Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Mecánica

CONTROL CINEMÁTICO DE ROBOTS SERIE CON ARDUINO

Autor:

Tejedor García, Noelia

Tutores:

Tristán Vega, Antonio

Pérez Rueda, Mª Ángeles

Departamento CMeIM/EGI/ICGF/IM/IPF

Valladolid, Julio de 2018.

AGRADECIMIENTOS

Gracias papá, por darme ánimos cada día durante todos estos años. Gracias mamá, porque aunque había otras opciones nunca me negaste esta. Gracias por todo vuestro esfuerzo.

Gracias Mario, por estar dispuesto a ayudarme siempre.

Por último, pero no por ello menos importante, gracias a mis dos tutores, Antonio y María Ángeles, quienes me han guiado en todo momento y me han aportado todo lo necesario para llegar al resultado final de este proyecto. Gracias por darme la oportunidad de realizarlo a vuestro lado.

RESUMEN

En el presente proyecto se estudia la cinemática de un robot serie antropomórfico de 6 grados de libertad, relativamente popular en proyectos de robótica. Los actuadores eléctricos que moverán las articulaciones del brazo robótico son servomotores controlados con una placa Arduino, una plataforma de hardware libre que se puede programar a través de un entorno de desarrollo integrado y que se puede controlar desde un ordenador.

El brazo robótico ha sido dimensionado en su totalidad y seguidamente se ha procedido a su montaje. Se han realizado las conexiones eléctricas y electrónicas necesarias entre los servomotores – placa Arduino – ordenador.

Se ha desarrollado el problema cinemático directo e inverso, utilizando el método de Denavit-Hartenberg, herramienta matemática ampliamente conocida en el área de la robótica. La cinemática inversa se ha resuelto con el método de desacoplo cinemático, separando los problemas de posición y orientación.

La cinemática se ha resuelto con la herramienta de software matemático de MATLAB, en la que también se ha diseñado y programado una interfaz gráfica desde la que controlar el brazo robótico.

Palabras Clave: Robot Serie, Brazo Robótico, Cinemática, Arduino, Matlab

ABSTRACT

In this project we study the kinematics of a six degrees of freedom series anthropomorphic robot, relatively popular in robotics projects. The electric actuators that will move the joints of the robotic arm are servomotors that will be controlled with an Arduino board, a free hardware platform that can be programmed through an integrated development environment and that can be controlled from a computer.

The robotic arm has been entirely dimensioned and then has been assembled. The necessary electrical and electronic connections have been made between servomotors - Arduino board - computer.

The direct and inverse kinematics' problem has been developed using the Denavit-Hartenberg method, a mathematical tool widely known in the area of robotics. The inverse kinematics has been solved with the kinematic decoupling method, which separates the problems of position and orientation.

The kinematics has been solved with MATLAB's mathematical software tool, in which a graphical interface has been designed from which we can control the robotic arm.

Key Words: Series Robot, Robotic Arm, Kinematics, Arduino, Matlab

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN Y OBJETIVOS DEL PROYECTO

1.1 OBJETIVOS

El objetivo es el estudio cinemático directo e inverso, la construcción, el control y la evaluación de un brazo robótico antropomórfico de 6 grados de libertad, cuyas articulaciones se mueven con actuadores eléctricos (servomotores). El brazo robótico se utilizará como material docente para diversas asignaturas del Área de Ingeniería Mecánica en la Escuela de Ingenierías Industriales, en especial en la asignatura de Mecánica de Robots.

El brazo robótico es un modelo relativamente popular en proyectos de robótica, que se puede adquirir sin montar, es decir, por piezas. Los servomotores y toda la tornillería también se incluyen en el kit de montaje del brazo robótico.

La completa definición de las piezas que componen el brazo robótico también es un objetivo del presente proyecto. Se han tomado las dimensiones de todas las piezas antes de montar el robot. Entre los equipos de medida utilizados se encuentra el Proyector de Perfiles de eje horizontal (equipo de medida sin contacto, ya que las dimensiones de las piezas son muy pequeñas) y el pie de rey analógico. Finalmente se han modelado todas las piezas con el programa de diseño CATIA V5.

Es necesario también diseñar una caja base cuya función es proporcionar una base estable y móvil para el brazo robótico. Se ha diseñado teniendo en cuenta varios requisitos, geométricos y funcionales.

Los actuadores eléctricos se controlan a través del IDE (entorno de desarrollo integrado) de Arduino, una plataforma de hardware libre, basada en una placa con un microcontrolador.

La programación para la resolución de la cinemática directa e inversa se ha hecho con la plataforma de programación de MATLAB, que se comunica a su vez con la placa Arduino a través del puerto serie de un ordenador.

El objetivo final del proyecto consiste en diseñar una interfaz gráfica (GUI) en MATLAB, con un entorno de desarrollo integrado en el propio programa (conocido como AppDesigner), que nos proporciona un gran conjunto de componentes para diseñar la aplicación mediante la cual vamos a interactuar con el robot. La interfaz gráfica diseñada proporcionará las siguientes funciones al usuario:

- Mover el brazo robótico eje a eje
- Mover en coordenadas cartesianas
- Grabar posiciones absolutas y relativas y moverse entre ellas

1.2 JUSTIFICACIÓN DEL PROYECTO

Este proyecto está concebido desde un principio con fines didácticos, para que los alumnos de la Escuela de Ingenierías Industriales (EII) puedan ver todos los componentes del brazo robótico y puedan controlarlo desde un ordenador.

En el área de Ingeniería Mecánica actualmente hay un robot, el SCORBOT ER 4u, que se utiliza para las prácticas de la asignatura optativa de 4º curso de Mecánica de Robots. Es un robot antropomórfico de 5 grados de libertad (todos pares de revolución). La muñeca sólo tiene 2 grados de libertad.

La idea es adquirir un nuevo robot que sea distinto del que se dispone, no sólo en el número de grados de libertad, sino en el modo de mover las articulaciones. Además este nuevo robot será de pequeñas dimensiones, para poder transportarlo si fuera necesario. También tendremos la opción de desmontarlo si en algún momento necesitamos hacer alguna modificación, como ajuste de componentes, modificar el recorrido de los servomotores o incluso sustituir los mismos servos por otros con mejores prestaciones.

Se pretende que el mismo código programado en Matlab y Arduino pueda ser utilizado para otros robots serie, cambiando únicamente la parte específica de cada uno. Al mismo tiempo, se pretende que el código de control cinemático sea visible y resulte didáctico a los alumnos.

1.3 ESTRUCTURA DE LA MEMORIA

Este texto está organizado en 9 capítulos, más bibliografía y anexos. Algunos de ellos incluyen introducciones a la robótica y al software y hardware utilizados, que aunque son ampliamente conocidos requieren de una pequeña base de conocimientos para poder entender el proyecto.

A continuación sigue una descripción más específica de cada capítulo.

El capítulo 1 es una introducción donde se exponen los objetivos del proyecto así como su justificación.

El capítulo 2 es una introducción a la robótica, donde se hace una posible clasificación de los robots, se explica la estructura mecánica de un robot y se hace un resumen del modo de representación de la posición y orientación de un sólido rígido.

El capítulo 3 presenta todas las características específicas del robot escogido para realizar este proyecto, se explican los componentes que se han diseñado para formar parte del robot y los componentes requeridos para su funcionamiento.

El capítulo 4 presenta el método utilizado para el dimensionamiento de las piezas del brazo robótico, los equipos utilizados y las medidas relevantes obtenidas.

El capítulo 5 es una introducción a Arduino, tanto al software como al hardware que vamos a utilizar para controlar el brazo robótico.

El capítulo 6 es la descripción del montaje del robot, donde se explica la conexión de los servos durante el montaje y la conexión de los mismos una vez montado el robot.

El capítulo 7 presenta la solución del modelo cinemático usando el método de representación de Denavit-Hartenberg, y el problema de la cinemática inversa resuelto mediante el enfoque geométrico específico para el robot estudiado y el desacoplo cinemático de los problemas de posición y orientación.

El capítulo 8 describe el entorno de programación en las plataformas de programación de MATLAB y Arduino, centrándose en la parte de MATLAB.

El capítulo 9 presenta las conclusiones derivadas del trabajo, limitaciones del brazo robótico y posibles líneas futuras de trabajo.

La Bibliografía son todas las referencias consultadas para realizar este proyecto.

En los Anexos se incluyen las Instrucciones de Montaje y los planos.

CAPÍTULO 2. INTRODUCCIÓN A LA ROBÓTICA

2.1 INTRODUCCIÓN

La robótica es un campo de muchas aplicaciones de la tecnología moderna que cruza los límites de la ingeniería tradicional. Entender la complejidad de los robots y sus aplicaciones requiere conocimientos de ingeniería eléctrica, ingeniería mecánica, ingeniería industrial y de sistemas, conocimientos de computación, economía y matemáticas. Las nuevas disciplinas de la ingeniería, tal como la ingeniería de la fabricación, aplicaciones de ingeniería y conocimientos de ingeniería han emergido para tratar con la complejidad de los campos de la robótica y la industria de la automatización.

El término robot fue introducido por primera vez por el dramaturgo checo Karel Capek en su obra *Rossum's Universal Robot* en 1920. Desde entonces el término ha sido aplicado a una gran variedad de dispositivos mecánicos.

Pero sin duda alguna, el término robot fue impulsado por las obras de ficción del escritor americano de origen ruso Isaac Asimov (1920 - 1992). Asimov usó la palabra robot por primera vez en un relato corto, titulado "Runaround", publicado en 1942. En una historia publicada en la revista "Galaxy Science Fiction" en 1947, Asimov formuló las tres leyes de la robótica. Y posteriormente añadió una ley previa, o ley cero. Las implicaciones de estas leyes para el desarrollo de la tecnología del futuro han sido analizadas por R. Clarke (Clarke, 1993/1994). Estas leyes son las siguientes:

> **Ley número cero**: un robot no puede perjudicar a la humanidad, o, por su inacción, permitir que la humanidad sufra daño.
>
> **Ley número uno**: un robot no puede perjudicar a un ser humano, ni con su inacción permitir que un ser humano sufra daño, excepto si tales órdenes entran en conflicto con la ley anterior.
>
> **Ley número dos**: un robot ha de obedecer las órdenes recibidas de un ser humano, excepto si tales órdenes entran en conflicto con la ley número cero o la ley número uno.
>
> **Ley número tres**: un robot debe proteger su propia existencia, mientras tal protección no entre en conflicto con la ley número cero, número uno o número dos.

Se le atribuye a Asimov la creación del término robotics (robótica) y sin lugar a duda, desde su obra literaria, ha contribuido decisivamente a la divulgación y difusión de la robótica.

Nosotros nos referiremos con el término de robot a un manipulador industrial controlado por un ordenador. Este tipo de robot es esencialmente un brazo mecánico operando bajo el control de un computador. Tales dispositivos, lejos de ser como los robots de ciencia ficción, son sin embargo sistemas electromecánicos extremadamente complejos cuya descripción analítica

requiere métodos avanzados, presentando muchos retos e interesantes problemas de búsqueda.

El robot, tal como lo hemos definido, nace de la suma de dos tecnologías como son los teleoperadores y las fresadoras de control numérico. Los teleoperadores se desarrollaron durante la segunda guerra mundial para encargarse de los materiales radioactivos. El control numérico por computador (CNC) fue desarrollado debido a la alta precisión requerida en la fabricación de ciertos artículos, como los componentes de aviones de alto rendimiento. Los primeros robots esencialmente combinaban los enlaces mecánicos de los teleoperadores con la autonomía y la programabilidad de las máquinas CNC.

EVOLUCIÓN DE LA ROBÓTICA INDUSTRIAL

La robótica industrial se desarrolla para lograr mayores niveles de automatización, en función de la incorporación de robots, que añaden a las capacidades de las máquinas automáticas una característica fundamental: la capacidad de ejecutar tareas variadas, empleando sensores y la inteligencia artificial, a fin de conseguir una determinada capacidad de aprendizaje.

La *Robotics Industries Association* (RIA) de EEUU estableció en 1979 la siguiente definición de robot industrial:

Definición: manipulador reprogramable y multifuncional, diseñado para mover materiales, herramientas o dispositivos especializados, mediante movimientos variables, programados para la ejecución de tareas de forma automática.

El elemento clave de la definición anterior es la reprogramabilidad, que da a un robot su utilidad y adaptabilidad. La revolución de los robots es en parte, la gran revolución de los ordenadores.

En 1982 se aceptó universalmente ésta definición y se establecieron cuatro clases genéricas de robots: robot secuencial de secuencia variable, robot de aprendizaje, robot de control numérico, y robot inteligente. El desarrollo acelerado de la tecnología robótica, y la aparición de nuevos robots de servicios, ha llevado a reformular estas definiciones.

Así, la **definición de robot industrial según la ISO** (*International Standard Organitation*) es la siguiente: "Manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas."

Los objetivos específicos de la robótica industrial son los siguientes: el aumento de la productividad, el aumento de la flexibilidad en la adaptación a las diferentes series de producción, la optimización del rendimiento de las máquinas y herramientas relacionadas con el robot, la amortización de la inversión en plazos de tiempo cortos (como consecuencia de la sustitución de

la mano de obra, reducidas averías, menores pérdidas de material residual, poco mantenimiento, y mejor uso y mayor duración de las herramientas), mejora de la calidad de los productos fabricados, disminución de los stocks de productos fabricados, y de los plazos de entrega, y finalmente, la realización de trabajos en condiciones peligrosas para la salud humana.

El diseño de robots industriales depende de la aplicación prevista. Pero una de sus características principales es su versatilidad: la capacidad para realizar funciones distintas a las originalmente requeridas a lo largo de su vida útil. Este es el rasgo que distingue a un robot de una máquina automática.

El desarrollo de la robótica industrial no ha tenido como objetivo inmediato realizar una máquina a semejanza del ser humano. Su objetivo inmediato ha sido aumentar la productividad, la calidad, y disminuir los costes de producción. El industrial, por lo tanto, no incorpora en su diseño elementos antropomórficos como una variable fundamental.

En la estructura de un robot industrial se distinguen cuatro partes. El manipulador está configurado como un conjunto de elementos relacionados entre sí que permiten el movimiento de la herramienta que tiene adaptada como elemento terminal. Todo el movimiento está controlado por el controlador, que es un computador que gobierna, recoge y procesa la información generada por los sensores, y determina los movimientos y el funcionamiento de los órganos motrices.

El desarrollo de los microprocesadores, el aumento de su potencia y la reducción de su precio han permitido el desarrollo de robots de última generación que pueden relacionarse con el medio exterior, aprender de la experiencia y tomar decisiones en tiempo real. Por ello, la evolución ascendente de la potencia y versatilidad de los microprocesadores, y la evolución descendente de su precio, permite prever el desarrollo de robots cada vez más potentes y versátiles, que automaticen cada vez más tareas. La evolución de la Robótica Industrial va ligada a la evolución de los microprocesadores y al desarrollo de la inteligencia artificial.

2.2 CLASIFICACIÓN DE ROBOTS INDUSTRIALES

Un robot puede ser clasificado atendiendo a diferentes criterios o características. Algunas de éstas serán dependientes de su propia esencia, otras de la aplicación o tarea a que se destinan.

Clasificación atendiendo a la generación

La generación de un robot hace referencia al momento tecnológico en que éste aparece. Se pasa de una generación a la siguiente cuando se da un hito que supone un avance significativo en las capacidades de los robots. Aun siendo

una división subjetiva, es interesante porque permite hacerse una idea de cuán de avanzado es un robot.

Una posible clasificación en generación:

> **La primera generación** se extiende desde el comienzo de la robótica hasta los años ochenta. Se repite la tarea programada secuencialmente. No toma en cuenta las posibles alteraciones de su entorno.
>
> **La segunda generación** se desarrolla en los años 80 y es la que mayoritariamente se puede encontrar hoy en día en las industrias. Adquiere información limitada de su entorno y actúa en consecuencia. Puede localizar, clasificar y detectar esfuerzos y adaptar sus movimientos en consecuencia.
>
> **La tercera generación** está desarrollándose en estos días. Su programación se realiza mediante el empleo de un lenguaje natural. Posee capacidad para la planificación automática de tareas.

Clasificación atendiendo al área de aplicación

Desde el punto de vista del uso que se da al robot es posible clasificarlos bien en base al sector económico en el que se encuentran trabajando o bien en base al tipo de aplicación o tarea que desarrollan, independientemente de en qué sector económico trabajen.

La clasificación que hace la IFR divide las aplicaciones en Personales o domésticas, profesionales y en general aplicaciones I+D.

Clasificación atendiendo al tipo de actuadores

Dependiendo de cuál sea el tipo de energía utilizada por los ejes principales del robot, éste puede ser clasificado como neumático, hidráulico y eléctrico. La mayor parte de los robots actuales son de accionamiento eléctrico. Los accionamientos hidráulicos en particular pueden estar aconsejados cuando se precise disponer de una elevada capacidad de carga-peso del robot o cuando se precise disponer de aislamiento eléctrico entre el robot y el resto del sistema.

Clasificación atendiendo al número de ejes

Esta característica es aplicable a los robots o telerobot con cadena cinemática abierta (es decir, sería aplicable a los robots manipuladores, pero no lo sería, por ejemplo, a los robots móviles). Se entiende por eje cada uno de los movimientos independientes con que está dotado el robot. Puesto que de acuerdo a la definición ISO el robot manipulador industrial debe tener al menos 3 ejes y extendiendo esta condición a los robots de servicio manipuladores, se podrían encontrar robots de cualquier número de ejes superior o igual a 3. En la práctica, la mayor parte de los robots tienen 6 ejes, seguidos por los de 4. Los robots con más de 6 ejes son poco frecuentes, estando justificado este

número para aumentar la capacidad de maniobra del robot y siendo en muchas ocasiones telerobots.

Clasificación atendiendo a la configuración

Esta clasificación es sólo aplicable a robots o telerobots con cadena cinemática. La configuración de un robot queda definida por el tipo de movimientos permitidos entre dos eslabones consecutivos de la cadena. De acuerdo a esto, se tienen los siguientes tipos de configuraciones de robots: cartesiano, cilíndrico, polar o esférico, articular, SCARA, paralelo.

Clasificación atendiendo al tipo de control

Atendiendo al tipo de robot, la norma ISO 8373 y en consonancia, la IFR, distingue entre los siguientes:

Robot secuencial (ISO): Robot con un sistema de control en el que un conjunto de movimientos se efectúa eje a eje en un orden dado, de tal forma que la finalización de un movimiento inicia el siguiente.

En este tipo de robots sólo es posible controlar una serie de puntos de parada, resultando un movimiento punto a punto. Un ejemplo de ellos son los manipuladores neumáticos.

Robot controlado por trayectoria (ISO): Robot que ejecuta un procedimiento controlado por el cual los movimientos de tres o más ejes controlados, se desarrollan según instrucciones que especifican en el tiempo la trayectoria requerida para alcanzar la siguiente posición (obtenida normalmente por interpolación).

Los robots controlados por trayectoria permiten la realización de movimientos en los que puede ser especificado toda la trayectoria de manera continua.

Robot adaptativo (ISO): Robot que tiene funciones de control con sensores, control adaptativo, o funciones de control de aprendizaje.

De este modo el robot puede modificar su tarea de acuerdo a la información captada del entorno, por ejemplo, a través de un sistema de visión por computador o por sensores de fuerza o contacto.

Robot teleoperado (ISO): Un robot que puede ser controlado remotamente por un operador humano, extendiendo las capacidades sensoriales y motoras de éste a localizaciones remotas.

2.3 ESTRUCTURA MECÁNICA DE UN ROBOT

Un robot industrial está compuesto por una serie consecutiva de eslabones y articulaciones para formar una cadena cinemática abierta, la cual es la

estructura mecánica de un robot industrial. Por lo general, el número de grados de libertad del robot, coincide con el número de articulaciones de que se compone. Se trata de una cadena cinemática abierta cuando sólo hay un camino posible para llegar de cualquier eslabón a cualquier otro.

La cadena cinemática abierta está formada de la siguiente manera: la primera articulación sirve para formar la base; a continuación siguen conexiones sucesivas entre articulaciones y eslabones, en el extremo final del último eslabón no hay articulación, generalmente se destina a colocar la herramienta de trabajo para llevar a cabo una aplicación específica. El extremo final del robot no se encuentra conectado físicamente a la base.

Las articulaciones se construyen por medio de actuadores y representan la interconexión entre dos eslabones consecutivos. El movimiento de cada articulación puede ser de desplazamiento, de giro, o una combinación de ambos. Las articulaciones son típicamente de rotación (de revolución) o lineales (prismáticas). Una articulación de revolución permite la rotación relativa entre dos eslabones. Una articulación prismática permite el movimiento lineal relativo entre dos eslabones. Cada uno de los movimientos independientes que puede realizar cada articulación con respecto a la anterior, se denomina grado de libertad (GDL).

Se puede establecer una **analogía** entre la anatomía de **brazo humano** y la constitución física de un **brazo robot**, por lo que en ocasiones, para hacer referencia a los distintos elementos que componen el robot, se usan términos como cuerpo, brazo, codo y muñeca. La articulación de la base corresponde a la cintura. La articulación del hombro debe ser la de mayor capacidad con respecto a las otras articulaciones, ya que es la que mueve y soporta el peso de la articulación del codo y de la herramienta de trabajo, así como la carga de objetos que realice durante una determinada aplicación.

En la práctica, en robótica, solo se emplean las articulaciones de rotación y prismáticas. En caso de que algún robot tuviera alguna articulación con más de un grado de libertad, se podría asumir que se trata de varias articulaciones diferentes, unidas por eslabones de longitud nula.

Dependiendo del **tipo de articulaciones (lineales o rotacionales)** que se encuentran incluidas en la estructura mecánica en cinemática abierta de la base, hombro y codo del robot (sin incluir las articulaciones de la orientación de la herramienta de trabajo), se desprende la **clasificación general de robots manipuladores industriales**, también conocidos como brazos robots: antropomórfico, esférico, cilíndrico, SCARA y cartesiano.

La nomenclatura empleada en robots industriales para representar el tipo de movimiento que realizan sus articulaciones está dada de la siguiente manera: R significa articulación tipo rotacional, mientras que la letra P representa una articulación prismática. El orden en que se representan corresponde a las articulaciones de la base, hombro y codo, respectivamente.

En la Tabla 1 se muestra una clasificación de robots industriales:

ROBOT	CARACTERÍSTICAS
Antropomórfico (RRR)	3 articulaciones rotacionales
SCARA (RRP)	2 articulaciones rotacionales y 1 prismática
Esférico (RRP)	2 articulaciones rotacionales y 1 prismática
Cilíndrico (RPP)	1 articulación rotacional y 2 prismáticas
Cartesiano (PPP)	3 articulaciones prismáticas

TABLA 1. CLASIFICACIÓN DE ROBOTS INDUSTRIALES (FUENTE: MATLAB APLICADO A ROBÓTICA Y MECATRÓNICA)

Dos ejemplos de robots, que se encuentran en la Escuela de Ingenierías Industriales, se muestran a continuación. En la Figura 2.1, el Robot SCORBOT ER 4u, robot antropomórfico de 5 grados de libertad.

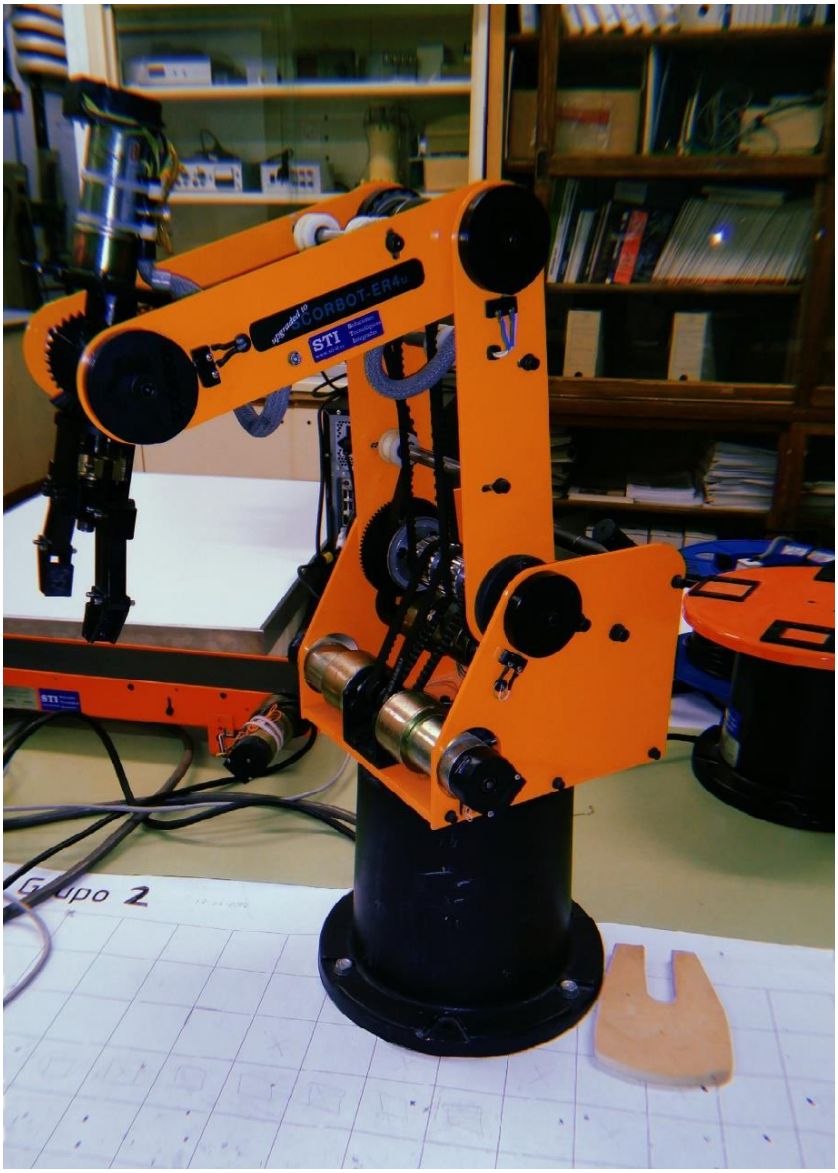


FIGURA 2.1. SCORBOT ER 4U

En la figura 2.2 , el Robot ABB (robot antropomórfico).

FIGURA 2.2. ROBOT ABB

2.4 REPRESENTACIÓN DE LA POSICIÓN Y ORIENTACIÓN DE UN SÓLIDO RÍGIDO

2.4.1 REPRESENTACIÓN DE LA POSICIÓN DE UN SÓLIDO RÍGIDO

Un punto queda totalmente definido en el espacio a través de los datos de su posición. Sin embargo, para el caso de un sólido rígido, es necesario además definir cuál es su orientación con respecto a un sistema de referencia. En el caso de un robot, no es suficiente con especificar cuál debe ser la posición de su extremo, sino que, en general, es también necesario indicar su orientación.

Ambas deben ser establecidas en relación a un sistema de referencia definido, pudiéndose hacer uso de diferentes modos o herramientas para especificar la relación entre la posición y orientación del cuerpo rígido y los sistemas de referencia.

Sistema cartesiano de referencia

Normalmente los sistemas de referencia se definen mediante ejes perpendiculares entre sí con un origen definido. Éstos se denominan sistemas cartesianos y en el caso de trabajar en el plano (2 dimensiones), el sistema de referencia OXY correspondiente queda definido por dos vectores coordenados OX y OY perpendiculares entre sí con un punto de intersección común O, tal como se ve en la Figura 2.3a.

Si trabajamos en el espacio (3 dimensiones), el sistema cartesiano OXYZ estará compuesto por una terna ortonormal de vectores unitarios OX, OY, OZ, tal y como se ve en la Figura 2.3b. Se trata de una terna ortonormal a derechas.

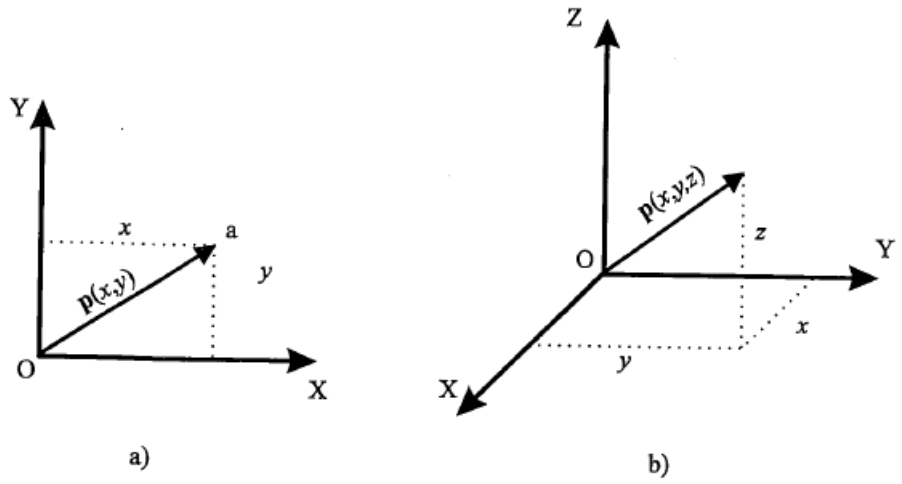


FIGURA 2.3. REPRESENTACIÓN DE UN VECTOR EN COORDENADAS CARTESIANAS EN 2 Y 3 DIMENSIONES

Coordenadas cartesianas

Si se trabaja en un plano, con su sistema coordenado OXY de referencia asociado, un punto a vendrá expresado por las componentes (x,y) correspondientes a los ejes coordenados del sistema OXY. Este punto tiene asociado un vector **p** (x,y), que va desde el origen O del sistema OXY hasta el punto a. Por tanto, la posición del extremo del vector **p** está caracterizada por las dos componentes (x,y), denominadas coordenadas cartesianas del vector y que son las proyecciones del vector **p** sobre los ejes OX y OY.

En el caso de que se trabaje en tres dimensiones, un vector viene definido con respecto al sistema de referencia OXYZ mediante las coordenadas correspondientes a cada uno de los ejes coordenados. En el caso de la Figura 2.4b el vector **p** estará definido por las componentes cartesianas (x,y,z).

Coordenadas polares y cilíndricas

Para un plano es posible también caracterizar la localización de un punto o vector **p** respecto a un sistema de ejes cartesianos de referencia OXY utilizando las denominadas coordenadas polares (r,θ). Donde r representa la distancia desde el origen O del sistema hasta el extremo del vector **p**, mientras que θ es el ángulo que forma el vector **p** con el eje OX. Se representa en la Figura 2.5a.

En el caso de trabajar en tres dimensiones (Figura 2.4b), un vector **p** podrá expresarse con respecto a un sistema de referencia OXYZ mediante las coordenadas cilíndricas **p** (r, θ, z). Las componentes r y θ tienen el mismo significado que en el caso de coordenadas polares, aplicado el razonamiento sobre el plano OXY, mientras que la componente z expresa la proyección sobre el eje OZ del vector **p**.

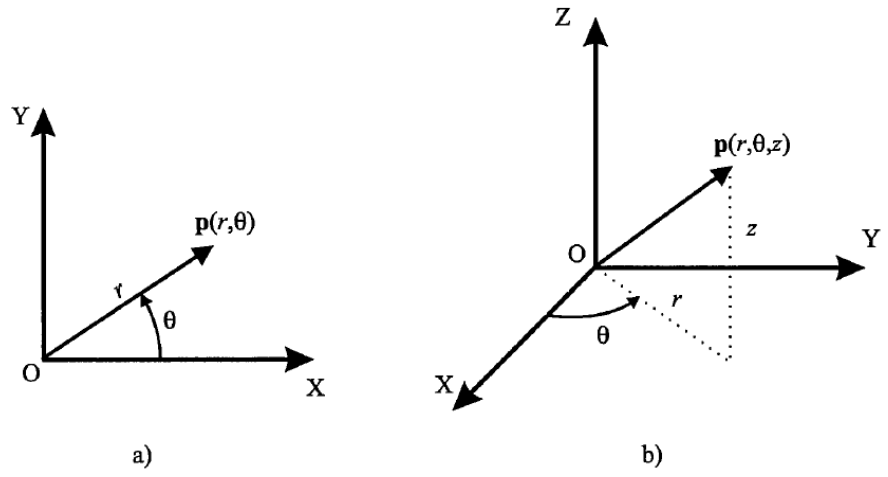


FIGURA 2.4. REPRESENTACIÓN DE A) COORDENADAS ESFÉRICAS Y B) CILÍNDRICAS

Coordenadas esféricas

También es posible utilizar coordenadas esféricas para realizar la localización de un vector en un espacio de tres dimensiones. Utilizando el sistema de referencia OXYZ, el vector **p** tendrá como coordenadas esféricas (r, θ, ϕ), donde la componente r es la distancia desde el origen O hasta el extremo del vector **p**; la componente θ es el ángulo formado por la proyección del vector **p** sobre el plano OXY con el eje OX; y la componente ϕ es el ángulo formado por el vector **p** con el eje OZ. Se representa en la Figura 2.5.

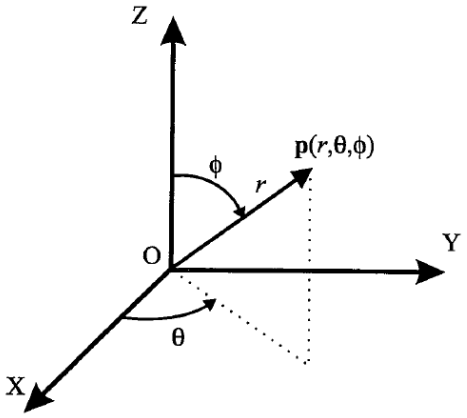


FIGURA 2.5. REPRESENTACIÓN DE UN VECTOR EN COORDENADAS ESFÉRICAS

2.4.2 REPRESENTACIÓN DE LA ORIENTACIÓN DE UN SÓLIDO RÍGIDO

Una orientación en el espacio tridimensional viene definida por tres grados de libertad o tres componentes linealmente independientes. Para poder describir de forma sencilla la orientación de un objeto respecto a un sistema de referencia, es habitual asignar solidariamente al objeto un nuevo sistema, y después estudiar la relación espacial existente entre los dos sistemas. De forma general, esta relación vendrá dada por la posición y orientación del sistema asociado al objeto respecto al de referencia. Para el análisis de los distintos métodos de representar orientaciones se supondrá que ambos sistemas coinciden en el origen, y que por tanto no existe cambio alguno de posición entre ellos.

Matrices de rotación

Las matrices de rotación son el método más extendido para la descripción de orientaciones debido principalmente a la comodidad que proporciona el álgebra matricial.

Supóngase que se tiene en el plano dos sistemas de referencia OXY y OUV con un mismo origen O, siendo el sistema OXY el de referencia fijo y el sistema OUV el móvil, solidario al objeto (Figura 2.6a). Los vectores unitarios de los ejes coordenados del sistema OXY son $\boldsymbol{i}_x, \boldsymbol{j}_y$, mientras que los del sistema OUV son $\boldsymbol{i}_u, \boldsymbol{j}_v$.

Un vector **p** del plano se puede representar como:

$$\mathbf{P} = p_u\,\mathbf{i}_\mathrm{u} + p_v\,\mathbf{j}_\mathrm{v} \qquad [2.1]$$

Además, se verifican las igualdades siguientes, por tratarse de productos escalares):

$$\begin{cases} p_x = \boldsymbol{i}_x\mathbf{P} \\ p_y = \boldsymbol{i}_v\mathbf{P} \end{cases} \qquad [2.2]$$

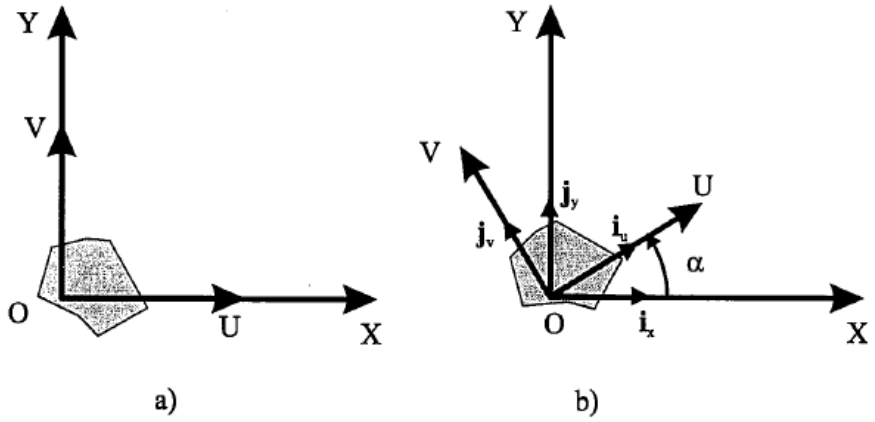


FIGURA 2.6. ORIENTACIÓN DE UN SISTEMA OUV RESPECTO A OTRO OXY EN UN PLANO

Sustituyendo la expresión [2.1] en [2.2] se obtiene:

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \mathbf{R}\begin{bmatrix} p_u \\ p_v \end{bmatrix} \qquad [2.3]$$

Donde:

$$\mathbf{R} = \begin{bmatrix} \mathbf{i}_\mathrm{x}\mathbf{i}_\mathrm{u} & \mathbf{i}_\mathrm{x}\mathbf{j}_\mathrm{v} \\ \mathbf{j}_\mathrm{y}\mathbf{i}_\mathrm{u} & \mathbf{j}_\mathrm{y}\mathbf{j}_\mathrm{v} \end{bmatrix} \qquad [2.4]$$

Es la llamada matriz de rotación, que define la orientación del sistema OUV con respecto al sistema OXY, y que sirve para transformar las coordenadas de un vector en un sistema a las de otro. También recibe el nombre de matriz de cosenos directores. Es fácil de comprobar que se trata de una matriz ortonormal, tal que $\mathbf{R}^{-1} = \mathbf{R}^\mathrm{T}$.

En el caso de dos dimensiones, la orientación viene definida por un único parámetro independiente. Si se considera la posición relativa del sistema OUV girado un ángulo α sobre el OXY (Figura 2.6b), tras realizar los correspondientes productos escalares, la matriz **R** de rotación será de la forma:

$$\mathbf{R} = \begin{bmatrix} \cos\alpha & -\operatorname{sen}\alpha \\ \operatorname{sen}\alpha & \cos\alpha \end{bmatrix} \quad [2.5]$$

Para el caso en que $\alpha = 0$, en el que los ejes coordenados de ambos sistemas coinciden, la matriz **R** corresponderá a la matriz identidad.

En un sistema tridimensional, el razonamiento es similar. Supóngase los sistemas OXYZ y OUVW, coincidentes en el origen, siendo el OXYZ, el sistema de referencia fijo, y el OUVW el solidario al objeto cuya orientación se desea definir (Figura 2.7a). Los vectores unitarios del sistema OXYZ serán $\boldsymbol{i}_x, \boldsymbol{j}_y, \boldsymbol{k}_z$, mientras que los del OUVW serán $\boldsymbol{i}_u, \boldsymbol{j}_v, \boldsymbol{k}_w$. Un vector **p** del espacio podrá ser referido a cualquiera de los sistemas de la siguiente manera:

$$\mathbf{p}_{uvw} = [p_u, p_v, p_w]^T = p_u \cdot \boldsymbol{i}_u + p_v \cdot \boldsymbol{j}_v + p_w \cdot \boldsymbol{k}_w \quad [2.6]$$

$$\mathbf{p}_{xyz} = [p_x, p_y, p_z]^T = p_x \cdot \boldsymbol{i}_x + p_y \cdot \boldsymbol{j}_y + p_z \cdot \boldsymbol{k}_z \quad [2.7]$$

Y al igual que en dos dimensiones, se puede obtener la siguiente equivalencia:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \mathbf{R} \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix} \quad [2.8]$$

Donde:

$$\mathbf{R} = \begin{bmatrix} \boldsymbol{i}_x \boldsymbol{i}_u & \boldsymbol{i}_x \boldsymbol{j}_v & \boldsymbol{i}_x \boldsymbol{k}_w \\ \boldsymbol{j}_y \boldsymbol{i}_u & \boldsymbol{j}_y \boldsymbol{j}_v & \boldsymbol{j}_y \boldsymbol{k}_w \\ \boldsymbol{k}_z \boldsymbol{i}_u & \boldsymbol{k}_z \boldsymbol{j}_v & \boldsymbol{k}_z \boldsymbol{k}_w \end{bmatrix} \quad [2.9]$$

Es la matriz de rotación que define la orientación del sistema OUVW con respecto al sistema OXYZ. Al igual que en dos dimensiones, también recibe el nombre de matriz de cosenos directores y se trata de una matriz ortonormal, tal que la inversa de la matriz **R** es igual a su traspuesta.

Es especialmente útil el establecer la expresión de la matriz de rotación correspondiente a sistemas girados únicamente sobre uno de los ejes del sistema de referencia.

En la Figura 2.7b, la orientación del sistema OUVW, con el eje OU coincidente con el eje OX, vendrá representada mediante la matriz:

$$\mathbf{Rot\ x}\ (\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\operatorname{sen}\alpha \\ 0 & \operatorname{sen}\alpha & \cos\alpha \end{bmatrix} \quad [2.10]$$

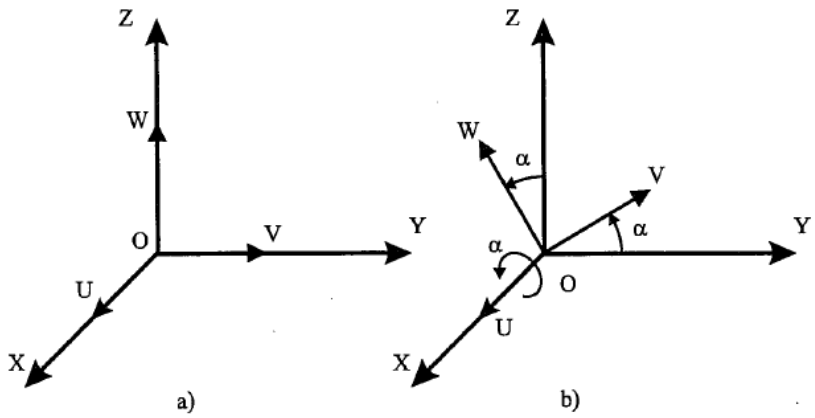


FIGURA 2.7. SISTEMA DE REFERENCIA OXYZ Y SOLIDARIO AL OBJETO OUVW

En la Figura 2.8a, la orientación del sistema OUVW, con el eje OV coincidente con el eje OY, vendrá representada mediante la matriz:

$$\mathbf{Rot\ y}\ (\varphi) = \begin{bmatrix} \cos\varphi & 0 & \text{sen}\,\varphi \\ 0 & 1 & 0 \\ -\text{sen}\,\varphi & 0 & \cos\varphi \end{bmatrix} \qquad [2.11]$$

En la Figura 2.8b, la orientación del sistema OUVW, con el eje OW coincidente con el eje OZ, vendrá representada mediante la matriz:

$$\mathbf{Rot\ z}\ (\theta) = \begin{bmatrix} \cos\theta & -\text{sen}\,\theta & 0 \\ -\text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad [2.12]$$

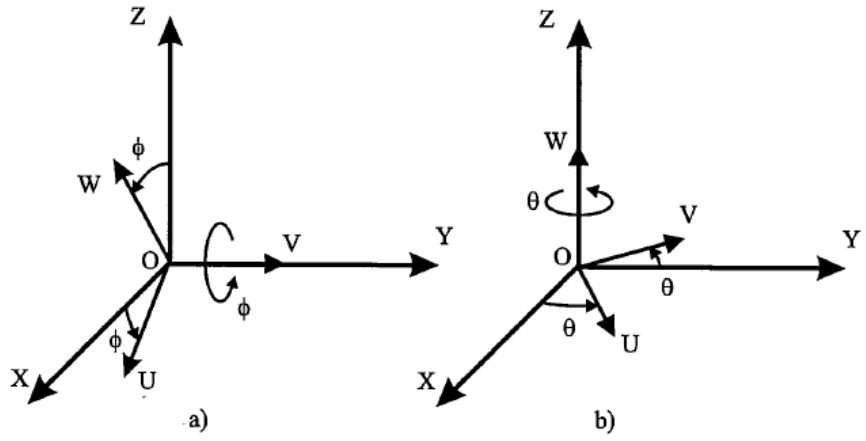


FIGURA 2.8. ROTACIÓN DEL SISTEMA DE REFERENCIA OUVW CON RESPECTO A LOS EJES OY Y OZ

Estas tres matrices, y sus Ecuaciones [2.10], [2.11] y [2.12], se denominan **matrices básicas de rotación** de un sistema espacial de tres dimensiones.

Composición de rotaciones

Las matrices de rotación pueden componerse para expresar la rotación continua de varias rotaciones. Así, una rotación en torno al eje x, seguida de una rotación en torno al eje y, y de una rotación en torno al eje z, globalmente se puede expresar como:

$$\mathbf{T} = \mathbf{Rotz}(\theta)\ \mathbf{Roty}\ (\varphi)\ \mathbf{Rotx}(\alpha) =$$

$$= \begin{bmatrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C\varphi & 0 & S\varphi \\ 0 & 1 & 0 \\ -S\varphi & 0 & C\varphi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\alpha & -S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix} =$$

$$= \begin{bmatrix} C\theta C\varphi & -S\theta C\alpha + C\theta S\varphi S\alpha & S\theta S\alpha + C\theta S\varphi C\alpha \\ S\theta C\varphi & C\theta C\alpha + S\theta S\varphi S\alpha & -C\theta S\alpha + S\theta S\varphi C\alpha \\ -S\varphi & C\varphi S\alpha & C\varphi C\alpha \end{bmatrix} \qquad [2.13]$$

Donde $C\theta$ expresa el coseno de θ y $S\theta$ expresa el seno de θ.

El producto de matrices no es conmutativo, por lo que el orden en que se realizan las rotaciones es relevante. Obsérvese, que los giros están definidos respecto de los ejes del sistema fijo OXYZ.

Ángulos de Euler

Para la representación de orientación en un espacio tridimensional mediante una matriz de rotación es necesario definir nueve elementos. Aunque la utilización de las matrices de rotación presente múltiples ventajas, los ángulos de Euler es un método de definición de orientación que hace únicamente uso de tres componentes para su descripción.

Todo sistema OUVW solidario al cuerpo cuya orientación se quiere describir, puede definirse con respecto al sistema OXYZ mediante tres ángulos denominados **ángulos de Euler** que representan los valores de los giros a realizar sobre tres ejes ortogonales entre sí, de modo que girando sucesivamente el sistema OXYZ sobre estos ejes ortonormales los valores de los ángulos de obtendrán del sistema OUVW. Es necesario, por tanto, conocer además de los valores de los ángulos, cuáles son los ejes sobre los que se realizan los giros. Existen diversas posibilidades, siendo las tres más usuales las que se muestran a continuación:

Ángulos de Euler WUW

Es una de las representaciones más habituales entre las que realizan los giros sobre ejes previamente girados. Se le suele asociar con los movimientos básicos de un giróscopo. Si se parte de los sistemas OXYZ y OUVW, inicialmente coincidentes, se puede colocar al sistema OUVW en cualquier orientación siguiendo los siguientes pasos (Figura 2.9).

1. Girar el sistema OUVW un ángulo ϕ con respecto al eje OZ, convirtiéndose así en el OU'V'W'.

2. Girar el sistema OU'V'W' un ángulo θ con respecto al eje OU' convirtiéndose así en el OU''V''W''.

3. Girar el sistema OU''V''W'' un ángulo ψ con respecto al eje OW'' convirtiéndose finalmente en el OU'''V'''W'''.

Es importante que estas operaciones se realicen en la secuencia especificada, pues las operaciones de giros consecutivos sobre ejes, no son conmutativas.

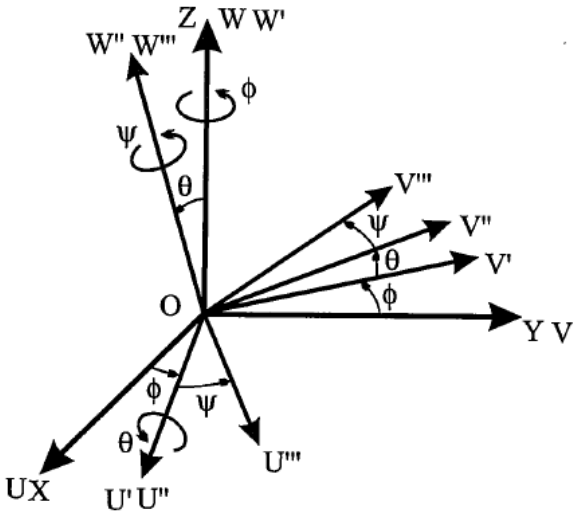


FIGURA 2.9. ÁNGULOS DE EULER WUW

Ángulos de Euler WVW

Es otra de las representaciones más habituales entre las que realizan los giros sobre ejes previamente girados. Sólo se diferencia del anterior en la elección del eje sobre el que se realiza el segundo giro. Si se parte de los sistemas OXYZ y OUVW, inicialmente coincidentes, se puede colocar el sistema OUVW en cualquier orientación siguiendo los siguientes pasos (Figura 2.10).

1. Girar el sistema OUVW un ángulo ϕ con respecto al eje OZ, convirtiéndose así en el OU'V'W'.

2. Girar el sistema OU'V'W' un ángulo θ con respecto al eje OV' convirtiéndose así en el OU''V''W''.

3. Girar el sistema OU''V''W'' un ángulo ψ con respecto al eje OW'' convirtiéndose finalmente en el OU'''V'''W'''.

Como antes, es preciso considerar que el orden de los giros no es conmutativo.

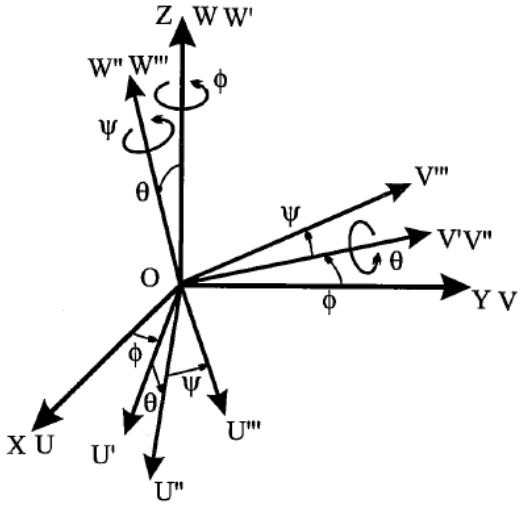


FIGURA 2.10. ÁNGULOS DE EULER WVW

Ángulos de Euler XYZ

Estos giros sobre los ejes fijos se denominan guiñada, cabeceo y alabeo (yaw, pitch, roll). Se trata de la representación utilizada generalmente en aeronáutica. Es también la más habitual de entre las que se aplican a los giros sobre los ejes del sistema fijo. Si se parte de los sistemas OXYZ y OUVW, al igual que en el caso anterior, se puede colocar al sistema OUVW en cualquier orientación siguiendo los siguientes pasos (Figura 2.11).

1. Girar el sistema OUVW un ángulo ϕ con respecto al eje OX (guiñada o yaw)

2. Girar el sistema OUVW un ángulo θ con respecto al eje OY (pitch o cabeceo)

3. Girar el sistema OUVW un ángulo ψ con respecto al eje OZ (roll o alabeo)

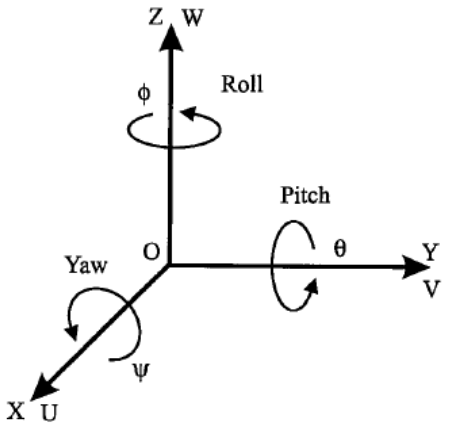


FIGURA 2.11. ÁNGULOS DE EULER XYZ (YAW, PITCH, ROLL)

Par de rotación

La representación de la orientación de un sistema OUVW con respecto al sistema de referencia OXYZ también puede realizarse mediante la definición de un vector **k** (kx, ky, kz) y un ángulo de giro θ, tal que el sistema OUVW corresponde al sistema OXYZ girado un ángulo θ sobre el eje **k**. El eje **k** ha de pasar por el origen O de ambos sistemas. Al par (**k**, θ) se le denomina par de rotación y se puede demostrar que es único.

Al igual que los ángulos de Euler, no se trata de un método que permita realizar una visualización sencilla de la orientación, salvo en casos muy concretos en los que el vector **k** coincida con algunos de los ejes coordenados del sistema OXYZ. Para la definición de orientación con este método, por tanto, es necesario definir cuatro parámetros distintos: kx, ky, kz y θ. Se puede representar como Rot (**k**,θ).

La aplicación de un par de rotación que rote un vector **p** un ángulo θ alrededor del vector unitario k se realiza a través de la siguiente expresión:

$$\mathbf{Rot}\,(\mathbf{k},\theta)\,\mathbf{p} = \mathbf{p}\cos\theta + (\mathbf{k}\times\mathbf{p})\,\mathrm{sen}\,\theta + \mathbf{k}(\mathbf{k}\cdot\mathbf{p})(1-\cos\theta) \qquad [2.14]$$

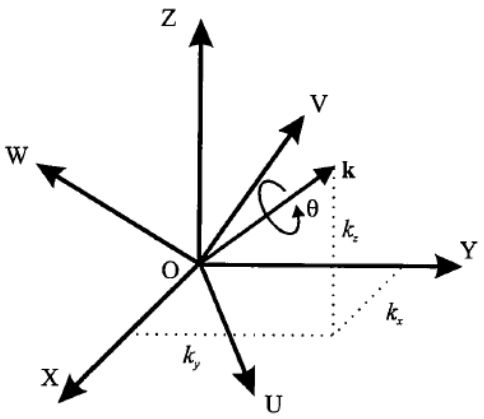


FIGURA 2.12. REPRESENTACIÓN DE ORIENTACIÓN POR EJE Y ÁNGULO DE GIRO (PAR DE ROTACIÓN)

Cuaternios

Los cuaternios, definidos por Hamilton, pueden ser utilizados como herramienta matemática de gran versatilidad computacional para trabajar con giros y orientaciones. En la bibliografía clásica sobre robótica suelen ser obviados o no tratados con el suficiente detalle, a pesar de ser empleados por algunos robots comerciales (ABB).

Un cuaternio Q está constituido por cuatro componentes $(q0, q1, q2, q3)$ que representan las coordenadas del cuaternio en una base {**e, i, j, k**}. Es frecuente denominar parte escalar del cuaternio a la componente en **e**: $q0$, y parte

vectorial al resto de componentes. De modo que un cuaternio se puede expresar como:

$$\mathrm{Q} = [\mathrm{q0}, \mathrm{q1}, \mathrm{q2}, \mathrm{q3}] = [s, \mathbf{v}] \qquad [2.15]$$

Donde s representa la parte escalar y $\mathbf{v}$ la parte vectorial.

Para la utilización de los cuaternios como metodología de representación de orientaciones se asocia el giro de un ángulo θ sobre el vector k al cuaternio definido por:

$$\mathrm{Q} = \mathrm{Rot}\,(\mathbf{k}, \theta) = (\cos(\theta/2)\,, \mathbf{k}\,\mathrm{sen}\,(\theta/2)) \qquad [2.16]$$

2.4.3 MATRICES DE TRANSFORMACIÓN HOMOGÉNEA

Anteriormente se han estudiado distintos métodos de representar la posición o la orientación de un sólido en el espacio. Pero ninguno de estos métodos por sí solo permite la representación conjunta de la posición y de la orientación (localización). Las matrices de rotación homogénea permiten la representación conjunta de posición y orientación, facilitando su uso mediante el álgebra matricial.

Coordenadas y matrices homogéneas

Un elemento de un espacio n-dimensional, se encuentra representado en coordenadas homogéneas por n + 1 dimensiones, de tal forma que un vector $\boldsymbol{p}(x, y, z)$ vendrá representado por $\boldsymbol{p}(wx, wy, wz, w)$, donde w tiene un valor arbitrario y representa un factor de escala. A partir de esta definición surge el concepto de matriz de transformación homogénea T, que se define como una matriz de dimensión 4x4 que representa la transformación de un vector de coordenadas homogéneas de un sistema de coordenadas a otro.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{3\mathrm{x}3} & \mathbf{p}_{3\mathrm{x}1} \\ \mathbf{f}_{1\mathrm{x}3} & \mathbf{w}_{1\mathrm{x}1} \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ \text{Perspectiva} & \text{Escalado} \end{bmatrix} \qquad [2.17]$$

Se puede considerar que una matriz homogénea se haya compuesta por cuatro submatrices: una submatriz $\boldsymbol{R}_{3x3}$ que corresponde a una matriz de rotación; una submatriz $\boldsymbol{p}_{3x1}$ que corresponde al vector de traslación; una submatriz $\boldsymbol{f}_{1x3}$ que representa una transformación de perspectiva, y una submatriz $\boldsymbol{w}_{1x1}$ que representa un escalado global.

En robótica generalmente sólo interesa conocer el valor de la matriz de rotación y del vector de traslación, considerándose la transformación de perspectiva nula y el escalado global unitario.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R}_{3\mathrm{x}3} & \mathbf{p}_{3\mathrm{x}1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \text{Rotación} & \text{Traslación} \\ 0 & 1 \end{bmatrix} \qquad [2.18]$$

Significado geométrico de las matrices homogéneas

La matriz T de transformación se suele escribir de la siguiente forma:

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [2.19]$$

Donde los vectores $\boldsymbol{n}, \boldsymbol{o}, \boldsymbol{a}$ forman una terna ortonormal que representa la orientación y el vector $\boldsymbol{p}$ representa la posición.

Si se aplica a un robot, la matriz de transformación homogénea permite describir la localización (posición y orientación) de su extremo con respecto a su base. Así, asociando a la base del robot un sistema de referencia fijo (OXYZ) y al extremo un sistema de referencia que se mueva con él, cuyo origen se encuentre en el punto p y los vectores directores son n, o, a escogidos de modos que:

$\boldsymbol{a}$ sea un vector en la dirección de aproximación del extremo del robot a su destino

$\boldsymbol{o}$ sea un vector perpendicular a a en el plano definido por la pinza del robot

$\boldsymbol{n}$ sea un vector que forme una terna ortogonal con los dos anteriores

Composición de matrices homogéneas

Una transformación compleja podrá descomponerse en la aplicación consecutiva de transformaciones simples (giros básicos y traslaciones).

De forma general, a la hora de componer diversas transformaciones mediante matrices homogéneas, se han de tener en cuenta los siguientes criterios:

1. Si el sistema de referencia fijo y el transformado son coincidentes, la matriz homogénea de transformación será la matriz identidad 4x4.

2. Si el sistema transformado se obtiene mediante traslaciones y rotaciones definidas con respecto al sistema fijo, la matriz homogénea que representa cada transformación se deberá premultiplicar sobre las matrices de las transformaciones previas.

3. Si el sistema transformado se obtiene mediante traslaciones y rotaciones definidas con respecto al sistema móvil, la matriz homogénea que representa cada transformación se deberá postmultiplicar sobre las matrices de las transformaciones previas.

Siguiendo estas indicaciones cualquier composición de matrices homogéneas puede estudiarse como si se realiza cada transformación con respecto al sistema fijo o con respecto al sistema móvil.

Por ejemplo, la transformación:

$$\mathbf{T} = \mathbf{Rotx}(\varphi)\mathbf{Rotz}(\gamma)\mathbf{Roty}(\theta) \qquad [2.20]$$

Puede verse como una rotación de θ sobre OY, seguida de una rotación γ sobre OZ y de una rotación φ sobre OX del sistema fijo.

2.4.4 COMPARACIÓN DE MÉTODOS DE LOCALIZACIÓN ESPACIAL

Todos los métodos expuestos son equivalentes, pero dependiendo del uso que se le vaya a hacer, será más adecuado emplear un procedimiento u otro.

La comparación se realiza, fundamentalmente, en razón a su capacidad para representar y manejar conjuntamente posición y orientación y su eficacia computacional.

Matrices de transformación homogénea

Sus principales ventajas residen en su capacidad de representación conjunta de posición y orientación y en la comodidad con la que se puede realizar la composición de transformaciones. Para ello basta con multiplicar, en el orden adecuado, las matrices de transformación correspondientes. Es posible, además, la aplicación de una transformación sobre un vector referido a un sistema fijo únicamente multiplicando la matriz de transformación correspondiente por el vector.

Como principal inconveniente presenta su alto nivel de redundancia (necesita definir 12 componentes para sólo 6 grados de libertad). Debe considerarse, además, que por los inevitables errores de redondeo, la multiplicación sucesiva de varias matrices de transformación homogénea, puede resultar en una matriz que no lo sea, lo que dificulta su implementación en computador.

Se trata del método más popular, ya que al trabajar con matrices permite el uso de su álgebra que es extensamente conocida.

Ángulos de Euler

Los ángulos de Euler, en cualquiera de sus modalidades, sólo son capaces de representar orientación, y aunque permiten una notación compacta (sólo tres números reales), no permiten la composición de rotaciones ni la aplicación sobre un vector de la rotación que definen.

Par de rotación

El par de rotación sólo sirve para la representación de orientaciones. Es una representación compacta, pues únicamente usa 4 parámetros para la definición de orientación de un sistema con respecto a otro. Se puede aplicar para obtener el efecto de la rotación de un vector r un ángulo θ alrededor del eje **k**. sin embargo, la composición de rotaciones presenta una expresión complicada, lo que limita su utilización práctica en algunas aplicaciones.

Cuaternios

El cuaternio, como tal sólo es capaz de representar la orientación relativa de un sistema O'UVW con respecto a otro, a través del uso de cuatro componentes. Sin embargo, es posible componer rotaciones junto con traslación de forma bastante simple y computacionalmente económica. Puede utilizarse también para obtener la transformación de un vector mediante rotaciones junto con traslaciones.

CAPÍTULO 3. BRAZO ROBÓTICO

3.1 CARACTERÍSTICAS DEL BRAZO ROBÓTICO

El brazo robótico que hemos escogido es un robot antropomórfico de 6 grados de libertad. Hemos adquirido un kit de montaje en el que viene por piezas (sin montar), con el resto de sus componentes, incluidos los servomotores que vamos a utilizar y toda la tornillería. Es un modelo de robot relativamente popular en proyectos de robótica, por lo que es fácil de encontrar.

Hemos aprovechado que el robot viene sin montar para medir las piezas de forma que queden completamente definidas, y comprobar las medidas necesarias antes del montaje. Las piezas se han medido con equipos de medida que se encuentran en el Laboratorio de Metrología de la Escuela de Ingenierías Industriales.

Las piezas son de chapa de aleación de aluminio y están mecanizadas y posteriormente dobladas, aunque no se proporcionan más detalles del proceso de fabricación. Hay un total de 41 piezas en el kit de montaje, la mayoría de dimensiones muy pequeñas. Podemos hacer una aproximación del peso total de las piezas del robot, modelando las piezas en el programa de diseño CATIA V5, que calcula el volumen de cada pieza.

El robot tiene 6 grados de libertad, todos pares de revolución, y además, el modelo escogido posee un cuadrilátero articulado, característica que lo destaca, a través del cual se mueve la tercera articulación del mismo.

Atendiendo al tipo de actuadores, nuestro robot es de accionamiento eléctrico. Cada articulación se moverá con un servomotor distinto. Es una cadena cinemática abierta con todas las articulaciones de tipo rotación (con un solo grado de libertad cada una). Las 4 primeras articulaciones se moverán con servos MG 996R, y las dos últimas lo harán con micro servos MG 90S.

Los servomotores se controlarán con el software y el hardware de Arduino. En concreto, la placa elegida es la Arduino MEGA 2560. Se harán conexiones en una placa de tiras soldada para comunicar los servos con Arduino (se explicará más adelante en el Capítulo 6). A su vez Arduino se comunica con un computador a través del puerto serie.

Se ha diseñado una caja base cuya función principal es proporcionar una base estable y móvil para el brazo robótico. Los requisitos a tener en cuenta se expondrán más adelante.

3.2 ACTUADORES ELÉCTRICOS

Las características de control, sencillez y precisión de los accionamientos eléctricos han hecho que sean los más utilizados en los robots industriales actuales.

Las ventajas y desventajas de un motor eléctrico son las siguientes:

Ventajas

- Amplia disponibilidad en el suministro de energía.
- El elemento de accionamiento básico en un motor eléctrico es normalmente más ligero que para la energía de fluidos.
- Alta eficiencia en la conversión de la energía.
- Sin contaminación del ambiente de trabajo.
- En relación con el costo, la precisión y repetibilidad de los robots accionados por energía eléctrica es comúnmente mejor que la de robots accionados por fluidos.
- Por ser relativamente silenciosos y limpios, resultan muy respetuosos con el ambiente.
- Son de fácil mantenimiento y reparación.
- Los componentes estructurales pueden ser de peso ligero.
- El sistema de accionamiento es muy conveniente para el control electrónico.

Desventajas

- Los robots de accionamiento eléctrico frecuentemente requieren la incorporación de algún tipo de sistema de transmisión mecánica. Esto agrega masa y movimiento no deseable que requieren energía adicional y que pueden complicar el control.
- Debido a la complejidad más alta del sistema de transmisión, se generan costos adicionales para su adquisición y mantenimiento.
- Los motores eléctricos no son intrínsecamente seguros. Por lo tanto, no pueden usarse, por ejemplo, en ambientes explosivos.

Las desventajas mencionadas anteriormente son paulatinamente superadas por la introducción de sistemas motrices de accionamiento directo, donde el motor eléctrico es una parte relevante de la articulación del brazo del robot, por lo que se eliminan los elementos de transmisión.

Además, la introducción de más recientes motores sin escobillas permite que se use los robots eléctricos en algunas aplicaciones de riesgo de incendios, tales como la pintura con pulverizador, puesto que se ha eliminado la posibilidad de chispas en las escobillas del motor. Los diferentes tipos de motores eléctricos son motores de paso, de CD y CA.

3.2.1 ACTUADORES ELÉCTRICOS: SERVOMOTORES

Los servomotores (comúnmente llamados servos) son sistemas electromecánicos que pertenecen a una clase particular de actuadores eléctricos (motores de corriente continua) encargados de transmitir energía para producir el movimiento del robot.

Un servomotor tiene la capacidad de ubicar su eje en una posición o ángulo determinado. Un servomotor está compuesto principalmente por 3 elementos: motor eléctrico, sensor de posición para medir el desplazamiento articular (rotacional o lineal), el cual está fabricado por un disco codificado y un ensamble de dispositivos emisores de luz y fotodefectores para generar una señal de salida proporcional al movimiento del motor, y un amplificador eléctrico (electronic driver) encargado de realizar el funcionamiento correcto del servomotor. También se llama servoamplificador y está constituido por un conjunto de microprocesadores y electrónica de potencia que se encarga de acoplar y acondicionar el motor la impedancia y señal de voltaje de baja potencia que proviene de la computadora o de un sistema mínimo digital.

Los robots pueden incluir sensores internos y externos. Los sensores internos proporcionan información sobre la posición, velocidad y aceleración de movimiento. Los sensores externos como fuerza, presión y visión, dotan al robot de un sistema de percepción.

En robótica los más usados son los de posición, que pueden ser analógicos y digitales. Los analógicos tienen una respuesta eléctrica (señal continua de voltaje) con información del movimiento del robot. Se requiere convertir esta señal a formato digital para que pueda ser procesada. Diversos sensores analógicos ya tienen integrado el sistema electrónico para convertir la respuesta analógica en formato digital y debido a que su señal de salida está digitalizada, se denomina sensor digital.

Ejemplos de sensores analógicos de posición son los resolvers, los potenciómetros y los tacómetros. Los sensores digitales de posición más comunes son los encoders. Un encoder se encuentra localizado en la parte posterior del rotor del motor para medir el desplazamiento articular.

Para el **uso de servomotores con Arduino**, la señal o dato que hay que enviarle al servo es una señal de PWM (modulación por ancho de pulso) donde el tiempo en alto es equivalente al ángulo o posición del servo. Estos valores pueden variar y van desde 0.5 a 1 milisegundo para la posición 0° y 2 a 2.4 milisegundos para la posición de 180°, el periodo de la señal debe ser cercano a 20 milisegundos. Se representa en la Figura 3.1.

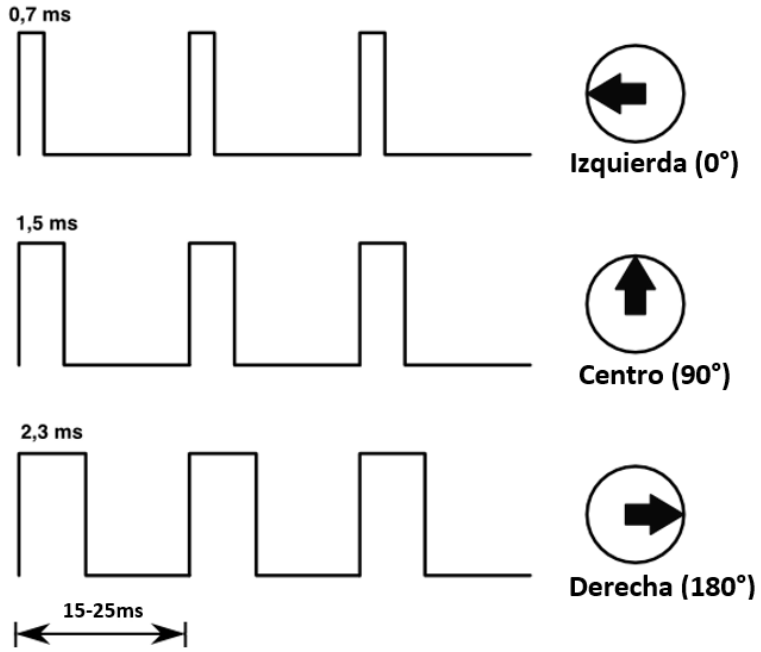


FIGURA 3.1. SEÑAL PWM PARA SERVOMOTORES (USO EN ARDUINO)

NOTA: Más adelante se explicará la modulación por ancho de pulso en Arduino (Capítulo 5).

3.2.2 MICRO SERVO MOTOR MG 90S TOWER-PRO

Fabricante: Tower-Pro

Características:

- Voltaje de operación: 4.8 V a 6 V
- Par de parada (par detenido): 1.8 kgf·cm (4.8 V), 2.2 kgf·cm (6 V)
- Velocidad de funcionamiento: 0.1 s/60º (4.8 V), 0.08 s/60º (6 V)
- Peso: 13.4 g
- Dimensiones compactas: Largo 22.8 mm, ancho 12.2 mm, altura 28.5 mm aprox.
- Rango de rotación: 180º
- Periodo: 20 ms (50Hz)
- Ancho de pulso: 1.0 – 2.5 ms
- Temperatura de operación: 0 – 55ºC
- Tipo de engranaje (piñonería): plástico
- Con cojinete
- Largo del cable: 25 cm aprox.
- Accesorios: brazos (cuernos) y tornillos de sujeción

A continuación, se indican en la Figura 3.2 las dimensiones del servo (en mm):

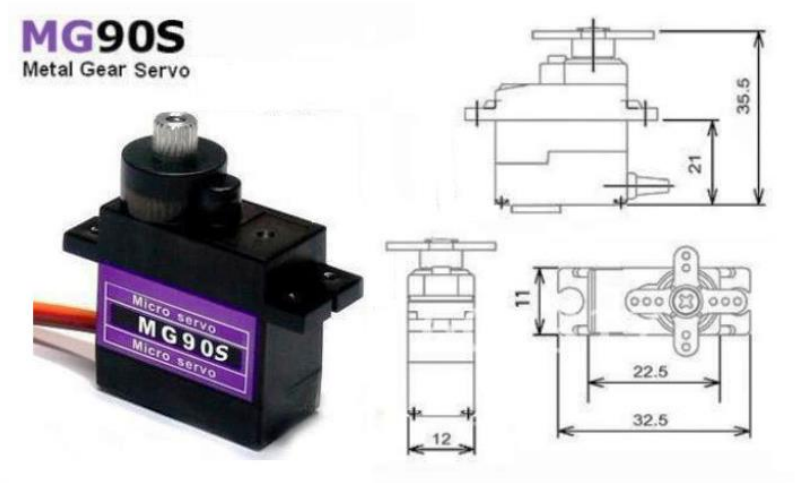


FIGURA 3.2. DIMENSIONES MICRO SERVO MG90S

En la Figura 3.3 se muestra la modulación por ancho de pulso (PWM) para este servo.

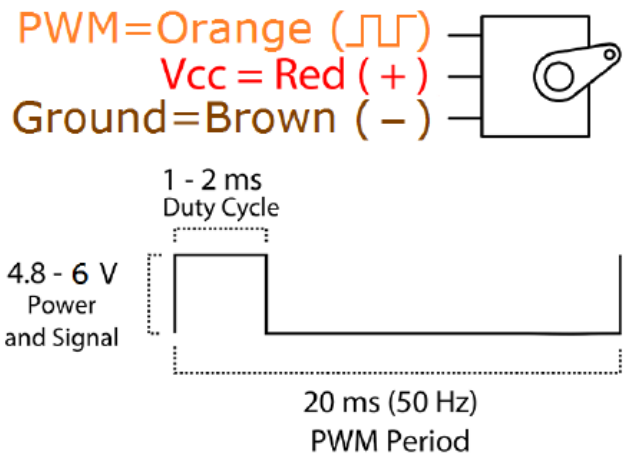


FIGURA 3.3. SEÑAL PWM MICRO SERVO MG90S

3.2.3 SERVO MOTOR MG 996R TOWER-PRO

Fabricante: Tower-Pro

Características:

- Voltaje de operación: 4.8 V a 7.2 V
- Rango de rotación: 180°
- Periodo: 20 ms (50 Hz)
- Ancho de pulso: 1.0 – 2.5 ms
- Temperatura de operación: 0 – 55°C
- Tipo de engranaje (piñonería): metálica
- Velocidad de funcionamiento: 0.17 s/60° (4.8 V), 0.14s/60° (6 V)
- Par de parada (par detenido): 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Con doble cojinete
- Peso: 55 g
- Dimensiones compactas: Largo 40.7 mm, ancho 19.7 mm, altura 42.9 mm aprox.
- Largo del cable: 31 cm aprox.
- Accesorios: brazos (cuernos) y tornillos de sujeción

A continuación, se indican en la Figura 3.4 las dimensiones del servo (en mm):

FIGURA 3.4. DIMENSIONES SERVO MG996R

En la Figura 3.5 se muestra la modulación por ancho de pulso (PWM) para este servo.

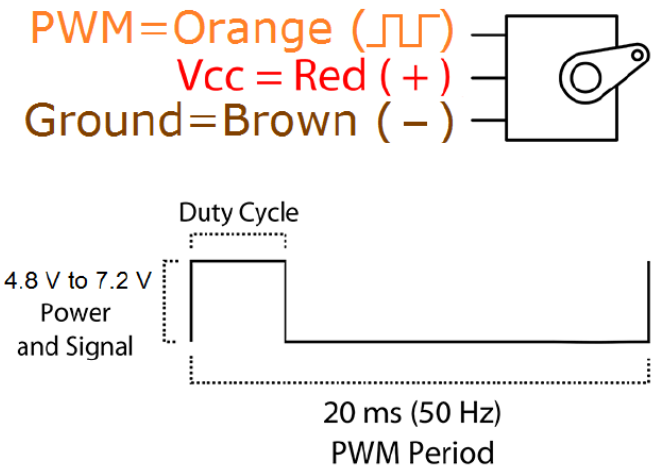


FIGURA 3.5. SEÑAL PWM SERVO MG996R

3.3 FUENTE DE ALIMENTACIÓN EXTERNA

Las características de la fuente de alimentación escogida para alimentar los servos son las que se especifican en la Figura 3.6:

Modelo: CWT – 235ATX

Corriente: vamos a utilizar +5 V, 22 A

Máxima potencia: 235 W

FIGURA 3.6. FUENTE DE ALIMENTACIÓN

Las dimensiones de la fuente de alimentación son las siguientes y se especifican en las Figuras 3.7 y 3.8: altura 86 cm, largo 150 cm, ancho 140 cm.

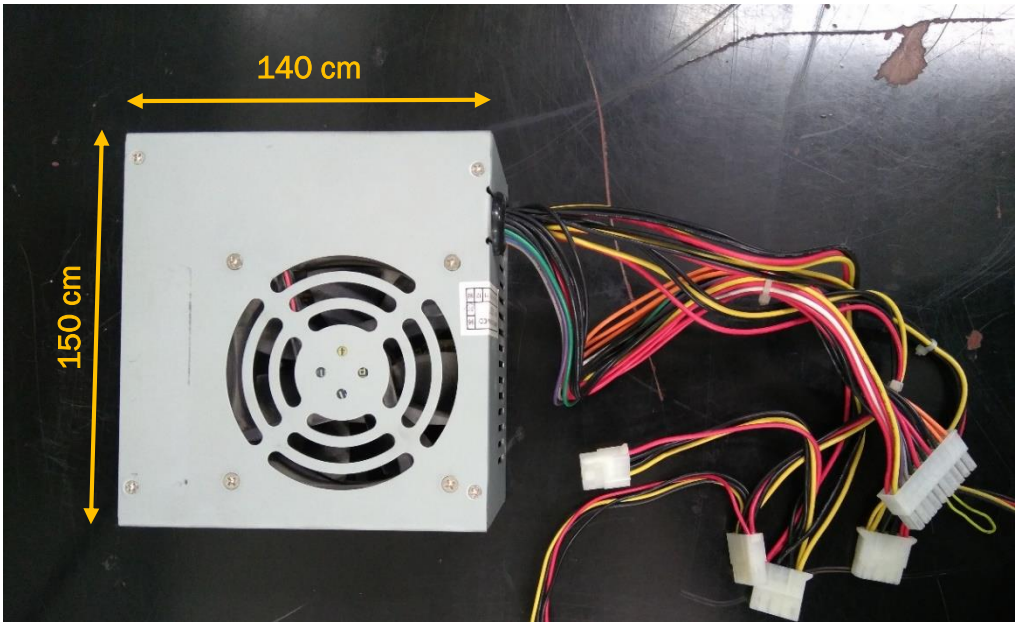


FIGURA 3.7. DIMENSIONES FUENTE DE ALIMENTACIÓN (1)

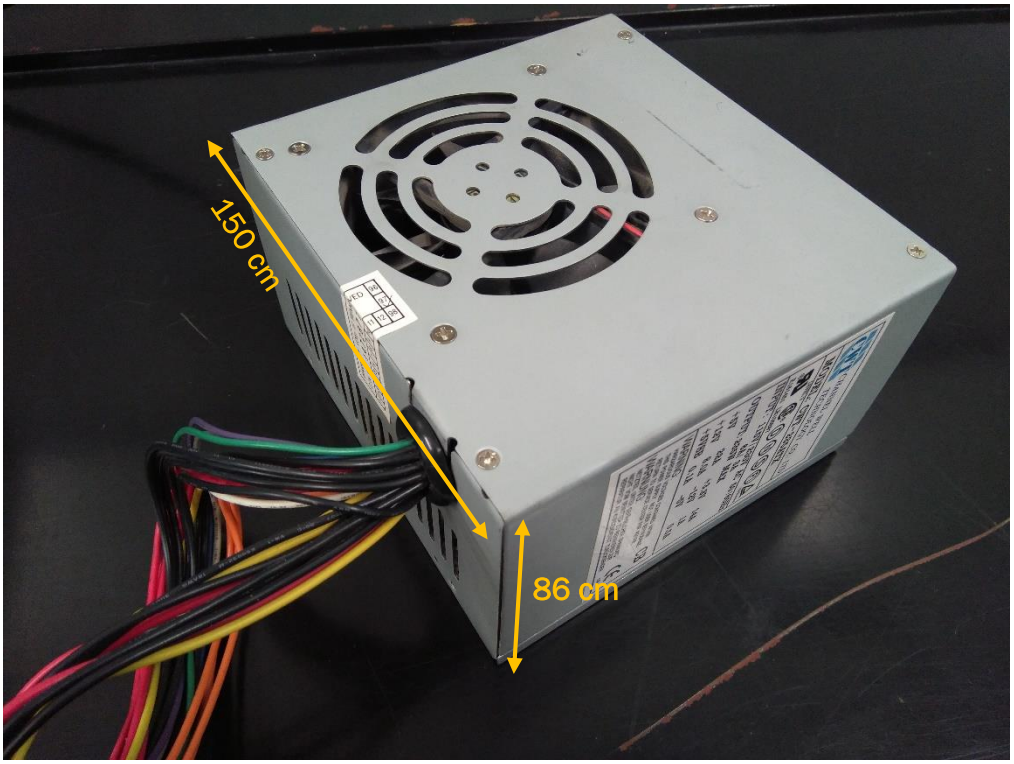


FIGURA 3.8. DIMENSIONES FUENTE DE ALIMENTACIÓN (2)

En la Figura 3.9 se muestra la toma de corriente de la fuente de alimentación y el botón de encendido/apagado.

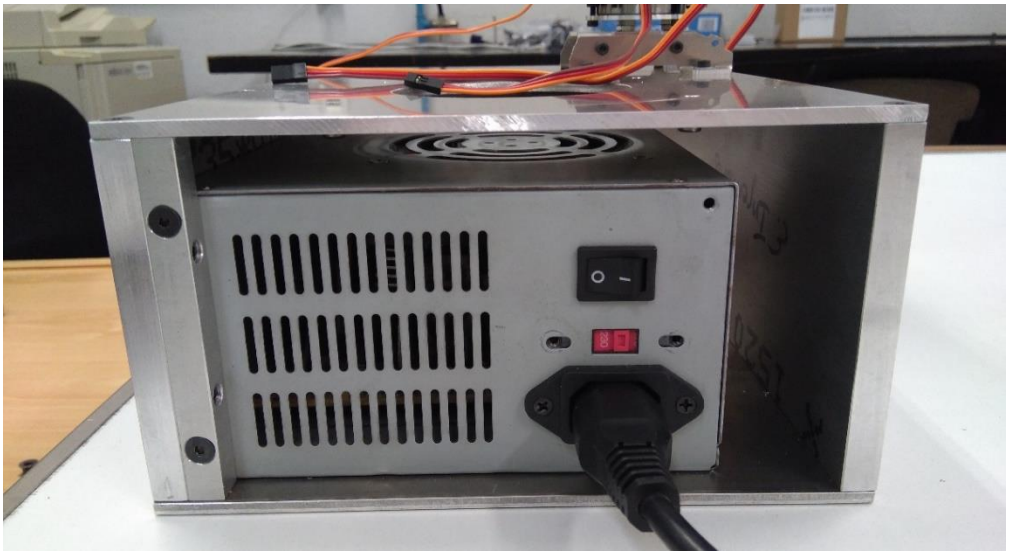


FIGURA 3.9. TOMA DE CORRIENTE FUENTE DE ALIMENTACIÓN

En la Figura 3.10 se muestra que la conexión de + 5V de la fuente de alimentación se hace con el cable rojo.

FIGURA 3.10. CONEXIÓN +5V FUENTE DE ALIMENTACIÓN

3.4 CAJA BASE DEL ROBOT

Una vez montado el robot, es necesario que la base se encuentre anclada en una superficie estable y que además, cuando el robot alcance las diferentes posiciones estando en movimiento, la base permanezca en la misma posición estable.

Por ello es necesario diseñar una caja base adecuada. La caja desempeñará, entre otras, la función de sujetar todo el peso del robot y de sus componentes.

El robot en su conjunto está compuesto no sólo por las piezas y la tornillería, sino también por los servomotores, por la fuente de alimentación para alimentarlos, por la placa Arduino, el cableado, etc. Por ello debemos tener en cuenta que todos estos elementos deberán estar en una determinada posición desde la cual podamos establecer una conexión entre todos ellos.

En concreto, la fuente de alimentación de los servos es un componente que debemos transportar junto con el robot. Además, vamos a aprovechar el peso de la fuente de alimentación para la base del robot que soportará el robot en movimiento.

Por todas estas razones, hemos decidido diseñar una caja base hueca y abierta en dos laterales, en cuyo interior reservaremos un espacio suficiente para ubicar la fuente de alimentación con todo el cableado.

El brazo robótico estará anclado a la caja mediante uniones roscadas (las piezas que forman la base del robot tienen los agujeros necesarios para pasar los tornillos). Estas uniones roscadas se harán sólo mediante tornillos de métrica M3 (y sus correspondientes arandelas), roscando los agujeros en la parte superior de la caja base del robot. La posición exacta de los agujeros roscados se especifica en el plano de la caja base, incluido en los anexos.

Como hemos visto, la fuente de alimentación necesita un sistema de ventilación, que se ubica en su parte superior. Vamos a realizar una perforación circular en la parte superior de la caja base para tal fin.

Además, para que la fuente de alimentación no se mueva dentro de la caja y la ventilación sea óptima, vamos a poner un tope en un lateral de la caja y una especie de rejilla que irá atornillada y ocupa el espacio del agujero de ventilación.

En cuanto a la altura de la caja base, debe ser suficiente para que quepa la fuente de alimentación. La altura de la fuente de alimentación es de 86 mm, por lo que hay que dejar un espacio a mayores para roscar dentro la rejilla y que la ventilación sea suficiente. Sin embargo, no debe ser demasiado alta, ya que uno de los propósitos de la caja base es también que el brazo robótico se encuentre a una cierta altura para poder alcanzar una superficie por debajo de la base del propio brazo.

No hemos medido exhaustivamente el peso del brazo robótico con todos sus componentes, ni el peso de la fuente de alimentación. El **material de la caja base** será de **acero** y teniendo en cuenta las dimensiones que son necesarias y el grosor que tendrá la caja, el peso de la misma junto con el de la fuente de alimentación es más que suficiente para soportar el brazo robótico en movimiento, quedando la base totalmente estable.

A continuación se resumen los **requisitos imprescindibles** que debe tener la caja base:

- Altura suficiente para que la base del brazo robótico quede a una altura tal que el TCP (punto central de la herramienta del robot) sea capaz de alcanzar la mesa de trabajo en su movimiento (alcance del robot).

- Además, debe tener un peso suficiente como para soportar el propio peso del brazo robótico, y al mismo en movimiento, de forma que la base quede totalmente estable.

- La caja junto con el robot debe poder transportarse de una posición a otra fácilmente.

- Debe tener espacio suficiente en su interior como para incluir la fuente de alimentación para los servos y la posibilidad de incluir otros componentes que formarán parte del robot.

- Debe tener una superficie adecuada para la ventilación de la fuente de alimentación.

- Debe ser posible su fabricación.

El diseño completo de la caja base se especifica en los planos, en concreto en el plano nº 34 dentro del anexo correspondiente.

Teniendo en cuenta todos estos requisitos se ha diseñado y construido la caja base, que se muestra a continuación en las Figuras 3.11, 3.12 y 3.13, en diferentes vistas, junto con el brazo robótico y la fuente de alimentación en su posición correspondiente.

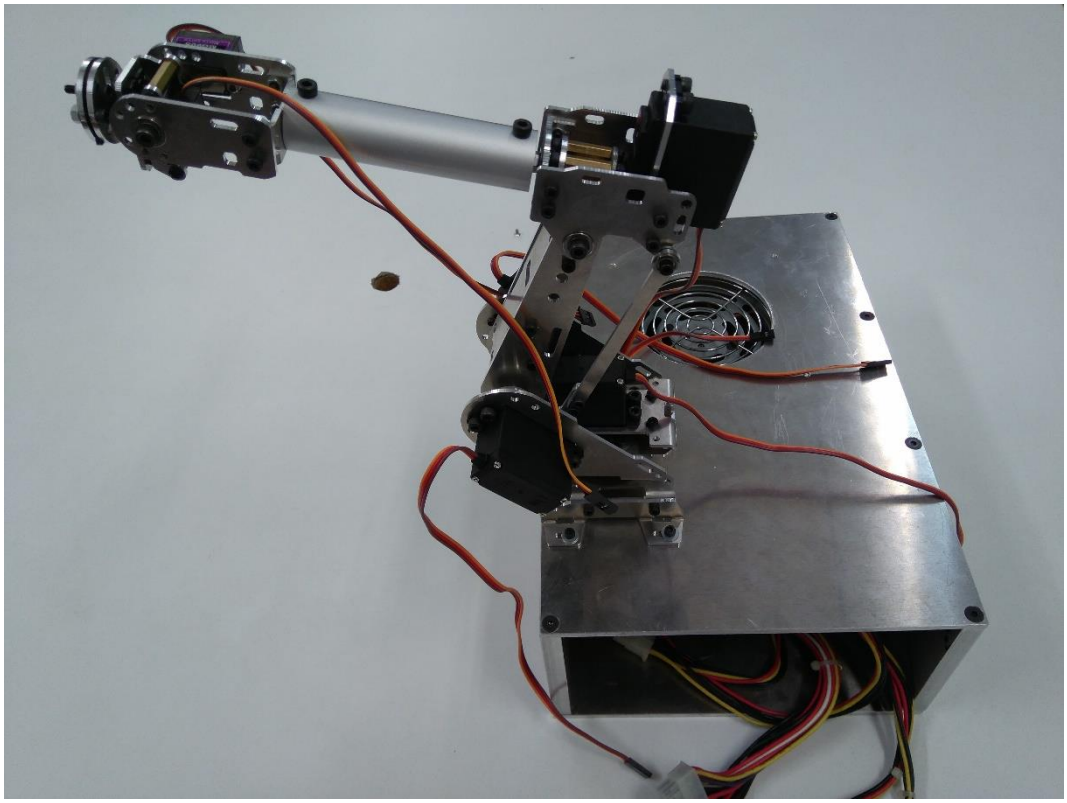


FIGURA 3.11. CAJA BASE CON EL BRAZO ROBÓTICO (1)

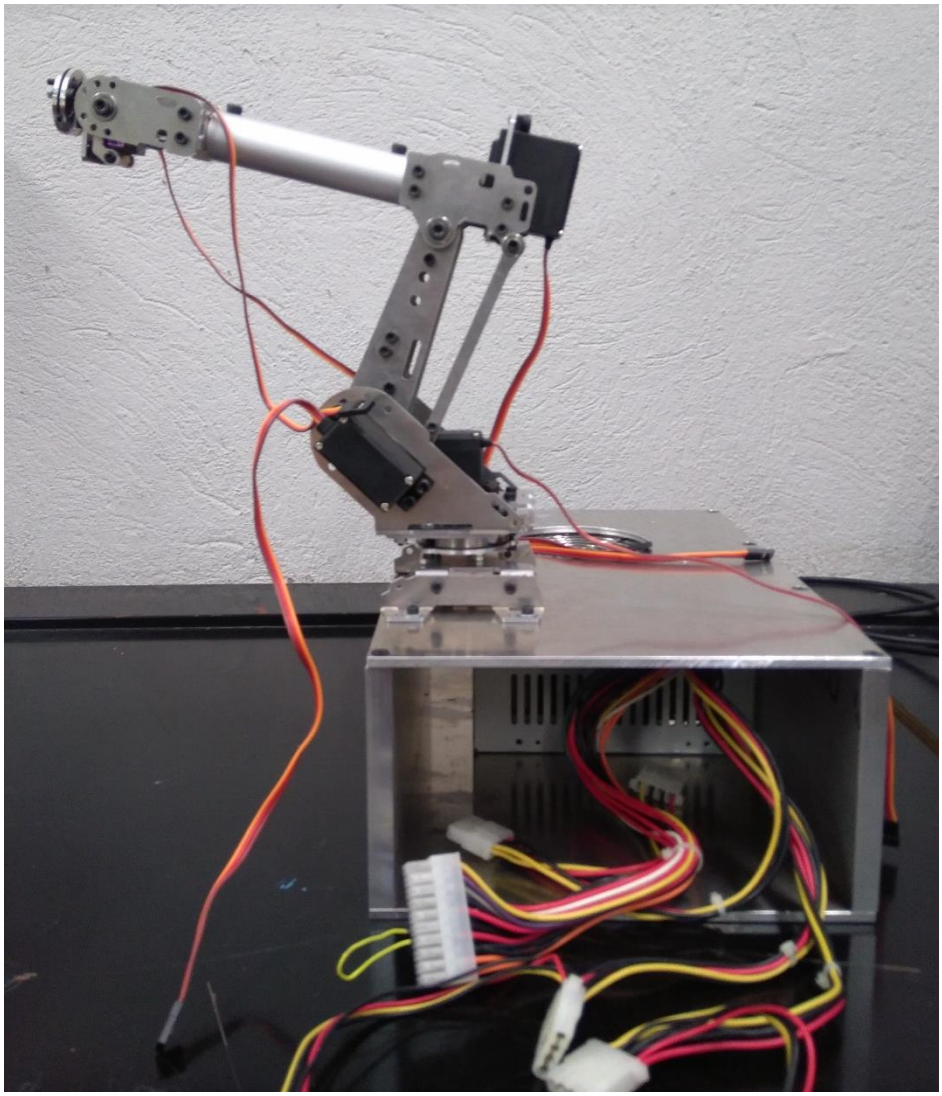


FIGURA 3.12. CAJA BASE CON EL BRAZO ROBÓTICO (2)

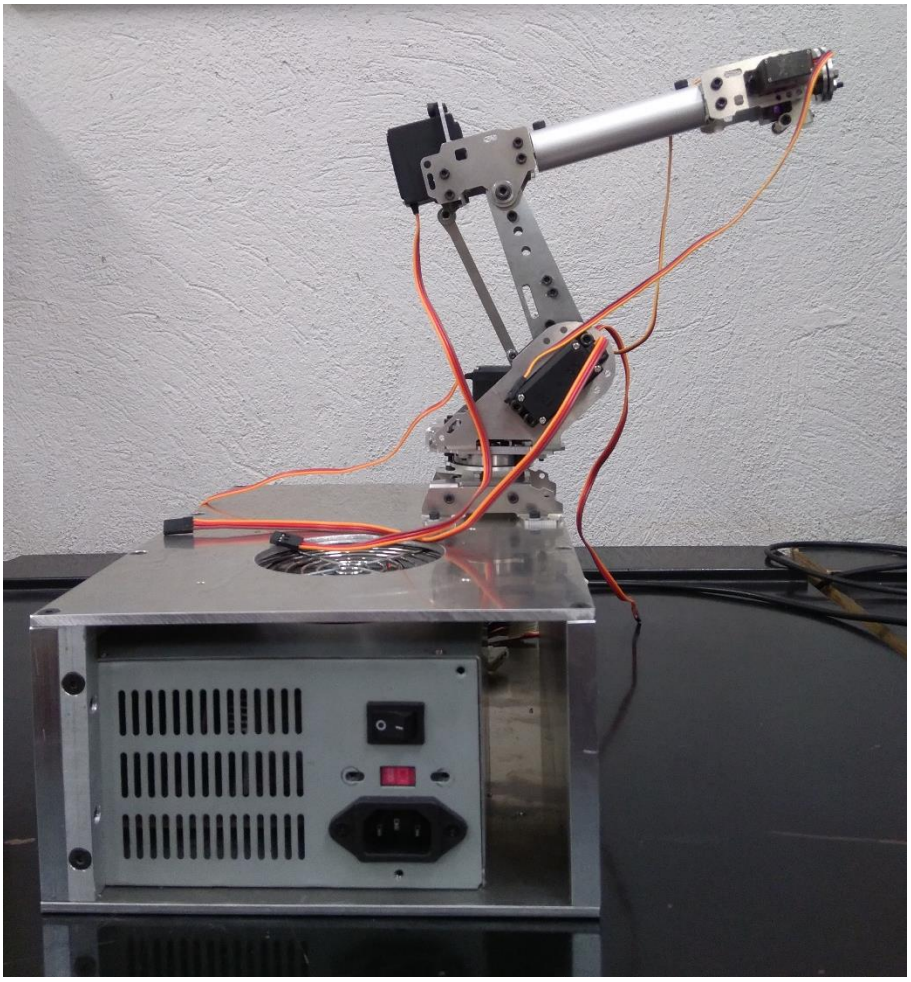


FIGURA 3.13. CAJA BASE CON EL BRAZO ROBÓTICO (3)

CAPÍTULO 4.

DIMENSIONAMIENTO DE LAS PIEZAS

4.1 INTRODUCCIÓN

Como se ha comentado anteriormente, vamos a aprovechar que el brazo robótico viene sin montar, para dimensionar las piezas completamente. En vistas a un futuro, tener las piezas del robot completamente definidas nos permitiría construir o diseñar un robot a una escala diferente tomando como referencia o copiando las piezas o las dimensiones de las piezas de este robot.

Para el estudio que nos ocupa, necesitaremos comprobar algunas medidas, y en especial determinar con una precisión suficiente aquellas que son necesarias para resolver el modelo cinemático.

Una vez tomadas todas las medidas, vamos a realizar el modelado de cada pieza con el programa de diseño CATIA V5. Posteriormente realizaremos los planos de cada pieza con este mismo programa y calcularemos su masa aproximada teniendo en cuenta el volumen y el material de las piezas (aluminio). Los planos están ordenados por orden de montaje (orden en el que se han ido necesitando las diferentes piezas para el montaje del robot, especificado en el anexo Instrucciones de Montaje).

4.2 MÉTODO DE MEDICIÓN

Se utilizan dos equipos de medida, que se encuentran en el Laboratorio de Metrología de la Escuela de Ingenierías Industriales. Las piezas tienen dimensiones y detalles de elementos pequeños que son imposibles de medir con instrumentos de medida tradicionales. Además necesitamos verificar distancias entre círculos, por lo que necesitaremos un equipo de medida sin contacto.

Existen numerosos equipos de medida sin contacto. En este caso, vamos a utilizar un **Proyector de Perfiles de eje horizontal**, dotado de un calculador geométrico (Quadra-Chek), y de un sistema de visión tridimensional (Deltec Vision) con su software específico. El proyector de perfiles proyecta mediante un haz de luz, que incide perpendicularmente a la pieza, el perfil de ésta en una pantalla mediante amplificación óptica.

El inconveniente de utilizar el proyector de perfiles, es que el haz de luz tiene que incidir perpendicularmente a la pieza y no disponemos de una base para sujetar las piezas que nos asegure la perpendicularidad. En este caso, hemos utilizado una pinza para sujetar las piezas, aunque al ser piezas tan pequeñas tampoco aseguramos la perpendicularidad. Las rebabas de las piezas también son un pequeño inconveniente a la hora de medir con el proyector de perfiles.

Aunque podemos utilizar el proyector de perfiles para medir completamente las piezas, para medir algunas dimensiones vamos a utilizar un **pie de rey analógico**, ya que en algunos casos es más fácil y rápido medir directamente con el pie de rey. El pie de rey también nos permite comprobar las dimensiones tomadas con el proyector de perfiles, entre caras paralelas por ejemplo.

Tanto el pie de rey como el proyector de perfiles tienen una resolución más que suficiente para nuestro propósito, por lo que vamos a considerar que las medidas tomadas son suficientemente fiables y no necesitamos hacer ningún cálculo de incertidumbre de los equipos de medida.

4.3 INVENTARIO DE EQUIPOS

Los equipos de medida utilizados son los siguientes:

Descripción: Proyector de Perfiles (PP) de eje horizontal

Marca: QUADRA-CHEK 200

Resolución: Eje X: 0,001 mm (digital), Eje Y: 0,001 mm (digital), angular: 1 s (digital)

Alcance de medida: Eje X: 200 mm, Eje Y: 200 mm, angular: 360º

Campo calibrado: Eje X: 200 mm, Eje Y: 200 mm, angular: 360º

RENAULT Nº401592

SIGMA M

Descripción: Pie de rey analógico con nonius longitudinal

Marca: METRICA

Modelo: -

Número de serie: -

Campo de medida: 0 – 150 mm

División de escala: 0,05 mm

A continuación se muestran unas imágenes de los equipos de medida utilizados. En las Figuras 4.1 y 4.2, el Proyector de Perfiles de eje horizontal y su calculador geométrico Quadra-Chek. En la Figura 4.3, el pie de rey analógico.

FIGURA 4.1. PROYECTOR DE PERFILES DE EJE HORIZONTAL

FIGURA 4.2. CALCULADOR GEOMÉTRICO QUADRA – CHEK (PP)

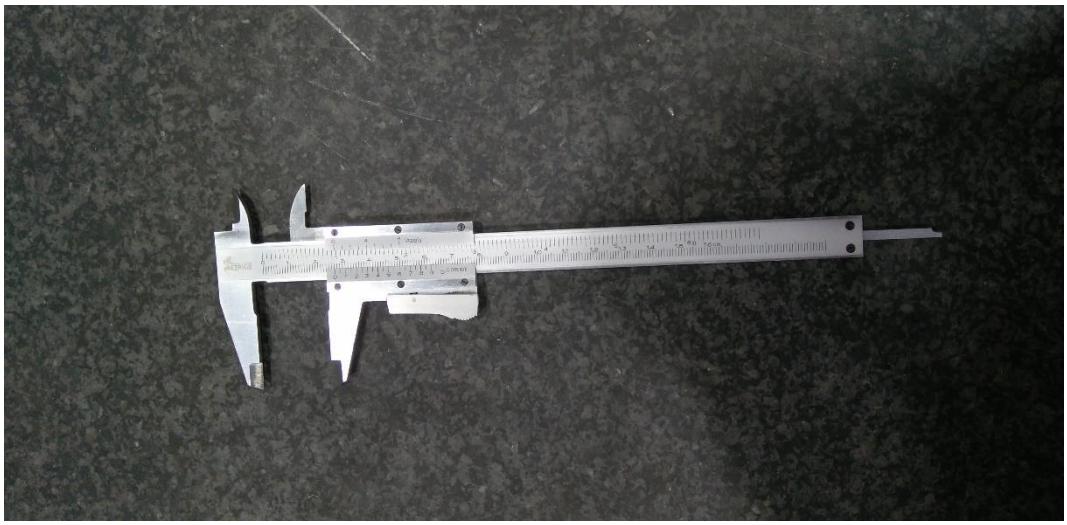


FIGURA 4.3. PIE DE REY ANALÓGICO

4.4 DETERMINACIÓN DE LA MASA

Las piezas han sido modeladas con el programa de diseño CATIA V5 (Dassault Systèmes), que nos permite calcular el volumen de cada pieza, como se ha comentado anteriormente.

A continuación se presenta una tabla donde se indica la masa aproximada de cada pieza, y la masa total de todas las piezas. El material de las piezas es aluminio y se toma como densidad la indicada.

La numeración de las piezas es la misma que indica en las Instrucciones de Montaje.

PIEZA	VOLUMEN (cm3)	DENSIDAD (Kg/cm3)	MASA (gramos)
Pieza 1	1,773	2,7	4,79
Pieza 2	1,594	2,7	4,30
Pieza 3	4,521	2,7	12,21
Pieza 4	1,209	2,7	3,26
Pieza 5	6,385	2,7	17,24
Pieza 6	2,956	2,7	7,98
Pieza 7	2,956	2,7	7,98
Pieza 8	5,244	2,7	14,16
Pieza 9	5,244	2,7	14,16
Pieza 10	5,112	2,7	13,80
Pieza 11	5,037	2,7	13,60
Pieza 12	4,235	2,7	11,43
Pieza 13	2,759	2,7	7,45
Pieza 14	0,762	2,7	2,06
Pieza 15	0,539	2,7	1,46
Pieza 16	2,850	2,7	7,70
Pieza 17	2,918	2,7	7,88
Pieza 18	1,541	2,7	4,16
Pieza 19	1,844	2,7	4,98
Pieza 20	0,576	2,7	1,56
Pieza 21	0,576	2,7	1,56
Pieza 22	0,576	2,7	1,56
Pieza 23	0,223	2,7	0,60
Pieza 24	4,341	2,7	11,72
Pieza 25	0,556	2,7	1,50
Pieza 26	1,156	2,7	3,12
Pieza 27	2,166	2,7	5,85
Pieza 28	1,614	2,7	4,36
Pieza 29	1,288	2,7	3,48
Pieza 30	0,543	2,7	1,47
Pieza 31	0,578	2,7	1,56
Pieza 32	0,223	2,7	0,60
Pieza 33	0,283	2,7	0,76
Pieza 34	1,157	2,7	3,12
Pieza 35	0,094	2,7	0,25
Pieza 36	0,932	2,7	2,52
Pieza 37	0,932	2,7	2,52
MASA TOTAL PIEZAS			208,69

TABLA 2. MASA DE LAS PIEZAS DEL BRAZO ROBÓTICO

4.5 COMPROBACIÓN DE LAS MEDIDAS

Si bien todas las medidas son necesarias para la definición de las piezas, debemos comprobar específicamente las medidas del cuadrilátero articulado que posee el robot y de algunas distancias que vamos a necesitar para el cálculo cinemático.

En la Figura 4.4 se muestra el **cuadrilátero articulado** una vez montado el brazo robótico.

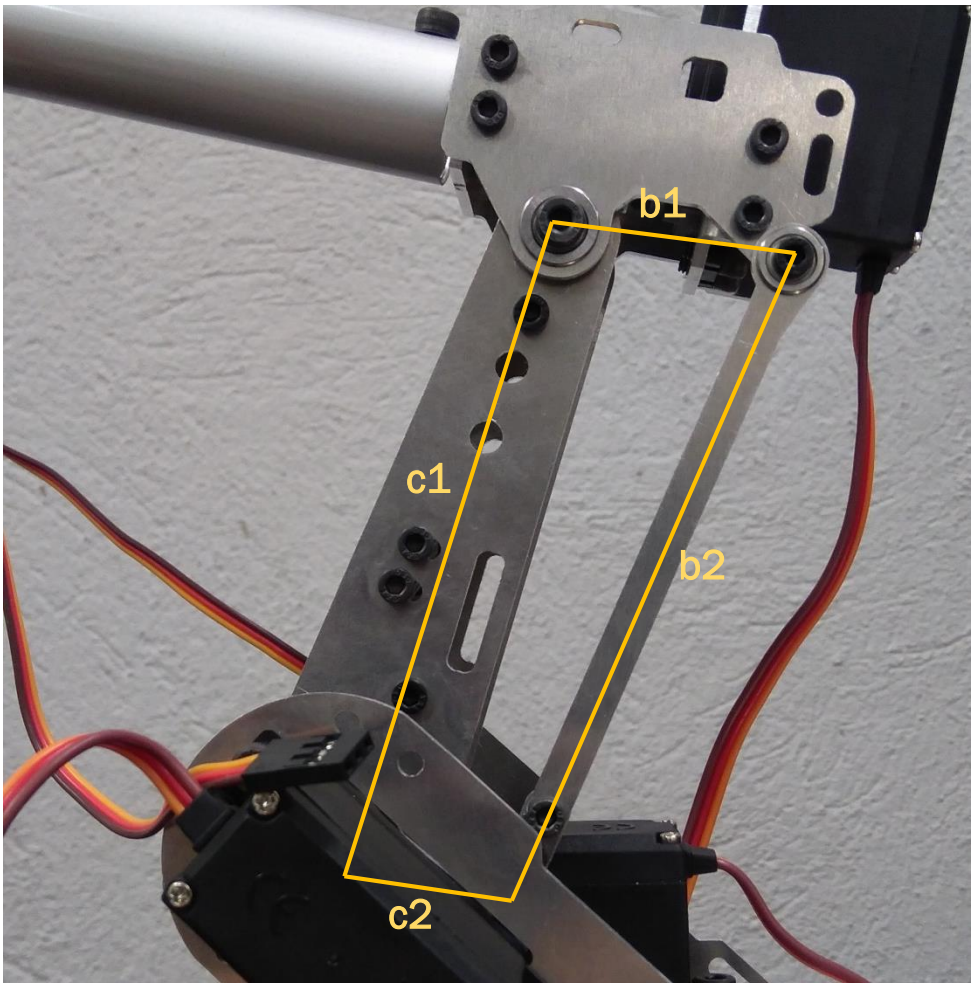


FIGURA 4.4. DIMENSIONES CUADRILÁTERO ARTICULADO

Las dimensiones se han obtenido de las piezas correspondientes:

Pieza nº 14 (nº plano 14): **c2 = 26,24 mm**
Pieza nº 10 (nº plano 10): **c1 = 95,2 mm**
Pieza nº 17 (nº plano 17): **b1 = 33 mm**
Piezas nº 33 y 34 (nº plano 30 y 31): **c2 = (91,3 – 6 + 17,67) mm = c2 = 102,97 mm**

Las **medidas del robot una vez montado** son las que se indican en la Figura 4.5 (la fuente de la imagen y las medidas es: micropede.de, donde también nos indican que **algunas distancias deben medirse según el ajuste de los componentes** del robot: je nach Justierung nachmessen).

Medidas en mm:

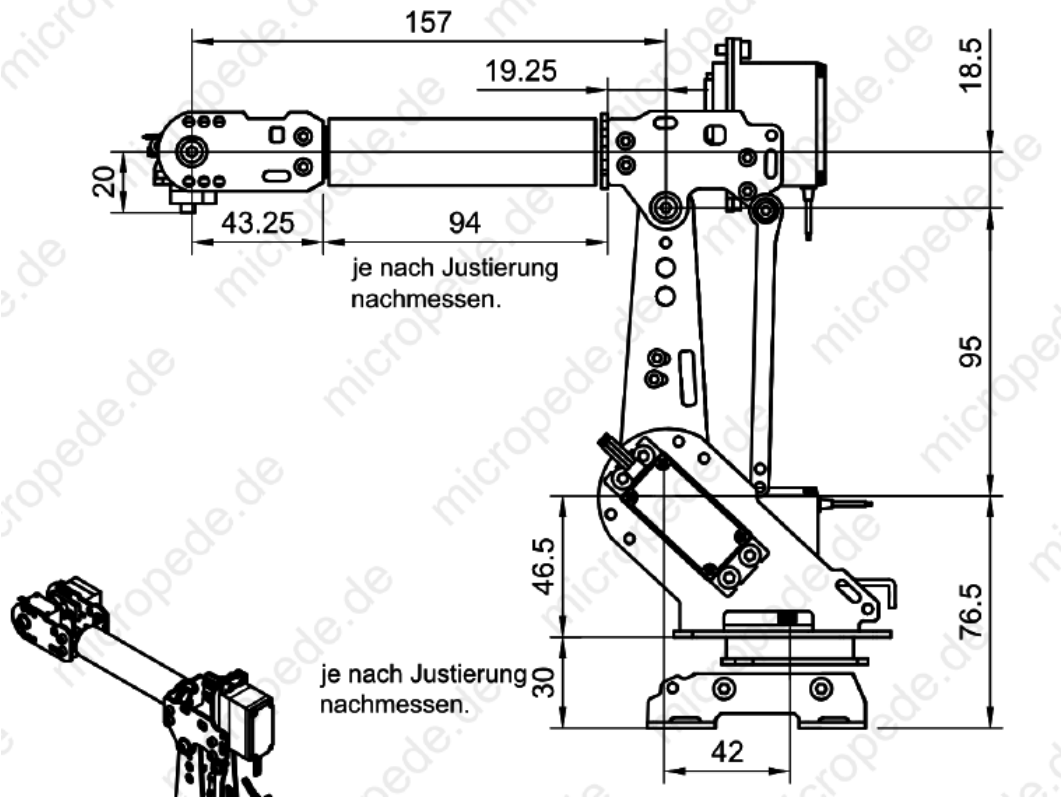


FIGURA 4.5. DIMENSIONES BRAZO ROBÓTICO (FUENTE: MICROPEDE.DE)

En nuestro caso, **una vez hecho el ajuste de los componentes, se ha comprobado que las medidas son las indicadas.**

CAPÍTULO 5. ARDUINO

5.1 INTRODUCCIÓN

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo integrado (IDE), diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares, desde sencillas aplicaciones domésticas hasta proyectos más complejos para la industria. Su principal ventaja es su facilidad de programación.

La placa Arduino se puede programar a través del IDE (entorno de desarrollo integrado) de Arduino.

5.2 MICROCONTROLADORES Y MICROPROCESADORES

Un **microprocesador** es un circuito integrado (también denominado chip), especializado en la ejecución de operaciones matemáticas, la transferencia y la manipulación de información. Estos chips son el cerebro de nuestros ordenadores y para funcionar necesitan memorias, discos, periféricos de distintos tipos, ratones, teclados, pantallas.

Un **microcontrolador** es un chip que contiene un pequeño microprocesador y lo mínimo imprescindible para su funcionamiento: memoria, algo que pueda funcionar como disco (flash memory), una memoria a largo plazo (EEPROM), un generador de frecuencia, convertidores analógico-digitales y sencillos periféricos para permitirles que interactúen con el mundo. Muchos microcontroladores tienen una potencia limitada, aunque también costes muy bajos. Por este motivo, los encontramos en muchos dispositivos que utilizamos en nuestro día a día. Gracias a una serie de innovaciones introducidas por ATMEL y Microchip en los años noventa, su uso es muy sencillo, lo que contribuye a su difusión. Un microcontrolador moderno no requiere aparatos especiales para ser programado y el firmware puede ser transferido al interior incluso si ya se ha soldado sobre un circuito impreso. Los microcontroladores modernos tienen la posibilidad de alojar un pequeño programa, denominado bootloader, que reside en un área de memoria especial.

El **bootloader** es un programa que puede escribir directamente las instrucciones en la memoria del chip. Así, no se necesita un programador particularmente complicado: con un simple puerto serie es posible programar el chip con el software que hemos escrito.

Existen diferentes familias de microcontroladores, que se distinguen por su nombre y la organización típica de sus circuitos internos, es decir, por la arquitectura del chip. Los chips con un conjunto de instrucciones muy numeroso se denominan CISC (Complex Instruction Set Computer). Se trata de circuitos de grandes dimensiones muy difíciles de diseñar y de realizar.

Por ello, nacieron los chips RISC (Reduced Instruction Set Computer), en los cuales el conjunto de instrucciones se redujo al mínimo indispensable. Los procesadores RISC más conocidos son los PIC, los AVR, los ARM o los SPARC. El chip que utiliza Arduino Uno es un ATmega328 de la familia AVR.

5.2.1 MICROCONTROLADORES ATMEL

La mayoría de los productos de Arduino utiliza **microcontroladores producidos por Atmel**. El más común es el ATmega328, utilizado para producir Arduino Uno. Atmel tiene una completa familia de microcontroladores ATmega, algunos con reducidas prestaciones, otros con mayor número de pines y funciones integradas.

Características principales del **microcontrolador ATmega2560**, que es el que utiliza **Arduino Mega 2560**:

- **Velocidad**: la frecuencia de trabajo máxima del chip es de 16 MHz.
- **RAM**: el tamaño de la memoria volátil, la que se utiliza para realizar cálculos y operaciones, es de 8 kB.
- **Flash**: la memoria que almacena nuestro programa, es de 256 kB.
- **EEPROM**: la memoria permanente donde se guardan los parámetros o configuraciones que no se deben modificar a menudo (sobre todo porque tiene un número reducido de escrituras), es de 4096 B.
- **GPIO**: indica el número de pines que pueden ser utilizados como entradas o salidas digitales. Son 86, para el microcontrolador ATmega2560.
- **Pines analógicos**: es el número de entradas analógicas que proporciona el chip. Estos pines corresponden a otros convertidores analógico-digitales, es decir, circuitos internos del microcontrolador que transforman una señal analógica en un número entero que puede ser utilizado en los programas. Atmega2560 tiene 16 pines analógicos.
- **PWM (Pulse Width Modulation)**: algunos pines pueden generar un determinado tipo de señal de onda cuadrada, muy eficaz para controlar el funcionamiento de motores y luces LED. ATmega tiene 16 pines PWM.
- **Puertos serie**: algunos microcontroladores ofrecen una o más líneas serie, útiles para intercambiar datos con otros dispositivos o para programar el propio chip. ATmega2560 tiene 4 puertos serie.
- **Alimentación:** para éste microcontrolador, el intervalo de tensiones de alimentación admisible es de 1,8 – 5,5 V.

5.3 HARDWARE

Arduino es una placa con un microcontrolador. Un microcontrolador es un pequeño ordenador, como se ha explicado anteriormente: en su interior se encuentra un microprocesador y una serie de dispositivos integrados que funcionan como memoria, disco y periféricos para permitir su comunicación con el mundo exterior.

La placa Arduino Uno utiliza un microcontrolador denominado ATmega328, producido por Atmel. Para programar Arduino se necesita solo un cable USB, porque de la programación se ocupa un pequeño circuito, que se encuentra ubicado en la misma placa electrónica, y un programa especial precargado en el ATmega328.

5.3.1 ARDUINO UNO (O GENUINO UNO)

No existe un único Arduino, sino muchos modelos, cada uno adecuado para exigencias concretas y con propiedades específicas. Seguidamente se describe el modelo **Arduino Uno**: la placa más conocida y común (Figura 5.1). Casi todos los otros modelos de Arduino presentan la misma disposición de las conexiones, para que haya compatibilidad con las placas de expansión. Los pines de Arduino con accesibles sencillamente insertando cables eléctricos con las extremidades peladas en unas pequeñas filas de orificios denominadas header.

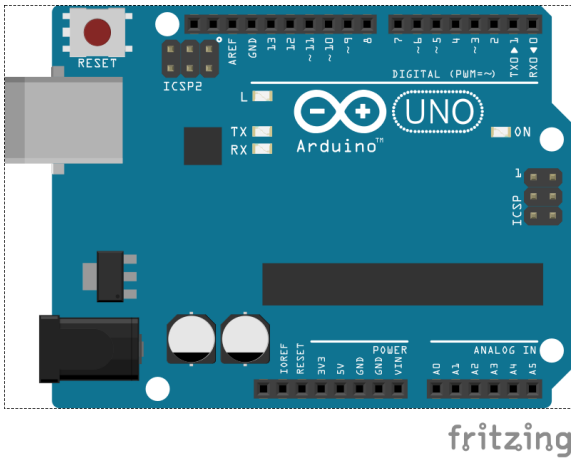


FIGURA 5.1. ARDUINO UNO (IMAGEN CREADA CON FRITZING)

Conector USB

Tiene una finalidad doble: se utiliza para alimentar la placa mediante los 5 V presentes en los puertos USB, pero también se puede usar para intercambiar datos con Arduino.

Alimentación

En un ángulo del circuito impreso se ve una pequeña clavija de color negro. En ella se puede insertar un jack de 5 mm (con un diámetro interior de 2,1 mm) para alimentar la placa. La tensión proporcionada será nivelada a 5 V por un regulador de voltaje.

GND

GND significa ground, es decir, tierra o masa. Es el polo negativo de alimentación, también denominado <<común>> o <<0 voltios>>. Entre los header de arduino existen hasta tres posiciones para insertar conexiones a tierra.

5 V

Este pin proporciona la tensión a 5 V regulada y estabilizada por Arduino. El estabilizador interno puede proporcionar hasta casi 1 amperio de corriente.

3,3 V

Muchos chips y sensores modernos se alimentan a 3,3 V, una tensión inferior respecto a los 5 V, que produce menos calor.

VIN

Este pin va conectado directamente a la entrada de alimentación y se puede utilizar para conseguir una tensión de alimentación más elevada, necesaria para hacer funcionar componentes conectados a Arduino.

AREF

Arduino puede leer tensiones analógicas comprendidas entre los 0 y los 5 V que después transformará en un número entre 0 y 1024. El paso mínimo, es decir, la precisión de la medida, será igual a 5 V dividido entre 1024, es decir, 4,88 milivoltios. Arduino utiliza siempre una tensión de referencia interna, obtenida de la tensión de alimentación que podría no ser demasiado precisa y que debería valer 5 V (pudiendo encontrar sólo 4,8 V. La tensión aplicada en AREF nunca debe ser superior a 5 V.

Reset

La placa cuenta con una tecla para resetearla. Al pulsarla, la ejecución del programa se detendrá y todo empezará de cero, como si acabáramos de encender Arduino.

Pin 0 – 13

Arduino tiene 14 pines que pueden ser configurados para funcionar como entrada o como salida. La configuración se lleva a cabo en el software. Arduino es un dispositivo digital, por lo que los pines pueden generar o leer un valor alto o bajo y, por tanto, igual a 0 o 5 V. un pin digital configurado como salida nunca podrá proporcionar, por ejemplo, un valor de 2,5 V, si no solo de 0 o 5 V. Algunos pines muestran junto al número de identificación una pequeña onda denominada tilde (~): estos pines pueden generar una señal particular muy útil para hacer funcionar motores eléctricos o para ajustar la intensidad luminosa de un LED. Esta señal eléctrica especial se denomina pulse-width modulation (PWM). Los pines que pueden proporcionar una señal PWM son: 3, 5, 6, 9, 10, 11.

Los pines 0 y 1 aparecen también marcados como TX0 y RX0 y están conectados a los puertos serie del chip: pueden ser utilizados para conectar Arduino a cualquier otro dispositivo que disponga de un puerto serie.

A0 – A5

Arduino tiene 6 pines capaces de leer niveles analógicos y convertirlos en un número que podrá ser utilizado dentro de los sketch. Estas entradas especiales se ubican a parte y están marcadas con las siglas A0, A1, A2, A3, A4, A5.

5.3.2 ARDUINO MEGA 2560

Cuando Arduino no es suficiente, podemos recurrir a **Arduino Mega** (Figura 5.2). Cuenta con unos 54 puertos de entrada/salida (pines digitales), 16 entradas analógicas y 4 puertos serie. Además, es compatible en la forma, las tensiones, y la velocidad (16 Hz) con la placa Uno. Para poder ofrecer tantos puertos, utiliza un **microcontrolador ATmega2560**. La gestión del puerto USB se confía a un segundo microcontrolador, un ATmega16u2, y se realiza completamente mediante un software.

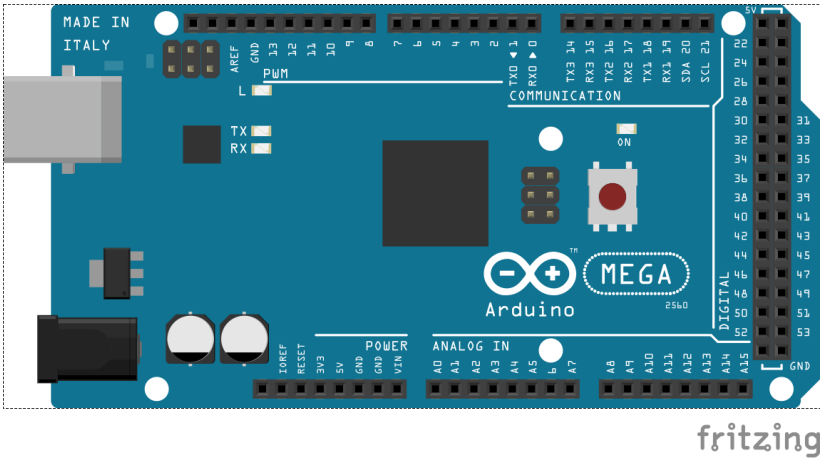


FIGURA 5.2. ARDUINO MEGA 2560 (IMAGEN CREADA CON FRITZING)

Arduino Mega 2560 es la placa utilizada para el presente proyecto. Una de las razones por las que se ha escogido esta placa es el número de pines digitales con capacidad PWM (16 pines), necesarios 6 en nuestro caso concreto para conectar los servomotores. En general, posee mejores prestaciones que Arduino Uno.

La placa adquirida no ha sido la original fabricada por Arduino. Arduino suministra los esquemas necesarios para fabricar sus placas y vende los microprocesadores. Esto provoca que el mercado esté lleno de placas Arduino de múltiples fabricantes con las mismas especificaciones técnicas. En vez de la original, se compró una versión igual a un menor coste.

La placa **Arduino Mega 2560** se muestra en la Figura 5.3 y es la que vamos a utilizar en este proyecto:

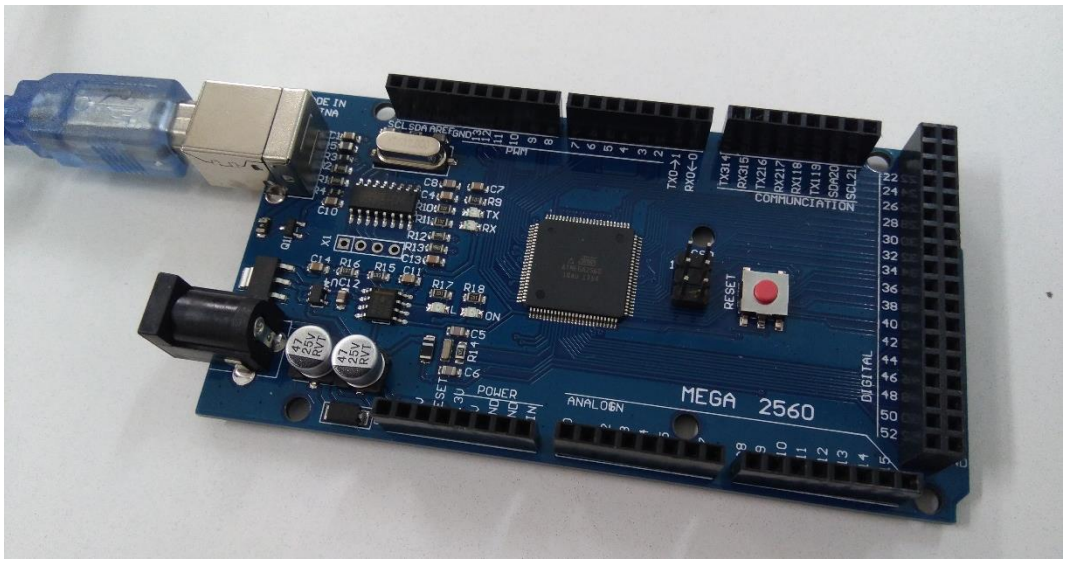


FIGURA 5.3. ARDUINO MEGA 2560

5.4 CONEXIÓN DE ARDUINO MEGA 2560

Para **conectar Arduino Mega 2560 a nuestro ordenador**, lo primero que tenemos que hacer es ir a la página web de Arduino, en la dirección https://www.arduino.cc/en/Main/Software, y descargar el IDE de Arduino (software de código abierto), que está disponible para Windows, Mac OS X, y Linux. El software puede usarse con cualquier placa de Arduino. Descargaremos la última versión disponible compatible con nuestro sistema, ARDUINO 1.8.5 (en el momento en que lo instalamos).

Además, en la página web de Arduino también podemos encontrar una **guía sobre cómo empezar con Arduino Mega 2560**, en la dirección https://www.arduino.cc/en/Guide/ArduinoMega2560.

Necesitamos la **placa Arduino Mega y un cable micro-USB** para conectarla al ordenador. Al conectar la placa al ordenador, ésta debe encenderse y un LED empezará a parpadear. Windows (en nuestro caso, es el sistema en que lo hemos instalado) detectará la presencia de un nuevo hardware e instalará nuevos controladores automáticamente. Si no lo hace automáticamente, deberemos instalarlos. Para Arduino Mega 2560, deberíamos instalar el driver CH-340 para el USB. En nuestro caso no ha hecho falta.

Una vez dentro del programa, en la pestaña Herramientas debemos seleccionar nuestra placa, el procesador que utiliza y el puerto serie al que está conectada (COM). En la parte inferior derecha del programa, nos aparecerá el nombre de nuestra placa, el procesador y el puerto COM al que está conectado (en nuestro caso COM3).

Dentro de la pestaña Archivo – Ejemplos – 01.Basics – Blink, cargamos el primer sketch para comprobar el funcionamiento de nuestra placa. Este sketch hace parpadear un LED que ya está incluido en la placa, conectado al pin número 13, por lo que no necesitamos nada más que nuestra placa para comprobar que funciona.

5.5 PROGRAMAR ARDUINO

A continuación se explican los comandos fundamentales de Arduino y la forma de escribir el código.

5.5.1 EL SKETCH

El Sketch es el código del programa y está organizado en dos secciones principales:

- **setup** – ejecutado solo al encender la placa o después de pulsar la tecla RESET; se utiliza para inicializar los modos de trabajo de los pines o el puerto serie.
- **loop** – que contiene las instrucciones repetidas continuamente (en bucle) hasta que Arduino es alimentado.

El sketch se configura de esta forma:

```
void setup() {
  //código que se ejecuta una sola vez;
}
void loop() {
  //código que se ejecuta repetidamente;
}
```

Las funciones que no devuelven ningún resultado van precedidas de la palabra clave void. Es obligatorio crear un sketch con loop y setup, pero no lo es insertar código dentro de las dos secciones. Puede existir código solo en una de ellas.

Una vez escrito el código y montado el circuito, debemos cargarlo. Pulsamos el botón Verificar para comprobar el código. Esto tardará unos segundos. Después pulsamos el segundo botón Subir para transferir el código a la placa. Los LED TX y RX parpadearán hasta que todo el programa se haya subido a la placa.

Al finalizar la transferencia, en la parte inferior de la placa aparecen indicados el tamaño en bytes del sketch, el porcentaje de espacio ocupado en memoria y otras indicaciones. Si ha surgido algún error, aparecerán mensajes en color naranja y la línea del sketch donde está el error en caso de que sea de escritura del código. Los errores más habituales se deben a la conexión del puerto serie.

5.5.2 COMANDOS EN ARDUINO

Los comandos fundamentales de Arduino:

- pinMode – para configurar los pines de la placa;
- digitalWrite – para encender un pin;
- digitalRead – para interpretar el estado de un pin;
- analogWrite – para crear señales variables;
- analogRead – para interpretar señales analógicas.

A continuación se explican con detalle estos comandos.

pinMode (pin, mode)

Esta instrucción es utilizada en la parte de configuración setup () y sirve para configurar el modo de trabajo de un PIN pudiendo ser INPUT (entrada) u OUTPUT (salida).

```
pinMode (pin, OUTPUT);  // configura el pin como salida
```

Los terminales de Arduino, por defecto, están configurados como entradas.

```
PinMode (pin, INPUT);   // configura el pin como entrada
```

digitalWrite (pin, value)

Envía al pin definido previamente como OUTPUT el valor HIGH o LOW (poniendo en 1 o 0 la salida). El pin se puede especificar ya sea como una variable o como una constante.

```
digitalWrite (pin, HIGH);  // deposita en el pin un valor HIGH (alto
o 1)
```

digitalRead (pin)

Lee el valor de un pin (definido como digital) dando un resultado HIGH (alto) o LOW (bajo). El pin se puede especificar ya sea como una variable o una constante.

```
valor = digitalRead (pin);  // hace que 'valor' sea igual al estado
leído en el pin
```

analogWrite (pin, value)

Esta instrucción sirve para escribir un pseudo-valor analógico utilizando el procedimiento de modulación por ancho de pulso (PWM) a uno de los pines de Arduino marcados como "pin PWM". El valor que se puede enviar a estos pines de salida analógica puede darse en forma de variable o constante, pero siempre con un margen de 0-255.

```
analogWrite (pin,  valor);    // escribe 'valor' en el pin definido
como analógico
```

Si enviamos el valor 0 genera una salida de 0 voltios en el pin especificado; un valor de 255 genera una salida de 5 voltios de salida en el pin especificado. Para valores de entre 0 y 255, el pin saca tensiones entre 0 y 5 voltios - el valor HIGH de salida equivale a 5v.

Debido a que esta es una función de hardware, en el pin de salida analógica (PWM) se generará una onda constante después de ejecutada la instrucción analogWrite hasta que se llegue a ejecutar otra instrucción analogWrite (o una llamada a digitalRead o digitalWrite en el mismo pin).

analogRead (pin):

Lee el valor de un determinado pin definido como entrada analógica con una resolución de 10 bits. El rango de valor que podemos leer oscila de 0 a 1023.

```
valor = analogRead (pin);   // asigna a valor lo que lee en la entrada pin
```

5.6 COMUNICACIÓN PUERTO SERIE

Hoy en día la manera más común de comunicación entre dispositivos electrónicos es la comunicación serial y Arduino no es la excepción. A través de este tipo de comunicación podremos enviar datos a y desde nuestro Arduino a otros microcontroladores o a un computador ejecutando alguna plataforma de medios.

El mismo cable con el que programamos el Arduino desde un computador es un cable de comunicación serial. La **velocidad de comunicación** con el puerto serie se ajusta normalmente a **9600 baudios** (grupos de bits transmitidos por segundo a través de una línea digital). En comunicación serial este valor es muy importante ya que todos los dispositivos que van a comunicarse deben tener la misma velocidad para poder entenderse.

Es preciso configurarlo en el setup:

```
void setup() {
Serial.begin(9600);
}
void loop() {
}
```

5.7 CONTROL DE UN SERVOMOTOR

Necesitamos conectar y controlar los servomotores que moverán las articulaciones del robot a la placa Arduino.

Dentro de un servomotor hay: un motor de corriente continua, un potenciómetro (o un sensor de posición, para detectar la posición del motor), un grupo de engranajes, y un circuito de control que recibe la señal de control, acciona el motor y detecta su posición leyendo el sensor de posición.

5.7.1 LIBRERÍA SERVO

Arduino incluye una librería especial para el control de los servomotores y es preciso incluirla al comienzo del sketch *(#include <Servo.h>).* En la siguiente dirección https://www.arduino.cc/en/Reference/Servo, se describen todas las funciones y se pueden consultar algunos ejemplos.

En la página web encontramos la siguiente información:

Esta biblioteca permite que una placa Arduino controle los servomotores. Los servos tienen engranajes integrados y un eje que se puede controlar con precisión. Los servos estándar permiten que el eje se posicione en varios ángulos, generalmente entre 0 y 180 grados. Los servos de rotación continua permiten ajustar la rotación del eje a varias velocidades.

La biblioteca Servo admite hasta 12 motores en la mayoría de las placas Arduino y 48 en la Arduino Mega. En Mega, se pueden usar hasta 12 servos sin interferir con la funcionalidad de PWM; el uso de 12 a 23 motores desactivará PWM en los pines 11 y 12.

Los servomotores tienen tres cables: potencia, tierra y señal. El cable de alimentación suele ser rojo y debe conectarse al pin de 5 V de la placa Arduino (o a una fuente de alimentación externa). El cable de tierra es típicamente negro o marrón y debe conectarse a un pin de tierra en la placa Arduino. El cable de señal de control (pulso enviado al servomotor) es típicamente amarillo, naranja o blanco y debe conectarse a un pin digital en la placa Arduino.

Hay que tener en cuenta que los servos consumen mucha energía, por lo que si se necesitan controlar más de uno o dos, debemos alimentarlos desde un suministro diferente (es decir, no el pin + 5V en la placa Arduino).

Las funciones de la librería servo son las siguientes:

attach()

Vincula la variable servo a un pin. En el sketch se escribe el siguiente código: *myservo.attach(pin)*, o *myservo.attach(pin,min,max)*.

Donde *pin* es el número de pin al que está vinculado el servo.

min es opcional; es el ancho de pulso en microsegundos, correspondiéndose con el mínimo ángulo (0 grados) del servo (por defecto 544).

max también es opcional; es el ancho de pulso en microsegundos, correspondiéndose con el máximo ángulo (180 grados) del servo (por defecto 2400).

myservo es la variable tipo Servo que tenemos que declarar; la declaramos de la siguiente manera: *Servo myservo;*

write()

Escribe un valor para el servo que controla en consecuencia el eje. Para un servo estándar, ajustará el ángulo de rotación en grados. En un servo de rotación continua, este valor será la velocidad del servo (siendo 0 la velocidad máxima en una dirección, 180 la velocidad máxima en la otra dirección, y un valor cerca de 90 servirá para parar el servo). En el sketch se escribe el siguiente código: *myservo.write(angulo)*.

angulo es el valor que se escribe en el servo, de 0 a 180.

writeMicroseconds()

Escribe un valor en microsegundos para el servo, controlando así su eje. Para un servo estándar, este valor será el ángulo del eje. Un valor de 1000 es completamente en sentido contrario a las agujas del reloj, un valor de 2000 completamente en el sentido de las agujas del reloj y 1500 es en el medio. Los servos de rotación continua responden a esta función de una manera análoga a la función anterior *write()*.En el sketch se escribe el siguiente código: *myservo.writeMicroseconds(microsegundos)*.

microsegundos es el valor del parámetro en microsegundos (número declarado como *int*, entero). Por ejemplo, *myservo.writeMicroseconds(1500),* lleva al servo al punto medio.

read()

Lee el valor actual del servo, el valor pasado en la última llamada a la función *write().* En el sketch se escribe el siguiente código: myservo.read(). Esta función retorna el valor del ángulo del servo, de 0 a 180 grados.

attached()

Comprueba si la variable *Servo* está vinculada a un pin. En el sketch se escribe el siguiente código: *myservo.attached()*. Esta función retorna verdadero (*true*) si el servo está vinculado, y falso (*false*) en caso contrario.

detach()

Desvincula a la variable Servo del pin correspondiente asociado a la instrucción attach(). Si todas las variables Servo están desvinculadas, los pines 9 y 10 pueden ser usados como señales PWM con *analogWrite().* En el sketch se escribe el siguiente código: *myservo.detach()*.

5.7.2 LIBRERÍA VARSPEEDSERVO

Para mover los servomotores vamos a utilizar una **nueva librería** que no es la que tiene Arduino (Servo.h). Esta nueva librería **VarSpeedServo.h** ya está creada, es **de libre acceso** y se puede obtener en el siguiente enlace (consultado en Junio de 2018):

https://github.com/netlabtoolkit/VarSpeedServo/blob/master/VarSpeedServo.h

Las instrucciones de esta librería son muy parecidas a las de la librería Servo.h, pero nos permiten tener un cierto control de la velocidad en la instrucción write().

Se proporciona la siguiente información:

Los métodos son:

VarSpeedServo – clase para manipular los servomotores conectados a los pines de Arduino.

attach(pin) – conecta un servomotor a un pin de entrada/salida.

attach(pin, min, max) – conecta un servomotor a un pin estableciendo valores máximos y mínimos en microsegundos.

Los valores predeterminados: mínimo 544, máximo 2400.

write(value) – establece el ángulo del servomotor en grados (Un ángulo no válido que es válido como pulso en microsegundos se trata como microsegundos)

write(value, speed) – speed varía la velocidad del movimiento a una nueva posición, 0 = velocidad máxima, 1 - 255 más lenta a más rápida.

write(value, speed, wait) – wait es una variable booleana que, si es verdadero (true), hace que la llamada a la función se bloquee hasta que se complete el movimiento.

writeMicroseconds () – establece el ancho de pulso del servo en microsegundos.

read () - obtiene el último ancho de pulso del servo escrito como ángulo entre 0 y 180 grados.

readMicroseconds () – obtiene el último ancho de pulso de servo escrito en microsegundos.

attached () – devuelve verdadero si hay un servo conectado.

detach () – detiene a los servos conectados pulsando su pin de entrada/salida.

slowmove (value, speed) - lo mismo que escribir write(value, speed), conservado para compatibilidad con la versión Korman.

stop () – detiene el servo en la posición actual.

sequencePlay (sequence, sequencePositions) – reproduce una secuencia de bucle comenzando en la posición 0.

sequencePlay (sequence, sequencePositions, loop, startPosition) – reproduce una secuencia con número de posiciones, loop si es verdadero, comenzar en posición.

sequenceStop () – detiene la secuencia en la posición actual.

5.7.3 SEÑALES ANALÓGICAS DE SALIDA EN ARDUINO (PWM)

La **modulación por ancho de pulso**, PWM (Pulse Width Modulation), es una de los **sistemas más empleados para el control de servos**. Este sistema consiste en generar una onda cuadrada en la que se varía el tiempo que el pulso está a nivel alto, manteniendo el mismo período con el objetivo de modificar la posición del servo según se desee.

Para la generación de una onda PWM en un microcontrolador, lo más habitual es usar un timer y un comparador (interrupciones asociadas), de modo que el microcontrolador quede libre para realizar otras tareas, y la generación de la señal sea automática y más efectiva. El mecanismo consiste en programar el timer con el ancho del pulso (el período de la señal) y al comparador con el valor de duración del pulso a nivel alto. Cuando se produce una interrupción de overflow del timer, la subrutina de interrupción debe poner la señal PWM a nivel alto y cuando se produzca la interrupción del comparador, ésta debe poner la señal PWM a nivel bajo. En este caso ya existe una librería para Arduino que se encarga de gestionar estas funciones.

El sistema de control de un servo se limita a indicar en qué posición se debe situar. Esto se lleva a cabo mediante una serie de pulsos tal que la duración del pulso indica el ángulo de giro del motor. Cada servo tiene sus márgenes de operación, que se corresponden con el ancho del pulso máximo y mínimo que el servo entiende. Los valores más generales se corresponden con pulsos de entre 1 ms y 2 ms de anchura, que dejarían al motor en ambos extremos (0º y 180º). El valor 1.5 ms indicaría la posición central (90º), mientras que otros valores del pulso lo dejan en posiciones intermedias. Estos valores suelen ser los recomendados, sin embargo, es posible emplear pulsos menores de 1 ms o mayores de 2 ms, pudiéndose conseguir ángulos mayores de 180°.

Si se sobrepasan los límites de movimiento del servo, éste comenzará a emitir un zumbido, indicando que se debe cambiar la longitud del pulso. El factor limitante es el tope del potenciómetro y los límites mecánicos constructivos.

Es importante destacar que para que un servo se mantenga en la misma posición durante un cierto tiempo, es necesario enviarle continuamente el pulso correspondiente. De este modo, si existe alguna fuerza que le obligue a abandonar esta posición, intentará resistirse. Si se deja de enviar pulsos (o el intervalo entre pulsos es mayor que el máximo) entonces el servo perderá fuerza y dejará de intentar mantener su posición, de modo que cualquier fuerza externa podría desplazarlo.

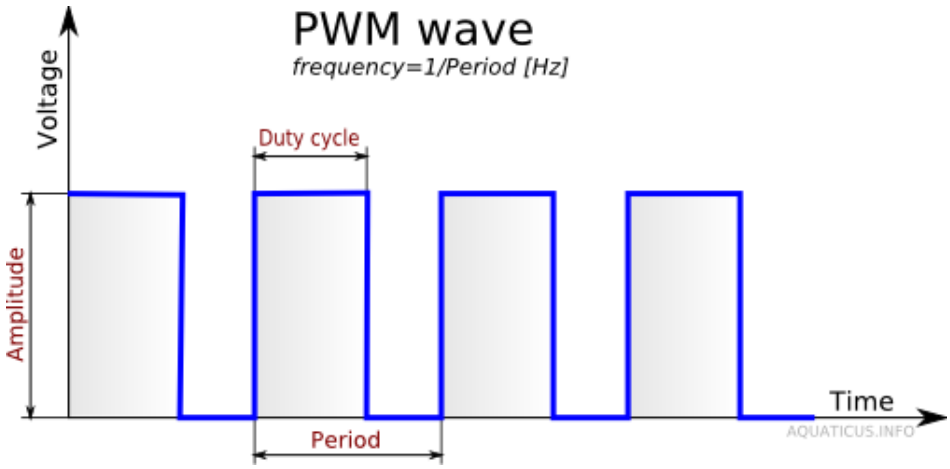


FIGURA 5.4. SEÑAL PWM (MODULACIÓN POR ANCHO DE PULSO)

En la Figura 5.4 se muestra **una señal PWM**, donde:

Amplitude: Es el voltaje que se aplica. Normalmente 5V.

Period: Es el % total de ciclo en el que la señal está en valor alto.

Pulse Width: Es el tiempo que está en el valor alto.

Duty cycle: Es el tiempo en el que el pulso está en estado (ON).

Frequency: Es la cantidad de pulsos completos por segundo; se mide Hz.

Por ejemplo, si una señal tiene un periodo de 10 ms y sus pulsos son de ancho 2ms, dicha señal tiene un duty cycle de 20% (20% ON y 80% OFF).

La señal PWM se utiliza como técnica para controlar circuitos analógicos. El periodo y la frecuencia del tren de pulsos pueden determinar la potencia entregada a dicho circuito. Si, por ejemplo, tenemos un voltaje de 9V y lo modulamos con un duty cycle del 10%, obtenemos 0.9V de señal analógica de salida.

Las señales PWM **son comúnmente usadas para el control de motores DC** (si bajas la frecuencia, la inercia del motor es más pequeña y se moverá más lentamente), ajustar la intensidad del brillo de un LED, etc.

En Arduino la señal de salida PWM es una señal de frecuencia constante (30769 Hz) y que sólo nos permite cambiar el "duty cycle" o el tiempo que el pulso está activo (ON) o inactivo (OFF), utilizando la función analogWrite().

CAPÍTULO 6. MONTAJE DEL BRAZO ROBÓTICO

6.1 INTRODUCCIÓN

En este capítulo se explica el proceso de montaje del brazo robótico. El **Anexo 1 son las Instrucciones de Montaje**, donde se explican paso por paso las piezas que se necesitan, así como todos los componentes necesarios y el modo de conectarlos.

Complementariamente a las instrucciones, el siguiente apartado se refiere a la conexión de los servos segundo y tercero a las piezas correspondientes del brazo robótico. Los servos tienen un recorrido de 180 grados y es necesario conectarlos en una determinada posición para que el extremo del brazo robótico alcance las posiciones deseadas que hemos elegido.

6.2 POSICIONAMIENTO Y RECORRIDO DE LOS SERVOS

En este apartado se explica la conexión del segundo y tercer servo (articulaciones del hombro y del codo).

Para determinar cuál será la posición de montaje del segundo y del tercer servo, vamos a hacer una **simulación en Working Model** del brazo robótico que nos permita comprobar visualmente cuál es el espacio de trabajo del robot en 2D (en el plano vertical), según los ángulos límite de los servos. En concreto, de los servos que mueven la segunda y la tercera articulación, que son los que determinan el comportamiento del cuadrilátero articulado.

El rectángulo rojo representa el brazo del robot y es el que estará controlado por el segundo servo (segunda articulación). El rectángulo azul representa la pieza 14 cuyo movimiento estará controlado por el tercer servo (que moverá la tercera articulación). Por lo tanto, el ángulo que forman el brazo y el antebrazo del robot (rectángulos rojo y verde respectivamente, en la Figura 6.1), dependerá de las posiciones del segundo y tercer servo (rectángulos rojo y azul).

Los servos segundo y tercero están unidos a los paneles laterales girados un ángulo de 45 grados respecto de la base. Vamos a considerar que son 45 grados aunque este ángulo es aproximado ya que al atornillar los servos a los paneles laterales puede variar y además existe una pequeña variación en el ángulo en la propia fabricación de las piezas.

Posicionamos el TCP (*Tool Centre Point*, punto central de la herramienta) del robot **de una forma aproximada** como se muestra en la Figura 6.1. Las medidas del cuadrilátero articulado son las reales.

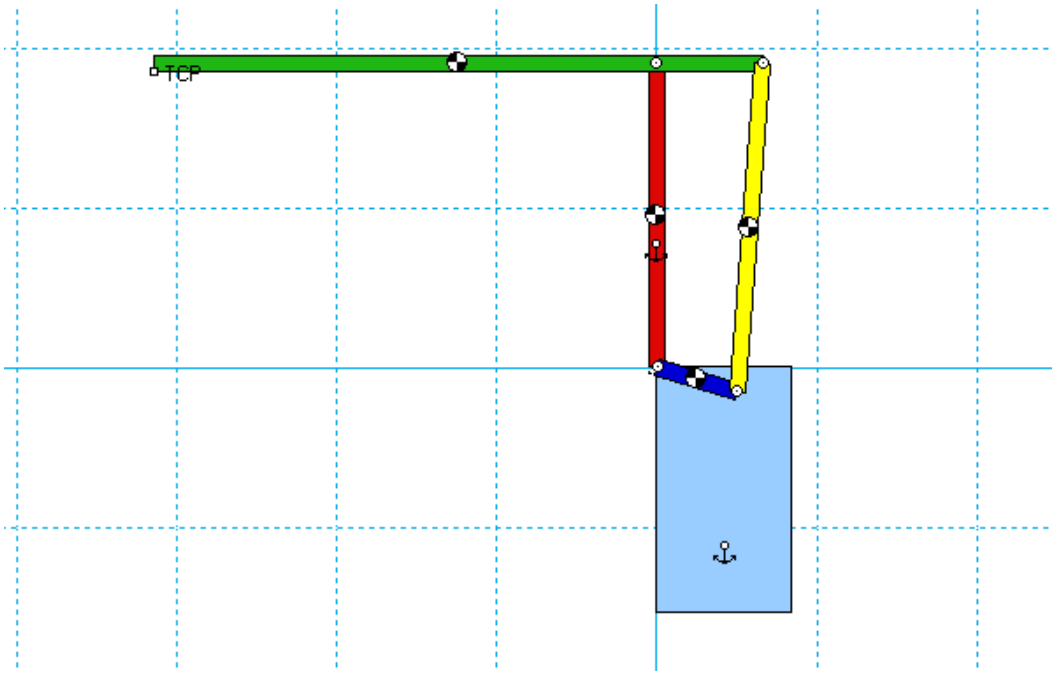


FIGURA 6.1. MODELO DEL BRAZO ROBÓTICO (WORKING MODEL)

Conectaremos el **tercer servo** antes que el segundo por razones de montaje. Vamos a conectarlo de manera que la pieza 14 (rectángulo azul) quede en vertical respecto a la base del robot, ya que en esta posición es más fácil alinear la pieza (debido a las separaciones del eje del servo que tiene que conectarse a la corona).

Hemos decidido que el recorrido de esta pieza irá desde – 30 grados hasta 150 grados (respecto de la vertical, tomando como sentido positivo el sentido horario). Por tanto, si vamos a encajar la pieza verticalmente, el tercer servo deberá marcar una posición de 30 grados.

Es decir, el servo estará en su posición de 0 grados cuando la pieza esté a – 30 grados de la vertical y en su posición de 180 grados cuando la pieza esté a 150 grados de la vertical.

A continuación se representa la posición de la pieza 14 (rectángulo azul) en la que se ha conectado el tercer servo y las dos posiciones límite de la misma (0 grados y 180 grados del servo, respectivamente).

Hay que tener en cuenta que esta pieza se conecta a la corona del servo individualmente, sin estar unida a ningún conjunto, por lo que cuando la conectemos durante el montaje sólo tenemos que asegurarnos de conectarla verticalmente.

Posición de la pieza 14 al conectarla al tercer servo (Figura 6.2):

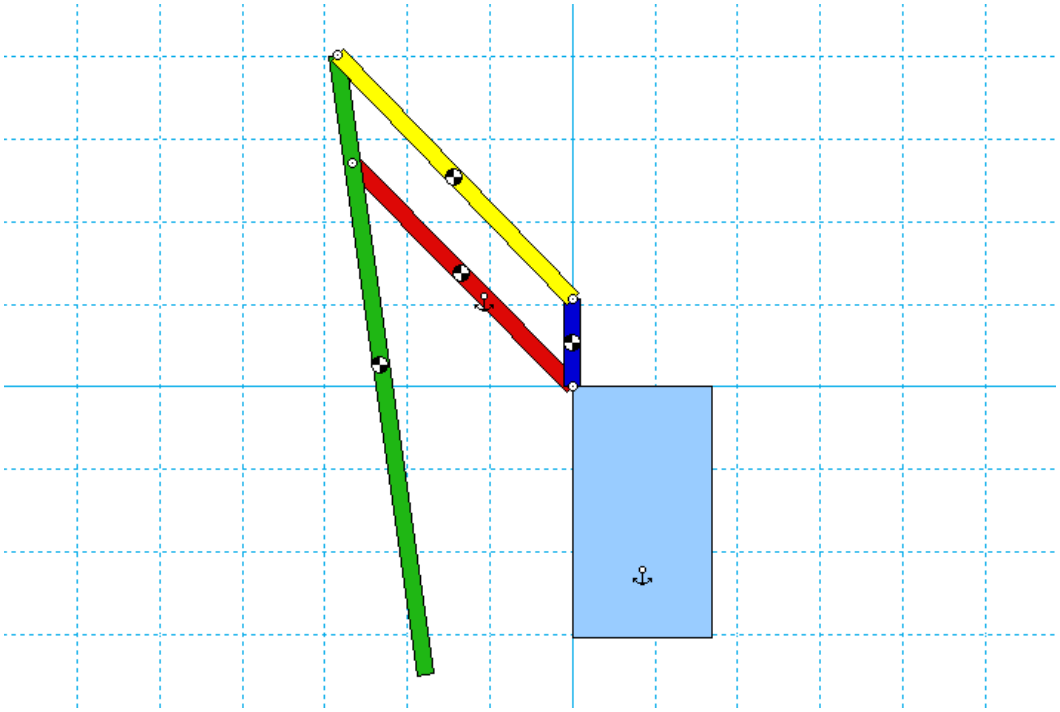


FIGURA 6.2. POSICIÓN DE CONEXIÓN DE LA PIEZA 14 (WORKING MODEL)

Posición de la pieza 14 cuando el tercer servo marca 0 grados (Figura 6.3):

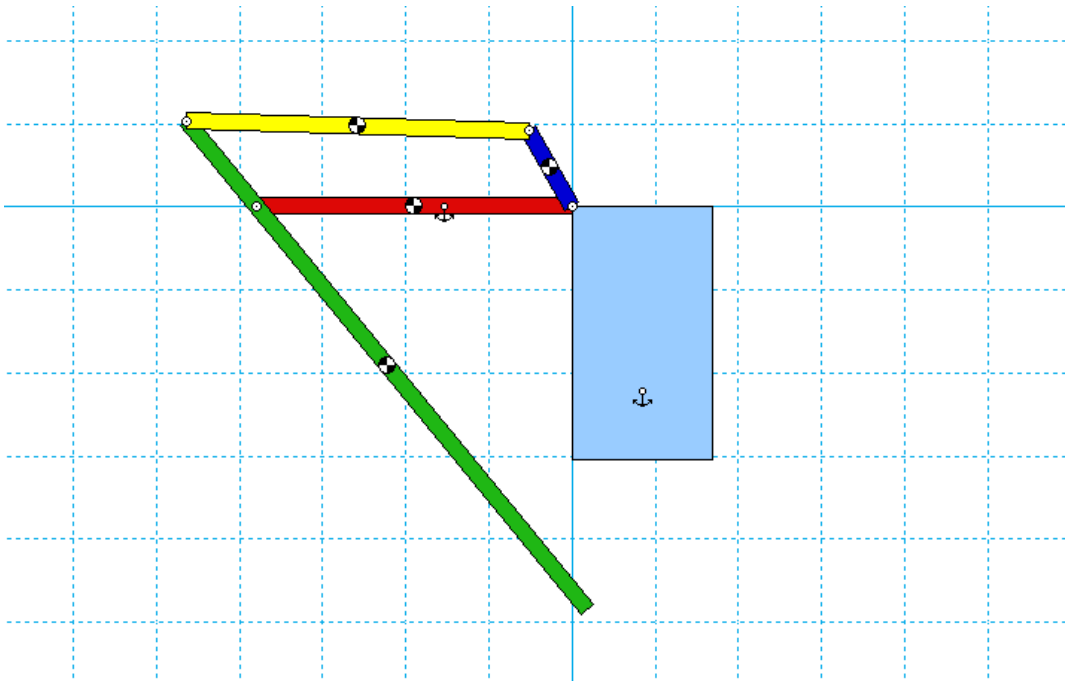


FIGURA 6.3. POSICIÓN DE LA PIEZA 14 LÍMITE 0 GRADOS (WORKING MODEL)

Posición de la pieza 14 cuando el tercer servo marca 180 grados (Figura 6.4):

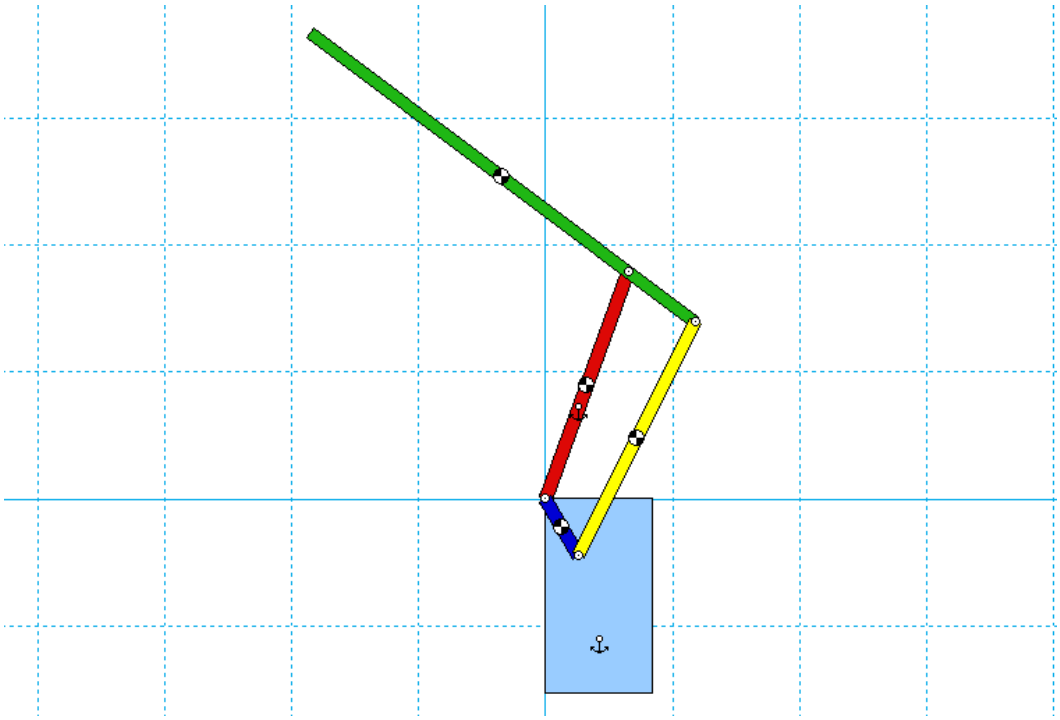


FIGURA 6.4. POSICIÓN DE LA PIEZA 14 LÍMITE 180 GRADOS (WORKING MODEL)

A continuación establecemos una posición intermedia para el **segundo servo**. Como se ha comentado antes, consideramos que los servos están unidos a los paneles laterales girados un ángulo de 45 grados respecto de la base.

Consideramos que si en esa posición el segundo servo está a 90 grados (en su posición intermedia), el recorrido completo del brazo del robot irá desde 45 grados hasta – 135 grados (respecto de la vertical, tomando como sentido positivo el sentido horario).

Es decir, el servo estará en su posición de 0 grados a 45 grados de la vertical y en su posición de 180 grados a – 135 grados de la vertical. De manera que cuando el antebrazo del robot esté en la vertical, la posición que marcará el servo será de 45 grados.

A continuación se representa la posición del antebrazo en la que se ha conectado el segundo servo (posicionado a 90 grados) y las dos posiciones límite del mismo (0 grados y 180 grados del servo, respectivamente).

Posición del brazo (rectángulo rojo) al conectarlo al segundo servo (Figura 6.5):

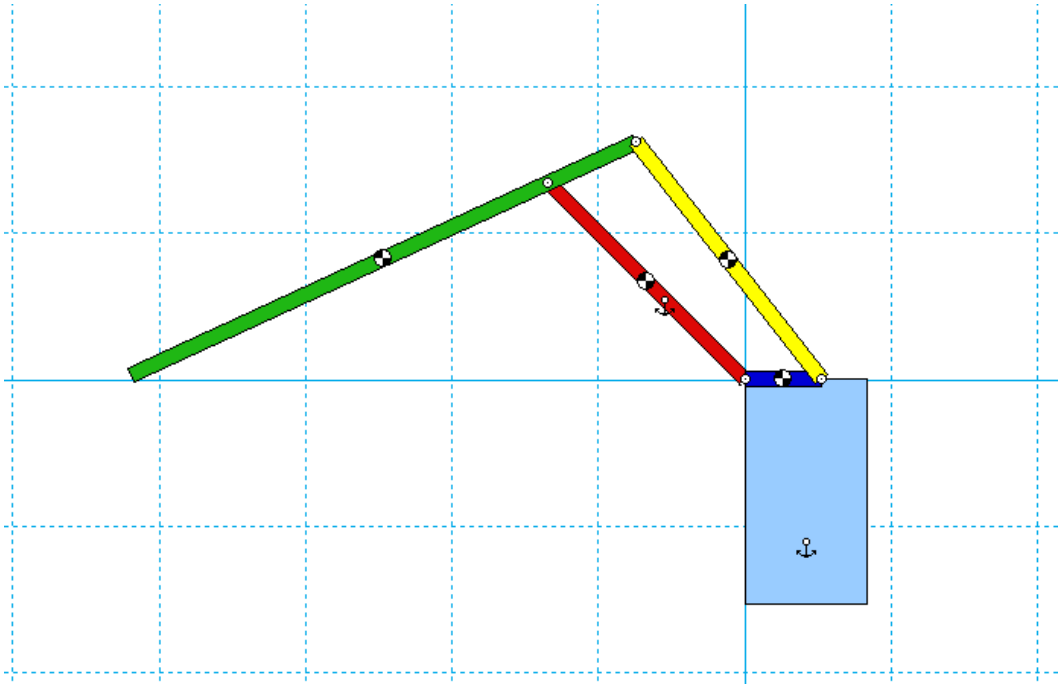


FIGURA 6.5. POSICIÓN DE CONEXIÓN DEL BRAZO (WORKING MODEL)

Como en el caso anterior, hay que tener en cuenta que el brazo se conecta a la corona del servo sin estar unido aún al antebrazo, por lo que cuando le conectemos durante el montaje sólo tenemos que asegurarnos de que el brazo esté a 45 grados (la posición del resto de los servos conectados da igual).

Posición del brazo cuando el segundo servo marca 0 grados (Figura 6.6):

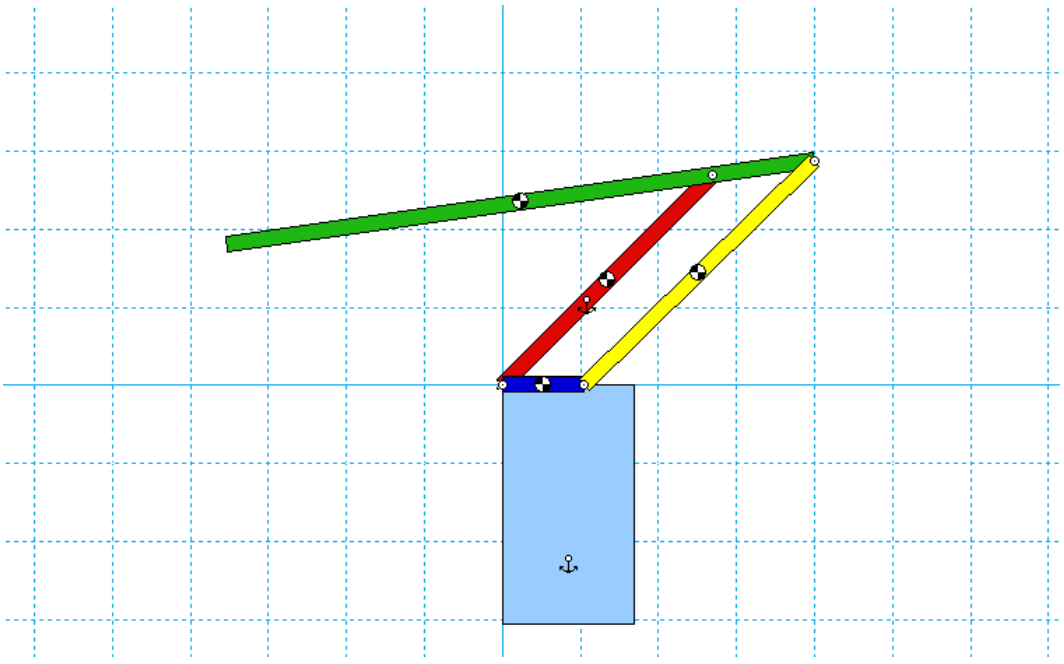


FIGURA 6.6. POSICIÓN DEL BRAZO LÍMITE 0 GRADOS (WORKING MODEL)

Posición del brazo cuando el segundo servo marca 180 grados (Figura 6.7):

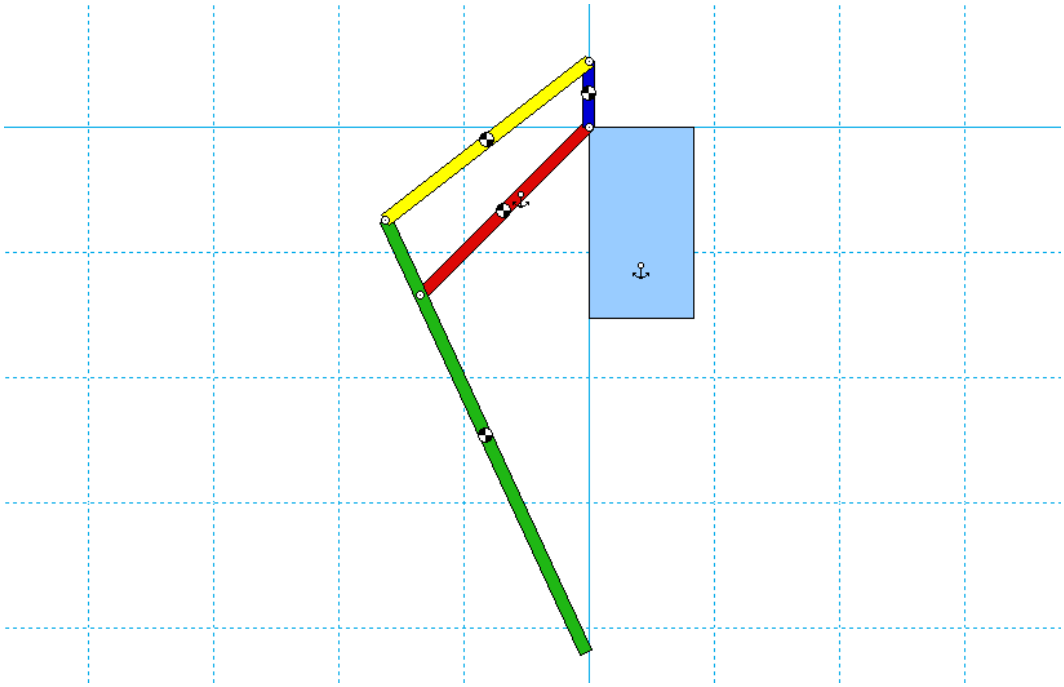


FIGURA 6.7. POSICIÓN DEL BRAZO LÍMITE 180 GRADOS (WORKING MODEL)

Las posiciones límite del cuadrilátero articulado (llevando al límite ambos servos) son las siguientes, en las Figuras 6.8 y 6.9.

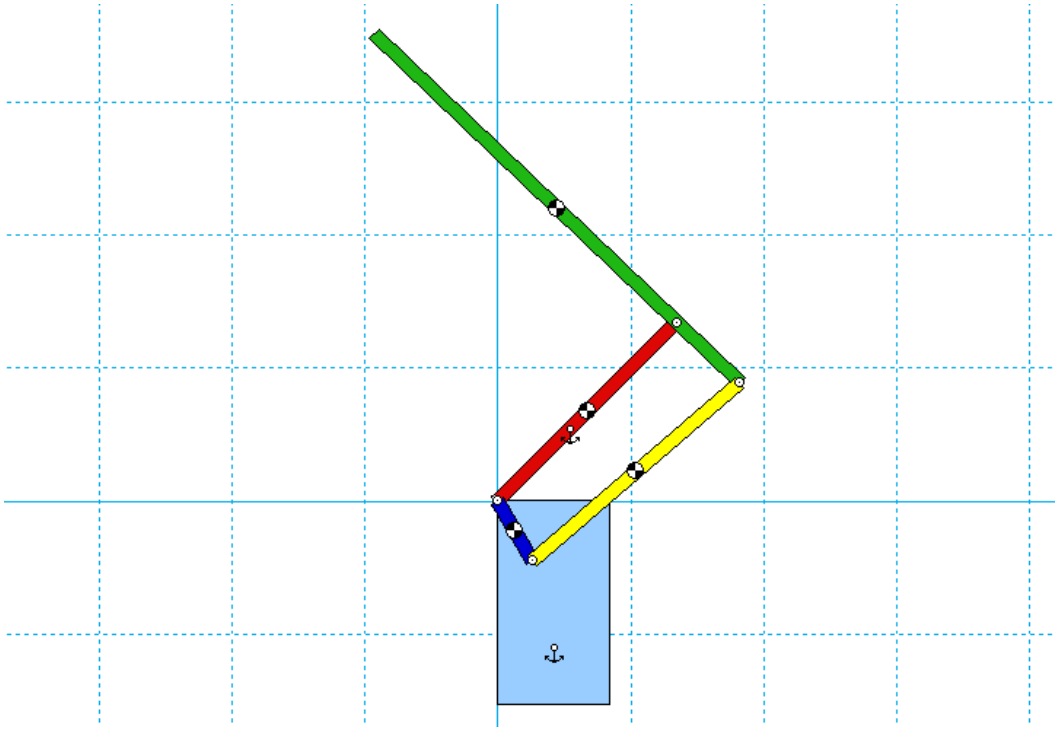


FIGURA 6.8. POSICIÓN LÍMITE 1 (WORKING MODEL)

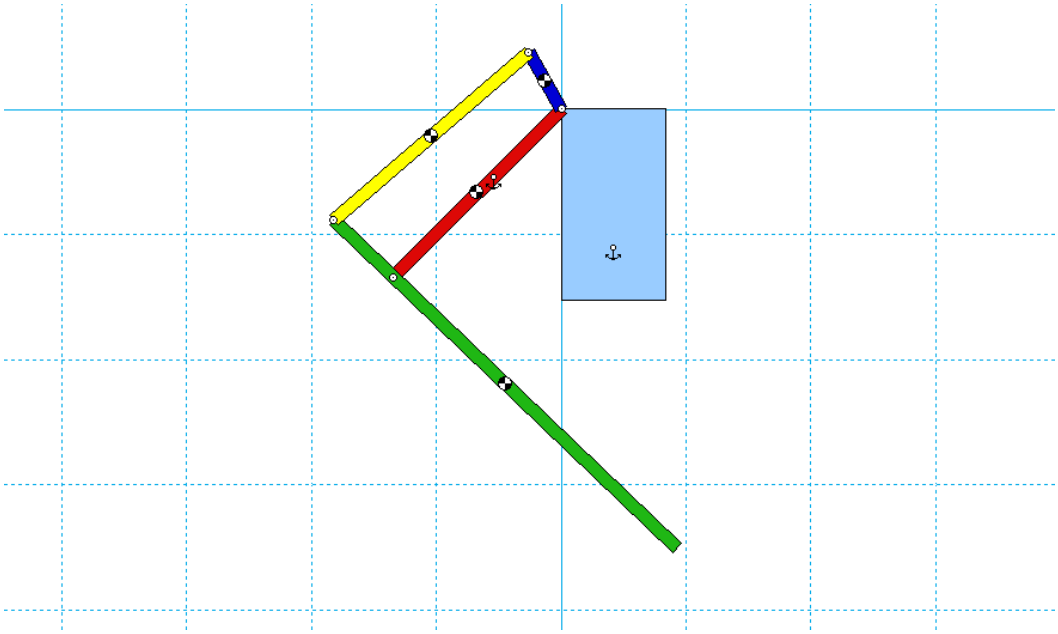


FIGURA 6.9. POSICIÓN LÍMITE 2 (WORKING MODEL)

6.3 CONEXIÓN SERVOS - ARDUINO

Durante la fase de montaje del robot, tenemos que llevar los servos a una determinada posición. Los servomotores consumen una potencia considerable. Si vamos a conectar más de un servomotor a la placa Arduino, necesitaremos una fuente de alimentación externa. Incluso si sólo vamos a conectar un servomotor, es preferible que lo hagamos con una fuente de alimentación externa y no con los + 5 V que nos proporciona Arduino.

Los servomotores MG 996R requieren un voltaje de alimentación mayor que los micro servomotores MG90S.

CONEXIÓN DURANTE EL MONTAJE

Vamos a realizar el siguiente **esquema de conexión** que se indica en la Figura 6.10, para llevarlos a las posiciones deseadas durante el montaje. Necesitaremos una protoboard y una fuente de alimentación externa (o una batería) para alimentar los servos. El pin de conexión es el pin 9 (PWM).

La **protoboard** es una tarjeta con uniones eléctricas preestablecidas para facilitar la conexión de los diferentes componentes. Las dos filas externas son los buses de alimentación y de tierra y están unidas por filas (horizontalmente). Los agujeros de las filas centrales forman el área de prototipado y están unidos eléctricamente por columnas (en vertical).

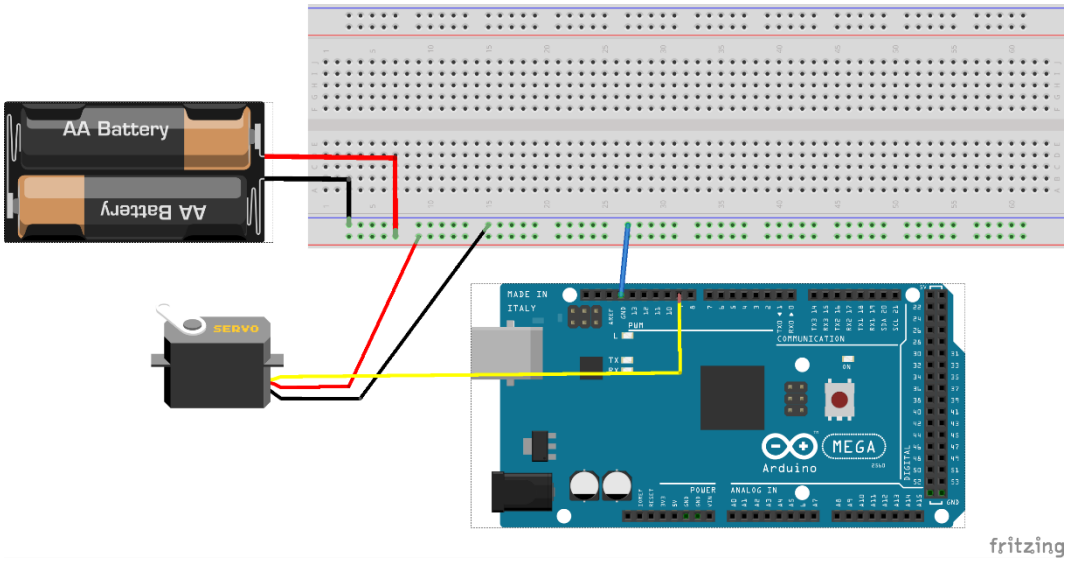


FIGURA 6.10. CONEXIÓN SERVO – ARDUINO DURANTE EL MONTAJE

Con este esquema de conexión podemos conectar todos los servos (uno a uno) y evitar posibles problemas, incluidos los micro servos, aunque éstos se puedan conectar directamente a la alimentación de la placa.

CONEXIÓN FINAL PLACA DE TIRAS - SERVOS

Después de montar el robot y sujetarlo ala caja base, la conexión entre los servomotores y la placa Arduino se va a realizar en una placa de tiras donde irán soldadas las uniones. Se va a soldar a la placa de tiras el conector de la fuente de alimentación que proporciona la energia a los servomotores.

Los tres cables que tiene cada servo irán conectados a unos pines soldados a la placa. Los pines PWM de cada servo irán conectados a la placa Arduino a los pines correspondientes mediante cableado (pines PWM del 8 al 13, del primer servo al último).

Los cables de alimentación y tierra de los servos irán conectados a los correspondientes cables del conector de la fuente de alimentacion a través de la placa de tiras.

El otro cable que sale de la placa Arduino es el cable GND, que irá conectado a la placa soldada (tierra de la fuente de alimentación) que se conecta a su vez con los cables de tierra de cada servo.

También vamos a incluir un interruptor con el que se encenderá la fuente de alimentación, siempre que el interruptor de la fuente también esté en encendido.

A continuación se muestran unas imágenes de diferentes vistas de la placa de tiras y las uniones con los servos y con la placa Arduino (Figuras 6.11, 6.12 y 6.13). La placa Arduino estará conectada a un ordenador mediante un clabe USB.

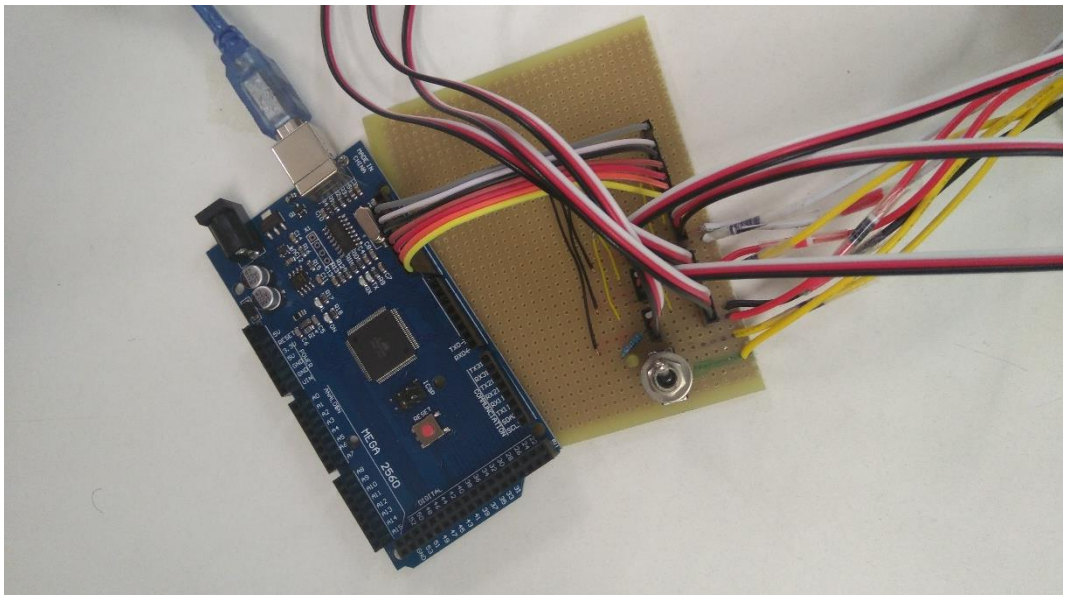


FIGURA 6.11. CONEXIONES PLACA DE TIRAS – ARDUINO (1)

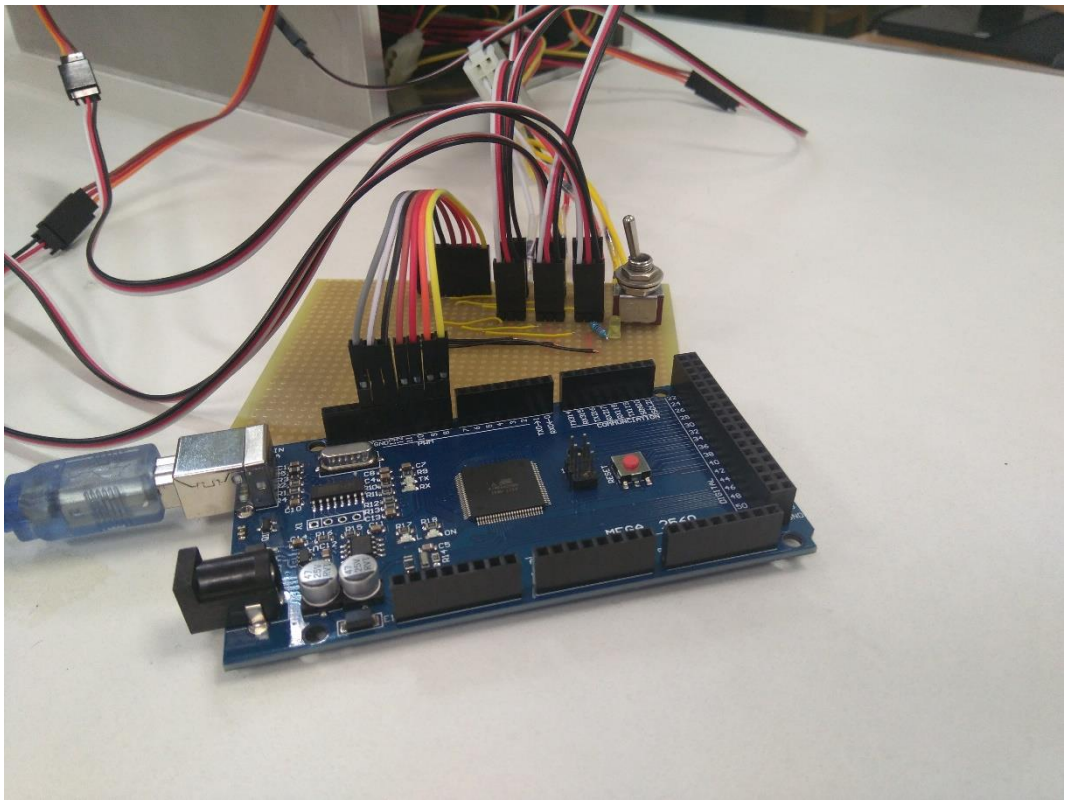


FIGURA 6.12. CONEXIONES PLACA DE TIRAS – ARDUINO (2)

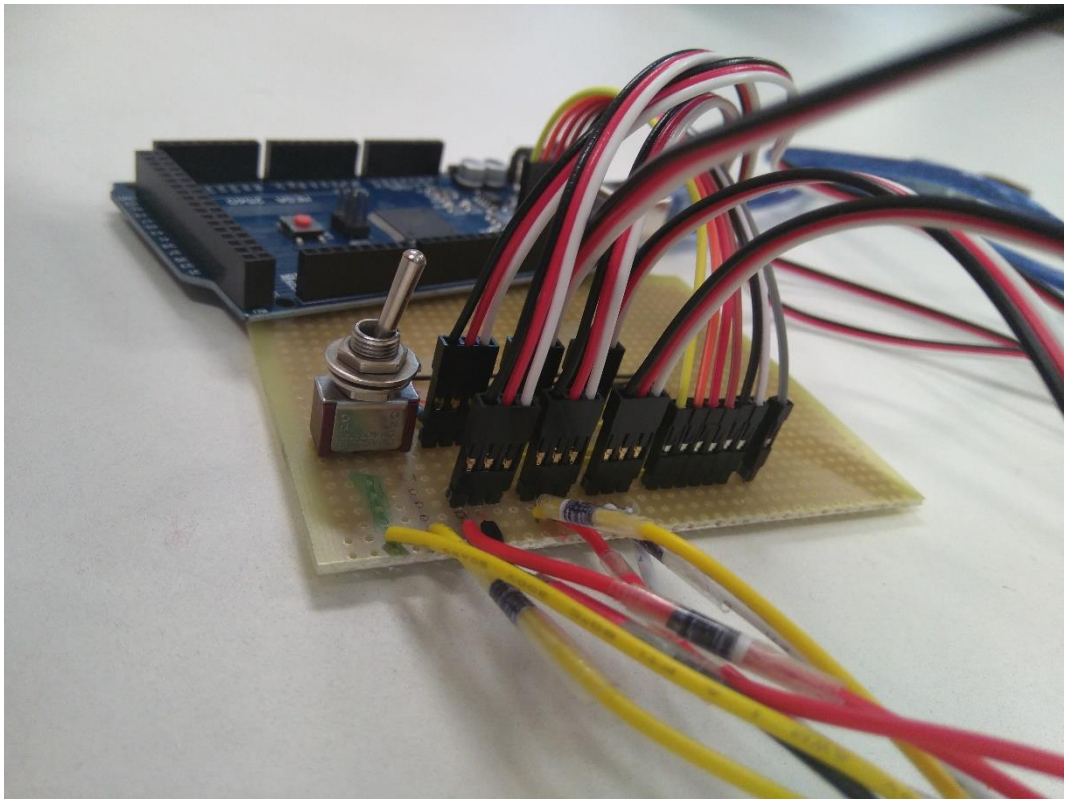


FIGURA 6.13. CONEXIONES SERVOS – PLACA DE TIRAS

CAPÍTULO 7. CINEMÁTICA DIRECTA E INVERSA

7.1 INTRODUCCIÓN

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia sin considerar las fuerzas que intervienen. Se interesa por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares.

La formulación de los modelos cinemáticos adecuados es crucial para analizar el comportamiento de manipuladores industriales.

La cinemática del robot se puede dividir en cinemática directa e inversa. El problema de la cinemática directa es fácil de resolver derivando las ecuaciones. Por lo tanto, siempre hay una solución para la cinemática directa de un manipulador. La cinemática inversa es un problema mucho más difícil que la cinemática directa. La solución del problema de la cinemática inversa es computacionalmente expansiva y, en general, lleva mucho tiempo en el control de manipuladores en tiempo real. Singularidades y no linealidades hacen que el problema sea más difícil de resolver. Por lo tanto, solo para una clase muy pequeña de manipuladores cinemáticamente simples (manipuladores con muñeca de Euler) existen soluciones analíticas completas.

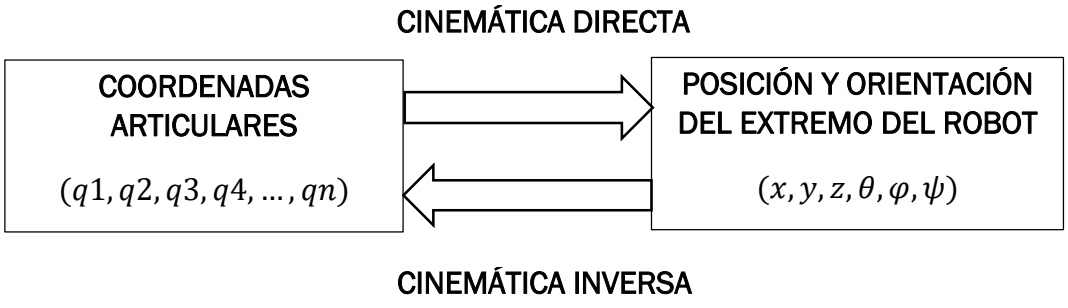
Diferenciamos dos problemas:

- Problema cinemático directo: consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas que se tome como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot.

- Problema cinemático inverso: resuelve la configuración que debe adoptar el robot para una posición y orientación del extremo conocidas.

El método de Denavit-Hartenberg (D-H) utiliza una matriz de transformación homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciéndose el problema cinemático directo a encontrar una matriz de transformación homogénea 4x4 que relaciona la localización espacial del extremo del robot con respecto al sistema de coordenadas de su base.

.



La cinemática del robot trata también de encontrar las relaciones entre las velocidades del movimiento de las articulaciones y las del extremo del robot. Es lo que se conoce como modelo diferencial, expresado mediante la matriz Jacobiana.

7.2 MODELO CINEMÁTICO DIRECTO

La resolución del problema cinemático directo permite conocer cuál es la posición y orientación que adopta el extremo del robot cuando cada una de las variables que fijan la posición u orientación de sus articulaciones toma valores determinados.

La obtención del modelo cinemático directo puede ser abordado mediante dos enfoques diferentes denominados métodos geométricos y métodos basados en cambios de sistemas de coordenadas.

Los primeros son adecuados para casos simples, pero al no ser sistemáticos, su aplicación queda limitada a robots con pocos grados de libertad. Los métodos basados en cambio de sistemas de referencia, permiten de una manera sistemática abordar la obtención del modelo cinemático directo del robot para robots de n grados de libertad, siendo éstos, los más utilizados, en particular los que usan matrices de transformación homogénea.

RESOLUCIÓN CON MATRICES DE TRANSFORMACIÓN HOMOGÉNEA

Dado que un robot se puede considerar como una cadena cinemática formada por objetos rígidos o eslabones unidos entre sí mediante articulaciones, se puede establecer un sistema de referencia fijo situado en la base del robot y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia. De esta forma el problema cinemático directo se reduce a encontrar una matriz de transformación homogénea T que relacione la posición y orientación del extremo del robot respecto del sistema de referencia fijo situado en la base del mismo. Esta matriz T será función de las coordenadas articulares.

7.2.1 CONVENCIÓN DENAVIT-HARTENBERG

El **método de Denavit-Hartenberg** es una herramienta ampliamente conocida en el área de la ingeniería y en concreto en robótica, ya que ofrece un procedimiento sencillo para obtener el modelo cinemático directo cuya estructura queda en términos de las transformaciones homogéneas.

Jacques Denavit y Richard S. Hartenberg en 1955 presentaron un procedimiento para obtener una representación mínima de la orientación y traslación de robots manipuladores. Consiste en determinar una tabla de parámetros relacionados con los eslabones del robot. La convención Denavit-Hartenberg toma como referencia el diagrama de un robot manipulador en cadena cinemática abierta.

Escogiendo los sistemas de coordenadas asociados a cada eslabón según la representación propuesta por Denavit-Hartenberg, será posible pasar de uno al siguiente mediante 4 transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón. En definitiva, este algoritmo es un recurso para obtener de manera sistemática la matriz de transformación homogénea.

Hay que hacer notar que si bien en general una matriz de transformación homogénea queda definida por 6 grados de libertad, el método de Denavit-Hartenberg, permite, en eslabones rígidos, reducir éste a 4 con la correcta elección de los sistemas de coordenadas.

Estas **cuatro transformaciones básicas** consisten en una sucesión de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento i-1 con el elemento i. Las transformaciones en cuestión son las siguientes (es importante recordar que el paso del sistema $\{S_{i-1}\}$ al $\{S_i\}$ mediante estas 4 transformaciones está garantizado sólo si los sistemas han sido definidos de acuerdo a unas normas determinadas):

1. Rotación alrededor del eje z_{i-1}un ángulo θ_i
2. Traslación a lo largo del eje z_{i-1}una distancia d_i
3. Traslación a lo largo del eje x_i una distancia a_i
4. Rotación alrededor del eje x_i un ángulo α_i

Donde las transformaciones se refieren al sistema móvil. Dado que el producto de matrices no es conmutativo, las transformaciones se han de realizar en el orden indicado. De este modo se tiene que:

$$A_{i-1}^{i} = Rotz(\theta_i)T(0,0,d_i)T(a_i,0,0)Rotx(\alpha_i) \qquad [7.1]$$

Y realizando el producto entre matrices se obtiene que:

$$A_{i-1}^{i} =$$

$$= \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{i-1}^{i} = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [7.2]$$

Donde θ_i , d_i , α_i , a_i son los parámetros D-H del eslabón i. De ese modo, basta con identificar los 4 parámetros para obtener las matrices A_{i-1}^{i} y así relacionar todos y cada uno de los eslabones del robot.

Como se ha indicado, para que la matriz A_{i-1}^{i} definida en [7.2] relacione los sistemas $\{S_{i-1}\}$ y $\{S_i\}$, es necesario que los sistemas se hayan escogido de acuerdo a unas **determinadas normas**. Éstas, **junto con la definición de los 4 parámetros de Denavit-Hartenberg**, conforman el siguiente **algoritmo para la resolución del problema cinemático directo**:

DH1. Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.

DH2. Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n.

DH3. Localizar el eje de cada articulación. Si ésta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.

DH4. Para i de 0 a $n-1$ situar el eje z_i sobre el eje de la articulación $i+1$.

DH5. Situar el origen del sistema de la base $\{S_0\}$ en cualquier punto del eje z_0. Los ejes x_0 e y_0 se situarán de modo que formen un sistema dextrógiro con z_0.

DH6. Para i de 1 a $n-1$, situar el sistema $\{S_i\}$ (solidario al eslabón i) en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i. Si ambos ejes se cortasen se situaría $\{S_i\}$ en el punto de corte. Si fuesen paralelos $\{S_i\}$ se situaría en la articulación $i+1$.

DH7. Situar x_i en la línea normal común a z_{i-1} y z_i.

DH8. Situar y_i de modo que forme un sistema dextrógiro con x_i y z_i.

DH9. Situar el sistema $\{S_n\}$ en el extremo del robot de modo que z_n coincida con la dirección de z_{n-1} y x_n sea normal a z_{n-1} y z_n.

DH10. Obtener θ_i como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos.

DH11. Obtener d_i como la distancia, mediad a lo largo de z_{i-1}, que habría que desplazar $\{S_{i-1}\}$ para que x_i y x_{i-1} quedasen alineados.

DH12. Obtener a_i como la distancia medida a lo largo de x_i (que ahora coincidiría con x_{i-1}) que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con $\{S_i\}$.

DH13. Obtener α_i como el ángulo que habría que girar en torno a x_i, para que el nuevo $\{S_{i-1}\}$ coincidiese totalmente con $\{S_i\}$.

DH14. Obtener las matrices de transformación A_{i-1}^{i} definidas en [7.2].

DH15. Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot $\boldsymbol{T = A_0^1 \cdot A_1^2 \cdot A_{n-1}^n}$.

DH16. La matriz $\boldsymbol{T}$ define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base, en función de las n coordenadas articulares.

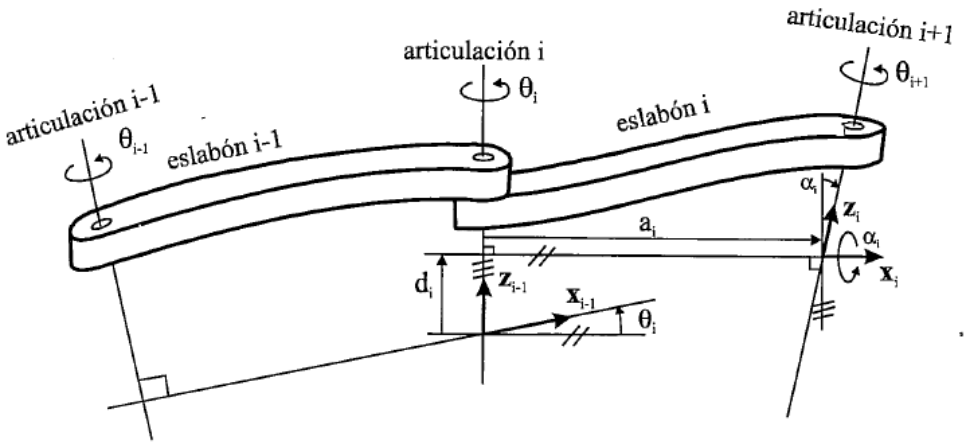


FIGURA 7.1. PARÁMETROS DH PARA UN ESLABÓN GIRATORIO

Los cuatro **parámetros de D-H** dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y siguiente. En concreto estos representan (Figura 7.1):

> θ_i es el ángulo que forman los ejes x_{i-1} y x_i medido en un plano perpendicular al eje z_{i-1} ,utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.
>
> d_i es la distancia a lo largo del eje z_{i-1} desde el origen del sistema de coordenadas (i - 1)-ésimo hasta la intersección del eje z_{i-1} con el eje x_i. Se trata de un parámetro variable en articulaciones prismáticas.
>
> a_i es la distancia a lo largo del eje x_i que va desde la intersección del eje z_{i-1} con el eje x_i hasta el origen del sistema i-ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes z_{i-1} y z_i.

α_i es el ángulo de separación del eje z_{i-1} y el eje z_i, medido en un plano perpendicular al eje x_i, utilizando la regla de la mano derecha.

Una vez obtenidos los parámetros D-H, el cálculo de las relaciones entre los eslabones consecutivos de robot es inmediato, ya que vienen dadas por las matrices $\boldsymbol{A_{i-1}^{i}}$. Las relaciones entre varios eslabones consecutivos dos a dos vienen dadas por las matrices $\boldsymbol{T}$, que se obtienen como producto conjunto de matrices $\boldsymbol{A}$.

Obtenida la matriz $\boldsymbol{T}$, ésta expresará la orientación (submatriz 3x3 de rotación) y posición (submatriz 3x1 de traslación) del extremo del robot en función de sus coordenadas articulares, con lo que quedará resuelto el problema cinemático directo.

7.2.2 DESAROLLO DE LA CINEMÁTICA DIRECTA

Siguiendo el algoritmo de DH, comenzamos con la primera norma DH1: es necesario enumerar los eslabones (links) que se enumerarán desde 0 (la bancada o base del robot) hasta el eslabón n (el elemento terminal). Las articulaciones se enumeran desde 1 hasta n.

Cada articulación tiene un grado de libertad. En nuestro caso n es igual a 6. La articulación i será la que permita el movimiento relativo entre el eslabón i – 1 y el eslabón i.

En la Figura 7.2 (Fuente del dibujo del brazo robótico: micropede.de) se enumeran los eslabones del brazo robótico.

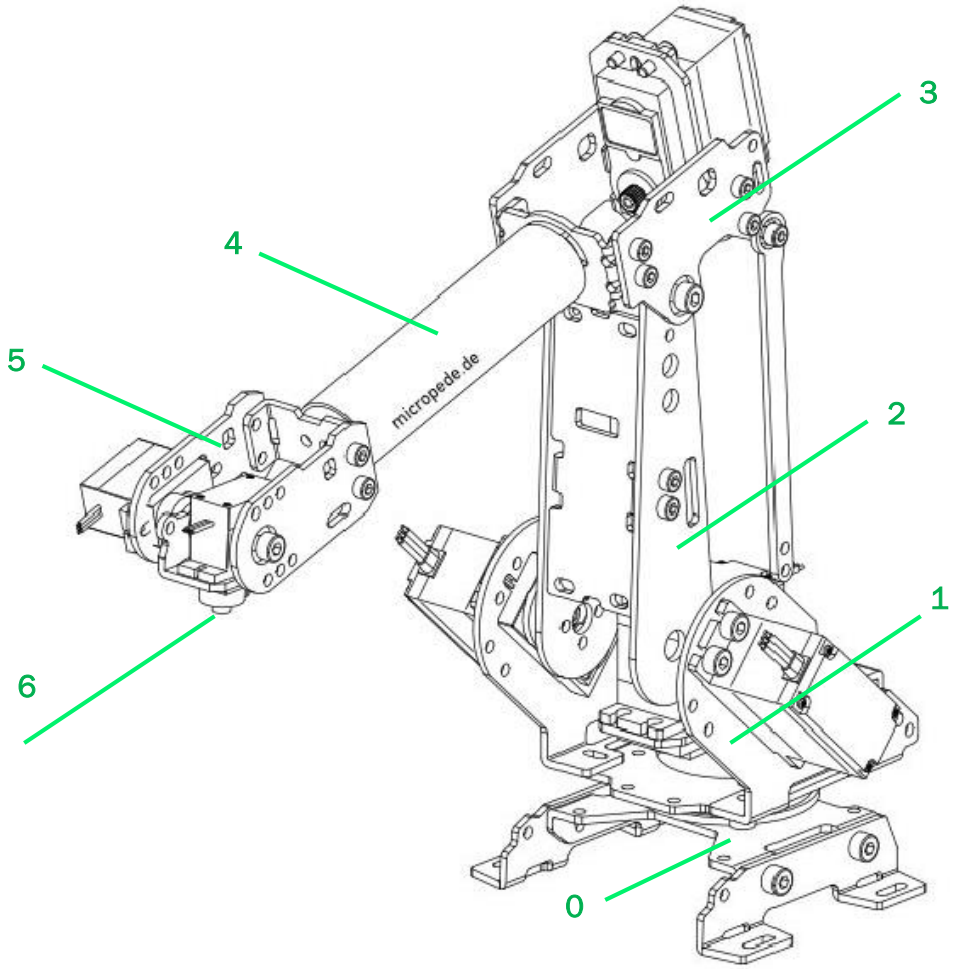


FIGURA 7.2. NUMERACIÓN DE LOS ESLABONES DEL BRAZO ROBÓTICO

Recordamos que las **dos condiciones que deben imponerse a los sistemas de referencia** solidarios a cada eslabón para que la matriz A correspondiente pueda caracterizarse con sólo 4 parámetros son las siguientes:

1: Los ejes z_{i-1} y x_i deben ser coplanarios, cortándose en un punto Pi.

2: Los ejes z_{i-1} y x_i deben formar un ángulo de 90 grados (perpendiculares)

A continuación se indican en la Figura 7.3 los sistemas de referencia escogidos para el brazo robótico (Fuente del dibujo del brazo robótico: micropede.de):

Los orígenes de los sistemas de referencia O_3 y O_4 están en la misma vertical, aunque se han representado separados para mayor claridad. Los orígenes de O_4 y O_5 coinciden, y el origen de O_6 está en el TCP (punto central de la herramienta del robot, o punto final del robot).

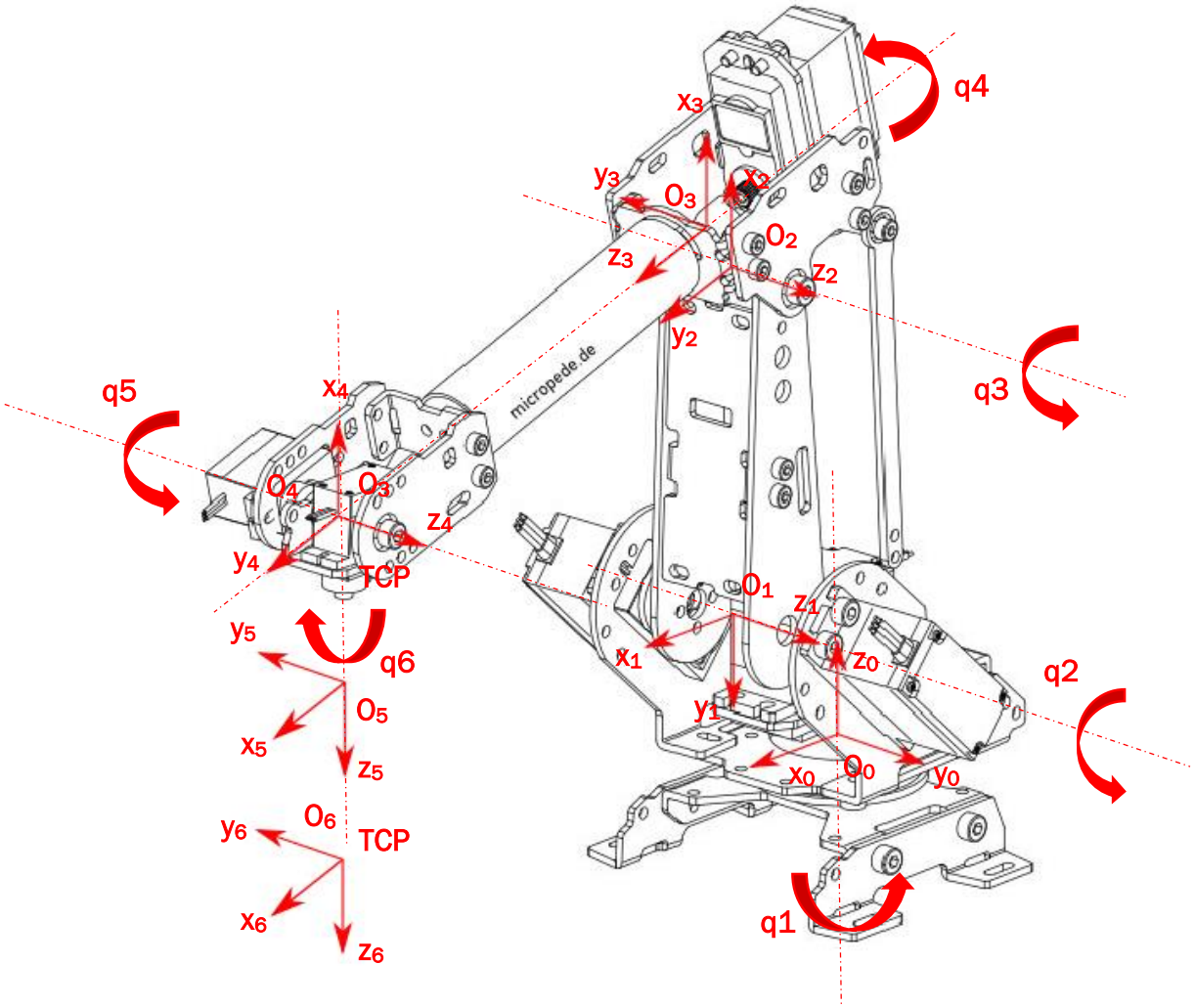


FIGURA 7.3. REPRESENTACIÓN DE LOS SISTEMAS DE REFERENCIA DEL BRAZO ROBÓTICO

El último sistema de referencia es el solidario al elemento terminal y se modela de manera diferente porque no hay articulación (i +1)-ésima. En la mayoría de los robots la última articulación será un par de revolución paralelo a la dirección de aproximación de la pinza. En la Figura 7.4 se representa el último sistema de referencia.

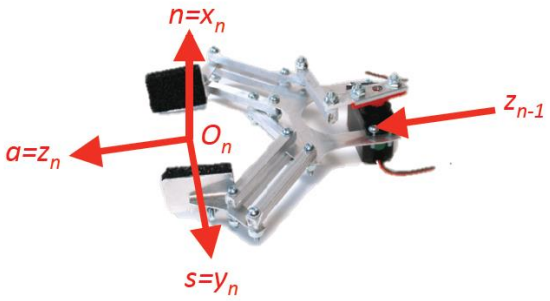


FIGURA 7.4. REPRESENTACIÓN DEL ÚLTIMO SISTEMA DE REFERENCIA

Donde O_n es el punto medio en el extremo de la pinza

Z_n es la dirección de aproximación de la pinza (approach, a)

X_n es la dirección normal a la pinza (normal, n)

Y_n es la dirección de deslizamiento de la pinza (slide, s)

En base a los sistemas de referencia escogidos y a la definición de los 4 parámetros de Denavit-Hartenberg, la **tabla de parámetros D-H** es la siguiente:

	z de partida		x de llegada	
i	θ_i	d_i	a_i	α_i
1	$q1$	$d1$	$a1$	$-90º$
2	$q2$	0	$a2$	0
3	$q3$	0	$a3$	$-90º$
4	$q4$	$d4$	0	$+90º$
5	$q5$	0	0	$-90º$
6	$q6$	$d6$	0	0

TABLA 3. TABLA DE PARÁMETROS D-H

Donde todos los parámetros son constantes excepto θ_i, ya que todas las articulaciones son de rotación.

Las constantes que se corresponden con distancias (Tabla 4) se indican seguidamente en la Figura 7.5.

Donde:

PARÁMETRO	MEDIDA (m)	MEDIDA (mm)
$d1$	0.0465	46.5
$a1$	0.0420	42.0
$a2$	0.0952	95.2
$a3$	0.0185	18.5
$d4$	0.157	157
$d6$	0.020	20.0

TABLA 4. PARÁMETROS CONSTANTES D-H

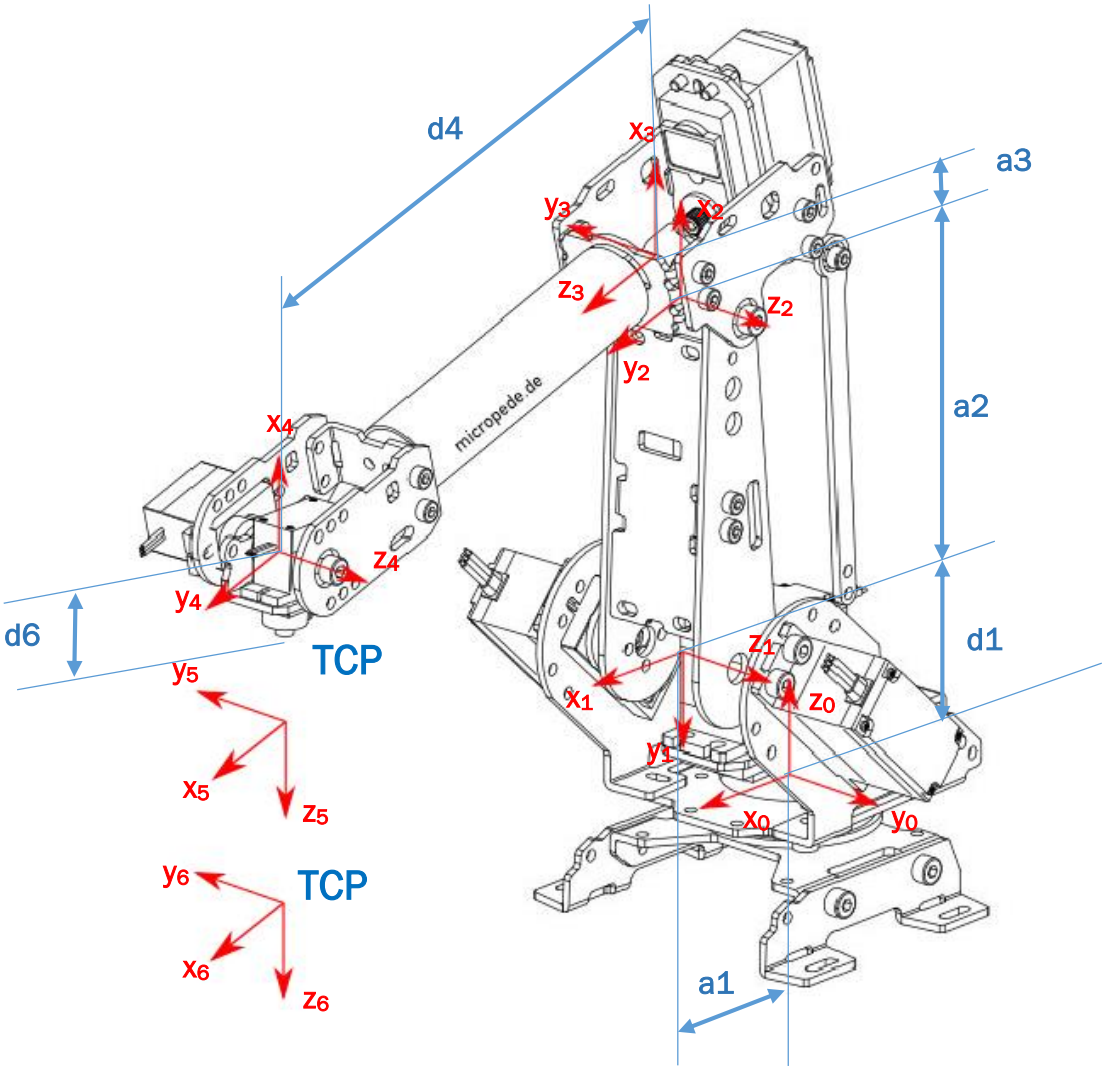


FIGURA 7.5. REPRESENTACIÓN DE LOS PARÁMETROS D-H EN EL BRAZO ROBÓTICO

En robótica se utliza una representación específica para los pares de revolución. En nuestro caso, todas las articulaciones del brazo robótico son de revolución y queda representado como se muestra en la Figura 7.6 (con el cuadrilátero articulado).

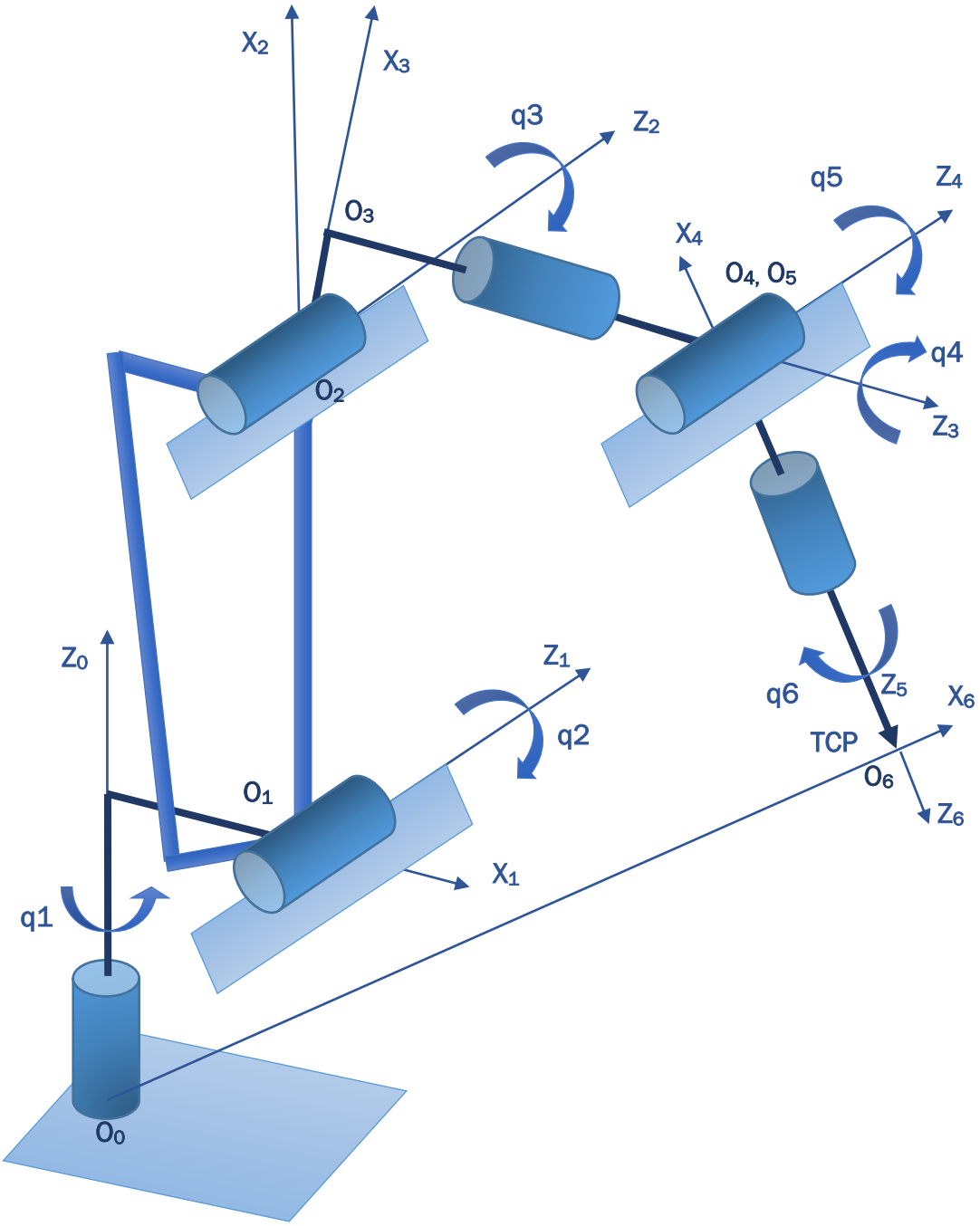


FIGURA 7.6. REPRESENTACIÓN ARTICULACIONES BRAZO ROBÓTICO

La matriz homogénea correspondiente que representa la posición y orientación del extremo del brazo robótico se obtiene como:

$$T = A_0^1(q_1)A_1^2(q_2)A_2^3(q_3)A_3^4(q_4)A_4^5(q_5)A_5^6(q_6) \quad [7.3]$$

$$T = A_0^6(q_1, q_2, q_3, q_4, q_5, q_6) \quad [7.4]$$

La matriz homogénea T se puede escribir como:

$$T = A_0^3(q_1, q_2, q_3)A_3^6(q_4, q_5, q_6) = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [7.5]$$

Donde $A_0^3(q_1, q_2, q_3)$ representa la posición de la muñeca, y $A_3^6(q_4, q_5, q_6)$ representa la orientación de la pinza (vectores n, s, a).

$$n = normal$$

$$s = slide = deslizamiento$$

$$a = approach = aproximación$$

Cálculo de la matriz de transformación homogénea para la primera articulación con la ecuación [7.1]:

$$A_0^1 = \begin{bmatrix} \cos(q_1) & -sen(q_1) & 0 & 0 \\ sen(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-90) & -sen(-90) & 0 \\ 0 & sen(-90) & \cos(-90) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_0^1 = \begin{bmatrix} C_1 & 0 & S_1 & C_1 a_1 \\ S_1 & 0 & -C_1 & S_1 a_1 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [7.6]$$

Donde $C_1 = \cos(q_1)$ y $S_1 = sen(q_1)$. Utilizaremos esta nomenclatura en lo que sigue para todas las variables articulares.

Cálculo de la matriz de transformación homogénea para la segunda articulación:

$$A_1^2 = \begin{bmatrix} \cos(q_2) & -sen(q_2) & 0 & 0 \\ sen(q_2) & \cos(q_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} I_4 \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} I_4$$

$$A_1^2 = \begin{bmatrix} C_2 & -S_2 & 0 & C_2 a_2 \\ S_2 & C_2 & 0 & S_2 a_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [7.7]$$

Donde I_4 es la matriz identidad 4x4.

Cálculo de la matriz de transformación homogénea para la tercera articulación:

$$A_2^3 = \begin{bmatrix} \cos(q_3) & -sen(q_3) & 0 & 0 \\ sen(q_3) & \cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} I_4 \begin{bmatrix} 1 & 0 & 0 & a_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2^3 = \begin{bmatrix} C_3 & 0 & S_3 & C_3 a_3 \\ S_3 & 0 & -C_3 & S_3 a_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [7.8]$$

Cálculo de la matriz de transformación homogénea para la cuarta articulación:

$$A_3^4 = \begin{bmatrix} \cos(q_4) & -sen(q_4) & 0 & 0 \\ sen(q_4) & \cos(q_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$I_4 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(90) & -sen(90) & 0 \\ 0 & sen(90) & \cos(90) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^4 = \begin{bmatrix} C_4 & 0 & S_4 & 0 \\ S_4 & 0 & -C_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [7.9]$$

Cálculo de la matriz de transformación homogénea para la quinta articulación:

$$A_4^5 = \begin{bmatrix} \cos(q_5) & -sen(q_5) & 0 & 0 \\ sen(q_5) & \cos(q_5) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} I_4 I_4 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-90) & -sen(-90) & 0 \\ 0 & sen(-90) & \cos(-90) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4^5 = \begin{bmatrix} C_5 & 0 & -S_5 & 0 \\ S_5 & 0 & C_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [7.10]$$

Cálculo de la matriz de transformación homogénea para la sexta articulación:

$$A_5^6 = \begin{bmatrix} \cos(q_6) & -sen(q_6) & 0 & 0 \\ sen(q_6) & \cos(q_6) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} I_4 I_4$$

$$A_5^6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [7.11]$$

De las ecuaciones [7.4] y [7.5] se tiene que:

$$T = A_0^6(q_1, q_2, q_3, q_4, q_5, q_6) = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [7.12]$$

Las coordenadas que representan la posición del brazo robótico, así como los vectores que representan la orientación del mismo ($\boldsymbol{n}, \boldsymbol{s}, \boldsymbol{a}$) se obtienen igualando términos en ambas matrices.

Por ejemplo, el término que determina la coordenada px es el siguiente:

$$px = (-C_1C_2C_3C_4C_5 + C_1S_2S_3C_4S_5 + S_1S_4S_5 - C_1C_2S_3C_5 - C_1S_2C_3C_5)d_6$$
$$-C_1S_2S_3d_4 - C_1S_2C_3d_4 + C_1C_2C_3a_3 - C_1S_2S_3a_3 + C_1C_2a_2 + C_1a_1 \quad [7.13]$$

El término que determina la coordenada py:

$$py = (-S_1C_2C_3C_4S_5 + S_1S_2S_3C_4S_5 - C_1S_4S_5 - S_1C_2S_3C_5 - S_1S_2C_3C_5)d_6 -$$
$$S_1C_2S_3d_4 - S_1S_2C_3d_4 + S_1C_2C_3a_3 - S_1S_2S_3a_3 + S_1C_2a_2 + S_1a_1 \quad [7.14]$$

El término que determina la coordenada pz:

$$pz = (S_2C_3C_4S_5 + C_2S_3C_4S_5 + S_2S_3C_5 - C_2C_3C_5)d_6 -$$
$$S_2S_3d_4 - C_2C_3d_4 + S_2C_3a_3 - C_2S_3a_3 - S_2a_2 + d_1 \quad [7.15]$$

7.3 RESOLUCIÓN DEL CUADRILÁTERO ARTICULADO Y RELACIÓN SERVOS – COORDENADAS ARTICULARES

A continuación se resuelve el cuadrilátero articulado, mediante el que calculamos el valor de la posición de los servos 2 y 3 en función del valor de las variables articulares $q2$ y $q3$, mediante relaciones trigonométricas.

En la figura 7.7 se representa el cuadrilátero articulado sobre el brazo robótico:

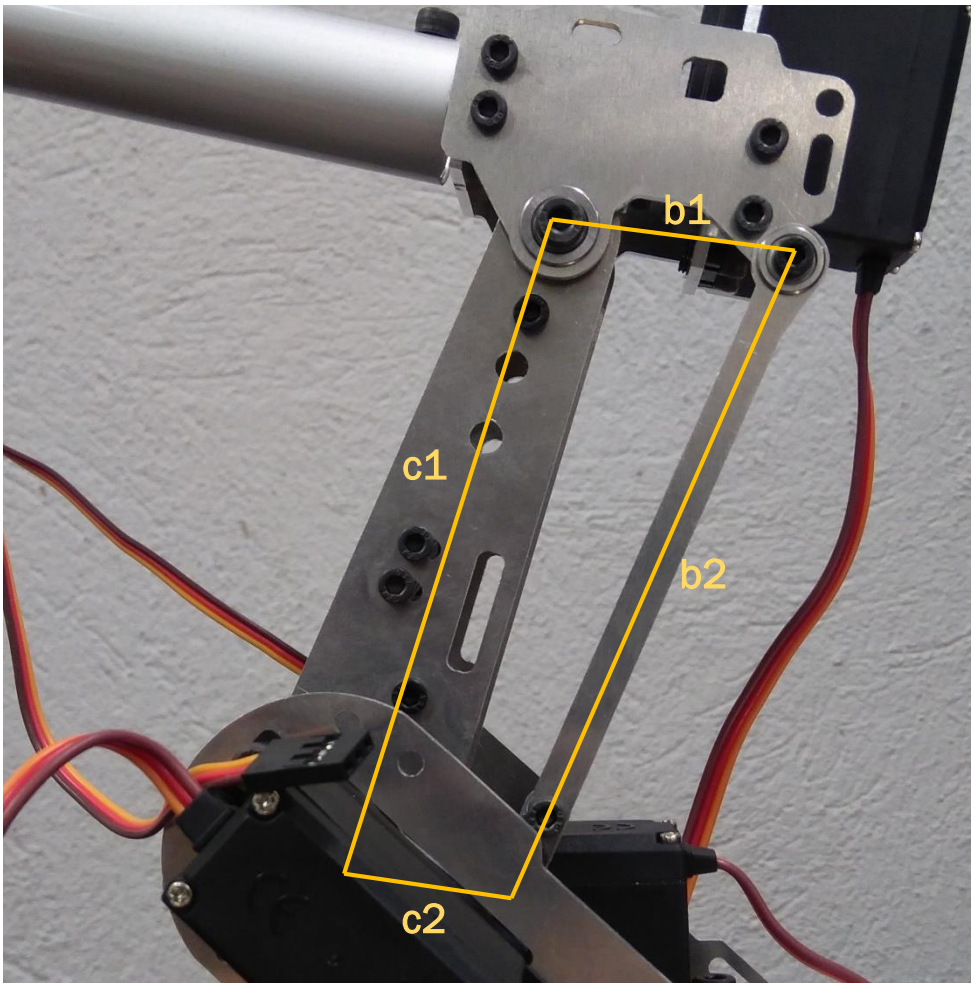


FIGURA 7.7. MEDIDAS DEL CUADRILÁTERO ARTICULADO SOBRE EL BRAZO ROBÓTICO

En el brazo robótico objeto de estudio, el valor de la tercera variable articular q_3 depende no sólo de la posición del tercer servo, sino de la posición del segundo y del tercer servo a la vez.

También, conocidas la posición del segundo servo y del tercer servo, se conoce el valor de q_3, a través de relaciones trigonométricas.

Se han tomado los siguientes valores de los lados del cuadrilátero:

$$c1 = 95{,}2 \text{ mm}$$
$$b1 = 33 \text{ mm}$$
$$b2 = 102{,}9 \text{ mm} \quad [7.16]$$
$$c2 = 26{,}2 \text{ mm}$$

En la Figura 7.8 se indican los parámetros para resolver el cuadrilátero articulado:

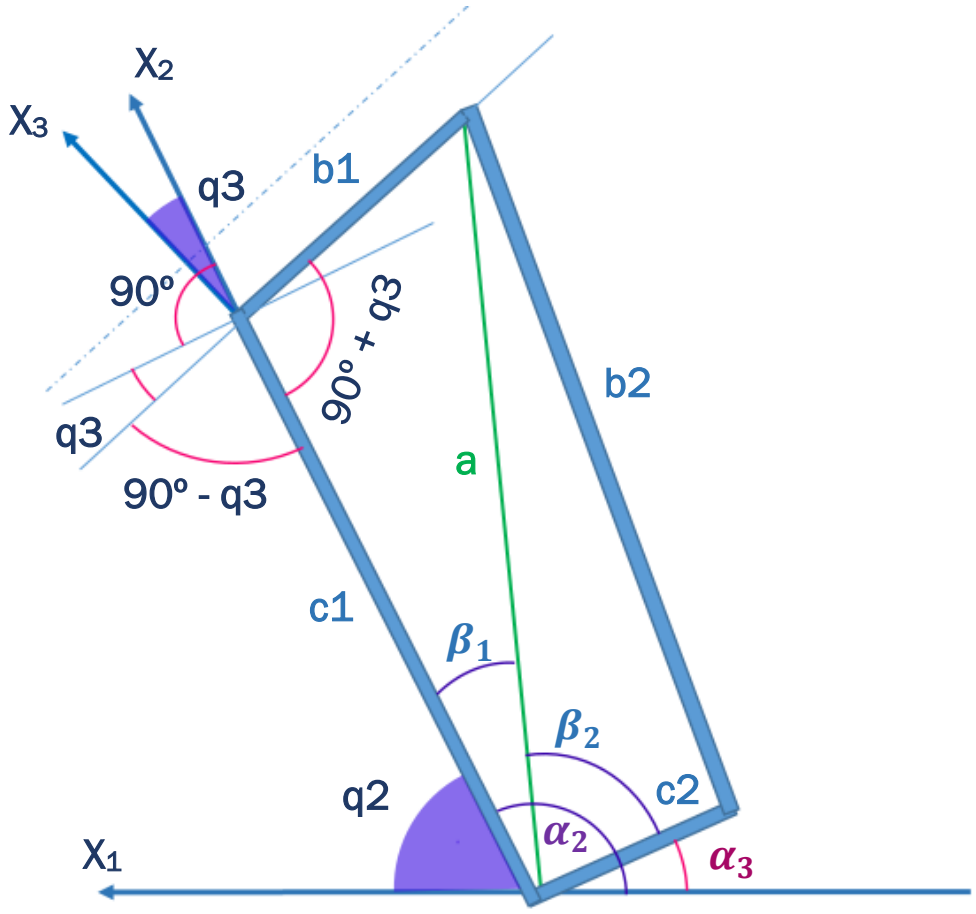


FIGURA 7.8. PARÁMETROS PARA LA RESOLUCIÓN DEL CUADRILÁTERO ARTICULADO

Resolvemos el cuadrilátero: q_3 es la tercera variable articular y es conocida, así como es conocido q_2.

De la geometría tenemos que el ángulo entre la pieza "b1" y la pieza "c1" (medidas indicadas en [7.16]), es de $(90 + q_3)$ grados.

Teorema del coseno para el triángulo $\widehat{c1\ b1}\ a$:

$$(a)^2 = (b_1)^2 + (c_1)^2 - 2\ b_1\ c_1 \cos(90 + q_3) \qquad [7.17]$$

Calculamos la distancia a con la ecuación [7.17]:

$$a = \sqrt{(b_1)^2 + (c_1)^2 - 2\ b_1\ c_1 \cos(90 + q_3)} \qquad [7.18]$$

Para ambos triángulos la distancia a es la misma.

$$a = a_1 = a_2 \qquad [7.19]$$

Teorema del coseno para el triángulo $\widehat{c1\ b1}\ a$:

$$(b_1)^2 = (a_1)^2 + (c_1)^2 - 2\ a_1\ c_1 \cos(\beta_1) \qquad [7.20]$$

Teorema del coseno para el triángulo $\widehat{c2\ b2}\ a$:

$$(b_2)^2 = (a_2)^2 + (c_2)^2 - 2\ a_2\ c_2 \cos(\beta_2) \qquad [7.21]$$

Calculamos los siguientes ángulos con las ecuaciones anteriores [7.20] y [7.21]:

$$\beta_1 = \operatorname{acos}\left[\frac{(a_1)^2+(c_1)^2-(b_1)^2}{2 \cdot a_1 \cdot c_1}\right] \qquad [7.22]$$

$$\beta_2 = \operatorname{acos}\left[\frac{(a_2)^2+(c_2)^2-(b_2)^2}{2 \cdot a_2 \cdot c_2}\right] \qquad [7.23]$$

Recordamos **DH10**: Obtener θ_i como el ángulo que hay que girar en torno a z_{i-1} para que x_{i-1} y x_i queden paralelos. Según DH10, la variable articular q_2 es el ángulo que hay que girar en torno a z_1 para que x_1 y x_2 queden alineados. Así también tenemos que q_3 es el ángulo que hay que girar en torno a z_2 para que x_2 y x_3 queden alineados.

Resolviendo la geometría, nos queda la siguiente ecuación:

$$\alpha_2 = 180 - q_2 \quad [7.24]$$

Pero dado que q_2 crece en sentido contrario, es negativo en la ecuación [7.24] y nos queda la siguiente ecuación que relaciona ambos parámetros:

$$\alpha_2 = 180 + q_2 \quad [7.25]$$

Calculamos β como:

$$\beta = \beta_1 + \beta_2 \quad [7.26]$$

De la geometría tenemos que: α_2 es el ángulo que forma la pieza "c1" con la horizontal, y α_3 es el ángulo que forma la pieza "c2" con la misma horizontal.

Relacionando β (que depende de la variable articular q_3) con los ángulos anteriores:

$$\boldsymbol{\alpha_3 = \alpha_2 - \beta} \quad [7.27]$$

Las relaciones que se establecen entre las variables articulares q_2 y q_3, y los servos 2 y 3, se obtienen a continuación.

Las posiciones límite del servo 2 se indican en la Figura 7.9:

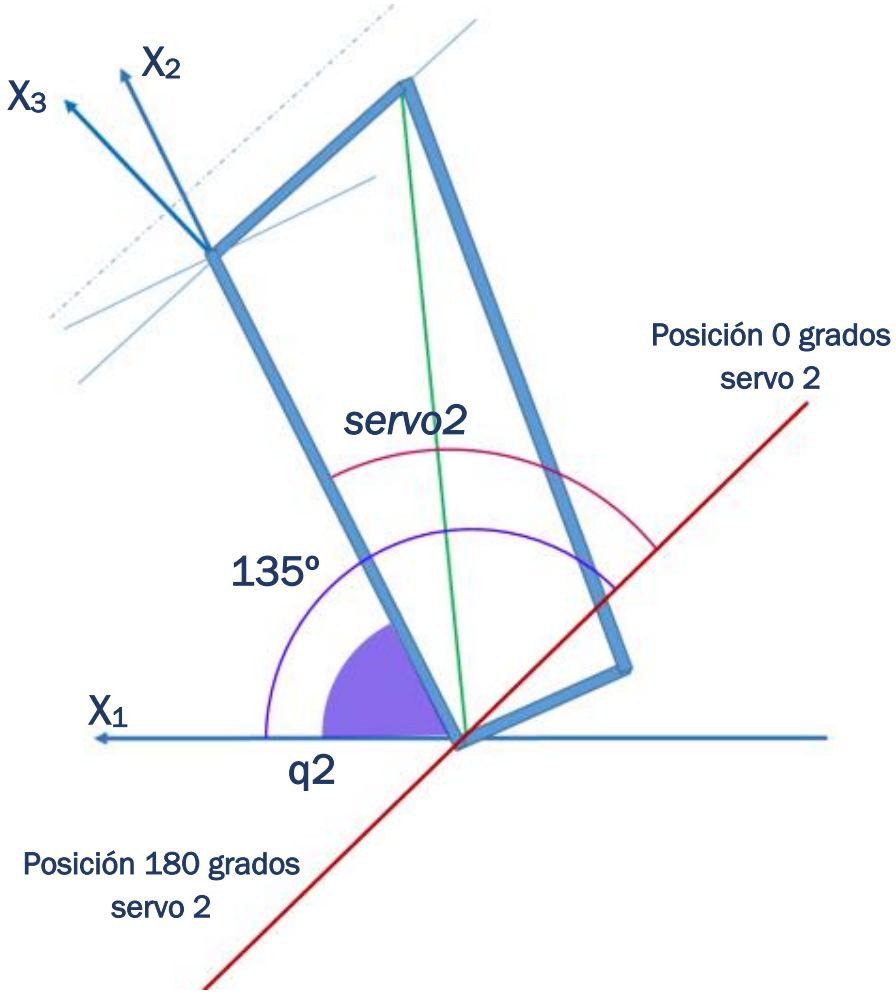


FIGURA 7.9. PARÁMETROS PARA LA RESOLUCIÓN DE LA POSICIÓN DEL SERVO 2

El eje x_1 forma 135 grados con el eje que determina la posición 0 grados del servo 2 (Figura 7.9), por tanto:

$$servo2 + q_2 - 135 = 0 \qquad [7.28]$$

Por la misma razón de antes, teniendo en cuenta DH10, q_2 es negativo según los sistemas de referencia escogidos, y por tanto:

$$servo2 - q_2 - 135 = 0 \qquad [7.29]$$

Las posiciones límite del servo 3 se indican en la Figura 7.10:

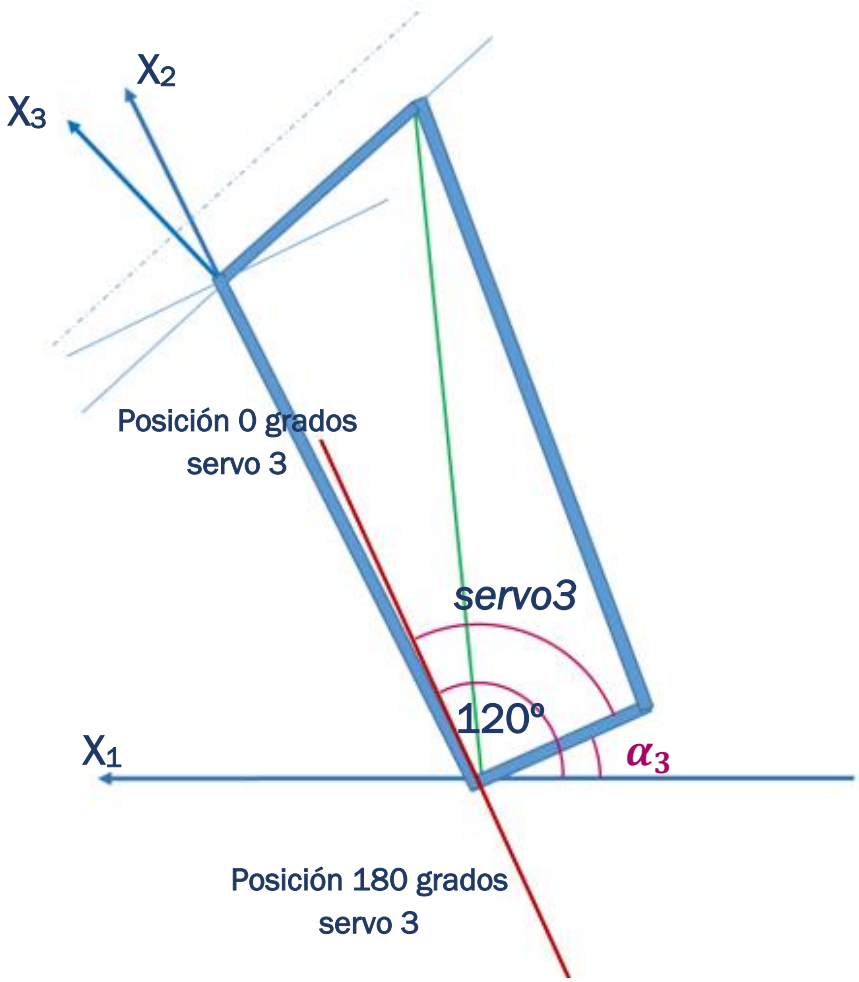


FIGURA 7.10. PARÁMETROS PARA LA RESOLUCIÓN DE LA POSICIÓN DEL SERVO 3

El eje x_1 forma 120 grados con el eje que determina la posición 0 grados del servo 3. El servo 3 aumenta los grados en sentido horario (Figura 7.10), por tanto, considerando α_3 positivo según el dibujo de las Figuras 7.8 y 7.10:

$$servo3 + \alpha_3 - 120 = 0 \qquad [7.30]$$

Por tanto, el **ángulo correspondiente en grados** que tiene que marcar cada servo, corregido por la posición en la que montamos los servos durante la fase de montaje del brazo robótico, son los siguientes (de las ecuaciones [7.29] y [7.30]):

$$\boldsymbol{servo2 = 135^{\circ} + q_2} \qquad [7.31]$$

$$\boldsymbol{servo3 = 120^{\circ} - \alpha_3} \qquad [7.32]$$

Para el resto de los servos, resolvemos la posición sumando 90 grados a cada coordenada articular, puesto que hemos conectado todos los servos ($servo1, servo4, servo5\ y\ servo6$) de forma que cuando la posición de los mismos es de 0 grados, las coordenadas articulares también lo son.

Las coordenadas articulares $q_1, q_4, q_5 y\ q_6$ van de $-90\ a\ 90\ grados$, y el recorrido de los servos de $0\ a\ 180\ grados$. Por tanto, el ángulo que tiene que marcar cada servo, en grados:

$$\begin{aligned} \boldsymbol{servo1} &= \boldsymbol{q_1} + \boldsymbol{90º} \\ \boldsymbol{servo4} &= \boldsymbol{q_4} + \boldsymbol{90º} \\ \boldsymbol{servo5} &= \boldsymbol{q_5} + \boldsymbol{90º} \\ \boldsymbol{servo6} &= \boldsymbol{q_6} + \boldsymbol{90º} \end{aligned} \qquad [7.33]$$

7.4 MODELO CINEMÁTICO INVERSO

El objetivo del problema cinemático inverso consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot para que su extremo se posicione y oriente según una determinada localización espacial.

El procedimiento de obtención de las ecuaciones del problema cinemático inverso depende fuertemente de la configuración específica de cada robot. La resolución por **métodos geométricos** es adecuada para robots de pocos grados de libertad o para el caso de que se consideren sólo los primeros grados de libertad, dedicados a posicionar el extremo. También se puede resolver el problema cinemático inverso **a partir de las matrices de transformación homogéneas**.

El **procedimiento geométrico** se basa en encontrar suficiente número de relaciones geométricas en las que intervendrán las coordenadas del extremo del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos. Este procedimiento permite obtener los valores de las tres primeras variables articulares del robot, aquellas que posicionan su extremo en unas coordenadas px, py, pz determinadas. La resolución con este procedimiento es **específica para cada robot**.

No basta con posicionar el extremo del robot en un punto del espacio, sino que es preciso orientar de una manera determinada la herramienta. Para ello, los robots cuentan con otros tres grados de libertad, situados al final de la cadena cinemática y cuyos ejes, con frecuencia se cortan en un punto, que se llamará informalmente muñeca del robot.

Si bien la variación de estos tres últimos grados de libertad origina un cambio en la posición final del extremo real del robot, su verdadero objetivo es poder orientar la herramienta del robot libremente en el espacio.

El método de **desacoplo cinemático** es aplicable a aquellos robots cuyos tres últimos grados de libertad se cortan en un punto, sacando partido de este hecho, separando los problemas de obtención del modelo cinemático inverso de posición y orientación. Para ello, dada una posición y orientación final deseadas, establece las **coordenadas del punto de corte de los tres últimos ejes (muñeca del robot)** calculándose los valores de las tres primeras variables articulares que consiguen posicionar ese punto. A continuación, a partir de los datos de orientación deseada para el extremo del robot y de los ya calculados, se obtienen los valores del resto de las variables articulares que consiguen la orientación deseada.

RESOLUCIÓN DEL PROBLEMA CINEMÁTICO INVERSO CON MATRICES DE TRANSFORMACIÓN HOMOGÉNEA

En principio es posible obtener el modelo cinemático inverso de un robot a partir del conocimiento del modelo directo. Sin embargo, este procedimiento no es trivial en muchas ocasiones. Puesto que el problema cinemático directo, en el caso de un robot de 6 grados de libertad, contiene 12 ecuaciones, y se buscan sólo 6 relaciones (una por cada grado de libertad), existirán necesariamente ciertas dependencias entre las 12 ecuaciones de partida, con lo que la elección de las ecuaciones a escoger debe hacerse con sumo cuidado.

El primer paso es conocer la matriz T, que relaciona el sistema de referencia $\{S_0\}$, con sistema de referencia $\{S_6\}$ asociado a su extremo. Obtenida la expresión de T en función de las coordenadas articulares $(q1, q2, q3)$ y supuesta una localización de destino para el extremo del robot definida por los vectores $\boldsymbol{n}, \boldsymbol{o}, \boldsymbol{a}$ y $\boldsymbol{p}$ se podría intentar manipular directamente las 12 ecuaciones resultantes de T a fin de despejar $q1, q2\ y\ q3$ en función de n, o, a, p. Este procedimiento directo es complicado y suele ser más adecuado aplicar el siguiente:

Puesto que:

$$T = A_0^1 A_1^2 A_2^3 \qquad [7.34]$$

Se tiene que:

$$(A_0^1)^{-1} T = A_1^2 A_2^3 \qquad [7.35]$$

$$(A_0^1)^{-1} (A_1^2)^{-1} T = A_2^3 \qquad [7.36]$$

Para poder aplicar este procedimiento es necesario obtener las inversas de las matrices A. De la ecuación [7.35] interesan las relaciones que expresan $q1$ en función de constantes y no de $q2$ y $q3$. De la ecuación [7.36] se obtendrían de las relaciones adecuadas las expresiones de $q2$ y $q3$.

RESOLUCIÓN DEL DESACOPLO CINEMÁTICO

Para obtener los valores del resto de las variables articulares $q4, q5, q6$ que consiguen la orientación deseada, nos quedamos con las submatrices de rotación de las matrices A y T. Para ello, denominamos R_0^6 a la submatriz de rotación de T_0^6:

$$R = R_0^6 = R_0^3(q_1, q_2, q_3)R_3^6(q_4, q_5, q_6) \quad [7.37]$$

La matriz R es conocida, por ser la orientación deseada del robot.

$$R = \begin{bmatrix} n_x & s_x & a_x \\ n_y & s_y & a_y \\ n_z & s_z & a_z \end{bmatrix} \quad [7.38]$$

Se tiene que:

$$R_3^6 = (R_0^3)^{-1}R_0^6 = (R_0^3)^T[\boldsymbol{n} \quad \boldsymbol{o} \quad \boldsymbol{a}] \quad [7.39]$$

Siendo el término de la derecha una matriz cuyas componentes numéricas son conocidas. La matriz R_3^6 corresponde con la submatriz de rotación de la matriz de transformación homogénea T_3^6 que relaciona el sistema $\{S_3\}$con el $\{S_6\}$.Por tanto:

$$R_3^6 = R_3^4 R_4^5 R_5^6 \quad [7.40]$$

Se obtienen de la ecuación [7.40] los valores de $q4, q5, q6$, con lo que quedaría resuelta la orientación del extremo del robot.

7.4.1 DESARROLLO DE LA CINEMÁTICA INVERSA

Para el caso del brazo robótico que estamos estudiando, se da el caso que se acaba de describir. Los tres últimos ejes concurren en un punto, con indicación de los sistemas de coordenadas asociados según el procedimiento de Denavit-Hartenberg, cuyos parámetros ya hemos obtenido anteriormente (Tabla 3).

Por tanto, se determinan $q1, q2\ y\ q3$ (articulaciones de la cintura, hombro y codo, respectivamente) con el procedimiento geométrico en este caso. Seguidamente se obtienen $q4, q5\ y\ q6$ con el método de desacoplo cinemático, conocidas ya las tres primeras variables articulares.

DESARROLLO DEL PROCEDIMIENTO GEOMÉTRICO

El punto final del robot será el origen del sistema $\{S_6\}$: O_6. Los orígenes de los sistemas $\{S_4\}$ y $\{S_5\}$ son coincidentes. Se utilizarán los vectores muñeca $\boldsymbol{p_m}$ y extremo del robot $\boldsymbol{p_r}$, que van desde el origen del sistema $\{S_0\}$ hasta los puntos centro de la muñeca y fin del robot, respectivamente.

$$\boldsymbol{p_m} = \overline{O_0O_5} = \overline{O_0O_4} \quad [7.41]$$

$$\boldsymbol{p_r} = \overline{O_0O_6} \quad [7.42]$$

Puesto que de acuerdo a la regla DH9, la dirección del eje $\boldsymbol{z_6}$ debe coincidir con la de $\boldsymbol{z_5}$ y la distancia entre O_5 y O_6 medida a lo largo del eje $\boldsymbol{z_5}$ es precisamente el parámetro $d6$, se tiene que:

$$\boldsymbol{p_m} = \boldsymbol{p_r} - d6 \cdot \boldsymbol{z_6} \quad [7.43]$$

El vector director $\boldsymbol{z_6}$ es el vector $\boldsymbol{a}$ correspondiente a la orientación deseada:

$$\boldsymbol{z_6} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \quad [7.44]$$

En la expresión [7.43] anterior $\boldsymbol{p_r}$ son las coordenadas del punto donde se pretende que se posicione el robot expresadas en $\{S_0\}$ (punto final del robot).

$$\boldsymbol{p_r} = T(1{:}3, 4) = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad [7.45]$$

Nos referiremos a la posición de la muñeca con las coordenadas px, py, pz, y calculamos a continuación:

$$\boldsymbol{p_m} = \overline{O_0O_5} = \overline{O_0O_4} = \boldsymbol{O_4} \quad [7.46]$$

Obteniendo la posición de componentes de la matriz T:

$$\boldsymbol{O_4} = T(1{:}3, 4) - d6 \cdot T(1{:}3, 3) \quad [7.47]$$

Cada coordenada, de la ecuación [7.47]:

$$\begin{aligned} px &= \boldsymbol{O_4}(1,1) \\ py &= \boldsymbol{O_4}(2,1) \\ pz &= \boldsymbol{O_4}(3,1) \end{aligned} \quad [7.48]$$

Las unidades de las coordenadas de la muñeca son las mismas que las unidades de los parámetros de la tabla D-H (Tabla 3).

Con estos datos que hemos obtenido en la ecuación [7.48], ya podemos **calcular las tres primeras coordenadas articulares**. Las ecuaciones de $q1, q2\ y\ q3$ se obtienen **con el procedimiento geométrico en este caso**.

A continuación se representa el brazo robótico en una determinada posición donde se indican los parámetros geométricos. Se obtienen las coordenadas articulares para cualquier posición, donde los parámetros que se calculan serían los equivalentes para otras posiciones a los representados en la posición de la Figura 7.11.

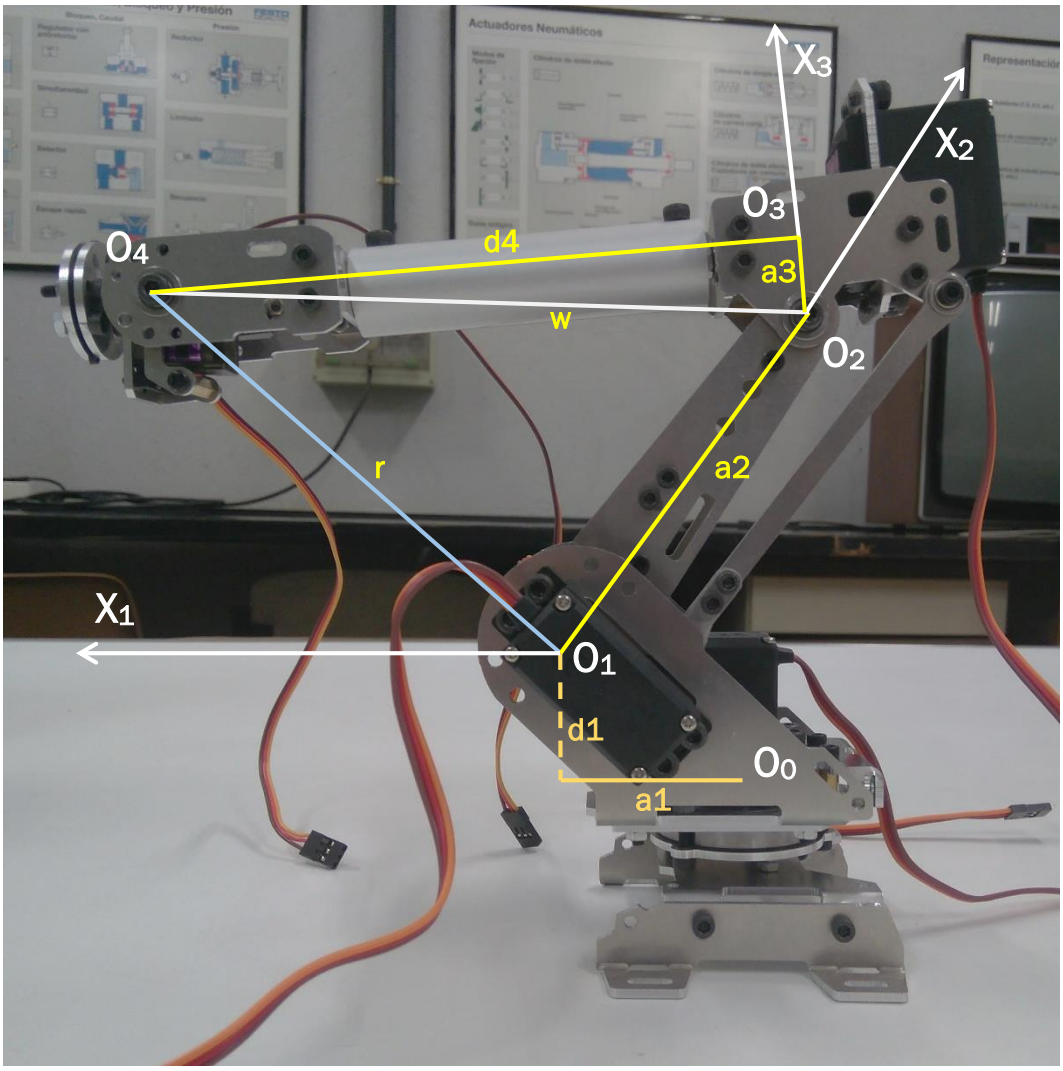


FIGURA 7.11. PARÁMETROS PARA LA RESOLUCIÓN DEL PROBLEMA CINEMÁTICO INVERSO CON EL PROCEDIMIENTO GEOMÉTRICO

El origen de referencia es el sistema $\{S_0\}$. La **primera coordenada articular** se obtiene inmediatamente conociendo las coordenadas px y py de la posición de la muñeca:

$$q1 = atan2(py, px) \qquad [7.49]$$

El brazo robótico queda siempre en un plano determinado por el ángulo $q1$. Como O_0 y O_1 no están en la misma vertical, si no separados una distancia $a1$ (distancia medida en la dirección del eje $x1$), calculamos:

$$u = \sqrt{px^2 + py^2} - a1 \quad [7.50]$$

$$s = d1 - pz \quad [7.51]$$

$$r^2 = s^2 + u^2 \quad [7.52]$$

Donde r es la recta que une O_1 con la muñeca (O_4). Una vez conocido el vector r, calculamos los siguientes parámetros:

$$\theta = \tan(-s/u) \quad [7.53]$$

$$w = \sqrt{(d4)^2 + (a3)^2} \quad [7.54]$$

$$\alpha = tan\left({}^{a3}/_{d4}\right) \quad [7.55]$$

Donde w y α son siempre constantes, ya que dependen de parámetros que lo son.

Por el teorema del coseno, obtenemos el ángulo:

$$q2' = -acos\left[\frac{(r)^2+(a2)^2+(w)^2}{2\,r\,(a2)}\right] \quad [7.56]$$

Finalmente, obtenemos el valor de la segunda variable articular:

$$q2 = q2' - \theta \quad [7.57]$$

Con relaciones trigonométricas, obtenemos:

$$q3' = \tan\left[\frac{r\,cos(q2) - a2}{-r\,sen(q2)}\right] \quad [7.58]$$

La tercera variable articular, por tanto:

$$q3 = \alpha - q3' \quad [7.59]$$

Una vez calculadas **las 3 primeras variables articulares**, ya son conocidas en la tabla de parámetros de Denavit-Hartenberg:

$$q1 = atan2(py, px) = dh(1,1)$$

$$q2 = -\,acos\left[\frac{(r)^2+(a2)^2+(w)^2}{2\,r\,(a2)}\right] - \theta = dh(2,1) \qquad [7.60]$$

$$q3 = \alpha - tan\left[\frac{r\,cos(q2) - a2}{-\,r\,sen(q2)}\right] = dh(3,1)$$

NOTA: el procedimiento de resolución del problema cinemático inverso con matrices de transformación homogéneas nos proporcionaría la misma solución para las tres primeras variables articulares.

DESARROLLO DEL DESACOPLO CINEMÁTICO

Aplicando la ecuación [7.39], que se repite a continuación:

$$R_3^6 = (R_0^3)^{-1} R_0^6 = (R_0^3)^T [\boldsymbol{n} \quad \boldsymbol{o} \quad \boldsymbol{a}]$$

Obtenemos los valores de $q4, q5, q6$ que consiguen la orientación deseada del robot. Denominamos R_0^6 a la submatriz de rotación de T_0^6:

Si nos quedamos con la submatriz de rotación de A_0^1:

$$R_0^1 = \begin{bmatrix} \cos(q_1) & -sen(q_1) & 0 \\ sen(q_1) & \cos(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} \cos(q_1) & 0 & -sen(q_1) \\ sen(q_1) & 0 & \cos(q_1) \\ 0 & -1 & 0 \end{bmatrix}$$

[7.61]

La submatriz de rotación de A_1^2:

$$R_1^2 = \begin{bmatrix} \cos(q_2) & -sen(q_2) & 0 \\ sen(q_2) & \cos(q_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad [7.62]$$

La submatriz de rotación de A_2^3:

$$R_2^3 = \begin{bmatrix} \cos(q_3) & -sen(q_3) & 0 \\ sen(q_3) & \cos(q_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} \cos(q_3) & 0 & -sen(q_3) \\ sen(q_3) & 0 & \cos(q_3) \\ 0 & -1 & 0 \end{bmatrix}$$

[7.63]

Calculamos la composición de matrices de rotación:

$$R_0^3 = R_0^1 R_1^2 R_2^3 =$$

$$= \begin{bmatrix} \cos(q_1) & 0 & -sen(q_1) \\ sen(q_1) & 0 & \cos(q_1) \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \cos(q_2) & -sen(q_2) & 0 \\ sen(q_2) & \cos(q_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(q_3) & 0 & -sen(q_3) \\ sen(q_3) & 0 & \cos(q_3) \\ 0 & -1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(q_1)\cos(q_2) & -\cos(q_1)\,sen(q_2) & -sen(q_1) \\ sen(q_1)\cos(q_2) & -sen(q_1)sen(q_2) & \cos(q_1) \\ -sen(q_2) & -\cos(q_2) & 0 \end{bmatrix} \begin{bmatrix} \cos(q_3) & 0 & -sen(q_3) \\ sen(q_3) & 0 & \cos(q_3) \\ 0 & -1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{C_1C_2C_3} - \mathrm{C_1S_2S_3} & \mathrm{S_1} & -\mathrm{C_1C_2S_3} - \mathrm{C_1S_2C_3} \\ \mathrm{S_1C_2C_3} - \mathrm{S_1S_2S_3} & -\mathrm{C_1} & -\mathrm{S_1C_2S_3} - \mathrm{S_1S_2C_3} \\ -\mathrm{S_2C_3} - \mathrm{C_2S_3} & 0 & \mathrm{S_2S_3} - \mathrm{C_2C_3} \end{bmatrix} \quad [7.64]$$

Para las últimas tres articulaciones, multiplicamos todas las matrices de rotación siguiendo el mismo ejemplo (quitando las constantes):

$$R_z(q_4)R_x(90)R_z(q_5)R_x(-90)R_z(q_6) = R_3^4 R_4^5 R_5^6 \quad [7.65]$$

Calculamos:

$$R_3^4 = R_z(q_4)R_x(90) = \begin{bmatrix} \cos(q_4) & -sen(q_4) & 0 \\ sen(q_4) & \cos(q_4) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad [7.66]$$

$$R_3^4 = \begin{bmatrix} \mathrm{C_4} & 0 & \mathrm{S_4} \\ \mathrm{S_4} & 0 & -\mathrm{C_4} \\ 0 & 1 & 0 \end{bmatrix}$$

$$R_4^5 = R_z(q_5)R_x(-90) \quad [7.67]$$

$$R_3^4 R_z(q_5) = \begin{bmatrix} \mathrm{C_4} & 0 & \mathrm{S_4} \\ \mathrm{S_4} & 0 & -\mathrm{C_4} \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{C_5} & -\mathrm{S_5} & 0 \\ \mathrm{S_5} & \mathrm{C_5} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathrm{C_4C_5} & -\mathrm{C_4S_5} & \mathrm{S_4} \\ \mathrm{S_4C_5} & -\mathrm{S_4S_5} & -\mathrm{C_4} \\ \mathrm{S_5} & \mathrm{C_5} & 0 \end{bmatrix} \quad [7.68]$$

$$R_3^4 R_4^5 = \left(R_3^4 R_z(q_5)\right) R_x(-90) = \begin{bmatrix} \mathrm{C_4C_5} & -\mathrm{C_4S_5} & \mathrm{S_4} \\ \mathrm{S_4C_5} & -\mathrm{S_4S_5} & -\mathrm{C_4} \\ \mathrm{S_5} & \mathrm{C_5} & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \quad [7.69]$$

$$R_3^4 R_4^5 = \begin{bmatrix} \mathrm{C_4C_5} & -\mathrm{S_4} & -\mathrm{C_4S_5} \\ \mathrm{S_4C_5} & \mathrm{C_4} & -\mathrm{S_4S_5} \\ \mathrm{S_5} & 0 & \mathrm{C_5} \end{bmatrix} \quad [7.70]$$

$$R_5^6 = R_z(q_6) \qquad [7.71]$$

$$R_3^6 = R_3^4 R_4^5 R_5^6 = (R_3^4 R_4^5) R_z(q_6) =$$

$$= \begin{bmatrix} C_4C_5 & -S_4 & -C_4S_5 \\ S_4C_5 & C_4 & -S_4S_5 \\ S_5 & 0 & C_5 \end{bmatrix} \begin{bmatrix} \cos(q_6) & -sen(q_6) & 0 \\ sen(q_6) & \cos(q_6) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad [7.72]$$

La composición de matrices de rotación R_3^6 queda de la siguiente manera:

$$R_3^6 = \begin{bmatrix} C_4C_5C_6 - S_4S_6 & -C_4C_5S_6 - S_4C_6 & -C_4S_5 \\ S_4C_5C_6 + C_4S_6 & -S_4C_5S_6 + C_4C_6 & -S_4S_5 \\ S_5C_6 & -S_5S_6 & C_5 \end{bmatrix} \qquad [7.73]$$

Bloqueo (singularidad) cuando $\cos(q_5) = \pm 1$ en la matriz [7.73], ya que $sen(q_5) = 0$

Si $\cos(q_5) = +1$,

$$R_3^6 = \begin{bmatrix} C_4C_6 - S_4S_6 & -C_4S_6 - S_4C_6 & 0 \\ S_4C_6 + C_4S_6 & -S_4S_6 + C_4C_6 & 0 \\ 0 & 0 & +1 \end{bmatrix} \qquad [7.74]$$

Resolvemos (al ser una matriz de rotación, la inversa es igual que la traspuesta):

$$R_3^6 = (R_0^3)^{-1} R = (R_0^3)^T \begin{bmatrix} n_x & s_x & a_x \\ n_y & s_y & a_y \\ n_z & s_z & a_z \end{bmatrix} \qquad [7.75]$$

$$R_3^6 = \begin{bmatrix} C_1C_2C_3 - C_1S_2S_3 & S_1C_2C_3 - S_1S_2S_3 & -S_2C_3 - C_2S_3 \\ S_1 & -C_1 & 0 \\ -C_1C_2S_3 - C_1S_2C_3 & -S_1C_2S_3 - S_1S_2C_3 & S_2S_3 - C_2C_3 \end{bmatrix} R =$$

$$= \begin{bmatrix} C_1C_2C_3 - C_1S_2S_3 & S_1C_2C_3 - S_1S_2S_3 & -S_2C_3 - C_2S_3 \\ S_1 & -C_1 & 0 \\ -C_1C_2S_3 - C_1S_2C_3 & -S_1C_2S_3 - S_1S_2C_3 & S_2S_3 - C_2C_3 \end{bmatrix} \begin{bmatrix} n_x & s_x & a_x \\ n_y & s_y & a_y \\ n_z & s_z & a_z \end{bmatrix}$$

La matriz R es conocida, y por tanto, se resuelven $q4, q5\ y\ q6$ de las ecuaciones que nos quedan.

Hay **dos situaciones** que pueden darse cuando $q_5 = 0, q_5 = \pi$.

Cuando $\cos(q_5) \geq 1$, tenemos para la ecuación [7.74] que:

$$sq = atan2(R_3^6(2,1), R_3^6(1,1))$$

$$q_5 = 0$$

$$q_{5p} = 0$$

$$q_4 = sq/2$$

$$q_{4p} = sq/2 \qquad [7.76]$$

$$q_6 = sq/2$$

$$q_{6p} = sq/2$$

Cuando $\cos(q_5) < 1$:

$$sq = atan2(R_3^6(2,1), -R_3^6(1,1))$$

$$q_5 = \pi$$

$$q_{5p} = \pi$$

$$q_4 = -sq/2$$

$$q_{4p} = -sq/2 \qquad [7.77]$$

$$q_6 = sq/2$$

$$q_{6p} = sq/2$$

La situación normal (de la ecuación [7.73]):

$$q_5 = \text{acos}(C5)$$

Siempre en el rango $[0, \pi]$, ya que el seno de $q_5 > 0$ siempre

$q_{5p} = -q_5$, la otra solución.

$R_3^6(3,3)$, es distinto de 1, el seno de q_5 es distinto de 0

$$S5 = sen(q_5)$$

$S5 = \sqrt{1 - (C5)^2}$, que siempre es mayor que 0

$S5p = -S5$, la otra solución que es siempre menor que 0 [7.78]

$$q_4 = atan2(-R_3^6(2,3)/S5, -R_3^6(1,3)/S5)$$

$$q_{4p} = atan2(-R_3^6(2,3)/S5p, -R_3^6(1,3)/S5p)$$

$$q_6 = atan2(-R_3^6(3,2)/S5, -R_3^6(3,1)/S5)$$

$$q_{6p} = atan2(-R_3^6(3,2)/S5p, -R_3^6(3,1)/S5p)$$

Al final se obtienen dos soluciones para $\boldsymbol{q4, q5\ y\ q6}$ para cada caso. Nos quedamos con la primera de ellas (de la situación normal).

$$q = [q1, q2, q3, q4, q5, q6] \qquad [7.79]$$

La otra solución sería: $q = [q1, q2, q3, q4p, q5p, q6p]$ [7.80]

Donde $q1, q2$, $q3$ se indican en [7.60].

7.4.2 INCONVENIENTES PARA LA SOLUCIÓN DE LA CINEMÁTICA INVERSA

Aunque la solución del problema cinemático directo se obtiene directamente, la solución para el problema cinemático inverso depende estrictamente de la estructura del robot. Hay algunas dificultades que deben tenerse en cuenta a la hora de resolverlo.

La estructura de un manipulador de 6 ejes con muñeca de Euler permite desacoplar la cinemática de orientación y de posición. La característica geométrica que genera el desacoplamiento es la intersección de los últimos ejes.

Considerando nuestro caso, siempre que $q5$ sea distinto de 0 y de 180 grados, $q4$ y $q6$ se pueden resolver. Una singularidad del mecanismo existe cuando $q5$ es igual a 0, y cuando $q5$ es igual a 180 grados. En este caso, el manipulador pierde uno o más grados de libertad.

La solución de la cinemática inversa para un manipulador cuya estructura comprende articulaciones de revolución generalmente produce múltiples soluciones. Cada solución debe ser comprobada para determinar si lleva o no al elemento terminal a la posición deseada. Aunque hay varias soluciones, sólo una de ellas llevara al elemento terminal a la posición deseada, estando en la configuración deseada.

Puede que las soluciones matemáticas para el problema cinemático inverso no siempre se correspondan con las soluciones físicas.

CAPÍTULO 8. ENTORNO DE PROGRAMACIÓN

8.1 INTRODUCCIÓN

Robótica y mecatrónica representan en la actualidad áreas estratégicas y claves para todo país que aspire a la modernidad y bienestar, ya que su impacto no sólo repercute en aspectos políticos y económicos, también forma parte importante de la vida cotidiana, educación, cultura, y en todos los ámbitos de la sociedad.

La simulación es una herramienta imprescindible para reproducir los fenómenos físicos de un robot o de un sistema mecatrónico, permite estudiar y analizar al detalle los aspectos prácticos que intervienen en tareas específicas que debe realizar un robot industrial. La simulación es un proceso previo a la etapa experimental donde es posible entender los conceptos claves de la robótica y mecatrónica.

Hoy en día, MATLAB es un lenguaje de programación matemático de alto nivel integrado con entorno gráfico amigable, visualización de datos, funciones, graficas 2D y 3D, procesamiento de imágenes, vídeo, computación numérica para desarrollar algoritmos matemáticos con aplicaciones en ingeniería y ciencias exactas. Particularmente, en ingeniería es una herramienta muy poderosa para realizar aplicaciones en mecatrónica, robótica, control y automatización.

MATLAB es un acrónimo que proviene de matrix laboratory (laboratorio matricial) creado por el profesor y matemático Cleve Moler en 1970. Las versiones recientes del lenguaje MATLAB se caracterizan por ser multiplataforma.

Un rasgo distintivo de MATLAB es que ofrece un entorno gráfico de programación amigable al usuario a través de herramientas y utilerías para realizar simulación de sistemas dinámicos, análisis de datos, procesamiento de imágenes y video, gráficas y métodos de visualización, desarrollo de interfaces de usuario (GUI); también permite realizar interfaces para sistemas electrónicos, por ejemplo adquisición de datos con una versatilidad de tarjetas de instrumentación electrónica comerciales con plataforma de microprocesadores, DSP's, PIC's, FPGA's, manejo de puertos USB, COM, paralelo y diseños electrónicos propios.

8.2 ENTORNO DE PROGRAMACIÓN

Los programas utilizados para programar el robot han sido **MATLAB y Arduino**. Ambas son **plataformas de programación ampliamente conocidas**, especialmente en el mundo de la robótica y mecatrónica.

Ya se ha explicado en el capítulo 5 el software y hardware de Arduino que vamos a utilizar. Sin embargo, sólo vamos a utilizar Arduino en la última etapa de programación. Es decir, enviaremos la información desde MATLAB a la placa Arduino a través del puerto serie, y con la librería VarSpeedServo.h de Arduino vamos a enviar la información a los servomotores, que moverán las articulaciones del brazo robótico.

Las etapas anteriores vamos a realizarlas con MATLAB (aunque bien se podría realizar toda la programación en Arduino), por las razones que explican a continuación.

MATLAB es un lenguaje de programación mucho más fácil de depurar que el lenguaje de Arduino. Si el código del programa tiene errores, con MATLAB podemos identificarlos rápidamente y corregirlos, sin embargo con Arduino es muy difícil encontrar donde están los errores.

Necesitamos resolver la cinemática directa e inversa para calcular las posiciones que alcanzará el brazo robótico. Hemos elegido el modelo de Denavit-Hartenberg para estudiar la cinemática del robot, por ser ampliamente conocido y utilizado en el área de la robótica. El código programado en MATLAB nos permite resolver la cinemática directa e inversa introduciendo los parámetros necesarios de Denavit- Hartenberg y resolviendo la geometría específica del brazo robótico, así como resolviendo la relación existente entre variables articulares y posiciones de los servomotores. Algunas de las funciones que se emplean en el código desarrollado en MATLAB pueden encontrarse en MathWorks.

Conocido el giro de cada servo (limitado entre 0 y 180 grados) para que el giro de cada articulación sea el deseado, enviaremos las instrucciones a los servos con Arduino. El código de Arduino estará enfocado a enviar instrucciones a los actuadores eléctricos, pero controlaremos qué instrucciones se envían desde la interfaz gráfica de MATLAB, como se explica más adelante.

8.3 INTRODUCCIÓN AL LENGUAJE DE MATLAB

El ambiente de programación de MATLAB es amigable al usuario, y está compuesto por una interfaz gráfica con varias herramientas distribuidas en ventanas que permiten programar, revisar, analizar, registrar datos, utilizar funciones, historial de comandos y desarrollar diversas aplicaciones.

Generalmente, el usuario registra sus archivos en una carpeta o directorio predefinido. Para dar de alta dicho directorio al momento de simular aparecerá un mensaje donde debemos seleccionar *Add to Path* para que MATLAB realice la simulación desde esa trayectoria de trabajo, de esta forma las trayectorias de otros directorios que estén habilitadas no se perderá, por lo que la trayectoria del usuario se añade a las ya existentes.

El lenguaje de programación de MATLAB es de alto nivel y permite programar matrices, arreglos e incorporar instrucciones de control de flujo del programa, funciones, comandos y estructura de datos.

MATLAB funciona como calculadora, sobre el *prompt* de la ventana de comandos se pueden escribir expresiones aritméticas mediante los operadores “+”, “-”, ”/”, ”*”, ”^” para la suma, resta, división, multiplicación y potencia, respectivamente.

El lenguaje de programación de MATLAB está compuesto por un conjunto de reglas gramaticales y sintaxis para escribir correctamente las variables, constantes, operadores, expresiones, funciones y todos los elementos que forman parte de la programación.

Las variables, constantes, operadores y funciones forman una expresión la cual será procesada por un analizador léxico y sintáctico antes de ser ejecutada por la computadora. A diferencia de otros lenguajes, las expresiones en MATLAB involucran matrices.

Las variables no requieren ningún tipo de declaración o definición. Cuando MATLAB encuentra el nombre de una nueva variable, automáticamente crea la variable y le asigna una localidad de memoria. MATLAB distingue las letras mayúsculas de las minúsculas.

La notación convencional que usa MATLAB para la representación de números es la decimal con un punto decimal, signo ± y un número determinado de dígitos después del punto decimal, esto depende del tipo de formato numérico empleado.

8.3.1 OPERADORES EN MATLAB

Los operadores en MATLAB juegan un papel determinante ya que manipulan a las variables y funciones.

Operador colon :

El operador colon : (dos puntos verticales) es uno de los operadores más importantes para programar, se emplea para diferentes formas, como por ejemplo en MATLAB técnicamente se conoce como vectorización al proceso de generar una secuencia de numero usando 1:10 (en este caso incremento de uno en uno). Si se requiere un paso de incremento especifico, por ejemplo 2, se procede de la siguiente forma 1:2:10.

Operador ()

Los operadores paréntesis son utilizados en diversas operaciones matemáticas, por ejemplo evaluar funciones o referenciar elementos de matrices A(2,3). En expresiones aritméticas indican preferencia.

Operador semicolon ;

El operador punto y coma alineados en forma vertical o mejor conocido como semicolon ; tiene varias funciones.

Una de ellas se encuentra relacionada con el desplegar el resultado que tienen las variables, constantes, funciones o gráficas. Cuando se inserta al final de la expresión, instrucción o comando se inhabilita el desplegado. Si el operador ; no se coloca al final de la instrucción, entonces se produce el desplegado del valor de la variable.

Otra funcionalidad del operador ; es generar renglones en matrices (si está dentro de los corchetes). A = [19 3; 5 4]. Por cada ; se genera un renglón dentro de la matriz.

Operador ,

El operador coma , tiene mas de una función en MATLA. Por ejemplo cuando se emplea en funciones indica la separación de los argumentos. Para referenciar a los elementos de una matriz se especifica el numero de renglón y de la columna separados por una coma A(3,4). En matrices indica la separación de los elementos de un renglón. A = [5,6,4;8,8,3]

Operador ‘

El operador ‘ se relaciona con el manejo de datos tipo char o cadena de caracteres. También representa la traspuesta de una matriz.

Operador %

El lenguaje MATLAB permite insertar comentarios usando el operador % el cual deberá emplearse por cada línea de comentarios. Los comentarios son importantes en todo programa ya que permiten la documentación técnica del algoritmo o aplicación a implementar.

Operador ~

El operador tilde ~ se emplea para deshabilitar una variable de salida de una función. Es muy útil cuando la función retorna más de una variable y no se requiere usar todas las variables.

También se emplea como negación en operadores lógicos, por ejemplo ~= que significa no es igual a.

8.4 APP DESIGNER

App Designer es un **entorno de desarrollo** que proporciona vistas de diseño y código, una versión totalmente integrada del editor MATLAB y un gran conjunto de componentes interactivos.

Vamos a utilizar esta aplicación de MATLAB para diseñar y crear la interfaz gráfica desde la que vamos a enviar instrucciones a la placa Arduino.

Para acceder a AppDesigner sólo necesitamos ejecutar el comando *appdesigner* desde la ventana de comandos de MATLAB y automáticamente se abrirá.

App Designer proporciona un gran conjunto de componentes para diseñar aplicaciones modernas y completas.

Los **componentes que están disponibles**:

- **Componentes comunes**: ejes para crear gráficos y varios componentes que respondan a las interacciones, como diferentes tipos de botones, controles deslizantes, tablas, listas desplegables y árboles.
- **Contenedores y herramientas de figura**: incluyen paneles y pestañas para agrupar componentes, así como barras de menú.
- **Instrumentación**: indicadores y lámparas para visualizar el estado, así como mandos e interruptores para seleccionar los parámetros de entrada.

Para agregar un componente a la aplicación, lo arrastramos desde la lista de componentes. Podemos modificar sus características, como el color, la fuente o el texto.

8.4.1 CALLBACKS EN APPDESIGNER

Una vez hecho el diseño de la aplicación con toda la información que queremos mostrar y todos los componentes que vamos a utilizar (botones, interruptores, etc), programamos el código de los Callbacks de cada componente.

Una devolución de llamada (Callback) es una función que se ejecuta cuando interactuamos con un componente en la aplicación. La mayoría de los componentes pueden tener al menos una devolución de llamada. Sin embargo, algunos componentes, como etiquetas y lámparas, no tienen Callbacks porque solo muestran información. Hay varias formas de crear una devolución de llamada para un componente.

Por ejemplo, para un componente de un botón (Button), haciendo click con el botón derecho del ratón sobre el componente, en la pestaña Callback nos aparece la opción *Add ButtonPushedFcn callback.* Con ésta opción creamos una devolución de llamada en un componente que no tenía ningún callback creado. Al crear el callback, en la vista de código podemos escribir la función que queremos que se ejecute cuando interactuamos con ese componente.

Si un componente ya tiene asignado un callback que hemos creado previamente, nos aparece la opción de *Go to ButtonPushed callback*, y directamente vamos a la vista del código del callback donde podemos modificarlo.

Esta opción para crear/modificar o ver el código de los callback vale para cualquier componente y es la forma que hemos utilizado.

8.4.2 DISEÑO DE APPSEDIGNER PARA EL BRAZO ROBÓTICO

A continuación se muestra el diseño que hemos realizado en AppDesigner para interacturar con el brazo robótico, con todos sus componentes, y se explica el modo de funcionamiento.

En primer lugar, vamos a mover las articulaciones del robot (q1, q2, q3, q4, q5, q6) con 12 botones, dos para articulación (uno para aumentar los grados denominado *plus* y otro para disminuir los grados denominado *minus*).

El número de grados que vamos a aumentar o disminuir dependerá de si seleccionamos *'Low', 'Medium', 'High'* (15 grados, 30 o 45 respectivamente). Para todos los movimientos, podemos seleccionar una velocidad entre 1 y 10 (donde la velocidad 1 no es recomendada por ser demasiado baja). Lo descrito hasta ahora se muestra en la Figura 8.1.

Cada vez que el valor de una de las articulaciones cambie de un modo u otro, el robot se moverá y se actualizará automáticamente el valor de la posición del

punto final del robot (px, py, pz), en las celdas que se muestran en la Figura 8.2.

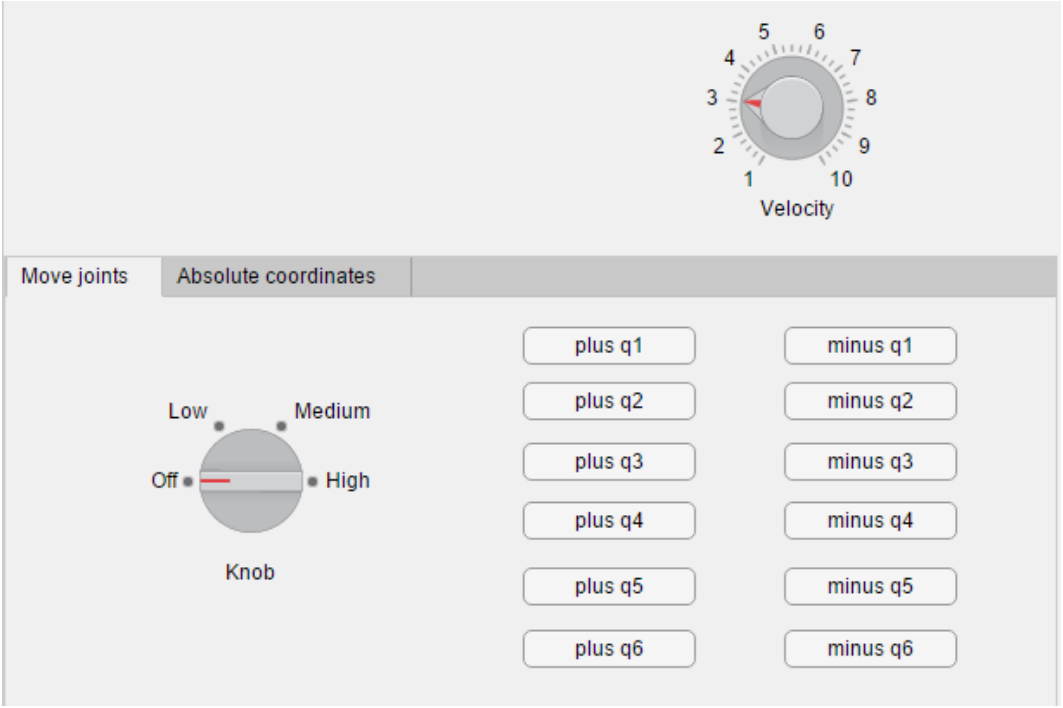


FIGURA 8.1. MOVER ARTICULACIONES EN APPDESIGNER

También tenemos la opción de escribir directamente el valor de cada articulación q (se han fijado unos límites para no sobrepasarlos), cuando habilitamos la edición de las celdas denominadas *"Position q1",* etc , como se muestra en la Figura 8.3.

Podemos modificar el valor de px, py, pz (coordenadas absolutas de la posición del punto final del robot) tecleando directamente el valor deseado para cada eje, o pulsando los botones correspondientes para aumentar o disminuir x, y, o z (*plus* y *minus*). En la Figura 8.2 se pueden ver las opciones para mover el robot en coordenadas absolutas.

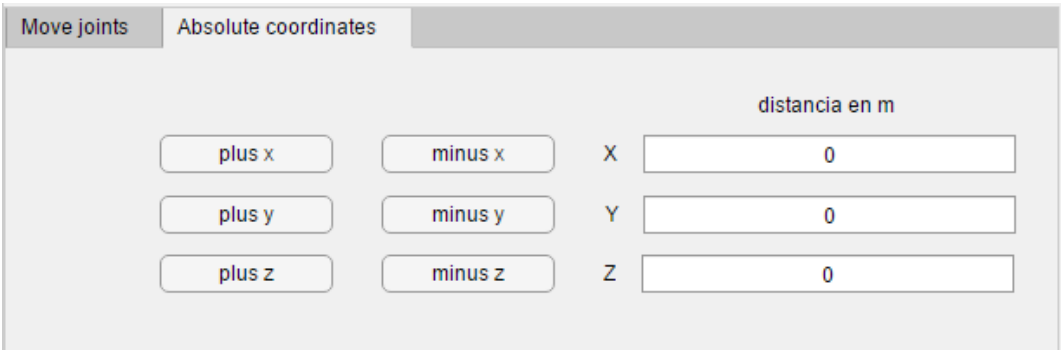


FIGURA 8.2. MOVER EN COORDENADAS ABSOLUTAS EN APPDESIGNER

Tenemos 2 botones para cada eje. Uno para aumentar la distancia y otro para disminuirla (la misma distancia siempre, que se fijará en un valor adecuado

que podemos cambiar en cualquier momento). Cada vez que cambie el valor de x, y, z, se actualizarán las posiciones de las coordenadas articulares q1, q2 y q3, y el robot se moverá. Es importante recalcar que al hacer un movimiento en coordenadas absolutas, la orientación de la pinza (de las últimas 3 articulares) no cambia, tal como lo hemos programado.

Siempre vamos a enviar a Arduino las posiciones de los servos (entre 0 y 180 grados). Las posiciones de las variables articulares q, que aparecen en las celdas de AppDesigner denominadas *"Position q1",* etc., se corresponden con los valores de las articulaciones y no de los servos.

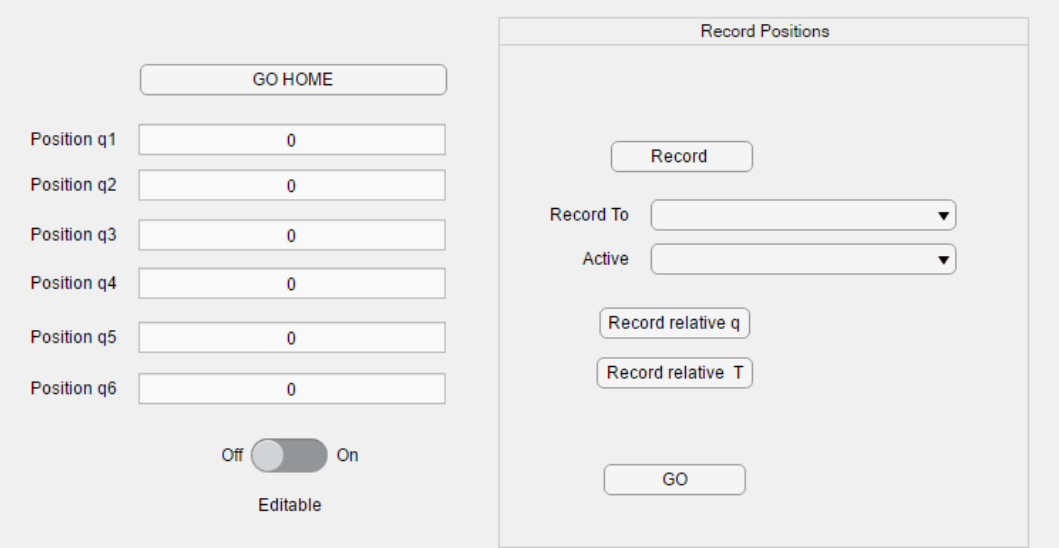


FIGURA 8.3. POSICIONES Y GRABACIÓN DE POSICIONES EN APPDESIGNER

El botón *GO HOME* lleva al robot a una posición predefinida, a la que podemos mover el robot en cualquier momento. Al iniciar el programa, el robot se mueve hasta esta posición.

Además, tenemos una opción para grabar posiciones: absolutas, y relativas. Las relativas pueden ser relativas a q (cuando movemos alguna articulación), y relativas a T, que designa a la matriz homogénea (cuando movemos el robot en x, y, z). Las posiciones relativas sólo se pueden grabar respecto a una posición absoluta, tal como lo hemos diseñado.

El botón *GO*, moverá el robot hasta la posición que esté seleccionada en la pestaña *Active*. El botón *Record*, sirve para grabar una nueva posición absoluta, y los botones *Record Relative q* y *Record Relative T*, sirven para grabar posiciones relativas respecto a la posicion absoluta seleccionada en la pestaña *Record To* (Figura 8.3).

CAPÍTULO 9. CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO

9.1 CONCLUSIONES

La robótica es un campo de muchas aplicaciones dentro de la tecnología moderna, cuyo desarrollo requiere conocimientos de diversas áreas de conocimiento como ingeniería eléctrica, ingeniería mecánica, ingeniería industrial y de sistemas, conocimientos de computación, economía y matemáticas. La configuración y el comportamiento de un robot condicionan su adecuación para un campo determinado de aplicaciones y viceversa. El estudio de la robótica, por tanto, es necesario en una gran diversidad de campos.

Nos referimos con el término de robot a un manipulador industrial controlado por un ordenador. Los robots son sistemas electromecánicos extremadamente complejos cuya descripción analítica requiere métodos avanzados, presentando muchos retos e interesantes problemas de búsqueda.

El movimiento de un robot está controlado por el controlador, que es un computador que gobierna, recoge y procesa la información generada por los sensores, y determina los movimientos. El desarrollo de los microprocesadores, el aumento de su potencia y la reducción de su precio han permitido el desarrollo de robots de última generación cada vez más potentes y versátiles.

Durante el desarrollo de este proyecto:

- Se ha diseñado el control cinemático directo e inverso de un robot serie antropomórfico de 6 grados de libertad, que permite mover sus articulaciones eje a eje, mover el robot en coordenadas absolutas, así como grabar posiciones absolutas y relativas y moverse de unas a otras. Se han utilizado actuadores eléctricos para mover las articulaciones del robot, controlados desde una placa con un microcontrolador.

- Primeramente, se han dimensionado todas las piezas que componen el robot, se han modelado en CATIA V5, y a continuación se ha procedido a su montaje. Por otra parte, se ha diseñado y construido una caja base a la que está sujeta la propia base del brazo robótico que le proporciona estabilidad en todos sus movimientos y cierta altura desde la mesa de trabajo.

- Se ha diseñado y realizado el circuito electrónico consistente en conexiones en una palca de tiras, mediante el cual se comunican los actuadores eléctricos con el microcontrolador. A su vez se ha programado la comunicación entre el microcontrolador y el computador que controla el robot. Se ha programado el código que resuelve la cinemática directa e inversa en un computador, y se ha diseñado una interfaz gráfica (GUI), permitiendo que el usuario pueda interactuar con el robot desde un computador, realizando movimientos articulación a articulación, en coordenadas absolutas y con la posibilidad de grabar posiciones y moverse

entre ellas, eligiendo una velocidad más rápida o más lenta para los movimientos.

No obstante, el brazo robótico presenta una serie de limitaciones:

- Una de las limitaciones es el tipo de actuadores eléctricos mediante el cual se mueven las articulaciones. Se trata de servomotores que tienen como sensor de posición un potenciómetro, y por tanto tienen una resolución baja. Es difícil que realicen movimientos de muy pocos grados, en especial para las primeras articulaciones que tienen que mover más peso.

- En la construcción física de las articulaciones existen pequeñas holguras difíciles de anular completamente.

- El robot es de pequeñas dimensiones, y aunque en nuestro caso no hemos incorporado ningún elemento terminal en el TCP (punto central de la herramienta) del robot, si se quisiera incluir tendría que ser también de pequeñas dimensiones y poco peso.

- El alcance del robot depende de la geometría del mismo y de la posición de montaje de cada servo. Los servos utilizados tienen un recorrido de 180 grados. Se podrían reescribir las librerías de Arduino para que se pudieran mover servos con un recorrido de 0 a 360 grados y así aumentar el alcance del brazo robótico. Otra posibilidad para modificar en cierta medida el alcance es conectar los mismos servos en diferentes posiciones a las bridas. Tal como hemos conectado los servos, el robot llega perfectamente a la mesa de trabajo y la altura que alcanza es suficiente.

- Una limitación importante es el difícil control de la velocidad de cada articulación, o el tiempo que tarda el robot en realizar un determinado movimiento. La librería utilizada en Arduino nos permite tener un cierto control de la velocidad, aunque no permite elegir una velocidad concreta. El tiempo que tarda el robot en realizar un movimiento depende directamente de cuantos grados tiene que moverse la articulación que va a recorrer un ángulo mayor, en el caso de movimientos donde se tenga que mover más de una articulación.

A pesar de estas pequeñas limitaciones, se ha conseguido construir y programar el brazo robótico, hacer un diseño eléctrico adecuado y establecer la comunicación entre el brazo robótico y un computador, por lo que la conclusión del trabajo es positiva e invita a continuar el estudio de este tipo de brazos robóticos controlados con el software y el hardware de Arduino desde un computador.

9.2 POSIBLES LÍNEAS DE TRABAJO FUTURO

A partir de este proyecto se establecen ciertas líneas de trabajo que pueden seguirse para profundizar en el concepto de robot antropomórfico de 6 grados de libertad y finalizar aquellos estudios que no se han cubierto en este trabajo.

En el presente trabajo se ha estudiado la cinemática directa e inversa del brazo robótico, de forma que el estudio de la cinemática y el estudio de la dinámica quedan pendientes para futuros trabajos.

Los actuadores eléctricos que se han elegido para este proyecto se podrían sustituir por otros con mejores características, aunque para proyectos de este tipo los servos utilizados son los más comunes. Asimismo se podría rediseñar el alcance del robot, cambiando el recorrido que tiene cada servo en la fase de montaje.

Las conexiones en la placa de tiras soldada pueden ampliarse y mejorarse incluyendo más elementos tal como condensadores, pulsadores, diodos (para una alimentación de los servomotores a 7V/6V), etc, para mejorar el control del robot. Se podría hacer un diseño con una placa PCB (placa de circuito impreso).

El robot tal como está construido tiene la posibilidad de que se incorpore en su extremo cualquier elemento terminal que se sujetará por medio de tornillos u otros elementos de unión, de forma que el robot pueda desempeñar alguna función en concreto.

Tanto en MATLAB como en Arduino se puede mejorar el diseño del código de programación. Las trayectorias que realiza el robot en cada movimiento son isócronas. Pueden programarse más adelante trayectorias circulares, lineales, o de cualquier otro tipo para ir de un punto a otro. Cuando el movimiento del robot es en coordenadas cartesianas, la orientación de la pinza (las 3 últimas articulaciones) no cambia, si no que se mantiene igual en cada movimiento. Programar su orientación sería un avance importante.

En cuanto a la interfaz gráfica de AppDesigner, se podría modificar en cualquier momento, ya que existen muchas opciones para diseñarla e incluir nuevas funciones. Además, un logro importante sería hacer un robot verdaderamente programable, de forma que se pudiera ejecutar un programa de control desde la interfaz gráfica (GUI) de MATLAB.

Por último, el dimensionamiento de las piezas permite que en un futuro puedan reproducirse o modificarse si es necesario, incluso para la construcción de otro brazo robótico a una escala diferente o de otro material.

BIBLIOGRAFÍA

LIBROS

ALIVERTI, PAOLO (2016). Manual de Arduino. Marcombo.

BARRIENTOS, A. (2007). Fundamentos de robótica. Segunda edición. Mc Graw Hill.

BLUM, JEREMY (2104). Arduino a fondo. Aprenda a crear fácilmente gadgets, gizmos, robots y mucho más. Anaya.

FÉLEZ, JESÚS; MARTÍNEZ, Mª LUISA (2008). Ingeniería gráfica y diseño. Editorial Síntesis S.A.

J. CRAIG, JOHN (2006). Robótica, 3º edición. Pearson Educación.

KUCUK, SERDAR; BINGUL, ZAFER (2006). Robot Kinematics: Forward and Inverse Kinematics, Industrial Robotics: Theory, Modelling and Control, Sam Cubero (Ed.), InTech,

Disponible en:
http://www.intechopen.com/books/industrial_robotics_theory_modelling_and_control/robot_kinematics__forward_and_inverse_kinematics

KUMAR SAHA, SUBIR (2008). Introducción a la Robótica. Indian Institute of Technology. Mc Graw Hill.

LÓPEZ PELÁEZ, ANTONIO (2003). Nuevas Tecnologías y sociedad actual: El Impacto de la Robótica. Ministerio de Trabajo y Asuntos Sociales. Instituto nacional de seguridad e higiene en el trabajo.

MARTÍNEZ, LUIS JAVIER (2016). Cómo buscar y usar información científica: Guía para estudiantes universitarios 2016.

REYES CORTÉS, FERNANDO (2012). Matlab Aplicado a Robótica y Mecatrónica. Alfaomega.

W. SPONG, MARK; HUTCHINSON, SETH; VIDYASAGAR, M. (2006). Robot Modeling and Control. Wiley.

PÁGINAS WEB Y OTROS DOCUMENTOS

Normas UNE sobre Dibujo Técnico: UNE 1032-82 ISO 128, UNE 1039-94 ISO 129.

Información sobre toda la plataforma de software y hardware de acceso libre de Arduino. Consultado por última vez en Junio de 2018:
https://www.arduino.cc/

Datos y especificaciones técnicas del brazo robótico de 6 grados de libertad. Consultado por última vez en Junio de 2018: http://micropede.de/shop/mp-robot-d

Programa de acceso libre de diseño electrónico. Consultado por última vez en Junio de 2018: http://fritzing.org/home/

Información y guía para el entorno de programación de AppDesigner (MATLAB). Consultado por última vez en Junio de 2018: https://www.mathworks.com/products/matlab/app-designer.html

Datos técnicos sobre los servomotores utilizados. Consultado por última vez en Junio de 2018: http://www.towerpro.com.tw/

Datos técnicos sobre los servomotores utilizados. Consultado por última vez en Junio de 2018: http://www.electronicoscaldas.com/85-motores-y-servos

Librería de acceso libre VarSpeedServo.h, utilizada en Arduino. Consultado por última vez en Junio de 2018: https://github.com/netlabtoolkit/VarSpeedServo/blob/master/VarSpeedServo.h

Caja de herramientas que incluye las funciones para modelar movimientos rígidos tridimensionales (funciones empleadas en la programación de MATLAB). Autor: Antonio Tristán Vega. Consultado por última vez en Junio de 2018: https://it.mathworks.com/matlabcentral/fileexchange/56758-rigid-motions-and-robotics-toolbox