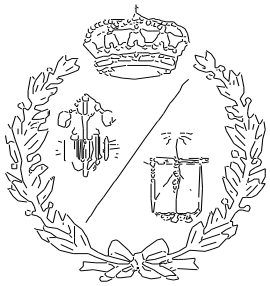
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

DISEÑO Y CONFIGURACIÓN DE UN ROBOT SCARA

(Design and configuration of a scara robot)

Para acceder al Título de

GRADUADA EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Lidia Torre San Emeterio

Septiembre – 2022

Índice general:

DOCUMENTO 1
MEMORIA

RESUMEN

El proyecto que se presenta consiste en la construcción y programación de un Robot Serie SCARA de 4 grados de libertad configurable, de forma que sus eslabones tengan varias posiciones de longitud. Se desarrollarán los modelos de las piezas mediante modelos CAD para su posterior fabricación mediante impresión 3D, utilizando servomotores de bajo coste para su movimiento y desarrollando la programación del microcontrolador mediante Arduino.

Previo a la adquisición del brazo robótico, se introducirán los principios teóricos necesarios para comprender los conceptos básicos que sustentan este trabajo. Luego, el trabajo procede en cuatro etapas hasta lograr el objetivo deseado. El primero consiste en la selección de los diferentes componentes que constituyen el brazo robótico. El segundo implica el diseño de las partes que componen el robot, y esto se realiza por computadora utilizando un software CAD, así como su fabricación y montaje. En la tercera etapa se realiza el estudio de la cinemática del robot necesaria para la transición a la cuarta y última etapa, a partir del sistema de control del brazo, gracias al cual el brazo podrá realizar las tareas de manera útil. Finalmente, se tratará sobre el lenguaje utilizado para la programación, así como el algoritmo desarrollado para realizar su funcionamiento.

Cabe señalar que el objetivo del presente trabajo no es obtener un manipulador con intereses industriales, sino el de un uso didáctico proporcionando un primer contacto en la rama de la robótica para estudiantes jóvenes, facilitando así la adquisición de conocimiento.

ABSTRACT

The project presented consists of the construction and programming of a SCARA Series Robot with 4 degrees of freedom that can be configured so that its links have various length positions. The models of the parts will be developed using CAD models for their subsequent manufacture by 3D printing, using low-cost servomotors for their movement and developing the programming of the microcontroller using Arduino.

Prior to the acquisition of the robotic arm, the theoretical principles necessary to understand the basic concepts underpinning this work will be introduced. Then, the work proceeds in four stages until the desired objective is achieved. The first involves the selection of the different components that make up the robotic arm. The second involves the design of the parts that make up the robot, and this is done by computer using CAD software, as well as its manufacture and assembly. The third stage involves the study of the kinematics of the robot necessary for the transition to the fourth and final stage, starting with the control system of the arm, thanks to which the arm will be able to perform the tasks in a useful way. Finally, the language used for programming will be discussed, as well as the algorithm developed to carry out its operation.

It should be noted that the aim of the present work is not to obtain a manipulator with industrial interests, but a didactic use providing a first contact in the field of robotics for young students, thus facilitating the acquisition of knowledge.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCION

1.1. MOTIVACIÓN

El trabajo desarrollado durante este proyecto se encuentra implícito dentro del campo de la robótica. El procesamiento robótico es el más utilizado en las industrias, convirtiéndose en una de las áreas más importantes para comprender. Por ello, es importante que los alumnos aprendan a programar y controlar este tipo de robots. Esta acerca a los estudiantes a la exploración de la tecnología y la programación, creando una forma de aprendizaje dinámica y lúdica en la que se involucran diferentes campos de estudio en el proceso: matemáticas, ciencias y programación. Esta razón, que es muy importante para el desarrollo futuro de los ingenieros en robótica, es la fuerza impulsora detrás de la creación de este proyecto que les ayudará a aprender a controlar un robot manipulador.

1.2. OBJETIVOS

A través del proyecto, el objetivo es utilizar las técnicas aprendidas a lo largo de la carrera e implementar estas en un robot real que además sea económico. Con este enfoque, nos aseguramos de que cada estudiante pueda tener un robot con el que aprender. Por lo que su objetivo principal será de uso didáctico, de forma que se proporcione un primer contacto en la rama de la robótica para los estudiantes jóvenes, facilitando así la adquisición de conocimiento.

Se comenzará encontrando materiales baratos que se puedan utilizar para construir robots. A continuación, se calculará su modelo cinemático para controlarlo más tarde y crear un generador de trayectorias. Toda la parte de control se hará primero por simulación y luego en el robot. Con esto también podemos comprobar si el robot se mueve como debería y que todos los cálculos son correctos.

2. CONCEPTOS BÁSICOS

En este apartado realizaremos una introducción de los pasos que vamos a seguir a la hora de diseñar y controlar el robot.

2.1. ROBÓTICA

La robótica es una rama de la ingeniería que se ocupa del diseño, la construcción, el funcionamiento, la estructura, la fabricación y la aplicación de robots.

La Real Academia Española define esta como "Técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales".

Robot viene de la palabra checa robota, que significa trabajo forzado o monótono. Esta palabra se refiere a un manipulador multifuncional automatizado que funciona por energía, para realizar una variedad de tareas.

2.1.1. HISTORIA

El origen de la robótica radica en la necesidad de encontrar soluciones alternativas que faciliten y optimicen cada tarea productiva, cronológicamente tiene su origen desde Aristóteles y sus ideas sobre 'herramientas automatizadas' hasta Henry Ford. A continuación, los hitos referenciados marcan el avance real en el campo de la automatización y la robótica a principios del siglo XX.

20's: Karel Capek utilizo por primera vez la palabra robota en la obra 'RUR (Rossum's Universal Robots)' para nombrar a un humanoide mecánico. La cual cosecho gran éxito y se exporto rápidamente a casi todos los idiomas del mundo.

40's: Isaac Asimov publicó en la revista *Astounding Science Fiction* las tres leyes de la Robótica, que establecen cómo los robots deben someterse a la voluntad humana. Además, William Grey Walter exhibió un robot con comportamiento biológico simple.

50's: G. C. Devol desarrolló un dispositivo capaz de registrar señales eléctricas por medios magnéticos y de reproducirlas para mover un simple brazo mecánico con cierta flexibilidad. Joseph Engelberger y George Devol crearon el primer robot comercial de la compañía Unimation, el cual estaba basado en una patente de Devol. Se fundó el Laboratorio de Inteligencia Artificial del MIT. El MIT fue la piedra angular de la robótica universitaria del siglo XX, solo igualado por el Instituto de Robótica de la Universidad Carnegie Mellon.

60's: Se instala el primer robot industrial. Y se aplica a la robótica el primer sistema de visión por ordenador.

70's: La universidad de Stanford desarrolló el llamado "brazo de Stanford", con accionamientos eléctricos. Primer robot con seis ejes electromecánicos. Brazo manipulador programable universal, PUMA, para operaciones de ensamblaje, basada en diseños realizados por un estudio de ingeniería de General Motors. El primer robot SCARA se desarrolló en la Universidad de Yamanashi (Japón) para operaciones de montaje.

80's: Primer robot de tracción directa, en el cual se emplearon motores eléctricos en las articulaciones sin las transmisiones mecánicas habituales utilizadas por la mayoría de los robots. Se presentó el robot de ensamblaje RS-1, un robot estructural en forma de caja que utiliza un brazo que consta de tres motores ortogonales. Demostración de varios sistemas de programación off-line, los cuales permitían desarrollar el programa del robot mediante gráficos interactivos en un ordenador personal y luego descargarlo en el robot. La empresa japonesa HONDA inicia un proyecto para construir un robot humanoide.

90's: HONDA anuncia el primer robot humanoide P2 que puede caminar de forma autónoma. SONY lanza "Aibo" un perro-robot.

S. XXI: SONY presenta el robot humanoide capaz de desplazarse de forma bípeda e interactuar con las personas. Hanson Robotics Co. Presenta un robot humanoide ginoide capaz de reconocer, recordar caras y simular expresiones.

2.1.2. CLASIFICACION SEGÚN SU CRONOLOGIA Y ESTRUCTURA

Los robots se pueden clasificar de dos maneras. La primera forma de categorizarlos es a través de la cronología:

- 1.ª Generación: Robots manipuladores. Son sistemas mecánicos multifuncionales con sistemas de control de secuencia simple, manual, fija o variable.
- 2.ª Generación: Robots de aprendizaje. Repiten la secuencia de movimientos realizados previamente por el operador. Esto se realiza a través de un dispositivo mecánico. El operador realiza los movimientos necesarios y el robot los rastreará y memorizará.
- 3.ª Generación: Robots con control sensorizado. Un controlador es una computadora que ejecuta comandos de programa y los envía a un operador o robot para realizar los movimientos necesarios.

La segunda manera es a través de su estructura:

La estructura de un robot viene definida por la configuración del mismo, esta puede llegar a ser metamórfica. El concepto metamórfico se ha introducido recientemente para aumentar la flexibilidad funcional del robot cambiando su propia configuración. El metamorfismo permite diferentes niveles de ejecución, desde los más básicos (cambio de herramientas o impacto final) hasta los más complejos, como la alteración o modificación de elementos estructurales individuales o sistémicos. La división de robots, en base a su arquitectura, se divide en los siguientes grupos:

- Poliarticulados: Este grupo cubre robots de muchas formas y configuraciones, cuya característica común es que en su mayoría son sedentarios (aunque en casos especiales pueden ser controlados para movilidad limitada) y tienen una posición definida para que sus elementos finitos se muevan de una manera particular en un espacio de trabajo según uno o más sistemas de coordenadas y un número limitado de grados de libertad. Se clasifican en este grupo los manipuladores, robots industriales y robots cartesianos, que se utilizan cuando es necesario cubrir un área de trabajo relativamente grande o extensa, actuando sobre objetos con un plano de simetría vertical o superficie reducida.

Ilustración 1. Robot poliarticulado

- Móviles: Son robots de servicio pesado, basados en carros o plataformas y equipados con sistemas locomotores rodantes. Siguen su camino mediante un mando a distancia o se guían por la información que reciben sus sensores del entorno. Estos robots transportan piezas de un punto a otro de la línea de producción. Al rastrear las pistas materializadas a través de la radiación electromagnética de los circuitos eléctricos incrustados en la tierra o por bandas detectadas por la energía fotovoltaica, pueden incluso evitar obstáculos y estar dotados de un grado de inteligencia relativamente alto.

Ilustración 2. Robot móvil

- Androides: Estos son tipos de robots que intentan reproducir total o parcialmente la forma y el comportamiento humanos cinéticos. Actualmente, los androides siguen siendo dispositivos que han tenido poco desarrollo y no tienen un uso práctico, principalmente para aprender y probar. Uno de los aspectos más complejos de estos robots, donde se concentra la mayor parte del trabajo, es el movimiento bípedo. En este caso, la cuestión clave es la gestión dinámica y coordinada de los procesos en tiempo real manteniendo el equilibrio del Robot. A menudo se les llama "títeres" cuando ve cables que le permiten verlos realizar sus procesos.

Ilustración 3. Androide

- Zoomórficos: Los robots zoomórficos, que incluyen, pero no se limitan a los androides, forman una clase caracterizada principalmente por sistemas de locomotoras que simulan varias criaturas vivientes. A pesar de las diferencias morfológicas entre sus posibles sistemas locomotores, es conveniente dividir a los robots zoomórficos en dos categorías principales: caminantes y no caminantes. El grupo de robots zoomórficos que no caminan está muy poco

desarrollado. Los experimentos realizados en Japón se basan en segmentos cilíndricos oblicuos que se unen entre sí y producen una rotación relativa. En varios laboratorios se están llevando a cabo numerosos experimentos y robots con zoomórficos para caminar de múltiples pedales para desarrollar vehículos terrestres reales, tripulados o autónomos capaces de desarrollarse en superficies muy irregulares. El uso de estos robots será de interés en los campos de la exploración espacial y la vulcanología.

Ilustración 4. Robot zoomórfico caminante

- Híbridos: Estos robots corresponden a robots de difícil clasificación, cuyo diseño se combina con cualquiera de los robots mencionados anteriormente, ya sea por conjunción o por yuxtaposición. Por ejemplo, un dispositivo con ruedas articuladas segmentadas es una de las propiedades de un robot móvil y un robot zoom al mismo tiempo.

Ilustración 5. Robot híbrido

2.1.3. CONFIGURACIONES MORFOLÓGICAS

El robot consta de una serie de elementos o eslabones conectados mediante articulaciones para proporcionar un movimiento relativo entre cada uno de los dos eslabones consecutivos. La

estructura física de la mayoría de los robots industriales está inspirada en la mano humana y la anatomía del brazo.

Los grados de libertad se definen como los posibles movimientos fundamentales independientes (movimiento de rotación y traslación). En la ilustración 6, se muestran los diferentes tipos de conexiones y los grados de libertad que proporciona cada una.

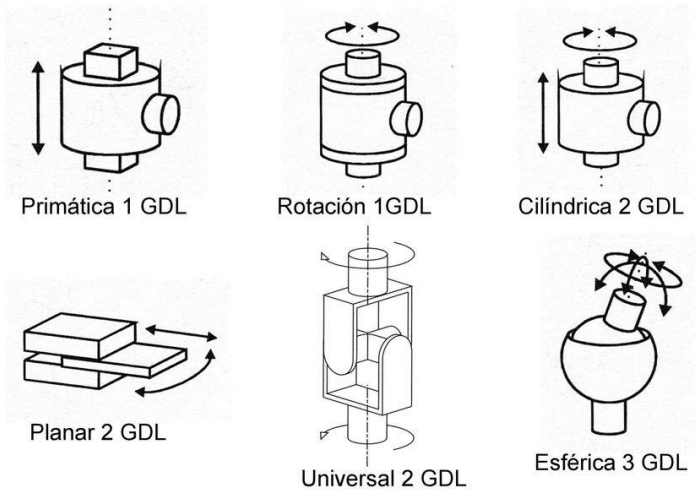


Ilustración 6. Tipos de articulaciones y sus grados de libertad

Diferentes combinaciones de estas conexiones dan lugar a diferentes configuraciones o arquitecturas mecánicas, cada una con características que la hacen más o menos recomendable para una tarea en particular. En la ilustración 7 se muestran las configuraciones más comunes utilizadas en robots industriales y destacan las conexiones y los grados de libertad de cada uno.

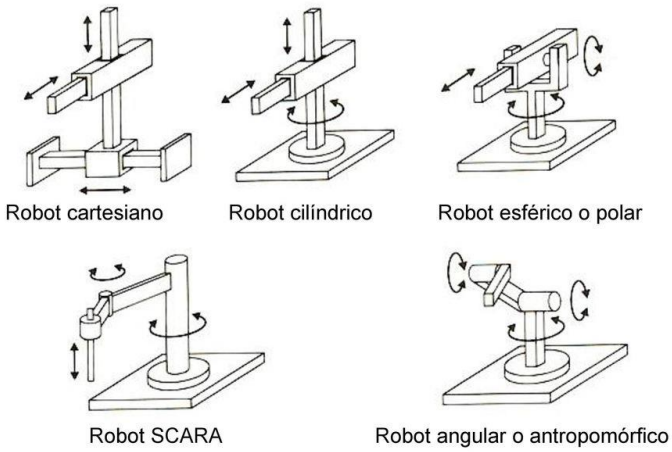


Ilustración 7. Diferentes arquitecturas mecánicas

Otro concepto importante a tener en cuenta al describir un robot es el tamaño del robot y el espacio de trabajo, que determina el movimiento de las diversas articulaciones. El espacio de trabajo del robot se define como el conjunto de puntos a los que puede llegar su efector final. El conocimiento preciso de la forma, el tamaño y la estructura del espacio de trabajo es esencial, ya que la forma es importante para determinar el entorno en el que operará el robot. Las dimensiones son importantes para determinar la masa de impacto final, y la estructura del espacio de trabajo es importante para garantizar las características cinéticas del robot en relación con la interacción entre el robot y su entorno.

Además, la forma, el tamaño y el diseño del espacio de trabajo dependen de las propiedades del robot en cuestión:

- Tamaño de la celda del robot y limitaciones mecánicas de las articulaciones.
- La forma depende de la geometría del robot (interferencia entre enlaces) y de las propiedades de los grados de libertad.
- La estructura del espacio de trabajo está determinada por la estructura del robot y el tamaño de sus enlaces.

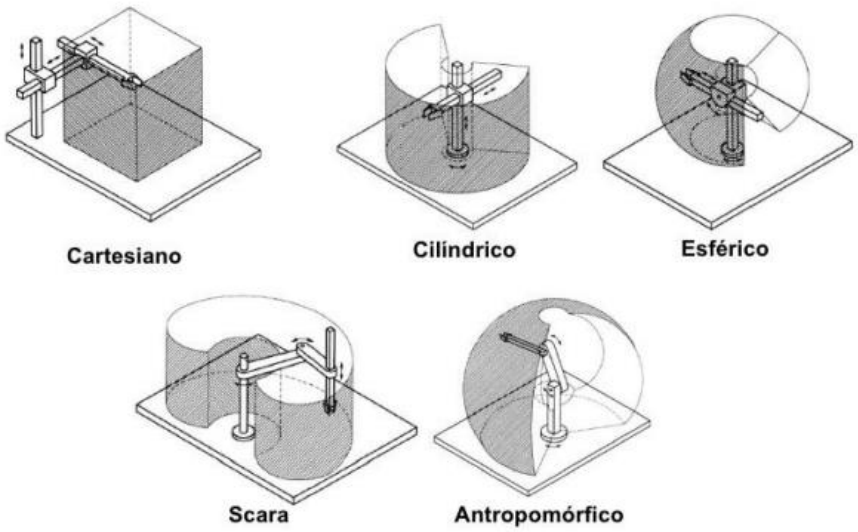


Ilustración 8. Posibles espacios de trabajo

2.1.4. ELEMENTOS DE UN ROBOT INDUSTRIAL

- **Manipulador:** conjunto de elementos mecánicos que aseguran el movimiento del efector final. Los actuadores, engranajes y elementos motrices que soportan el movimiento de las cuatro partes que normalmente componen el actuador se encuentran comúnmente en la construcción interna del actuador, los cuales son: base, cuerpo, brazo y antebrazo.

  Estos cuatro elementos rígidos del manipulador forman eslabones que están conectados entre sí por bisagras (articulaciones) que permiten el movimiento relativo de los eslabones adyacentes. Estas conexiones pueden girar (o ser de revolución) cuando el movimiento permitido es la rotación; en este caso, su desplazamiento se denomina ángulo de articulación o articulación deslizante (o prismática), donde el movimiento relativo entre los eslabones es un movimiento de traslación, a veces denominado compensación de articulación. El número de grados de libertad que tiene el operador es el número de variables independientes de posición que se deben especificar para posicionar todas las partes del mecanismo.
  Los grados de libertad del robot operativo están determinados por el número de enlaces y bisagras que conectan estos enlaces.

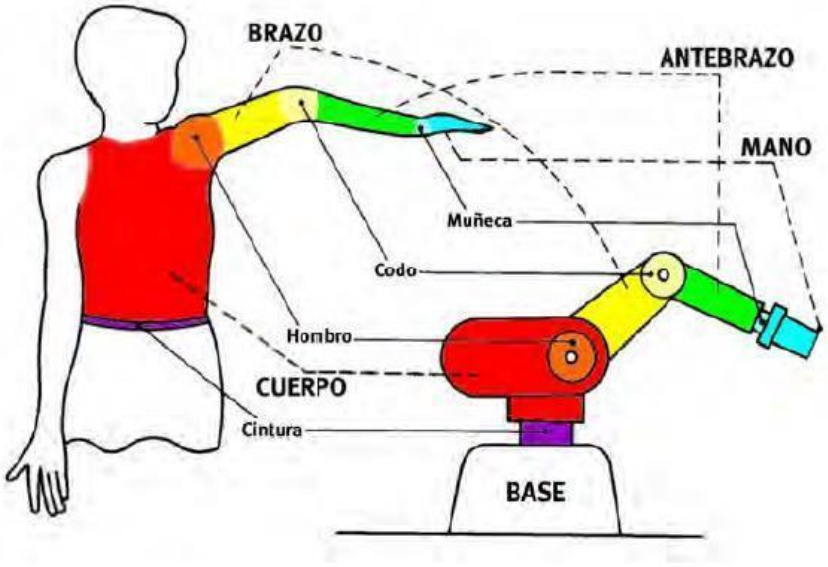


Ilustración 9. Estructura de un manipulador

- **Actuadores:** Son los encargados de producir el movimiento de las articulaciones directamente o a través de cadenas, poleas, cables, etc. Se dividen en tres grandes grupos:

  Neumáticos: utilizan aire comprimido como fuente de energía y pueden controlar movimientos rápidos, pero su precisión es limitada.

  Hidráulicos: los actuadores hidráulicos se recomiendan para operadores de alto rendimiento con control de velocidad preciso.

  Eléctrico: Los motores eléctricos son los más utilizados debido a su precisión y facilidad de control.

Ilustración 10. Actuadores (elementos finales)

- **Controlador:** Este dispositivo se encarga de regular el movimiento de todos los elementos del operador, realizar cálculos y procesar información. La complejidad del control varía según los parámetros que se ajusten.

Ilustración 11. Controlador

- **Efector final:** Garra o herramienta que se sujeta a la muñeca del manipulador y se encarga de realizar el trabajo planificado; por ejemplo, pueden ser pinzas, electroimanes o algún otro dispositivo. En general, y según el tipo de aplicación, el problema de los equipos de actuador final se debe a que estos deben tener una gran capacidad de carga, a la vez que es

importante que reduzcan el peso y las dimensiones. Por esta razón, a menudo es necesario diseñar el efecto final de acuerdo con los requisitos de la aplicación en la que se utilizará.

Ilustración 12. Efector Terminal

- **Sensores de información:** Los robots inteligentes son aquellos que pueden adaptarse a su entorno y tomar decisiones en tiempo real, para una variedad de situaciones. La información que reciben les da la capacidad de programarse a sí mismos, es decir, cambian su comportamiento de acuerdo con las circunstancias externas, lo que les otorga un grado de inteligencia artificial. En este sentido, la información más necesaria para los robots es la posición, la velocidad, la aceleración, la fuerza, el par, el tamaño y el perfil de los objetos y la temperatura.

2.1.5. ROBOT SCARA

El acrónimo de SCARA es "Selective Compilant Assembly Robot Arm", que significa "Brazo robótico de montaje selectivamente adaptable" en español. Dispone normalmente de 3 o 4 grados de libertad con posicionamiento horizontal.

Estos robots son conocidos por sus rápidos ciclos de trabajo, excelente repetibilidad, alta capacidad de carga y una amplia gama de aplicaciones. Una de sus aplicaciones típicas es recoger y dejar objetos desde el punto A hasta el punto B, lo que se denomina "pick and place".

El brazo robótico de SCARA está diseñado como un brazo humano. El primer eje (eje X) se mueve de forma horizontal. El segundo eje (eje Y), superpuesto al primer eje, nos permite posicionarnos en dos dimensiones. El tercer eje (eje Z) consiste en un eje con un componente vertical adicional. El cuarto eje, en caso de existir, es la rotación del propio husillo, que ayuda a dirigir el producto en la posición deseada en el plano.

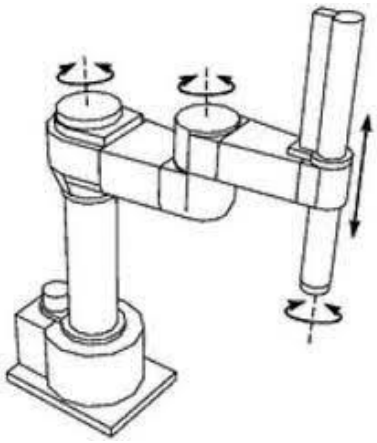


Ilustración 13. Movimientos de las articulaciones del robot SCARA

2.2. CINEMÁTICA

La cinemática de un robot es el estudio del movimiento de los robots. En el análisis cinemático, la posición, la velocidad y la aceleración de cada elemento del robot se calculan sin tener en cuenta las fuerzas que provocan el movimiento. Existen dos formas de conseguir la cinemática de un robot: la cinemática inversa y directa.

La cinemática directa, es el problema geométrico estático de calcular la posición y orientación del efector final de un manipulador. En cambio, la cinemática inversa, es el método el cual mediante las ecuaciones cinemáticas del robot determinará los parámetros comunes que proporcionan la posición deseada del efector final.

2.3.1. Cinemática directa

La cinemática directa es una técnica para calcular la posición apropiada de partes de una estructura a partir de sus miembros fijos y las diferencias causadas por las juntas de una estructura, en relación con una referencia. Tomado como referencia un sistema de coordenadas base, se conocen los valores correspondientes y los parámetros geométricos del robot.

En una cadena cinemática, la solución siempre es única: dado un conjunto de vectores, siempre corresponden a una posición del efector. En general, un robot tiene n grados de libertad que consisten en n enlaces conectados por n articulaciones, de modo que cada par de enlaces forma un grado de libertad. Cada enlace puede vincularse a un marco de referencia integrado y, mediante transformaciones homogéneas, puede representar rotaciones y traslaciones relativas entre los diferentes enlaces que componen el robot.

Existen diferentes métodos para el análisis de la cinemática directa:

- Transformación de matrices.
- Geometría.
- Denavit-Hartenberg.

En nuestro caso utilizaremos la parametrización de Denavit-Hartenberg. Este sigue una serie de pasos con los cuales determinaremos la cinemática directa de nuestro robot.

A continuación, se muestra dicho procedimiento, en el apartado 5.1 obtendremos dicha resolución para nuestro robot.

1. Numerar los eslabones: se denominará “0” a la base fija donde se ancla el robot, “1” al primer eslabón móvil y así sucesivamente.
2. Numerar las articulaciones: Siendo “1” el primer grado de libertad y “n” el ultimo.
3. Localizar el eje de cada articulación: Para pares de revolución, este será el eje de rotación. Para prismáticos, este será el eje sobre el que se moverá el enlace.
4. Eje Z: Comenzamos a instalar los sistemas XYZ. Colocamos los Zi-1 en los ejes de la articulación i, con i = 1, ..., n. es decir, Z0 va sobre el eje de la primera articulación, Z1 va sobre el eje del segundo grado de libertad, y así sucesivamente.
5. Sistema de coordenadas 0: Origen en cualquier punto a lo largo de Z0. La dirección de X0 e Y0 puede ser arbitraria, siempre que quede claro que XYZ es un sistema en el sentido de las agujas del reloj.
6. Resto de sistemas: Para el resto de los sistemas i = 1, ..., N-1, coloque el origen en la intersección de Zi con el punto común (intersección) Zi y Zi + 1. En el caso de que Zi-1 y Zi sean paralelos este origen se situará en la misma unión, por el contrario, si estos no son coplanares, lo situaremos en la línea de mínima distancia.
7. Ejes X: En el caso de que Zi y Zi-1 se crucen, situaremos el eje Xi en la línea perpendicular al plano formado por ellos, si estos son paralelos, el eje Xi debe situarse perpendicular a ambos y preferiblemente pasar por Oi-1, en cambio, si los ejes Zi y Zi-1 no son coplanares, Xi debe alinearse con la perpendicular común.
8. Ejes Y: Una vez situados los ejes Z y X, los ejes Y tienen su dirección determinada por la regla de la mano derecha.

9. Sistema del extremo del robot: el n-ésimo marco de coordenadas se coloca al final del robot (herramienta), con el eje Z paralelo a Zn-1, X e Y en cualquier dirección válida.
10. Ángulos teta: El ángulo de la articulación del eje x_{i-1} con el eje xi, medido con respecto al eje z_{i-1}, utilizando la regla de la derecha.
11. Distancia d: Distancia medida desde el origen del sistema i-1, a lo largo del eje z_{i-1}, hasta la intersección del eje z_{i-1} con el eje x_i.
12. Distancia a: Distancia de separación entre los orígenes de los sistemas de referencia i-1 e i, medida a lo largo del eje x_i hasta la intersección con el eje z_{i-1} (o la distancia más corta entre los ejes z_{i-1} y z_i, cuando no se cruzan).
13. Ángulos alfa: Ángulo que separa los ejes z_i y z_{i-1}, medido a lo largo del eje x_i.
14. Matrices individuales: Cada eslabón definirá una matriz de transformación.

$$ {}^{i-1}A_i = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \cdot \sin\theta_i & \sin\alpha_i \cdot \sin\theta_i & a_i \cdot \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cdot \cos\theta_i & -\sin\alpha_i \cdot \cos\theta_i & a_i \cdot \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

15. Matriz de transformación total: Vincula la base del robot con su herramienta, es decir, es la concatenación (multiplicación) de todas las matrices individuales:

$$ T = {}^{0}A_n = {}^{0}A_1 \cdot {}^{1}A_2 \dots {}^{n-1}A_n $$

La matriz T muestra que es posible resolver completamente el problema de cinemática directamente en el robot de control, porque al dar valores específicos para todos los grados de libertad de cada robot, obtenemos la posición y orientación 3D de la herramienta en el extremo de la cadena de articulaciones.

2.3.2. Cinemática inversa

El problema de la cinemática inversa consiste en encontrar los valores que deben adoptar las coordenadas de las articulaciones del robot $\theta_1, \theta_2, \dots, \theta_n$ para que su extremo se posicione y oriente según una ubicación espacial determinada. A diferencia del problema de cinemática directa, el cálculo de la cinemática inversa no es sencillo ya que consiste en resolver una serie de ecuaciones (generalmente no lineales) que dependen fuertemente de la configuración del robot, y además existen diferentes soluciones que resuelven el problema. Estas soluciones pueden depender de la colocación que adquieran las articulaciones, así como se puede observar en la ilustración 14.

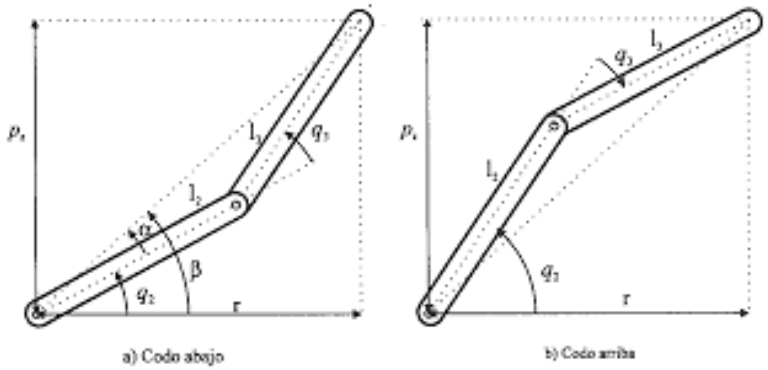


Ilustración 14. Posibles configuraciones en el manipulador

Se puede apreciar como en la articulación 2 existen dos posibles soluciones. Es importante tener esto en cuenta a la hora de calcular la cinemática inversa para determinar qué solución se quiere que tome el robot.

Existen 3 métodos principales para conseguirlo:

- Método geométrico: consiste en el uso de razones trigonométricas y la resolución de los triángulos formados a partir de los elementos y articulaciones del robot. Este procedimiento es adecuado para robots con pocos grados de libertad o para el caso en que, por simplificaciones, sólo se consideren los primeros grados de libertad.
- Resolución a partir de matrices de transformación homogéneas: El objetivo es encontrar el modelo cinemático inverso de un robot a partir del conocimiento de su modelo directo. Es decir, conociendo las relaciones que expresan el valor de la posición y orientación del extremo del robot en función de sus coordenadas articulares, se pueden obtener las relaciones inversas manipulándolas. En la práctica, esta tarea no es trivial y suele ser tan compleja que hay que descartarla.
- Desacoplamiento cinemático: este método permite, para ciertos tipos de robots, resolver los primeros grados de libertad, dedicados al posicionamiento, independientemente de la resolución de los últimos grados de libertad, dedicados a la orientación. Este método es típico para los robots con 6 grados de libertad, donde los tres primeros son para el posicionamiento del brazo y los tres últimos para la orientación de la herramienta.

2.3.3. Control cinemático

El control cinemático tiene como objetivo definir los caminos que cada articulación del robot debe seguir a lo largo del tiempo para lograr los objetivos establecidos por el usuario (Punto de destino,

tiempo invertido, tipo de trayectoria del extremo, etc). En este se debe prestar atención a las limitaciones físicas del manipulador y los estándares de calidad.

Consiste en describir el movimiento deseado del manipulador como una serie de puntos en el espacio (con posición y dirección). El control interpola la trayectoria deseada utilizando una clase de funciones polinómicas y genera una serie de puntos a lo largo del tiempo.

Para poder controlar el movimiento de un robot, es necesario generar previamente una trayectoria en el espacio para que el robot la realice. Estas pueden definirse como:

- Generación de la trayectoria en el espacio articular: Desplazamiento del brazo de un punto a otro en un tiempo determinado, independientemente de la trayectoria concreta.
- Generación de trayectoria en el espacio cartesiano: Cuando es necesario que el robot siga una determinada trayectoria (por ejemplo, una línea de soldadura).

Para crear una trayectoria, primero se calcula la ecuación cartesiana orbital a partir del orden de movimiento. Luego se obtienen los puntos específicos muestreando el orbital cartesiano en el tiempo. Se transforman los puntos cartesianos en un espacio común (articular) en un modelo cinemático inverso. Finalmente, se crea la ruta (posición, velocidad y aceleración) en el espacio articular interpolando las coordenadas correspondientes a lo largo del tiempo.

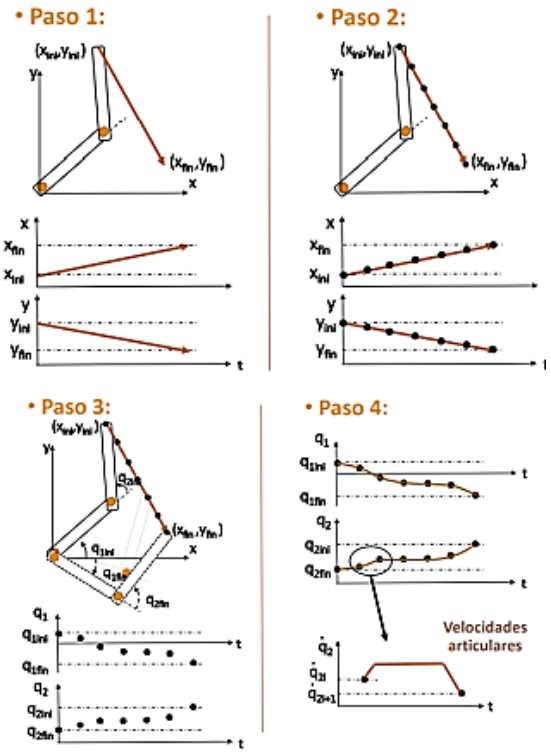


Ilustración 15. Pasos para la generación de la trayectoria

Los diferentes tipos de trayectorias se pueden clasificar principalmente en tres categorías:

- Trayectorias punto a punto.

  En este tipo de trayectoria, cada articulación se mueve de forma independiente de las influencias de las otras articulaciones. A su vez existen otras dos categorías dentro de esta: movimiento eje a eje y movimiento axial síncrono. En la primera solo se mueve un eje cada vez, a medida que aumenta el tiempo de ciclo; en cambio, en el movimiento síncrono los ejes se mueven al mismo tiempo y terminan el movimiento cuando finalice el eje que más tarde.

  Este tipo de trayectoria suele ser frecuente en robots simples o con una unidad de control limitada.

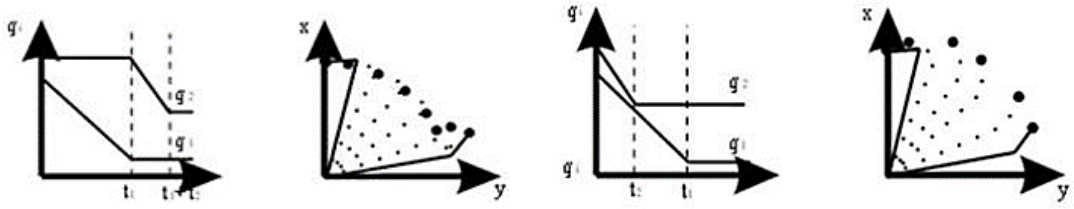


Ilustración 16. Trayectorias punto a punto.

- Trayectorias coordinadas o isócronas.

  En este tipo de trayectoria, el movimiento de todos los actuadores está coordinado y tiene la misma sincronización. Esto significa que el eje que tarda más en llegar a la posición deseada ralentizará al resto, de forma que ningún movimiento finalice antes que otra articulación, acabando todos a la vez. El tiempo total invertido en el movimiento es lo más corto posible, de forma que requiere menos velocidad y aceleración que otros tipos de movimiento. La desventaja de este tipo de planificador es que no se conoce de antemano la trayectoria que describe el punto extremo del robot.

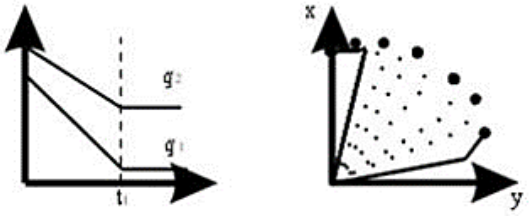


Ilustración 17. Trayectoria coordinada

- Trayectorias continuas.

  En este tipo de trayectoria, se conoce de antemano el recorrido seguido por el punto extremo del robot. Para esto, las trayectorias correspondientes deben estar alineados entre sí. Cada articulación individualmente parece tener un movimiento desorganizado, pero el resultado es moverse a lo largo de la ruta prevista como resultado.

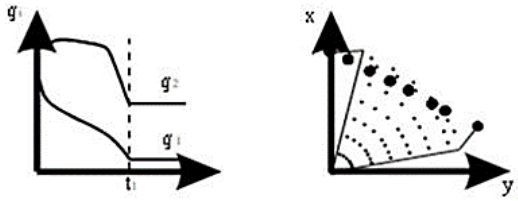


Ilustración 18. Trayectorias continuas

Suponiendo que la ruta del manipulador se define en el espacio cartesiano, existen dos alternativas para describirla:

La primera consiste en ubicar los lazos de control directamente en el espacio cartesiano y controlar el robot para que el error de la trayectoria se cancele en este espacio. Este caso se usa comúnmente en robots móviles, donde la curvatura de la trayectoria generada en el espacio cartesiano está directamente relacionada con la variable de control utilizada para seguir las trayectorias.

El segundo consiste en convertir la ruta del espacio cartesiano al espacio de trabajo correspondiente y controlar la progresión de cada variable adecuada definiendo los bucles de control en este espacio.

El problema del planificador de trayectorias debe resolverse en tiempo real. Por lo tanto, también es necesario hacer que la generación orbital sea computacionalmente eficiente.

Por otro lado, la interpolación de trayectorias implica encontrar un dato en un período de tiempo con valores máximos conocidos. En robótica, la interpolación es un conjunto sucesivo de puntos en el espacio articular por los que deben pasar las articulaciones del robot en un tiempo dado.

Para su determinación se emplean diferentes funciones polinómicas las cuales corresponden a interpoladores lineales, cuadráticos, cúbicos (splines), quinticos y a tramos.

- **Interpolador lineal:** Consiste en mantener constante la velocidad de movimiento por cada dos valores sucesivos (q_{i-1}, q_i) de la articulación. La evolución del valor articular a lo largo del tiempo se describe mediante la siguiente ecuación:

$$q(t) = \frac{q_i - q_{i-1}}{T} \cdot dt + q_{i-1} = V_q \cdot dt + q_{i-1}$$

Para asegurar una velocidad más o menos constante, es necesario asegurar que los instantes de la ruta elegida por los puntos q_i se tomen de acuerdo con los siguientes criterios: se intentara mover cada articulación q_i al destino lo más lejos y antes posible sin tener en cuenta otras conjugaciones. Se ajustarán los tiempos de transición para que las articulaciones tarden más en lograr movimientos coordinados. Se elegirá el tiempo de las especificaciones en el espacio de tareas de los terminales para describir una ruta predefinida.

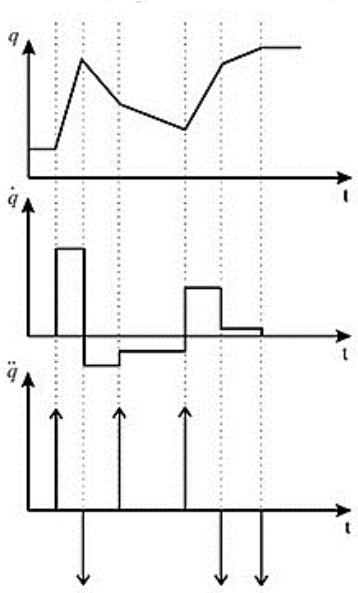


Ilustración 19. Posición, velocidad y aceleración para un interpolador lineal.

- **Interpoladores cúbicos (splines):** Curva suave que está parcialmente determinada por polinomios. Se suele utilizar la interpolación cubica ya que da resultados similares y solo requiere del uso de polinomios de bajo orden, evitando así oscilaciones no deseadas en la mayoría de las aplicaciones.

  Se utiliza cuando el objetivo es conseguir que la ruta que une los puntos por los que debe pasar la articulación en cuestión tenga una velocidad constante. Se utiliza un polinomio de grado 3 para conectar cada par de puntos.

$$q(t) = a + b \cdot \left(t - t^{i-t}\right) + c \cdot (t - t^{i-t})^2 + d \cdot (t - t^{i-t})^3$$

  Donde T = $\left(t - t^{i-t}\right)$

  Además, es capaz de hacer cumplir 4 condiciones límite usando 4 parámetros (2 posiciones y 2 velocidades).

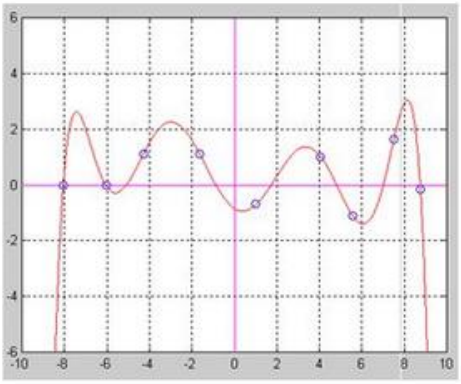


Ilustración 20. Trayectoria de un interpolador cúbico

Los coeficientes a y b, pueden determinarse a partir de las condiciones iniciales. De forma que:

$$a = q_{i-t}$$
$$b = \dot{q}_{i-t}$$

Una vez hallados estos dos parámetros, los demás coeficientes se hallan mediante las condiciones finales:

$$c = \frac{3}{T^2} \cdot (q - q_{i-1}) + \frac{1}{T} \cdot (\dot{q} + 2\dot{q}_{i-1})$$
$$d = -\frac{2}{T^3} \cdot (q - q_{i-1}) + \frac{1}{T^2} \cdot (\dot{q} + 2\dot{q}_{i-1})$$

- **Interpoladores quinticos.**

Al igual que el interpolador cúbico, este tipo de interpolación garantiza que la ruta que conecta los puntos que debe atravesar la articulación muestre continuidad en la velocidad. Un polinomio de quinto grado. La rotación pentagonal también muestra continuidad en la aceleración, lo que da como resultado una trayectoria suave que se puede lograr en un robot real.

$$q(t) = a + b \cdot \left(t - t^{i-t}\right) + c \cdot (t - t^{i-t})^2 + d \cdot (t - t^{i-t})^3 + e \cdot (t - t^{i-t})^4 + f \cdot (t - t^{i-t})^5$$

3. ELECTRÓNICA

En este apartado se tratará sobre los dispositivos electrónicos utilizados para el desarrollo del robot, seleccionando así los mecanismos que mejor se ajusten a la aplicación deseada.

3.1. ESPECIFICACIONES DEL ROBOT

La etapa principal del proceso de elección de los componentes consiste en definir lo que se desea lograr con el objeto diseñado. Solo cuando se determinan los objetivos a alcanzar, se puede seleccionar, entre las múltiples opciones disponibles, la que mejor se adapta a la aplicación prevista. Todo ello sin olvidar nunca las limitaciones que puedan existir, que vienen impuestas por multitud de factores como el coste, el proceso de fabricación, los materiales empleados, la tecnología disponible, etc.

En este trabajo se quiere obtener un brazo robótico capaz de manipular piezas pequeñas capaces de encontrarse en cualquier ambiente moviéndolas dentro de su área de trabajo de forma rotacional y traslacional. Para lograr esto, se decidió que el robot tendría que ser un manipulador con 4 grados de libertad, poseyendo así articulaciones rotacionales y prismáticas y un elemento final en forma de pinza.

3.2. SELECCIÓN SERVOMOTORES

Para el posicionamiento preciso de cada articulación, se requiere un par motor adecuado y una velocidad angular no demasiado alta. Los elementos de accionamiento más adecuados para esta aplicación son los motores de corriente continua y los imanes permanentes, ya que se pueden encontrar en tamaños reducidos y son fáciles de manejar. Sin embargo, el tipo de motor mencionado anteriormente funciona mejor a altas revoluciones, por lo que es necesario utilizar un reductor de engranajes que impulse el engranaje, aumentando así el par disponible.

Es evidente que se requiere de un sensor para medir el ángulo del eje de salida del reductor para determinar con precisión la posición de cada articulación; Esta medida se realiza con un potenciómetro conectado a dicho eje, cuya resistencia será proporcional al ángulo de giro.

Finalmente, se debe verificar la rotación del motor para llevar el eje de salida a la posición deseada y mantenerlo allí, este control se realiza electrónicamente.

La ilustración 15 se muestra el diagrama de bloques de este sistema: El ángulo del eje se mide usando un potenciómetro. La medida se compara con la referencia para obtener una señal de error que el regulador usa para ajustar la cantidad de impacto y, por lo tanto, la velocidad de rotación del

motor. La rotación del motor, a través de la caja de cambios, provoca un cambio en el ángulo del eje de salida que se mide constantemente para mantenerlo en la posición deseada.

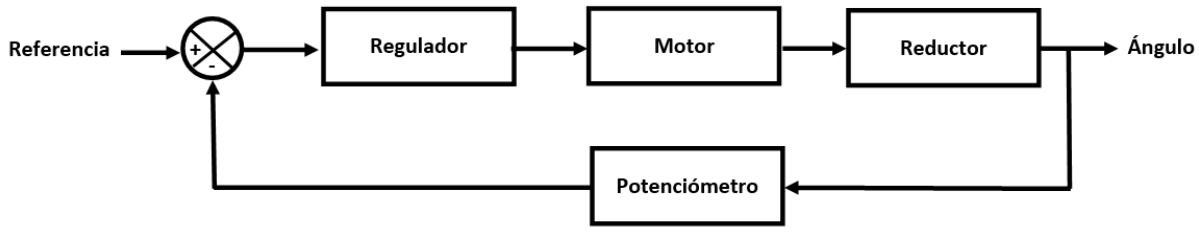


Ilustración 21. Diagrama de bloques del sistema de control

Dicho sistema de control de bucle cerrado se representa en lo que denominamos servomotor. Así mismo este es la combinación de un motor eléctrico y un reductor mecánico, un sistema que mide la rotación del eje del generador y un sistema de control que recibe el ángulo deseado y lleva el eje a la posición especificada, manteniéndolo incluso en presencia de ruido (diferencia de par resistivo). Utilizará como entrada la señal de control, analógica o digital, que representa el comando de la posición final del eje.

En la ilustración 16 se observa el esquema de los componentes que constituyen la estructura de un servomotor.

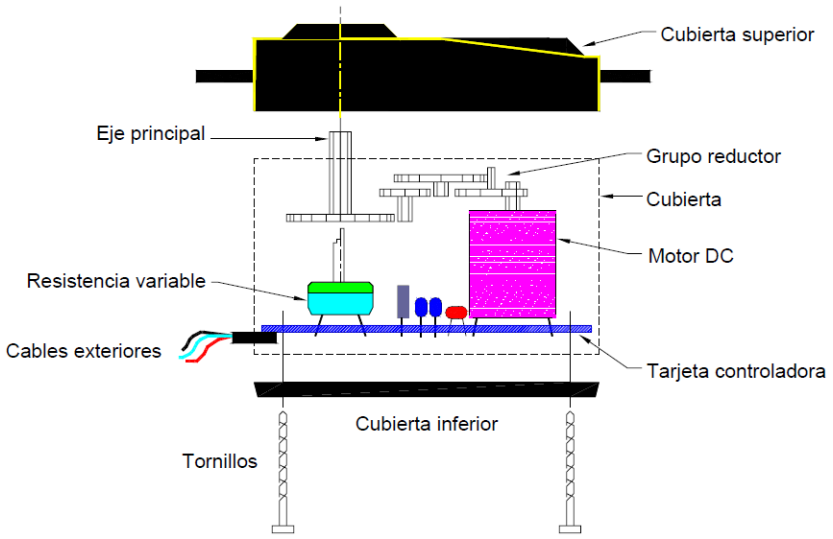


Ilustración 22. Esquema mecánico de los componentes de un servomotor

Los servomotores son muy utilizados porque combinan todos los elementos necesarios para poder realizar un posicionamiento angular preciso, con una amplia gama de tamaños y pares en el mercado. En este trabajo se utilizaron pequeños servomotores comunes en radiocontrol.

Existen diferentes modelos de servomotores en el mercado, y la principal diferencia entre ellos es el par. Es necesario entender bien este punto para elegir el servomotor adecuado. Es mejor elegir un servo con más par del que necesitamos, ya que el consumo de corriente es proporcional a la carga. Por otro lado, si un servomotor se somete a una carga superior a su par, existe el riesgo de dañar tanto la parte mecánica (engranaje) como la eléctrica del servo, y puede generar ruido en la fuente.

A continuación, se describe el procedimiento realizado para determinar el valor del torque adecuado. Debemos tener en cuenta que la configuración de nuestro robot no es igual a la de un robot serie, por lo que en este caso la fuerza de la gravedad no se va a oponer al giro del motor, aunque sí que vamos a encontrar algo de fricción. Para seleccionar los motores habrá que tener en cuenta el par máximo que se desea transmitir a la carga y de ahí definir el que deben tener los motores.

Comenzaremos estableciendo una trayectoria, la cual va a servir de referencia para el ciclo de prueba. Dicho ciclo va a constar de dos movimientos principales, uno en el plano X/Y y otro en dirección Z, puesto que nuestro robot trabajara realizando estos dos movimientos.

La coordenada utilizada para describir la trayectoria será cartesiana (x, y, z) con algún origen fijado arbitrariamente, medido en milímetros. La trayectoria del TCP comienza en el origen y la orientación del TCP estará alineada con el eje Z durante toda la trayectoria. Teniendo en cuenta estas condiciones la trayectoria será:

$$(0,0,0) \rightarrow (0,0,50) \rightarrow (75,0,50) \rightarrow (75,0,0) \rightarrow (75,0,50) \rightarrow (0,0,50) \rightarrow (0,0,0)$$

Como se puede ver en la trayectoria, el movimiento horizontal será de 75 mm (d_{top}) y el movimiento vertical será de 50 mm ($d_{vertical}$).

Como se ha descrito anteriormente, el ciclo de prueba consiste en dos movimientos principales, uno en la dirección Z y otro en el plano X/Y. Hay varios motores en funcionamiento para completar la secuencia, pero intuitivamente M1 o M2, motores que realizan el movimiento en el plano X/Y, serán los motores que realizarán el mayor trabajo. Por lo tanto, se elegirá primero uno de estos motores.

Se supone que el patrón de velocidad es triangular en el ciclo de prueba y el ciclo de trabajo se calcula en consecuencia. La justificación detrás de la elección del modelo de velocidad es el rendimiento medido por los ciclos de prueba. Aquí se supone que el robot debe aparecer con la máxima eficiencia, por lo que debe tener los servomotores del tamaño adecuado para que no alcance su máxima velocidad durante este ciclo. Sólo debería alcanzar el máximo en otros ciclos con recorridos más largos.

Siendo realistas, supondremos que el movimiento en el plano X/Y tardará un total de 0,5s, lo que supone un tiempo de recorrido horizontal de 0,25s (t_{top}). Esto deja un total de 0,25 s para todo el movimiento vertical.

$$V_{topAvg} = \frac{d_{top}}{t_{top}} = \frac{0.075}{0.25} = 0.3 \ [m/s] \tag{3.1}$$

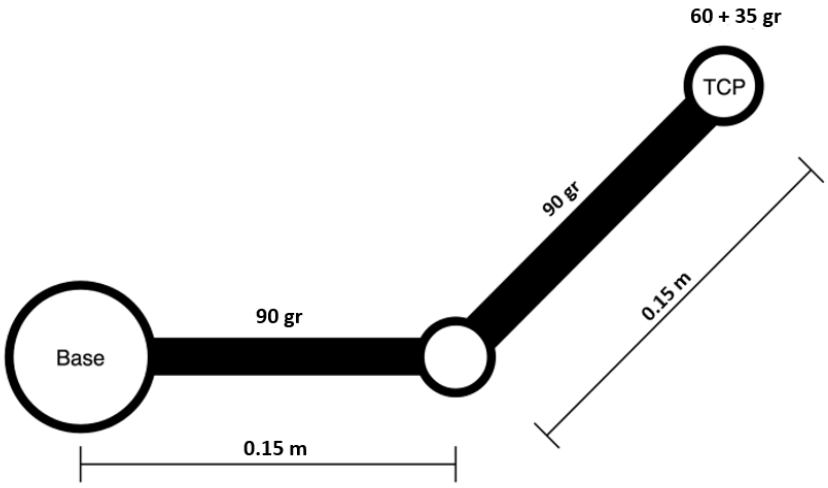


Ilustración 23. Figura simple del brazo, antebrazo y TCP con sus supuestos pesos y longitudes en el plano X/Y.

Como se ve en la ecuación 3.1 la velocidad media de la trayectoria horizontal es de 0.3 m/s. Suponiendo un patrón de velocidad triangular y un par de torsión constante en el que se cambia de signo a mitad de camino para disminuir la velocidad en lugar de aumentarla, la velocidad máxima alcanzará 0.6 m/s (V_{topMax}). Como se menciona en el tiempo para completar la trayectoria horizontal una vez es de 0,25 s. La aceleración será

$$a_{top} = \frac{V_{topMax}}{t_{top}/2} = \frac{0.6}{0.25/2} = 4.8 \ [m/s^2] \tag{3.2}$$

En la ilustración 17 se puede ver una estimación aproximada de la masa de los componentes, soportando así desde la base un total de 275 gr. También se especifican las longitudes de los brazos. Utilizando esta imagen el centro de masa (I_m) se supone que está a 0.1527 m de la base en el peor de los casos. Dado que la aceleración escala linealmente gracias a la aceleración angular constante y a la palanca (l), la aceleración del centro de masa será:

$$a_{top,m} = a_{top} \cdot \frac{l_m}{l} = 4.8 \cdot \frac{0.1527}{0.3} = 2.44 \ [m/s^2] \tag{3.3}$$

Así se obtiene el par deseado que actúa sobre el brazo:

$$\tau = a_{top,m} \cdot l_m \cdot m = 2.44 \cdot 0.1527 \cdot 0.275 = 0.1025\ [N \cdot m] \qquad (3.4)$$

Puesto que la mayoría de los modelos de los servomotores trabajan con las unidades de $kg \cdot cm$, ya que se usa para medidas métricas con respecto al valor del torque. Se realiza una conversión del valor obtenido en la ecuación (3.4) para obtener dicho resultado en el sistema de medida adecuado.

$$\tau = 0.1025\ [N \cdot m] \cdot \frac{0.102kg}{1N} \cdot \frac{10^{-2}cm}{1m} = 1.045\ [kg \cdot cm] \qquad (3.5)$$

Basándonos en este dato, se elige el modelo de servomotor más adecuado para la aplicación.

A continuación, se describe el modo de funcionamiento de los denominados "micro servos", que son servomotores compactos y potentes. Se utilizo un único modelo para todas las articulaciones del robot. Antes de enumerar las características del modelo elegido, explicaremos el uso de este.

Ilustración 24. Servomotor utilizado

Todos los servos utilizados en el robot tienen conectores de tres hilos: VCC (rojo), GND (marrón) y Señal (naranja).

Se controlan enviando un pulso eléctrico de ancho variable, o modulación de ancho de pulso (PWM), a través del cable de control (naranja). Hay un pulso mínimo, pulso máximo, que es igual al ángulo o posición del servo y la frecuencia de repetición.

Normalmente, los servomotores pueden girar 90° en cualquier dirección para un movimiento completo de 180°. La posición neutral del motor se define como la posición en la que el servo tiene la misma capacidad de girar en sentido horario y antihorario.

El circuito de control electrónico está configurado para recibir la referencia de posición deseada en el eje utilizando una señal de onda cuadrada con amplitud de 0 V a 5 V a una frecuencia de 50 Hz (ciclo de 20 ms) de ancho de pulso (PWM). El PWM enviado al motor determina la posición del eje y se basa en la duración del pulso transmitido a través del cable de control, donde este variará dentro del periodo total en el que la señal vale 5V (t_{ON}) y 0V (t_{OFF}).; el rotor girará a la posición deseada.

Cada servo espera ver un pulso cada 20 milisegundos (ms) y la longitud del pulso (t_{ON_0}) determinará hasta dónde gira el motor. Estos tienen su propio margen operativo, que corresponde a los anchos de pulso máximos y mínimos que comprende el servo. Los valores más comunes corresponden a pulsos de 1 ms a 2 ms de ancho, que accionarán el motor en ambos extremos (0º y 180º). Un valor de 1,5 ms indicará la posición central o neutra (90 grados), mientras que otros valores de pulso lo dejarán en posiciones intermedias. Si el tiempo es inferior a 1,5 ms, se moverá en sentido contrario a las agujas del reloj hasta la posición de 0°, si el tiempo es superior a 1,5 ms, el servo girará en el sentido de las agujas del reloj hasta la posición de 180°.

De este modo, para un servomotor con un ángulo total de rotación θ_{T0}, el valor mínimo del tiempo de pulso ($t_{ON_{min}}$) establecerá el eje en un ángulo de $\theta_{min} = -\theta_{T0}/2$; y el valor máximo del tiempo de pulso ($t_{ON_{max}}$) establecerá el eje en un ángulo de $\theta_{max} = \theta_{T0}/2$. Los valores de umbral de la duración del pulso y el ángulo de rotación dependen de diferentes fabricantes y modelos. La ilustración 3.2.5 muestra la forma de onda de la señal PWM para diferentes posiciones deseadas en el eje.

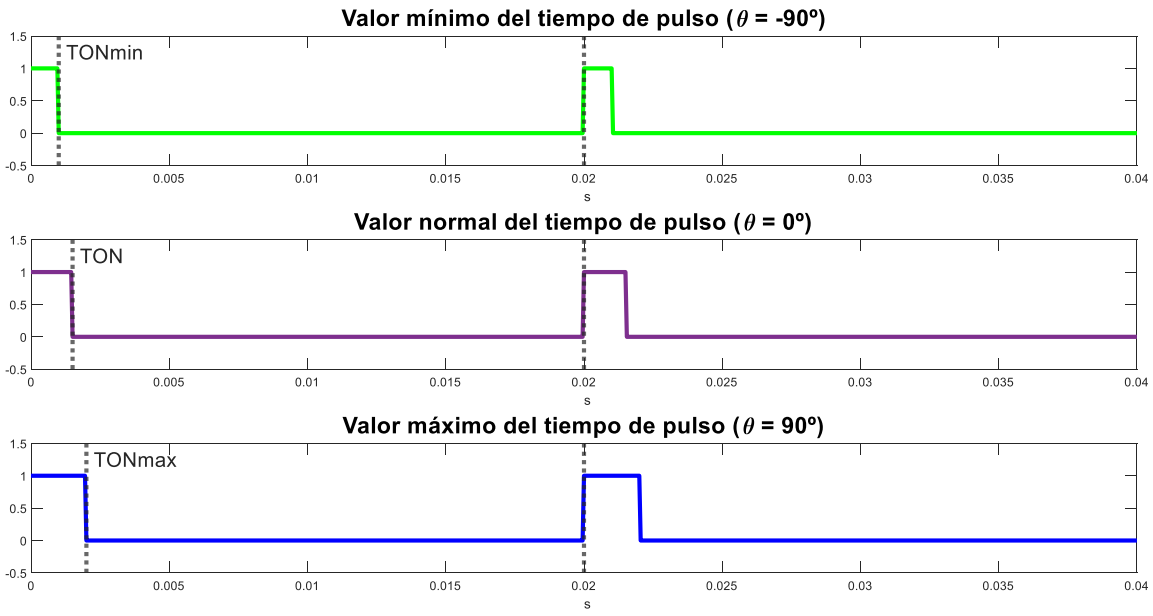


Ilustración 25. Señal PWM para posicionamiento del servomotor en distintos ángulos

Se puede establecer la relación entre el tiempo de pulso (t_{ON}) y el ángulo de giro (θ) mediante la expresión:

$$t_{ON}(\theta) = t_{ON_0} + \frac{t_{ON_{max}} - t_{ON_{min}}}{\theta_{T0}} \cdot \theta = t_{ON_0} + k \cdot \theta \qquad (3.6)$$

3.2.1. CARACTERÍSTICAS DEL SERVOMOTOR UTILIZADO

En el brazo robótico a diseñar se requieren un total de cuatro servomotores para las distintas uniones que lo componen y las abrazaderas finales. Se seleccionó los micro servomotores de la marca Tower Pro.

La Tabla 1 presenta las principales características de estos servomotores a una tensión de alimentación mínima de 4,8V (estos servomotores pueden operar con tensiones de 4,8V a 6,6V). Cuanto mayor sea la tensión de alimentación, mayor será el par y la velocidad. Dado que la potencia utilizada en este trabajo es de 5V, los valores de par y velocidad serán ligeramente superiores a los de la Tabla 1. Las dimensiones del actuador necesarias para diseñar los componentes se muestran en la Ilustración 20.

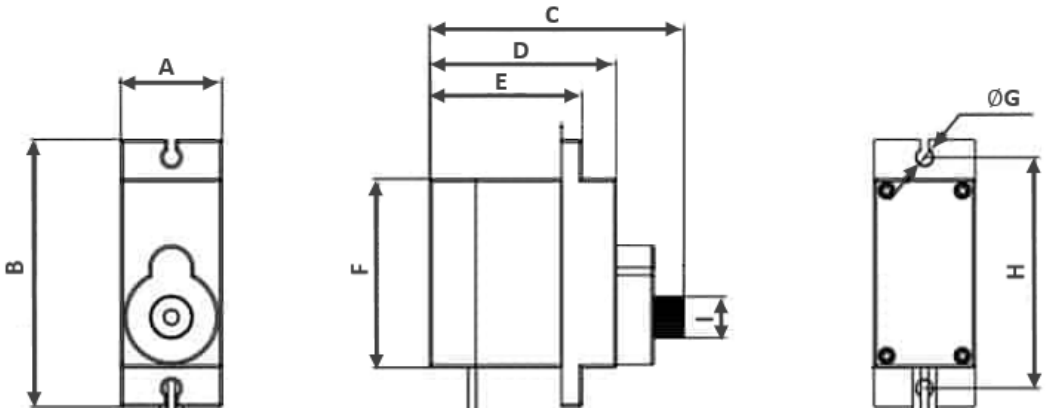


Ilustración 26. Esquema de los servomotores acotados

Modelo	**Par**	**Velocidad**	**Angulo**	**Masa**	**Dimensiones (mm)** (ilustración 3.2.5)								
					A	B	C	D	E	F	ØG	H	I
SG90	1,8 kg·cm	0.10 sec/60°	180º	9 g	12	32	32	23	19	22	3	27	5

Tabla 1. Características de los micro servomotores utilizados

El valor de la constante debe ajustarse empíricamente para obtener la mayor precisión posible en la localización del robot, ya que el fabricante no facilita ningún dato. Los valores obtenidos durante la prueba del turbocompresor se dan en la Tabla 2 y este valor se determina a partir de ellos.

Modelo	θ_{T0}	$t_{ON_{min}}$	t_{ON_0}	$t_{ON_{max}}$	k
SG90	180 º	1000µs	1500 µs	2000 µs	5.5̂ º/s

Tabla 2. Valor de la constante k para el micro servomotor utilizado

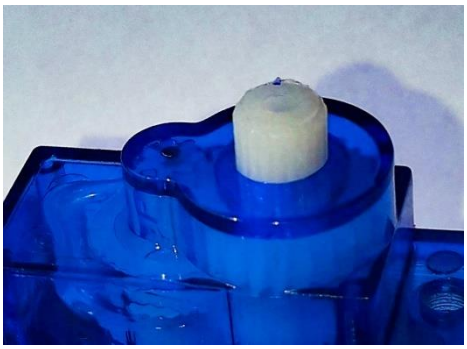


Ilustración 27. Detalle de la corona dentada del micro servo

La salida del movimiento del servomotor se produce mediante una corona dentada, como la que se muestra en la Ilustración 21, a la que se le debe acoplar algún tipo de pala o brazo para transmitir el movimiento. La ilustración 22 muestra las dimensiones de estas alas, que serán necesarias para diseñar diferentes partes del brazo robótico.

Debido al hecho de que el engranaje anular tiene 22 dientes, cada acoplamiento tendrá un error angular de ± 10 °, por lo que incluso si se forma un cierto ángulo determinado entre los eslabones del engranaje, cuando todos los engranajes están en sus posiciones cero, este ángulo solo puede lograrse ajustando o calibrando, utilizando el software de control.

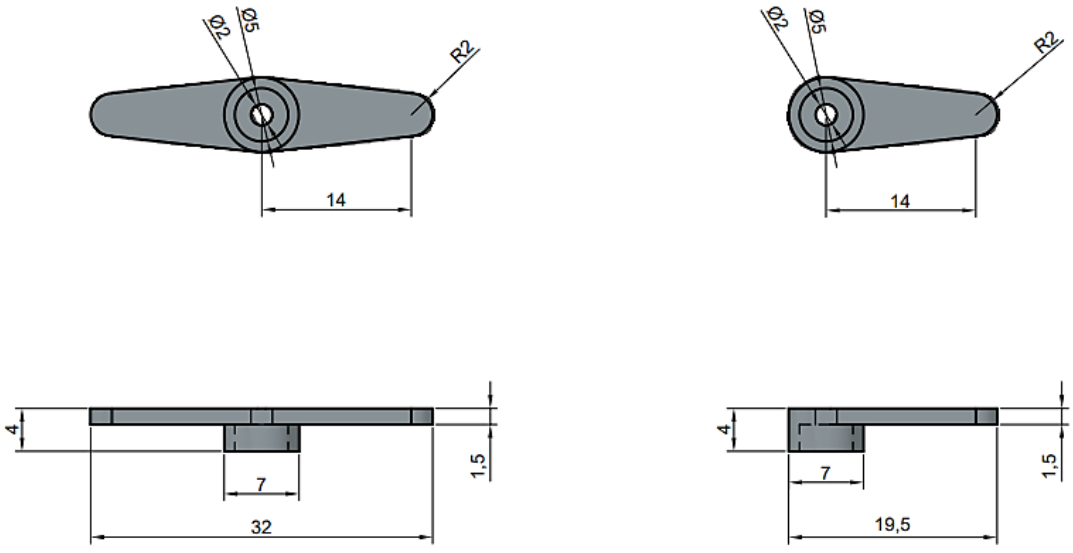


Ilustración 28. Dimensiones de las palas empleadas

Se utilizarán un total de 4 dispositivos, uno para el hombro, otro para el codo, otro para la muñeca y el ultimo para las pinzas. El servo correspondiente a la articulación del hombro se conectará al pin 13, el servo del codo al pin 12, el servo de la muñeca al pin 11 y por último el servo correspondiente a las pinzas se conectará al pin 10, así como se puede observar en la ilustración 23.

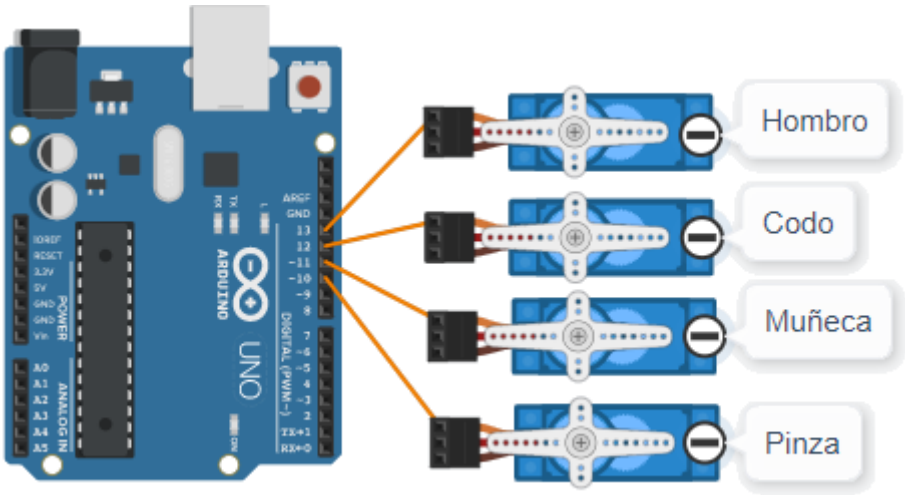


Ilustración 29. Circuito de conexión entre los servomotores y la placa Arduino

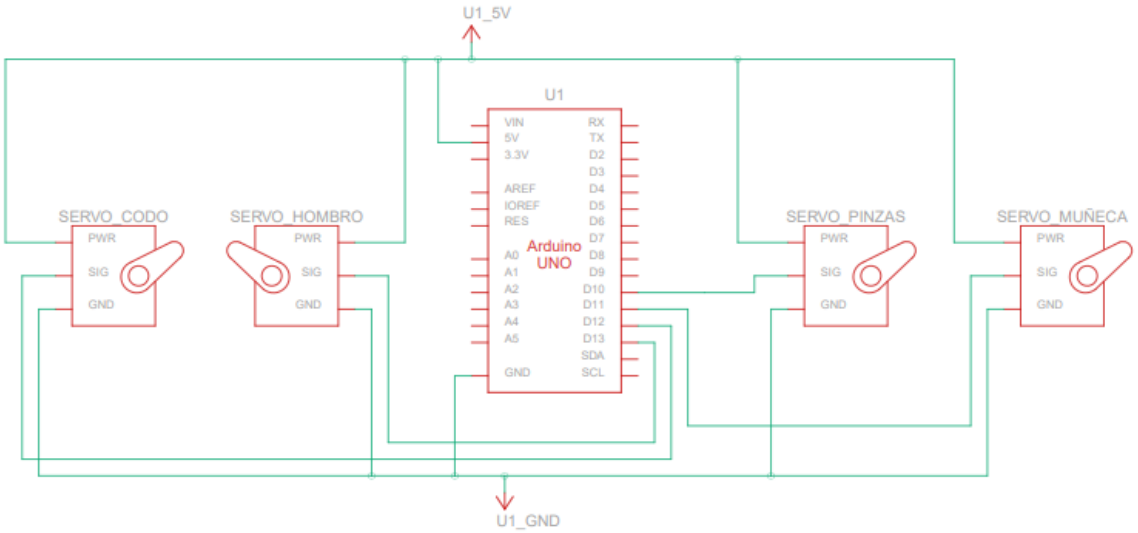


Ilustración 30. Esquemático de los servomotores

3.3. ACTUADOR LINEAL

Otro dispositivo que utilizaremos será un actuador lineal el cual nos permitirá realizar el movimiento en el eje z. Este mini actuador lineal es compacto, pequeño en tamaño y ligero en peso, con protección de cortocircuito, protección de sobrecarga, y diseño de protección de sobretensión. En la ilustración 24 se muestra el actuador lineal empleado estando este en su posición retraída y

extendida, además se proporciona otra ilustración (ilustración 25) con las dimensiones del actuador, así como en la Tabla 3 las características del mismo.

Ilustración 31. Actuador lineal empleado

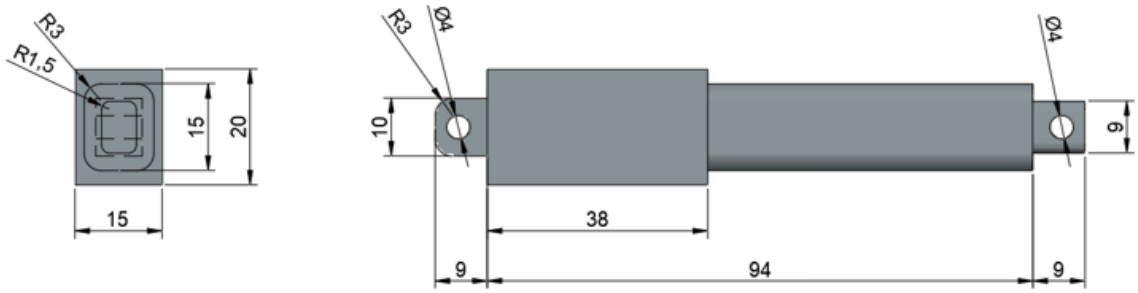


Ilustración 32. Dimensiones del actuador lineal empleado

Modelo	V_{in}	$Capacidad\ de\ carga$	Velocidad sin carga	Longitud de carrera	Duty Circle
Linear Actuator ER-60-90	12VDC	90N	9.5mm/s	50mm	10%

Tabla 3. Características del actuador lineal empleado

El actuador lineal eléctrico es alimentado por 12VDC. Cableando correctamente los polos positivo y negativo y aplicando el voltaje necesario, el actuador lineal se extenderá, luego para que se retraiga deberemos invertir la polaridad. Para realizar dicha operación de retroceso y avance deberemos utilizar un módulo controlador de motores L298N H-bridge, el cual permita controlar la velocidad y la dirección de los motores de paso a paso o de corriente continua.

Esto se consigue gracias a los dos puentes H que incluye, en nuestro caso solo tendremos que utilizar uno de ellos. Un puente en H consta de 4 transistores que nos permiten invertir el sentido de la corriente, de forma que podamos invertir el sentido de giro del motor. Este incluye un conector de 6 pines para entrada de señal TTL para controlar los motores, un bloque de terminales de 3 pines para la fuente de alimentación y dos bloques de terminales de 2 pines para la salida de los motores. En la ilustración 26 se muestran los diferentes componentes que constituyen el módulo.

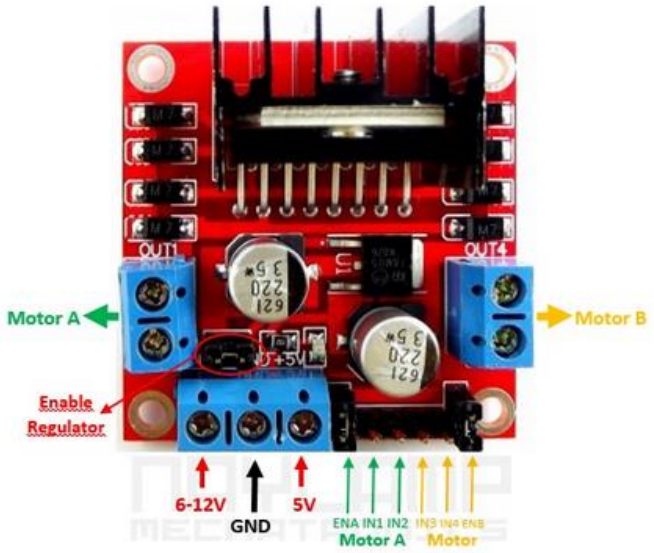


Ilustración 33. Esquema de componente del módulo L298N

El rango de voltaje en el que opera esta unidad es de 3V a 35V y la corriente es de hasta 2A. Se debe tener en cuenta el valor de la tensión del motor, puesto que la forma de alimentar el módulo variará de una forma u otra. Existen dos formas de alimentar el módulo: mediante una sola fuente o utilizando dos fuentes. En esta primera se utilizará una única fuente de tensión (6 – 12V), se conectará a la entrada de 12V y el jumper del regulador permanecerá cerrado, de forma que la entrada de 5V no se debe conectarse a ninguna fuente, ya que existirán 5V en este pin el cual puede usarse como una salida de 5V, pero que no debería consumir más de 500mA. Esta conexión debe realizarse para una tensión inferior a 12V para que el regulador no se sobrecaliente. Por otro lado, la conexión con dos fuentes de tensión deberá configurarse conectando una fuente de 5V a la entrada de 5V y la otra fuente con el valor del voltaje que trabaja el motor deberá conectarse al pin de 12V. En este caso el jumper le dejaremos abierto de forma que el regulador este desactivado.

Se utilizan estos dos tipos de alimentación del módulo puesto que el regulador solo trabaja con tensiones de hasta 12V, por lo que cuando excedamos este valor deberemos alimentar a la parte lógica del módulo desde otra fuente y quitar el jumper.

Por otro lado, para controlar el módulo los pines IN1, IN2, IN3, IN4 sirven para controlar el sentido de giro del motor, siendo los dos primeros para el motor A y los restantes para el B. El sentido del motor girara en un sentido u otro dependiendo si el valor de los pines se encuentra en alto o bajo, por ejemplo, si IN1 se encuentra en HIGH y IN2 en LOW el motor girara en un sentido, si cambiamos los valores girara en el otro. Si se quiere controlar la velocidad de giro se emplearán los pines ENA (motor A) y ENB (motor B) y se quitara el jumper, de forma que conectemos los pines ENA y ENB a dos salidas PWM en donde se enviarán valores entre 0 y 225 para controlar la velocidad de giro.

En la ilustración 27 podemos observar la conexión que realizaremos en nuestro caso, utilizando una fuente de tensión de 12V. Se utilizará únicamente los pines para controlar el sentido del giro del motor IN1 e IN2 conectador a los pines 6 y 7 respectivamente de la placa Arduino, y el pin empleado para variar la velocidad del motor A (ENA) el cual ira conectado al pin 8 de la placa.

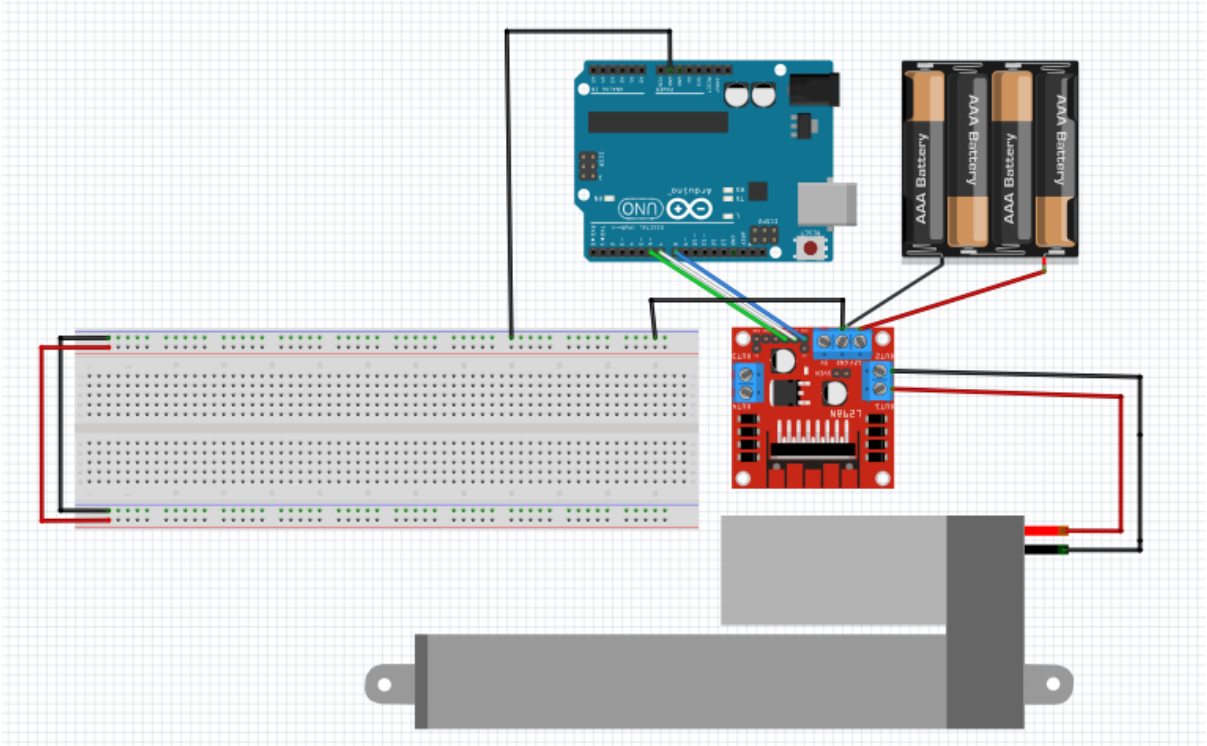


Ilustración 34. Circuito de conexión entre L928N, actuador lineal y placa Arduino

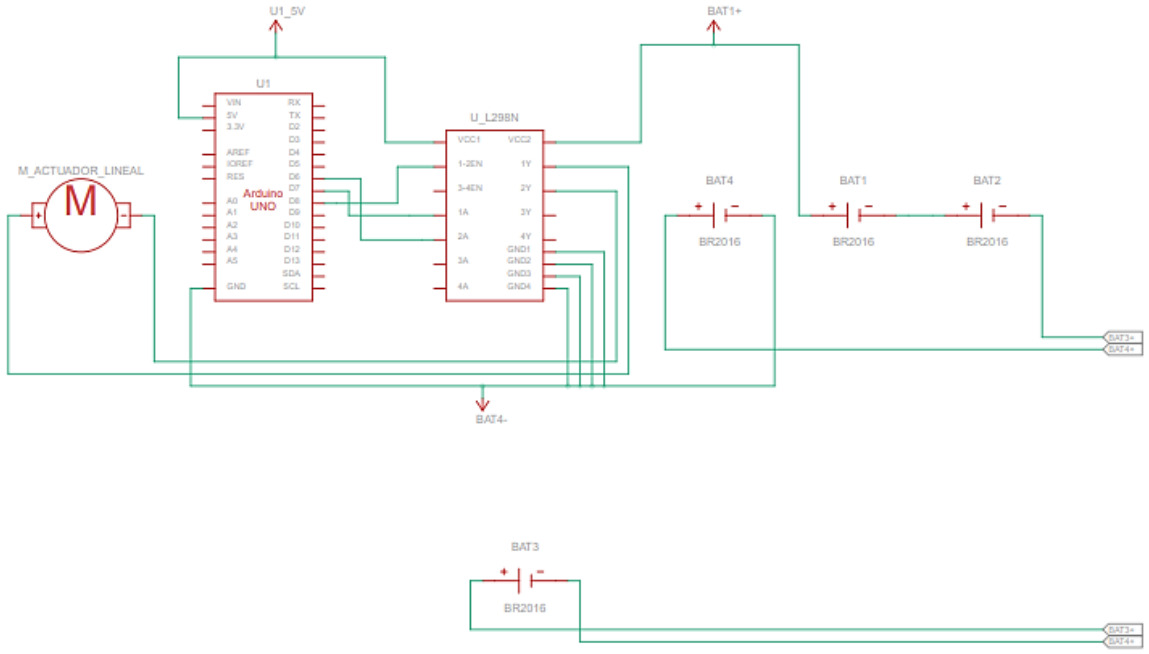


Ilustración 35. Esquemático entre L928N, actuador lineal y placa Arduino

3.4. OTROS DISPOSITIVOS

Además de los dispositivos nombrados anteriormente, también utilizaremos algunos más comunes como los leds.

Ilustración 36. Leds utilizados

Los Leds se utilizarán para indicar el modo de configuración del robot visualmente, puesto que el robot diseñado es configurable en 4 posiciones habrá que indicar en la posición de los enlaces en los que se encuentra, ya que la longitud de los mismo variará. Se utilizarán 4 leds: verde, azul, amarillo y blanco, así como se pueden observar en la ilustración 28. Si el led verde se encuentra encendido, este nos indicara que la configuración del brazo y antebrazo están en la primera posición, es decir, la longitud de estos es la menor. En cambio, si el led azul se encuentra encendido indicara que la

configuración del brazo y antebrazo están en la última posición, es decir, la longitud de estos es la mayor. Por otro lado, los leds amarillo y blanco indicaran las otras dos posibles combinaciones de posiciones, siendo para el led amarillo la posición 1 del brazo y la posición 2 del antebrazo, y viceversa para el led blanco.

Además, se añadirán a la conexión 4 resistencias de 10kΩ, una para cada componente anteriormente mencionado. La función principal de la resistencia en el circuito del led es evitar que estos se quemen, ya que la resistencia directa de los leds es baja.

En la ilustración 29 se muestra el esquemático de conexión entre los componentes y la placa, estando conectado el led verde al pin 2, el led azul al pin 3, el led amarillo al pin 4 y el led blanco al pin 5.

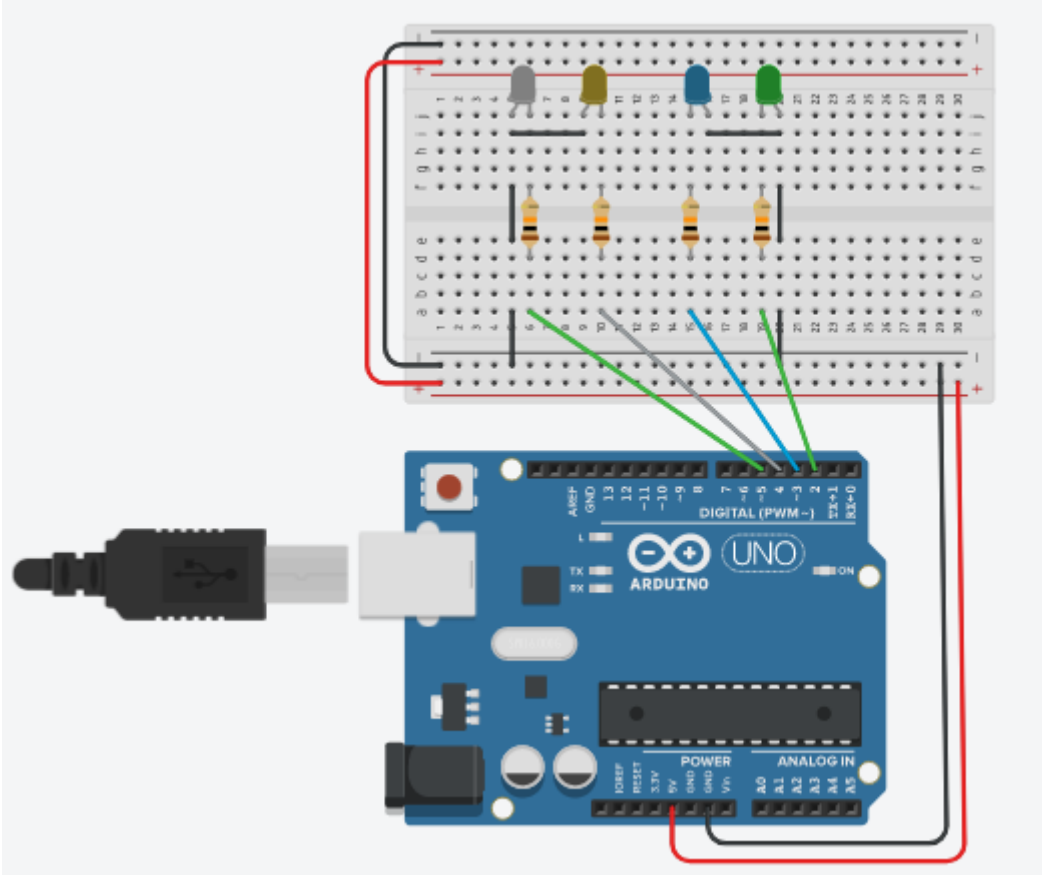


Ilustración 37. Circuito de conexión entre los leds y la placa Arduino

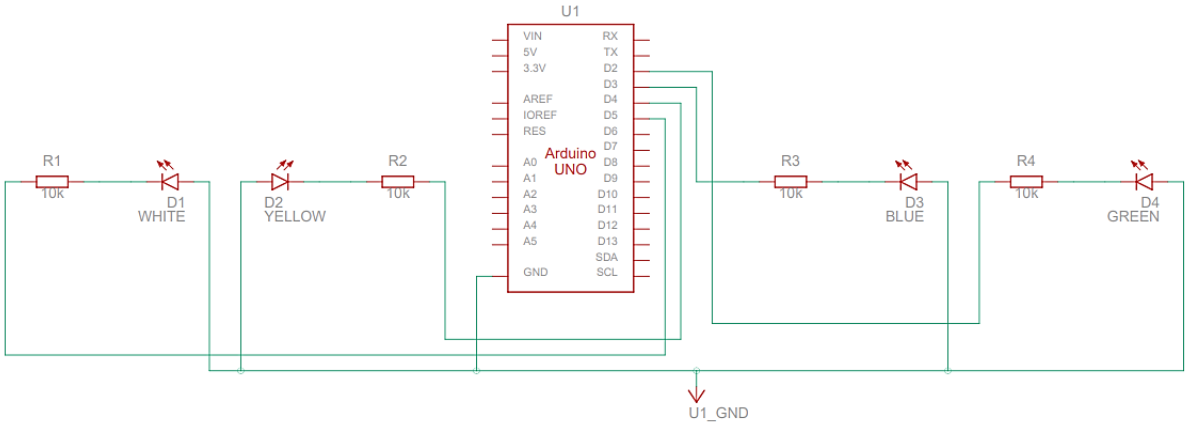


Ilustración 38. Esquemático entre los Leds y la placa Arduino

4. DISEÑO MECÁNICO

En este apartado realizaremos el proceso de diseño del robot. En primer lugar, se describen los requisitos que debe cumplir el robot. Luego, en base a la elección de los actuadores que mejor se ajustan a la aplicación y teniendo en cuenta el proceso de fabricación (impresión 3D), se diseñarán los elementos del robot. Una vez que se completen el diseño de las piezas, se llevará a cabo el montaje del robot.

4.1. ESPECIFICACIONES

Como se había comentado anteriormente en las especificaciones del robot, la fase principal en el proceso de diseño consta en determinar los requisitos que se quieran conseguir con el objeto a diseñar. Una vez definidas estas se podrá elegir la opción más adecuada a nuestra aplicación. Todo sin olvidarse de las limitaciones que puedan existir, las cuales vienen impuestas por muchos factores como el costo, el proceso de fabricación, los materiales utilizados, la tecnología disponible, etc.

En este trabajo se quiere obtener un brazo robótico capaz de manipular piezas pequeñas capaces de encontrarse en cualquier ambiente moviéndolas dentro de su área de trabajo de forma rotacional y traslacional. Para lograr esto, se decidió que el robot tendría que ser un manipulador con 4 grados de libertad, poseyendo así articulaciones rotacionales y prismáticas y un elemento final en forma de pinza.

4.2. SELECCIÓN DE MATERIALES

Se ha optado por realizar la fabricación del brazo mediante impresión 3D, ya que esta supone una gran ventaja a la hora de diseñar, fabricar y probar las piezas en el menor tiempo posible. Además, que con esta se consigue una mayor flexibilidad a la hora de diseñar las piezas específicamente para la aplicación en la que se requieren. Mediante la fabricación convencional, cada cambio en el diseño de una nueva pieza o cada pieza nueva requiere la producción de una nueva herramienta, matriz, molde o accesorio para crear la nueva pieza.

Hay docenas de materiales plásticos disponibles para la impresión 3D, cada uno con distintas propiedades que los hacen más adecuados para usos específicos.

Existen dos tipos principales de plástico:

- Los **termoplásticos** son el tipo de material plástico más utilizado. La principal ventaja que los distingue es su capacidad para resistir múltiples ciclos de fusión y endurecimiento. Los termoplásticos se pueden calentar y moldear a la forma deseada. Este proceso es reversible,

ya que no se forman enlaces químicos, por lo que los termoplásticos se pueden reciclar o fundir y reutilizar. Los termoplásticos a menudo se comparan con la mantequilla, que puede derretirse, endurecerse y derretirse nuevamente. Con cada ciclo de moldeo, las características cambian ligeramente.

- Los **plásticos termoendurecibles** (resinas curadas con calor) permanecen en un estado sólido permanente después del curado. Los polímeros en materiales termoendurecibles durante el curado generados por luz, calor o radiación apropiados se entrecruzan. Cuando se calientan, los termoplásticos se descomponen en lugar de derretirse y no cambian al enfriarse. Los plásticos termoestables no se pueden reciclar ni devolver a su composición. Una sustancia termoendurecible es como la masa para pasteles, una vez horneada no se puede volver a convertir en harina.

Los materiales más habituales para la impresión 3D son el ABS, PLA y sus diversas mezclas. Las impresoras de gama alta también pueden imprimir con otros materiales especializados que ofrecen propiedades como resistencia al calor, resistencia al impacto, resistencia química y dureza.

El filamento 3d seleccionado para la impresión de las piezas es el PLA, ya que este es uno de los materiales de modelado por deposición fundida utilizado en la impresión 3d basada en extrusión más fáciles de imprimir, ya que este puede imprimirse a baja temperatura y no requiere una cama calentada. Además, poseen unas características como la rigidez, fragilidad, posee una menor resistencia al calor y a los productos químicos, sin biodegradable e inodoros.

4.3. DISEÑO DE LAS PIEZAS

Como se mencionó anteriormente, el diseño de piezas está intrínsecamente ligado a dos determinantes: los actuadores elegidos y el proceso de fabricación de impresión 3D.

En este apartado se describe los componentes diseñados y se explica las características y la función de cada parte. Las dimensiones exactas y las proyecciones estándar se pueden mostrar en los dibujos incluidos en el anejo de planos. Todas las articulaciones del robot están conectadas mediante los actuadores elegidos, de forma que se artillen estos a las piezas. Los tornillos utilizados son tornillos autorroscantes de diferentes diámetros, y cabe señalar que los diámetros de los agujeros preparados para ellos se alcanzan después de varias pruebas de impresión porque si bien se pueden determinar, al realizar la impresión mediante aproximaciones poligonales de los círculos, el diámetro interior de estos siempre es menor al esperado. Además, en ciertas partes de las piezas, en las que encajan entre sí, será necesario limarlas de forma que encajen perfectamente. Puesto que

al realizar la impresión y ciertas piezas tener huecos, para imprimir estos utilizan columnas las cuales al retirarlas no dejan la superficie totalmente regular.

Todas las piezas han sido diseñadas utilizando el programa AutoCAD 2021 así como también se han realizado los planos normalizados en este.

4.3.1. Base

La base o cuerpo será el primer eslabón del brazo (eslabón 0). Este no se ha diseñado mediante impresión 3D puesto que se querían utilizar varillas roscadas para realizar la estructura o cuerpo del robot. Por lo que se optó por sujetar dichos elementos mediante una tabla como superficie de trabajo, lo suficientemente larga y del grosor necesario para otorgarle estabilidad al robot completo. Esta posee 4 agujeros pasantes, en forma de cuadrado, de tal modo que las varillas roscadas pasen a través de la superficie del material, se ajusten a la altura deseada mediante unas roscas y se unan al segundo eslabón del brazo.

4.3.2. Hombro

El hombro constituirá el segundo eslabón del brazo (eslabón 1). Cuenta con una hendidura para introducir la pala del micro servomotor (articulación 1), esta cavidad posee dos perforaciones de forma que sirvan de guía para atornillar la pala a la pieza mediante tornillos rosca chapa. De esta forma se busca que la pieza donde se atornilla el servo quede estática, otorgando el giro solo al servo. Además, consta de cuatro agujeros pasantes, uno en cada esquina, para poder fijar la pieza al cuerpo mediante las varillas roscadas mencionadas anteriormente y rosca.

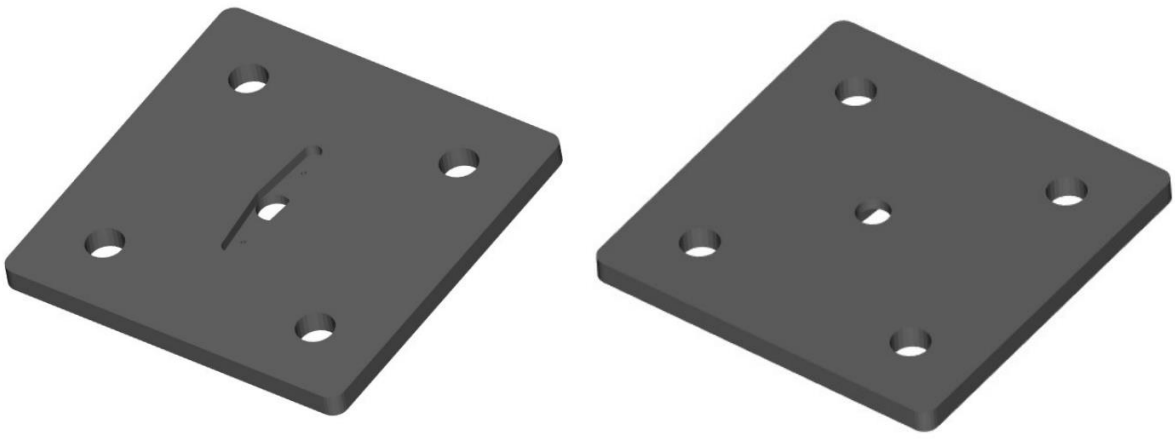


Ilustración 39. Hombro

4.3.3. Brazo

El brazo lo constituyen 2 piezas (eslabón 2). Puesto que una de las características del brazo era que este fuese configurable, se ha optado por diseñar dos piezas que vayan encajadas entre sí de forma que se consigan dos diferentes medidas del conjunto del eslabón.

La primera pieza de este eslabón cuenta con una hendidura donde se colocará el servo proveniente del hombro (articulación 1), de forma que se transfiera el movimiento de rotación a las restantes partes del robot. Como se comentó el robot es configurable, por lo que esta pieza consta de una cavidad en la cual se introducirá la segunda pieza del brazo. Para sujetar ambas piezas se han realizado dos agujeros, coincidentes con la segunda pieza del brazo de tal forma que se pueda introducir por ellos una varilla roscada para fijar la longitud especifica.

La segunda pieza de este eslabón al igual que la primera posee dos agujeros pasantes de tal forma que se pase una varilla roscada por uno de los dos agujeros proporcionando la longitud final del brazo, a excepción de que esta pieza no está hueca como la anterior ya que es la parte que completa a la primera pieza. Además, posee una hendidura en su extremo para introducir la pala del microservo (articulación 2) con el fin de atornillarlo a la propia pieza y que desde ese punto se transmita el movimiento de rotación sobre las demás partes del brazo sin afectar a la actual.

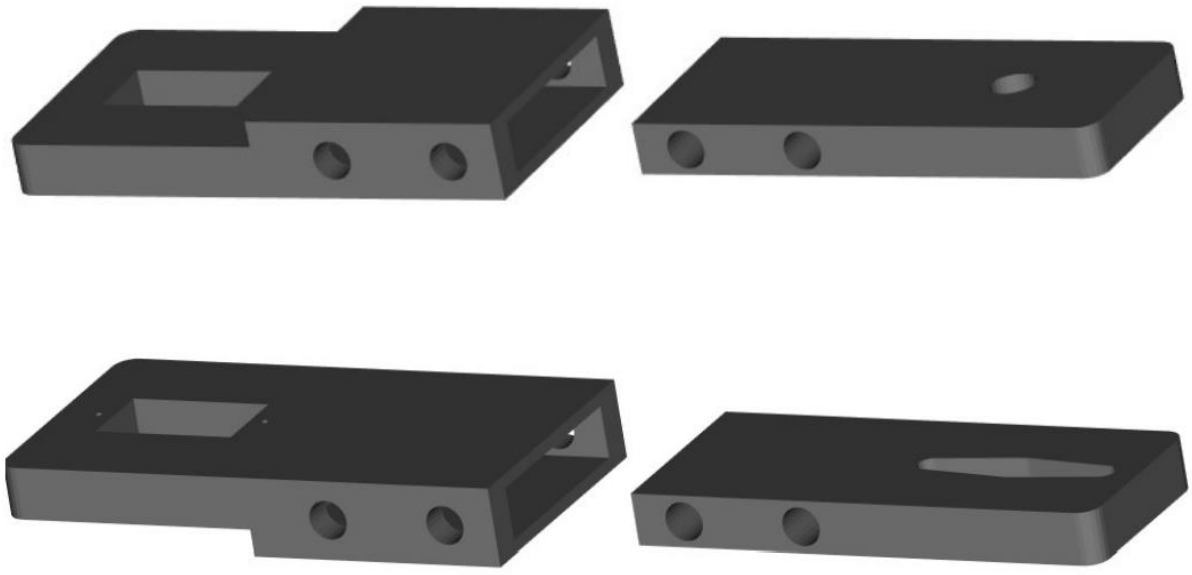


Ilustración 40. Brazo

4.3.4. Antebrazo

La morfología del antebrazo corresponde en muchos aspectos con la del brazo: está formado por dos piezas idénticas conectadas entre sí encajando la una de las piezas en la cavidad de la otra; con una hendidura donde se ajustará el servo (articulación 2), el cual cuenta con agujeros para atornillar a los mismos; y con agujeros pasantes en ambas partes de las piezas de forma que estas queden sujetas. A excepción del eslabón anterior, en la segunda pieza, ahora se ha realizado una perforación diferente puesto que en dicha hendidura se ajustará el actuador lineal (articulación 3). Para poder sujetar el mismo a la pieza se ha añadido una aleta cerca de la hendidura con un agujero de forma que se sujeten los dos elementos mediante un tonillo y una rosca.

Las dos partes juntas forman el tercer eslabón de la cadena cinética con las articulaciones 2 y 3 en sus extremos.

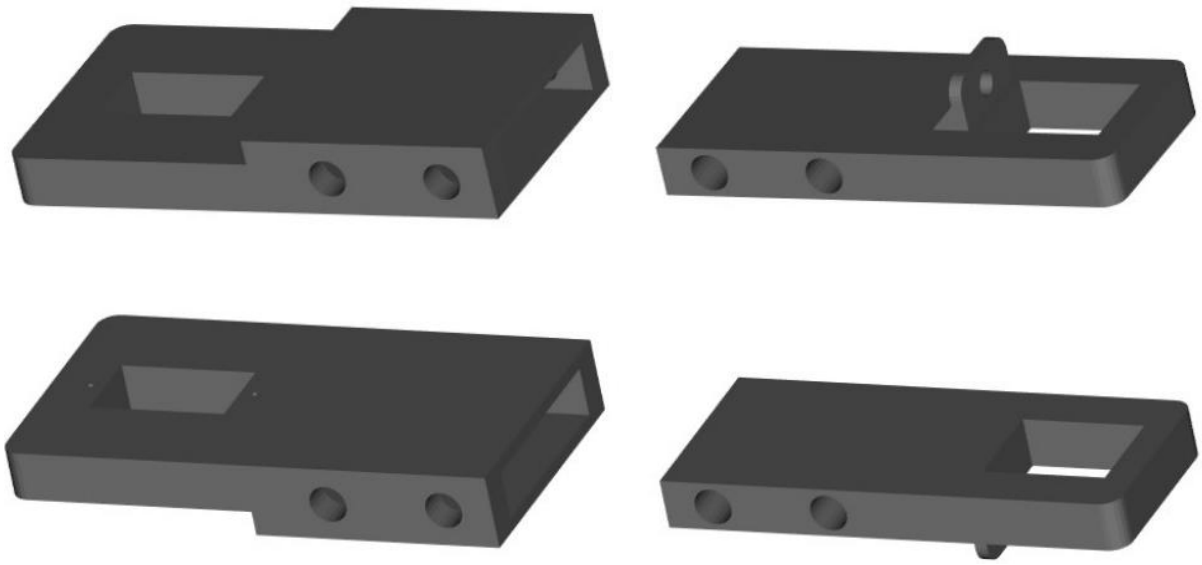


Ilustración 41. Antebrazo

4.3.5. Soporte actuador lineal

El soporte del actuador lineal, a pesar de ser una de las piezas más pequeñas, ha sido uno de los elementos más difíciles de imprimir, por su morfología. Cuenta con una cavidad en la cual se introducirá el servomotor (articulación 4) que posteriormente ira unido al soporte de las pinzas. Este también consta de dos pestañas perforadas las dos a la misma altura y con el mismo diámetro con el fin de sujetar la pieza al extremo del actuador lineal (articulación 4). Además, dispone en cada una

de las paredes verticales del soporte de dos soportes triangulares de forma que confiera mas dureza a la pieza en las zonas donde podría llegar a fracturarse la misma.

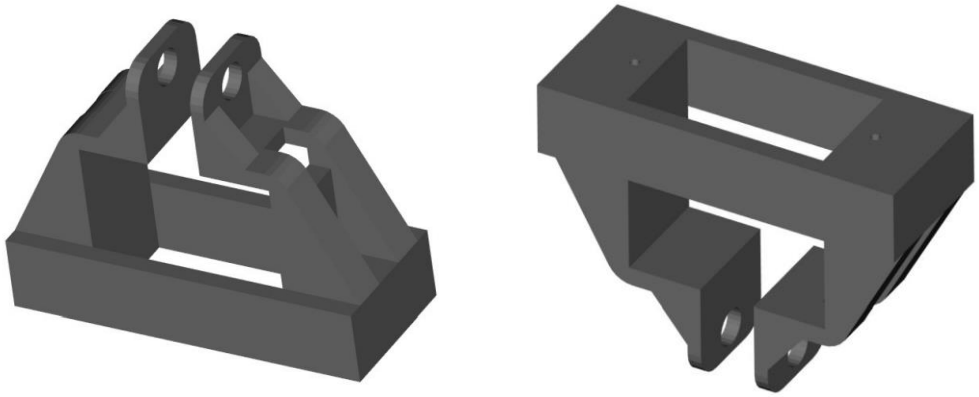


Ilustración 42. Soporte del actuador lineal

Esta pieza no forma parte de la cadena cinematica del robot, sino que es parte de la herramienta del brazo robótico.

4.3.6. Soporte de las pinzas

Al igual que ocurria con el soporte del actuador lineal, el soporte de las pinzas es otro de los elementos mas dificiles de imprimir por su morfología. Cuenta con una ranura en forma de pala por donde se sujeta el servomotor del soporte del actuador lineal (articulación 4). Se atornillará otro servo en la parte trasera, se hicieron varias incisiones en la pieza de acuerdo al tamaño de las aletas del mismo. También tiene un resalto que actuará como soporte para la pinza que no esté conectada al servomotor.

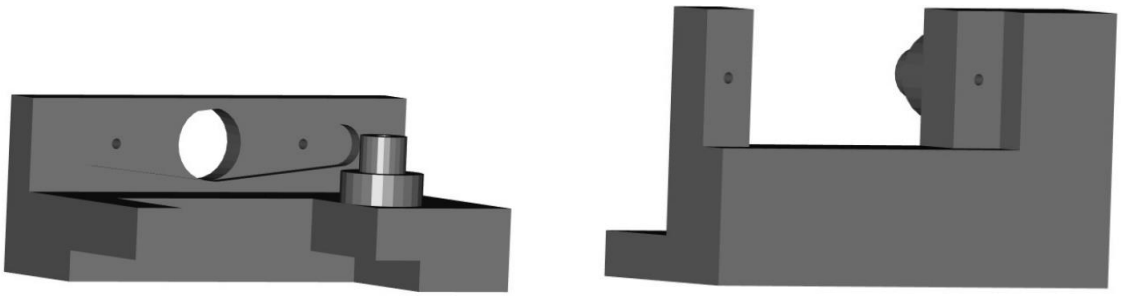


Ilustración 43. Soporte de las pinzas

Esta pieza no forma parte de la cadena cinematica del robot, sino que es parte de la herramienta del brazo robótico.

4.3.7. Pinzas A y B

Por último, el elemento terminal se diseñó en forma de pinza, está diseñado para que el robot pueda realizar la aplicación deseada: manipular piezas pequeñas desplazándolas de un punto a otro dentro del espacio de trabajo. Aunque sólo una de las pinzas vaya unida al servo, ambas tienen un rebaje en forma de pala para facilitar el agarre y por motivos estéticos de simetría. La pinza que va conectada al servomotor transmitirá movimiento a la otra unidad gracias a la conexión en forma de engranaje. Ambas piezas terminan con una superficie plana, lo que facilita agarrar objetos de pequeña escala.

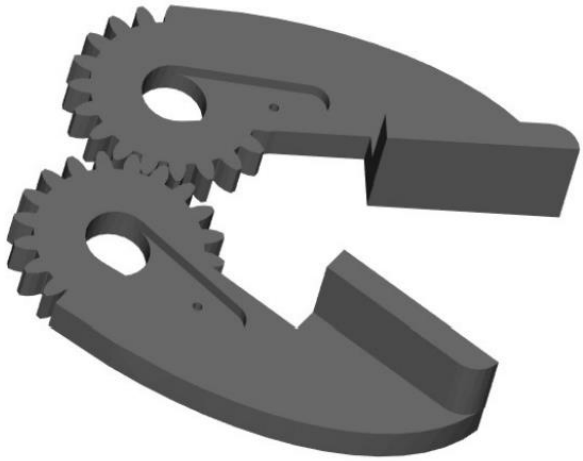


Ilustración 44. Pinzas

4.4. FABRICACIÓN

El siguiente paso en el diseño de las piezas es su fabricación.

Puesto que no se disponía de una impresora 3D propia, se optó por realizar el proceso de fabricación en una tienda especializada en impresiones 3D y Arduino.

Posterior al proceso de fabricación, las piezas impresas han tenido que ser limadas para un perfecto ajuste y acabado, ya que la impresión 3D posee tolerancias y las medidas que se proporcionan durante la elaboración del diseño no son las mismas en el acabado.

4.5. MONTAJE

Una vez fabricadas las piezas, se ensambla el robot. El montaje paso a paso de cada pieza se ilustra a continuación; Primero, se ha realizado el montaje de los eslabones puesto que es algo independiente del resto del robot (se pueden montar sin el requerimiento de que las demás partes estén también montadas) y luego las piezas que unen a los anteriores (no se pueden enlazar mientras no estén listas las piezas anteriores).

4.5.1. Montaje base-hombro

La base va unida al hombro mediante unas varillas roscadas. Se atornillaran estas mediante roscas a la altura la cual sea mas adecuada para realizar la aplicación. Por otro lado se atornillará la pala del servo que une el hombro y el brazo mediante tornillos rosca chapa en la cavidad de la parte posterior de la base. A continuación, el servo de la parte inferior se une a la pala y la parte trasera se instala en el eje del servo, de modo que el acoplamiento quede fijo.

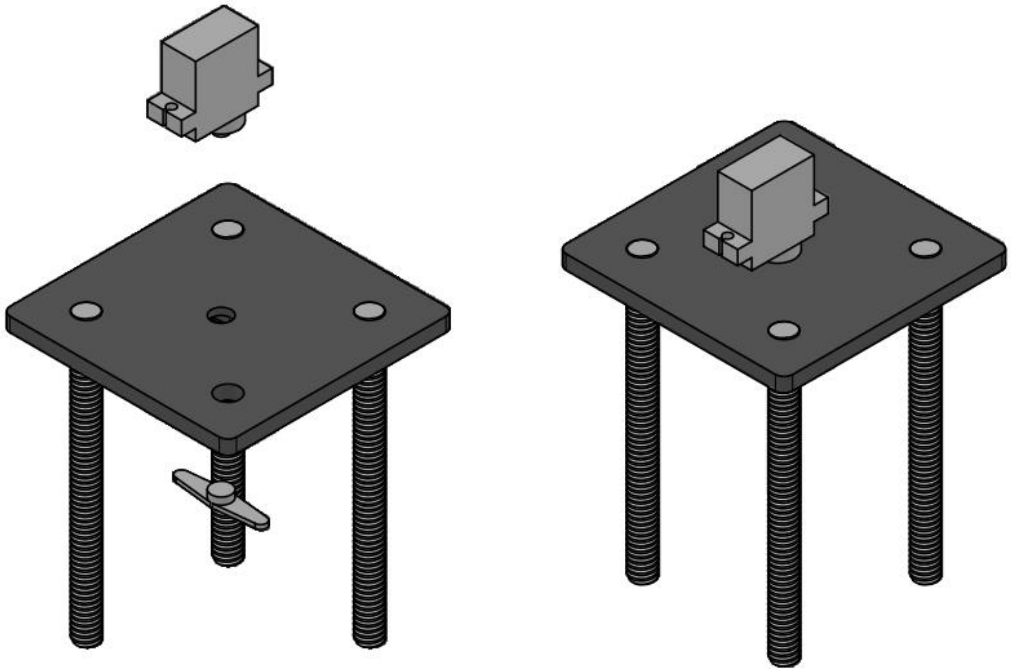


Ilustración 45. Montaje base-hombro

4.5.2. Montaje brazo

El brazo del robot consiste en la union de dos piezas de las cuales una de ellas se introduce en la cavidad de la otra y se fija mediante un tornillo pasante por uno de los dos orificios que tienen en comun las dos piezas. En cada extremidad de las piezas se le introduce un servo, el primero sera el que se acopla desde el el hombro y proporciona el primer movimiento articular a toda la estructura;

el segundo se acoplara insertando la pala del servo en la cavidad diseñada para el y atornillando el mismo a la pieza.

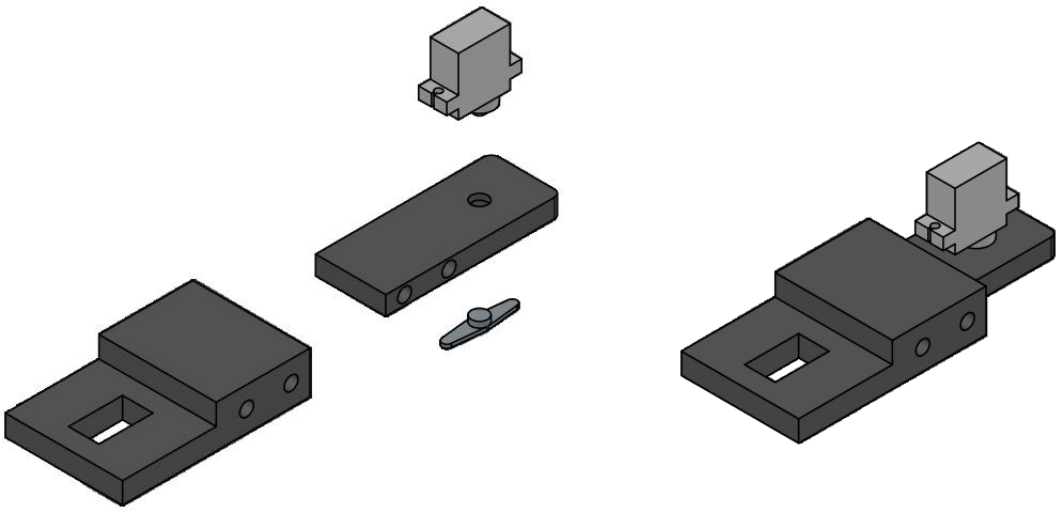


Ilustración 46. Montaje brazo

4.5.3. Montaje Antebrazo

Al igual que ocurre con el montaje del brazo, el antebrazo tambien consiste en la union de de dos piezas de las cuales una de ellas se introduce en la cavidad de la otra y se fija mediante un tornillo pasante por uno de los dos orificios que tienen en comun las dos piezas. A diferencia del anterior, en uno de los extremos no se introducira un servomotor, sino el actuador lineal el cual proporcionara el movimiento prismatico al manipulador. Este se atornillara a la pestaña al lado de la cavidad mediante dos tornillos y un aplique de sujeccion en forma de U. El servo introducido en el otro extremo sera el que se acopla desde el brazo.

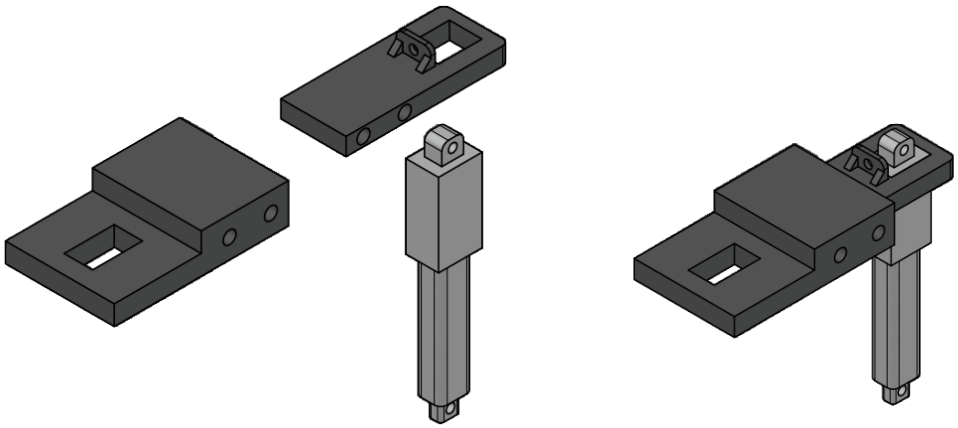


Ilustración 47. Montaje antebrazo

4.5.4. Montaje soporte pinzas

En este montaje, las pinzas se acoplarán al soporte. Por un lado, una pala simple se atornilla en cada pinza. Por otro lado, dos servos se unirán al soporte: uno se atornillará por las alas de este en las ranuras en la parte posterior del soporte; el otro se acoplará mediante una doble pala la cual ha sido previamente situada y atornillada en la hendidura del soporte con su misma forma. Finalmente, se encajarán ambas pinzas; la pinza A se atornillará al eje del servo en la parte posterior de la pieza y la pinza B a una protuberancia diseñada de modo que la pinza antes mencionada quede asentada y no en el aire. En este paso es importante que el servo esté en su posición neutra (0º) y que las pinzas estén completamente cerradas.

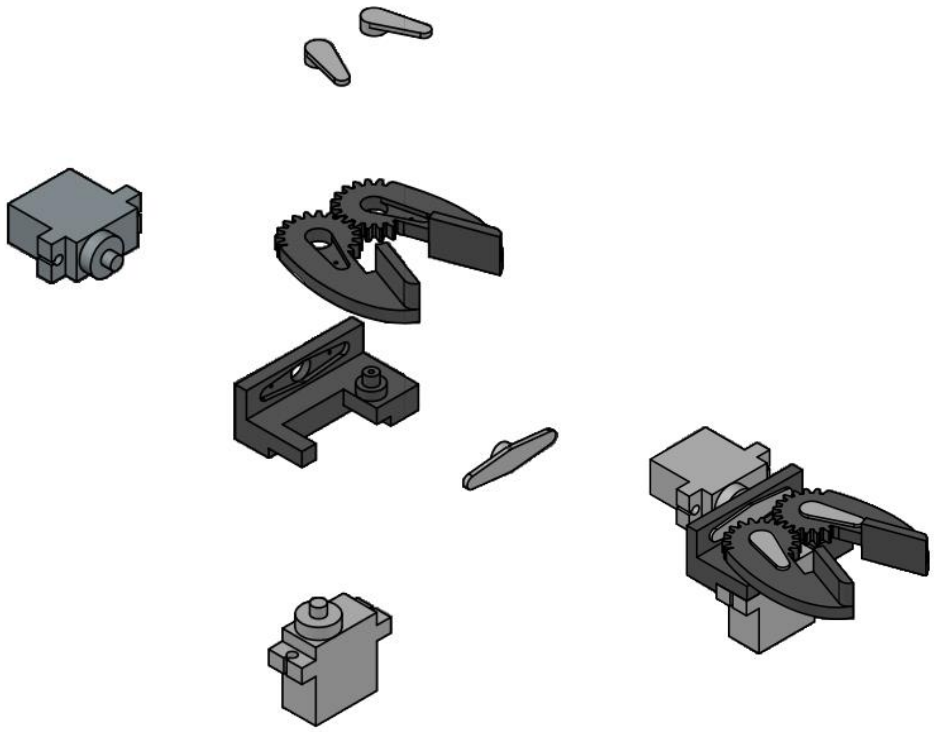


Ilustración 48. Montaje del soporte de las pinzas

4.5.5. Montaje completo robot

Una vez montados cada uno de los eslabones con sus respectivas piezas, se procederá a realizar el montaje de las piezas que conectan las anteriormente mencionadas entre sí.

Se comenzará uniendo el montaje base-hombro con el brazo, los cuales se acoplan mediante el micro servomotor proveniente del hombro el cual se atornillará a la pieza mediante unos tornillos rosca chapa en la parte posterior del brazo.

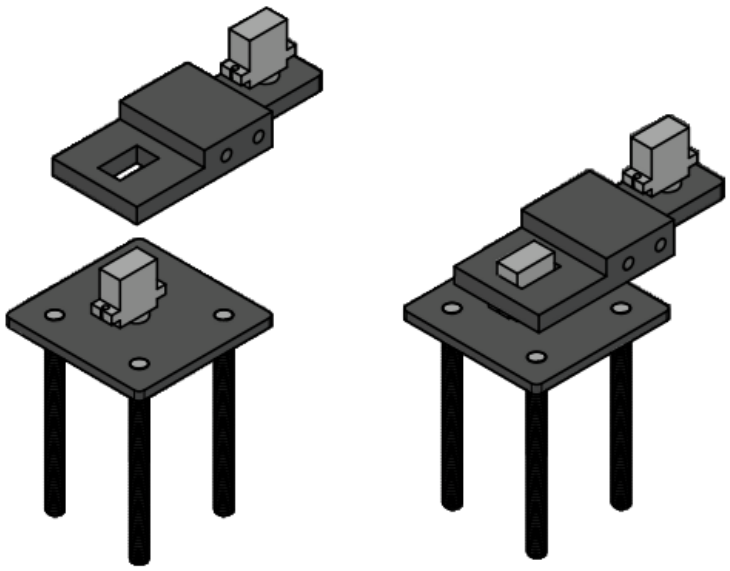


Ilustración 49. Montaje base-hombro con el brazo

Una vez unidas estos dos elementos se procede a acoplar al anterior el antebrazo. En la misma línea en la que se realizó el montaje del hombro-brazo se acoplara el antebrazo a la estructura. Se acoplan mediante el micro servomotor proveniente del brazo el cual se atornillará a la pieza mediante unos tornillos rosca chapa en la parte posterior del antebrazo.

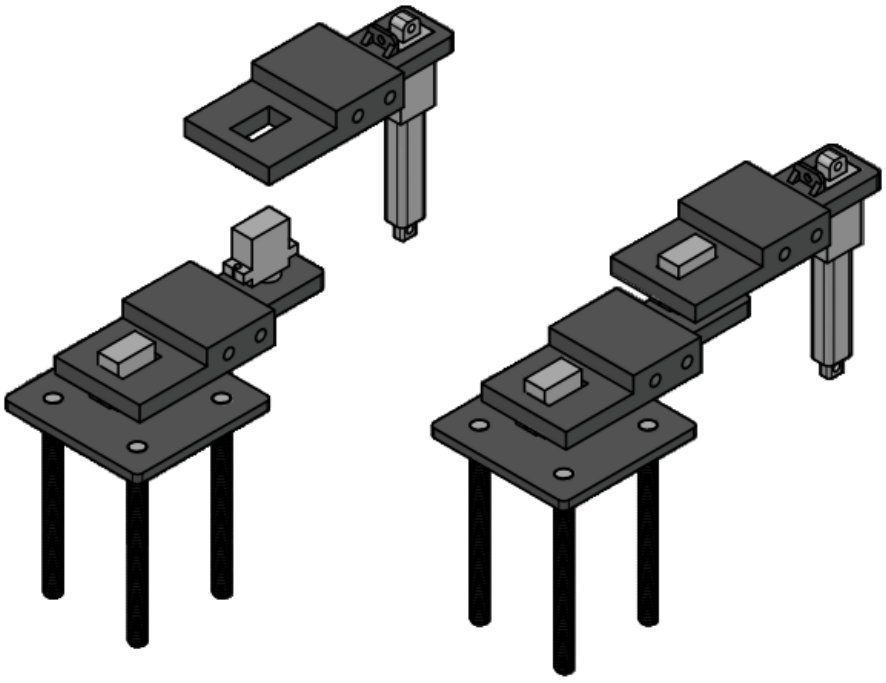


Ilustración 50. Montaje estructura-antebrazo

Finalmente se conecta el elemento terminal con el resto del manipulador. Para ello se utiliza un soporte en el actuador lineal el cual ayudara a la sujeción entre el actuador lineal y el soporte de la pinza. Estos se acoplan mediante el micro servomotor proveniente del soporte de la pinza el cual se atornillará a la pieza mediante unos tornillos rosca chapa en la parte posterior del soporte del actuador lineal. Una vez acoplados estos elementos, se acoplan al actuador lineal mediante las pestañas que posee el sopor en su parte superior al cual se introducirá un tornillo y se ajustará con una tuerca de forma que quede fijo.

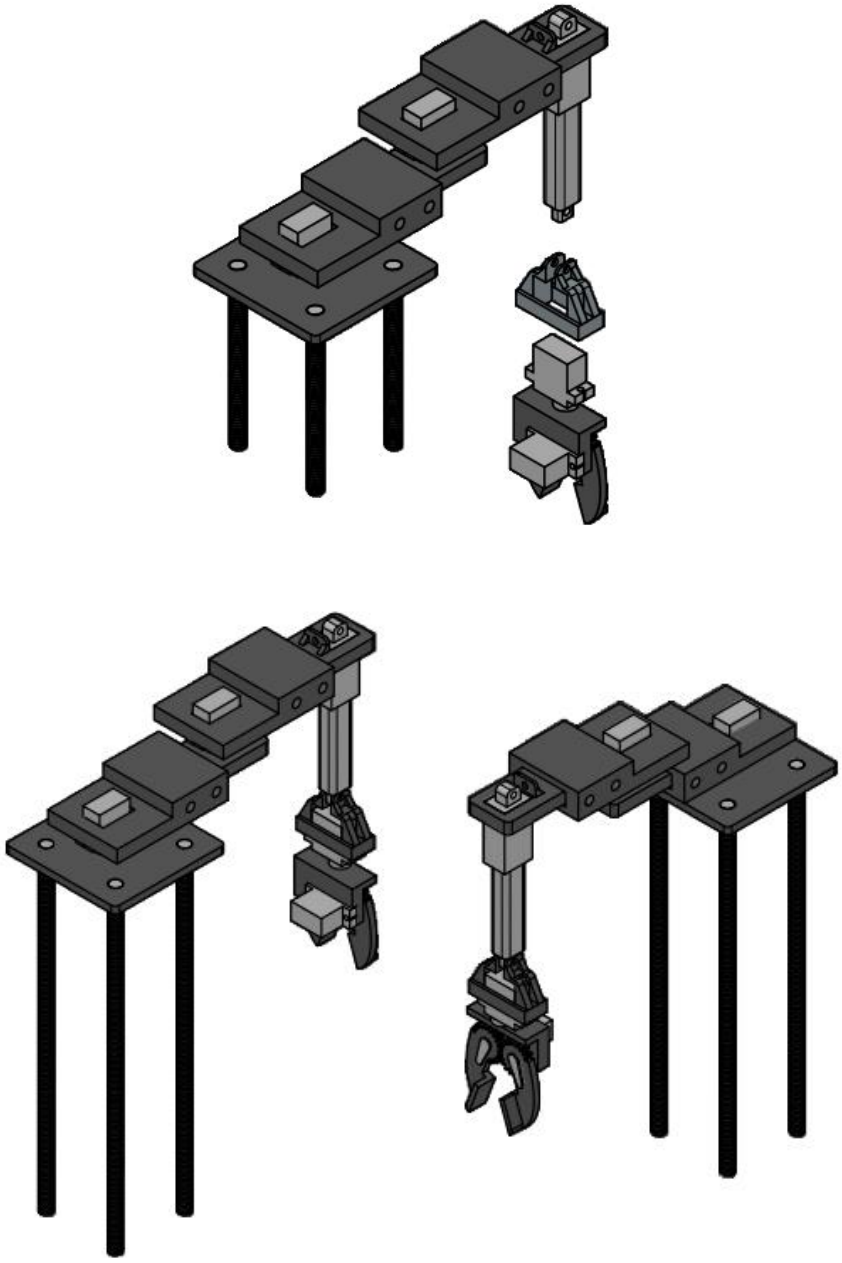


Ilustración 51. Montaje completo del manipulador

5. ANÁLISIS CINEMÁTICO DEL ROBOT

En este apartado se calcularán los modelos de cinemática directa e inversa, ya que serán necesarios para su programación y control. El conocimiento de la cinemática del robot será necesario para alcanzar las posiciones deseadas para que pueda realizar tareas útiles. Comenzaremos calculando el modelo mediante el método de Denavit-Hartenberg, y luego deduciremos la cinemática inversa a partir del mismo.

5.1. CINEMÁTICA DIRECTA

Una vez armado el robot, se comenzará a deducir su modelo cinemático. Este se basa en determinar la posición y dirección del punto extremo del robot en relación con el sistema de coordenadas base, conociendo los valores de las articulaciones y los parámetros geométricos de los elementos del robot. Se realizará mediante el algoritmo de Denavit-Hartenberg, el cual nos permite determinar la matriz de transformación homogénea que relaciona la orientación y posición del extremo del robot con respecto al sistema de coordenadas inicial. Para determinar dicha matriz de transformación seguiremos el algoritmo anteriormente explicado en el apartado 2.2.1.

En primer lugar, se identificarán los eslabones y uniones que componen la cadena cinemática del robot. Como se muestra en la ilustración 43, el robot tiene 4 eslabones y 4 articulaciones, por lo que tendrá 4 grados de libertad. Los eslabones y las articulaciones iniciando en 1 y finalizando en 0. En el caso de los enlaces, habrá un enlace 0 que coincide con la base fija del robot. Luego, se localizará la ubicación de los sistemas de coordenadas. Se encontrará y etiquetará los ejes de las articulaciones desde Z_0 hasta Z_{n-1}. Puesto que nuestro brazo robótico consta de 4 articulaciones, se tendrá que encontrar los ejes de articulación de Z_0a Z_3.

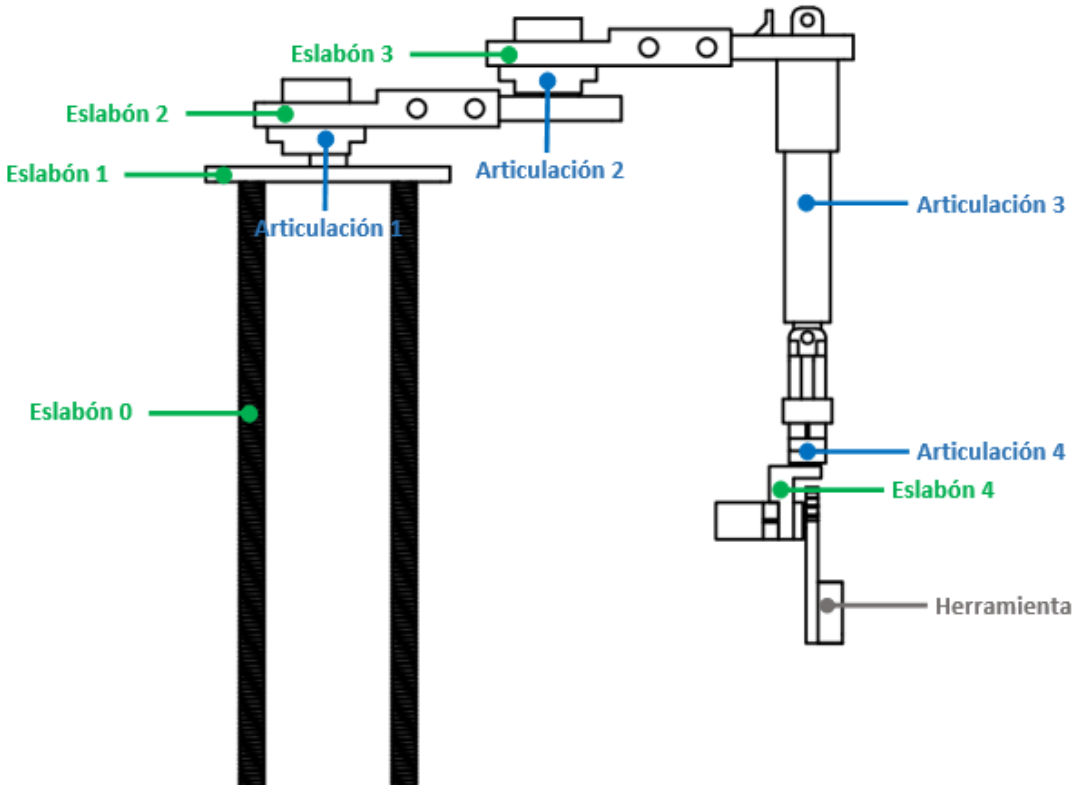


Ilustración 52. Numeración de las articulaciones y eslabones del brazo robótico

En la ilustración 44, se observa los correspondientes ejes Z_{n-1} de las articulaciones sobre las que se realizarán los correspondientes movimientos rotacionales y primaticos. Además, se define el marco de coordenadas inicial colocado en la base, compuesto por X0 e Y0 de forma que se cumpla la regla de la mano derecha.

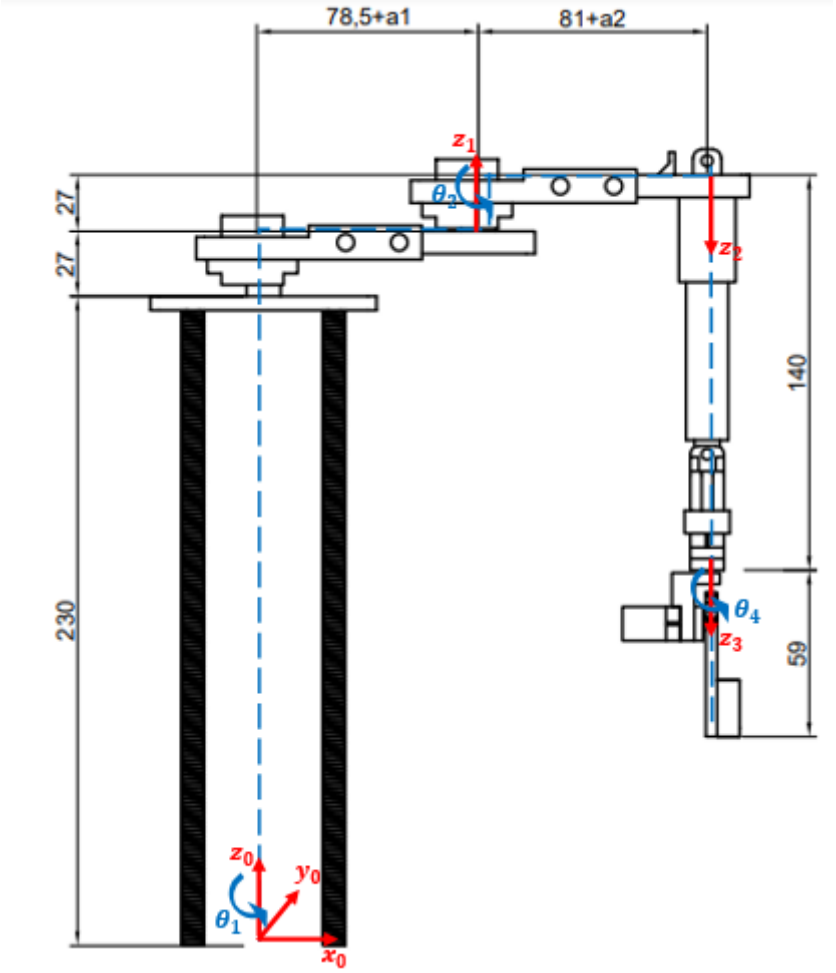


Ilustración 53. Sistema de coordenadas inicial y colocación de los ejes de revolución y prismático

Una vez colocadas las direcciones del movimiento articular de cada articulación y el sistema de coordenadas de referencia (base) se determinarán los ejes de los sistemas de coordenadas restantes. Se realizará una serie de pasos los cuales se aplicarán por igual a cada articulación. Se utilizará para nombrar los ejes el subíndice i, que iniciará en 1 y finalizara en n-1, donde n es el número total de eslabones (en nuestro caso 4).

En primer lugar, se determinará dónde colocar el origen del sistema de coordenadas para el eje, en el caso de que Zi-1 y Zi sean paralelos este origen se colocará en la misma unión, por el contrario, si estos no son coplanares, se colocará en la línea de mínima distancia.

Una vez determinado el origen de los ejes, se situará la dirección del eje Xi. En el caso de que Zi y Zi-1 se crucen, se colocará el eje Xi en la línea perpendicular al plano formado por ellos, si estos son paralelos, el eje Xi deberá colocarse perpendicular a ambos y preferiblemente pasando por Oi-1, en cambio, si los ejes Zi y Zi-1 no son coplanares, Xi deberá alinearse con la perpendicular común.

Por último, se colocará el eje Yi, el cual se determinará siguiendo la regla de la mano derecha.

En la ilustración 45 se muestra la colocación final de los sistemas de coordenadas para cada articulación.

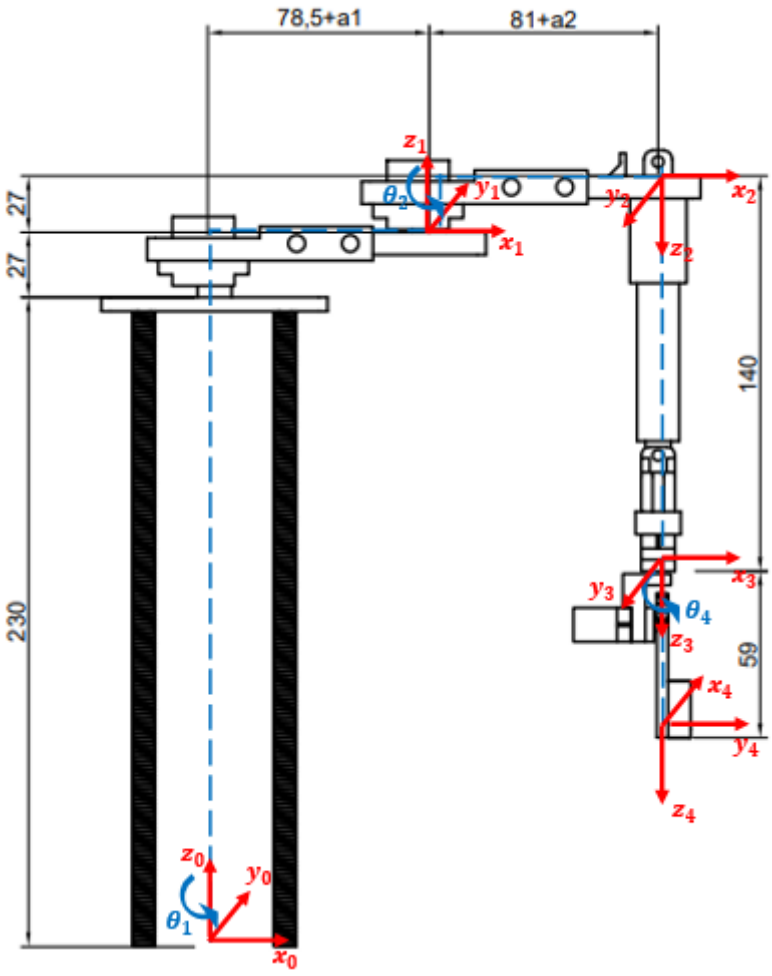


Ilustración 54. Sistemas de coordenadas de cada articulación

Cabe señalar que en este análisis no se toma en cuenta el elemento terminal (pinza A y B), ya que, si bien se abre y se cierra para poder agarrar objetos, este movimiento no es considerado un grado de libertad por parte del robot.

Una vez asignados los sistemas de referencia a cada enlace, se calcularán los parámetros de Denavit-Hartenberg. Estos son necesarios para construir las matrices de transformación de cada articulación, así como la matriz de transformación homogénea total. La fila i-ésima de esa tabla contiene los parámetros que transforman el marco de coordenadas "i-1" en el marco de coordenadas "i".

- θi: Es el ángulo de la articulación del eje x_{i-1} al eje x_i, medido con respecto al eje z_{i-1}, utilizando la regla de la mano derecha.

- di: Es la distancia medida desde el origen del sistema i-1, a lo largo del eje z_{i-1}, hasta la intersección del eje z_{i-1} con el eje x_i.
- ai: es la distancia de separación entre los orígenes de los sistemas de referencia i-1 e i, medida a lo largo del eje x_i hasta la intersección con el eje z_{i-1} (o la distancia más corta entre los ejes z_{i-1} y z_i, cuando no se cruzan).
- αi: el ángulo que separa los ejes z_i y z_{i-1}, medido a lo largo del eje x_i.

Teniendo en cuenta las normas para la construcción de la matriz de transformación y siguiendo los valores de la ilustración 5.1.3, se muestra a continuación la determinación de dichos parámetros:

Primera fila de la tabla (transformación del sistema de coordenadas 0 al 1):

θ1: Ángulo entre los ejes X0 y X1 al girar alrededor del eje Z0. Puesto que es una variable de articulación rotacional se asignará la correspondiente variable θ.

d1: Distancia desde el origen del sistema 0 a la intersección entre Z0 y X1 medida a lo largo del eje Z0. En este caso el punto de intersección es el origen del sistema 1. Por lo tanto, la distancia es **253**.

a1: Distancia entre los ejes Z0 y Z1, medida a lo largo del eje X1. En este caso hay un desplazamiento desde el origen del sistema de coordenadas 0 al 1, por lo que la distancia entre ellos será **78.5+a_1**. Cabe destacar que puesto que nuestro manipulador es configurable la distancia a_1 puede variar dependiendo el ajuste que se le otorgue, por lo que a la medida inicial se le añade una variable la cual indicará este cambo de configuración.

α1: Ángulo entre los ejes Z0 y Z1, al girar alrededor de X1. Como son paralelos, ese ángulo es **cero**.

Articulación	θ	d	a	α
1	$*\theta_1$	257	78.5+a_1	0

Tabla 4. Parámetros de Denavit- Hartenberg para la primera articulación

Segunda fila de la tabla (transformación del sistema de coordenadas 1 al 2):

θ2: Ángulo entre los ejes X1 y X2 al girar alrededor del eje Z1. Puesto que es una variable de articulación rotacional se asignará la correspondiente variable θ.

d2: Distancia entre el origen del sistema de coordenadas 1 y la intersección entre Z1 y X2 medida a lo largo del eje Z1. En este caso existe un desplazamiento desde el origen del sistema de coordenadas 1 al 2, por lo que habrá que tener en cuenta esta distancia de **20**.

a2: Distancia entre los ejes Z1 y Z2, medida a lo largo del eje X2. Como son paralelos, esta distancia será **81+a_2**. Cabe destacar, al igual que ocurría anteriormente, que puesto que nuestro manipulador es configurable la distancia a_2 puede variar dependiendo el ajuste que se le otorgue, por lo que a la medida inicial se le añade una variable la cual indicará este cambo de configuración.

α2: Ángulo entre los ejes Z1 y Z2, al girar alrededor de X2. Como son paralelos, pero el eje Z2 está colocado en la dirección opuesta al eje Z1 el ángulo para esta articulación es de **180**.

Articulación	θ	d	a	α
2	$*\theta_2$	27	81+a_2	180

Tabla 5. Parámetros de Denavit- Hartenberg para la segunda articulación

Tercera fila de la tabla (transformación del sistema de coordenadas 2 al 3):

θ3: Ángulo entre los ejes X2 y X3 al girar alrededor del eje Z2. Puesto que es una variable de articulación prismática no existe ángulo de rotación sino un desplazamiento lineal, por lo que se le asignara un valor de **cero** a este parámetro.

d3 : Distancia entre el origen del sistema de coordenadas 2 y la intersección entre Z2 y X3 medida a lo largo del eje Z2. En este caso el punto de intersección es el origen del sistema 3. Por lo tanto, la distancia es **20**.

a3 : Distancia entre los ejes Z2 y Z3, medida a lo largo del eje X3. Como se cruzan, esa distancia es **cero**.

α3 : Ángulo entre los ejes Z2 y Z3 , al girar alrededor de X2. Como son paralelos, ese ángulo es **cero**.

Articulación	θ	d	a	α
3	0	140+d_3	0	0

Tabla 6. Parámetros de Denavit- Hartenberg para la tercera articulación

Última fila de la tabla (transformación del sistema de coordenadas 3 al 4):

θ4: Ángulo entre los ejes X1 y X2 al girar alrededor del eje Z1. Puesto que es una variable de articulación rotacional se asignará la correspondiente variable θ.

d4: Distancia entre el origen del sistema de coordenadas 3 y la intersección entre Z3 y X4 medida a lo largo del eje Z3. En este caso el punto de intersección es el origen del sistema 4. Por lo tanto, la distancia es **80**.

a4: Distancia entre los ejes Z3 y Z4, medida a lo largo del eje X4. Como se cruzan, esa distancia es **cero**.

α4: Ángulo entre los ejes Z3 y Z4, al girar alrededor de X3. Como son paralelos, ese ángulo es **cero**.

Articulación	θ	d	a	α
4	$*\theta_4$	59	0	0

Tabla 7. Parámetros de Denavit- Hartenberg para la última articulación

Una vez obtenidos todos los parámetros de la tabla, el cálculo de las relaciones entre los eslabones consecutivos del robot es inmediato, ya que vienen dadas por las matrices ${}^{i-1}A_i$ que se calculan según la expresión general:

$$ {}^{i-1}A_i = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \cdot \sin\theta_i & \sin\alpha_i \cdot \sin\theta_i & a_i \cdot \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cdot \cos\theta_i & -\sin\alpha_i \cdot \cos\theta_i & a_i \cdot \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

En la tabla 8 se muestran los parámetros de Denavit-Hartenberg calculados:

Articulación	θ	d	a	α
1	$*\theta_1$	257	78.5+a_1	0
2	$*\theta_2$	27	80+a_2	180
3	0	140+d_3	0	0
4	$*\theta_4$	59	0	0

Tabla 8. Parámetros de Denavit-Hartenberg

A continuación, se calculan las matrices de transformación para cada uno de los sistemas de coordenadas.

$$ {}^0A_1 = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & (78.5+a_1)\cdot\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & (78.5+a_1)\cdot\sin\theta_1 \\ 0 & 0 & 1 & 253 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad {}^1A_2 = \begin{pmatrix} \cos\theta_2 & \sin\theta_2 & 0 & (80+a_2)\cdot\cos\theta_2 \\ \sin\theta_2 & -\cos\theta_2 & 0 & (80+a_2)\cdot\sin\theta_2 \\ 0 & 0 & -1 & 20 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^2A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 20+d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad {}^3A_4 = \begin{pmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 80 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Finalmente, la matriz de transformación resultante es el producto de todas las matrices anteriores.

$$ T = {}^0A_n = {}^0A_1 \cdot {}^1A_2 \cdot {}^2A_3 \cdot {}^3A_4 $$

Esta matriz nos dará la posición y orientación del extremo con respecto a la base a partir de las coordenadas de las articulaciones $\theta_1, \theta_2, d_3, \theta_4$.

En las siguientes matrices de transformación se utilizará la nomenclatura C = cos() y S = sin() ya que estas matrices serán cada vez más grandes y será más fácil y práctico nombrarlas como tales.

$$ {}^0A_2 = \begin{pmatrix} \mathrm{C}\,\theta_{1,2} - \mathrm{S}\,\theta_{1,2} & \mathrm{C}\,\theta_1\cdot\mathrm{S}\,\theta_2 + \mathrm{S}\,\theta_1\cdot\mathrm{C}\,\theta_2 & 0 & a_2\cdot\mathrm{C}\,\theta_{1,2} - a_2\cdot\mathrm{S}\,\theta_{1,2} + a_1\cdot\mathrm{C}\,\theta_1 \\ \mathrm{C}\,\theta_1\cdot\mathrm{S}\,\theta_2 + \mathrm{S}\,\theta_1\cdot\mathrm{C}\,\theta_2 & \mathrm{S}\,\theta_{1,2} - \mathrm{C}\,\theta_{1,2} & 0 & a_2\cdot\mathrm{C}\,\theta_1\cdot\mathrm{S}\,\theta_2 + a_2\cdot\mathrm{S}\,\theta_1\cdot\mathrm{C}\,\theta_2 + a_1\cdot\mathrm{S}\,\theta_1 \\ 0 & 0 & -1 & d_1+d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Se simplificarán las matrices a lo largo de la determinación de estas aplicando una serie de identidades trigonométricas, de forma que se sea más sencilla el cálculo de estas.

$$\sin(\alpha \pm \beta) = \sin(\alpha)\cos(\beta) \pm \cos(\alpha)\sin(\beta)\,; \quad \cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \mp \sin(\alpha)\sin(\beta)$$

$${}^{0}A_2 = \begin{pmatrix} \mathrm{C}\,\theta_{1+2} & \mathrm{S}\,\theta_{1+2} & 0 & a_2 \cdot \mathrm{C}\,\theta_{1+2} + a_1 \cdot \mathrm{C}\,\theta_1 \\ \mathrm{S}\,\theta_{1+2} & -\mathrm{C}\,\theta_{1+2} & 0 & a_2 \cdot \mathrm{S}\,\theta_{1+2} + a_1 \cdot \mathrm{S}\,\theta_1 \\ 0 & 0 & -1 & d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{0}A_3 = \begin{pmatrix} \mathrm{C}\,\theta_{1+2} & \mathrm{S}\,\theta_{1+2} & 0 & a_2 \cdot \mathrm{C}\,\theta_{1+2} + a_1 \cdot \mathrm{C}\,\theta_1 \\ \mathrm{S}\,\theta_{1+2} & -\mathrm{C}\,\theta_{1+2} & 0 & a_2 \cdot \mathrm{S}\,\theta_{1+2} + a_1 \cdot \mathrm{S}\,\theta_1 \\ 0 & 0 & -1 & d_1 + d_2 - d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^{0}A_4$$

$$= \begin{pmatrix} \mathrm{C}\,\theta_{1+2,4} + \mathrm{S}\,\theta_{1+2,4} & -\mathrm{C}\,\theta_{1+2} \cdot S\theta_4 + \mathrm{S}\,\theta_{1+2} \cdot C\theta_4 & 0 & a_2 \cdot \mathrm{C}\,\theta_{1+2} + a_1 \cdot \mathrm{C}\,\theta_1 \\ \mathrm{S}\,\theta_{1+2} \cdot C\theta_4 - \mathrm{C}\,\theta_{1+2} \cdot S\theta_4 & -\mathrm{C}\,\theta_{1+2,4} - \mathrm{S}\,\theta_{1+2,4} & 0 & a_2 \cdot \mathrm{S}\,\theta_{1+2} + a_1 \cdot \mathrm{S}\,\theta_1 \\ 0 & 0 & -1 & d_1 + d_2 - d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Al igual que se hizo anteriormente, se aplican unas identidades trigonométricas sobre la suma de ángulos y se obtiene la matriz de transformación homogénea total:

$${}^{0}A_4 = \begin{pmatrix} \mathrm{Cos}(\theta_1 + \theta_2 - \theta_4) & \mathrm{Sin}(\theta_1 + \theta_2 - \theta_4) & 0 & a_2 \cdot \mathrm{Cos}(\theta_1 + \theta_2) + a_1 \cdot \mathrm{Cos}\,\theta_1 \\ \mathrm{Sin}(\theta_1 + \theta_2 - \theta_4) & -\mathrm{Cos}(\theta_1 + \theta_2 - \theta_4) & 0 & a_2 \cdot \mathrm{Sin}(\theta_1 + \theta_2) + a_1 \cdot \mathrm{Sin}\,\theta_1 \\ 0 & 0 & -1 & d_1 + d_2 - d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5.2. CINEMÁTICA INVERSA

El problema de la cinemática inversa consiste en encontrar los valores que tienen las coordenadas de las articulaciones del robot $\theta_1, \theta_2, \theta_3, \ldots, \theta_n$. A diferencia del problema de la cinemática directa, calcular la cinemática inversa no es fácil ya que implica resolver una serie de ecuaciones (normalmente no lineales) que dependen en gran medida de la configuración del robot, y que además tienen diferentes soluciones para resolver el problema.

Existen diversos métodos para resolver esta, en nuestro caso se realizará a través del método geométrico puesto que el manipulador posee 4 grados de libertad siendo 3 de estos coplanares, lo que facilita la obtención de las relaciones trigonométricas. Además, el grado de libertad final no estará involucrado en la obtención del modelo cinemático inverso, puesto que la rotación del robot no cambia la posición final de la extremidad.

Primero, se desecha el método de matriz de transformación homogénea, ya que dará lugar a sistemas complejos de ecuaciones no lineales, que pueden tener infinitas soluciones. Con 3 grados de libertad coplanares, el método geométrico propuesto es el más adecuado. Obviamente, este procedimiento es el más factible para el robot considerado, tanto por esta propiedad geométrica como porque, como ya se mencionó, en el problema de la cinemática inversa se eliminará el último grado de libertad. Luego, se obtiene la cinemática inversa a partir de las coordenadas del elemento terminal, así como el ángulo de ataque de la herramienta.

La ilustración 46 muestra la secuencia cinemática del brazo en una posición dada.

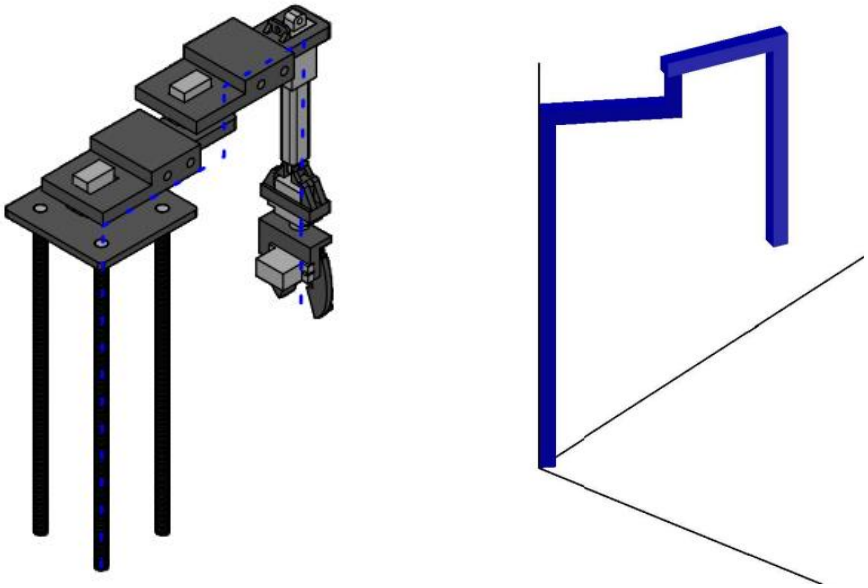


Ilustración 55. Secuencia cinemática del brazo en una posición dada

Para resolver el problema cinemático inverso del robot SCARA se utilizarán planos de proyección que contengan los 4 movimientos de las 4 articulaciones. Se comenzará tomando el plano X_0 e Y_0, de forma que se vea proyectada los eslabones 2 y 3 y el punto de la pinza con componentes R_x y R_y.

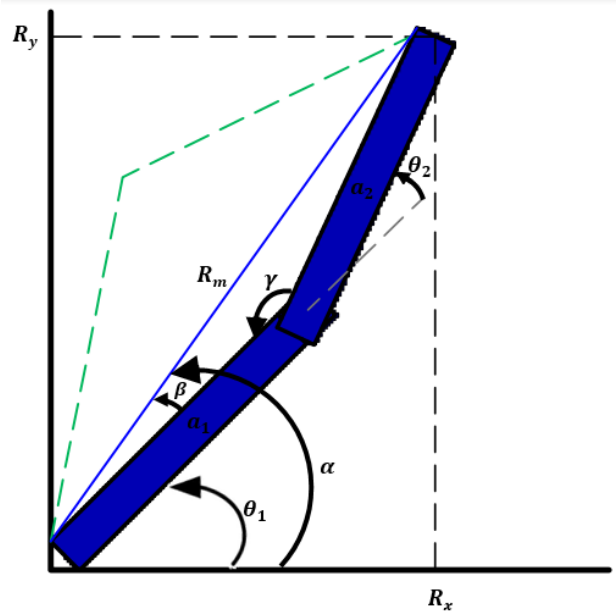


Ilustración 56. Proyección del manipulador sobre los ejes

El primer ángulo que se puede determinar en este plano formado por X_0 e Y_0 es θ_2, ya que para determinar θ_1 nos va a ser necesario resolver este con anterioridad. A la hora de determinar los ángulos θ_1 y θ_2 se debe tener en cuenta que no poseen una solución única, sino que dependiendo la configuración del manipulador estos resultaran en un valor distinto. Se realizarán los cálculos sobre la configuración mostrada en la ilustración 47 teniéndose en cuenta el segundo posible resultado dependiendo si el codo del manipulador se encuentra hacia la derecha (trazado azul) o la izquierda (trazado verde).

Aplicando métodos geométricos y utilizando el triángulo formado por los eslabones a_1, a_2 y el lado $\mathrm{R_m}$, como se puede observar en la ilustración 47, se determina dicho ángulo aplicando directamente el teorema del coseno.

$$\boldsymbol{{R_m}^2 = {R_x}^2 + {R_y}^2 = {a_1}^2 + {a_2}^2 - 2 \cdot a_1 \cdot a_2 \cdot cos(\gamma)}$$

Donde $\boldsymbol{\gamma = 180 - \theta_2}$

$$\boldsymbol{{R_x}^2 + {R_y}^2 = {a_1}^2 + {a_2}^2 - 2 \cdot a_1 \cdot a_2 \cdot cos(180 - \theta_2)}$$

Se aplica la identidad geométrica del coseno cuando a este se le resta un ángulo $\pm\pi$.

$$\boldsymbol{\cos(\pi \pm \theta_2) = -cos(\theta_2)}$$

$$\boldsymbol{{R_x}^2 + {R_y}^2 = {a_1}^2 + {a_2}^2 + 2 \cdot a_1 \cdot a_2 \cdot cos(\theta_2)}$$

$$\boldsymbol{cos(\theta_2) = D = \frac{{R_x}^2 + {R_y}^2 - {a_1}^2 - {a_2}^2}{2 \cdot a_1 \cdot a_2}}$$

Para determinar el valor de este ángulo se podría utilizar directamente la relación hallada anteriormente, pero la función arco coseno tiene ciertas limitaciones por lo que se utilizara la función arco tangente de precisión 2 la cual tiene en cuenta todos los cuadrantes del plano.

Dado que $\boldsymbol{sin(\theta) = \sqrt{1 - cos(\theta)^2}}$ → $\boldsymbol{tan(\theta) = \frac{\sqrt{1-cos(\theta)^2}}{cos(\theta)}}$

$$\boldsymbol{\theta_2 = atan2\left(\frac{\pm\sqrt{1 - D^2}}{D}\right)}$$

Se observa como la ecuación obtenida en el numerador de dicha fracción se escriben ambos signos negativo y positivo, esto se debe a que eligiendo uno de estos dos signos se obtendrá el resultado de las dos posibles configuraciones. Siendo la solución positiva la que se ha dibujado (codo hacia la derecha) y la negativa la trazada por una línea discontinua verde (codo hacia la izquierda).

Una vez determinado el valor de la segunda articulación, se procede a determinar el de la primera.

Este ángulo viene dado por la diferencia entre los ángulos α y β, por lo que aplicando métodos geométricos y utilizando los triángulos formados por eslabones a_1 y a_2 y el lado R_m y el formado por el punto de la herramienta R_x y R_y como se puede observar en la ilustración 47.

Utilizando la función trigonométrica del arco tangente en los triángulos anteriormente mencionados se obtiene:

$$\boldsymbol{\alpha = arcTg\left(\frac{R_y}{R_x}\right)}$$

Para determinar el ángulo de β se buscará la una primera distancia que depende solo del ángulo θ_2 a través del seno y multiplicado por la distancia del eslabón 2; dividido entre la distancia proyectada en el eje X_1, el cual será la distancia al eslabón a_1 mas la distancia restante al punto donde se encuentra la herramienta.

$$\boldsymbol{\beta = arcTg\left(\frac{a_2 \cdot \sin(\theta_2)}{a_1 + a_2 \cdot \cos(\theta_2)}\right)}$$

$$\boldsymbol{\theta_1 = \alpha - \beta}$$

Se han obtenido dos soluciones para el ángulo θ_2, las cuales sustituidas sobre θ_1 proporcionan otras dos soluciones a este ángulo las cuales se denominan para codo hacia la derecha y codo hacia la izquierda.

Para el caso de codo arriba se obtendría un valor de θ_1 el cual se define como:

$$\boldsymbol{\theta_1 = \alpha + \beta}$$

De esta forma se obtienen los valores para las dos primeras articulaciones. Para calcular la articulación traslacional d_3 se escoge un plano de proyección el cual contenga dicha articulación, en este caso se expone el plano formado por los ejes Z_0 (donde se proyecta el movimiento d_3) y el X_0.

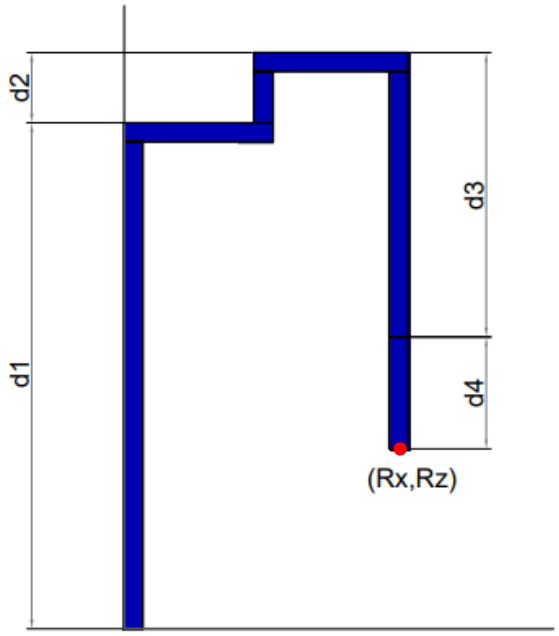


Ilustración 57. Proyección del plano formado por los ejes Z_0 y X_0

Se tendrían las coordenadas de la herramienta (R_x, R_z) donde R_z directamente, si se observa la matriz de transformación homogénea, es la cantidad de $d_1 + d_2 - d_3 - d_4$. En la ilustración 48 se puede observar dichas cotas y el punto de coordenadas de la herramienta.

$$R_Z = d_1 + d_2 - d_3 - d_4$$

De donde se despeja el valor de la articulación prismática.

$$d_3 = d_1 + d_2 - d_4 - R_Z$$

Debe tenerse en cuenta que existe un offset en la articulación 3 exactamente de 140 mm, por lo que a la hora calcular el parámetro será necesario incluirlo para que se calcule adecuadamente su valor.

$$d_3 = d_1 + d_2 - d_4 - R_Z - 140$$

Con esta última ecuación quedaría resuelto el problema cinemático para las tres primeras articulaciones de forma que el ultimo parámetro por calcular sea el de la articulación 4 correspondiente a la rotación de la herramienta.

El valor de la cuarta articulación θ_4 puede determinarse mediante dos métodos. Uno de ellos seria definiendo la articulación θ_4 como el giro, en este caso, contrario a θ_1 y θ_2. La otra seria equiparando la matriz de transformación homogénea de la articulación a una de elementos, de las cuales mediante relaciones trigonométricas se obtenga dicho ángulo.

Por lo tanto, comenzando por el primer método, si se considerándose δ como el resultado de todas las articulaciones rotacionales θ_1, θ_2 y θ_4, se puede deducir de la matriz de transformación homogénea dicho ángulo.

La Matriz de Rotación Total adquirida anteriormente nos da esta información, por lo que es importante conocer la posición y orientación final de la herramienta. Para obtener los valores de orientación de la herramienta se necesita postmultiplicar esta matriz a la matriz ${}^0R_3$. De forma que se obtendría:

$$ {}^0R_4 = {}^0R_3 \cdot {}^3R_4 \quad \rightarrow \quad {}^3R_4 = {}^0R_3{}^{-1} \cdot {}^0R_4 $$

Ahora que se conoce la matriz de rotación de las últimas articulaciones es el momento de extraer los ángulos de las articulaciones.

Se puede definir la matriz R_3^4 de esta otra manera:

$$ R_3^4 = T_3^4 $$

Utilizando la matriz de transformación para la cuarta articulación, se obtiene la siguiente matriz:

$$ R_3^4 = \begin{pmatrix} \mathrm{Cos}(\theta_1 + \theta_2 - \theta_4) & \mathrm{Sin}(\theta_1 + \theta_2 - \theta_4) & 0 \\ \mathrm{Sin}(\theta_1 + \theta_2 - \theta_4) & -\mathrm{Cos}(\theta_1 + \theta_2 - \theta_4) & 0 \\ 0 & 0 & -1 \end{pmatrix} $$

Se debe tener presente esta matriz ya que permite ver que la muñeca esférica posee la misma matriz de rotación que los ángulos de Euler en la secuencia z-y-z.

Conociendo las relaciones de ángulos, se puede equiparar la matriz R_3^4 con una matriz de elementos. De forma que se obtengan los últimos ángulos de la articulación utilizando las relaciones trigonométricas entre esos elementos.

$$ R_3^4 = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} $$

Si de esta se puede despejar que:

$$tg(\delta) = tg(\theta_1 + \theta_2 - \theta_4) = \frac{r_{22}}{r_{12}}$$

$$\theta_1 + \theta_2 - \theta_4 = arctg\left(\frac{r_{22}}{r_{12}}\right) \quad \rightarrow \quad \theta_4 = \theta_1 + \theta_2 - arctg\left(\frac{r_{22}}{r_{12}}\right)$$

El valor de este arco tangente es directamente el valor de la orientación deseada, por lo que el resultado de la articulación 4 se puede deducir inmediatamente.

El otro método consiste mediante

Con esto se tendría calculado el valor de todas las articulaciones rotacionales.

6. PROGRAMACIÓN

En este se tratará sobre la programación utilizada. Se comenzará explicando las simulaciones realizadas previas a la implementación del robot, de esta forma se asegura el correcto funcionamiento en el cálculo de los modelos de cinemática inversa, directa y el control cinemático. Posteriormente, se implementarán las funciones que generarán el movimiento del robot.

Para el desarrollo del proyecto se utilizarán los programas Matlab, el cual se usará para realizar las simulaciones; y Arduino, el cual se utilizará para la programación del microcontrolador.

6.1. Matlab

Es un sistema informático digital que proporciona un entorno de desarrollo integrado (IDE) el cual utiliza su propio lenguaje de programación (lenguaje M). Entre sus funciones principales se encuentran el procesamiento de matrices, la representación de funciones y datos, la implementación de algoritmos, la creación de interfaces de usuario y la comunicación con otros dispositivos y con programas en otros lenguajes.

Para asegurar que los resultados obtenidos son los correctos, se comprobará dicha validez a través de la herramienta Matlab. Esta proporciona, entre sus muchas funciones, la capacidad de escribir scripts en su propio lenguaje informático y con un asistente. Mencionado esto, se programan dos scripts diferentes en Matlab para comprobar el correcto desarrollo de los modelos cinemáticos.

6.1.1. Función cinemática directa

Se comienza implementando el modelo cinemático directo. El script se ha programado como una función de forma que a la hora de simularlo sea menos complejo, teniendo únicamente que introducir los valores de los datos de entrada. La función posee tres argumentos de entrada los cuales serán: la tabla de parámetros de Denavit-Hartenberg, una matriz de doble precisión la cual posea los valores de cada articulación (en grados y mm) y un último vector de la misma longitud que el número de articulaciones quien indicará el tipo de articulación en cada caso (Rotacional = 1 y Prismática = 0). Cabe recalcar que, en la tabla de los parámetros, en el valor de los eslabones se debe introducir el valor comprendido entre cada articulación, no como la suma de los propios eslabones, puesto que esto podría llevar a producir algún error llevando incluso al robot a lugares inaccesibles. Esto resulta así puesto que asumimos que las uniones miden más o menos de lo que realmente son, es decir, se estaría diciendo que el vínculo es mayor o menor de lo que es, lo que

provocará errores en el cómputo que pueden ser significativos o no, dependerá del error de la distancia entre las juntas.

Se obtienen las matrices de transformación homogénea a partir de otra función a la cual se le introduce como parámetros de entrada los parámetros de la articulación y devuelve la matriz calculada.

```
function T= DH(theta,d,a,alpha)

   T=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta);
     sind(theta) cosd(theta)*cosd(alpha) -cosd(theta)*sind(alpha) a*sind(theta);
     0 sind(alpha) cosd(alpha) d;
     0 0 0 1];
end
```

Se comienza extrayendo de la matriz de doble precisión el número de pasos de tiempo, así como el número de articulaciones que componen el robot. Se calcula el valor de los límites máximos del espacio de dibujo y una variable la cual servirá para establecer las dimensiones de la herramienta. A partir del número de pasos de tiempo se creará un bucle para cada uno de ellos. Dentro de este se comenzará inicializando la matriz donde se ira calculando la matriz de transformación para cada articulación y por cada paso de tiempo. También se guardará en una variable el tiempo en el que comienza la simulación, para posteriormente poder dejar el tiempo establecido entre dos poses consecutivas. Además, se añade la condición de que, si se establece mas de un paso de tiempo, la representación del robot en el paso anterior al que se encontraría se borre.

```
[ntimes,njoints]=size(in);

maxlim=2.5*max(vertcat(tabla(:,2),tabla(:,3)));

delta = 0.5*tabla(njoints,2);

for t=1:ntimes

    t0 =clock;

    T(:,:,t) = eye(4);

      if t ~= 1

         delete(hlink);

         delete(htool);
```

```
    delete(hproj);

end
```

Posteriormente se creará un bucle que vaya pasando por cada una de las articulaciones. En cada una de ellas se comprobará el tipo de articulación con la que se trabaja, y según esto se llamará a la función DH nombrada anteriormente para realizar el cálculo de la matriz de transformación. A partir del cálculo de esta matriz y con los datos iniciales se dibujará las diferentes partes que componen la estructura del manipulador.

Para ello se crea una condición en la que si se cumple la misma solo se representarán las uniones del manipulador, dejando de esta forma la representación de la herramienta diferenciada de la estructura del manipulador. Mientras el contador sea menor que el número de articulaciones se representaran las uniones del manipulador mediante la función RectFill3() (la cual se explicara en el siguiente apartado), , dependiendo el número del contador en el que nos encontremos se introducirán unos datos de entrada u otros de forma que se represente correctamente las uniones deseadas. Antes de la llamada a dicha función, se calcula para esas articulaciones el punto inicial y final de la unión, los cuales serán de utilidad para representar los mismos.

Por otro lado, para representar la estructura de la herramienta se comenzará determinando la posición final de la misma sin tener en cuenta la pinza. Este se puede hallar multiplicando a la matriz de transformación total con una matriz de 1x4 donde la coordenada z venga definida por el parámetro delta anteriormente mencionado. Una vez determinado este punto se representará el mango de la herramienta de forma que parta desde la posición final hallada hasta la terminación de la tercera unión. Posteriormente, se determinarán los puntos que definen la pinza de la herramienta y se dibujan en función del giro que de la misma. Además, se establecerán los límites máximos del plot.

Cada vez que se llame a la función que ayuda a representar los componentes del robot, se mandara como argumento de salida de esta las representaciones realizadas. De forma que, si se guardan estos datos en variables, posteriormente si se desea visualizar alguna posición diferente en el manipulador, se podrían borrar y representar la nueva configuración del robot sobre el mismo elemento gráfico.

```
for i=1:njoints
  if articulacion(i)==1
```

```
    T = T*DH(tabla(i,1)+in(i),tabla(i,2),tabla(i,3),tabla(i,4));
  elseif articulacion(i)==0
    T = T*DH(tabla(i,1),tabla(i,2)+in(i),tabla(i,3),tabla(i,4));
  end
  if number<njoints(2)
  Pai(number,:)= T(:,:,t)*[-tabla(number,3),0,0,1]';
  Oi(number,:)= T(:,:,t)*[0,0,0,1]';
  if number==1
    hlink(1,:) = RectFill3([0 0 0],[20 0 0],20,tabla(1,2),0)
    hlink(2,:) = RectFill3(Pai(number,:),Oi(number,:),20,20,0)
    hlink(3,:) = RectFill3(Oi(number,:),[Oi(number,1)+20 Oi(number,2)
Oi(number,3)],20,tabla(2,2),0)
  elseif number ==2
    hlink(4,:) = RectFill3(Pai(number,:),Oi(number,:),20,20,0)
  else
    hlink(5,:) = RectFill3(Pai(number,:),[Pai(number,1)+20 Pai(number,2)
Pai(number,3)],20,((tabla(1,2)+tabla(2,2))-Oi(number,3)+20),0)
  end
else
  Pcon= T(:,:,t)*[0,0,-delta,1]';
  htool(1,:) = RectFill3(Pcon,[Pcon(1)+20 Pcon(2) Pcon(3)],20,Pcon(3,1)-T(3,3,t),0)
  P1= T(:,:,t)*[0,delta,-delta,1]';
  P2= T(:,:,t)*[0,-delta,-delta,1]';
  P3= T(:,:,t)*[0,delta,0,1]';
  P4= T(:,:,t)*[0,-delta,0,1]';
  if in(t,1)==0 && in(t,2)==0 && in(t,4)==0
        htool(2,:) = RectFill3(P1,[P1(1,1)+20 P1(2,1) P1(3,1)],P2(2,1)-P1(2,1)+20,10);
```

```
        elseif in(t,4)~=0
            htool(2,:) = RectFill3(P1,[P2(1,1)+20 P2(2,1) P2(3,1)],20,10);
        else
            htool(2,:) = RectFill3([P1(1,1)+20 P1(2,1) P1(3,1)],P2,20,10);
        end
    htool(3,:) = RectFill3(P4,[P4(1,1)+20 P4(2,1) P4(3,1)],20,P1(3,1)-P4(3,1))
    htool(4,:) = RectFill3(P3,[P3(1,1)+20 P3(2,1) P3(3,1)],20,delta)
end
axis([-maxlim maxlim -maxlim maxlim 0 maxlim]);
```

Posteriormente, se saldrá del bucle y se hallará el punto 3D de la herramienta el cual nos servirá si hemos establecido dos puntos consecutivos. Mediante este y el punto anterior de la herramienta 3D, se trazará una línea la cual una dicha sucesión de puntos de forma que se muestre la trayectoria visualmente.

Finalmente se devuelven como valores de salida la posición y orientación de la herramienta para ese instante. Se pausará el programa el tiempo establecido entre cada paso.

```
P3D=T(1:3,4,t);
if not(t==1)
  plot3([P3D(1) P3Dold(1)],[P3D(2) P3Dold(2)],[P3D(3) P3Dold(3)],'k',
'LineWidth',5);
end
P3Dold=P3D;
posicion = [T(1,4);T(2,4);T(3,4)];
orientacion = [T(1,1:3);T(2,1:3);T(3,1:3)];
if t<ntimes
      pause(tiempo-etime(clock,t0));
   end
```

6.1.2. Función RectFill3()

En la implementación de la función de la cinemática directa se hace referencia a la función RectFill3(), la cual permite dibujar las caras que componen las uniones del robot. La función posee cinco argumentos de entrada: el punto inicial y final de la unión, el ancho y alto de la misma y un parámetro el cual permitirá dibujar de forma distinta la pinza de la herramienta. Devolverá como argumento de salida las representaciones realizadas, de forma que se puedan almacenar en variables.

Se han dibujado las uniones que componen la estructura del manipulador basándose en las caras de un cubo y sus esquinas. Se han establecido los puntos de cada vértice del cubo y se han modificado de forma que se obtengan rectángulos en la posición y orientación deseada. En los dos casos se comienza estableciendo las coordenadas x-y-z de los vértices de la estructura, dependiendo en qué modo se encuentre la definición será diferente. Posteriormente se han establecido 6 índices en los cuales se establecerán el número de los vértices, como se pude observar en la ilustración 55, que forman cada una de las caras del rectángulo. Finalmente se creará un bucle el cual recorra cada uno de los índices en las tres coordenadas estableciendo las caras de la estructura mediante la función fill3() de Matlab.

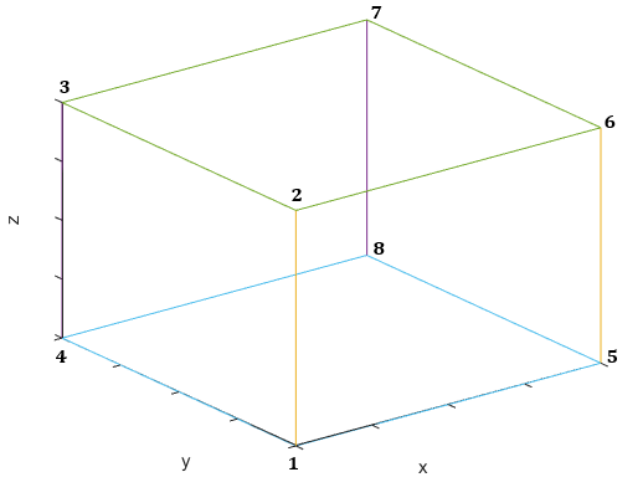


Ilustración 58. Cubo numerando sus vértices

```
x = [Pai(1) Pai(1)     Pai(1)      Pai(1)      Oi(1) Oi(1)     Oi(1)      Oi(1)];

y = [Pai(2) Pai(2)     Pai(2)+ancho Pai(2)+ancho  Oi(2) Oi(2)     Oi(2)+ancho Oi(2)+ancho];

z = [Pai(3) Pai(3)+alto Pai(3)+alto  Pai(3)       Oi(3) Oi(3)+alto Oi(3)+alto  Oi(3)];

index(1,:) = [1 2 3 4 1];

index(2,:) = [5 6 7 8 5];

index(3,:) = [1 2 6 5 1];

index(4,:) = [4 3 7 8 4];
```

```
    index(5,:) = [2 6 7 3 2];
    index(6,:) = [1 5 8 4 1];
    for k = 1:6
       hlink(k) = fill3(x(index(k,:)), y(index(k,:)), z(index(k,:)),'b');
       hold on;
    end
    xlabel('x')
    ylabel('y')
    zlabel('z')
end
```

6.1.3. Función cinemática inversa

El segundo programa que se utilizará implementa el modelo cinemático inverso, el cual proporcionará los ángulos necesarios para poder llegar a la posición deseada establecida. La función posee cuatro argumentos de entrada los cuales son: la tabla de parámetros de Denavit-Hartenberg, la posición y orientación deseada del TCP y una variable la cual indique el tipo de configuración. Además, la función devolverá como argumento de salida los valores de las articulaciones que se usarán posteriormente en la función de cinemática directa para llevar al TCP a la posición deseada.

Se comienza determinando la matriz de rotación total de forma que se multiplique las matrices de rotación para rotaciones alrededor de los ejes z-y-x. En el caso de nuestro manipulador, el TCP posee una rotación constante sobre el eje x de 180º y en el eje y de 0º, por lo que la rotación que influye en la orientación del TCP es la correspondiente al eje z. A este le introduciremos la orientación deseada del TCP. También se guardarán en variables los datos necesarios de la tabla de parámetros de Denavit-Hartenberg y se calculará el valor del coseno de la segunda articulación el cual será necesario para obtener el valor de las articulaciones 1 y 2.

```
function values = InverseKinematics(tabla,posicion,orientacion,codo)
   R0_4=rotz(orientacion(1))*roty(0)*rotx(180);
   d1 = tabla(1,2);
```

```
d2 = tabla(2,2);

d3 = tabla(3,2);

d4 = tabla(4,2);

a1 = tabla(1,3);

a2 = tabla(2,3);

Rx = posicion(1);

Ry = posicion(2);

Rz = posicion(3);

values = 0;

D = (Rx^2+Ry^2-a1^2-a2^2)/(2*a1*a2);
```

Para obtener el valor de las dos primeras articulaciones hay que tener en cuenta que existen dos posibles configuraciones (codo a la derecha y codo a la izquierda) y que dependiendo cual se elija los valores serán unos u otros. Por esto, mediante el argumento de entrada denominado codo se utilizarán unas ecuaciones u otras previamente desarrolladas en el apartado 5.2.

```
if codo == 1

   values(1) = atan2d(Ry,Rx) - atan2d(a2*sqrt(1-D^2),a1+a2*D);

   values(2) = atan2d(sqrt(1-D^2),D);

else

   values(1) = atan2d(Ry,Rx) + atan2d(a2*sqrt(1-D^2),a1+a2*D);

   values(2) = atan2d(-sqrt(1-D^2),D);
```

Una vez determinados los valores de las dos primeras articulaciones, se pasa a determinar el valor de la tercera articulación. Al estar está definida como un movimiento prismático, únicamente se deberá realizar una operación de sumas y restas entre los valores de las uniones que componen el manipulador y el valor de la posición deseada en el eje z.

Por otro lado, para determinar el valor de la última articulación se calculará la matriz de transformación referida desde el sistema de coordenadas de referencia (base) hasta el sistema de coordenadas de la muñeca. Mediante esta y la matriz de transformación total hallada anteriormente

se determinará la matriz de rotación referida desde el sistema de coordenadas de la muñeca hasta el de la herramienta. A partir de esta, teniendo en cuenta la similitud con la matriz de Euler y utilizando la función de la arcotangente se determina el valor de la última articulación.

```
    values(3) = d1+d2-d3-d4-Rz;

    T0_1=DH(values(1),tabla(1,2),tabla(1,3),tabla(1,4));

    T0_2=T0_1*DH(values(2),tabla(2,2),tabla(2,3),tabla(2,4));

    T0_3=T0_2*DH(tabla(3,1),values(3)+tabla(3,2),tabla(3,3),tabla(3,4));

    R3_4=T0_3(1:3,1:3)\R0_4(1:3,1:3);

    values(4)=atan2d(R3_4(2,1),R3_4(1,1))
end
```

6.1.4. Función control cinemático

Finalmente se implementará la función del control cinemático con la cual se determinarán los caminos que cada articulación debe seguir en el tiempo para lograr un camino dado con limitaciones.

En la ilustración 56 se muestra la estructura del control cinemático esquemáticamente la cual utilizaremos para implementar el código. Se reciben como datos de entrada los proporcionados por el usuario, posteriormente con estos se realiza una interpolación en el espacio cartesiano, mediante el modelo cinemático del robot se generarán los puntos de cruce cartesianos y se hallara la resolución del PCI para cada punto de cruce. Se comprobarán que los valores hallados se encuentren dentro de las especificaciones establecidas, una vez corroborado esto, se realizara la interpolación en el espacio conjunto de los cuales se realizarán las trayectorias conjuntas.

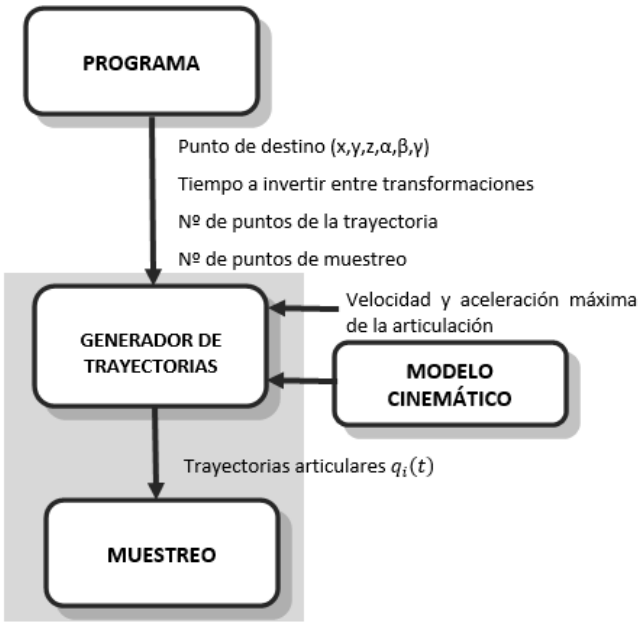


Ilustración 59. Funciones del control cinemático

Dicha función a implementar tiene como variables de entrada: La tabla de parámetros de la convención Denavit-Hartenberg para el robot, un vector que indique la posición y orientación inicial del TCP, un vector que indica la posición y orientación final del TCP, el número de segundos para realizar el movimiento, el número de puntos de cruce en el espacio cartesiano y el número de puntos a evaluar para cada intervalo de interpolación. Por otro lado, la función devolverá como argumento de salida una matriz la cual contenga las trayectorias conjuntas de las articulaciones.

Se comenzará definiendo el número total de articulaciones del robot y el tiempo de cada intervalo. Para definir el número de articulaciones se determinará dicho valor a partir del tamaño de la tabla de parámetros de Denavit-Hartenberg, concretamente el valor de las columnas ya que indicará el número de articulaciones totales. Por otro lado, para determinar el tiempo de cada intervalo se dividirá el tiempo que se ha establecido para cada transformación por el número de puntos que definirán la trayectoria. Hay que tener en cuenta que el primer y el último punto no se tienen en cuenta, por lo que se sumará +1 a este número de puntos.

```
function JTs=KinematicControl(table, Init_pose, Final_pose, Time, NCrossPoints, NSampPoints)

  Njoints = size(table,1);

  inct = Time/(NCrossPoints+1);
```

Una vez obtenidos estos dos valores, se pasará a determinar a la primera parte del código, que consiste en la generación y resolución del ICP para cada uno de los puntos de cruce cartesianos y en el espacio de unión.

En primer lugar, se inicializará la variable en la que se almacenaran los valores de los puntos de cruce. El tamaño de esta variable se fijará en nxm mediante la función ceros(n,m) donde n es el número de puntos definidos en la trayectoria +2 ya que no se ha tenido en cuenta ni el inicio ni el final y m el número de uniones. Posterior a esto, se generará un bucle que pasará por cada uno de los puntos de la trayectoria que se quieren definir, calculando a su vez los puntos de paso en el espacio cartesiano. Dentro de éste se comenzará realizando una interpolación lineal en el espacio cartesiano. La interpolación lineal consiste en mantener constante la velocidad del movimiento para cada dos valores sucesivos. Interpolaremos tanto la posición como la orientación, esta interpolación lineal de dos puntos viene dada por la fórmula anteriormente mencionada en el apartado 2.3.3.

Se utilizará el valor de la posición final y el valor de la posición inicial que se han introducido como variables de entrada y el tiempo entre estos, es decir, el tiempo que se va a utilizar para ese movimiento. Se determinará mediante una única ecuación ya que hemos definido las posiciones como (xi,yi,zi,αi,βi,γi).

Una vez obtenido el punto en el espacio cartesiano, este deberá trasladarse al espacio articular. Esto se realiza mediante la cinemática inversa (previamente resuelta). Los resultados obtenidos serán los puntos de cruce para las trayectorias conjuntas que serán interpoladas mediante splines cúbicos en los que utilizaremos la función InterpoladorCubico(). Antes de proceder a esta interpolación cúbica, se debe comprobar que los valores de los puntos de cruce en el espacio articular están dentro del rango de las especificaciones del robot.

```
Jvals = zeros(NCrossPoints+2, Njoints);

for k = 1:NCrossPoints+2

   pose_k = Init_pose+(Final_pose-Init_pose)/Time*(k-1)*inct;

   Jvals(k,:) =  InverseKinematics(table, pose_k(1:3)',pose_k(4:6),1);

   if Jvals(k,1)>90 || Jvals(k,1)<-90

      fprintf('No se puede alcanzar el ángulo 1 = %0.2f \n',Jvals(k,1))

      return

   elseif Jvals(k,2)>90 || Jvals(k,2)<-90
```

```
        fprintf('No se puede alcanzar el ángulo 1 = %0.2f \n',Jvals(k,1))
        return
    elseif Jvals(k,2)>90 || Jvals(k,2)<-90
        fprintf('No se puede alcanzar el ángulo 2 = %0.2f \n',Jvals(k,2))
        return
    elseif Jvals(k,3)>50 || Jvals(k,3)<0
        fprintf('No se puede alcanzar el ángulo 3 = %0.2f \n',Jvals(k,3))
        return
    elseif Jvals(k,4)>90 || Jvals(k,4)<-90
        fprintf('No se puede alcanzar el ángulo 4 = %0.2f \n',Jvals(k,4))
        return
    end
end
```

Una vez analizados los rangos de los puntos, se procederá a realizar a la interpolación de las trayectorias de cada punto.

Se comenzará por calcular el número de puntos totales necesarios utilizando el número de puntos que definen la trayectoria+1 (para determinar cuántos intervalos habrá) y el número de puntos de cruce en el espacio de unión tras conectar los puntos +1. Si se multiplican entre sí, se obtendrá el valor total de los puntos necesarios. También se inicializará la variable de salida que contendrá todos los caminos de unión. El tamaño de esta variable se fijará en nxm utilizando la función zeros(n,m) donde n es el número de puntos totales necesarios y m es el número de articulaciones del robot.

Para calcular la interpolación de la trayectoria en cada punto se creará un bucle en el que las posiciones resultantes de la interpolación cúbica se sumarán, articulación por articulación, a la variable 'JTs'. Este argumento de salida llenará para cada NPuntos una columna de JTs. Antes de llamar a la función, se deben preparar los datos de entrada que se introducirán en ella. Estas serán una variable que contenga un vector de tiempo con el tiempo total de cada transformación con un paso de tiempo para cada intervalo y la matriz de los puntos de cruce en el espacio de unión.

```
NPoints = (NCrossPoints+1)*(NSampPoints+1)+1;

JTs = zeros(NPoints, Njoints);

for n = 1:NPoints

   DATA = [(0:inct:Time); Jvals(:,n)'];

   JTs(:,n) = InterpoladorCubico(DATA,inct,NSampPoints,n);

end

end
```

6.1.5. Función de interpolación cubica

Esta función interpola una trayectoria conjunta utilizando N puntos de cruce en DATA y el tiempo por intervalo y devuelve NSampPoints en la variable POS.

Cuando hay que ajustar un gran número de datos a una curva suave, la interpolación lagrangiana no es adecuada. Para ello se utiliza el método del spline cúbico, que ajusta un polinomio cúbico en cada intervalo entre dos puntos consecutivos.

El método de interpolación por aproximación polinómica segmentaria consiste en construir splines, es decir, un nuevo polinomio interpolador, cada dos puntos de datos. Como ya se ha dicho, se trabajará con polinomios interpoladores de grado 3. Esto se realiza así ya que proporciona un excelente ajuste a los puntos tabulados y su cálculo no es excesivamente complejo.

Tiene algunas condiciones:

- Cada dos datos tienen que ser un polinomio cúbico.
- Debe pasar por todos los datos de la interpolación.
- Condición de continuidad en los nodos interiores.
- Deben ser derivables en los nodos interiores.
- Continuidad en la primera derivada. Para que sea continua, la primera derivada y la segunda derivada deben ser iguales.

El algoritmo de este método establece que las condiciones para un ajuste spline cúbico son que se pase un conjunto de polinomios cúbicos por los puntos, utilizando un nuevo polinomio cúbico en cada intervalo. Para corresponder a la idea del curvigrama, se requiere que tanto la pendiente como la curvatura sean iguales para el par de polinomios cúbicos que se encuentran en cada punto. El

polinomio cúbico para el intervalo i-ésimo, que cae entre los puntos (x_i, y_i) y (x_{i+1}, y_{i+1}) de la forma

Se comenzará inicializando los vectores donde se introducirán los valores de posición, velocidad, aceleración y tiempo. También se almacenarán dos variables, el número de puntos del tramo y el número de tramos a evaluar.

```
function POS=InterpoladorCubico(DATA,inct,NSampPoints,n)

   POS=[]; SPD=[]; ACL=[]; TIME=[];

   Nints=size(DATA,1)-1;

   Sints=NSampPoints+1;
```

Una vez obtenidos estos valores, se determinan las restricciones de velocidad para cada tramo. Se determinan los coeficientes de las ecuaciones polinómicas utilizando el valor de las velocidades inicial y final. A la hora de calcular estas velocidades, hay que tener en cuenta que algunos valores no se calcularán de la misma manera al estar en zonas de tramos diferentes. Podemos distinguir las zonas de estiramiento como inicial, central y final.

En el tramo inicial se debe tener en cuenta que se parte de una velocidad inicial de 0, ya que en ese momento se encontraría en reposo. Por tanto, para encontrar la velocidad final habrá que tener en cuenta la velocidad actual y la velocidad del siguiente tramo. Se halla la velocidad actual como la diferencia del valor del tiempo del punto siguiente menos el actual entre la diferencia del valor de la articulación n en el punto siguiente menos el actual.

Por otro lado, la velocidad en el siguiente tramo se calculará como la diferencia del valor del tiempo de los dos puntos siguientes menos el actual entre la diferencia del valor de la articulación n en los dos puntos siguientes menos el actual.

$$V_c = \frac{t_{i+1} - t_i}{Jval_{i+1} - Jval_i} \qquad V_n = \frac{t_{i+2} - t_{i+1}}{Jval_{i+2} - Jval_{i+1}}$$

Para encontrar la velocidad final, se comprobará el signo de las velocidades determinadas. Utilizando el comando sign() este nos devolverá un valor negativo o positivo (-1 o 1) o incluso 0 dependiendo del valor que estemos analizando.

La pendiente es la velocidad. Si se tienen cambios en el signo de la velocidad, se asignará a la velocidad final como 0, ya que se cambiará el signo de negativo a positivo o de positivo a negativo,

ya que recordemos que la velocidad es la pendiente entre los dos puntos. En caso contrario, se promediarán las dos pendientes (velocidades) para conocer la velocidad final en ese punto.

$$V_f = \frac{V_c + V_n}{2}$$

En los tramos centrales se deberá calcular la velocidad del tramo anterior ya que ésta se utilizará para encontrar la velocidad inicial. Así mismo, como se hizo anteriormente también se calculará la velocidad en el tramo actual y en el siguiente. Se determinará la velocidad anterior como la diferencia del valor del tiempo del punto actual menos el anterior entre la diferencia del valor de la junta n en el punto actual menos el anterior.

$$V_p = \frac{t_i - t_{i-1}}{Jval_i - Jval_{i-1}}$$

Al igual que se realizó en el tramo inicial, se deberá comprobar el signo de las velocidades (pendientes negativas o positivas) y en función de ello asignar un valor. Al igual que con la velocidad final, si para determinar la velocidad inicial hay un cambio de signo en las pendientes (velocidades) el valor de ésta será 0, en caso contrario será la media entre la velocidad anterior y la actual.

$$V_i = \frac{V_c + V_p}{2} \quad V_f = \frac{V_c + V_n}{2}$$

Por último, en el tramo final se debe tener en cuenta que la velocidad final será 0. Se calculará la velocidad inicial como se ha explicado anteriormente y se comprobará el valor del signo de la pendiente (velocidad).

```
for j=1:Nints
    if j==1
        Vini=0;
        Vcurr=(DATA(j+1,2)-DATA(j,2))/(DATA(j+1,1)-DATA(j,1));
        Vnext=(DATA(j+2,2)-DATA(j+1,2))/(DATA(j+2,1)-DATA(j+1,1));
        if sign(Vcurr)==sign(Vnext)
            Vfin=(Vcurr+Vnext)/2;
        else
            Vfin=0;
        end
```

```
    elseif j<Nints
        Vprev=(DATA(j,2)-DATA(j-1,2))/(DATA(j,1)-DATA(j-1,1));
        Vcurr=(DATA(j+1,2)-DATA(j,2))/(DATA(j+1,1)-DATA(j,1));
        Vnext=(DATA(j+2,2)-DATA(j+1,2))/(DATA(j+2,1)-DATA(j+1,1));
        if sign(Vcurr)==sign(Vprev)
            Vini=(Vcurr+Vprev)/2;
        else
            Vini=0;
        end
        if sign(Vcurr)==sign(Vnext)
            Vfin=(Vcurr+Vnext)/2;
        else
            Vfin=0;
        end
    else
        Vfin=0;
        Vprev=(DATA(j,2)-DATA(j-1,2))/(DATA(j,1)-DATA(j-1,1));
        Vcurr=(DATA(j+1,2)-DATA(j,2))/(DATA(j+1,1)-DATA(j,1));
        if sign(Vcurr)==sign(Vprev)
            Vini=(Vcurr+Vprev)/2;
        else
            Vini=0;
        end
    end
```

```
    else
        Vfin=0;
        Vprev=(DATA(j,2)-DATA(j-1,2))/(DATA(j,1)-DATA(j-1,1));
        Vcurr=(DATA(j+1,2)-DATA(j,2))/(DATA(j+1,1)-DATA(j,1));
        if sign(Vcurr)==sign(Vprev)
            Vini=(Vcurr+Vprev)/2;
        else
            Vini=0;
        end
    end
```

Una vez obtenidas las restricciones, se determinarán los coeficientes del tramo. Antes de estos se almacenarán en una variable el intervalo de tiempo transcurrido en ese tramo, que es la diferencia del tiempo en el siguiente punto menos el actual.

Los coeficientes a0 y a1 se calcularán mediante la posición y la velocidad iniciales. Si se observan las ecuaciones polinómicas que se han calculado anteriormente para la posición y la velocidad inicial, el tiempo es 0, sólo se tendrán estos dos coeficientes que son iguales a las restricciones encontradas.

$$Jvals = a_3 0^3 + a_2 0^2 + a_1 \cdot 0 + a_0$$

$$V_i = 3a_3 \cdot 0^2 + 2a_2 \cdot 0 + a_1$$

Una vez encontrados estos parámetros, se utilizarán utilizar las mismas ecuaciones polinómicas con las restricciones encontradas en la posición y velocidad finales para encontrar los parámetros a2 y a3. En estos el valor del tiempo será el del intervalo de cada sección, por lo que ahora no faltarán los coeficientes que faltaban y se podrán determinar.

$$Jvals = a_3 T^3 + a_2 T^2 + a_1 T + a_0$$

$$V_f = 3a_3 T^2 + 2a_2 T + a_1$$

```
    T=(DATA(j+1,1)-DATA(j,1));
    a0=DATA(j,2);
    a1=Vini;
```

```
A=[T^2 T^3;2*T 3*T^2];
B=[DATA(j+1,2)-a0-a1*T; Vfin-a1];
X=A\B;
a2=X(1); a3=X(2);
```

Una vez hallados los coeficientes, se pueden realizar las interpolaciones cúbicas de la sección. En este caso el valor de t será el tiempo de simulación. Se sumará la interpolación encontrada en cada tramo para que finalmente se obtenga un trazado cúbico (arco) en el que haya continuidad.

```
if j==1
   ts=0:T/Sints:T;
else
   ts=T/Sints:T/Sints:T;
end
POS=[POS,a0+a1*ts+a2*ts.^2+a3*ts.^3];
SPD=[SPD,a1+2*a2*ts+3*a3*ts.^2];
ACL=[ACL,2*a2+6*a3*ts];
TIME=[TIME,ts+DATA(j,1)];
end
```

Una vez encontradas las interpolaciones cúbicas para todos los tramos, se representarán las correspondientes trayectorias de posición, velocidad y aceleración. Para representar la posición, primero se trazarán los datos de la secuencia discreta (el valor de la posición de los N valores que definen la trayectoria) utilizando la función stem(). Tanto para la velocidad como para la aceleración encontraremos las interpolaciones por otro método, haciendo la derivada de la posición y de la velocidad respectivamente. En cada caso pintaremos las interpolaciones encontradas de las dos maneras.

```
figure;
  subplot(3,1,1);
  stem(DATA(:,1),DATA(:,2)); grid on; hold on;
```

```
    plot(TIME,POS,'.g');
    xlabel('Time in secs')
    ylabel('Degrees');
    title(['Angular Position of the Joint ',num2str(n)]);
    subplot(3,1,2);
    SPDn=[0 diff(POS)/(inct/(NSampPoints+1))];
    plot(TIME,SPDn,'b'); hold on;
    plot(TIME,SPD,'g'); grid on;
    xlabel('Time in secs')
    ylabel('Degrees/sec');
    title(['Angular Speed of the Joint ',num2str(n)]);
    subplot(3,1,3);
    ACLn=diff([SPDn,0])/(inct/(NSampPoints+1));
    plot(TIME,ACLn,'r'); hold on;
    plot(TIME,ACL,'m'); grid on;
    xlabel('Time in secs')
    ylabel('Degrees/seg^2');
    title(['Angular Acceleration of the Joint ',num2str(n)]);
end
```

6.1.6. SimuladorSCARA

Con objeto de poder acercar el simulador a la ejecución real del robot manipulador desarrollado anteriormente, se crea una función la cual permitirá al usuario introducir los datos necesarios del robot para realizar la simulación.

Dicha función no posee ni variables de entrada ni de salida, de forma que únicamente se tenga que escribir el nombre de la función en la ventana de comandos de Matlab para que comience a funcionar la simulación.

Se comenzará pidiendo al usuario que introduzca el número de puntos de cruce, una vez introducidos estos se comprobará que están dentro del rango valido para la simulación. Sino se encuentra dentro de este rango se enviará por pantalla un mensaje de error y se acabará la simulación. Cuando esto sea válido, se preguntará y comprobara también el número de puntos de muestreo. Si el valor dado no se encuentra dentro del rango especificado se enviará por pantalla un mensaje de error y se acabará la simulación. Cuando este sea válido, se pedirán los datos restantes siendo estos la duración de la trayectoria, la configuración de la estructura manipulador (codo hacia la derecha o la izquierda), el punto final de la trayectoria y el modo en el que se opera. Recordemos que, puesto que el manipulador posee eslabones configurables, este puede tener 4 diferentes modos de funcionamiento.

Una vez obtenidos todas las variables de entrada necesarias, se comprobará los valores de la variable codo y modo, de forma que, si no se cumple con los posibles valores, se enviará por pantalla un mensaje de error y se acabará la simulación.

```
function SimuladorSCARA()
 disp('Rellena los datos necesarios')
 NCrossPoints = input('Numero de puntos de cruce: ') ;
 if ((NCrossPoints>6)||(NCrossPoints<0))
    disp('/***** ERROR ****/')
    disp('/*DATO NO VALIDO*/')
 else
    NSampPoints = input('Numero de puntos de muestreo: ');
    if ((NSampPoints>=5)||(NSampPoints<0))
      disp('/***** ERROR ****/')
      disp('/*DATO NO VALIDO*/')
    else
      Time = input('Tiempo que dura la trayectoria: ');
      disp('Configuracion estructura manipulador')
      codo = input('Codo derecha (1) - Codo izquierda (0): ');
```

```
disp('Punto final de la trayectoria')
a = input('Coordenada X: ');
b = input('Coordenada Y: ');
c = input('Coordenada Z: ');
d = input('Coordenada alpha: ');
disp('Modo funcionamiento:');
disp('(1)Led Verde - Modo 1: Eslabones cortos');
disp('(2)Led Azul - Modo 2: Eslabones largos');
disp('(3)Led Amarillo - Modo 3: Eslabon1 corto - Eslabon2 largo');
disp('(4)Led Blanco - Modo 4: Eslabon1 largo - Eslabon2 corto');
modo = input('Modo: ');
if ((modo<1)||(modo>4))
   disp('/************** ERROR **************/')
   dips('/*MODO DE FUNCIONAMIENTO INCORRECTO*/')
elseif ((codo<0)||(codo>1))
   disp('/************** ERROR **************/')
   dips('/*CONFIGURACION DEL ROBOT INCORRECTO*/')
      else
```

Una vez hechas las comprobaciones necesarias, dependiendo el modo de configuración que se haya elegido se asignaran unos parámetros u otros a la tabla de parámetros de Denavit-Hartenberg. Posteriormente se llama a la función del control cinemático guardando en una variable los datos de salida, puesto que estos serán necesarios para observar la trayectoria del manipulador y comprobar su posición final. Dicha variable se introducirá como variable de entrada de la función de cinemática directa, así como esta devolverá la posición y orientación final del manipulador.

```
if modo==1
      parametro1 = 78.5;
      parametro2 = 81;
```

```
elseif modo ==3
            parametro1 = 78.5;
            parametro2 = 100;
        elseif modo ==4
            parametro1 = 97.5;
            parametro2 = 81;
        else
        end
        tabla = [0 257 parametro1 0; 0 27 parametro2 180; 0 140 0 0; 0 59 0 0];
        Init_pose = [158.5 0 85 0 0 180];
        Final_pose= [a b c d 0 180];
        JTs=KinematicControl(tabla, Init_pose, Final_pose, Time, NCrossPoints,
NSampPoints,codo);
        articulacion = [1 1 0 1];
        [posicion, orientacion]=DirectKinematics(tabla,JTs,articulacion,Time/(size(JTs,1)-1))
     end
    end
 end
end
```

6.1.7. Simulación

Finalmente se mostrarán dos ejemplos de la implementación de todos los códigos. En el primero de ellos se utilizarán las funciones creadas por separado y en el segundo se empleará la función creada para realizar la simulación. De esta forma se comprobará que el simulador del manipulador es capaz de realizar una trayectoria más larga que la que fuese posible desde el punto de referencia dado.

Se comenzará simulando la función anteriormente explicada la cual proporcionará los valores de las posiciones de las articulaciones para cada punto de la trayectoria. Luego, estos valores se introducirán en la función DirectKinematics() de forma que se represente tanto la trayectoria como el movimiento del manipulador deseado.

Para el primer ejemplo se han escogido como variables de entrada:

- La tabla de parámetros de Denavit-Hartenberg.
- Posición y orientación inicial del robot en la trayectoria,

  Init_pose = [122.5682 94.4591 85 50 0 180].
- Posición y orientación final en la trayectoria,

  Final_pose= [112.0764 -112.0764 65.0000 -135 0 180].
- Duración entre transformaciones, Time = 8.
- Número de puntos definidos en la trayectoria, NCrossPoints = 10.
- Número de puntos de muestreo, NSampPoints = 4.

Con estas se obtiene a continuación las gráficas de la posición, velocidad y aceleración de cada una de las articulaciones que componen el robot, así como los diferentes ángulos que van tomando los mismos a lo largo de la trayectoria.

JTs

56x4 double

	1	2	3	4
1	25.0000	24.9999	0	-5.3870e-05
2	23.5786	27.0265	0.1309	1.8161
3	19.9874	32.1077	0.4655	6.4006
4	15.2358	38.7453	0.9164	12.4575
5	10.3333	45.4412	1.3964	18.6908
6	6.2894	50.6971	1.8182	23.8047
7	3.1538	54.4151	2.1818	27.7508
8	0.2535	57.5940	2.5455	31.3929
9	-2.4861	60.3807	2.9091	34.8037
	⋮	⋮	⋮	⋮
50	-63.5935	49.9936	17.8182	100.6749
51	-62.1406	44.7589	18.1818	100.8002
52	-59.1309	36.1778	18.6036	99.6809
53	-54.7357	24.6047	19.0836	97.0118
54	-50.1351	12.8540	19.5345	93.8261
55	-46.5093	3.7397	19.8691	91.1571
56	-45.0385	0.0763	20	90.0378

Ilustración 60. Valores de los ángulos tomados por las articulaciones a lo largo de la trayectoria

En la ilustración 57 se muestran los diferentes valores que toma cada una de las articulaciones desde el primer punto de la trayectoria, en este caso se partirá de rotación solo en las dos primeras articulaciones, ambas con un ángulo de 25º; hasta el último punto de la trayectoria, en el cual en

este se desea que exista una rotación en la articulación 1 de -45º, en la articulación 2 de 0º, y en la articulación 4 de 90º, por otro lado, la articulación 3 realizara un desplazamiento de 20mm.

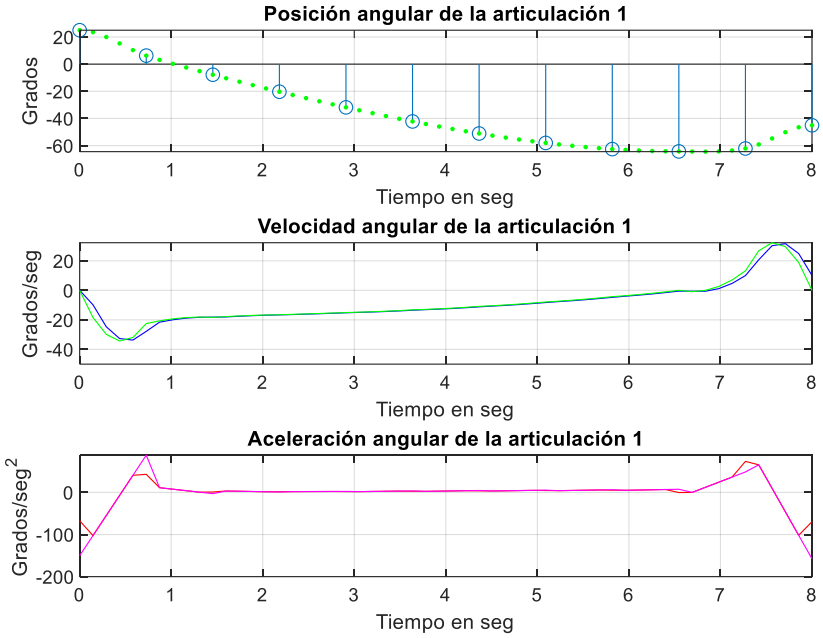


Ilustración 61. Posición, velocidad y aceleración de la primera articulación del primer ejemplo.

En la ilustración 58 se representan la posición, velocidad y aceleración de la primera articulación del robot. En esta se puede observar como la posición angular sigue una trayectoria uniforme desde el primer punto de la trayectoria, en este caso parte la articulación de un valor de 25º; hasta el último, en el que finalmente se posicione en los -45º deseados del punto deseado de destino. Los círculos azules representados en la gráfica de la posición angular indican los puntos que han sido definidos en la trayectoria, en el intervalo entre cada dos de estos puntos se pueden apreciar otros puntos verdes, los cuales representan los puntos a evaluar en cada tramo. Para cada uno de estos puntos se ha calculado su velocidad y aceleración correspondiente, por lo que, a lo largo de la trayectoria se observa la sucesión de estos valores formando así graficas con continuidad uniforme desde que comienza el movimiento hasta que este termina.

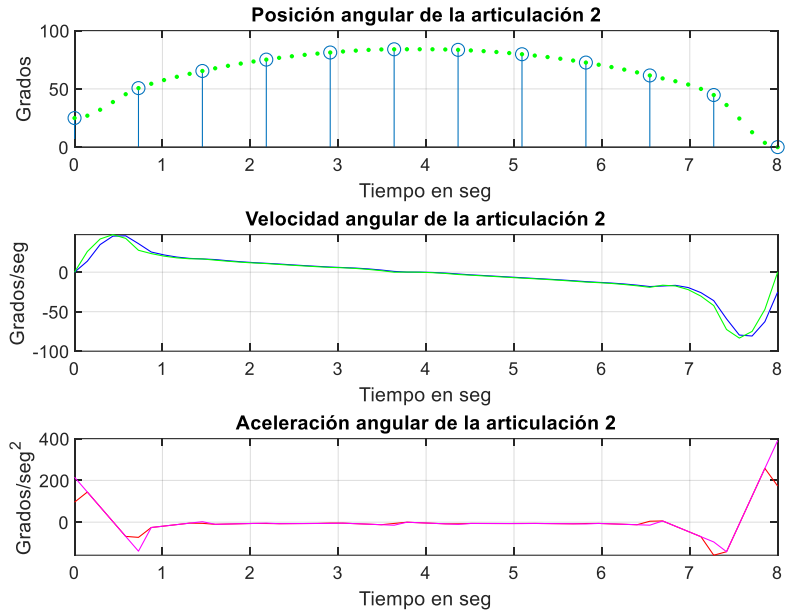


Ilustración 62. Posición, velocidad y aceleración de la segunda articulación del primer ejemplo.

En la ilustración 59 se representan la posición, velocidad y aceleración de la segunda articulación del robot. En esta se puede observar como la posición angular sigue una trayectoria uniforme desde el primer punto de la trayectoria, en este caso parte la articulación de un valor de 25º; hasta el último punto, en el que finalmente se posicione en los 0º deseados del punto de destino deseado. Al igual que en la gráfica anterior se observan las velocidades y aceleraciones correspondientes a cada punto de la trayectoria de forma que se proyecten graficas con continuidad uniforme desde que comienza el movimiento hasta que este termina.

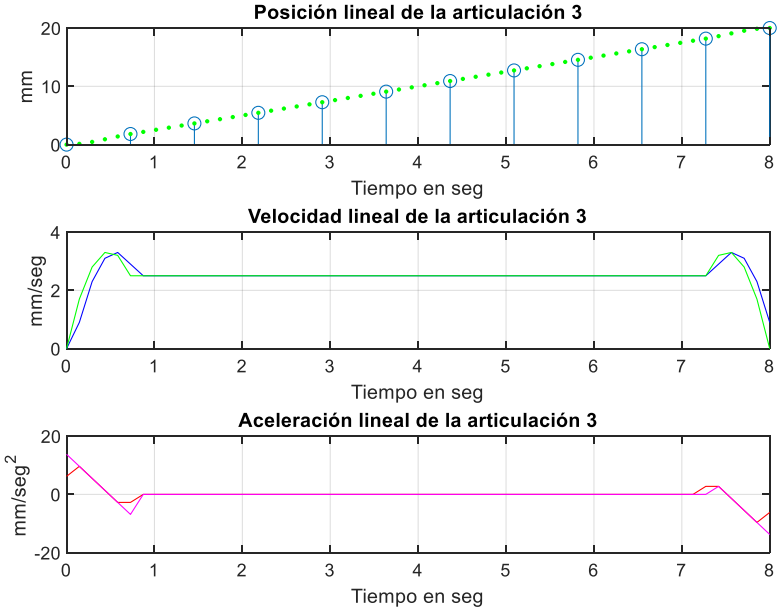


Ilustración 63. Posición, velocidad y aceleración de la tercera articulación del primer ejemplo.

En la ilustración 60 se representan la posición, velocidad y aceleración de la tercera articulación del robot. En esta se puede observar como la posición lineal sigue una trayectoria uniforme desde el primer punto de la trayectoria, en este caso parte la articulación de un valor de 0mm; hasta el último punto, en el que finalmente se posicione en los 20mm deseados del punto de destino deseado. Puesto que la tercera articulación del manipulador permite un movimiento de traslación lineal a lo largo del eje de la articulación, se visualiza la trayectoria de la posición de forma lineal (rampa). Al igual que en las gráficas anteriores se observan las velocidades y aceleraciones correspondientes a cada punto de la trayectoria de forma que se proyecten graficas con continuidad uniforme desde que comienza el movimiento hasta que este termina.

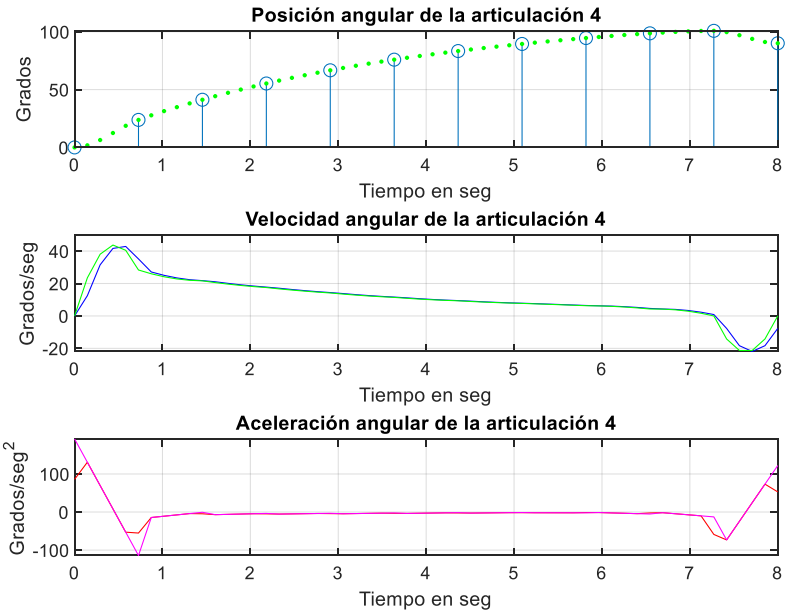


Ilustración 64. Posición, velocidad y aceleración de la cuarta articulación del primer ejemplo.

En la ilustración 61 se representan la posición, velocidad y aceleración de la cuarta articulación del robot. En esta se puede observar como la posición angular sigue una trayectoria uniforme desde el primer punto de la trayectoria, en este caso parte la articulación de un valor de 0º; hasta el último punto, en el que finalmente se posicione en los 90º deseados del punto de destino deseado. Al igual que en las gráficas anteriores se observan las velocidades y aceleraciones correspondientes a cada punto de la trayectoria de forma que se proyecten graficas con continuidad uniforme desde comienza el movimiento hasta que este termina.

Una vez obtenida la matriz que contiene la trayectoria de cada una de las articulaciones, introducimos estos datos como argumento de entrada en la función de la cinemática directa, la cual

nos devolverá la posición y orientación del manipulador en el espacio cartesiano a lo largo de las trayectorias, así como se visualizará el recorrido de las mismas en el manipulador.

```
>> posicion(:,:,1)

ans =

  122.5682
   94.4591
   85.0000
```

```
>> orientacion(:,:,1)

ans =

    50     0   180
```

Ilustración 65. Primer punto de la trayectoria en coordenadas cartesianas

```
>> posicion(:,:,end)

ans =

  112.0764
 -112.0764
   65.0000
```

```
>> orientacion(:,:,end)

ans =

 -135.0000         0  180.0000
```

Ilustración 66. Punto final de la trayectoria en coordenadas cartesianas

En las ilustraciones 62 y 63 se muestra el primer y último punto de la trayectoria, respectivamente, los cuales coinciden con los establecidos inicialmente a la hora de obtener las trayectorias de las articulaciones. De esta forma se comprueba que la trayectoria parte y llega a los puntos deseados, además indica que el generador de trayectorias funciona correctamente.

A continuación, se muestra en la Ilustración 64 el robot en el punto inicial de la trayectoria y en la ilustración 65 el punto final de la misma con la trayectoria dibujada.

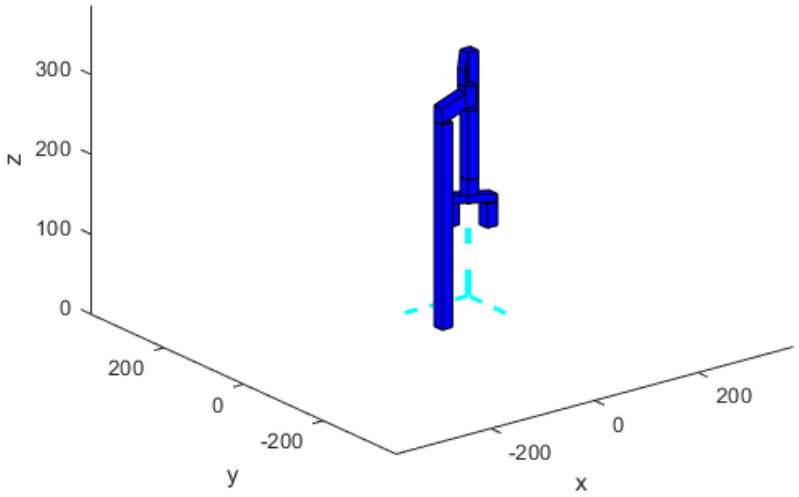


Ilustración 67. Representación del punto inicial de la trayectoria en el manipulador

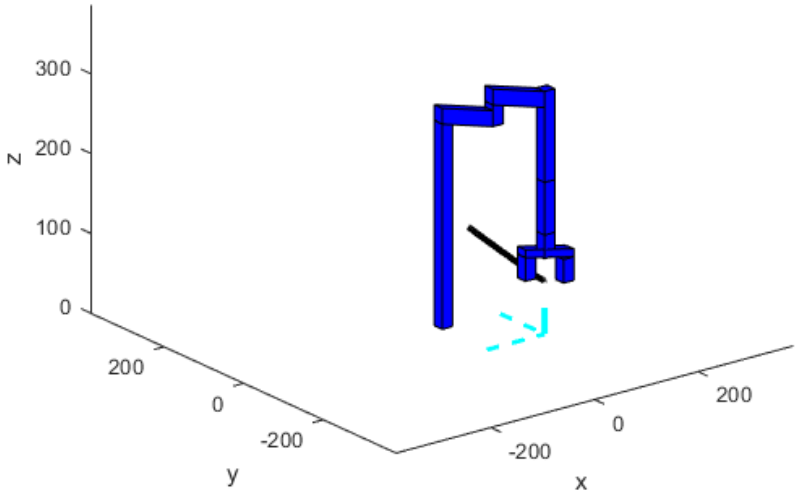


Ilustración 68. Representación del punto final de la trayectoria en el manipulador, así como la propia trayectoria dibujada

Para el segundo ejemplo se han llamado a la función SimuladorSCARA y se han introducido los siguientes datos:

```
>> SimuladorSCARA
Rellena los datos necesarios
Numero de puntos de cruce: 6
Numero de puntos de muestreo: 4
Tiempo que dura la trayectoria: 10
Configuracion estructura manipulador
Codo derecha (1) - Codo izquierda (0): 1
Punto final de la trayectoria
Coordenada X: 118.5
Coordenada Y: 69.282
Coordenada Z: 85
Coordenada alpha: 60
Modo funcionamiento:
(1)Led Verde - Modo 1: Eslabones cortos
(2)Led Azul - Modo 2: Eslabones largos
(3)Led Amarillo - Modo 3: Eslabon1 corto - Eslabon2 largo
(4)Led Blanco - Modo 4: Eslabon1 largo - Eslabon2 corto
Modo: 1
```

Ilustración 69. Datos de entrada de la simulación

Con estas se obtiene a continuación las gráficas de la posición, velocidad y aceleración de cada una de las articulaciones que componen el robot, así como los diferentes ángulos que van tomando los mismos a lo largo de la trayectoria.

JTs

36x4 double

	1	2	3	4
1	0	0	0	0
2	-0.9790	2.4600	0	0.8638
3	-3.3943	8.5794	0	2.9908
4	-6.4631	16.4670	0	5.6839
5	-9.4026	24.2317	0	8.2463
6	-11.4301	29.9826	0	9.9811
7	-12.4431	33.4769	0	10.7721
⋮	⋮	⋮	⋮	⋮
31	-4.4579	59.4005	0	3.5140
32	-3.4732	59.6297	0	2.7394
33	-2.3025	59.7992	0	1.8167
34	-1.1784	59.9141	0	0.9300
35	-0.3334	59.9794	0	0.2632
36	-2.3577e-05	60.0000	0	-5.4836e-12

Ilustración 70. Valores de los ángulos tomados por las articulaciones a lo largo de la trayectoria

En la ilustración 66 se muestran los diferentes valores que toma cada una de las articulaciones desde el primer punto de la trayectoria, en este caso se partirá desde el estado de reposo en el que todas las articulaciones se encuentran en los 0º; hasta el último punto de la trayectoria, en el cual en este se desea que exista una rotación en la articulación 2 de 60º, y en las articulaciones 1 y 4 de 0º, por otro lado, la articulación 3 realizara un desplazamiento de 0mm.

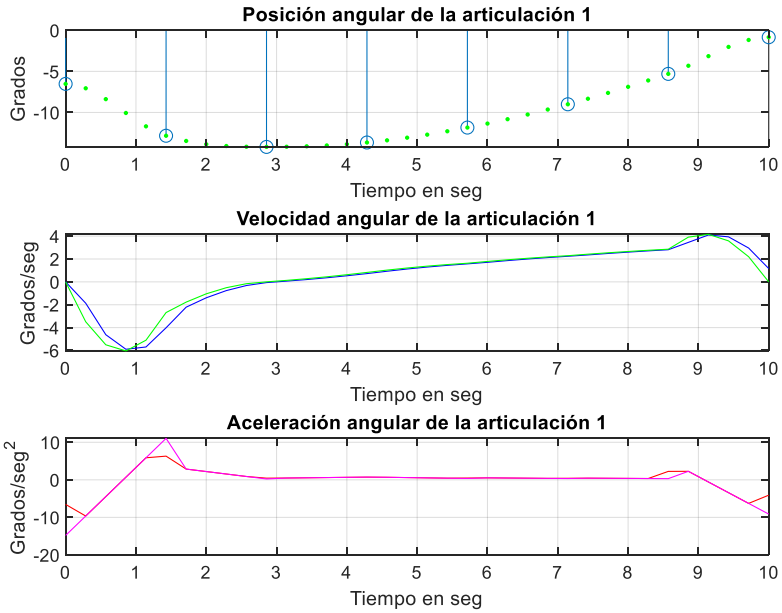


Ilustración 71. Posición, velocidad y aceleración de la primera articulación del segundo ejemplo.

En la ilustración 67 se representan la posición, velocidad y aceleración de la primera articulación del robot. Aunque en este caso el ángulo de la articulación inicial y final sea el mismo (0º), esto no quiere decir que durante la trayectoria dicha articulación posea este valor constantemente. Esto se debe a que el manipulador no puede llegar a la posición deseada sin modificar el valor de la articulación, por lo que la modifica de forma que cumpla con los rangos establecidos. Para cada uno de los puntos de la trayectoria se ha calculado su velocidad y aceleración correspondiente, por lo que, a lo largo de la trayectoria se observa la sucesión de estos valores formando así graficas con continuidad uniforme desde que comienza el movimiento hasta que este termina.

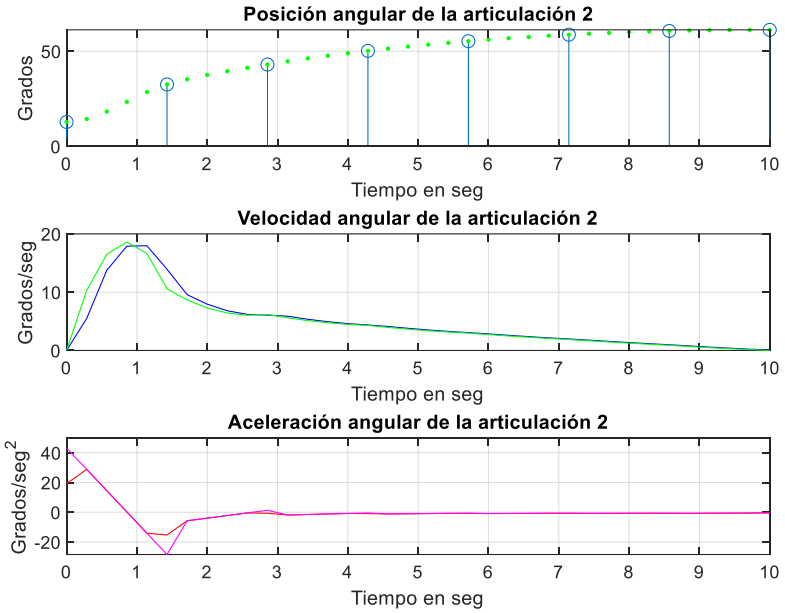


Ilustración 72. Posición, velocidad y aceleración de la segunda articulación del segundo ejemplo.

En la ilustración 68 se representan la posición, velocidad y aceleración de la segunda articulación del robot. En esta se puede observar como la posición angular sigue una trayectoria uniforme desde el primer punto de la trayectoria, en este caso parte la articulación de un valor de 0º; hasta el último punto, en el que finalmente se posicione en los 60º deseados del punto de destino deseado. Al igual que en la gráfica anterior se observan las velocidades y aceleraciones correspondientes a cada punto de la trayectoria de forma que se proyecten graficas con continuidad uniforme desde que comienza el movimiento hasta que este termina.

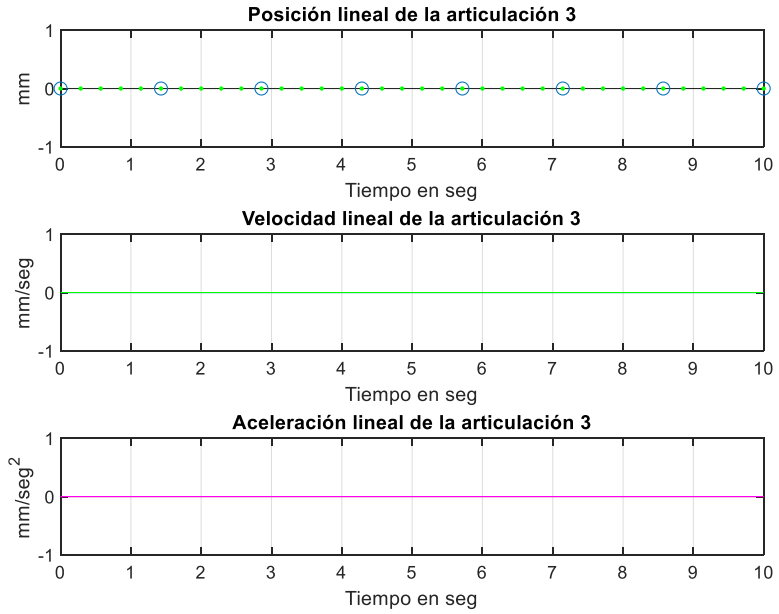


Ilustración 73. Posición, velocidad y aceleración de la tercera articulación del segundo ejemplo.

En la ilustración 69 se representan la posición, velocidad y aceleración de la tercera articulación del robot. En esta se puede observar como la posición lineal sigue una trayectoria uniforme, dicha articulación no influye en el posicionamiento de los ángulos de las demás articulaciones, por lo que, al no modificarse su valor en el primer y último punto, la trayectoria será una línea recta constante de valor 0. Puesto que no existe desplazamiento lineal, la velocidad y aceleración de esta articulación será de cero.

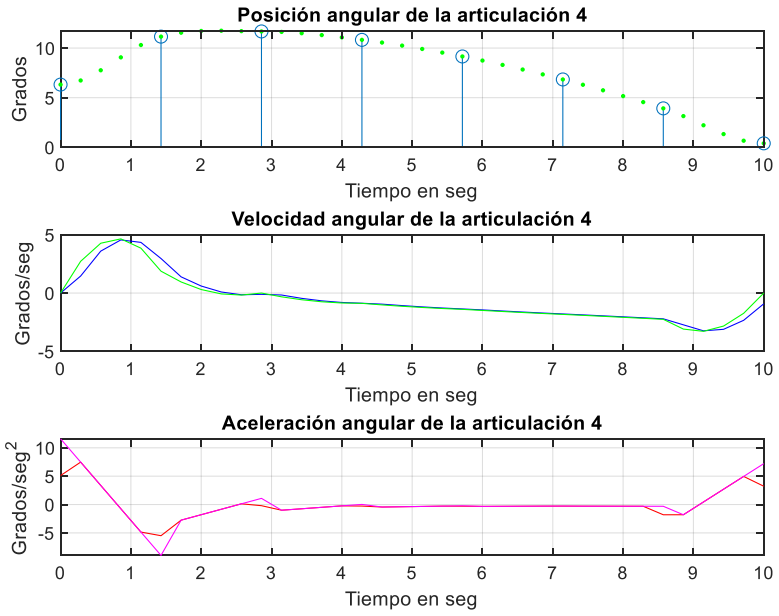


Ilustración 74. Posición, velocidad y aceleración de la cuarta articulación del segundo ejemplo.

En la ilustración 67 se representan la posición, velocidad y aceleración de la primera articulación del robot. Al igual que ocurría en la primera articulación, aunque en este caso el ángulo de la articulación inicial y final sea el mismo (0º), esto no quiere decir que durante la trayectoria dicha articulación posea este valor constantemente debido que el manipulador necesitara modificar este con el fin de llegar a la posición deseada. Para cada uno de los puntos de la trayectoria se ha calculado su velocidad y aceleración correspondiente, por lo que, a lo largo de la trayectoria se observa la sucesión de estos valores formando así graficas con continuidad uniforme desde que comienza el movimiento hasta que este termina.

Una vez obtenida la matriz que contiene la trayectoria de cada una de las articulaciones, introducimos estos datos como argumento de entrada en la función de la cinemática directa, la cual nos devolverá la posición y orientación del manipulador en el espacio cartesiano a lo largo de las trayectorias, así como se visualizará el recorrido de las mismas en el manipulador.

```
>> posicion(:,:,1)

ans =

   158.5000
          0
    85.0000
```

```
>> orientacion(:,:,1)

ans =

     0     0   180
```

Ilustración 75. Primer punto de la trayectoria en coordenadas cartesianas

```
>> posicion(:,:,end)

ans =

  118.5000
   69.2820
   85.0000
```

```
>> orientacion(:,:,end)

ans =

   60.0000         0  180.0000
```

Ilustración 76. Punto final de la trayectoria en coordenadas cartesianas

En las ilustraciones 71 y 72 se muestra el primer y último punto de la trayectoria, respectivamente, los cuales coinciden con los establecidos inicialmente a la hora de obtener las trayectorias de las articulaciones. De esta forma se comprueba que la trayectoria parte y llega a los puntos deseados, además indica que el generador de trayectorias funciona correctamente.

A continuación, se muestra en la Ilustración 73 el robot en el punto inicial de la trayectoria y en la ilustración 74 el punto final de la misma con la trayectoria dibujada.

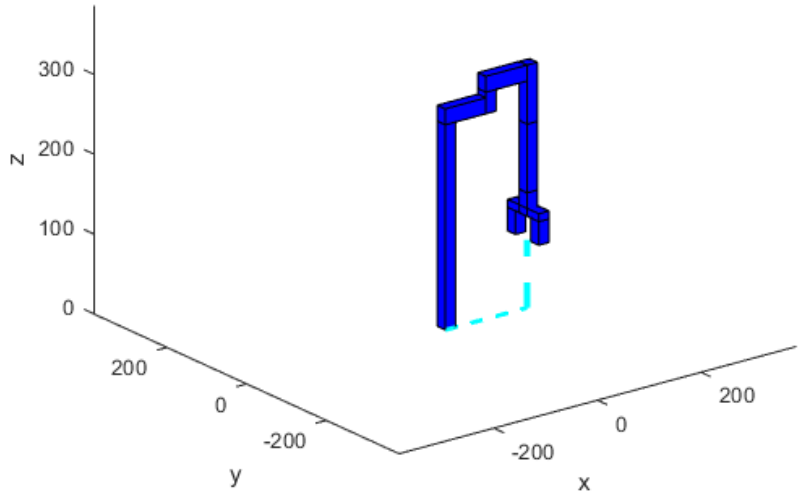


Ilustración 77. Representación del punto inicial de la trayectoria en el manipulador

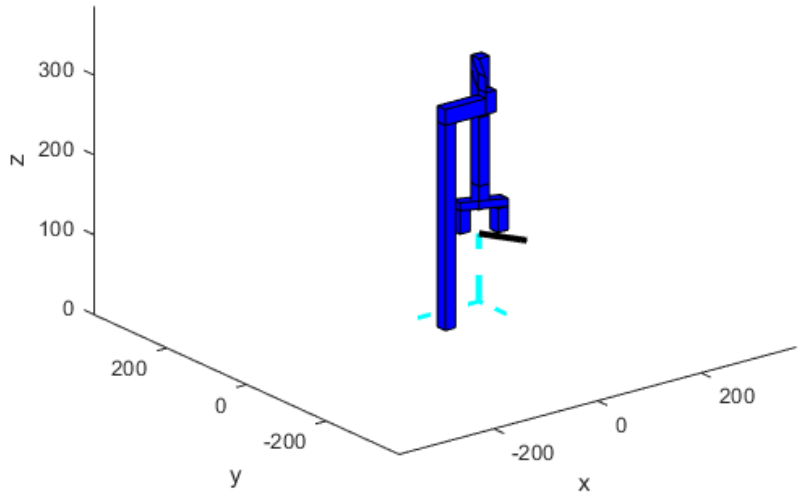


Ilustración 78. Representación del punto final de la trayectoria en el manipulador, así como la propia trayectoria dibujada

6.2. Arduino

Empresa de desarrollo de software y hardware de código abierto, así como una comunidad internacional dedicada a diseñar y fabricar placas de desarrollo de hardware, como la que se usa en el proyecto, para construir dispositivos digitales e interactivos que pueden descubrir y manipular objetos. Las placas de Arduino se programan a través de un ordenador, mediante comunicación serie.

Una vez realizada la comprobación de que las funciones programadas funcionan correctamente, se procede a programar la placa Arduino Uno. En este se incluirán las funciones de cinemática inversa, control cinemático e interpolación cubica desarrolladas en Matlab. Su estructura será semejante a la explicada anteriormente, teniendo en cuenta la programación en el lenguaje c/c++.

A continuación, se procede a explicar las funciones que han sido necesarias para el desarrollo del control del manipulador (funciones nuevas y en las que ha sido necesario hacer alguna modificación), así como su función principal.

Tanto los códigos desarrollados en Matlab como en Arduino se pueden encontrar en el segundo documento de este trabajo (anexos).

6.3.1. Funcion ControlCinematico

Al igual que se realizó en el entorno de Matlab, en esta función se determinarán los caminos que cada articulación debe seguir para lograr un camino dado con limitaciones.

El único cambio que se ha realizado en esta función es el de que solo va a ser una función de variables de entrada, no va a devolver ningún dato puesto que a la vez que se vaya escribiendo por pantalla el valor de cada articulación, se llamara a una función creada específicamente para que posicione las articulaciones en su correspondiente articulación.

De manera que, primero se determinen todos los valores de la trayectoria para cada articulación y posteriormente vayan pasando por estos puntos todas las articulaciones conjuntamente.

```
inct = Time/(NCrossPoints+1);

for (int i=0;i<Time;i++){

  DATA1[i] = inct*i;

}

for (int n=0;n<Njoints;n++){
```

```
  if(n==0){
    InterpoladorCubico(DATA1,Jvals1,inct,NSampPoints,n,NCrossPoints);
  }else if(n==1){
    InterpoladorCubico(DATA1,Jvals2,inct,NSampPoints,n,NCrossPoints);
  }else if(n==2){
    InterpoladorCubico(DATA1,Jvals3,inct,NSampPoints,n,NCrossPoints);
  }else{
    InterpoladorCubico(DATA1,Jvals4,inct,NSampPoints,n,NCrossPoints);
  };
 };
 float tespera = inct/(NSampPoints+1);
 int valorF = ((NCrossPoints+1)*(NSampPoints+1))+1;
 for(int c=0;c<valorF;c++){
  Serial.print("              ");Serial.print(Jvals[0][c]);
  Serial.print("    ");Serial.print(Jvals[1][c]);
  Serial.print("    ");Serial.print(Jvals[2][c]);
  Serial.print("    ");Serial.println(Jvals[3][c]);
  MoveServos(tespera,Jvals[0][c],Jvals[1][c],Jvals[2][c],Jvals[3][c],c,valorF);
 };
 VAL=0;
};
```

6.3.2. Funcion InterpoladorCubico

Al igual que se realizó en el entorno de Matlab, esta función interpola una trayectoria conjunta utilizando N puntos de cruce en DATA, el tiempo por intervalo y N puntos de muestreo para cada

intervalo entre los puntos de cruce, de forma que devuelva todas las posiciones por las que pasa la trayectoria en la variable Jvals.

La única modificación que se realizó en esta función fue a la hora de hallar los coeficientes del tramo, puesto que para determinar la inversa de una matriz en el lenguaje c se necesita realizar todos los pasos para resolverla. Lo contrario que pasaba en Matlab, ya que con un simple comando se podía resolver.

```
T = (DATA1[j+1]-DATA1[j]);

a0 = DATA2[j];

a1 = Vini;

A[0][0] = pow(T,2);

A[0][1] = pow(T,3);

A[1][0] = 2*T;

A[1][1] = 3*pow(T,2);

B[0][0] = DATA2[j+1]-a0-a1*T;

B[1][0] = Vfin-a1;

determinante = A[0][0]*A[1][1]-A[0][1]*A[1][0];

adjunta[0][0]=A[1][1];

adjunta[0][1]=-A[1][0];

adjunta[1][0]=-A[0][1];

adjunta[1][1]=A[0][0];

TrasA[0][0]=adjunta[0][0];

TrasA[0][1]=adjunta[1][0];

TrasA[1][0]=adjunta[0][1];

TrasA[1][1]=adjunta[1][1];

InvA[0][0]=(1/determinante)*TrasA[0][0];

InvA[0][1]=(1/determinante)*TrasA[0][1];
```

```
InvA[1][0]=(1/determinante)*TrasA[1][0];

InvA[1][1]=(1/determinante)*TrasA[1][1];

X[0][0] = (InvA[0][0]*B[0][0])+(InvA[0][1]*B[1][0]);

X[1][0] = (InvA[1][0]*B[0][0])+(InvA[1][1]*B[1][0]);

a2 = X[0][0]; a3 = X[1][0];
```

Otra de las modificaciones que se ha realizado ha sido a la hora de hallar la interpolación cubica una vez que se habían obtenido los coeficientes del tramo. Dentro de este tramo se establecen N puntos de muestreo, en los que en cada uno de ellos existirá un valor de tiempo específico. Por lo que se creó una condición inicial para saber si se estaba trabajando en el primer tramo o en los restantes. Esto es importante puesto que en el primer tramo hay que tener en cuenta un punto inicial, sumándole así un punto mas a ese numero de puntos de muestreo, cosa que no es necesaria en los demás tramos de la trayectoria.

En cada una de estas condiciones se establece un bucle for() de forma que se halle las posiciones de las articulaciones para esos puntos del tramo (Sints). En el bucle de la segunda condición se determina la posición de la variable donde se guardará el dato, puesto que se deberá almacenar teniendo en cuenta lo hallado en el primer tramo.

```
if (j==0){
  for(int k=0;k<(Sints+1);k++){
    ts = k*(inct/Sints);
    Jvals[n][k]=a0+a1*ts+a2*pow(ts,2)+a3*pow(ts,3);
  };
}else{
  for(int k=0;k<Sints;k++){
    ts=(k+1)*(inct/Sints);
    int number;
    number = k+((5*j)+1);
    Jvals[n][number]=a0+a1*ts+a2*pow(ts,2)+a3*pow(ts,3);
  };
```

```
  };
 };
};
```

6.3.3. Función MoveServos

Esta función tiene la tarea de colocar los servomotores y el actuador lineal en la posición del tramo correspondiente. Posee como argumentos de entrada el tiempo de espera que deberá espera entre cada punto de la trayectoria, el valor de cada articulación para el punto de la trayectoria, el punto de la trayectoria en el que se este llamando a la función y el numero total de puntos en la trayectoria.

Se comienza comprobando en que punto del tramo se esta llamando a la función. Si se encuentra en el primer punto del tramo se mandará al servo que mueve las pinzas que se las habrá, mandándole a este un valor de cero grados. Por otro lado, si se encuentra en el ultimo punto del tramo se mandará al servo que cierre las pinzas, mandándole a este un valor de 88 grados. Esta última comprobación se realizará después de haber movido las articulaciones del robot de forma que para el ultimo punto se cierre la pinza después de posicionar correctamente al manipulador.

Mediante la instrucción servo.write() de la librería Servo.h se mandará a los servomotores a la posición deseada. Así como se comprobará si en la trayectoria se ha pedido que actúe el actuador, de forma que si es cierto este también proceda a hacer su trayectoria. Además, se calculará la velocidad a la que debe funcionar el mismo durante la trayectoria de forma que llegue a punto deseado.

```
void MoveServos(float Tespera,float dataV1,float dataV2,float dataV3,float dataV4, int np,int fb,
float Time,int vFinal){
 int espera = Tespera*1000;
 int espera = Tespera*1000;
 int tvelocidad;
 int velocidad;
 if(np==0){
  servo4.write(0);
  delay(1000);  }   tvelocidad = (5.3*vFinal)/50;
```

```
  velocidad =(255*tvelocidad)/Time;
  servo1.write(90+dataV1);
  servo2.write(90+dataV2);
  servo3.write(90+dataV4);
  if(dataV3>0){
    digitalWrite (IN1, HIGH);
    digitalWrite (IN2, LOW);
    analogWrite (ENA, velocidad);
  }
  delay(espera);
  if(np==(fb-1)){
    /*cierra las pinzas*/
    servo4.write(88);
    delay(1000);
  }
}
```

6.3.4. Funcion Loop

La función principal del sketch se encargará de preguntar al usuario por los datos necesarios para comenzar la trayectoria, siendo estos el numero de puntos de cruce, el numero de puntos de muestreo en cada tramo, la duración de la trayectoria, la configuración de la estructura del manipulador, la posición final de la trayectoria y el modo de funcionamiento en el que se esta trabajando una vez configurado el robot. A la vez que se van introduciendo los datos, se comprobara que estos son correctos, de forma que si no lo fueran terminaría la ejecución y se enviaría un mensaje de error por pantalla.

```
  Serial.print("Numero de puntos de cruce: ");delay(6000);
  NCrossPoints=Serial.parseInt();
```

```
Serial.println(NCrossPoints);
delay(1000);
if((NCrossPoints<1)||(NCrossPoints>5)){
 Serial.println ("/*ERROR*/");delay(100);
 exit(0);
}
Serial.print("Numero de puntos de muestreo: ");delay(6000);
NSampPoints=Serial.parseInt();
Serial.println(NSampPoints);
delay(1000);
if((NSampPoints<1)||(NSampPoints>4)){
 Serial.println("/*ERROR*/");delay(100);
 exit(0);
}
Serial.print("Duracion de la trayectoria (seg): ");delay(6000);
Time=Serial.parseFloat();
Serial.println(Time);
delay(1000);
if(Time<0){
 Serial.println("/*ERROR*/");delay(100);
 exit(0);
}
Serial.println("Configuracion estructura manipulador: (1) codo derecha - (0) codo izquierda ");
delay(6000);
Serial.print("Codo: ");
codo=Serial.parseFloat();
```

```
Serial.println(codo);
delay(1000);
if((codo<0)||(codo>1)){
 Serial.println("/*ERROR*/");delay(100);
 exit(0);
}
Serial.println("Posicion final del robot: ");
Serial.print("Coordenada X: ");delay(6000);
Final_pose[0]=Serial.parseFloat();
Serial.println(Final_pose[0]);
delay(1000);
Serial.print("Coordenada Y: ");delay(6000);
Final_pose[1]=Serial.parseFloat();
Serial.println(Final_pose[1]);
delay(1000);
Serial.print("Coordenada Z: ");delay(6000);
Final_pose[2]=Serial.parseFloat();
Serial.println(Final_pose[2]);
delay(1000);
Serial.print("Orientacion alpha: ");delay(6000);
Final_pose[3]=Serial.parseFloat();
Serial.println(Final_pose[3]);
delay(1000);
Final_pose[4]=0;
Final_pose[5]=180;
Serial.println("Modo funcionamiento: ");
```

```
Serial.println("(1)Led Verde - Modo 1: Eslabones cortos");
Serial.println("(2)Led Azul - Modo 2: Eslabones largos");
Serial.println("(3)Led Amarillo - Modo 3: Eslabon1 corto - Eslabon2 largo");
Serial.println("(4)Led Blanco - Modo 4: Eslabon1 largo - Eslabon2 corto");delay(6000);
MODO = Serial.parseInt();
Serial.print("MODO: ");Serial.println(MODO);
delay(1000);
```

Una vez obtenidos todos los datos necesarios para realizar la trayectoria, se comprobará en qué modo se esta trabajando de forma que se asigne correctamente el valor de las dimensiones de los eslabones. Si el numero introducido de modo fuese erróneo, se enviaría un mensaje por pantalla y se acabaría la operación. Además, cada vez que se entre a trabajar en un modo se encenderá un LED, de forma que se pueda apreciar visualmente el modo de funcionamiento.

```
if (MODO==1){
  digitalWrite(LED_M1, HIGH);
  Parametros[0]=78.5;
  Parametros[1]=81;
}else if(MODO==2){
  digitalWrite (LED_M2, HIGH);
  Parametros[0]=97.5;
  Parametros[1]=100;
} else if(MODO==3){
  digitalWrite (LED_M3, HIGH);
  Parametros[0]=78.5;
  Parametros[1]=100;
} else if(MODO==4){
  digitalWrite (LED_M4, HIGH);
```

```
  Parametros[0]=97.5;
  Parametros[1]=81;
 }else{
  Serial.println("/** ERROR **/");
  Serial.println("/** MODO DE FUNCIONAMIENTO NO VALIDO **/");
  delay(100);
  exit(0);
 }
```

Una vez seleccionados los valores se dará comienzo a la ejecución de la trayectoria. Para cerciorarnos de que se realiza completamente el control, se crea un bucle while() de forma que mientras no cambien el valor de una variable global se mantendrá dentro del bucle. Una vez haya finalizado este se mandará apagar todos los LEDs y se saldrá de la ejecución finalizando así el programa.

```
 while(VAL==1){
  ControlCinematico(Parametros,Init_pose,Final_pose,Time,NCrossPoints,NSampPoints,codo);
  }
  digitalWrite (LED_M1, LOW);
  digitalWrite (LED_M2, LOW);
  digitalWrite (LED_M3, LOW);
  digitalWrite (LED_M4, LOW);
  exit(0);
 }
```

6.3.5. Simulación

Finalmente se mostrará un ejemplo de la comunicación entre el usuario y la interfaz. Se utilizarán como datos de entrada los utilizados en la simulación del entorno Matlab (segundo ejemplo), de forma que estos sirvan para verificar si la simulación en el entorno de Arduino es correcta.

Puesto que los scripts desarrollados en el entorno de Matlab se han diseñado para garantizar que el manipulador funcione correctamente, si los datos recibidos en el entorno Arduino son equivalentes a los proporcionados en Matlab, el robot funcionara correctamente generando la trayectoria deseada.

Se comenzará subiendo el sketch a la placa de Arduino de forma que cuando esté verificada empiece a interactuar con el usuario mediante el monitor serie donde se pedirán los datos necesarios.

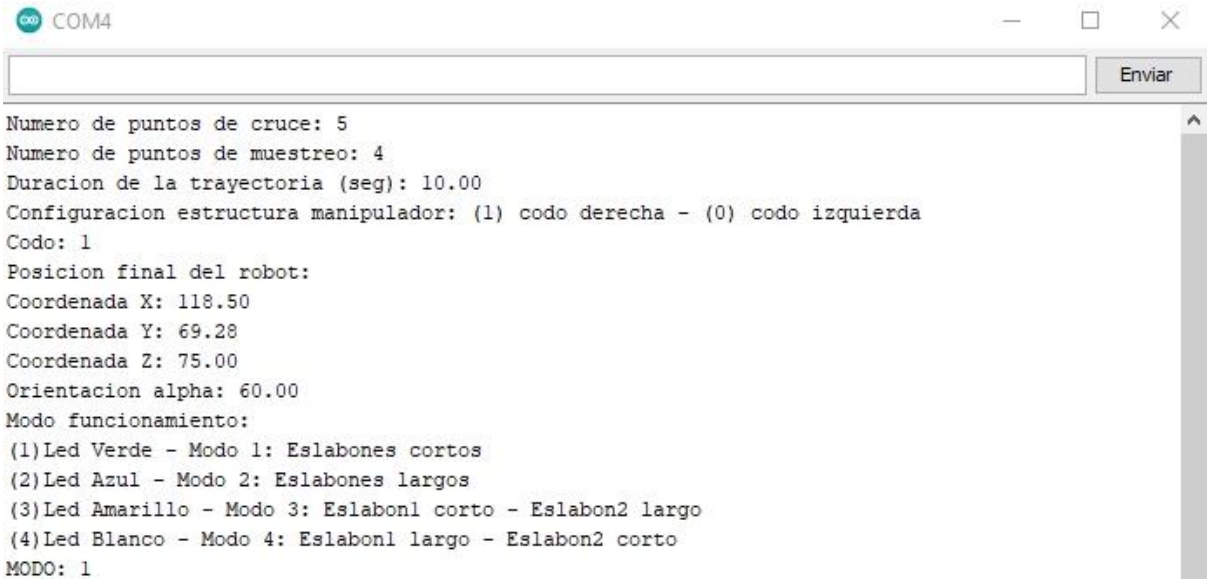


Ilustración 79. Comunicación serie con el usuario

Una vez introducidos los datos, el programa comenzara a realizar los cálculos necesarios para proporcionar los valores a las articulaciones de forma que estos realicen la trayectoria hasta llegar al punto final deseado.

Articulacion:	1	2	3	4
	-4.00	0.00	0.00	6.00
	-4.11	0.78	0.00	5.95
	-4.44	2.86	0.00	5.85
	-4.97	5.83	0.00	5.82
	-5.67	9.29	0.00	5.94
	-6.52	12.84	0.00	6.32
	-7.69	16.80	0.00	7.11
	-9.19	21.43	0.00	8.24
	-10.77	26.26	0.00	9.49
	-12.21	30.82	0.00	10.61
	-13.25	34.63	0.00	11.39
	-13.84	37.59	0.00	11.73
	-14.13	40.01	0.00	11.80
	-14.22	42.06	0.00	11.71
	-14.21	43.89	0.00	11.58
	-14.18	45.69	0.00	11.51
	-14.12	47.43	0.00	11.42
	-13.93	48.99	0.00	11.19
	-13.64	50.42	0.00	10.86
	-13.29	51.73	0.00	10.47

-12.89	52.97	0.00	10.08
-12.44	54.11	0.00	9.67
-11.93	55.15	0.00	9.23
-11.35	56.09	0.00	8.74
-10.72	56.95	0.00	8.23
-10.05	57.73	0.00	7.68
-11.99	58.49	0.00	7.10
-16.53	59.21	0.00	6.50
-19.69	59.83	0.00	5.87
-17.50	60.26	0.00	5.19
-5.97	60.43	0.00	4.45

Ilustración 80. Resultado de los valores de las articulaciones durante la trayectoria.

Se puede observar en la ilustración 80 como los valores otorgados a las articulaciones son bastante semejantes a los hallados en Arduino (ilustración 70), existe un pequeño error numérico en los datos proporcionados por Arduino, pero estos no influyen drásticamente en los resultados, por lo que se asume como correcta la simulación.

DOCUMENTO 2
ANEXOS

ÍNDICE DE CONTENIDO

1. DATASHEETS

1.1. Datasheet Arduino uno R3

Arduino® UNO R3

Product Reference Manual
SKU: A000066

Description

The Arduino UNO R3 is the perfect board to get familiar with electronics and coding. This versatile microcontroller is equipped with the well-known ATmega328P and the ATMega 16U2 Processor.
This board will give you a great first experience within the world of Arduino.

Target areas:

Maker, introduction, industries

Arduino® UNO R3

1 The Board

1.1 Application Examples

The UNO board is the flagship product of Arduino. Regardless if you are new to the world of electronics or will use the UNO as a tool for education purposes or industry-related tasks.

First entry to electronics: If this is your first project within coding and electronics, get started with our most used and documented board; Arduino UNO. It is equipped with the well-known ATmega328P processor, 14 digital input/output pins, 6 analog inputs, USB connections, ICSP header and reset button. This board includes everything you will need for a great first experience with Arduino.

Industry-standard development board: Using the Arduino UNO board in industries, there are a range of companies using the UNO board as the brain for their PLC's.

Education purposes: Although the UNO board has been with us for about ten years, it is still widely used for various education purposes and scientific projects. The board's high standard and top quality performance makes it a great resource to capture real time from sensors and to trigger complex laboratory equipment to mention a few examples.

1.2 Related Products

- Starter Kit
- Tinkerkit Braccio Robot
- Example

2 Ratings

2.1 Recommended Operating Conditions

Symbol	Description	Min	Max
	Conservative thermal limits for the whole board:	-40 °C (-40°F)	85 °C (185°F)

NOTE: In extreme temperatures, EEPROM, voltage regulator, and the crystal oscillator, might not work as expected due to the extreme temperature conditions

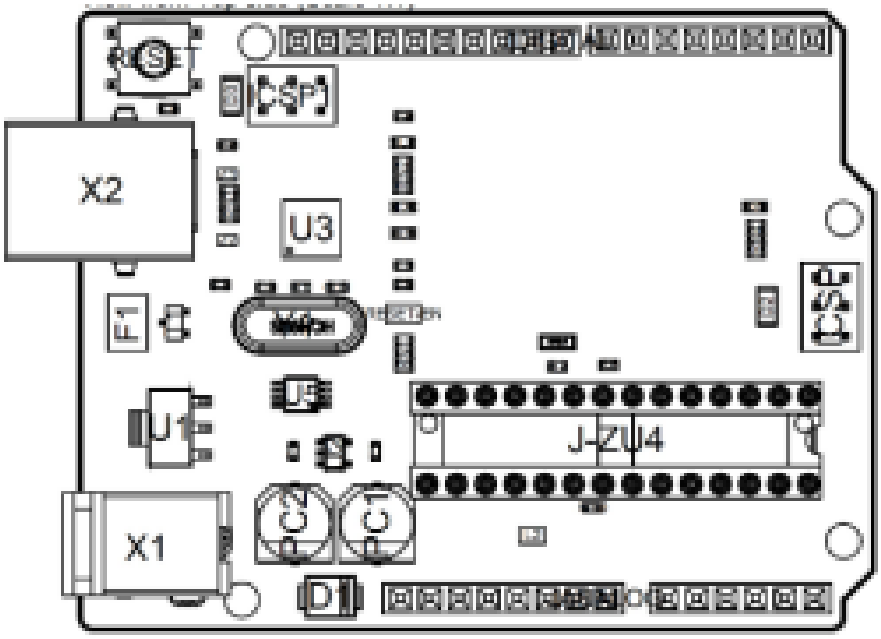
2.2 Power Consumption

Symbol	Description	Min	Typ	Max	Unit
VINMax	Maximum input voltage from VIN pad	6	-	20	V
VUSBMax	Maximum input voltage from USB connector		-	5.5	V
PMax	Maximum Power Consumption	-	-	xx	mA

3 Functional Overview

3.1 Board Topology

Top view



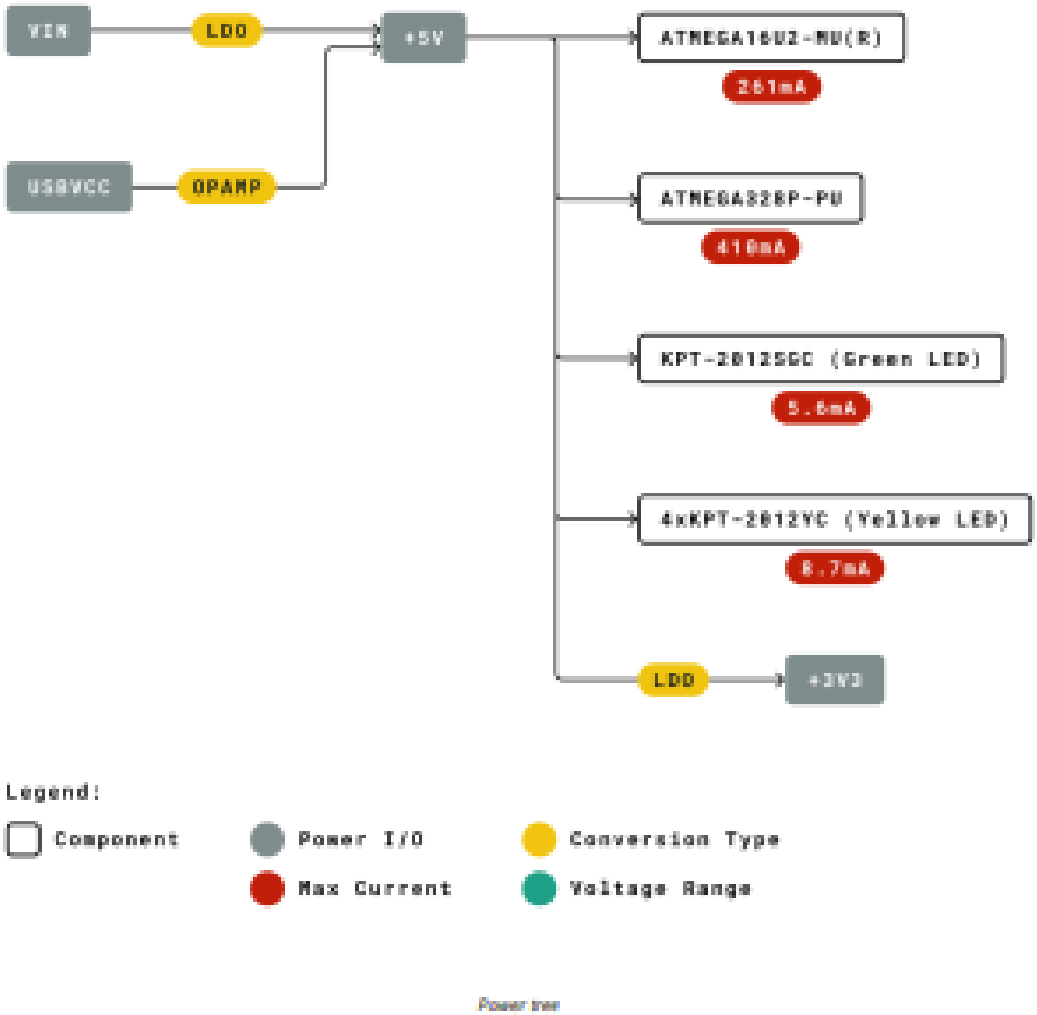
Board topology

Ref.	Description	Ref.	Description
X1	Power jack 2.1x5.5mm	U1	SPX1117M3-L-5 Regulator
X2	USB B Connector	U3	ATMEGA16U2 Module
PC1	EEE-1EA470WP 25V SMD Capacitor	U5	LMV358LIST-A.9 IC
PC2	EEE-1EA470WP 25V SMD Capacitor	F1	Chip Capacitor, High Density
D1	CGRA4007-G Rectifier	ICSP	Pin header connector (through hole 6)
J-ZU4	ATMEGA328P Module	ICSP1	Pin header connector (through hole 6)
Y1	ECS-160-20-4X-DU Oscillator		

3.2 Processor

The Main Processor is a ATmega328P running at up tp 20 MHz. Most of its pins are connected to the external headers, however some are reserved for internal communication with the USB Bridge coprocessor.

3.3 Power Tree



Power tree

4 Board Operation

4.1 Getting Started - IDE

If you want to program your Arduino UNO while offline you need to install the Arduino Desktop IDE [1] To connect the Arduino UNO to your computer, you'll need a Micro-B USB cable. This also provides power to the board, as indicated by the LED.

4.2 Getting Started - Arduino Web Editor

All Arduino boards, including this one, work out-of-the-box on the Arduino Web Editor [2], by just installing a simple plugin.

The Arduino Web Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow **[3]** to start coding on the browser and upload your sketches onto your board.

4.3 Getting Started - Arduino IoT Cloud

All Arduino IoT enabled products are supported on Arduino IoT Cloud which allows you to Log, graph and analyze sensor data, trigger events, and automate your home or business.

4.4 Sample Sketches

Sample sketches for the Arduino XXX can be found either in the "Examples" menu in the Arduino IDE or in the "Documentation" section of the Arduino Pro website [4]

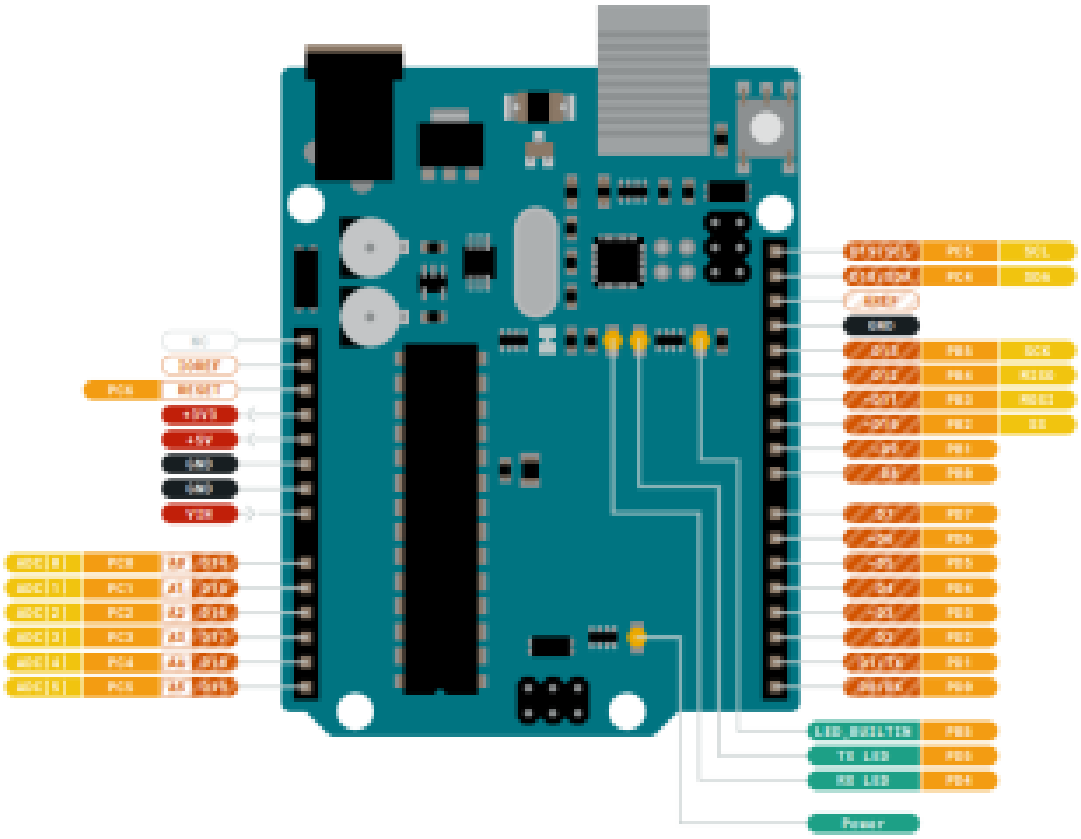
4.5 Online Resources

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking exciting projects on ProjectHub **[5]**, the Arduino Library Reference **[6]** and the online store **[7]** where you will be able to complement your board with sensors, actuators and more

4.6 Board Recovery

All Arduino boards have a built-in bootloader which allows flashing the board via USB. In case a sketch locks up the processor and the board is not reachable anymore via USB it is possible to enter bootloader mode by double-tapping the reset button right after power up.

5 Connector Pinouts



Pinout

5.1 JANALOG

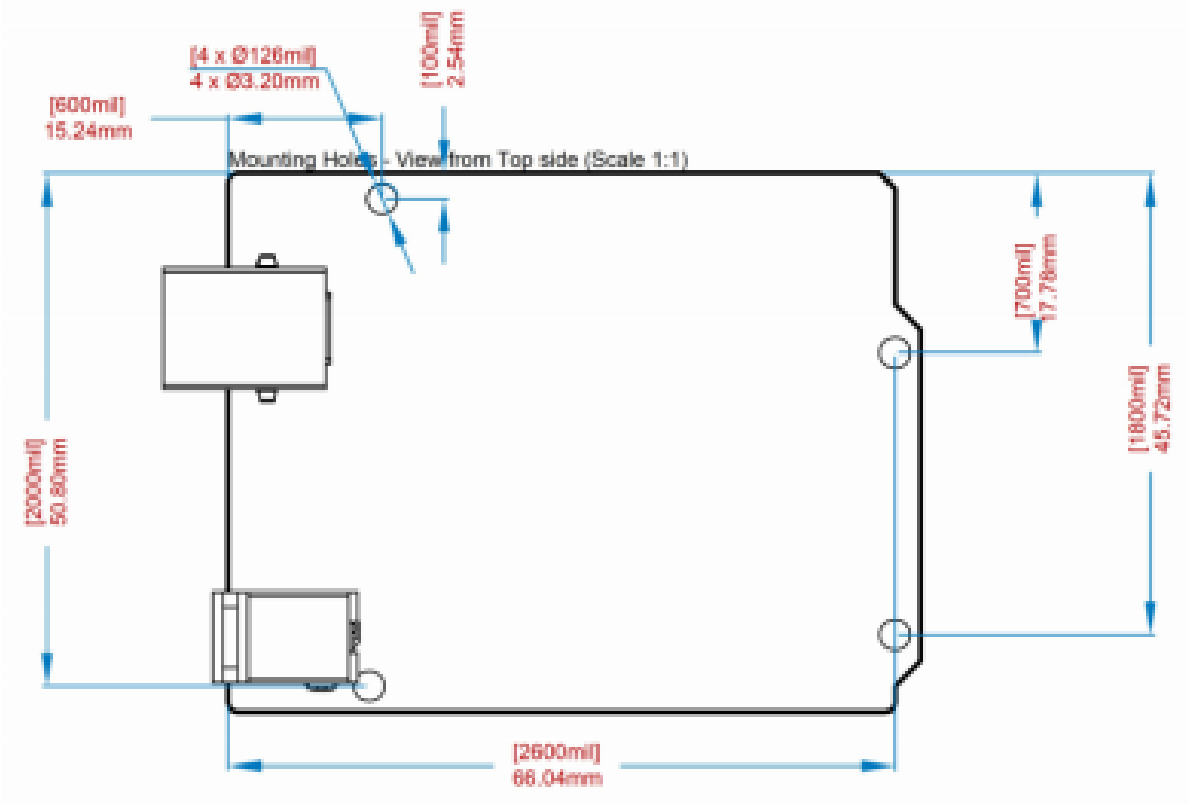
Pin	Function	Type	Description
1	NC	NC	Not connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog/GPIO	Analog input 0 /GPIO
10	A1	Analog/GPIO	Analog input 1 /GPIO
11	A2	Analog/GPIO	Analog input 2 /GPIO
12	A3	Analog/GPIO	Analog input 3 /GPIO
13	A4/SDA	Analog input/I2C	Analog input 4/I2C Data line
14	A5/SCL	Analog input/I2C	Analog input 5/I2C Clock line

5.2 JDIGITAL

Pin	Function	Type	Description
1	D0	Digital/GPIO	Digital pin 0/GPIO
2	D1	Digital/GPIO	Digital pin 1/GPIO
3	D2	Digital/GPIO	Digital pin 2/GPIO
4	D3	Digital/GPIO	Digital pin 3/GPIO
5	D4	Digital/GPIO	Digital pin 4/GPIO
6	D5	Digital/GPIO	Digital pin 5/GPIO
7	D6	Digital/GPIO	Digital pin 6/GPIO
8	D7	Digital/GPIO	Digital pin 7/GPIO
9	D8	Digital/GPIO	Digital pin 8/GPIO
10	D9	Digital/GPIO	Digital pin 9/GPIO
11	SS	Digital	SPI Chip Select
12	MOSI	Digital	SPI1 Main Out Secondary In
13	MISO	Digital	SPI Main In Secondary Out
14	SCK	Digital	SPI serial clock output
15	GND	Power	Ground
16	AREF	Digital	Analog reference voltage
17	A4/SD4	Digital	Analog input 4/I2C Data line (duplicated)
18	A5/SD5	Digital	Analog input 5/I2C Clock line (duplicated)

5.3 Mechanical Information

5.4 Board Outline & Mounting Holes



Board outline

6 Certifications

6.1 Declaration of Conformity CE DoC (EU)

We declare under our sole responsibility that the products above are in conformity with the essential requirements of the following EU Directives and therefore qualify for free movement within markets comprising the European Union (EU) and European Economic Area (EEA).

ROHS 2 Directive 2011/65/EU	
Conforms to:	EN50581:2012
Directive 2014/35/EU. (LVD)	
Conforms to:	EN 60950-1:2006/A11:2009/A1:2010/A12:2011/AC:2011
Directive 2004/40/EC & 2008/46/EC & 2013/35/EU, EMF	
Conforms to:	EN 62311:2008

6.2 Declaration of Conformity to EU RoHS & REACH 211 01/19/2021

Arduino boards are in compliance with RoHS 2 Directive 2011/65/EU of the European Parliament and RoHS 3 Directive 2015/863/EU of the Council of 4 June 2015 on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

Substance	Maximum limit (ppm)
Lead (Pb)	1000
Cadmium (Cd)	100
Mercury (Hg)	1000
Hexavalent Chromium (Cr6+)	1000
Poly Brominated Biphenyls (PBB)	1000
Poly Brominated Diphenyl ethers (PBDE)	1000
Bis(2-Ethylhexyl) phthalate (DEHP)	1000
Benzyl butyl phthalate (BBP)	1000
Dibutyl phthalate (DBP)	1000
Diisobutyl phthalate (DIBP)	1000

Exemptions: No exemptions are claimed.

Arduino Boards are fully compliant with the related requirements of European Union Regulation (EC) 1907 /2006 concerning the Registration, Evaluation, Authorization and Restriction of Chemicals (REACH). We declare none of the SVHCs (https://echa.europa.eu/web/guest/candidate-list-table), the Candidate List of Substances of Very High Concern for authorization currently released by ECHA, is present in all products (and also package) in quantities totaling in a concentration equal or above 0.1%. To the best of our knowledge, we also declare that our products do not contain any of the substances listed on the "Authorization List" (Annex XIV of the REACH regulations) and Substances of Very High Concern (SVHC) in any significant amounts as specified by the Annex XVII of Candidate list published by ECHA (European Chemical Agency) 1907 /2006/EC.

6.3 Conflict Minerals Declaration

As a global supplier of electronic and electrical components, Arduino is aware of our obligations with regards to laws and regulations regarding Conflict Minerals, specifically the Dodd-Frank Wall Street Reform and Consumer Protection Act, Section 1502. Arduino does not directly source or process conflict minerals such as Tin, Tantalum, Tungsten, or Gold. Conflict minerals are contained in our products in the form of solder, or as a component in metal alloys. As part of our reasonable due diligence Arduino has contacted component suppliers within our supply chain to verify their continued compliance with the regulations. Based on the information received thus far we declare that our products contain Conflict Minerals sourced from conflict-free areas.

7 FCC Caution

Any Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference

(2) this device must accept any interference received, including interference that may cause undesired operation.

FCC RF Radiation Exposure Statement:

1. This Transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.
2. This equipment complies with RF radiation exposure limits set forth for an uncontrolled environment.
3. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

English: User manuals for license-exempt radio apparatus shall contain the following or equivalent notice in a conspicuous location in the user manual or alternatively on the device or both. This device complies with Industry Canada license-exempt RSS standard(s). Operation is subject to the following two conditions:

(1) this device may not cause interference

(2) this device must accept any interference, including interference that may cause undesired operation of the device.

French: Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes :

(1) l' appareil nedoit pas produire de brouillage

(2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

IC SAR Warning:

English This equipment should be installed and operated with minimum distance 20 cm between the radiator and your body.

French: Lors de l' installation et de l' exploitation de ce dispositif, la distance entre le radiateur et le corps est d 'au moins 20 cm.

Important: The operating temperature of the EUT can't exceed 85°C and shouldn't be lower than -40°C.

Hereby, Arduino S.r.l. declares that this product is in compliance with essential requirements and other relevant provisions of Directive 2014/53/EU. This product is allowed to be used in all EU member states.

8 Company Information

Company name	Arduino S.r.l
Company Address	Via Andrea Appiani 25 20900 MONZA Italy

9 Reference Documentation

Reference	Link
Arduino IDE (Desktop)	https://www.arduino.cc/en/Main/Software
Arduino IDE (Cloud)	https://create.arduino.cc/editor
Cloud IDE Getting Started	https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-4b3e4a
Arduino Pro Website	https://www.arduino.cc/pro
Project Hub	https://create.arduino.cc/projecthub?by=part&part_id=11332&sort=trending
Library Reference	https://www.arduino.cc/reference/en/
Online Store	https://store.arduino.cc/

10 Revision History

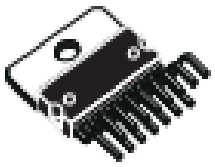
Date	Revision	Changes
xx/06/2021	1	Datasheet release

1.2. Datasheet L298n

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15

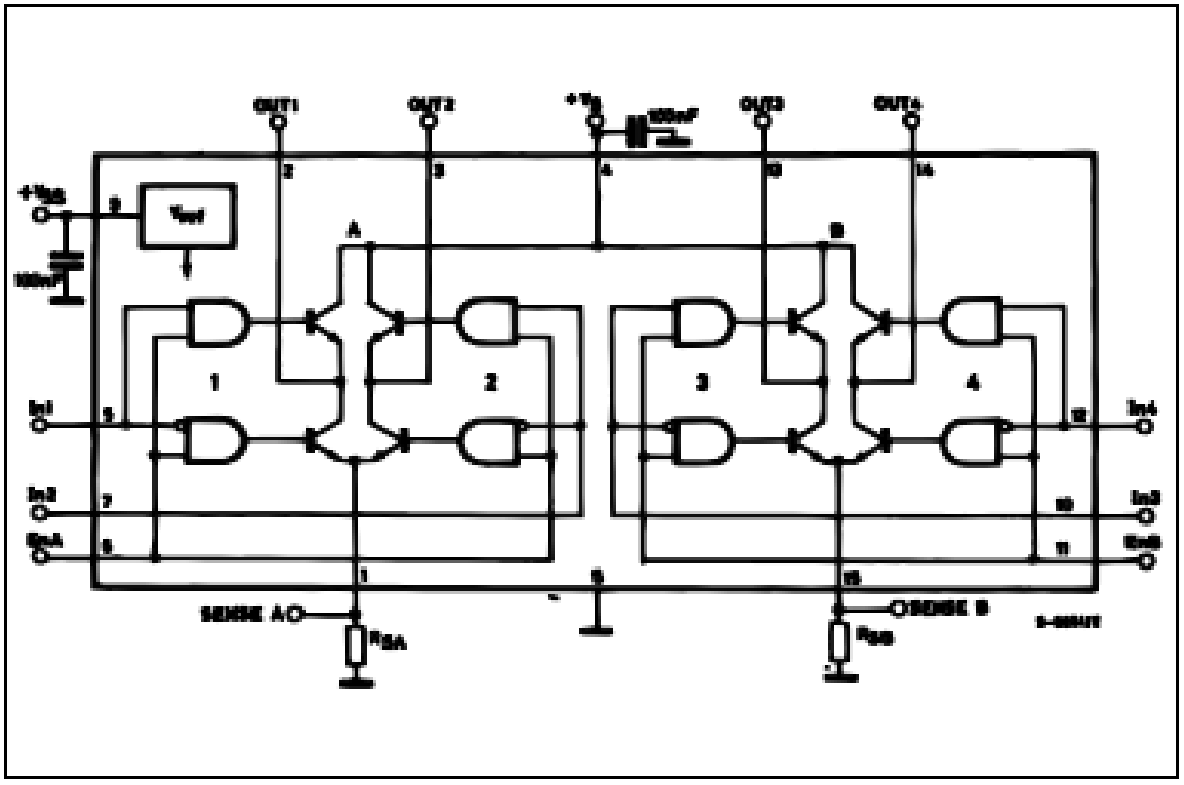
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

BLOCK DIAGRAM

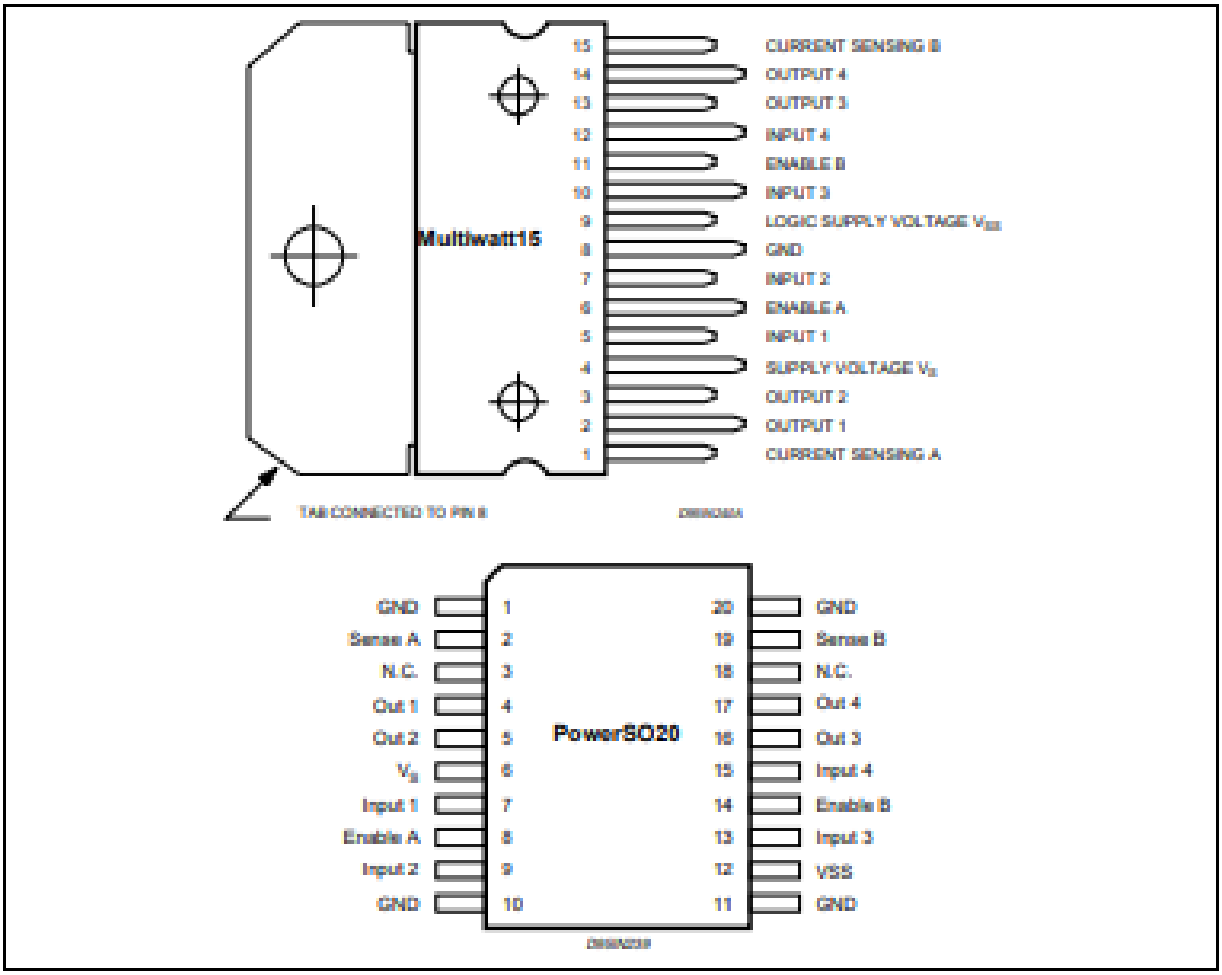


L298

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{IL}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{IH}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{IL}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{IH}	High Voltage Input Current (pins 5, 7, 10, 12)	V_i = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat\ (H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat\ (L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$T_1 (V_i)$	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
$T_2 (V_i)$	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
$T_3 (V_i)$	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
$T_4 (V_i)$	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
$T_5 (V_i)$	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
$T_6 (V_i)$	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
$T_7 (V_i)$	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
$T_8 (V_i)$	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
$T_1 (V_{en})$	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
$T_2 (V_{en})$	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
$T_3 (V_{en})$	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
$T_4 (V_{en})$	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
$T_5 (V_{en})$	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
$T_6 (V_{en})$	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
$T_7 (V_{en})$	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
$T_8 (V_{en})$	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ – 0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

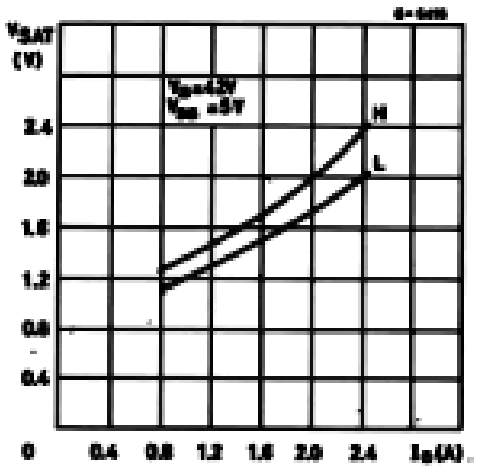
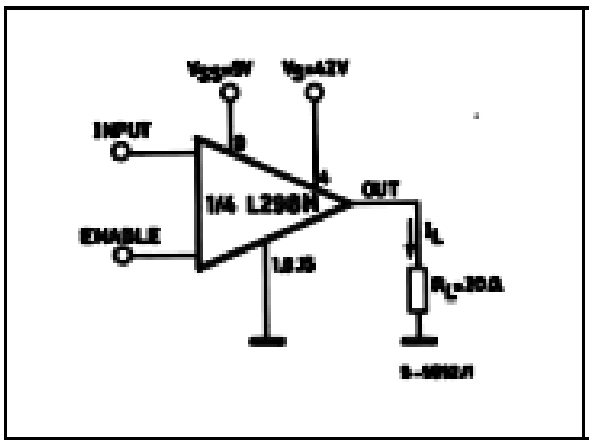


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 3 : Source Current Delay Times vs. Input or Enable Switching.

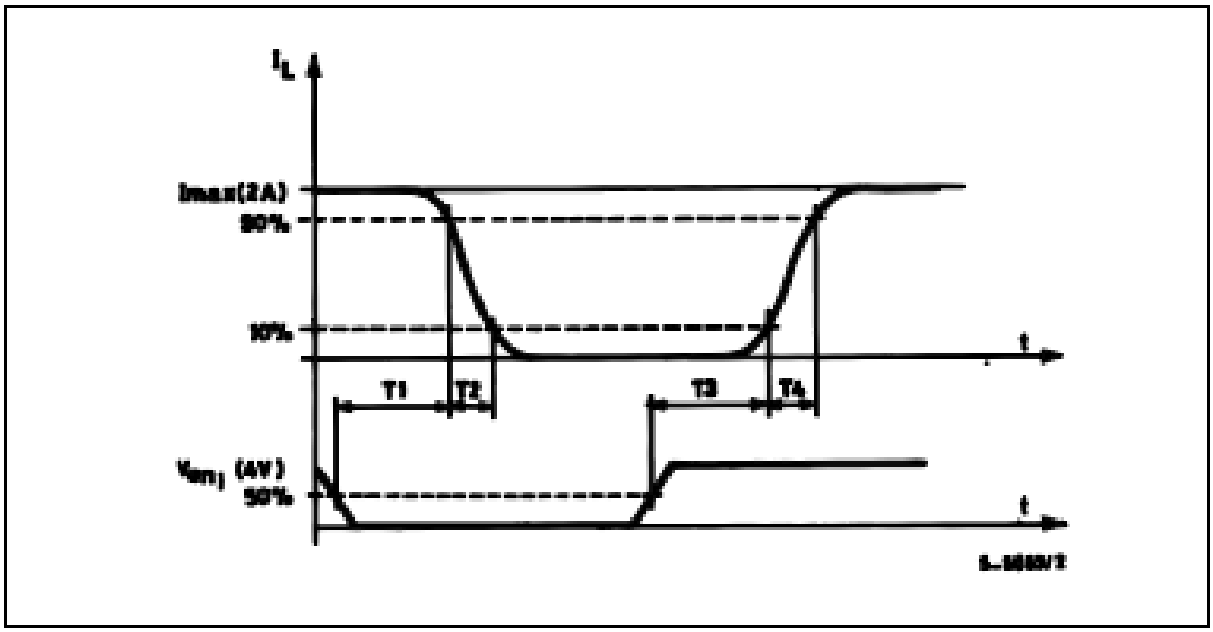
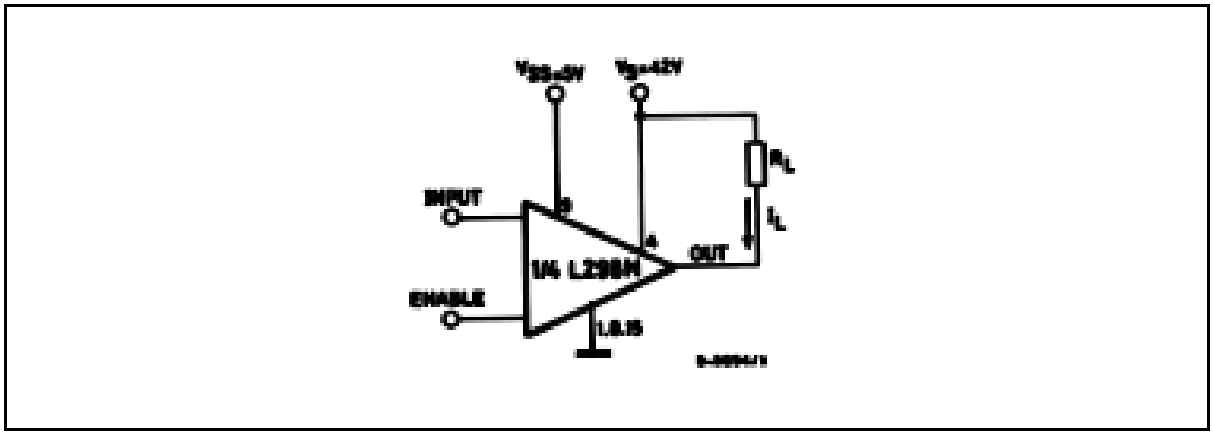


Figure 4 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

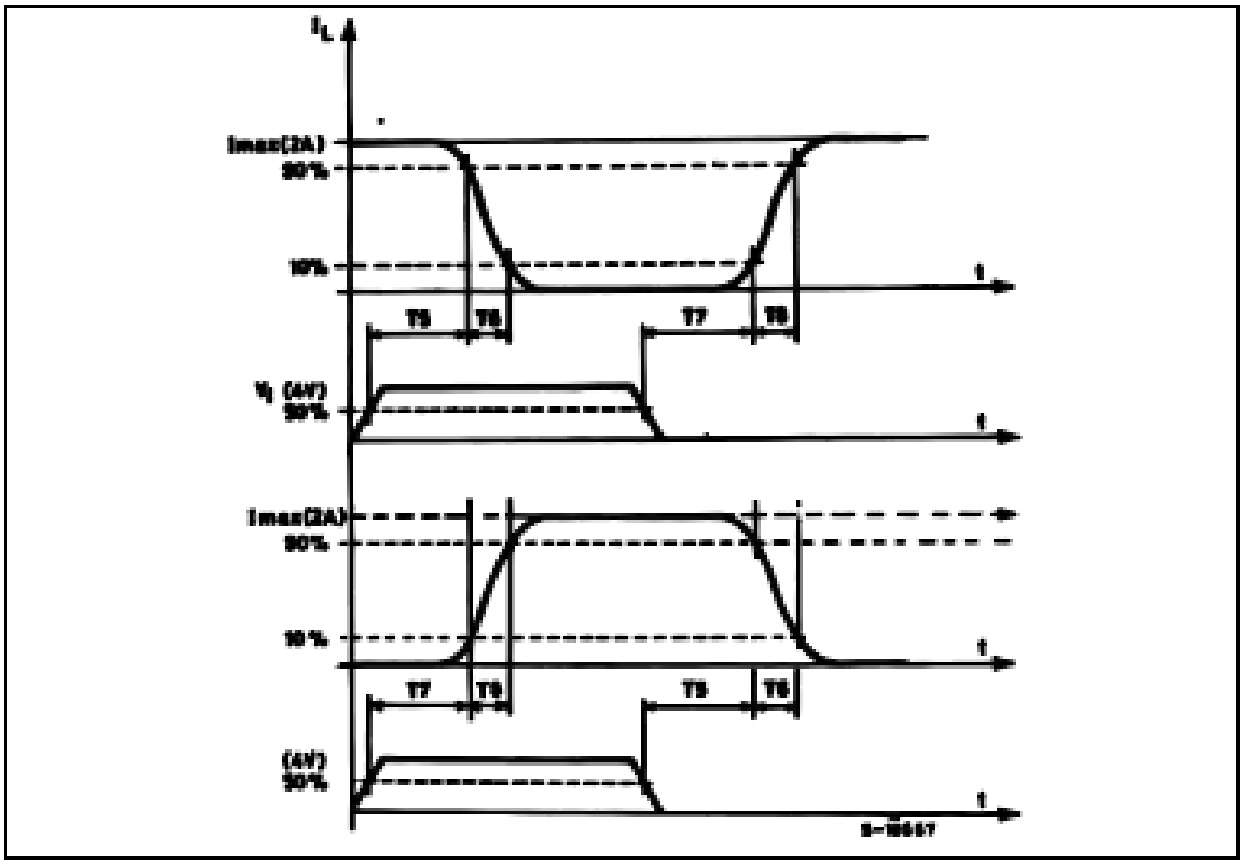
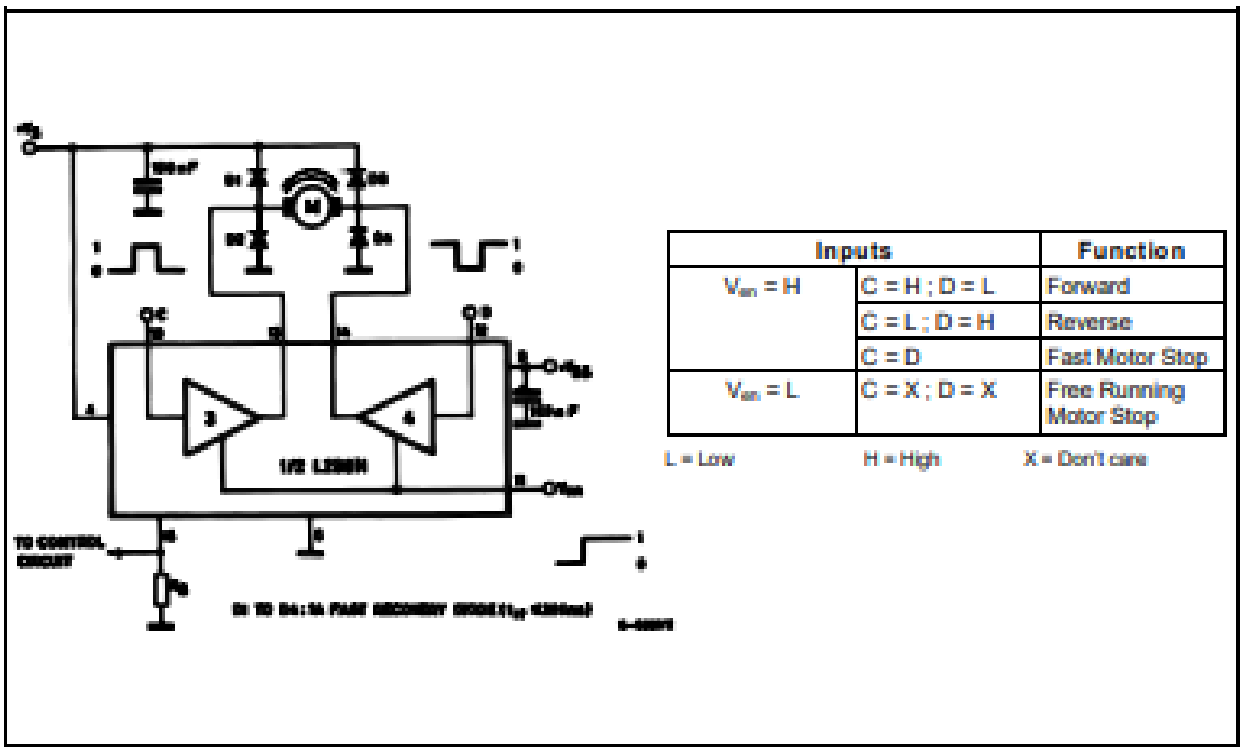


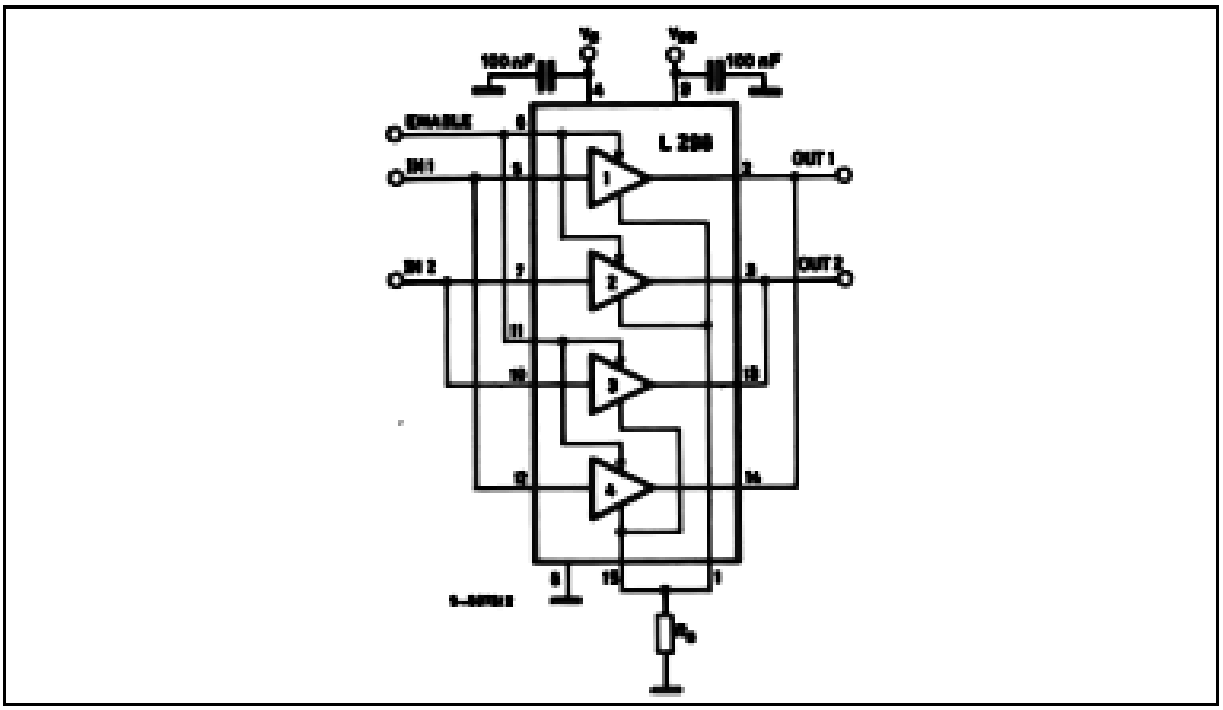
Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Figure 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



APPLICATION INFORMATION (Refer to the block diagram)

1.1. POWER OUTPUT STAGE

The L298 integrates two power output stages (A ; B). The power output stage is a bridge configuration and its outputs can drive an inductive load in common or differenzial mode, depending on the state of the inputs. The current that flows through the load comes out from the bridge at the sense output : an external resistor (R_{SA} ; R_{SB}.) allows to detect the intensity of this current.

1.2. INPUT STAGE

Each bridge is driven by means of four gates the input of which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. The In inputs set the bridge state when The En input is high ; a low state of the En input inhibits the bridge. All the inputs are TTL compatible.

2. SUGGESTIONS

A non inductive capacitor, usually of 100 nF, must be foreseen between both Vs and Vss, to ground, as near as possible to GND pin. When the large capacitor of the power supply is too far from the IC, a second smaller one must be foreseen near the L298.

The sense resistor, not of a wire wound type, must be grounded near the negative pole of Vs that must be near the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements (trr ≤ 200 nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped ; Shottky diodes would be preferred.

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.

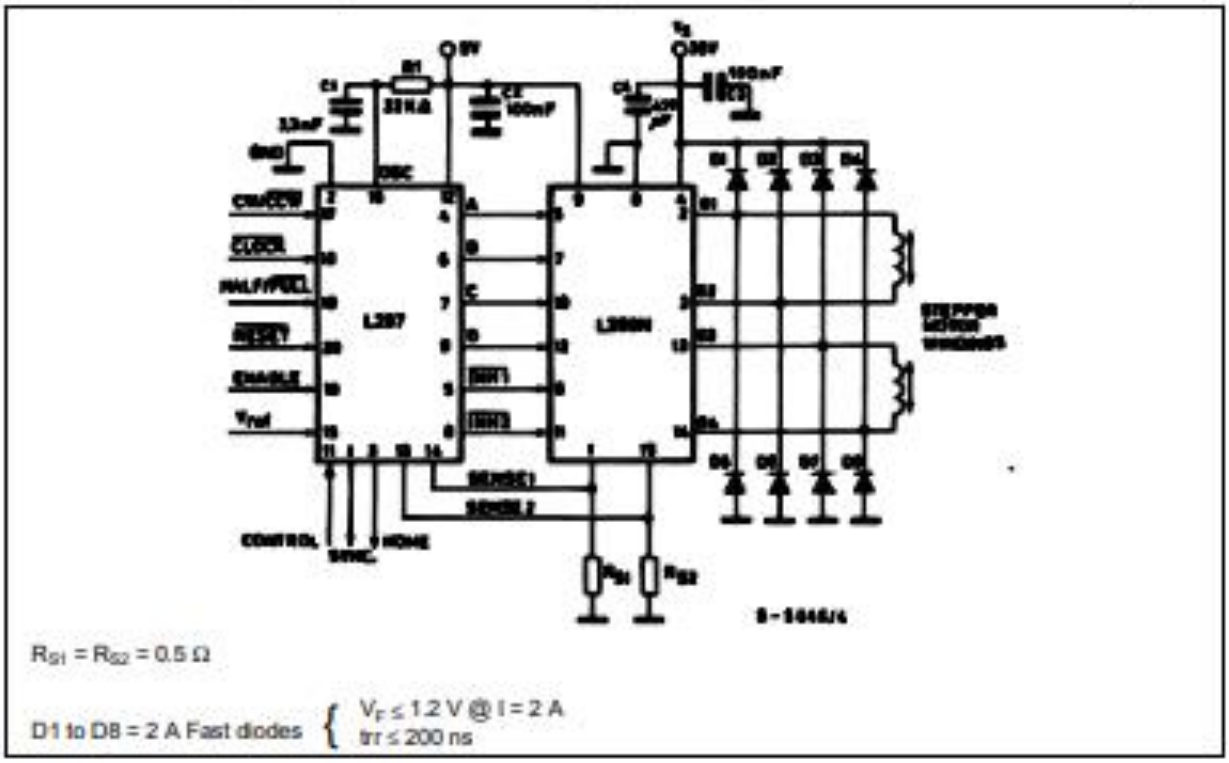


Figure 9 : Suggested Printed Circuit Board Layout for the Circuit of fig. 8 (1:1 scale).

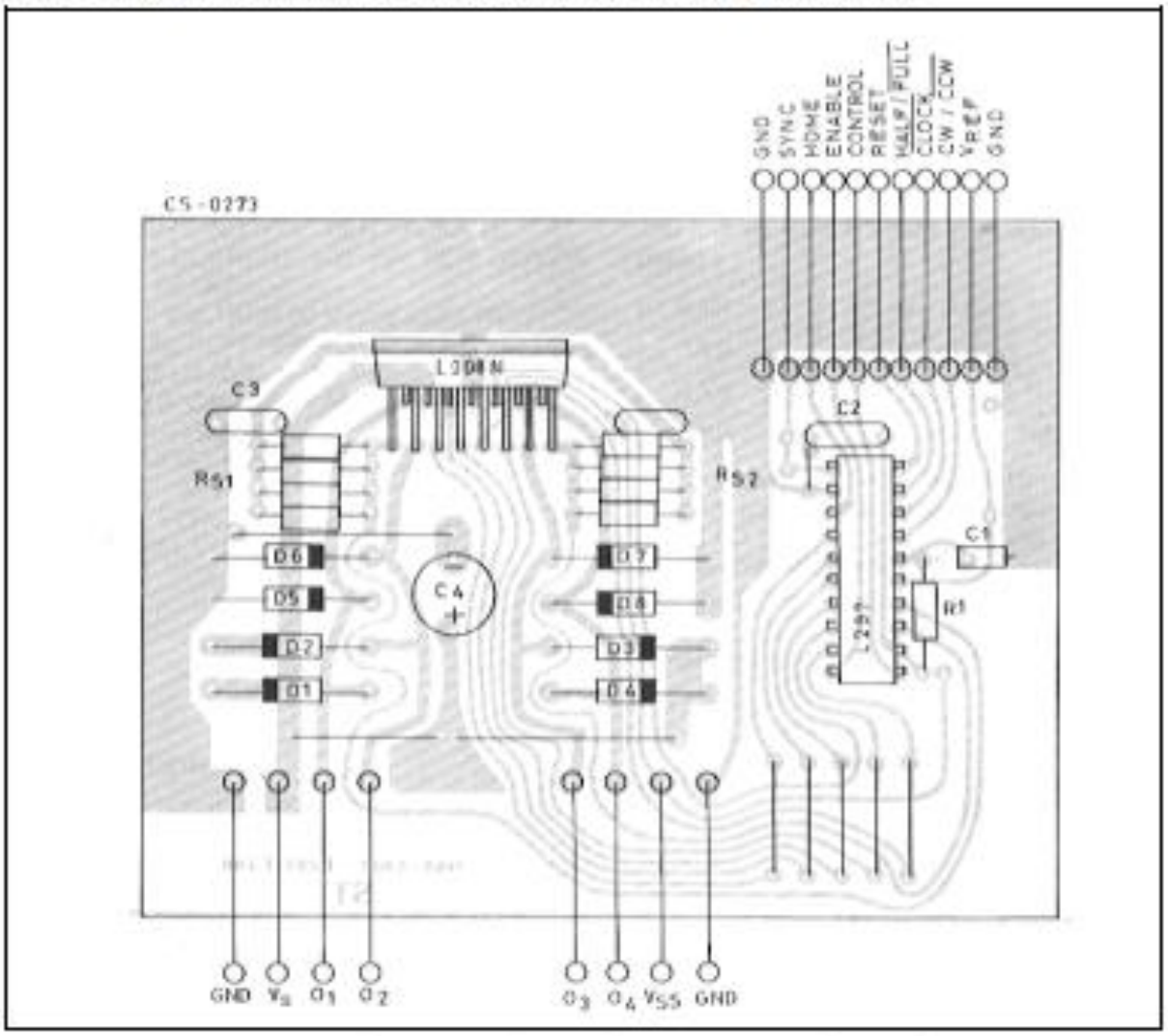
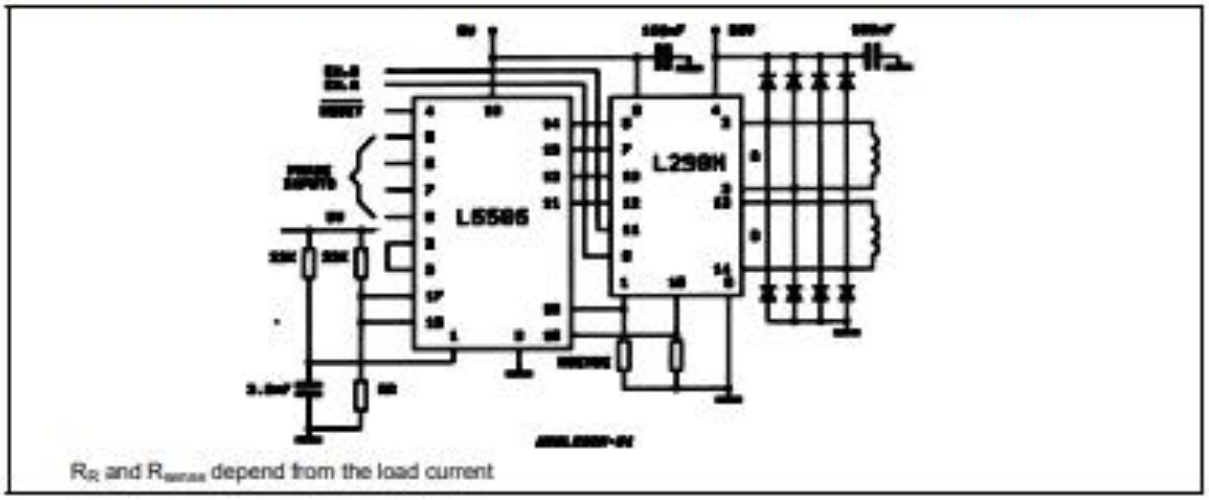
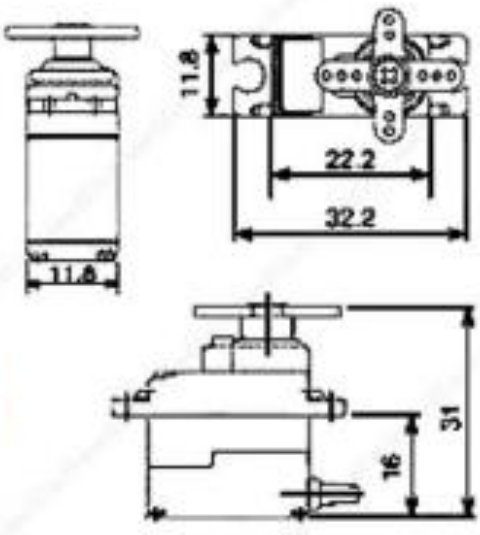


Figure 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.



1.3. Datasheet micro servomotor sg90 TowerPro

SG90 9 g Micro Servo



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but *smaller*. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

Specifications

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kgf-cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Dead band width: 10 µs
- Temperature range: 0 ºC – 55 ºC

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is all the way to the left. ms pulse) is all the way to the right, ""-90" (~1ms pulse) is all the way to the left.

TowerPro SG90 - Micro Servo

Basic Information

Modulation: Analog

Torque: **4.8V:** 25.0 oz-in (1.80 kg-cm)

Speed: **4.8V:** 0.10 sec/60°

Weight: 0.32 oz (9.0 g)

Dimensions:

Length: 0.91 in (23.1 mm)

Width: 0.48 in (12.2 mm)

Height:1.14 in (29.0 mm)

Motor Type: 3-pole
Gear Type: Plastic
Rotation/Support: Bushing

Additional Specifications

Rotational Range: 180°
Pulse Cycle: ca. 20 ms
Pulse Width: 500-2400 µs

2. CÓDIGO FUENTE

Se incluyen a continuación los códigos fuentes completos realizados tanto en Matlab como en la programación de la placa Arduino.

2.1. Función cinemática directa Matlab

Se incluye a continuación el código completo de las funciones necesarias para realizar la cinemática directa en Matlab.

```
function [posicion,orientacion] = DirectKinematics(tabla, in,articulacion,tiempo)
   % Se extrae del vector in el numero de articulaciones
   [ntimes,njoints]=size(in);
   for t=1:ntimes
      t0=clock;
      % Inicializa la matriz de transformacion
      T(:,:,t) = eye(4);
      if t>1
         %Cuando pasa al siguiente paso de tiempo, elimina lo hallado en el anterior
%          delete(hlink);
      end
      for i=1:njoints
         % Se calcula la matriz de Denavit-Hartenberg en funcion de si si la
         % articulacion  es rotacional o prismatica
         if articulacion(i)==1 % Rotacional
            T(:,:,t) = T(:,:,t)*DH(tabla(i,1)+in(t,i),tabla(i,2),tabla(i,3),tabla(i,4));
         elseif articulacion(i)==0 % Prismatica
            T(:,:,t) = T(:,:,t)*DH(tabla(i,1),tabla(i,2)+in(t,i),tabla(i,3),tabla(i,4));
         end
```

```
        DrawJoints(tabla,T(:,:,t),i,in);
    end
    % Punto 3D de la herramienta
    P3D=T(1:3,4,t);
          hproj(1)=plot3([P3D(1) P3D(1)],[P3D(2) P3D(2)],[0 P3D(3)],'c--','LineWidth',3);
    hproj(2)=plot3([P3D(1) P3D(1)],[0 P3D(2)],[0 0],'c--','LineWidth',2);
    hproj(3)=plot3([0 P3D(1)],[P3D(2) P3D(2)],[0 0],'c--','LineWidth',2);
    % Dibuja el recorrido de la posicion de la pinza desde el punto anterior
    if not(t==1)
       plot3([P3D(1) P3Dold(1)],[P3D(2) P3Dold(2)],[P3D(3) P3Dold(3)],'k-','LineWidth',5);
    end
    % Guardamos el punto 3D de la herramienta en el paso anterior para
    % usarlo cuando halla que dibujar el recorrido de la herramienta en el paso siguiente
    P3Dold=P3D;
    % Se extraen los valores de la posicion de la herramienta y se devuelven como salida de la
    % funcion
    posicion = [T(1,4);T(2,4);T(3,4)];
    orientacion = [T(1,1:3);T(2,1:3);T(3,1:3)];
  end
  %Pausa el programa el tiempo que le hemos establecido entre cada paso de tiempo
  if t<ntimes
    pause(tiempo-etime(clock,t0));
  end
end
```

```
% FUNCION DH
% Proporciona el valor de la matriz de transformacion para la articulación deseada
function T= DH(theta,d,a,alpha)
   T=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha) a*cosd(theta);
     sind(theta) cosd(theta)*cosd(alpha) -cosd(theta)*sind(alpha) a*sind(theta);
     0 sind(alpha) cosd(alpha) d;    0 0 0 1];
end
% FUNCION DRAWJOINTS
% Establece los valores necesarios para representar el manipulador
function DrawJoints(tabla,T,number,in)
   % Numero de articulaciones
   njoints=size(tabla);
   % Limites maximos del plot (rep grafica)
   maxlim=2.5*max(vertcat(tabla(:,2),tabla(:,3)));
   % Valor para dibujar la herramienta
   delta = 0.5*tabla(njoints(2),2);
   % Pinta los links del robot
   if number<njoints(2)
      % Punto inicial del link
      Pai(number,:)=T*[-tabla(number,3),0,0,1]';
      % Posicion final del link
      Oi(number,:)=T*[0,0,0,1]';
      if number==1
         RectFill3([0 0 0],[20 0 0],20,tabla(1,2),0)
         RectFill3(Pai(number,:),Oi(number,:),20,20,0)
```

```
        RectFill3(Oi(number,:),[Oi(number,1)+20 Oi(number,2) Oi(number,3)],20,tabla(2,2),0)
    elseif number ==2
        RectFill3(Pai(number,:),Oi(number,:),20,20,0)
    else
        RectFill3(Pai(number,:),[Pai(number,1)+20 Pai(number,2)
        Pai(number,3)],20,((tabla(1,2)+tabla(2,2))-Oi(number,3)+20),0)
    End
else
    % Posicion final de la herramienta
    Pcon=T*[0,0,-delta,1]';
    % Dibujamos la herramienta
    RectFill3(Pcon,[Pcon(1)+20 Pcon(2) Pcon(3)],20,Pcon(3,1)-T(3,3),0)
    %Determinamos los puntos de la herramienta para poder dibujarla
    P1=T*[0,delta,-delta,1]';
    P2=T*[0,-delta,-delta,1]';
    P3=T*[0,delta,0,1]';
    P4=T*[0,-delta,0,1]';
    %Parte larga
    if in(1,4)==0
        RectFill3(P1,[P1(1,1)+20 P1(2,1) P1(3,1)],P2(2,1)-P1(2,1)+20,10,0)
    else
        RectFill3(P1,P2,20,10,1)
    end
    %Pinza izq
    RectFill3(P4,[P4(1,1)+20 P4(2,1) P4(3,1)],20,P1(3,1)-P4(3,1),0)
```

```
    %Pinza drch
    RectFill3(P3,[P3(1,1)+20 P3(2,1) P3(3,1)],20,delta,0)
  end
  axis([-maxlim maxlim -maxlim maxlim 0 maxlim]);
end
% FUNCION RECTFILL3
% Representa la estructura del manipulador en forma de rectangulos
function RectFill3(Pai, Oi, ancho, alto,n)
%tabla = [0 257 78.5 0; 0 27 80 180; 0 140 0 0; 0 59 0 0];
  if n==1
    % Coordenadas vertices
    % Dimension 1x8
    x = [Pai(1) Pai(1)    Oi(1)    Oi(1)  Pai(1)      Pai(1)    Oi(1)     Oi(1)    ];
    y = [Pai(2) Pai(2)    Oi(2)    Oi(2)  Pai(2)+ancho  Pai(2)+ancho Oi(2)+ancho  Oi(2)+ancho];
    z = [Pai(3) Pai(3)+alto Oi(3)+alto  Oi(3)  Pai(3)      Pai(3)+alto  Oi(3)      Oi(3)+alto ];
    % Caras
    index(1,:) = [1 2 3 4 1];
    index(2,:) = [5 6 7 8 5];
    index(3,:) = [1 2 6 5 1];
    index(4,:) = [4 3 7 8 4];
    index(5,:) = [2 6 7 3 2];
    index(6,:) = [1 5 8 4 1];
    % Bucle que pinta las caras de las estructuras (rectangulos)
    for k = 1:6
      fill3(x(index(k,:)), y(index(k,:)), z(index(k,:)),'b');
```

```
        hold on;
    end
    xlabel('x')
    ylabel('y')
    zlabel('z')
  else
    % Coordenadas vértices % Dimension 1x8
    x = [Pai(1) Pai(1)     Pai(1)     Pai(1)     Oi(1) Oi(1)     Oi(1)     Oi(1)];
    y = [Pai(2) Pai(2)     Pai(2)+ancho Pai(2)+ancho  Oi(2) Oi(2)     Oi(2)+ancho Oi(2)+ancho];
    z = [Pai(3) Pai(3)+alto Pai(3)+alto  Pai(3)       Oi(3) Oi(3)+alto Oi(3)+alto  Oi(3)];
    % Cara
    index(1,:) = [1 2 3 4 1];  index(2,:) = [5 6 7 8 5];
    index(3,:) = [1 2 6 5 1];  index(4,:) = [4 3 7 8 4];
    index(5,:) = [2 6 7 3 2];   index(6,:) = [1 5 8 4 1];
    % Bucle que pinta las caras de las estructuras (rectangulos)
    for k = 1:6
        fill3(x(index(k,:)), y(index(k,:)), z(index(k,:)),'b');
        hold on;
    end
    xlabel('x')
    ylabel('y')
    zlabel('z')
  end
end
```

2.2. Función cinemática inversa Matlab

Se incluye a continuación el código completo de las funciones necesarias para realizar la cinemática inversa en Matlab.

```
function values = InverseKinematics(tabla,posicion,orientacion,codo)

  % Se halla la matriz de transformacion total ya que posteriormente sera de utilidad

  R0_4=rotz(orientacion(1))*roty(0)*rotx(180);

  % Se guardan en variables los datos necesarios de la tabla D-H

  d1 = tabla(1,2);   d2 = tabla(2,2);

  d3 = tabla(3,2);   d4 = tabla(4,2);

  a1 = tabla(1,3);

  a2 = tabla(2,3);

  Rx = posicion(1);

  Ry = posicion(2);

  Rz = posicion(3);

  values = 0;

  % Se determina el valor del coseno de la segunda articulacion

  D = (Rx^2+Ry^2-a1^2-a2^2)/(2*a1*a2);

  % Calculo segunda articulacion

  if codo == 1

    values(1) = atan2d(Ry,Rx) - atan2d(a2*sqrt(1-D^2),a1+a2*D);

    values(2) = atan2d(sqrt(1-D^2),D);

  else

    values(1) = atan2d(Ry,Rx) + atan2d(a2*sqrt(1-D^2),a1+a2*D);

    values(2) = atan2d(-sqrt(1-D^2),D);

  end
```

```
% Se calcula el valor de la articulacion prismatica
   values(3) = d1+d2-d3-d4-Rz;
   % Se calcula la matriz de transformacion desde la base referida a la muñeca
   T0_1=DH(values(1),tabla(1,2),tabla(1,3),tabla(1,4));
   T0_2=T0_1*DH(values(2),tabla(2,2),tabla(2,3),tabla(2,4));
   T0_3=T0_2*DH(tabla(3,1),values(3)+tabla(3,2),tabla(3,3),tabla(3,4));
   R3_4=T0_3(1:3,1:3)\R0_4(1:3,1:3);
   % values(4) = values(1)+values(2)-orientacion(1);
   values(4)=atan2d(R3_4(2,1),R3_4(1,1))
    end
```

2.3. Función Control Cinemático

Se incluye a continuación el código completo de las funciones necesarias para realizar el control cinemático en Matlab.

```
function JTs=KinematicControl(table, Init_pose, Final_pose, Time, NCrossPoints, NSampPoints)
% Numero de articulaciones
Njoints = size(table,1);
% Se define el tiempo para cada intervalo
inct = Time/(NCrossPoints+1);
%**** Puntos de cruce en el espacio cartesiano y en el espacio conjunto ****
%*************************************************************************
% Se inicializa la variable la cual se llenara con los puntos de cruce
% el espacio de unión (incluyendo los puntos iniciales y finales, las dimensiones de
Jvals = zeros(NCrossPoints+2, Njoints);
```

```
% Cada punto de cruce se añadirá a la variable 'jvals' mediante un bucle:

for k = 1:NCrossPoints+2

   % Se aplica la interpolacion lineal en el espacio cartesiano

   pose_k = Init_pose+(Final_pose-Init_pose)/Time*(k-1)*inct;

   % Se rellenara la variable de los puntos de cruce en el espacio

   % articular mediante la funcion de cinematica inversa

   Jvals(k,:) = InverseKinematics(table, pose_k(1:3)',pose_k(4:6),1);

   % Se comprueba si los valores añadidos se encuentran dentro del rango

   % que se ha establecido

   if Jvals(k,1)>90 || Jvals(k,1)<-90

      fprintf('No se puede alcanzar el ángulo 1 = %0.2f \n',Jvals(k,1))

      return

   elseif Jvals(k,2)>90 || Jvals(k,2)<-90

      fprintf('Cannot reach the angle 2 = %0.2f \n',Jvals(k,2))

      return

   elseif Jvals(k,3)>50 || Jvals(k,3)<0

      fprintf('Cannot reach the angle 3 = %0.2f \n',Jvals(k,3))

      return

   elseif Jvals(k,4)>90 || Jvals(k,4)<-90

      fprintf('Cannot reach the angle 4 = %0.2f \n',Jvals(k,4))

      return

   end

end

%*********************************************************************

%********** Interpolación de las trayectorias de cada articulación *******
```

```
% Se calculan el numero totales de puntos necesarios

NPoints = (NCrossPoints+1)*(NSampPoints+1)+1;

% Se inicializa la variable de salida, la cual contendra a todas las

% trayectorias conjuntas.

JTs = zeros(NPoints, Njoints);

% Las posiciones resultantes de la articulacion cubica se añaden

% articulacion por articulacion a la variable de salida.

for n = 1:NJoints

   % Se preparan las variables de entrada de la funcion que realiza la

   % interpolacion cubica

   DATA = [(0:inct:Time); Jvals(:,n)']';

   JTs(:,n) = InterpoladorCubico(DATA,inct,NSampPoints,n);

% End of the loop

end

%*********************************************************************

% End of the function

end

% FUNCION de interpolador cubico

% interpola una trayectoria conjunta utilizando N puntos de cruce en DATA y

% el tiempo por intervalo y devuelve NSampPoints en la variable POS.

function POS=InterpoladorCubico(DATA,inct,NSampPoints,n)

   % Inicializa los vectores de posicion, velocidad y aceleracion

   POS=[]; SPD=[]; ACL=[]; TIME=[];

   % Numero de tramos

   Nints=size(DATA,1)-1;
```

```
% Numero de puntos
Sints=NSampPoints+1;
% Se hallan las restriccion en cada tramo de forma que se usen estas para
% determinar los coeficientes de las polinomiales de interpolacion.
% Para cada tramo que es evaluado hallar las veloccidades (pendientes)  (inicial y final)
for j=1:Nints
    % Se calcula la velocidad inicial y final de cada tramo
    % Estas van a depender de las pendientes previas, actuales y
finales.
    if j==1 % Tramo inicial
        %Velocidad en el tramo inicial
        Vini=0;
        %Velocidad (pendiente) en el tramo actual
        Vcurr=(DATA(j+1,2)-DATA(j,2))/(DATA(j+1,1)-DATA(j,1));
        %Velocidad (pendiente) en el tramo siguiente
        Vnext=(DATA(j+2,2)-DATA(j+1,2))/(DATA(j+2,1)-DATA(j+1,1));
        % Se comprueba el signo de las velocidades (pendiente entre dos puntos)
        % Si el signo de la velocidad actual coincide con el de la
        % velocidad siguiente, la velocidad final sera la media de las dos velocidades
        if sign(Vcurr)==sign(Vnext)
            Vfin=(Vcurr+Vnext)/2;
        else % De lo contrario la velocidad final sera 0
            Vfin=0;
        end
```

```
elseif j<Nints % Tramos que no son ni el inicial ni el ultimo
  % Velocidad (pendiente) en el tramo anterior
  Vprev=(DATA(j,2)-DATA(j-1,2))/(DATA(j,1)-DATA(j-1,1));
  % Velocidad (pendiente) en el tramo actual
  Vcurr=(DATA(j+1,2)-DATA(j,2))/(DATA(j+1,1)-DATA(j,1));
  % Velocidad (pendiente) en el tramo siguiente
  Vnext=(DATA(j+2,2)-DATA(j+1,2))/(DATA(j+2,1)-DATA(j+1,1));
  % Se comprueba el signo de las velocidades (pendiente entre dos puntos)
  % Si el signo de la velocidad actual coincide con el de la
  % velocidad previa, la valocidad inicial sera la media de las dos velocidades
  if sign(Vcurr)==sign(Vprev)
    Vini=(Vcurr+Vprev)/2;
  else
  % De lo contrario la velocidad inicial del tramo sera 0
    Vini=0;
  end
  % Se comprueba el signo de las velocidades (pendiente entre dos puntos)
  % Si el signo de la velocidad actual coincide con el de la
  % velocidad siguiente, la valocidad final sera la media de las dos velocidades
  if sign(Vcurr)==sign(Vnext)
    Vfin=(Vcurr+Vnext)/2;
  else
  % De lo contrario la velocidad inicial del tramo sera 0
    Vfin=0;
  end
```

```
    else % Tramos final
        %La velocidad final es 0 ya que estamos en el ultimo tramo
        Vfin=0;
        % Velocidad (pendiente) en el tramo anterior
        Vprev=(DATA(j,2)-DATA(j-1,2))/(DATA(j,1)-DATA(j-1,1)); % Previous interval speed
        % Velocidad (pendiente) en el tramo actual
        Vcurr=(DATA(j+1,2)-DATA(j,2))/(DATA(j+1,1)-DATA(j,1)); % Current interval speed
        % Se comprueba el signo de las velocidades (pendiente entre dos puntos)
        % Si el signo de la velocidad actual coincide con el de la
        % velocidad previa, la valocidad final sera la media de las
        % dos velocidades
        if sign(Vcurr)==sign(Vprev)
            Vini=(Vcurr+Vprev)/2;
        else
        % De lo contrario la velocidad inicial del tramo sera 0
            Vini=0;
        end
    end
    % Intervalo de tiempo en cada tramo
    T=(DATA(j+1,1)-DATA(j,1));
    % Se hallan los coeficientes en cada tramo
    % Coeficiente hallado por la restriccion de la posicion inicial
    a0=DATA(j,2);
    % Coeficiente hallado por la restriccion de la velocidad inicial
    a1=Vini;
```

```
% Coeficientes hallados mediante los polinomios para la interpolación de la velocidad y posicion
    A=[T^2 T^3;2*T 3*T^2];
    B=[DATA(j+1,2)-a0-a1*T; Vfin-a1];
    X=A\B;
    a2=X(1); a3=X(2);
    % Se halla el tiempo de simulacion en el tramo en el que esta
    if j==1 % Primer tramo
       ts=0:T/Sints:T;
    else %Tramos siguientes
       ts=T/Sints:T/Sints:T;
    end
    % Interpolaciones polinomiales en el tramo
    % Posicion
    POS=[POS,a0+a1*ts+a2*ts.^2+a3*ts.^3];
    % Velocidad
    SPD=[SPD,a1+2*a2*ts+3*a3*ts.^2];
    % Aceleracion
    ACL=[ACL,2*a2+6*a3*ts];
    % Tiempo
    TIME=[TIME,ts+DATA(j,1)];
end
% Se dibujan las interpolaciones
% Posicion
figure;
subplot(3,1,1);
```

```
 % Pinta en el plot el valor de la posicion de los N  valores que definen la trayectoria
stem(DATA(:,1),DATA(:,2)); grid on; hold on;
% Pinta la trayectoria de los puntos
plot(TIME,POS,'.g');
xlabel('Time in secs')
ylabel('Degrees');
title(['Angular Position of the Joint ',num2str(n)]);
% Velocidad
subplot(3,1,2);
% Velocidad haciendo la derivada de la posicion
SPDn=[0 diff(POS)/(inct/(NSampPoints+1))];
plot(TIME,SPDn,'b'); hold on;
plot(TIME,SPD,'g'); grid on;
xlabel('Time in secs')
ylabel('Degrees/sec');
title(['Angular Speed of the Joint ',num2str(n)]);
% Aceleracion
subplot(3,1,3);
% Aceleracion haciendo la derivada de la velocidad
ACLn=diff([SPDn,0])/(inct/(NSampPoints+1));
plot(TIME,ACLn,'r'); hold on;
plot(TIME,ACL,'m'); grid on;
xlabel('Time in secs'); ylabel('Degrees/seg^2');
title(['Angular Acceleration of the Joint ',num2str(n)]);
end
```

2.4. Funcion SimuladorSCARA

```
function SimuladorSCARA()
 % Se pide al usuario que introduzca los datos
 disp('Rellena los datos necesarios')
 NCrossPoints = input('Numero de puntos de cruce: ') ;
 % Se comprueba si el dato es correcto
 if ((NCrossPoints>6)||(NCrossPoints<0))
    disp('/***** ERROR ****/')
    disp('/*DATO NO VALIDO*/')
 else
    NSampPoints = input('Numero de puntos de muestreo: ');
    % Se comprueba si el dato es correcto
    if ((NSampPoints>=5)||(NSampPoints<0))
      disp('/***** ERROR ****/')
      disp('/*DATO NO VALIDO*/')
    else
      Time = input('Tiempo que dura la trayectoria: ');
      disp('Configuracion estructura manipulador')
      codo = input('Codo derecha (1) - Codo izquierda (0): ');
      disp('Punto final de la trayectoria')
      a = input('Coordenada X: ');
      b = input('Coordenada Y: ');
      c = input('Coordenada Z: ');
      d = input('Coordenada alpha: ');
      disp('Modo funcionamiento:');
```

```
disp('(1)Led Verde - Modo 1: Eslabones cortos');
disp('(2)Led Azul - Modo 2: Eslabones largos');
disp('(3)Led Amarillo - Modo 3: Eslabon1 corto - Eslabon2 largo');
disp('(4)Led Blanco - Modo 4: Eslabon1 largo - Eslabon2 corto');
modo = input('Modo: ');
% Se comprueba si los datos son correctos
if ((modo<1)||(modo>4))
   disp('/************** ERROR **************/')
   dips('/*MODO DE FUNCIONAMIENTO INCORRECTO*/')
elseif ((codo<0)||(codo>1))
   disp('/************** ERROR **************/')
   dips('/*CONFIGURACION DEL ROBOT INCORRECTO*/')
else
   % Se establecen los valores de los parametros del modo
   % seleccionado
   if modo==1
      parametro1 = 78.5;
      parametro2 = 81;
   elseif modo ==2
      parametro1 = 97.5;
      parametro2 = 100;
   elseif modo ==3
      parametro1 = 78.5;
      parametro2 = 100;
   elseif modo ==4
```

```
            parametro1 = 97.5;
            parametro2 = 81;
        else
        end
        % Se preparan los datos para introducirlos en la funcion de
        % control cinematico
        tabla = [0 257 parametro1 0; 0 27 parametro2 180; 0 140 0 0; 0 59 0 0];
        Init_pose = [158.5 0 85 0 0 180];
        Final_pose= [a b c d 0 180];
        % Control cinematico
        JTs=KinematicControl(tabla, Init_pose, Final_pose, Time, NCrossPoints,
NSampPoints,codo);
        pause(2)
        % Se preparan los datos para introducirlos en la funcion de
        % cinematica directa
        articulacion = [1 1 0 1];
        pause(2)
        % Cinematica directa
        [posicion, orientacion]=DirectKinematics(tabla,JTs,articulacion,Time/(size(JTs,1)-1));
        % Primer punto de la trayectoria
        posicion(:,:,1)
        orientacion(:,:,1)
        % Ultimo punto de la trayectoria
        posicion(:,:,end)
        orientacion(:,:,end)      end     end  end   end
```

2.5. Sketch Arduino

Se incluye a continuación el código fuente completo del sketch implementado para realizar el control del manipulador.

```
//Bibliotecas necesarias
#include <Servo.h>
//Se establecen los valores de los pines
/*Leds*/
 int LED_M1 = 2; //Led verde modo 1
 int LED_M2 = 3; //Led azul modo 2
 int LED_M3 = 4; //Led amarillo modo 3
 int LED_M4 = 5; //Led blanco modo 4
/*Pines L298N Motor A*/
 int ENA = 6; /*PWM*/
 int IN1 = 7;
 int IN2 = 8;
  /*Variables objeto de clase "Servo"*/
  Servo servo1; //Hombro
  Servo servo2; //codo
  Servo servo3; //Muñeca
  Servo servo4; //Pinzas
//Resultado Cinematica inversa
float q[4];
//Resultado Control cinematico
float Jvals[4][32];
//Variable para salir del while()
int VAL;
void InicioServos(){
  /*Se posicionan los servos en el estado de reposo 90º*/
```

```
  servo1.write(90);
  servo2.write(90);
  servo3.write(90);
  servo4.write(88); /*Pinza cerrada*/
  /*Si el actuador lineal esta extendido, lo cierra*/
  digitalWrite (IN1, LOW);
  digitalWrite (IN2, HIGH);
  delay(3000);
}
void MoveServos(float Tespera,float dataV1,float dataV2,float dataV3,float
dataV4, int np,int fb, float Time,int vFinal){
  int espera = Tespera*1000;
  int espera = Tespera*1000;
  int tvelocidad;
  int velocidad;
  if(np==0){
    servo4.write(0);
    delay(1000);  }
 tvelocidad = (5.3*vFinal)/50;
 velocidad =(255*tvelocidad)/Time;
  servo1.write(90+dataV1);
  servo2.write(90+dataV2);
  servo3.write(90+dataV4);
  if(dataV3>0){
    digitalWrite (IN1, HIGH);
    digitalWrite (IN2, LOW);
    analogWrite (ENA, velocidad);
  }
```

```
  delay(espera);
  if(np==(fb-1)){
    /*cierra las pinzas*/
    servo4.write(88);
    delay(1000);
  }
}
float CinematicaInversa(float Param[],float pos[], float ori, int c){
  //Variables locales
  float tabla[4][4] =
{{0,257,Param[0],0},{0,27,Param[1],180},{0,140,0,0},{0,59,0,0}};
  float pi = 3.14159265358979323846;
  float d1 = tabla[0][1];
  float d2 = tabla[1][1];
  float d3 = tabla[2][1];
  float d4 = tabla[3][1];
  float a1 = tabla[0][2];
  float a2 = tabla[1][2];
  float Rx = pos[0];
  float Ry = pos[1];
  float Rz = pos[2];
  float D,a;
  //Valor del coseno de la segunda articualcion
  D = (pow(Rx,2)+pow(Ry,2)-pow(a1,2)-pow(a2,2))/(2*a1*a2);
  //Calculo de las dos primeras articulaciones en función de la configuracion
del codo
  //(derecha-izquierda)
  if ((1-pow(D,2))<0) {
    a = 0;
```

```
  }else{
    a = 1-pow(D,2);
  }
  if (c == 1){
    q[0] = (atan2(Ry,Rx)-atan2(a2*(sqrt(a)),a1+a2*D))*180/pi;
    q[1] = (atan2(sqrt(a),D))*180/pi;
  }else{
    q[0] = (atan2(Ry,Rx)+atan2(a2*(sqrt(a)),a1+a2*D))*180/pi;
    q[1] = (atan2(-sqrt(a),D))*180/pi;
  };
  //Se calcula el valor de la articulacion prismatica
  q[2] = d1+d2-d3-d4-Rz;
  //Calculo de la ultima articulacion
  q[3] = q[0]+q[1]-ori;
}
float InterpoladorCubico(float DATA1[],float DATA2[],float inct,int
NSampPoint,int n,int NCP){
  //Variables locales
  /*Inicializa los vectores de velocidad y aceleracion*/
  /*Velocidades inicial,final,anterios,actual y posterior*/
  float Vini,Vfin,Vprev,Vcurr,Vnext;
  float T,a0,a1,a2,a3;
  float X[2][1],ts;
  float A[2][2];
  float B[2][1];
  float determinante;
  float adjunta[2][2];
  float TrasA[2][2];
  float InvA[2][2];
```

```
/*Numero de tramos*/
int Nints = NCP+2;
/*Numero de puntos*/
int Sints = NSampPoint+1;
//Restricciones en cada tramo
for (int j=0;j<Nints;j++){
  /*Se calcula la velocidad inicial y final de cada tramo
  Estas van a depender de las pendientes previas, actuales
  y finales.*/
  if(j==0){
    /*Velocidad en el tramo inicial*/
    Vini=0;
    /*Velocidad (pendiente) en el tramos actual*/
    Vcurr=(DATA2[j+1]-DATA2[j])/(DATA1[j+1]-DATA1[j]);
    /*Velocidad (pendiente) en el tramo siguiente*/
    Vnext=(DATA2[j+2]-DATA2[j+1])/(DATA1[j+2]-DATA1[j+1]);
    /*Se comprueba el signo de las velocidades (pendiente entre dos
    puntos). Si el signo de la velocidad actual coincide con el de
    lavelocidad siguiente, la velocidad final sera la media de las
    dos velocidades*/
    if(((Vcurr<=0)&&(Vnext<=0))||((Vcurr>0)&&(Vnext>0))){
      Vfin=(Vcurr+Vnext)/2;
    }else{
      Vfin=0;
    };
  }else if(j<Nints){
    /*Velocidad (pendiente) en el tramo anterior*/
    Vprev=(DATA2[j]-DATA2[j-1])/(DATA1[j]-DATA1[j-1]);
```

```
      /*Velocidad (pendiente) en el tramo actual*/
      Vcurr=(DATA2[j+1]-DATA2[j])/(DATA1[j+1]-DATA1[j]);
      /*Velocidad (pendiente) en el tramo siguiente*/
      Vnext=(DATA2[j+2]-DATA2[j+1])/(DATA1[j+2]-DATA1[j+1]);
      /*Se comprueba el signo de las velocidades (pendiente entre dos
      puntos). Si el signo de la velocidad actual coincide con el de
      la velocidad previa, la valocidad inicial sera la media de las
      dos velocidades*/
      if(((Vcurr<=0)&&(Vprev<=0))||((Vcurr>0)&&(Vprev>0))){
        Vini=(Vcurr+Vprev)/2;
      }else{
        Vini=0;
      };
      /*Se comprueba el signo de las velocidades (pendiente entre dos
      puntos). Si el signo de la velocidad actual coincide con el de
      lavelocidad siguiente, la velocidad final sera la media de las
      dos velocidades*/
      if(((Vcurr<=0)&&(Vnext<=0))||((Vcurr>0)&&(Vnext>0))){
        Vfin=(Vcurr+Vnext)/2;
      }else{
        Vfin=0;
      };
    }else{
      /*La velocidad final es 0 ya que estamos en el ultimo tramo*/
      Vfin=0;
      /*Velocidad (pendiente) en el tramo anterior*/
      Vprev=(DATA2[j]-DATA2[j-1])/(DATA1[j]-DATA1[j-1]);
      /*Velocidad (pendiente) en el tramo actual*/
```

```
    Vcurr=(DATA2[j+1]-DATA2[j])/(DATA1[j+1]-DATA1[j]);
    /*Se comprueba el signo de las velocidades (pendiente entre dos
    puntos). Si el signo de la velocidad actual coincide con el de
    la velocidad previa, la valocidad inicial sera la media de las
    dos velocidades*/
    if(((Vcurr<=0)&&(Vprev<=0))||((Vcurr>0)&&(Vprev>0))){
      Vini=(Vcurr+Vprev)/2;
    }else{
      Vini=0;
    };
  };
  /*Intervalo de tiempo en cada tramo*/
  T = (DATA1[j+1]-DATA1[j]);
  //Se hallan los coeficientes en cada tramo
  /*Coeficiente hallado por la restriccion de la posicion inicial*/
  a0 = DATA2[j];
  /*Coeficiente hallado por la restriccion de la velocidad inicial*/
  a1 = Vini;
  /*Coeficientes hallados mediante los polinomios para la interpolación
  de la velocidad y posicion*/
  A[0][0] = pow(T,2);
  A[0][1] = pow(T,3);
  A[1][0] = 2*T;
  A[1][1] = 3*pow(T,2);
  B[0][0] = DATA2[j+1]-a0-a1*T;
  B[1][0] = Vfin-a1;
  determinante = A[0][0]*A[1][1]-A[0][1]*A[1][0];
  adjunta[0][0]=A[1][1];
```

```
    adjunta[1][0]=-A[0][1];
    adjunta[1][1]=A[0][0];
    TrasA[0][0]=adjunta[0][0];
    TrasA[0][1]=adjunta[1][0];
    TrasA[1][0]=adjunta[0][1];
    TrasA[1][1]=adjunta[1][1];
    InvA[0][0]=(1/determinante)*TrasA[0][0];
    InvA[0][1]=(1/determinante)*TrasA[0][1];
    InvA[1][0]=(1/determinante)*TrasA
    InvA[1][0]=(1/determinante)*TrasA[1][0];
    InvA[1][1]=(1/determinante)*TrasA[1][1];
    X[0][0] = (InvA[0][0]*B[0][0])+(InvA[0][1]*B[1][0]);
    X[1][0] = (InvA[1][0]*B[0][0])+(InvA[1][1]*B[1][0]);
    a2 = X[0][0]; a3 = X[1][0];
    if (j==0){
      for(int k=0;k<(Sints+1);k++){
        ts = k*(inct/Sints);
        Jvals[n][k]=a0+a1*ts+a2*pow(ts,2)+a3*pow(ts,3);
      };
   }else{
      for(int k=0;k<Sints;k++){
        ts=(k+1)*(inct/Sints);
        int number;
        number = k+((5*j)+1);
        Jvals[n][number]=a0+a1*ts+a2*pow(ts,2)+a3*pow(ts,3);
   };
  };
 };
```

```
float ControlCinematico(float Param[],float Init_pos[],float Final_pos[],float
Time,int NCrossPoints, int NSampPoints,int codo){
  //Variables locales
  float pos[3],ori;
  float DATA1[20];
  //Datos funcion cinematica inversa
//  float posicion[] = {135.0685,56.5685,85.0000};
//  float orientacion = 45;
  /*Inicializacion de la variable interpolacion lineal*/
  float pose_k[NCrossPoints+2][6];
  /*Numero de articulaciones**/
  int Njoints = 4;
  /*Se define el tiempo para cada intervalo*/
  float inct = Time/(NCrossPoints+1);
  /*Se calculan el numero totales de puntos necesarios*/
  int NPoints = (NCrossPoints+1)*(NSampPoints+1)+1;
  float Jvals1[NCrossPoints+2];
  float Jvals2[NCrossPoints+2];
  float Jvals3[NCrossPoints+2];
  float Jvals4[NCrossPoints+2];
  int i;
  float offset;
  //Puntos de cruce en el espacio cartesiano y en el espacio conjunto//
  /*Cabecero de los datos de las articulaciones*/
  Serial.println("Articulacion:    1        2        3        4");
  delay(100);
  /*Se calculan las posiciones para los tramos*/
  for (int i=0;i<(NCrossPoints+2);i++){
```

```
for (int j=0;j<6;j++){
  pose_k[i][j]=Init_pos[j]+(Final_pos[j]-Init_pos[j])/Time*(i-1)*inct;
};
/*Se guardan en variables locales los valores de la orientacion y
posicion hallados en la operacion anterior*/
pos[0] = pose_k[i][0];
pos[1] = pose_k[i][1];
pos[2] = pose_k[i][2];
ori = pose_k[i][3];
/*Se llama a la funcion cinematica inversa.
Esta devuelve una variable global (q) en la cual se guardan los valores
de las articulaciones para ese punto de cruce*/
CinematicaInversa(Param,pos,ori,codo);
/*Pequeño offset del actuador*/
offset = ((85-Final_pos[2])*(-8.33))/50;


/*Se guardan los valores obtenidos de la cinematica directa en
variables globales para cada articulacion*/
Jvals1[i]=q[0];
Jvals2[i]=q[1];
Jvals3[i]=q[2]-offset;
Jvals4[i]=q[3];
/*Comprobacion de los angulos*/
if (Jvals1[i]>90 || -90>Jvals1[i]){
  Serial.print("Error en angulo = ");
  Serial.println(Jvals1[i]);
  delay(100);
  exit(0);
```

```
    }else if (Jvals2[i]>90 || -90>Jvals2[i]){
      Serial.print("Error en angulo = ");
      Serial.println(Jvals2[i]);
      delay(100);
      exit(0);
    }else if ((Jvals3[i]>50) || (0>Jvals3[i])){
      Serial.print("Error en angulo = ");
      Serial.println(Jvals3[i]);
      delay(100);
      exit(0);
    }else if (Jvals4[i]>180 || -180>Jvals4[i]){
      Serial.print("Error en angulo = ");
      Serial.println(Jvals4[i]);
      delay(100);
      exit(0);
    };
  };
  //Interpolación de las trayectorias de cada articulación//
  /*Se prepara la variable del tiempo total transcurrido para las
  articulaciones*/
  /*Se define el tiempo para cada intervalo*/
  inct = Time/(NCrossPoints+1);
  /*Se preparan las variables que serviran de datos de entrada a la
  funcion del interpolador cubido.
  Se crea una variable DATA1. En la que se establecen los
  incrementos de tiempo para cada articulación*/
  for (int i=0;i<Time;i++){
    DATA1[i] = inct*i;}
```

```
  for (int n=0;n<Njoints;n++){
    /*Se llama a la funcion del interpolador cubico.
    Este devuelve una variable global (POS) la cual se guardara en otra
    variable global (JTs) la cual devolvera la funcion actual*/
  if(n==0){
      InterpoladorCubico(DATA1,Jvals1,inct,NSampPoints,n,NCrossPoints);
    }else if(n==1){
      InterpoladorCubico(DATA1,Jvals2,inct,NSampPoints,n,NCrossPoints);
    }else if(n==2){
      InterpoladorCubico(DATA1,Jvals3,inct,NSampPoints,n,NCrossPoints);
    }else{
      InterpoladorCubico(DATA1,Jvals4,inct,NSampPoints,n,NCrossPoints);
    };
  };
  float tespera = inct/(NSampPoints+1);
  int valorF = ((NCrossPoints+1)*(NSampPoints+1))+1;
  for(int c=0;c<valorF;c++){
    Serial.print("               ");Serial.print(Jvals[0][c]);
    Serial.print("     ");Serial.print(Jvals[1][c]);
    Serial.print("     ");Serial.print(Jvals[2][c]);
    Serial.print("     ");Serial.println(Jvals[3][c]);

MoveServos(tespera,Jvals[0][c],Jvals[1][c],Jvals[2][c],Jvals[3][c],c,valorF);
  };
  VAL=0;
};
void setup() {
   //Comunicacion Serial
```

```
  /*Se comunica que va a recibir un dato*/
  Serial.begin(9600);
  //Asignacion de los pines utilizados para los servos
  servo1.attach(13);
  servo2.attach(12);
  servo3.attach(11);
  servo4.attach(10);
  // Se configuran pines como salida
  pinMode (ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(LED_M1,OUTPUT);
  pinMode(LED_M2,OUTPUT);
  pinMode(LED_M3,OUTPUT);
  pinMode(LED_M4,OUTPUT);
  InicioServos();
};
void loop() {
  //Variables locales
  float Init_pose[6] = {158.5,0,85,0,0,0};
  float Final_pose[6];
  float Time;
  int NCrossPoints;
  int NSampPoints;
  float Parametros[2];
  int MODO;
  int codo;
  VAL=1;
```

```
  /*Se pide al usuario que inroduzca los datos y se van comprobando*/
 Serial.print("Numero de puntos de cruce: ");delay(6000);
  NCrossPoints=Serial.parseInt();
  Serial.println(NCrossPoints);
  delay(1000);
  if((NCrossPoints<1)||(NCrossPoints>5)){
    Serial.println ("/*ERROR*/");delay(100);
    exit(0);
  }
  Serial.print("Numero de puntos de muestreo: ");delay(6000);
  NSampPoints=Serial.parseInt();
  Serial.println(NSampPoints);
  delay(1000);
  if((NSampPoints<1)||(NSampPoints>4)){
    Serial.println("/*ERROR*/");delay(100);
    exit(0);
  }
  Serial.print("Duracion de la trayectoria (seg): ");delay(6000);
  Time=Serial.parseFloat();
  Serial.println(Time);
  delay(1000);
  if(Time<0){
    Serial.println("/*ERROR*/");delay(100);
    exit(0);
  }
  Serial.println("Configuracion estructura manipulador: (1) codo derecha - (0)
codo izquierda ");
  delay(6000);
```

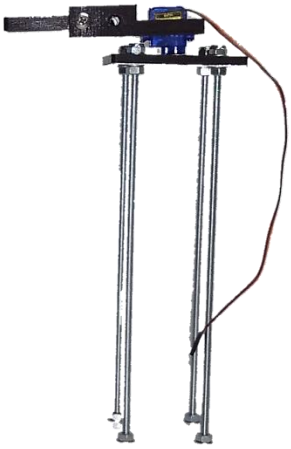
```
  Serial.print("Codo: ");
  codo=Serial.parseFloat();
Serial.println(codo);
  delay(1000);
  if((codo<0)||(codo>1)){
    Serial.println("/*ERROR*/");delay(100);
    exit(0);
  }
  Serial.println("Posicion final del robot: ");
  Serial.print("Coordenada X: ");delay(6000);
  Final_pose[0]=Serial.parseFloat();
  Serial.println(Final_pose[0]);
  delay(1000);
  Serial.print("Coordenada Y: ");delay(6000);
  Final_pose[1]=Serial.parseFloat();
  Serial.println(Final_pose[1]);
  delay(1000);
  Serial.print("Coordenada Z: ");delay(6000);
  Final_pose[2]=Serial.parseFloat();
  Serial.println(Final_pose[2]);
  delay(1000);
  Serial.print("Orientacion alpha: ");delay(6000);
  Final_pose[3]=Serial.parseFloat();
  Serial.println(Final_pose[3]);
  delay(1000);
  Final_pose[4]=0;
  Final_pose[5]=180;
  Serial.println("Modo funcionamiento: ");
```

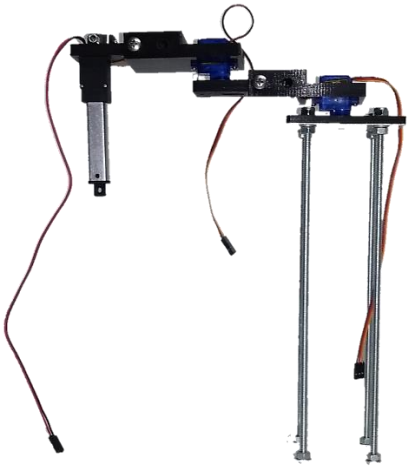
```
  Serial.println("(1)Led Verde - Modo 1: Eslabones cortos");
  Serial.println("(2)Led Azul - Modo 2: Eslabones largos");
  Serial.println("(3)Led Amarillo - Modo 3: Eslabon1 corto - Eslabon2 largo");
  Serial.println("(4)Led Blanco - Modo 4: Eslabon1 largo - Eslabon2 corto");delay(6000);
  MODO = Serial.parseInt();
  Serial.print("MODO: ");Serial.println(MODO);
  delay(1000);
  /*Se establecen los valores de los eslabones en funcion del modo seleccionado*/
if (MODO==1){
    digitalWrite(LED_M1, HIGH);
    Parametros[0]=78.5;
    Parametros[1]=81;
  }else if(MODO==2){
    digitalWrite (LED_M2, HIGH);
    Parametros[0]=97.5;
    Parametros[1]=100;
  } else if(MODO==3){
    digitalWrite (LED_M3, HIGH);
    Parametros[0]=78.5;
    Parametros[1]=100;
  } else if(MODO==4){
    digitalWrite (LED_M4, HIGH);
    Parametros[0]=97.5;
    Parametros[1]=81;
  }else{
    Serial.println("/** ERROR **/");
```

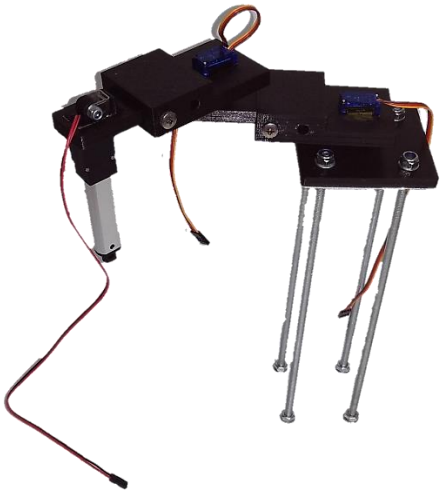
```
    Serial.println("/** MODO DE FUNCIONAMIENTO NO VALIDO **/");
    delay(100);
    exit(0);
  }
  //Comienzo programa
  /*Mientras la variable no cambie, se permanece dentro del bucle*/
while(VAL==1){

ControlCinematico(Parametros,Init_pose,Final_pose,Time,NCrossPoints,NSampPoints
,codo);
  }
  /*Se apagan los LEDs*/
  digitalWrite (LED_M1, LOW);
  digitalWrite (LED_M2, LOW);
  digitalWrite (LED_M3, LOW);
  digitalWrite (LED_M4, LOW);
  exit(0);
}
```

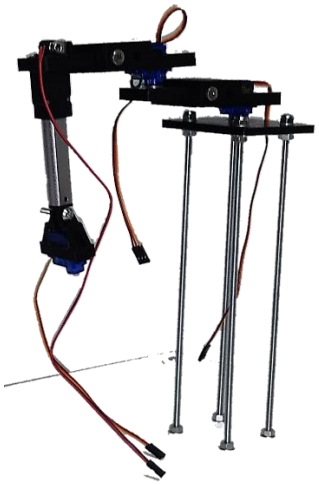
3. GALERÍA FOTOGRÁFICA

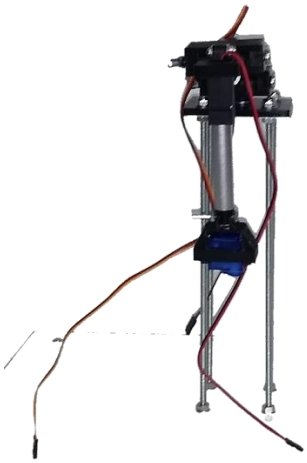
El manipulador junto con la protoboard, la placa Arduino Uno y el Modulo Controlador de Motores L298N han sido fijados a una base de madera la cual posee cuatro tacos de forma que sirvan como patas para una mayor estabilidad, transporte y manejo. A continuación, se muestra el manipulador terminado, así como un conjunto de fotografías de lo que ha sido el proceso de montaje.

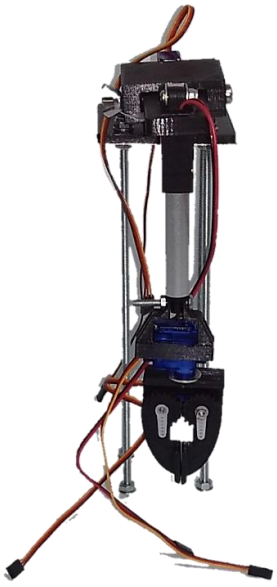








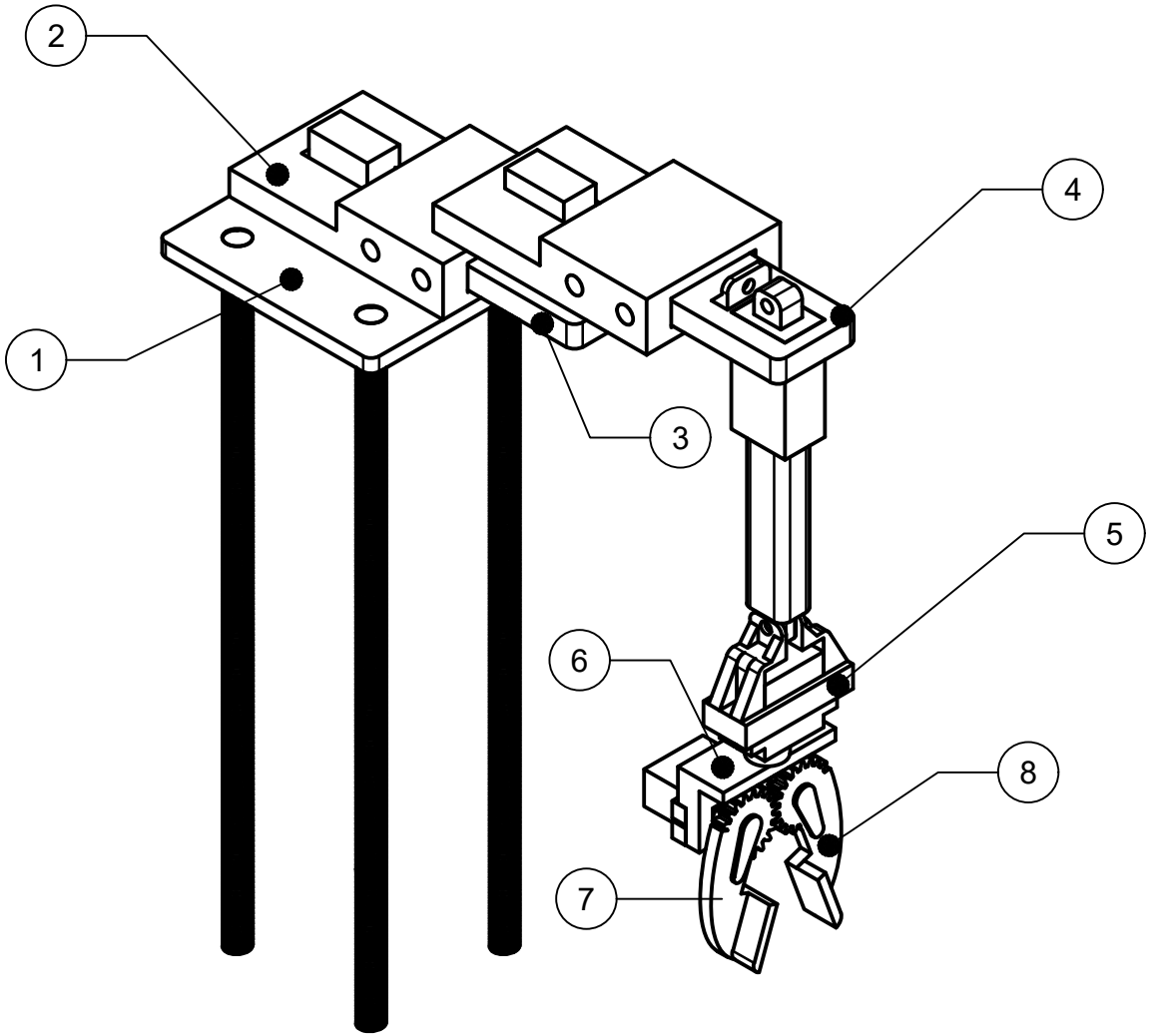




DOCUMENTO 3

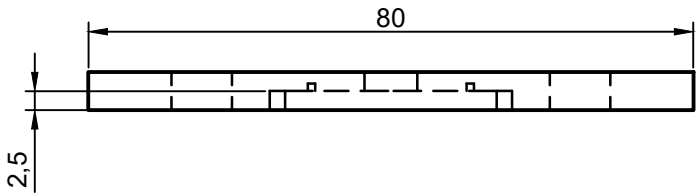
PLANO

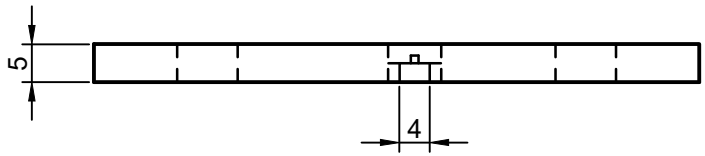
Se incluyen a continuación los planos de todas las piezas diseñadas. En la tabla se muestra el conjunto de todos ellos.

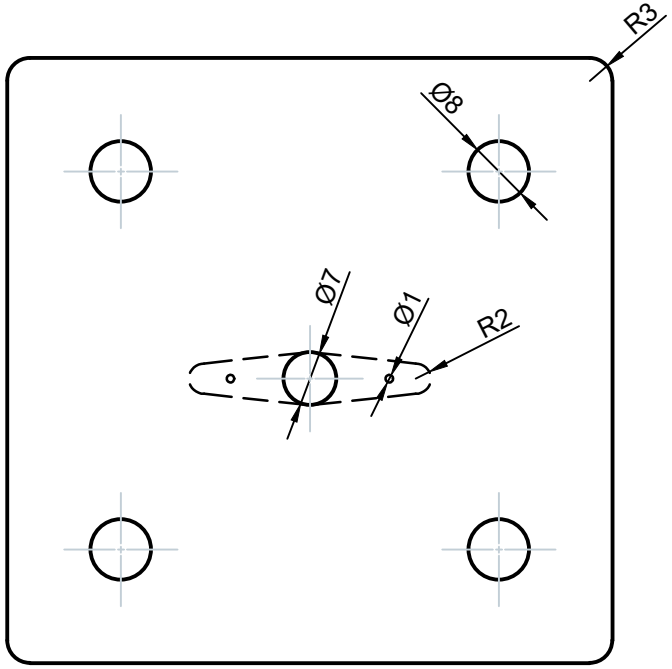


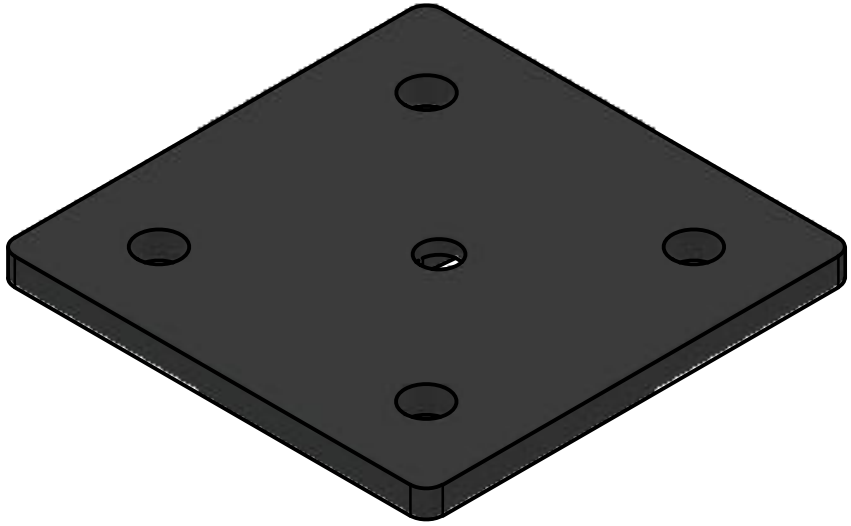
LISTA DE PIEZAS			
MARCA	**CTDAD**	**DESCRIPCIÓN**	**MATERIAL**
1	1	Hombro	PLA
2	2	Brazo parte 1	PLA
3	1	Brazo parte 2	PLA
4	1	Antebrazo	PLA
5	1	Soporte actuador lineal	PLA
6	1	Soporte pinzas	PLA
7	1	Pinza A	PLA
8	1	Pinza B	PLA

UNIDADES	DIBUJADO	L. Torre San Emeterio	UNIVERSIDAD DE CANTABRIA
mm	REVISADO	C. Torre Ferrero	DISEÑO Y CONFIGURACIÓN DE UN ROBOT SCARA
	FECHA	05/07/2022	
ESCALA	LAMINA		N° DE PLANO
1:2	PLANO DEL CONJUNTO DEL ROBOT COMPLETO		1/9

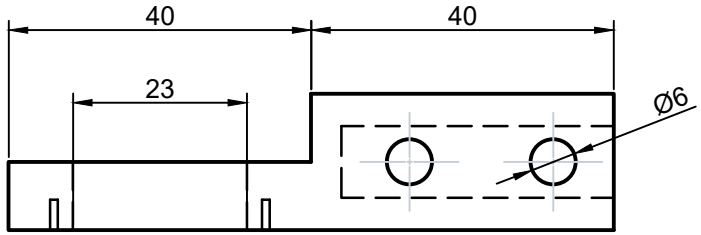


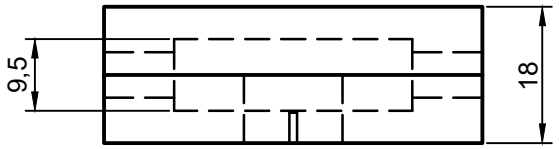


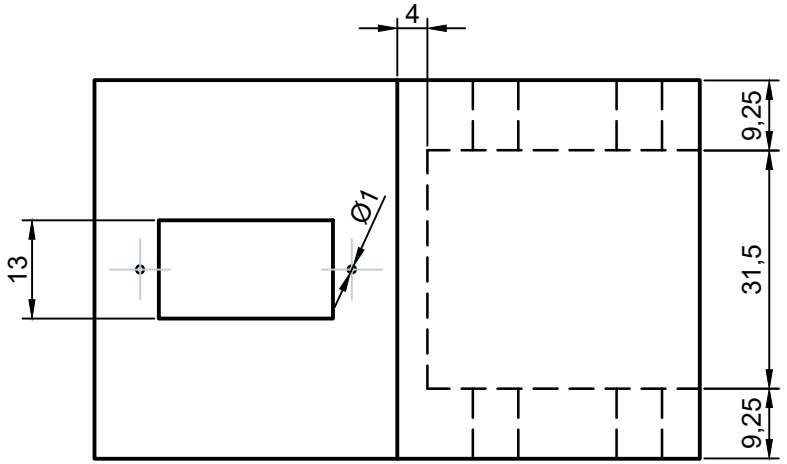


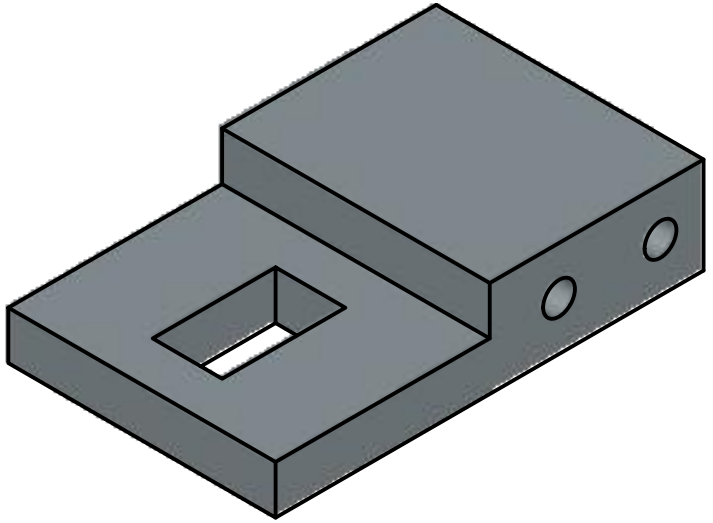


UNIDADES	DIBUJADO	L. Torre San Emeterio	UNIVERSIDAD DE CANTABRIA
mm	REVISADO	C. Torre Ferrero	DISEÑO Y CONFIGURACIÓN DE UN ROBOT SCARA
	FECHA	05/07/2022	
ESCALA	LAMINA		N° DE PLANO
1:1	HOMBRO		2/9

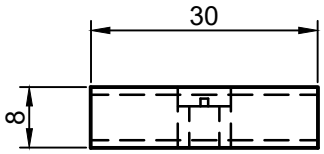


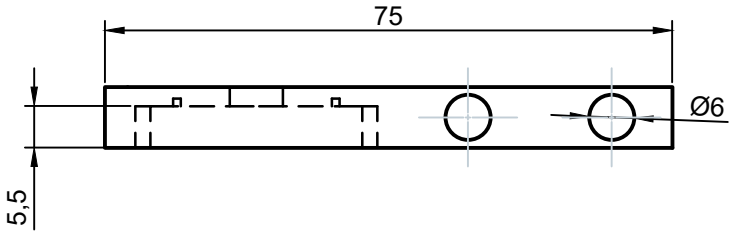


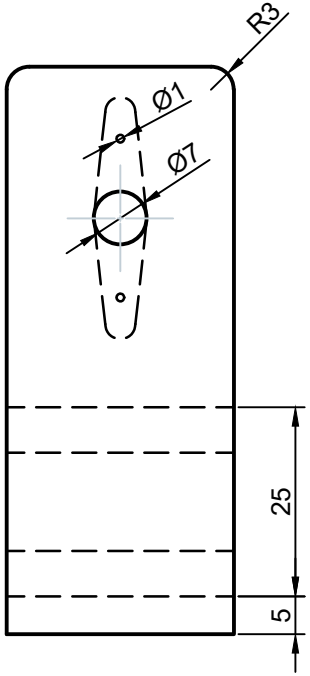


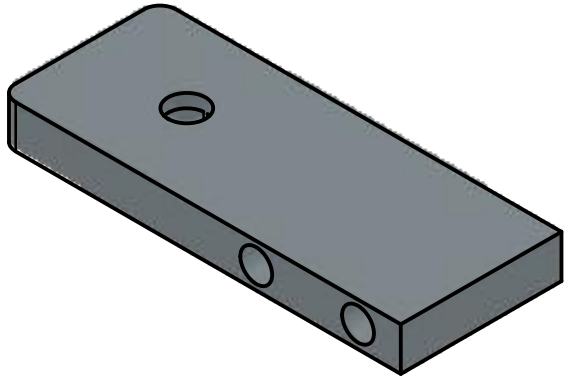


UNIDADES	DIBUJADO	L. Torre San Emeterio	UNIVERSIDAD DE CANTABRIA
mm	REVISADO	C. Torre Ferrero	DISEÑO Y CONFIGURACIÓN DE UN ROBOT SCARA
	FECHA	05/07/2022	
ESCALA	LAMINA		N° DE PLANO
1:1	BRAZO Y ANTEBRAZO PARTE 1		3/9

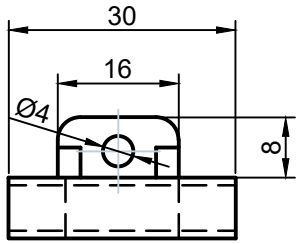


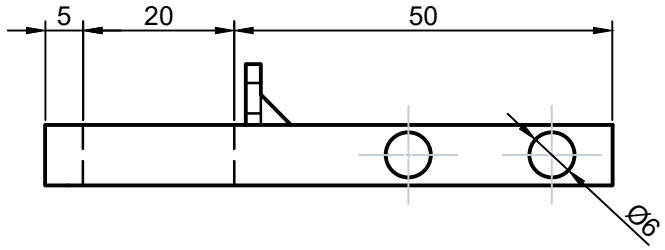


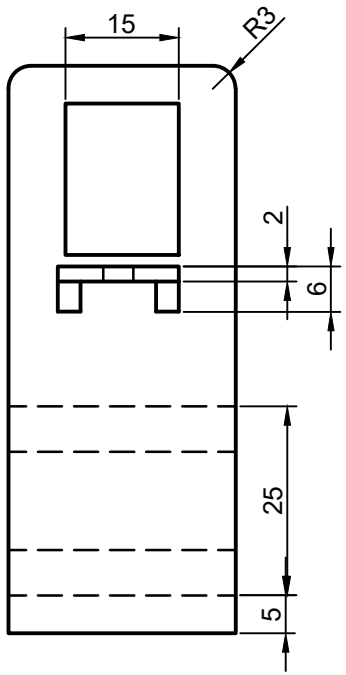


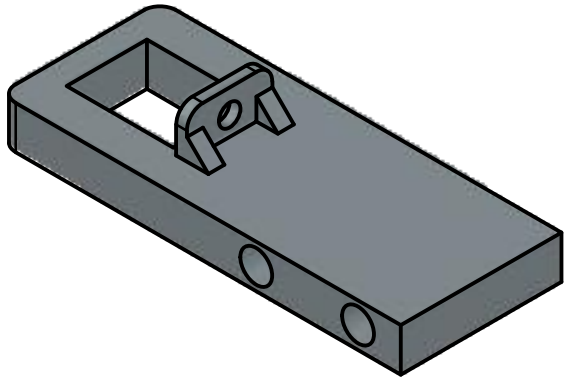


UNIDADES	DIBUJADO	L. Torre San Emeterio	UNIVERSIDAD DE CANTABRIA
mm	REVISADO	C. Torre Ferrero	DISEÑO Y CONFIGURACIÓN DE UN ROBOT SCARA
	FECHA	05/07/2022	
ESCALA	LAMINA		N° DE PLANO
1:1	BRAZO PARTE 2		4/9

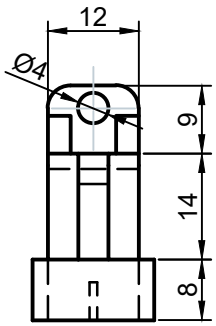


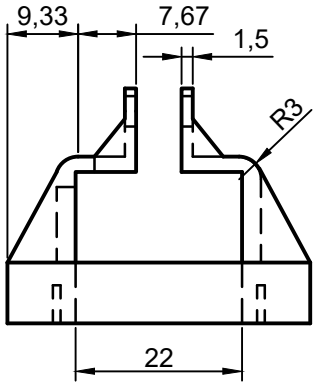


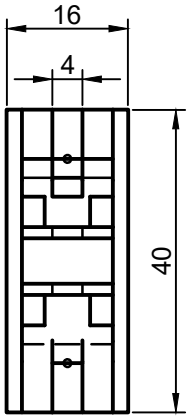


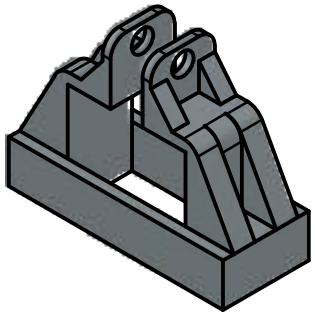


UNIDADES	DIBUJADO	L. Torre San Emeterio	UNIVERSIDAD DE CANTABRIA
mm	REVISADO	C. Torre Ferrero	DISEÑO Y CONFIGURACIÓN DE UN ROBOT SCARA
	FECHA	05/07/2022	
ESCALA	LAMINA		N° DE PLANO
1:1	ANTEBRAZO PARTE 2		5/9

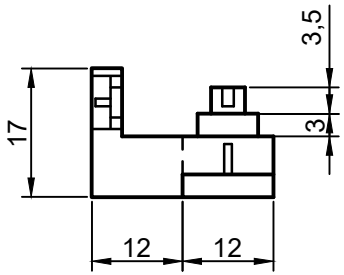


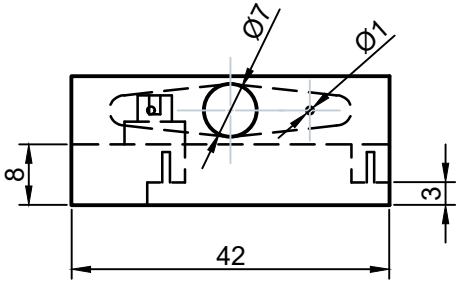


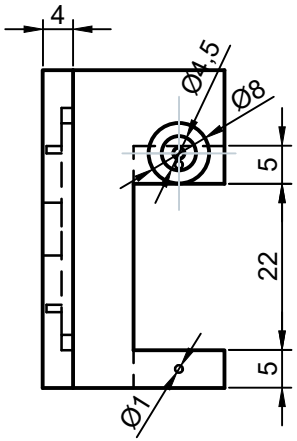


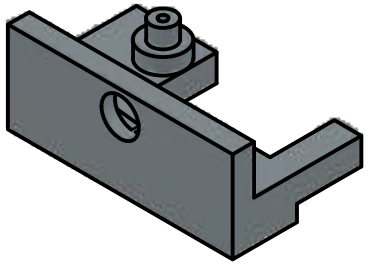


UNIDADES	DIBUJADO	L. Torre San Emeterio	UNIVERSIDAD DE CANTABRIA
mm	REVISADO	C. Torre Ferrero	DISEÑO Y CONFIGURACIÓN DE UN ROBOT SCARA
	FECHA	05/07/2022	
ESCALA	LAMINA		N° DE PLANO
1:1	SOPORTE DEL ACTUADOR LINEAL		6/9

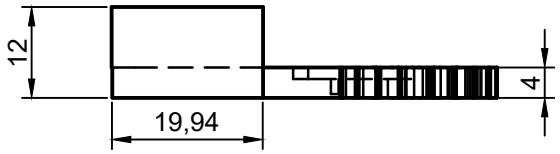


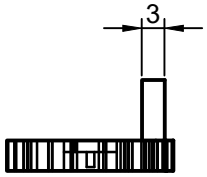


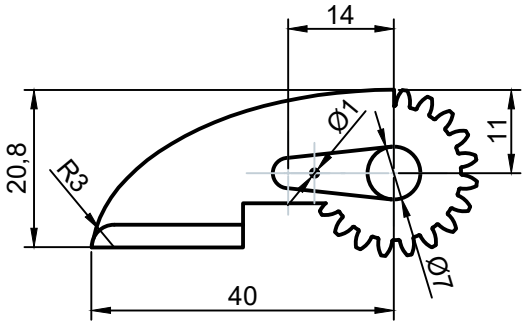


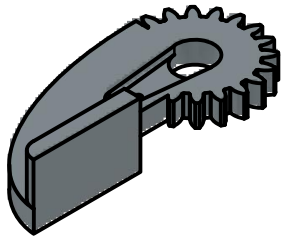


UNIDADES	DIBUJADO	L. Torre San Emeterio	UNIVERSIDAD DE CANTABRIA
mm	REVISADO	C. Torre Ferrero	DISEÑO Y CONFIGURACIÓN DE UN ROBOT SCARA
	FECHA	05/07/2022	
ESCALA	LAMINA		N° DE PLANO
1:1	SOPORTE DE LAS PINZAS		7/9

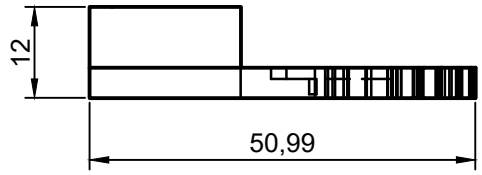


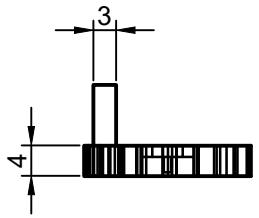


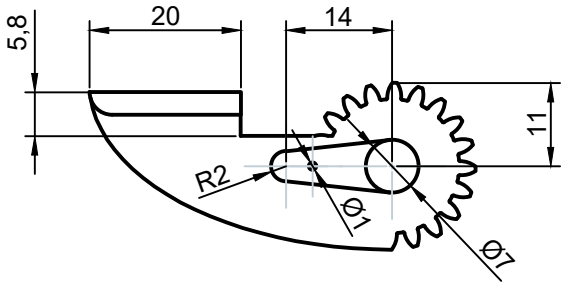


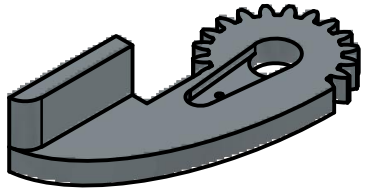


UNIDADES	DIBUJADO	L. Torre San Emeterio	UNIVERSIDAD DE CANTABRIA
mm	REVISADO	C. Torre Ferrero	DISEÑO Y CONFIGURACIÓN DE UN ROBOT SCARA
	FECHA	05/07/2022	
ESCALA	LAMINA		N° DE PLANO
1:1	PINZA A		8/9









UNIDADES	DIBUJADO	L. Torre San Emeterio	UNIVERSIDAD DE CANTABRIA
mm	REVISADO	C. Torre Ferrero	DISEÑO Y CONFIGURACIÓN DE UN ROBOT SCARA
	FECHA	05/07/2022	
ESCALA	LAMINA		N° DE PLANO
1:1	PINZA B		9/9

DOCUMENTO 4
PLIEGO DE CONDICIONES

ÍNDICE GENERAL

ÍNDICE DE TABLAS

1. PLIEGO DE CONDICIONES GENERALES

1.1. Objeto del pliego de condiciones técnicas particulares

Este Pliego de Condiciones Técnicas Particulares comprende el conjunto de características que deberán cumplir los materiales empleados en la construcción, así como los técnicos de su colocación en la obra y los que deberán mandar en la ejecución de cualquier tipo de instalaciones y de obras accesorias.

1.2. Documentos del proyecto

El presente proyecto consta de los siguientes documentos:

- Documento nº 1: Memoria.
- Documento nº 2: Anexos.
- Documento nº 3: Planos
- Documento n° 4: Pliego de Condiciones.
- Documento n° 5: Mediciones.
- Documento nº 6: Presupuesto.
- Documento nº 7: Bibliografía.

Se entiende por documentos contractuales aquéllos que estén incorporados en el contrato y que sean de obligado cumplimiento:

- Planos
- Pliego de Condiciones
- Cuadro de precios n° 1
- Cuadros de precios n° 2
- Presupuesto total

El resto de los documentos del Proyecto tienen carácter informativo.

1.3. Normas aplicables

Además de las condiciones especificadas en las siguientes especificaciones, se deben tener en cuenta las normas aplicables a este diseño en el curso del trabajo, que siempre debe corresponder al nivel de calidad pretendido por el ingeniero requerido por el diseñador. Se deben cumplir las siguientes condiciones:

La normativa sobre sistemas de visión artificial para medidas de seguridad y salud es la contenida en el Real Decreto 1215/1997.

Normativas pertinentes aplicables a los robots industriales:

- UNE-EN ISO 10218-1:2011 Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 1: Robots.
- UNE-EN ISO 10218-2:2011 Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales. Parte 2: Sistemas robot e integración.
- ISO/TR 13309:1995 Manipulación de robots industriales. Guía informativa sobre los equipos de ensayo y métodos de metrología de operación para la evaluación del desempeño del robot de acuerdo con la norma ISO 9283.
- ISO 9283:1998 Robots manipuladores industriales. Criterios de análisis de prestaciones y métodos de ensayo relacionados.
- ISO 9409-1:2004 Manipulación de robots industriales. Interfaces mecánicas. Parte 1: Placas.
- ISO 9409-2:2002 Manipulación de robots industriales. Interfaces mecánicas. Parte 2: Ejes.
- ISO 9787:2013 Robots y dispositivos robóticos. Sistemas de coordenadas y nomenclaturas de movimiento.
- ISO 8373:2012 Robots y dispositivos robóticos. Vocabulario.
- ISO 14539:2000 Robots manipuladores industriales. Transporte de objetos con dispositivos de agarre tipo empuñadura. Vocabulario y presentación de características.

1.4. Equipos para la ejecución de los trabajos

El personal de ejecución utilizará los materiales y equipos indicados en este proyecto y realizará los trabajos de montaje correspondientes. Los materiales y equipos deben ser de buena calidad.

Si el equipo no cumple con las características y condiciones especificadas en esta especificación, el organismo ejecutor deberá reemplazar el equipo por otro que las cumpla.

1.5. Otras consideraciones que cumplir por los materiales y equipos

Los materiales y equipos deben cumplir con los requisitos de las normas y reglamentos aplicables.

Si se selecciona un producto, tipo o modelo en particular para usarse en cualquier punto de este proyecto, se puede reemplazar con un producto que cumpla con los mismos estándares de calidad. Las partes del equipo o equipo podrán ser reemplazadas por otras, siempre que tengan características idénticas o muy similares y siempre con la aprobación del ingeniero de diseño.

1.6. Objeto de los planos y especificaciones

Los planos y especificaciones tienen por objeto mostrar al cliente la forma, tamaño, calidad y cantidad de piezas y sistemas a fabricar, y su disposición relativa durante el montaje, la instalación, la mano de obra utilizada, el equipo y las instalaciones de montaje necesarios para llevar a cabo el proyecto como siempre que el ingeniero de diseño no indique lo contrario.

El Promotor deberá realizar todo el trabajo descrito en los planos y descrito en el presupuesto o presupuesto de todo el trabajo que se considere necesario para completar la compilación de manera aceptable.

1.7. Programa de ejecución y plazos

El programa previsto para la ejecución del proyecto se estima de 7 a 11 días, puesto que la maquinaria necesaria y edificación para el proceso ya ha sido producida anteriormente y partiremos del actual.

- Día 1. *Inicio del proyecto:* obtención de los materiales y escaneado 3D de las piezas.
- Día 4. *Continuación del proyecto:* Calibración de los componentes y comprobación del estado de los mismos.
- Día 5. *Continuación del proyecto:* Limado de piezas.
- Día 6. *Continuación y montaje del proyecto:* Instalación de las conexiones de los componentes externos y comprobación de funcionamiento.
- Día 8. *Final de montaje:* Construcción final del proyecto.
- Día 10. *Finalización de montajes e instalaciones:* Pruebas y puestas en marcha definitivas.

El autor del proyecto puede ordenar el cambio de cualquier unidad planificada por una nueva y presentar los planos finales al contratista, quien a partir de ese momento formará parte del proyecto.

Los ajustes se ingresarán en el libro de ordenes requerido, el cual se devolverá al mercado al finalizar el trabajo y permanecerá allí a discreción del gerente o su designado. Siempre que los ajustes impliquen la sustitución de una unidad de obra por otra unidad de obra de características similares a las unidades de obra mostradas en el presupuesto, los ajustes no modificarán el precio unitario que figura en el proyecto.

Al implementar este proyecto, se aplicarán todas las reglas y decretos enumerados en la sección legal de la especificación. Para todo lo no presentado, se aplicarán las Condiciones Técnicas de la Administración General de Arquitectura (Decreto de 4 de junio de 1973).

2. PLIEGO DE CONDICIONES TÉCNICAS

2.1. Manipulador

El modelo utilizado será el propio diseñado durante el proyecto, el cual será una versión propia del robot SCARA.

En particular, el modelo anterior tiene las siguientes características:

El robot SCARA es un robot industrial de 4 grados de libertad diseñado para industrias de manipulación de objetos (pick and place). El robot posee una preconfiguración en dos de sus uniones (brazo y antebrazo) disponiendo de esta forma la posibilidad de ampliar o disminuir el espacio de trabajo, también posee una excelente comunicación con los sistemas externos.

El modelo diseñado está disponible para instalarse en el suelo.

2.1.1. Masa del manipulador

Datos	**Descripción**
Manipulador	280 gr.

Tabla 1. Masa del manipulador

2.1.2. Dimensiones y planos

Las dimensiones y los planos se mencionan en el documento del plano adjunto.

2.2. Normas y seguridad

2.2.1. Normas

El manipulador cumple las siguientes normas:

Estándar EN:

EN ISO 12100 -1 Seguridad de maquinaria, terminología básica.

EN ISO 12100 -2 Seguridad de maquinaria, especificaciones técnicas.

EN 954-1 Seguridad de maquinaria, partes de los sistemas de control relacionadas con la seguridad.

EN 775 Robots industriales con manipulación, seguridad.

Estándar ISO:

ISO 10218 Robots industriales con manipulación, seguridad.

ISO 9787 Robots industriales con manipulación, sistemas de coordenadas y Movimientos.

ISO 9409-1 Robots industriales con manipulación, interfaz mecánica.

Estándar IEC:

IEC 204-1 Equipos eléctricos de máquinas industriales.

Estándar ANSI/CAN/Normas federales EE. UU.:

ANSI/RIA R15.06/1999 (opción) Requisitos de seguridad para robots industriales y sistemas Robotizados.

ANSI/UL 1740-1998 (opción) Norma de seguridad para robots y equipo robotizado.

CAN/CSA Z 434-03 Robots industriales y sistemas robotizados - Requisitos generales de seguridad Norma federal 209 de los EE.UU. Clasificación de sala limpia.

El robot cumple con todos los estándares de salud y seguridad estipulados en la Directiva de Maquinaria de la CEE.

2.2.2. Seguridad

El robot está diseñado para una seguridad absoluta.

Selección del modo de funcionamiento:

El robot únicamente puede utilizarse de forma automática. Siendo este previamente configurado desde la unidad de programación. No se admite la utilización desde un dispositivo externo.

Velocidad reducida:

En el modo automático,

Seguridad contra incendios:

El sistema de control y operación cumple con los estrictos requisitos de UL (Underwriters Laboratories) para la seguridad contra incendios.

2.3. Instalación

2.3.1. Consideraciones generales

El manipulador está disponible en una sola configuración adaptada al entorno de ámbito educativo.

2.3.2. Requisitos de funcionamiento

Temperatura ambiente.

Descripción	Temperatura
Manipulador durante el funcionamiento.	De +5ºC a +30ºC
Robot completo durante el transporte y el almacenamiento.	De -5ºC a +35ºC

Tabla 2. Requisitos de temperaturas

Humedad relativa.

Descripción	Temperatura
Manipulador durante el funcionamiento.	95% como máximo a temperatura constante.
Robot completo durante el transporte y el almacenamiento.	95% como máximo a temperatura constante.

Tabla 3. Requerimientos de humedades

2.3.3. Mantenimiento

Consideraciones generales.

El robot requiere un mantenimiento mínimo durante su funcionamiento.

Ha sido diseñado para permitir el servicio técnico más simple posible:

- Se utilizan micro servomotores de CC sin mantenimiento. Además, si se produce una avería, su diseño modular significa que se pueden reemplazar fácilmente.
- El enrutamiento de cables está optimizado para una vida útil máxima. Además, si se produce una avería, su diseño modular significa que se pueden reemplazar fácilmente.

Mantenimiento.

Se requieren las siguientes actividades de mantenimiento:

- Cada 6 meses sustituir las baterías que proporcionan energía al actuador lineal.
- Cada 4 meses comprobar el estado de las resistencias de protección de la protoboard.

El intervalo de mantenimiento depende del uso del robot.

2.3.4. Movimiento del robot.

Tipo de movimiento	Área de movimiento
Eje 1: Movimiento de rotación del brazo	De +90º a -90º
Eje 2: Movimiento de rotación del antebrazo	De +88º a -90º
Eje 3: Movimiento lineal	De 0mm a 50mm
Eje 4: Movimiento de rotación de la muñeca	De +90º a -90º

Tabla 4. Movimiento de los ejes

2.3.5. Velocidad

Nº de eje	Manipulador SCARA diseño propio
1	0.10 s/60º
2	0.10 s/60º
3	0.10 s/60º
4	0.10 s/60º

Tabla 5. Velocidades en cada eje

2.4. Objetos de manipulación y colocación

2.4.1. Objeto de manipulación

El objeto para manipular se trata de un objeto común, posible de encontrar en cualquier instante de no más de 150gr.

2.4.2. Entorno de trabajo

El proceso debe desarrollarse sobre una mesa de trabajo con medidas:

- 800 x 750 mm.

3. PLIEGO DE ESPECIFICACIONES DE EJECUCIÓN

3.1. Proceso

El proceso requerido es que el robot tome una pieza del propio entorno de trabajo y la coloque en otro punto prefijado. Debe reunir las siguientes características:

- Debido a la fabricación y el procesamiento posterior, las piezas pueden cambiar de posición según el ancho de la mesa.

3.2. Ergonomía y seguridad de operarios

El operador no puede ingresar al cuadro de trabajo mientras el manipulador se está ejecutando. Los túneles de trabajo deben estar separados físicamente del exterior y se deben prever medidas de bloqueo de energía si el operador debe ingresar a la sala de trabajo o mantenimiento.

El pulsador de emergencia debe estar visible, sin obstrucciones.

La iluminación debe ser suficiente para evitar posibles caídas o resbalones. Los espacios de trabajo deben ser relativamente proporcionales para que los operadores puedan ingresar y realizar tareas de mantenimiento sin comprometer su seguridad física debido a posiciones forzadas o componentes de gran altura, nivel de la cabeza, etc.

DOCUMENTO 5
MEDICIONES

ÍNDICE:

1. MEDICIONES DE MATERIALES

Unidades	Material	Cantidad
Uds.	Arduino uno R3	1
Uds.	Módulo de placa L298N	1
Uds.	Micro servomotor SG90 TowerPro	4
Uds.	Micro lineal actuador	1
Uds.	USB cable hama usb-A a USB-B	1
Uds.	Cables de puente Jumper Wire Macho-Macho	40
Uds.	Limas de diamante mini	1
Uds.	Portapilas LR6 para pilas AA negro	1
Uds.	Pilas LR6 AA 1.5V	8
Uds.	Leds de colores	4
Uds.	Pulsadores	2
Uds.	Resistencias 10k	6
Uds.	Tabla madera	1
Uds.	Tuerca hexagonal	12
Uds.	Tuerca hexagonal ciega	4
Uds.	Varilla roscada 6mm	4
Uds.	Tornillo	4
Uds.	Piezas diseñadas impresora 3D	9

2. MEDICIONES TIEMPOS DE OPERACIÓN

UNIDADES	OPERACIÓN	TIEMPO (min)
min	Taladrado y atornillado de las piezas	20
min	Instalación de los servomotores y el actuador lineal	15
min	Montaje completo elementos	80
min	Limado de piezas	360
min	Instalación del hardware	30
min	Total	1525

DOCUMENTO 6
PRESUPUESTO

ÍNDICE

1. JUSTIFICACIÓN DE LOS PRECIOS

1.1. Justificación de los precios de los materiales

MATERIAL	PRECIO U(€/Unidad)
Arduino uno R3	26.99
Módulo de placa L298N	6.20
Micro servomotor SG90 TowerPro	3.49
Micro lineal actuador	27.59
USB cable hama usb-A a USB-B	2.99
Cables de puente Jumper Wire Macho-Macho	0.157
Limas de diamante mini	7.99
Portapilas LR6 para pilas AA negro	4.50
Pilas LR6 AA 1.5V	0.288
Leds de colores	0.033
Resistencias 10k	0.12
Tabla madera	20
Tuerca hexagonal	0.05
Tuerca hexagonal ciega	0.33
Varilla roscada 6mm	0.61
Tornillo	0.189
Piezas diseñadas impresora 3D	4.44

1.2. Justificación del coste de la mano de obra

Hora de trabajo de un peón es de 15€/h o de 0.25€/minuto.

2. PRESUPUESTOS PARCIALES

2.1. Presupuesto parcial relativo a los materiales

MATERIAL	CANTIDAD	PRECIO U(€/Unidad)	PRECIO TOTAL (€)
Arduino uno R3	1	26.99	26.99
Módulo de placa L298N	1	6.20	6.20
Micro servomotor SG90 TowerPro	4	3.49	13.99
Micro lineal actuador	1	27.59	27.59
USB cable hama usb-A a USB-B	1	2.99	2.99
Cables de puente Jumper Wire Macho-Macho	40	0.157	6.29
Limas de diamante mini	1	7.99	7.99
Portapilas LR8 para pilas AA negro	1	4.50	4.50
Pilas AA 1.5V	8	0.288	2.31
Leds de colores	4	0.033	0.132
Resistencias 10k	6	0.12	0.72
Tabla madera	1	20	20
Tuerca hexagonal	12	0.05	0.6
Tuerca hexagonal ciega	4	0.33	1.32
Varilla roscada 6mm	4	0.61	2.44
Tornillo	10	0.189	1.89
Piezas diseñadas impresora 3D	9	4.44	40
Total			168.952

2.2. PRESUPUESTO PARCIAL RELATIVO A LAS OPERACIONES DE MONTAJE

MANO DE OBRA				
ORDEN	**OPERACIÓN**	**TIEMPO (min)**	**PRECIO DE MANO DE OBRA (€/min)**	**PRECIO TOTAL (€)**
1	Taladrado y atornillado de las piezas	20	0.25	5
2	Instalación de los servomotores y el actuador lineal	15		3.75
3	Montaje completo elementos	80		20
4	Limado de piezas	360		90
5	Instalación del hardware	30		7.5
Total		1525		126.25

3. PRESUPUESTO GLOBAL

PRESUPUESTO	
Materiales	168.952
Mano de obra	126.25€
Impuestos(21%)	61.99
Presupuesto total	357.192

El presupuesto para la implantación física del prototipo ascendió a un total de 357.192 euros (trescientos cincuenta y siete euros y dos centimos).

DOCUMENTO 7
BIBLIOGRAFÍA

Enlaces consultados:

1) Historia de los robots y la robótica [sitio web]. [Consulta: 29 marzo 2022]. Disponible en: https://robotnik.eu/es/historia-de-los-robots-y-la-robotica/
2) ¿Qué es un robot? [sitio web]. [Consulta: 29 marzo 2022]. Disponible en: http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.1.htm
3) Clases de robots [sitio web]. [Consulta: 29 marzo 2022]. Disponible en: http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.2.htm
4) Robots industriales [sitio web]. [Consulta: 1 abril 2022]. Disponible en: http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.4.htm
5) Cinemática directa [sitio web]. [Consulta 10 abril 2022]. Disponible en: https://es.wikipedia.org/wiki/Cinem%C3%A1tica_directa
6) Algoritmo de Denavit–Hartenberg. Caso de estudio SSRMS [sitio web]. [Consulta 10 abril 2022]. Disponible en: https://maferstech.com/robotica-algoritmo-de-denavit-hartenberg-caso-de-estudio-ssrms/
7) Cinemática inversa [sitio web]. [Consulta: 11 abril 2022]. Disponible en: https://es.wikipedia.org/wiki/Cinem%C3%A1tica_inversa
8) Servomotores [sitio web]. [Consulta: 15 abril 2022]. Disponible en: http://platea.pntic.mec.es/vgonzale/cyr_0204/ctrl_rob/robotica/sistema/motores_servo.htm#PWM
9) Towerpro [sitio web]. [Consulta: 20 abril 2022]. Disponible en: http://www.towerpro.com.tw/
10) Guía de materiales de impresión 3D: Tipos, aplicaciones y propiedades [sitio web]. [Consulta 15 mayo 2022]. Disponible en: https://formlabs.com/es/blog/materiales-impresion-3d/
11) EL MÓDULO CONTROLADOR DE MOTORES L298N [sitio web]. [Consulta: 15 junio 2022]. Disponible en: https://www.prometec.net/l298n/
12) Arduino uno [sitio web]. [Consulta: 15 junio 2022]. Disponible en: https://es.wikipedia.org/wiki/Arduino_Uno
13) Control Cinemático del Robot [sitio web]. [Consulta 15 julio 2022]. Disponible en: https://slideplayer.es/slide/3734660/
14) Planificación de Trayectorias [sitio web]. [Consulta 15 julio 2022]. Disponible en: http://www.udesantiagovirtual.cl/moodle2/mod/book/view.php?id=24819
15) Spline [sitio web]. [Consulta 18 julio 2022]. Disponible en: https://es.wikipedia.org/wiki/Spline#Interpolaci%C3%B3n_Segmentaria_C%C3%BAbica

16) Control Cinemático [sitio web]. [Consulta 18 julio 2022]. Disponible en: https://docplayer.es/62500645-Control-cinematico-funciones-de-control-cinematico-tipos-de-trayectorias-interpolacion-de-trayectorias-robotica-industrial-control-cinematico-2.html
17) Arduino en español [sitio web]. [Consulta 16 agosto 2022]. Disponible en: http://manueldelgadocrespo.blogspot.com/
18) Introducción a los Tipos de Dato con Arduino [sitio web]. [Consulta 16 agosto 2022]. Disponible en: https://arduino.cl/introduccion-a-los-tipos-de-dato-con-arduino/
19) Enviar Datos y Leer Puerto Serie con Arduino - Comunicación Serial. [sitio web] .[Consulta 18 agosto 2022]. Disponible en: https://www.wexterhome.com/curso-arduino/enviar-datos-y-leer-puerto-serie/
20) Curso Arduino Avanzado. Introducción. [sitio web] .[Consulta 18 agosto 2022]. Disponible en: https://www.electrodaddy.com/curso-basico-arduino-introduccion/

Libros y artículos consultados:

13. GUPTA, A.K; ARORA, S.K y WESTCOTT, J.R. 2017. Industrial Automation and Robotics: An Introduction. Archive pdf. Disponible en: https://es.es1lib.org/dl/3704850/05b450
14. Barrientos, A.; PEÑÍN, L.F; BALAGUER, C y ARACIL, R. 2007. Fundamentos de robótica. 2ª ed. McGraw-Hill. Archivo pdf. Disponible en: https://www.academia.edu/10479201/Fundamentos_de_robotica
15. NAGATA, F y WATANABE, K. 2013. Controller design for industrial robots and machine tools: Applications to manufacturing processes. Archivo pdf. Disponible en: https://es.es1lib.org/dl/2270023/21a060
16. TORRENTE, O. (2013). Arduino. Curso práctico de formación. Libros RC. Archivo pdf. Disponible en: https://es.es1lib.org/dl/3487874/64d559
17. LUNDBLAD, F y FARMANGEN, H. 2019. Elbow design, servo motor selection and control implementation of the Agile Parallel Kinematic Manipulator. Archivo pdf. Disponible en: https://lup.lub.lu.se/student-papers/search/publication/8997583
18. FARFAN PERDOMO, J; RAMÍREZ AGUILAR, M.A; MANUEL RAMÍREZ CONTRERAS, H. M y RODRÍGUEZ TENORIO, C. Robot SCARA de 4 Grados de Libertad con Efector Mecánico. Departamento de Ingeniería Mecánica e Industrial, Universidad Nacional Autónoma de México. Archivo pdf. Disponible en: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwit9fDi3uL4AhXHgv0HHVxFBBMQFnoECAQQAQ&url=https%3A%2F%2Fjorgefarfanblog.files.wordp

ress.com%2F2017%2F06%2Fpaper_scara_psm1.pdf&usg=AOvVaw0mwGCEUZEikR1ymgqLF5rl

19. CORTÉS PAREJO, J. 2008. La representación Denavit-Hartenberg. Archivo pdf. Disponible en: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjahInn3-L4AhUPt6QKHWxABq8QFnoECAMQAQ&url=https%3A%2F%2Fpersonal.us.es%2Fjcortes%2FMaterial%2FMaterial_archivos%2FArticulos%2520PDF%2FRepresentDH.pdf&usg=AOvVaw0IwlcVG5r2yoscPiihiK-b
20. Normalización. Las normas UNE. Archivo pdf. Disponible en: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiGou-Bh-L4AhUM-BoKHdjoDZ0QFnoECAMQAQ&url=https%3A%2F%2Fyoquieroaprobar.es%2F5_bachiller%2F8%2Fnormalizacion.pdf&usg=AOvVaw0a6-UFIyXApnVtnqLKn8E0
21. MARÍN MARTINEZ, D. 2016. DESARROLLO DE UN SISTEMA ROBOTIZADO PARA INSPECCIÓN VISUAL DE PIEZAS. Pliegos de condiciones. Archivo pdf. Disponible en: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiX8dHQsYT5AhXuQEEAHaWZDQwQFnoECAUQAQ&url=https%3A%2F%2Friunet.upv.es%2Fbitstream%2Fhandle%2F10251%2F75785%2FPliego%2520de%2520condiciones_14690969696177003302072675876288.pdf%3Fsequence%3D3&usg=AOvVaw34rpzSvA5jd3y2vJFk4hux
22. LEGARRETA, J y MARTÍNEZ, R. 2018. Dinámica de robots y control. Archivo pdf. Disponible en: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwj32-7LqIX5AhWhQ_EDHcmaAPcQFnoECA0QAQ&url=https%3A%2F%2Focw.ehu.eus%2Fmod%2Fresource%2Fview.php%3Fid%3D38616&usg=AOvVaw1sMuhlDtz85IH7eKRqVu6p