de libertad del codo

El 5 eslabón se corresponde a la rotación del antebrazo respecto su propio eje, para lo que se hace uso de otro encoder lineal curvo. Dicha rotación se corresponde con una de las tres rotaciones de la muñeca equivalente en un brazo robot.

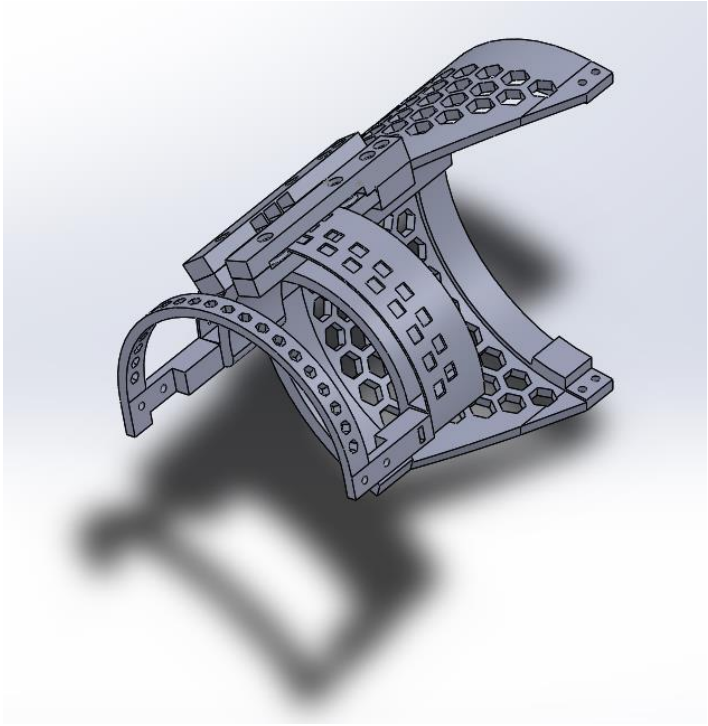


Ilustración 31. Sistema de lectura de 1 grado de libertad de la muñeca

Finalmente, el eslabón 6 y 7 recogen los dos grados de libertad restantes en la muñeca, teniendo en cuenta que estos deben de ser coincidentes en un único punto con el anterior eje de giro de la muñeca, recogiendo los 3 giros para la orientación de la muñeca. Asimismo, también se localiza en este último eslabón la sujeción de la mano a la estructura y se incorpora una zona para alojar los pulsadores de control.

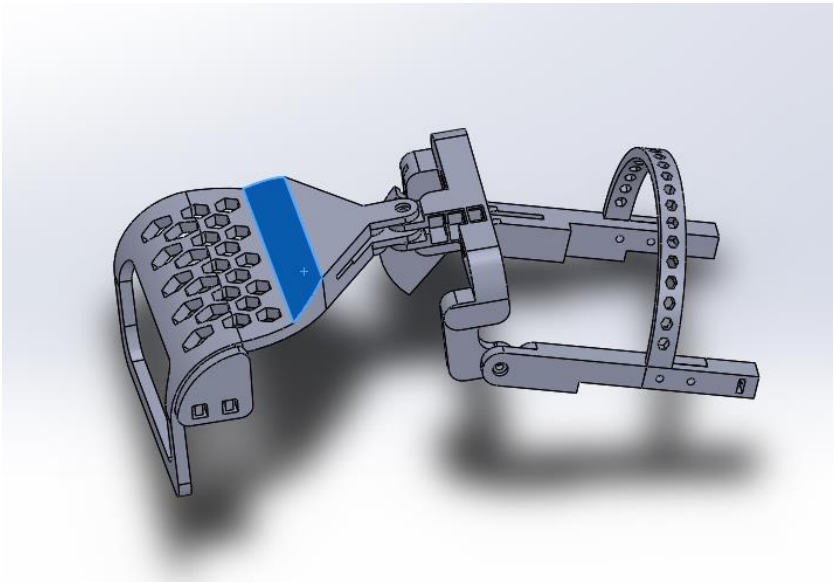


Ilustración 32. Sistema de lectura de los 2 grados de libertad de la muñeca restantes

Todos los elementos son fabricados mediante impresión 3D para verificar el correcto ajuste y dimensionamiento de la estructura. Además, se hace uso de un patrón hexagonal de perforado en la mayoría de las piezas para realizar un termoconformado in situ, ya que las piezas son impresas sin la curvatura de la parte que entra en contacto con el cuerpo, para poder mejorar la sujeción de estas con cada usuario. Las piezas son impresas con las siguientes características mediante una boquilla de 0.5mm y con PLA.

Tipo de piezas	Relleno(%)	Altura de capa(mm)
Termoconformadas	100	0.5
Ruedas de encoder lineal	100	0.1
Resto de piezas	25	0.5

Para las piezas termoconformadas es necesario tener un relleno macizo, ya que sino al realizar la curvatura de cada una de estas su resistencia disminuiría notablemente.

Finalmente se incorporan tiras de velcro para facilitar la colocación de toda la estructura y se añade un soporte en la estructura derecha para soportar la electrónica.

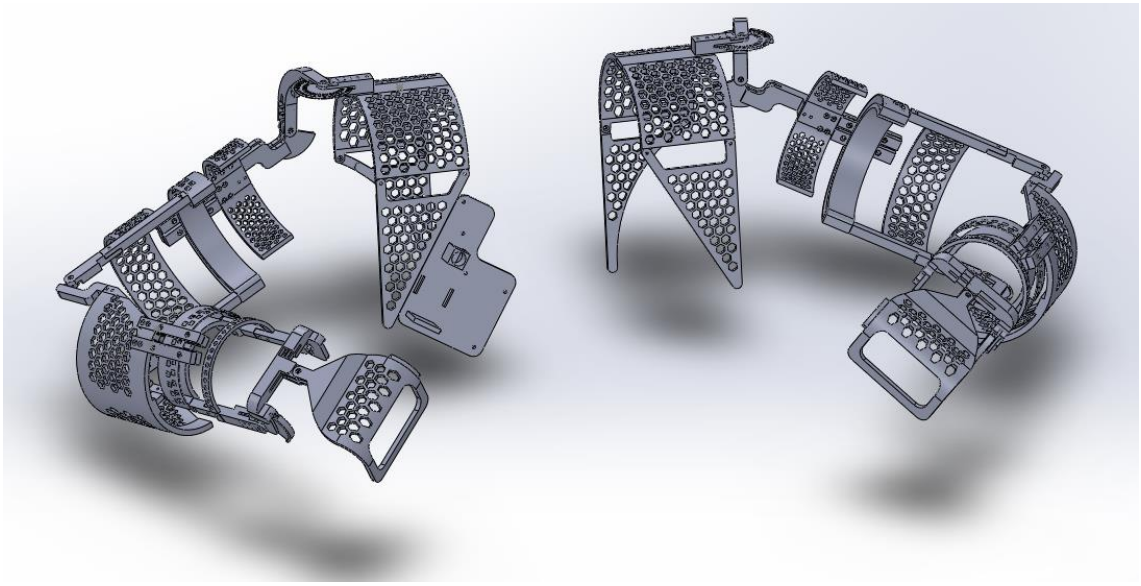


Ilustración 33. Ensamblaje final de la estructura sensorizada

5. CALCULO DE POSICIONAMIENTO

Con los datos obtenidos de las rotaciones de cada articulación, se procede a realizar las ordenes de movimiento de cada tipo de funcionamiento. Estos datos luego conformarán la trama que se enviará y que se detalla en el siguiente apartado.

5.1. POSICIONAMIENTO POR GIRO DE ARTICULACIONES

En este modo de funcionamiento, se toman los valores directamente obtenidos de los encoders y se ajustan al sistema de coordenadas de cada articulación del robot, es decir, se ajusta el valor de 0 de cada encoder al valor de la posición correspondiente en cada articulación del robot. Se muestra a continuación la correspondencia de dichos valores entre encoders y ejes de robots y el ajuste del 0 de cada articulación.

Eje de la estructura	Eje del robot	Ajuste del 0 en Izq	Ajuste del 0 en Der
1	2	-108	108
2	1	-125	-125
3	7	0	0
4	3	-8	8
5	4	10	-10
6	5	-5	5
7	6	30	-30

Tabla 4. Relación entre ejes de robot y estructura y desfase entre ellos del 0.

Este modo de funcionamiento es útil para mantener una configuración similar de cada miembro en cuanto a su orientación, copiando la forma general del brazo, sacrificando la precisión por posicionamiento. La configuración del brazo será similar, pero no la posición.

5.2. POSICIONAMIENTO EN EJES CARTESIANOS

En este segundo modo de funcionamiento se pretende replicar la posición final de la palma de la mano respecto del sistema de coordenadas de origen dado. De esta manera, si se hacen coincidir los sistemas de coordenadas de la estructura sensorizada con el sistema de coordenadas base del robot, se obtendría que la palma de la mano y el manipulador ocupan el mismo espacio.

Para ello es necesario calcular la posición de la palma mediante las rotaciones de cada miembro de cada brazo. Al tener un sistema consistente en 7 rotaciones de elementos, se tienen 7 matrices de transformación. Para ello se hace uso de los parámetros y algoritmos Denavit-Hartenberg. Este método ofrece una estandarización para referenciar los diferentes sistemas de coordenadas de una cadena cinemática y poder obtener así la matriz de transformación para resolver la cinemática directa.

Aplicando el algoritmo de Denavit-Hartenberg, se tienen lo siguientes sistemas de coordenadas para cada eje.

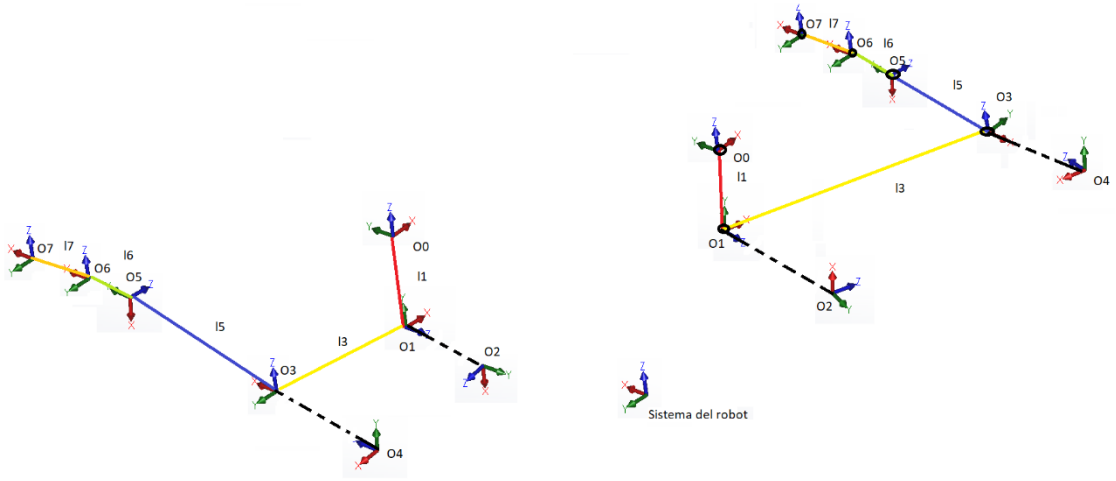


Ilustración 34. Sistema de coordenadas de cada articulación según algoritmo de Denavit-Hartenberg

Una vez establecidos los sistemas de coordenadas se obtienen las siguientes tablas con los 4 parámetros de Denavit-Hartenberg, que consisten en un desplazamiento y rotación respecte al eje z de una articulación y al eje x de otra.

	Brazo derecho				Brazo Izquierdo			
Articulación	Θ	d	a	α	Θ	d	a	α
1	q1	l1	0	90	q1	l1	0	90
2	q2+90	0	0	90	q2+90	0	0	90
3	q3+90	l3	0	90	q3+90	l3	0	90
4	q4-90	0	0	90	q4-90	0	0	90
5	q5-90	l5	0	90	q5-90	l5	0	90
6	q6+90	0	l6	-90	q6+90	0	l6	-90
7	q7	0	l7	0	q7	0	l7	0

Tabla 5. Tabla de parámetros de Denavit-Hartenberg

Gracias a estos parámetros, se pueden obtener las matrices de transformación de un sistema de coordenadas a otro, ya que dicha matriz es siempre la misma como se muestra a continuación. Dicha matriz esta formada por una matriz de 3x3 que nos indica la orientación del sistema de coordenadas final y de una matriz 3x1 situada a la derecha de esta que indica el posicionamiento xyz en ejes cartesianos.

$$ {}^{n-1}T_n = \left[\begin{array}{ccc|c} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & r_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & r_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right] $$

Si se realiza la multiplicación de cada matriz de transformación, se tiene la matriz de transformación del sistema de coordenadas final situado en la palma de la mano respecto del sistema de origen de la estructura sensorizada. Cada matriz dispone de valores que dependen del ángulo de cada articulación. Para calcular cada posición y orientación es necesario calcular todos los valores de cada matriz y realizar la multiplicación de estas. La matriz O es un cambio adicional realizado para referencia todo el sistema a el sistema de coordenadas del robot

$$^{7}T_{O} = O * {}^{1}T_{O} \cdot {}^{2}T_{1} \cdot {}^{3}T_{2} \cdot {}^{4}T_{3} \cdot {}^{5}T_{4} \cdot {}^{6}T_{5} \cdot {}^{7}T_{6}$$

No obstante, también será necesario el calculo de la orientación en cuaterniones mediante la matriz de orientación descrita anteriormente, ya que los robots de ABB funcionan con cuaterniones y no con ángulos de Euler, que los primeros no sufren el problema del bloqueo de cardan, que implica que cuando dos ejes de alinean se pierde un grado de libertad. Las ecuaciones que relacionan cada cuaternion con la matriz de rotación se muestran a continuación.

$$R = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix}$$

$$q_0 = \frac{1}{2}\sqrt{(n_x + o_y + a_z + 1)}$$

$$q_1 = \frac{1}{2} sign(o_z - a_y)\sqrt{(n_x - o_y - a_z + 1)}$$

$$q_2 = \frac{1}{2} sign(a_x - n_z)\sqrt{(-n_x + o_y - a_z + 1)}$$

$$q_3 = \frac{1}{2} sign(n_y - o_x)\sqrt{(-n_x - o_y + a_z + 1)}$$

5.3. IMPLEMENTACIÓN EN CÓDIGO

Para implementar el calculo de las trayectorias es necesario implementar las funciones de prod_matrices() para calcular la multiplicación de las 7 matrices de transformación y la función sign() que se corresponde con la función matemática signo, necesaria para el cálculo de los cuaterniones.

En primer lugar, se declaran las longitudes de cada elemento que rota. A continuación se realiza la conversión de los valores de ángulo de los encoders a radianes, se generan las matrices de transformación, se multiplican con la función prod_matrices(), se calculan los cuaterniones y se conforma la trama a enviar como se detallará posteriormente.

```
//Modo movimiento en ejes cartesianos
q1 = pos_actual.getCont_1_izq() * 3.1415 / 180;
q2 = (pos_actual.getCont_2_izq() * 3.1415 / 180) - 1.57;
q3 = (pos_actual.getCont_3_izq() * 3.1415 / 180) - 1.57;
q4 = (pos_actual.getCont_4_izq() * 3.1415 / 180) + 1.57;
q5 = (pos_actual.getCont_5_izq() * 3.1415 / 180) - 1.57;
q6 = (pos_actual.getCont_6_izq() * 3.1415 / 180) + 1.57;
q7 = pos_actual.getCont_7_izq() * 3.1415 / 180;

double R[4][4] = {{0,0,0,0},          {0,0,0,0},          {0,0,0,0},     {0,0,0,0}};
double O[4][4] = {{0,1,0,0},          {-1,0,0,220},          {0,0,1,300},      {0,0,0,1}};
double T0_1[4][4] = {{cos(q1),0,sin(q1),0},          {sin(q1),0,-cos(q1),0},          {0,1,0,-l1},
{0,0,0,1}};
double T1_2[4][4] = {{cos(q2),0,sin(q2),0},          {sin(q2),0,-cos(q2),0},          {0,1,0,0},
{0,0,0,1}};
double T2_3[4][4] = {{cos(q3),0,sin(q3),0},          {sin(q3),0,-cos(q3),0},          {0,1,0,l3},
{0,0,0,1}};
double T3_4[4][4] = {{cos(q4),0,sin(q4),0},          {sin(q4),0,-cos(q4),0},          {0,1,0,0},
{0,0,0,1}};
double T4_5[4][4] = {{cos(q5),0,sin(q5),0},          {sin(q5),0,-cos(q5),0},          {0,1,0,l5},
{0,0,0,1}};
double T5_6[4][4] = {{cos(q6),0,-sin(q6),l6*cos(q6)},  {sin(q6),0,cos(q6),l6*sin(q6)},
{0,-1,0,0},       {0,0,0,1}};
double T6_7[4][4] = {{cos(q7),-sin(q7),0,l7*cos(q7)},  {sin(q7),cos(q7),0,l7*sin(q7)},
{0,0,1,0},        {0,0,0,1}};

prod_matrices(O,T0_1,T1_2,T2_3,T3_4,T4_5,T5_6,T6_7,R);

cuat0 = sqrt(R[0][0] + R[1][1] + R[2][2] + 1)/2;
cuat1 = sign(R[2][1]-R[1][2]) * sqrt(R[0][0] - R[1][1] - R[2][2] + 1)/2;
cuat2 = sign(R[0][2]-R[2][0]) * sqrt(-R[0][0] + R[1][1] - R[2][2] + 1)/2;
cuat3 = sign(R[1][0]-R[0][1]) * sqrt(-R[0][0] - R[1][1] + R[2][2] + 1)/2;

trama << "(" << trunc(R[0][3]*10)/10 << "," << trunc(R[1][3]*10)/10 << "," <<
trunc(R[2][3]*10)/10 << "," << trunc(cuat0*10000)/10000 << "," <<
trunc(cuat1*10000)/10000 << "," << trunc(cuat2*10000)/10000 << "," <<
trunc(cuat3*10000)/10000 << "," << pos_actual.getGripperR() << "," <<
pos_actual.getModo() << ")" << endl;
```

6. COMUNICACIONES

Por tal de incrementar al máximo la portabilidad del sistema, se implementará una comunicación por sockets entre el microcontrolador, que dispone de un modulo Wi-Fi ya integrado, y el robot, que se conectará mediante el puerto de ethernet al mismo router que el microcontrolador.

La comunicación por sockets es una herramienta muy sencilla de comunicación. Existen diferentes tipos de sockets, pero básicamente se reducen a dos, sockets TCP y UDP. En el socket UDP se obtiene una mayor eficiencia, ya que no es necesario realizar una conexión previa, pero con el inconveniente de que no se garantiza la fiabilidad en la transmisión, es decir, se pueden duplicar o incluso perder mensajes.

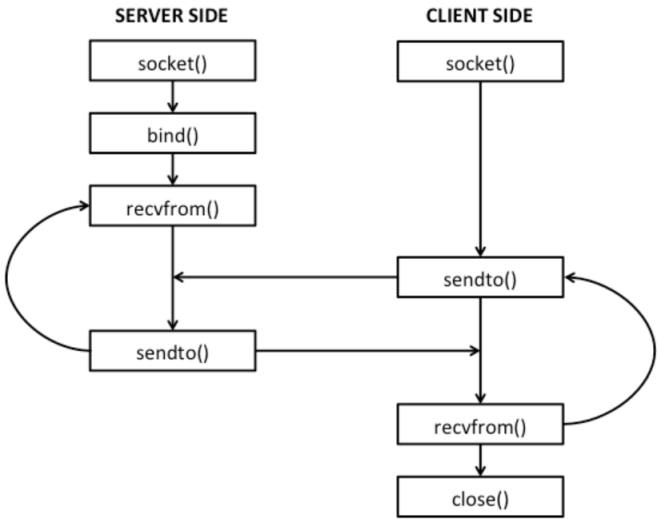


Ilustración 35.Esquema de interacción de servidor y cliente en conexión UDP(www.it.uu.se)

La comunicación TCP sin embargo es un poco más ineficiente, pero garantiza la integridad del mensaje. En este tipo de socket un dispositivo actúa como servidor que recibe peticiones de conexión y otros dispositivos actúan como clientes que realizan dichas peticiones. Una vez realizada la conexión se produce el intercambio de información.

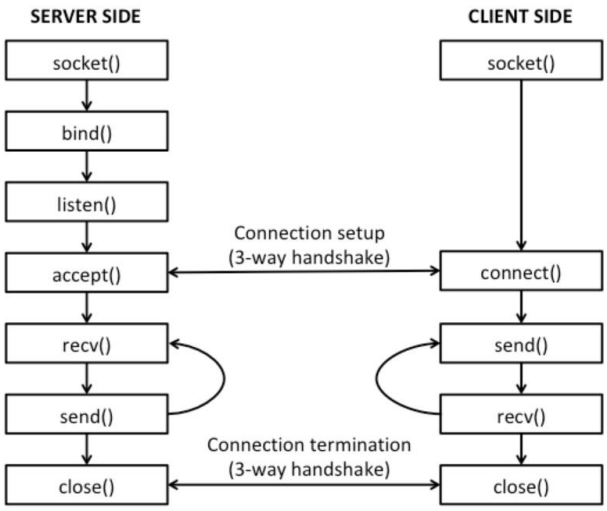


Ilustración 36. Esquema de interacción de servidor y cliente en conexión TCP (www.it.uu.se)

La arquitectura de la comunicación estará formada por sockets TCP/IP, consistentes en dos servidores con misma IP pero diferente puerto alojado en el microcontrolador y los diferentes brazos que actuarán como clientes.

Posteriormente a la obtención de los datos necesarios, ya sean del funcionamiento por giro de articulaciones o movimiento en ejes cartesianos, se genera la trama correspondiente a enviar por el socket de cada brazo. Dicha trama se inicia con el carácter "(" y se finaliza con ")", conteniendo un total de 9 campos en su interior se parados por comas, de los cuales los 7 primeros definen el movimiento ya sea o bien los siete ángulos correspondientes a las articulaciones o la posición y orientación formada por las coordenadas x, y y z y los 4 cuaterniones. Los dos últimos campos se utilizan para indicar el cierre o apertura del manipulador y el modo de selección. Los valores son truncados a 4 decimales para los cuaterniones y a 1 decimal para coordenadas cartesianas.

Modo	1	2	3	4	5	6	7	8	9
Articulacion	Eje 1	Eje 2	Eje 3	Eje 4	Eje 5	Eje 6	Eje 7	Activación gripper	Modo
Cartesianas	x	y	z	q0	q1	q2	q3	Activación gripper	Modo

Tabla 6. Campos de trama en función del modo de funcionamiento

Todas las tramas enviadas y recibidas se encapsulan en un buffer para su envió o para su recepción.

6.1. MICROCONTROLADOR

Como se ha comentado anteriormente, el microcontrolador alojará dos servidores en la misma IP con diferente puerto. En primer lugar, se crean las variables de los sockets. Se inicializa la configuración del socket del servidor y se liga a una IP específica mediante el comando "bind", en este caso la IP 192.168.1.224 y los puertos 40008 y 40009 para cada brazo. Cada servidor acepta una conexión mediante el "accept". A continuación, se envían y reciben los mensajes mediante "write" y "read". Se muestra un fragmento del código de generación del servidor hasta realizar el "accept".

```
int server_fd, socket_Yumi;
struct sockaddr_in address;
int addrlen = sizeof(address);
```

```
        // Creating socket file descriptor
        if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
            perror("socket creation failed");
            exit(EXIT_FAILURE);
        }

        address.sin_family = AF_INET;
        address.sin_addr.s_addr = INADDR_ANY;
        address.sin_port = htons( PORTR );

        // Poner socket en el puerto 40008
        if (bind(server_fd, (struct sockaddr *)&address, sizeof(address))<0) {
            perror("bind failed");
            exit(EXIT_FAILURE);
        }

        //Socket espera 1 cliente
        cout << "Esperando conexion de brazo R..." << endl;
        if (listen(server_fd, 1) < 0) {
            perror("listen");
            exit(EXIT_FAILURE);
        }

        //Socket acepta cliente
        if ((socket_Yumi = accept(server_fd, (struct sockaddr *)&address,
    (socklen_t*)&addrlen))<0) {
            perror("accept");
            exit(EXIT_FAILURE);
}
```

Aquí se muestran las ordenes de envío y recepción de mensaje con los comandos “write” y “read”.

```
    n = write(socket_Yumi,bufferenv,sizeof(bufferenv));
    printf("Enviado a brazo R: %s\n",bufferenv);
    if (n < 0){
            cerr << "ERROR writing to brazo R" << endl;
    }


    //LEE DE ROBOT
    cout << "Esperando ok brazo R..." << endl;
    n = read(socket_Yumi,bufferrec,255);
    if (n < 0){
            cerr << "ERROR reading from brazo R socket" << endl;
    }

    printf("%s recibido de brazo R\n",bufferrec);
```

6.2. ROBOT

Como se ha indicado anteriormente, cada brazo del robot realiza la conexión al microcontrolador como cliente a la IP y puerto especificados en el servidor con el módulo conecta_cliente() en RAPID.

```
PROC conecta_cliente()
SocketCreate socket_client;
SocketConnect socket_client, server_ip,puerto;
TPWrite"Conectado a la BBB";
ENDPROC
```

El robot queda esperando la recepción de una trama que empiece por el carácter "(". La trama comentada es recibida en el robot y es troceada y guardada en las diferentes variables en formato numérico en el módulo lectura_socket()

```
PROC lectura_socket()
        longitud_trama:= 0;
        trama_recibida := "";
        SocketReceive socket_client\Str:=trama_recibida \Time:=10000;
        WHILE  StrPart(trama_recibida,1,1) <> "(" DO
          SocketReceive socket_client\Str:=trama_recibida \Time:=10000;
        ENDWHILE

        !Calcular longitud de la cadena
        longitud_trama:=StrLen(trama_recibida);

        !Lectura de la string
        !IF longitud_trama>0 THEN
        IF StrPart(trama_recibida,1,1) = "(" THEN
            posicion_inicial:=2;
            posicion_final:=2;
            valores:=1;
            WHILE valores<=valores_esperados DO
              IF valores < valores_esperados THEN
                  car:=StrMemb(trama_recibida,posicion_final,",");
               ELSE
                  car:=StrMemb(trama_recibida,posicion_final,")");
               ENDIF
              WHILE car=FALSE DO ! se busca el caracter separador ","
                  posicion_final:=posicion_final+1;
                  IF valores < valores_esperados THEN
                    car:=StrMemb(trama_recibida,posicion_final,",");
                  ELSE
                    car:=StrMemb(trama_recibida,posicion_final,")");
                  ENDIF
              ENDWHILE
              valor:=StrPart(trama_recibida,posicion_inicial,(posicion_final-
posicion_inicial));
```

```
                !Asignacion de valores a variables
                IF valores=1 THEN
                   ok := StrToVal(valor,val_rec_1);
                ELSEIF valores=2 THEN
                   ok := StrToVal(valor,val_rec_2);
                ELSEIF valores=3 THEN
                      ok := StrToVal(valor,val_rec_3);
                ELSEIF valores=4 THEN
                      ok := StrToVal(valor,val_rec_4);
                ELSEIF valores=5 THEN
                   ok := StrToVal(valor,val_rec_5);
                ELSEIF valores=6 THEN
                      ok := StrToVal(valor,val_rec_6);
                ELSEIF valores=7 THEN
                      ok := StrToVal(valor,val_rec_7);
                ELSEIF valores=8 THEN
                     ok := StrToVal(valor,gripper_L);
                ELSE
                     ok := StrToVal(valor,modo);
                ENDIF
                valores:=valores+1;
                posicion_final:=posicion_final+1;
                posicion_inicial:=posicion_final;
            ENDWHILE
        ELSE
           TPWrite"Error recibiendo trama socket";
        ENDIF
ENDPROC
```

Finalmente se realizan todas las acciones, comenzando por la apertura de las garras, el posicionamiento inicial, de manera sincronizada, primero el brazo R y luego el L, los movimientos en función del tipo de modo seleccionado con MoveAbsJ, para el movimiento de articulación, y MoveL para los movimientos en ejes cartesianos y el control de los grippers. Al finalizar cada acción, se envía un “ok” a modo de ACK en el módulo movimientos()

```
PROC movimientos()
    !g_GripOut;
    WaitSyncTask YuMi_App_sync1, Yumi_App_task_list;
    continuar:=TRUE;
    mov_joint.extax.eax_a:=30;
    MoveAbsJ mov_joint,v1000,z10,GripperL;
    mov_joint.robax.rax_1:=-108;
    MoveAbsJ mov_joint,v1000,z10,GripperL;
    mov_joint.robax.rax_2:=-125;
    MoveAbsJ mov_joint,v1000,z10,GripperL;
    mov_joint.robax.rax_3:=0;
    mov_joint.robax.rax_4:=-8;
    mov_joint.robax.rax_5:=10;
    mov_joint.robax.rax_6:=-5;
```

```
      MoveAbsJ mov_joint,v1000,z10,GripperL;
   WHILE continuar DO
     lectura_socket;
        IF gripper_L<>gripper_L_ant THEN
           IF gripper_L=1 THEN
              !g_GripIn;
           ELSE
              !g_GripOut;
           ENDIF
           gripper_L_ant:=gripper_L;
        ENDIF

        IF modo=0 THEN
           mov_joint.robax.rax_1:=val_rec_2;
           mov_joint.robax.rax_2:=val_rec_1;
           mov_joint.robax.rax_3:=val_rec_4;
           mov_joint.robax.rax_4:=val_rec_5;
           mov_joint.robax.rax_5:=val_rec_6;
           mov_joint.robax.rax_6:=val_rec_7;
           mov_joint.extax.eax_a:=val_rec_3; !El septimo eje del Yumi se define como
un eje exterior
           MoveAbsJ mov_joint,v1000,z10,GripperL;
        ELSE
           pos_move_L.trans.x:=val_rec_1;
           pos_move_L.trans.y:=val_rec_2;
           pos_move_L.trans.z:=val_rec_3;
           pos_move_L.rot.q1:=val_rec_4;
           pos_move_L.rot.q2:=val_rec_5;
           pos_move_L.rot.q3:=val_rec_6;
           pos_move_L.rot.q4:=val_rec_7;
           MoveL pos_move_L,v100,fine,GripperL\WObj:=wobj0;
        ENDIF
        SocketSend socket_client\Str:="ok";
      ENDWHILE
   ENDPROC
```

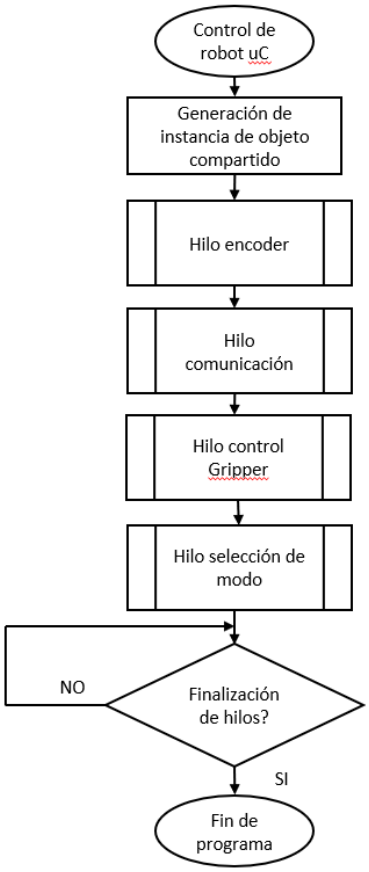
7. RESULTADOS Y FUNCIONAMIENTO DEL SISTEMA

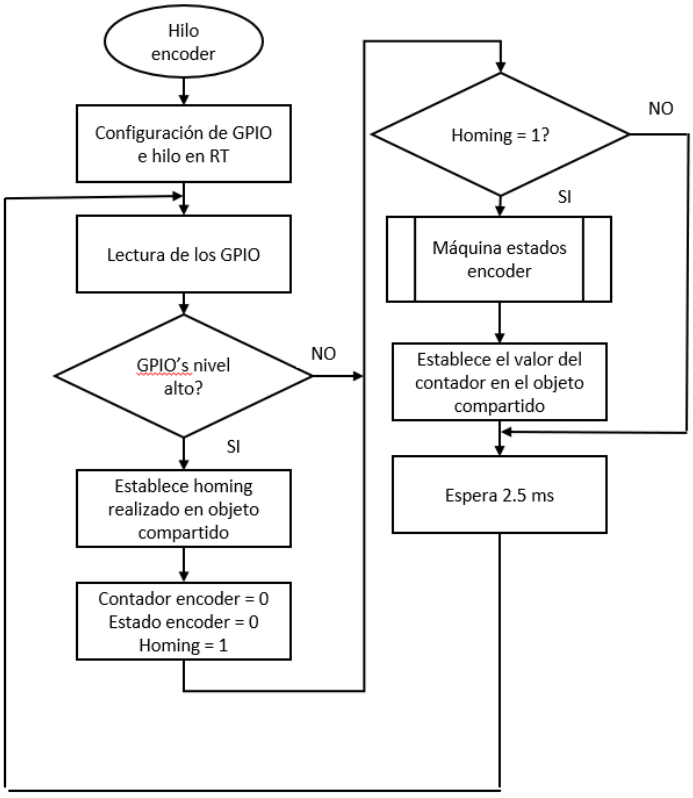
Después de realizar pruebas del sistema de encoder desarrollado con el microcontrolador, de la implementación de comunicación entre robot y microcontrolador y del calculo de posiciones, se realiza una prueba integrando todos los sistemas con resultados favorables.

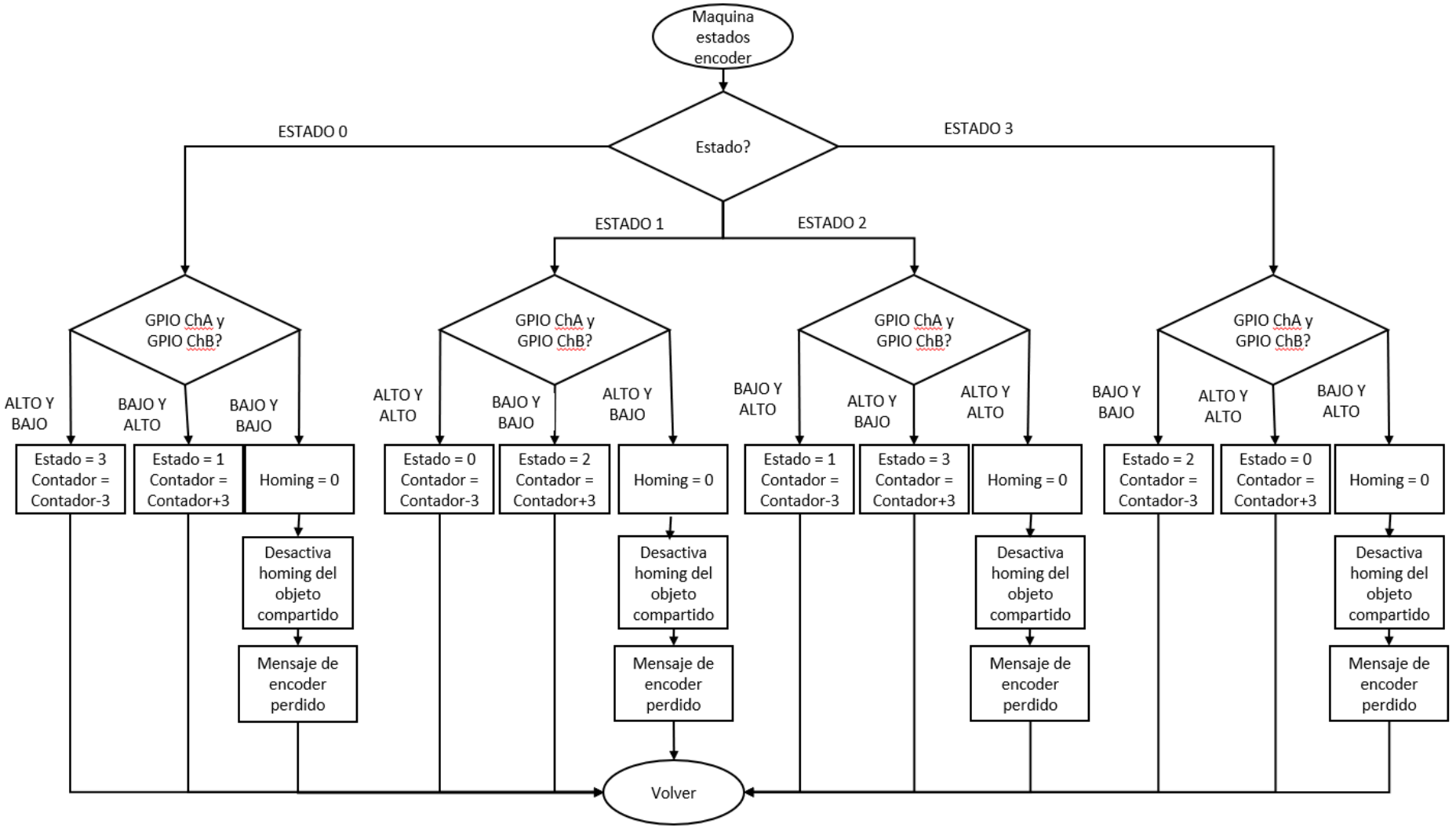
La prueba se inicia realizando un homing de cada uno de los encoders. Una vez el encoder pasa por el punto de 0, el hilo empieza a incrementar o disminuir el valor del contador. Asimismo, hasta que no se realice la puesta a 0 de cada uno de ellos, no se realiza en inicio de la transmisión de datos, aunque si que se produce previamente la conexión entre los sistemas. Una vez calibrados todos los encoders, empieza el envió de las tramas desde el microcontrolador, utilizando la composición de la trama anteriormente descrita. El funcionamiento de cada brazo es independiente, es decir, se puede iniciar el funcionamiento de un brazo sin necesidad de haber calibrado los encoders pertenecientes al otro. En caso de que se pierda el valor de algún encoder, se interrumpe la emisión de las tramas desde el controlador hasta que se recupere este. Se muestra asimismo un mensaje por la consola del microcontrolador para identificar el encoder que se ha perdido.

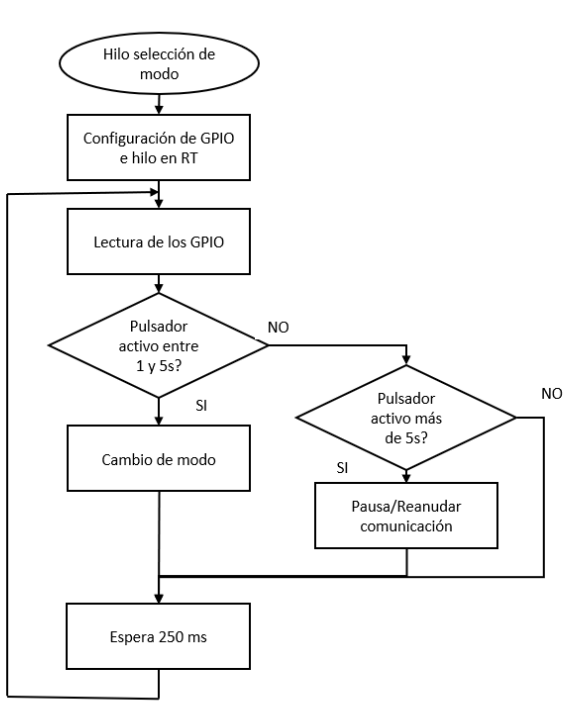
El robot por su parte recibe las tramas y ejecuta los movimientos en función de la modalidad descrita. El modo inicial de funcionamiento es el de giro de articulaciones. El robot recibe la trama, la descompone y realiza los movimientos correspondientes.

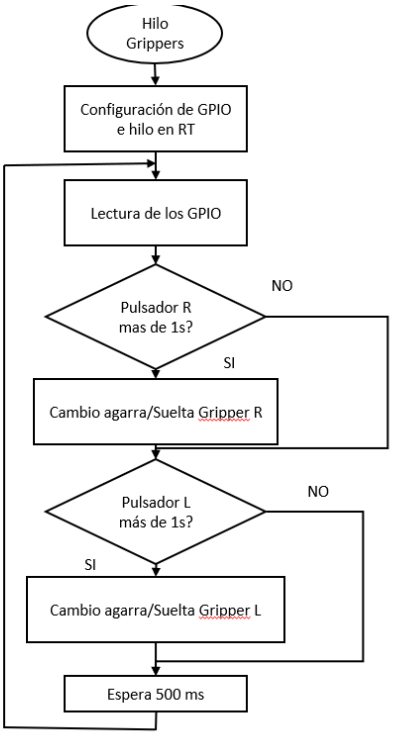
A continuación, se muestran los diagramas de flujo del comportamiento del sistema. Para simplificar el funcionamiento, solo se muestran los diagramas de un brazo y los de uno de los encoder.

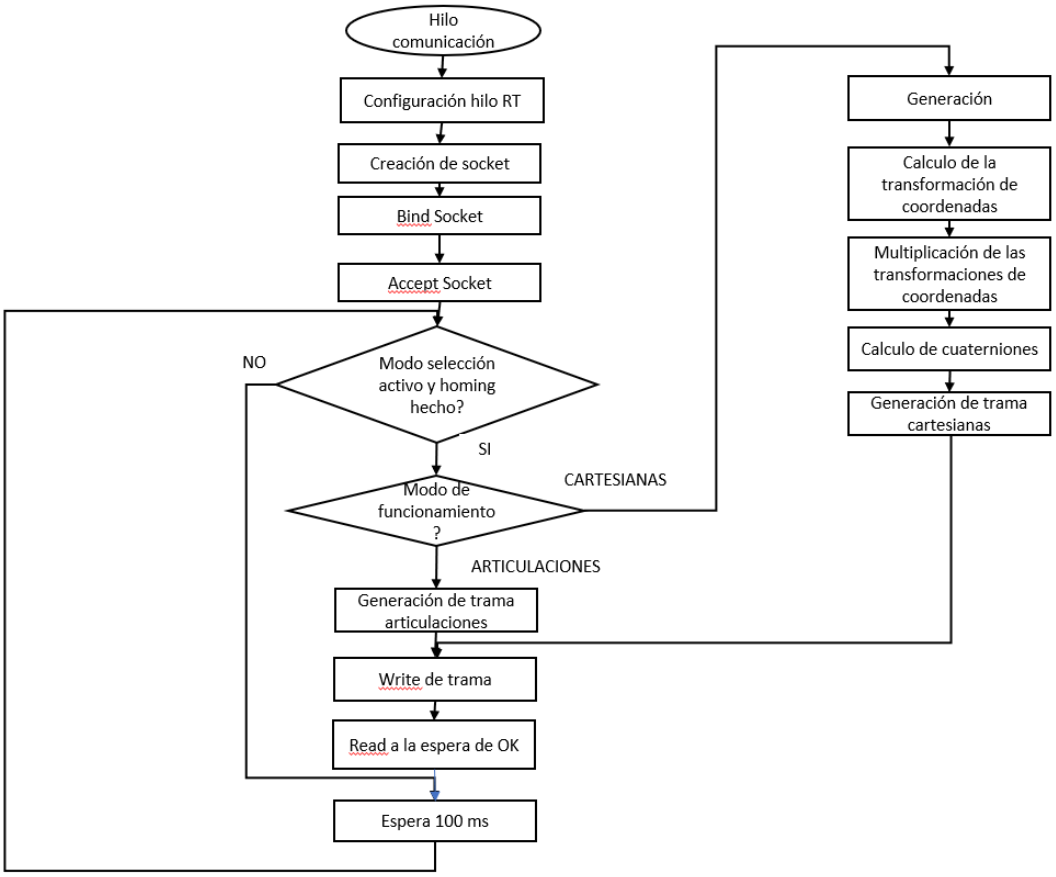

Control de robot uC
Generación de instancia de objeto compartido
Hilo encoder
Hilo comunicación
Hilo control Gripper
Hilo selección de modo
NO
Finalización de hilos?
SI
Fin de programa

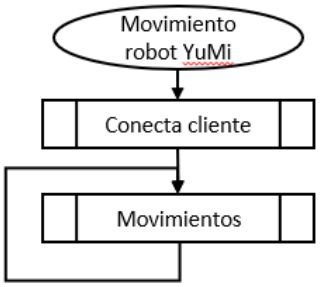

Hilo encoder
Configuración de GPIO e hilo en RT
Lectura de los GPIO
GPIO's nivel alto?
NO
SI
Establece homing realizado en objeto compartido
Contador encoder = 0
Estado encoder = 0
Homing = 1
Homing = 1?
NO
SI
Máquina estados encoder
Establece el valor del contador en el objeto compartido
Espera 2.5 ms

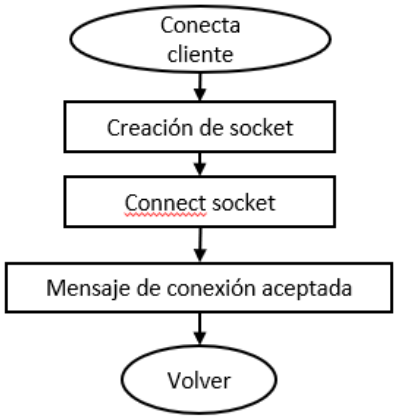

Maquina estados encoder
Estado?
ESTADO 0
ESTADO 1
ESTADO 2
ESTADO 3
GPIO ChA y GPIO ChB?
ALTO Y BAJO
BAJO Y ALTO
BAJO Y BAJO
ALTO Y ALTO
BAJO Y BAJO
ALTO Y BAJO
BAJO Y ALTO
ALTO Y BAJO
ALTO Y ALTO
BAJO Y BAJO
ALTO Y ALTO
BAJO Y ALTO
Estado = 3 Contador = Contador-3
Estado = 1 Contador = Contador+3
Homing = 0
Estado = 0 Contador = Contador-3
Estado = 2 Contador = Contador+3
Homing = 0
Estado = 1 Contador = Contador-3
Estado = 3 Contador = Contador+3
Homing = 0
Estado = 2 Contador = Contador-3
Estado = 0 Contador = Contador+3
Homing = 0
Desactiva homing del objeto compartido
Mensaje de encoder perdido
Desactiva homing del objeto compartido
Mensaje de encoder perdido
Desactiva homing del objeto compartido
Mensaje de encoder perdido
Desactiva homing del objeto compartido
Mensaje de encoder perdido
Volver

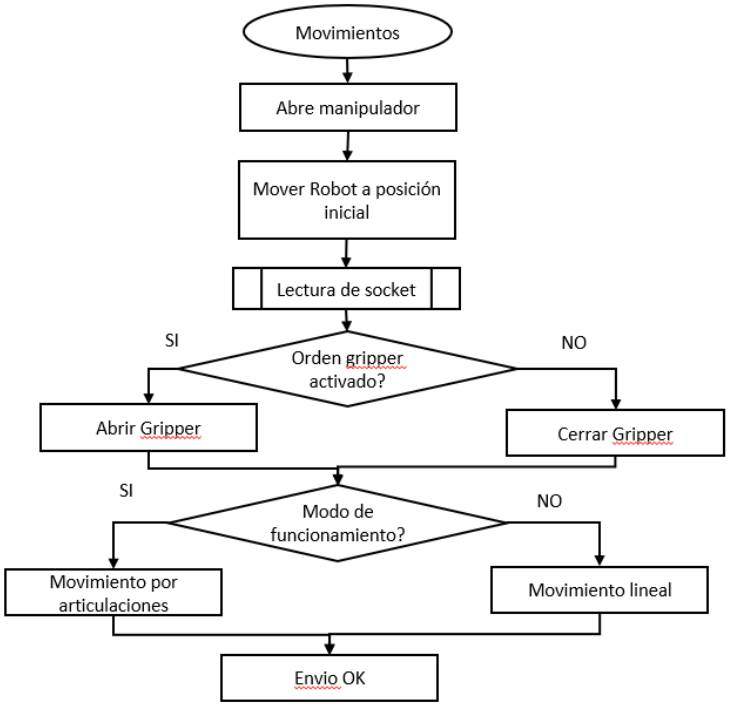

Hilo selección de modo
Configuración de GPIO e hilo en RT
Lectura de los GPIO
Pulsador activo entre 1 y 5s?
NO
SI
Pulsador activo más de 5s?
NO
SI
Cambio de modo
Pausa/Reanudar comunicación
Espera 250 ms

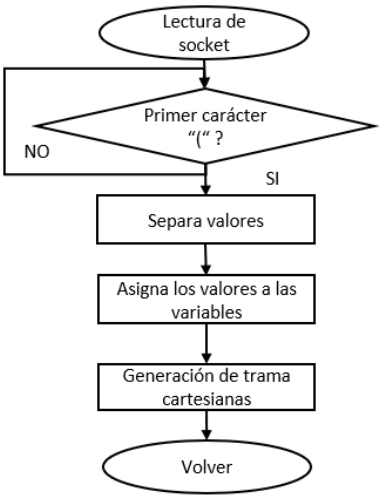

Hilo Grippers
Configuración de GPIO e hilo en RT
Lectura de los GPIO
Pulsador R mas de 1s?
NO
SI
Cambio agarra/Suelta Gripper R
Pulsador L más de 1s?
NO
SI
Cambio agarra/Suelta Gripper L
Espera 500 ms


Hilo comunicación
Configuración hilo RT
Creación de socket
Bind Socket
Accept Socket
NO
Modo selección activo y homing hecho?
SI
Modo de funcionamiento ?
CARTESIANAS
ARTICULACIONES
Generación de trama articulaciones
Write de trama
Read a la espera de OK
Espera 100 ms
Generación
Calculo de la transformación de coordenadas
Multiplicación de las transformaciones de coordenadas
Calculo de cuaterniones
Generación de trama cartesianas


Movimiento robot YuMi
Conecta cliente
Movimientos


Conecta cliente
Creación de socket
Connect socket
Mensaje de conexión aceptada
Volver


Movimientos
Abre manipulador
Mover Robot a posición inicial
Lectura de socket
SI
Orden gripper activado?
NO
Abrir Gripper
Cerrar Gripper
SI
Modo de funcionamiento?
NO
Movimiento por articulaciones
Movimiento lineal
Envio OK


Lectura de socket
Primer carácter "(" ?
NO
SI
Separa valores
Asigna los valores a las variables
Generación de trama cartesianas
Volver

8. ANÁLISIS ECONÓMICO

A continuación, se muestran los gasto debidos a los materiales y sistemas requeridos para la realización del proyecto:

Nombre	Unidades	Precio unidad	Subtotal
Conector Alimentación 2.1x5.5x9mm	1	0,52 €	0,52 €
Cinta Velcro	1	7,11 €	7,11 €
Interruptor 1C 2P	1	0,72 €	0,72 €
Conector Alimentación 2.1x5.5x9mm Borne Tornillo	1	0,67 €	0,67 €
Conector Alimentación 2.1x5.5x9mm Borne Tornillo Hembra	1	0,67 €	0,67 €
Termoretractil 1.6mm 1m	1	0,46 €	0,46 €
Termoretractil 2.4mm 1m	1	0,55 €	0,55 €
Termoretractil 13mm 1m	1	0,75 €	0,75 €
Pulsador reducido	3	0,63 €	1,89 €
Optoacoplador TCST2300	50	0,72 €	36,00 €
BeagleBone Black	1	73,99 €	73,99 €
Robot IRB14000 YuMi	1	40.000,00 €	40.000,00 €
Router Wifi	1	63,99 €	63,99 €
Pack resistencia SMD 10K 1/4W 3*2mm	1	1,62 €	1,62 €
Pack resistencia SMD 100 1/4W 3*2mm	1	1,62 €	1,62 €
Placa positiva fenolica	1	2,95 €	2,95 €
Sulfato de Sodio 500cc	1	5,45 €	5,45 €
Tira de pins 80 doble linea	6	0,59 €	3,54 €
Baterias AAA	3	1,24 €	3,72 €
Pasta de soldar	1	5,99 €	5,99 €
Set de pinzas para electronica	1	8,99 €	8,99 €
Reductor de tensión ajustable	1	4,96 €	4,96 €
Rollo de PLA BQ 1kg 1.75mm	2	18,99 €	37,98 €
		TOTAL	**40.264,14 €**

El coste dedicado a manufactura y diseño de programa y componentes se detalla a continuación:

FASES	DURACIÓN	PRECIO HORA	SUBTOTAL
SENSORIZACIÓN Y PRUEBAS	12	35,00 €	420,00 €
DISEÑO EXOESQUELETO	30	35,00 €	1.050,00 €
IMPRESIÓN 3D Y POSTERIOR MECANIZADO	46	35,00 €	1.610,00 €
MONTAJE Y DISEÑO ELECTRONICA	15	35,00 €	525,00 €
PROGRAMA RAPID	3	35,00 €	105,00 €
PROGRAMA MICROCONTROLADOR	22	35,00 €	770,00 €
TOTAL HORAS	128	TOTAL	4.480,00 €

Los costes totales del proyecto se muestran a continuación:

COSTES TOTALES	44.744,14 €

9. CONCLUSIONES

La realización de este proyecto engloba campos muy diversos, desde la electrónica instrumental hasta tareas de robótica pasando por la programación de sistemas empotrados. Se ha logrado desarrollar el conjunto de sistemas que integran el completo funcionamiento para la aplicación de teleoperación del robot. Se han realizado las pruebas pertinentes de la integración de cada sistema para su correcto funcionamiento, observando el correcto funcionamiento del sistema.

El modelo obtenido es muy fiel al comportamiento del brazo humano, consistente en los 7 grados de libertad comentados. Además, gracias al uso de las transformaciones según el algoritmo de Denavit-Hartenberg el modelo resulta fácilmente ajustable para cualquier otra fisonomía de brazo humano, reajustando los parámetros de longitud de cada miembro.

Se ha obtenido un sistema consistente en:

- Un modelo preciso del comportamiento del brazo humano.
- Una estructura sensorizada capaz de recoger los movimientos de los 7 grados de libertad con una precisión de 3º por articulación.
- Electrónica y diseños necesarios para la realización de un sensor de bajo coste de dimensiones reducidas y robusto.
- Un sistema de comunicaciones Wi-Fi que permite obtener una portabilidad del sistema y un uso más cómodo de este.
- Dos modos de funcionamiento para copiar configuraciones de los brazos o su posicionamiento.
- Control de fuerza de los manipuladores.
- Un sistema rápido y fácilmente replicable.

Cabe destacar que, aunque ofrece una precisión bastante elevada, no podría utilizarse para las mismas aplicaciones que otros sistemas utilizados para cirugía a distancia. No obstante, el coste de ambos sistemas es muy diferente por lo que se han obtenido muy buenos resultados teniendo en cuenta el coste del prototipo. Para lograr una mayor precisión se requieren de otros sistemas de fabricación del disco con codificación gray que garanticen el correcto y preciso mecanizado.

Los sistemas desarrollados son robustos, gracias a una comunicación TCP/IP. Además, gracias a la implementación realizada en máquina de estados del encoder, el sistema puede recuperarse de una perdida de pasos de cualquier articulación.

Gracias a todo el conjunto desarrollado, se tiene un sistema de bajo coste, si lo comparamos con otros sistemas de prestaciones similares, que es útil para realizar tareas de teleoperación. Asimismo, este sistema también resulta fácilmente reutilizable para cualquier otro tipo de robot, ya que el único aspecto a modificar sería el código del controlador del robot a controlar, modificando la lectura de las tramas en este. En el caso de utilizar cualquier otro robot de ABB, las modificaciones serian mínimas, ya que todos utilizan lenguaje RAPID. De la misma manera, para otros robots, las modificaciones no serían complejas, ya que los lenguajes de los controladores comparten muchas instrucciones en común en cuanto a lo que comunicación por sockets se refiere.

Además, el sistema implementado permitiría controlar varios robots de manera simultánea, ya que, al actuar el microcontrolador como servidor, podría admitir más clientes y hacer un control simultaneo de varios sistemas.

Por tanto, el actual sistema implementado en el microcontrolador seleccionado permite una continuación del desarrollo de esta aplicación para otros usos y otras funcionalidades más amplias de manera flexible y sin realizar modificaciones significativas.

10. TRABAJO FUTURO

Gracias al anterior sistema, se pueden implementar mejoras futuras que permitirán un uso más amplio del robot tales como:

- Mejora de la precisión del robot mediante la manufactura de discos codificados más precisos.
- Diseño de una aplicación que permita realizar la programación de robots y repetición de movimientos para utilizarlo como una herramienta de programación para gente no especializada en la programación de estos.
- Sistemas de retroalimentación del robot, incorporando servos a las articulaciones de la estructura para recibir un feedback del funcionamiento de este.
- Uso de otras herramientas en el robot que permitan aplicaciones más diversas, ya que solo seria necesario la retirada de los manipuladores actuales y pequeñas modificaciones en el software.
- Sistemas de monitorización del robot para visualizar en cada instante la posición real del robot con su entorno.
- De acuerdo con el punto anterior, esto también permitiría la integración de sistemas de visión incorporados en el propio robot.
- Uso de la misma estructura sensorizada para el control de varios brazos robots, pudiendo ser estos de diferentes tipologías y marcas.

11. BIBLIOGRAFIA

1Diaz-Hernandez Octavio, G.-V. V.-Z.-M.-S.-P. (2014). Un análisis cinemático del brazo humano para biomecánica. *Avances de la Ingeniería Mecánica en Mecánica Teórica*, 1068-1075.

(10 de Septiembre de 2019). Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Telerrob%C3%B3tica

(10 de Septiembre de 2019). Obtenido de Cyberneticzoo: http://cyberneticzoo.com/early-industrial-robots/1958-62-versatran-industrial-robot-harry-johnson-veljko-milenkovic/

ABB. (10 de Septiembre de 2019). Obtenido de ABB: https://new.abb.com/products/robotics/es/robots-industriales/yumi

Bolero, A. J. (2005). *Descripciones y transformaciones espaciales.*

Cecilia Garcia*, R. S. (2010). Diseño de un Controlador Híbrido en Ambientes Virtuales para teleoperación Robótica. *Revista Iberoamericana de Automática e Informática Industrial*, 53-62.

DÍAZ, M. A. (2016). Manipulación y posicionamiento de objetos desconocidos por parte de un robot autónomo. *Memoria para optar a titulo de ingeniero civil en computación*. Santiago de Chile.

Emanuel Slawiñski, J. F. (2006). Experiencias en teleoperación bilateral de robots. *Revista Iberoamericana de Automática e Informática Industrial*, 29-38.

Emmanuel Nuño Ortega, L. B. (2004). *Teleoperación de Robots: Técnicas, Aplicaciones, Entorno Sensorial y Teleoperación Inteligente.* Barcelona: UPC Commons.

Gutiérrez, M. H. (2002). Arquitectura de control, planificación y simulación para teleprogramación de robots. *Tesis doctoral*. Madrid, España.

Instruments, T. (10 de 9 de 2019). *Begleboard.org*. Obtenido de Begleboard.org: https://beagleboard.org/black

JAYC, F. M. (2018). CONTROL DE UN BRAZO ROBÓTICO VIRTUAL USANDO UN EXOESQUELETO ROBOT DE MIEMBRO SUPERIOR. *Trabajo de final de grado*. Bogotá.

Jhon Alejandro Montañez Barrera[1], M. L. (2017). Análisis de elementos en zona local y remota para la teleoperación del brazo robótico AL5A. *Prospect*, 53-63.

Nelson David Muñoz, N. A. (2006). Diseño de un escenario "en linea" para robots teleoperados desde internet. *Scientia et Technics Año XII*, 85-90.

Or, E. N. (2008). La teleoperación de robots y su impacto en la sociedad. *La gaceta*, 16.

Robotics, A. (2018). *Especificaciones del producto IRB14000.* Vasteras: ABB AB Robotics.

Robotics, A. (2018). *Manual de producto IRB14000.* Vasteras: ABB AB Robotics.

Robotics, A. (2018). *Manual de referencia técnica: Instrucciones, funciones y tipos de datos de RAPID.* Vasteras: ABB AB Robotics.

Robotics, A. (2018). *Manual del producto IRB 14000 Gripper.* Vasterás: ABB AB Robotics.

Semiconductors, V. (Agosto de 2017). Transmissive Optical Sensor with Phototransistor Output. *Datasheet*.

12. ANEXOS

MATRICES DE DENAVIT-HARTENBERG

Brazo derecho

$$O = \begin{matrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & -220 \\ 0 & 0 & 1 & 300 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$${}^{1}T_{O} = \begin{matrix} \cos(q1) & 0 & \sin(q1) & 0 \\ \sin(q1) & 0 & -\cos(q1) & 0 \\ 0 & 1 & 0 & -l1 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$${}^{2}T_{1} = \begin{matrix} \cos(q2+90) & 0 & \sin(q2+90) & 0 \\ \sin(q2+90) & 0 & -\cos(q2+90) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$${}^{3}T_{2} = \begin{matrix} \cos(q3+90) & 0 & \sin(q3+90) & 0 \\ \sin(q3+90) & 0 & -\cos(q3+90) & 0 \\ 0 & 1 & 0 & l3 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$${}^{4}T_{3} = \begin{matrix} \cos(q4-90) & 0 & \sin(q4-90) & 0 \\ \sin(q4-90) & 0 & -\cos(q4-90) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$${}^{5}T_{4} = \begin{matrix} \cos(q5-90) & 0 & \sin(q5-90) & 0 \\ \sin(q5-90) & 0 & -\cos(q5-90) & 0 \\ 0 & 1 & 0 & l5 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$${}^{6}T_{5} = \begin{matrix} \cos(q6+90) & 0 & -\sin(q6+90) & l6*\cos(q6+90) \\ \sin(q6+90) & 0 & \cos(q6+90) & l6*\sin(q6+90) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$${}^{7}T_{6} = \begin{matrix} \cos(q7) & -\sin(q7) & 0 & l7*\cos(q7) \\ \sin(q7) & 0 & 0 & l7*\sin(q7) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Brazo izquierdo

$$O = \begin{matrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 220 \\ 0 & 0 & 1 & 300 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$^{1}T_{O} = \begin{matrix} \cos(q1) & 0 & \sin(q1) & 0 \\ \sin(q1) & 0 & -\cos(q1) & 0 \\ 0 & 1 & 0 & -l1 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$^{2}T_{1} = \begin{matrix} \cos(q2-90) & 0 & \sin(q2-90) & 0 \\ \sin(q2-90) & 0 & -\cos(q2-90) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$^{3}T_{2} = \begin{matrix} \cos(q3-90) & 0 & \sin(q3-90) & 0 \\ \sin(q3-90) & 0 & -\cos(q3-90) & 0 \\ 0 & 1 & 0 & l3 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$^{4}T_{3} = \begin{matrix} \cos(q4+90) & 0 & \sin(q4+90) & 0 \\ \sin(q4+90) & 0 & -\cos(q4+90) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$^{5}T_{4} = \begin{matrix} \cos(q5-90) & 0 & \sin(q5-90) & 0 \\ \sin(q5-90) & 0 & -\cos(q5-90) & 0 \\ 0 & 1 & 0 & l5 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$^{6}T_{5} = \begin{matrix} \cos(q6+90) & 0 & -\sin(q6+90) & l6*\cos(q6+90) \\ \sin(q6+90) & 0 & \cos(q6+90) & l6*\sin(q6+90) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$^{7}T_{6} = \begin{matrix} \cos(q7) & -\sin(q7) & 0 & l7*\cos(q7) \\ \sin(q7) & 0 & 0 & l7*\sin(q7) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

CÁLCULO DE RESISTENCIAS PARA POLARIZAR EL OPTOACOPLADOR

Vled=3.3V

Imax = 60mA

$$R_{emisor} = \frac{V_{led}}{I_{max}} = \frac{3.3V}{60mA} = 55ohm \rightarrow 100ohm$$

La resistencia del colector simplemente tiene que ser lo suficientemente elevada para que con una corriente pequeña en el colector se produzca la caída de tensión deseada.

Vinput=3.3V

Ic=1mA

$$R_{emisor} = \frac{V_{input}}{I_C} = \frac{3.3V}{1mA} = 3.3Kohm \rightarrow 10Kohm$$

CÁLCULO DE ANCHO DE PISTA

Pista	Intensidad(A)	Ancho(mils)	mm	Aprox	Recomend
General Conectores	0,01365	0,084815861	0,002078837	0,2	1
General Emisores	1,02	6,337888507	0,155341647	0,2	1
Colector	0,00065	0,004038851	9,89922E-05	0,2	0,2
Emisor	0,04875	0,302913789	0,007424417	0,2	0,2
Pin Digital	0,00065	0,004038851	9,89922E-05	0,2	0,2

DATOS PRUEBA POTENCIOMETRO

Valor angular (º)	Valor en 1000				
0,0	0				
5	0				
10	0				
15,0	0				
20	0	Aprox linea	Erro1	Aprox 4 grado	Error4
23,5	0	29,721	-6,221	22,464	1,036
30,0	22,2	35,24658	-5,24658	29,2602374	0,739763
35	39,56	39,56748	-4,567484	34,5708136	0,429186
40	58,36	44,2468	-4,246804	40,3109978	-0,311
45,0	77,41	48,98835	-3,988349	46,1095635	-1,10956
50	95,23	53,42375	-3,423747	51,5120405	-1,51204
60	126	61,0824	-1,0824	60,7781903	-0,77819
70	164,59	70,68745	-0,687451	72,2625661	-2,26257
80	189,25	76,82533	3,174675	79,5092502	0,49075
90	224,42	85,57914	4,420862	89,7079084	0,292092
100	260	94,435	5,565	99,854352	0,145648
110	299,14	104,1769	5,823054	110,815473	-0,81547
120	339,43	114,2051	5,794873	121,888263	-1,88826
130	370,94	122,048	7,952034	130,412863	-0,41286
140	418,07	133,7786	6,221377	142,979442	-2,97944
150	463,73	145,1434	4,856603	155,006329	-5,00633
160	507	155,9133	4,0867	166,349063	-6,34906
170	557,75	168,545	1,455025	179,71103	-9,71103
180	599,26	178,8768	1,123186	190,804169	-10,8042
190	652,25	192,066	-2,066025	205,371101	-15,3711
200	695,97	202,9479	-2,947933	217,908355	-17,9084
210	736,75	213,0981	-3,098075	230,184806	-20,1848
220	776,31	222,9446	-2,944559	242,777814	-22,7778
230	820,02	233,824	-3,823978	257,658224	-27,6582
240	855,92	242,7595	-2,759488	270,787744	-30,7877
250	891,57	251,6328	-1,632773	284,773953	-34,774
260	930,89	261,4195	-1,419521	301,458994	-41,459
270	968,25	270,7184	-0,718425	318,706211	-48,7062
		Total error	101,347981	Total error	306,7006

CÓDIGO PRUEBA ENCODER

```
//para compilar g++ -lpthread -o Prueba_encoder_completo Prueba_encoder_completo.cpp
/TFM/HW/*.o
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <cstdlib>
#include <ctime>
#include <math.h>
#include "HW/GPIOuniv.h"

using namespace BBB;
using namespace std;

int main(void){
	GPIO pin_ChA(GPIO::P9_12);
	GPIO pin_ChB(GPIO::P9_14);
	GPIO pin_Home(GPIO::P9_16);
	pin_ChA.setDirection(GPIO::INPUT);
	pin_ChB.setDirection(GPIO::INPUT);
	pin_Home.setDirection(GPIO::INPUT);
	pin_ChA.setActiveLow();
	pin_ChB.setActiveLow();
	pin_Home.setActiveLow();

	int cont_1_izq = 0;

	cout << "Encoder iniciado" << endl;
	GPIO::VALUE estado1A,estado1B,estado1H;
	int homing = 0;
	int estado = 0;

	while(1){
		estado1H = pin_Home.getValue();
		estado1A = pin_ChA.getValue();
		estado1B = pin_ChB.getValue();

		if ((estado1H == GPIO::HIGH && estado1A == GPIO::HIGH) && estado1B ==
GPIO::HIGH){
			homing = 1;
			estado = 0;
		}
		if (homing == 1){
			switch(estado){
				case 0:
```

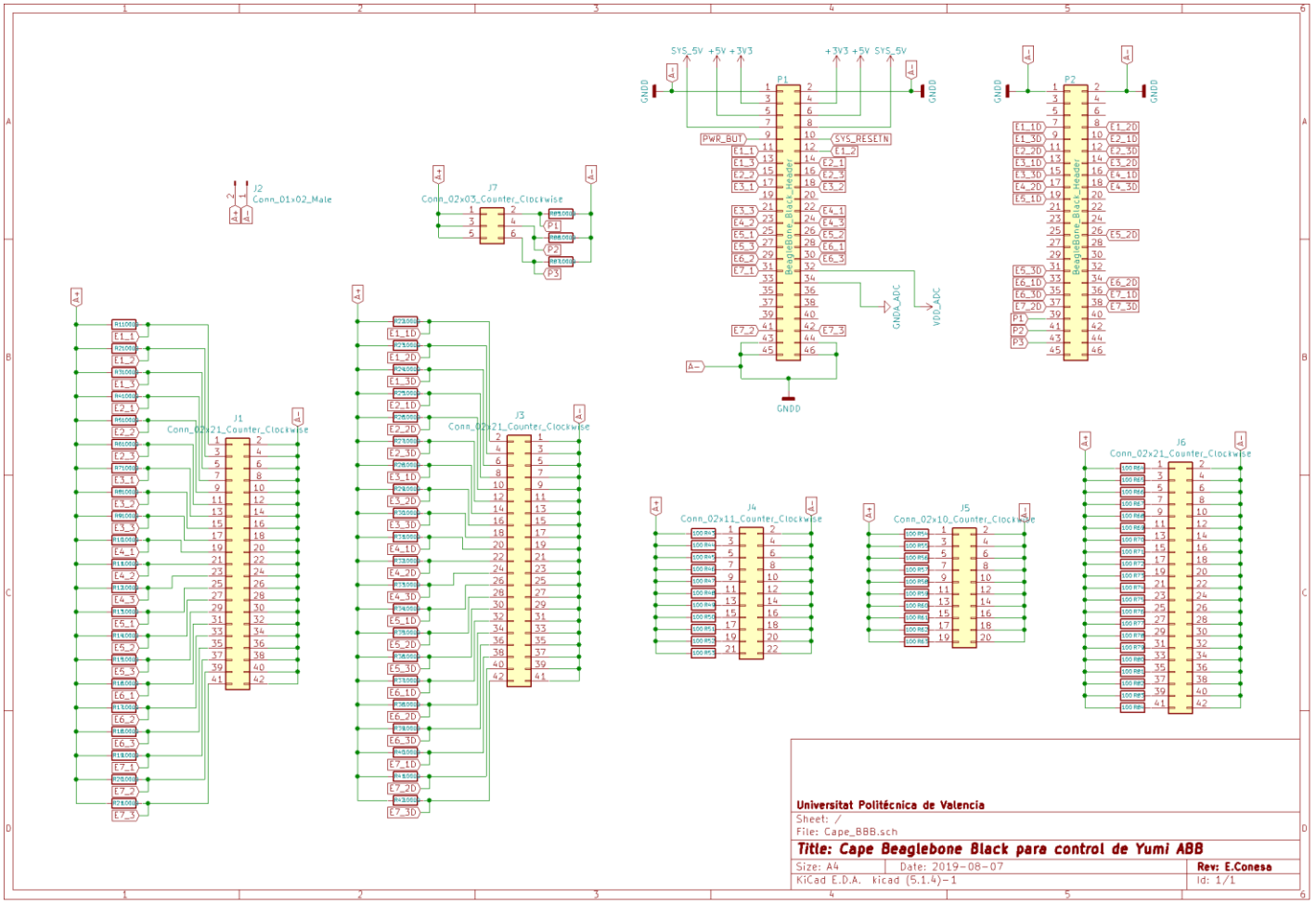
```
                        if (estado1A == GPIO::HIGH && estado1B ==
GPIO::LOW){
                                cont_1_izq--;
                                estado = 3;
                        }else if (estado1A == GPIO::LOW &&
estado1B == GPIO::HIGH){
                                cont_1_izq++;
                                estado = 1;
                        }else if((estado1A == GPIO::LOW &&
estado1B == GPIO::LOW)){
                                homing = 0;
                                cout << "Encoder 1 Izquierdo
perdido, recalibra" << endl;
                        }
                        break;
                case 1:
                        if (estado1A == GPIO::HIGH && estado1B ==
GPIO::HIGH){
                                cont_1_izq--;
                                estado = 0;
                        }else if (estado1A == GPIO::LOW &&
estado1B == GPIO::LOW){
                                cont_1_izq++;
                                estado = 2;
                        }else if((estado1A == GPIO::HIGH &&
estado1B == GPIO::LOW)){
                                homing = 0;
                                cout << "Encoder 1 Izquierdo
perdido, recalibra" << endl;
                        }
                        break;
                case 2:
                        if (estado1A == GPIO::LOW && estado1B ==
GPIO::HIGH){
                                cont_1_izq--;
                                estado = 1;
                        }else if (estado1A == GPIO::HIGH &&
estado1B == GPIO::LOW){
                                cont_1_izq++;
                                estado = 3;
                        }else if((estado1A == GPIO::HIGH &&
estado1B == GPIO::HIGH)){
                                homing = 0;
                                cout << "Encoder 1 Izquierdo
perdido, recalibra" << endl;
                        }
                        break;
                case 3:
                        if (estado1A == GPIO::LOW && estado1B ==
GPIO::LOW){
                                cont_1_izq--;
                                estado = 2;
```

```
                                                }else if (estado1A == GPIO::HIGH &&
estado1B == GPIO::HIGH){
                                                        cont_1_izq++;
                                                        estado = 0;
                                                }else if((estado1A == GPIO::LOW &&
estado1B == GPIO::HIGH)){
                                                        homing = 0;
                                                        cout << "Encoder 1 Izquierdo
perdido, recalibra" << endl;
                                                }
                                                break;


                                }
                        }
                        //cout << "Homing: " << homing << endl;
                        //cout << "Estado actual: " << estado << endl;
                        //cout << "Contador: " << cont_1_izq << endl;
                        usleep(1000000);
                }
}
```

DATASHEETS

ESQUEMÁTICO PCB



PLANOS DE PIEZAS