Universidad Zaragoza
1542

Trabajo Fin de Grado

Planificación de trayectorias y control de un robot manipulador con 5 grados de libertad

Trajectory planning and control of a robotic manipulator with 5 degree of freedoom

Autor/es

Juan Calvo Mora

Director/es

Antonio González Sorribes
Alfonso Blesa Gascón

Escuela Universitaria Politécnica de Teruel / Unizar

2019

A robot may not injure a human being or, through inaction, allow a human being to come to harm.

A robot must obey the orders given it by human beings except where such orders would conflict with the First Law.

A robot must protect its own existence as long as such protection does not conflict with the First or Second Laws.

Runaround, Isaac Asimov, 1942.

Para mi familia,

Agradecimientos

Un trabajo de investigación nunca sería posible sin el compromiso de un grupo de personas que, ofreciendo su aportación personal, ya sea académica o bien sentimental, se preocupen por guiar, animar o motivar al autor.

He tenido el gusto de cursar el grado de Ingeniería Electrónica y Automática en la Universidad de Zaragoza, donde he recibido la oportunidad de adquirir valiosos conocimientos, los cuales han resultado esenciales en este trabajo de fin de grado. Primeramente, hay que hacer una especial mención a todo el personal académico de la Escuela Politécnica de Teruel, en especial a mi tutor Antonio González Sorribes y cotutor Alfonso Blesa Gascón por su voluntad y ayuda en guiarme durante todo el proceso de desarrollo del proyecto. Posteriormente, tengo que dar las gracias a Carlos Catalán Cantero, quien, juntamente con Alfonso, fueron los responsables, entre otros ejemplos, de proporcionarme la idea de trabajar con una comunicación TCP/ IP.

Tanto la familia como los amigos han jugado, en general, un papel realmente importante en el progreso de este estudio. Por una parte, quisiera destacar a Juan Calvo Besolí, mi padre, quien me ha echado una mano con el montaje del robot. Por otra parte, mi hermano David Calvo Mora me ha dado algunos consejos en la gestión del informe. Mi madre Cinta Mora Micolau y mis cuatro abuelos, a pesar de presentar un gran desconocimiento acerca del tema del proyecto, me han animado y motivado a seguir adelante en todo momento.

Resumen

Este proyecto se centra en la utilización de un método suavizado para la generación de trayectorias, de un brazo robótico, en un espacio cartesiano.

Se describirán dos principales bloques (*hardware* y *software*) que conforman el diseño integro de este trabajo. El *hardware* utilizado para la acción de los motores, mientras que, el *software* se implementarán algoritmos para cumplir el propósito de este trabajo, tanto la utilización de S-Curve y Curvas de Bézier que conformarán el método suavizado de trayectorias, como el algoritmo de Bresenham, que dará lugar a la sincronización de todas las articulaciones existentes del robot.

La verificación se lleva a cabo mediante el uso del robot *OpenSource BCN MOVEO*. Los resultados verifican la viabilidad del método utilizado.

Palabras clave: Trayectorias · S-Curve · Curvas Bézier · Bresenham

Índice

Índice de tablas

Índice de figuras

1. Introducción

Los grandes avances de la robótica, desde los conceptos básicos hasta la interacción hombre–máquina con algoritmos de inteligencia artificial, se van ampliando frecuentemente. Estos avances de los robots en la sociedad nos han aportado muchísimos cambios afectando, de forma directa, al mercado, ya que son prácticamente parte de nuestras vidas.

Según la Real Academia Española, un robot se define como "máquina o ingenio electrónico programable que es capaz de manipular objetos y realizar diversas operaciones." En resumen, es una máquina capaz de trasladar y/o rotar para hacer una determinada tarea.

El Comité Español de Automática CEA clasifica los robots en función de su uso. De esta forma aparecen distintos tipos de robots, de los cuales destacan los robots aéreos, robots autónomos, robots caminantes, etc.

2. Objetivos

El principal objetivo de este proyecto es estudiar la generación de trayectorias a partir de unos puntos dados y, posteriormente, su implementación en los motores para controlar el robot.

Se requiere la construcción de un brazo robot de cinco ejes que permita validar los algoritmos de cálculo de trayectorias y también se destaca la integración de plataformas *hardware* y *software*, así como los protocolos de comunicación para el acceso local y remoto al robot y a cada eje

Se entiende la generación de trayectorias como el desarrollo, la implementación o la validación experimental de una estrategia de planificación de trayectorias y de algoritmos de control.

La finalidad de este proyecto se relaciona con el propósito personal. Se ampliarán los conocimientos acerca de la robótica, y se pondrán en práctica todos los conocimientos aprendidos en la universidad, siempre aportando distintos puntos de vista acerca del tema.

3. Hardware

Antes de controlar un robot o crear el algoritmo, es necesario identificar qué componentes o actuadores se deben utilizar para su correcto funcionamiento. Para ello, la elección adecuada del hardware constituye una parte fundamental en este trabajo, viéndose su eficiencia claramente favorecida.

Para poder desarrollar este proyecto, se ha utilizado el robot *OpenSource BCN3D MOVEO de BCN3D Technologies* [1]. La implementación del *hardware* y el *software* se ha realizado siendo fiel a las características del robot, en cuanto a sus dimensiones como sus cinco articulaciones.

Con la finalidad de poder imprimir las piezas del robot y como fase previa al desarrollo del presente proyecto, se ha se ha diseñado y construido una impresora 3D de dimensiones 50 cm de largo y 50 cm de ancho.

Todas las piezas se han imprimido con un filamento de poliácido láctico (PLA). Cabe citar que se ha utilizado este tipo de material ya que es biodegradable y se obtiene a partir del almidón de maíz.

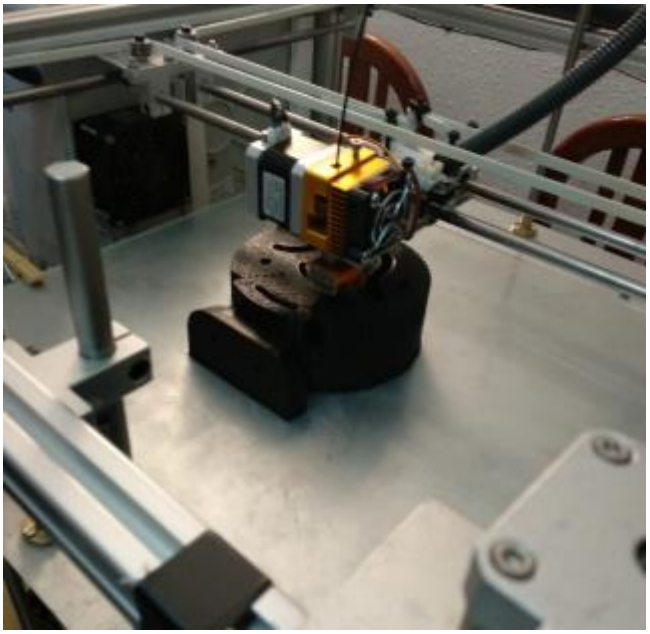


Figura 1. Impresión piezas robot. [Fuente propia].

Para la impresión de las piezas se han utilizado en total tres bobinas de amarillo y dos de negro con un peso de cada bobina de 1kg de PLA. También se ha elegido este material por ser más ligero en cuanto peso y poder despreciar las inercias sufridas por el movimiento del robot.

Se trata de un robot de cinco articulaciones accionados por motores, es decir, con una configuración de cinco grados de libertad. Por lo que respecta, todas las articulaciones tendrán un rango de operación de -180 a 180 grados, para evitar colisiones del robot con su estructura. En la *Figura 2* aparece la ubicación de estos motores. De esta forma, ha sido necesario adaptar el hardware al modelo del robot.

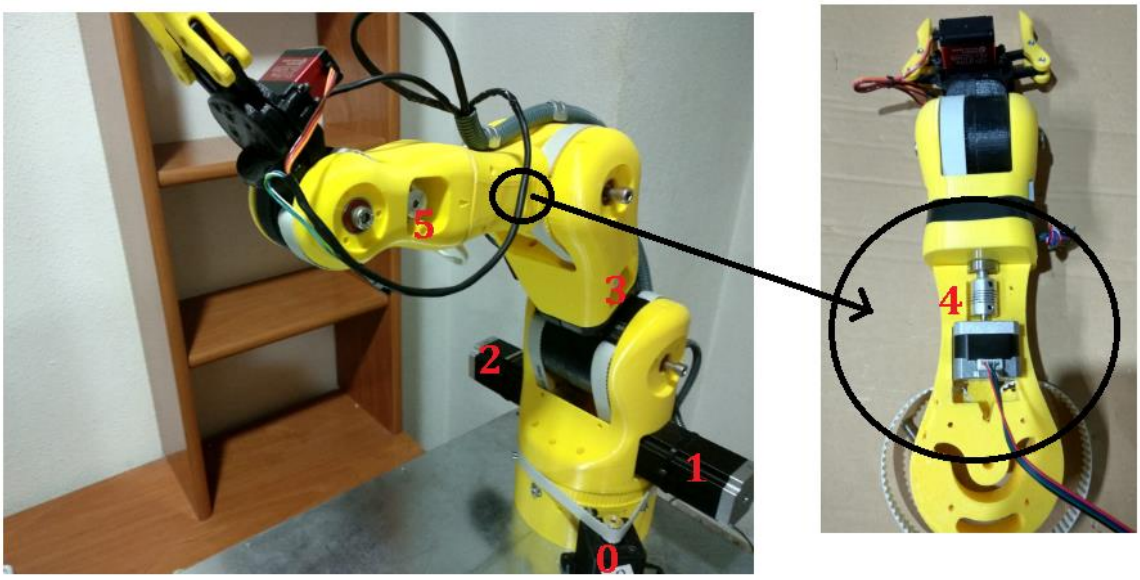


Figura 2. Numeración motores robot. [Fuente propia].

Si se sitúa este proyecto en una situación desfavorable, de forma que todos los motores actúen a la vez se obtiene un consumo total de 318.4 W.

Se tiene que considerar las limitaciones físicas en cuanto peso del robot en el punto más desfavorable, así poder determinar que carga máxima se puede añadir en el extremo del robot (aproximadamente 650 g). En este caso se tiene una carga de 15.1312 N distribuida por todo el brazo.

Todas las características técnicas y especificaciones de los fabricantes se detallan en el CD adjunto ubicado en la contraportada de este proyecto.

3.2 ESP32-DevKitC V2

Para la programación del sistema, se ha empleado el circuito integrado ESP32, de bajo coste y con alta integración SoC, haciendo uso de las siguientes prestaciones: un microprocesador con núcleo dual, una pila TCP/IP, conexión WIFI y 18 puertos E/S disponibles.

Este circuito integrado pertenece a la compañía Espressif Systems y fabricado por TSMC.

Por lo que a la ESP32-DevKitCV2 se refiere, la propia compañía ofrece esta descripción: *ESP32-DevKitC V2 is a small-sized ESP32-based development board produced by Espressif. Most of the I/O pins are broken out to the pin headers on both sides for easy interfacing. Developers can either connect peripherals with jumper wires or mount ESP32-DevKitC V4 on a breadboard.*

Por lo tanto, la fácil conexión entre la programación en lenguaje C++ y los periféricos, la conexión WIFI facilita el desarrollo de este estudio.

3.3 Motores

Como motores, se han utilizado los mismos del *OpenSource BCN MOVEO*, ya que se ha considerado el más apropiado por su diseño.

Se trata de motores *paso a paso*, que son síncronos capaces de dividir una rotación en un gran número de pasos y, en consecuencia, se puede conseguir el control, en grados, de las articulaciones del sistema robotizado. Estos tienen muchas otras aplicaciones, como por ejemplo en las impresoras 3D.

A diferencia de otros tipos de motores como los servomotores, los *paso a paso* se pueden encontrar en un sistema en bucle abierto. Esta característica facilita el problema de diseño.

Existen dos tipos en función de las disposiciones del devanado de las bobinas electromagnéticas, los bipolares y los unipolares.

3.3.1 Motores unipolares

El motor *paso a paso* unipolar funciona con un devanado con una toma central por bobina, cada sección activándose para cada una de las direcciones del campo magnético.

Se distinguen de los bipolares según si tienen cinco cables de conexión a la salida o no. Estos, a diferencia de los otros, son más fáciles de controlar.

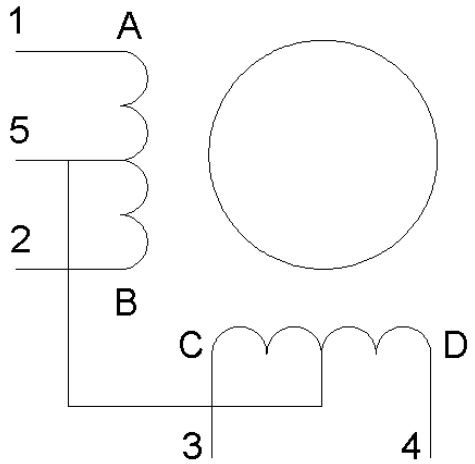


Figura 3. Motor paso a paso unipolar. [Fuente propia].

3.3.2 Motores bipolares

Estos actuadores se caracterizan por tener cuatro cables de salida. Para estos, se necesita unas controladoras de motores debido a que requieren un cambio de la dirección del flujo de corriente a través de las bobinas en la secuencia correcta para poder realizar el movimiento.

Cómo esta variación de la dirección del campo magnético creado por el estator producirá un movimiento del rotor, se intentará alinear el imán del rotor con el campo magnético inducido de las bobinas del estator.

En este caso, se tiene un control casi perfecto, con una posición exacta de ±5%, según características técnicas del fabricante.

En este proyecto se han usado estos por su precisión. Generalmente, existen tres tipos de modos de paso:

- Paso entero (*Full step*)

- Paso medio (*Half step*)

- Micro paso (*Micro step*)

3.3.2.1 Paso entero

En una operación de paso entero, cada paso tendrá un movimiento de 1.8 grados, dados por las características técnicas del fabricante. Es decir, para conseguir una revolución completa, se necesitan 200 pasos. En este proyecto se han utilizado los siguientes motores con el mismo ángulo de paso:

-3 Nema 17 (Articulación 1, 3 y 4)

-2 Nema 23 (Articulación 2)

-1 Nema 14 (Articulación 5)

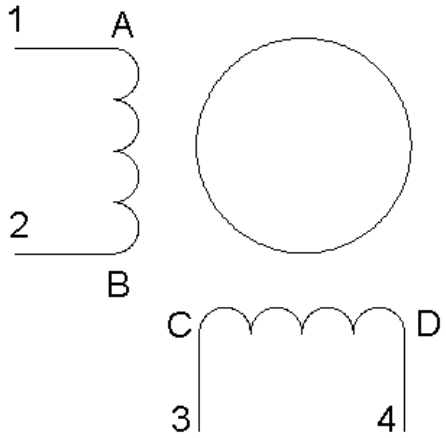


Figura 4. Motor paso a paso bipolar. [Fuente propia].

Esta operación es posible gracias a la activación de los devanados del estator. De acuerdo con esta operación los dos devanados se activan al mismo tiempo durante la operación de doble fase; el par y la velocidad son mayores, mientras que, si la operación

se activa una sola fase, se requerirá menos cantidad de energía, donde se verán desfavorecidas estas características.

Figura 5. Interior motor bipolar paso a paso. [Fuente propia].

En consecuencia, para la operación de paso entero existirán dos posibilidades, de acuerdo con la aplicación que nos interese:

-Modo doble fase

-Modo una fase

Modo doble fase

En el modo de doble fase los motores operan con las dos fases que quedan excitadas al mismo tiempo. Este hecho proporciona un mayor torque y una velocidad más elevada, pero también hace que se necesite más energía.

Modo una fase

En el modo de una fase, los motores operan con solo una fase excitada a al mismo tiempo. De esta manera, esta operación requiere menor cantidad de energía.

Tabla 1. Motor bipolar con modo operación una fase.

Paso	A	B	C	D
1	1	0	0	1
2	0	1	0	1
3	0	1	1	0
4	1	0	1	0

3.3.2.2 Paso medio

El rotor se mueve a través de la mitad del ángulo de la base en un solo paso, lo que se traduce en un par motor mejorado que la operación de un solo paso de fase única. También se ven duplicadas la suavidad de rotación y resolución.

En el modo de medio paso, tienen solo la mitad del ángulo de paso básico en comparación con el modo de paso completo, lo que resulta en un control de movimiento más preciso y un rendimiento del motor más suave. La resolución también se ve incrementada. Empleando esta configuración, la producción del motor es menor en relación con otros modos.

Tabla 2. Motor bipolar con modo operación paso medio.

Paso	A	B	C	D
1	0	0	1	0
2	0	1	1	0
3	0	1	0	0
4	0	1	0	1
5	0	0	0	1
6	1	0	0	1
7	1	0	0	0
8	1	0	1	0

3.3.2.3 Micro paso

En el modo de micro paso, se dividen los pasos del motor hasta 256 veces, lo que mejora la suavidad a baja velocidad y los efectos de la resonancia de baja velocidad. Pero en este modo, el motor produce menos par en comparación con otros modos.

En la operación de micro paso, el ángulo básico se divide en valores de minutos, también en un máximo de 256 veces. La operación del micro paso es preferible cuando se requiere una mayor suavidad de rotación y más precisión. Desafortunadamente, el torque disminuye hasta un 30% según esta operación.

En este caso, se tiene un control casi perfecto, con una posición exacta de ±5% según las características técnicas de fábrica. Por lo tanto, los motores del proyecto se configurarán como si fueran micro pasos.

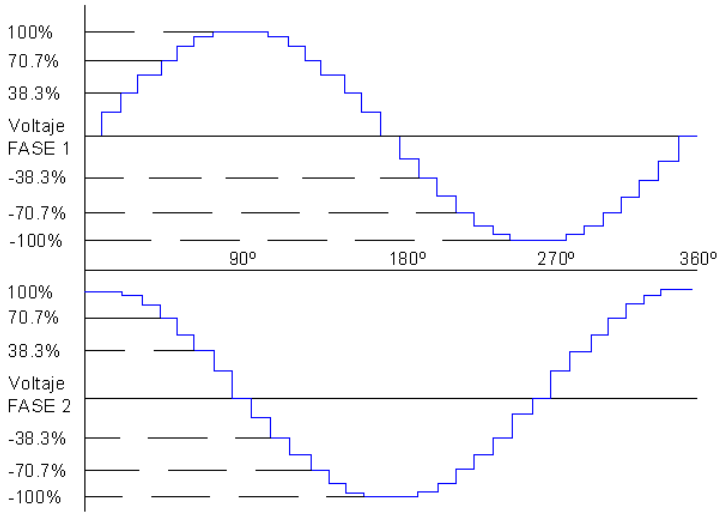


Figura 6. Representación voltaje frente modo operación de micro pasos. [Fuente propia].

3.3.3 Consideraciones de frecuencia

Las características de la frecuencia del torque de un motor paso a paso vienen dadas por la variación del torque electromagnético en función de los pulsos por segundo.

En la *Figura 7* se detallan las dos curvas características de estas consideraciones en frecuencia. La curva que se denota con una línea de color rojo que se conoce como par de torsión. Esta, muestra el máximo valor de pasos para los distintos valores del par de carga a los que el motor puede arrancar, detener o invertir.

La otra curva representada por la línea de color azul se conoce con el nombre de características de par de extracción. Esta curva muestra el máximo valor de pasos del motor, el cual puede funcionar para los distintos valores de par de carga. Pero en este caso, no se puede iniciar, detener o revertir a este ritmo.

Para entenderlo de forma correcta, en dicha figura aparece un ejemplo con las denotaciones T1, P1 y P2, que a continuación se desarrollará.

El motor podrá arrancar, parar o invertir para el torque T1 siempre que la frecuencia sea menor que P1. Una vez arrancado el motor, los pulsos o pasos se pueden aumentar para la misma carga. Después de iniciar el actuador, los pasos pueden aumentar hasta P2 sin perder el sincronismo.

Si los pulsos aumentan más allá de P2, perderá el sincronismo, con lo cual se puede decir que el motor paso a paso pierde pasos.

En resumen, el área comprendida entre las curvas roja y azul es el rango que siguen los motores sin perder el sincronismo (denominado *Slew Range*).

Para este proyecto se considera, según las características del fabricante, la máxima frecuencia de trabajo de los motores, es decir, cuantos pulsos como máximo, en un segundo, se puede enviar a través de nuestro micro controlador.

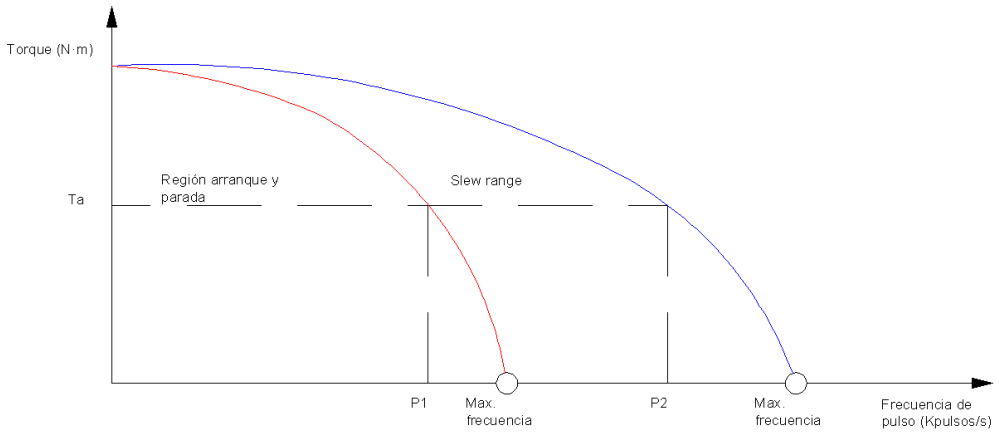


Figura 7. Relación torque y frecuencia de pulsos. [Fuente propia].

3.4 Drivers

Para hacer un control sobre las bobinas de los motores paso a paso como se ha explicado anteriormente, es necesaria la utilización de unos drivers, ya que son los encargados de controlar esos micros pasos.

Previamente, se quería controlar los motores paso a paso mediante controladoras o drivers DVR8825, lo que llevan las impresoras 3D. Pero con estas se tenía una delimitación de intensidad, ya que los motores de la articulación dos tienen una corriente de 3A. Por este motivo, se ha descartado la utilización de estas controladoras.

Finalmente se ha optado por la utilización de la tarjeta con el integrado TB6560. Con esta tarjeta se puede hacer un control de micro pasos de 0.1125 grados.

Para adaptar los motores a esta tarjeta, se ha hecho de acuerdo a las especificaciones técnicas del fabricante, como se explicarán detalladamente a continuación.

En la siguiente tabla se muestra la distinta simbología utilizada en la placa y su correspondiente descripción. Ya que se ha tenido en consideración, a la hora de cablear el hardware.

Tabla 3. Explicación simbología drivers TB6560.

Simbología	Descripción
A+ A-	Conexión de los puntos A y B según *Figura 4.*
B+ B-	Conexión de los puntos C y D según *Figura 4.*
+24V GND	Alimentación de 24V proveniente de la fuente de alimentación.
EN+ EN-	Activación de los motores (*Enable*)
CLK+ CLK-	Pulso o *step* para hacer los pasos, medio pasos o micro pasos.
CW+ CW-	Dirección del motor.

En referencia a las entradas nombradas previamente en la *Tabla* 3, es decir, EN+, CLK+, CW+ deberán ir conectadas a la salida del micro controlador ESP32, configurando este como pines de salida (garantizando una salida de 5V).

Mientras que los pines EN-, CLK- y CW- irán conectados a masa de nuestra placa ESP32.

Una conocido los aspectos principales de nuestra controladora. Se necesita ajustar mediante los *switches* 1, 2, 3 y S1, la corriente de trabajo de nuestro motor a conectar.

Por ejemplo, si a una controladora se conecta un Nema 17, se necesitará conocer la corriente de trabajo según especificaciones técnicas del fabricante, que será de 1.68A.

De este modo, se ajusta la corriente de trabajo según la controladora de la siguiente configuración de 1.6A: SW1-ON, SW2-OFF, SW3-OFF y S1-OFF.

En este proyecto, como se tenían distintas corrientes de trabajo según el motor, estas se han configurado según las características técnicas de cada uno.

-3 Nema 17: 1.68A

-2 Nema 23: 3A

-1 Nema 14 0.8A

Una vez ajustada la corriente de trabajo de cada motor, hará falta asignar la corriente máxima con el S2. Eso, determinará que si la corriente aumenta más de un 20% o 50% con respecto a su valor nominal la corriente de trabajo se detendrá

La utilización de motores *paso a paso* permite el control en bucle abierto de la posición de cada motor. Por ese motivo, existen distintos tipos de pasos, que se diferencian según los *switches* S3 yS4.

En este trabajo se ha utilizado una alta resolución de 1/16, es decir, cada pulso hace 1.8 grados, por lo tanto, se dispone de una resolución de 0.1125 grados.

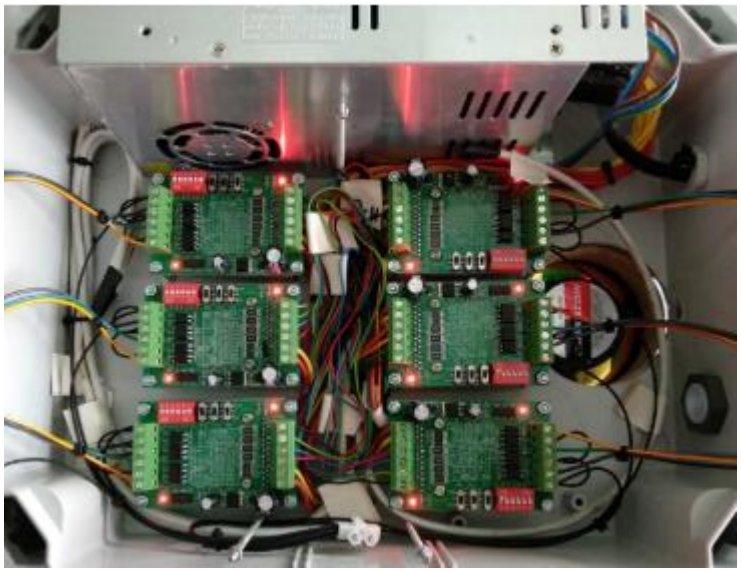


Figura 8. Distribución drivers TB6560. [Fuente propia].

Por otra parte, se tiene un parámetro llamado *decay setting*, que es el factor de decaimiento de la corriente de *on* a *off*. Para algunos motores, la transición entre los pasos es suave si la corriente decae de forma rápida, mientras que, para otros, esta es suave si decae lentamente. El parámetro se ajusta con el S5 y S6.

3.5 Fuente de alimentación

Se ha utilizado una fuente de alimentación de 24V, 20A y 480W, que es la encargada de alimentar toda la electrónica anterior. Se han hecho los cálculos correspondientes para garantizar una intensidad suficiente para el correcto movimiento de los motores.

3.6 Conexiones

En el *Anexo 8.5* se detalla el esquema eléctrico de este proyecto. Así mismo a continuación se muestra una tabla donde se detalla cada entrada con su correspondiente función y nombre de variable. Las entradas y salidas corresponden al ESP32 capaz de controlar cada una de las articulaciones.

Tabla 4. Tabla de conexiones.

Articulaciones	Variable	I/O
Motor 1	STEP_PIN_1	GPIO 18
	DIR_PIN_1	GPIO 26
	EN_PIN_1	GPIO 32
Motor 2 y 3	STEP_PIN_2	GPIO 17
	DIR_PIN_2	GPIO 16
	EN_PIN_2	GPIO 4
Motor 4	STEP_PIN_3	GPIO 5
	DIR_PIN_3	GPIO 14
	EN_PIN_3	GPIO 12
Motor 5	STEP_PIN_4	GPIO 15
	DIR_PIN_4	GPIO 2
	EN_PIN_4	GPIO 21
Motor 6	STEP_PIN_5	GPIO 13
	DIR_PIN_5	GPIO 22
	EN_PIN_5	GPIO 23

4. Software

Una vez descrito todo el hardware para controlar el robot, en este apartado se describirá el algoritmo implementado para calcular la trayectoria correspondiente a partir a N puntos con sus correspondientes velocidades, y procesarlos para poder ejecutar sobre los motores del robot.

En primer lugar, se expondrá una visión general y finalmente se mostrará una de específica de cada uno de los elementos que forman el algoritmo.

La programación consta de dos partes: en la primera, se ha utilizado el lenguaje de programación MATLAB y, en la otra, la programación en C++ del propio micro controlador, ESP32.

Para unir de forma eficaz MATLAB, ejecutado sobre el sistema operativo Windows, y el ESP32, se necesita disponer de algún sistema de envió de información o comunicación. Por este motivo MATLAB nos calcula los N puntos descritos en la trayectoria y el ESP32 actúa sobre los motores, en términos generales.

En los siguientes apartados se detallarán las características del propio algoritmo clasificándolos en dos apartados, el algoritmo correspondiente a MATLAB y el algoritmo correspondiente al ESP32.

Una de las problemáticas de tener dos algoritmos, uno en MATLAB y el otro en el micro controlador, ESP32, es que se pudieran intercambiar información entre sí en tiempo real.

En un primer momento, se barajó la posibilidad de disponer de una comunicación puerto serie entre ellos. Una comunicación puerto serie es una comunicación a través de la cual se envía o se recibe la información enviando bits en puerto serie.

Esta posibilidad se descartó ya que como se trabaja con un sistema operativo Windows, dificultaba el envío de información al micro controlador. Cada dato a enviarse tardaba aproximadamente 1.2s, mediante una monitorización de los puertos del ordenador. Eso se debía a que el propio sistema operativo daba prioridad a otras tareas antes que al acceso, envío y recibimiento de datos del puerto serie. Finalmente, se utilizó la comunicación TCP/IP socket.

Toda la programación mencionada se detalla en el CD adjunto ubicado en la contraportada de este proyecto.

4.1 Comunicación TCP/IP

Antes de adentrarse en el entorno de programación de MATLAB y en el propio algoritmo, se ha de hacer mención a la utilización de una comunicación socket TCP/IP.

Cuando se trabaja en entornos inalámbricos, es importante conocer y entender los conocimientos relacionados con el WIFI. A un alto nivel, el WIFI es capaz de poder

conectarse en TCP/IP a través de un enlace de comunicación inalámbrica. El WIFI es un conjunto de protocolos descritos en la IEEE 802.11.

Un Punto de Acceso es un dispositivo de red que interconecta dispositivos de comunicación inalámbricos, para poder formar una red inalámbrica. Son dispositivos que son intermediarios entre por ejemplo un ordenador y una red local o internet. Eso permite facilitar la conexión de varias máquinas cliente sin la necesidad de un cable.

Se tiene un dispositivo llamado Punto de Acceso (AP) que actúa como el centro de todas las comunicaciones. Las conexiones de WIFI se forman al Punto de Acceso, a través de dispositivos llamados estaciones y el tráfico TCP/IP fluye a través del Punto de Acceso a internet.

Un conjunto de estaciones que quieren comunicarse entre sí se denomina Conjunto de Servicios Básicos (BSS). De esta forma, todas las comunicaciones entrantes y salientes desde una estación individual se redirigirán a través del punto de acceso. Una estación solo puede asociarse con un único punto de acceso en un momento dado.

Cada participante o cliente tiene un identificador único llamado dirección MAC o dirección física. De la cual esta dirección MAC consta de 48 bits.

Cuando se tienen múltiples Puntos de Acceso en un sitio determinado, la estación necesita saber con cuál deber conectarse. Por lo tanto, cada Punto de Acceso tiene un identificador llamado SSID. Este identificador de 32 caracteres representa el destino de los paquetes de información enviados a través de la red. Normalmente protegidos por un Acceso de WIFI protegido, ya que es un sistema para proteger las redes inalámbricas.

Según la Real Academia de Ingeniería, el protocolo TCP/IP se define como "Conjunto de protocolos desarrollados por el Departamento de Defensa de Estados Unidos para comunicaciones a través de redes interconectadas y posiblemente heterogéneas, conocido también como arquitectura ARPANET." En resumen, permite identificar al grupo de protocolos de red que hay en Internet y posibilitando la transferencia de datos entre redes de ordenadores.

Una vez descrito cómo funciona la red, hace falta centrarse en la comunicación socket, pare ello, primero se ha de conocer una serie de conceptos.

Cada dispositivo en una red TCP/IP tiene una dirección IP. Esta dirección IP es un número o etiqueta que está asignado a cada dispositivo conectado a una red que utiliza el Protocolo de Internet para la comunicación. La dirección IP a diferencia de la MAC, nos identifica la interfaz de red y el direccionamiento de la ubicación.

Por ejemplo, se podría plantear una dirección IP que identificara el ordenador. No obstante, una dirección IP no es suficiente para hacer aplicaciones en la red. La IP sirve para identificar el ordenador y el puerto de red identifica el servicio o aplicación ejecutado en el ordenador.

Si se hace una analogía sobre estos conceptos, un bloque de pisos sería la dirección IP que correspondería a la dirección de la calle. Mientras cada piso de este bloque de pisos correspondería al puerto.

Un puerto es un entero positivo de 16 bits, del cual su rango vale de 0 a 65535. Por ejemplo, el puerto 0 está reservado y no puede ser utilizado.

La suma de una dirección IP con el número de puerto es lo que se llama dirección socket. La conexión de dos de dos ordenadores se utiliza un socket, es decir, que sirve como enlace y permite que el proceso envíe y reciba datos a través de la red.

El sistema operativo de red (NOS) tiene la tarea de transmitir los datos salientes de todos los puertos de la aplicación a la red y reenviar los paquetes de red que llegan a los procesos haciendo coincidir la dirección IP y el número de puerto del paquete.

Se debe tener cuidado con la utilización de estas comunicaciones, que no haya varios programas que intenten usar el mismo número de puerto en la misma dirección IP con el mismo protocolo.

En este proyecto se dispone de un servidor creado con MATLAB donde le se abre un puerto y se usa la IP que identifica nuestro propio ordenador dentro de la red local.

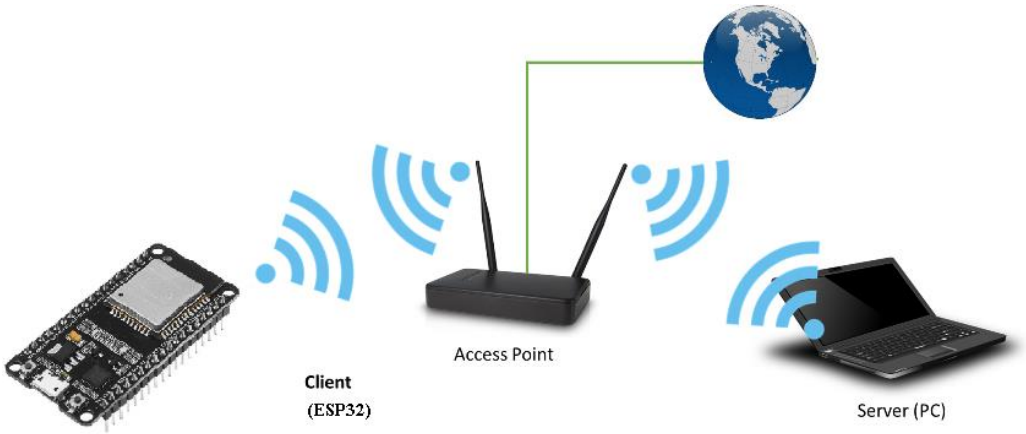


Figura 9. Descripción gráfica comunicación cliente y servidor. [Fuente propia].

4.2 MATLAB

En este apartado se exponen los principales bloques que conforman el algoritmo de MATLAB. Una vez explicados de forma general, se procederá en los apartados siguientes con la explicación de cada uno de ellos en detalle.

Un robot cuando se programa en lenguaje de programación, como por ejemplo Rapid, es necesario asignarle unos los puntos por donde se quiere que pase. De esta forma de pasar de un punto a otro punto se requiere definir una trayectoria.

Según la definición de trayectoria es el lugar geométrico de las trayectorias sucesivas por las que pasa un cuerpo en su movimiento.

Como aparece en la *Figura 10*, es necesario tener unas condiciones iniciales de posición (conocer todos los puntos donde se desea que pase el robot) y velocidad, y unos límites que delimitarán la propia velocidad a lo largo de la trayectoria.

Una vez generado los distintos puntos que describen una trayectoria, será necesario pasar cada uno de los puntos de la trayectoria descritos en el espacio XYZ en las articulaciones en forma de grados. Este procedimiento es lo que se conoce como cinemática inversa, dónde es una técnica que determina el movimiento de una cadena de articulaciones para poder lograr que el extremo final del robot alcance la posición correcta, es decir cada uno de los puntos de cada trayectoria.

Una vez se hayan obtenido los grados de cada articulación para poder alcanzar cada una de ellas, se debe convertir estas articulaciones a pasos a partir de la relación de que un paso equivale a 0.1125 grados, tal y como se ha configurado.

Finalmente, MATLAB genera un servidor donde el ESP32 se conecta mediante una comunicación socket TCP/IP que pasa, mediante la “nube”, cada uno de los pasos que debe hacer cada motor para alcanzar cada uno de los puntos cartesianos de cada trayectoria.

Interesa pasar los pasos de cada articulación al ESP32 para que finalmente actúe sobre los motores.

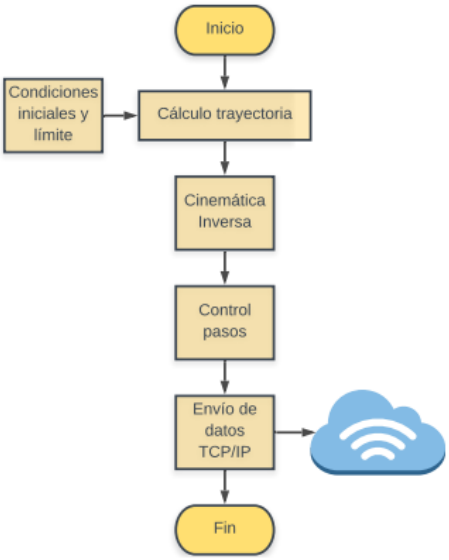


Figura 10. Diagrama de flujo algoritmo MATLAB. [Fuente propia].

4.2.1 Cálculo de trayectorias cartesianas

Dados unos puntos iniciales cartesianos, se puede construir una trayectoria cartesiana que los contenga.

Se puede afirmar que un punto cartesiano irá cambiando y describiendo una trayectoria distinta en función del tiempo desde t hasta el tiempo máximo que le costará en alcanzar cada punto final de la trayectoria (T).

A medida que aumenta el tiempo, es decir, de t hasta T el punto sigue el camino que se muestra en la *Figura 11*.

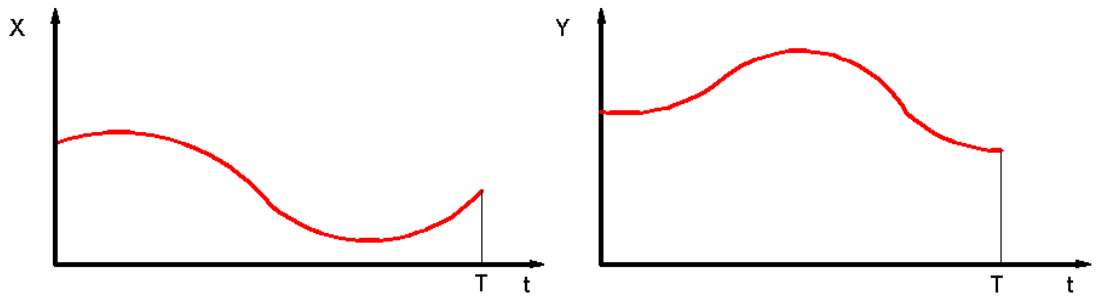


Figura 11. Evolución punto con el tiempo formando una trayectoria. [Fuente propia].

El en este proyecto se han utilizado dos algoritmos que conforman el cálculo de la trayectoria:

- *S-Curve*

- *Bézier*

4.2.1.1 ***S-Curve***

Uno de los puntos clave de este proyecto es la planificación de trayectorias a partir de la generación de *S-Curves*. Se ha utilizado este procedimiento en lugar de los perfiles trapezoidales ya que no genera tanto esfuerzo mecánico y se reduce las vibraciones del brazo robot. Estas características se deben por los perfiles de velocidad, ya que presenta una forma de campana.

Para poder conseguir estos resultados es necesario hacer un control del *jerk*. El *jerk* o sobre-aceleración es el cambio de la aceleración en un determinado periodo de tiempo. Por lo tanto, se define formalmente el jerk como la derivada de la aceleración con respecto al tiempo.

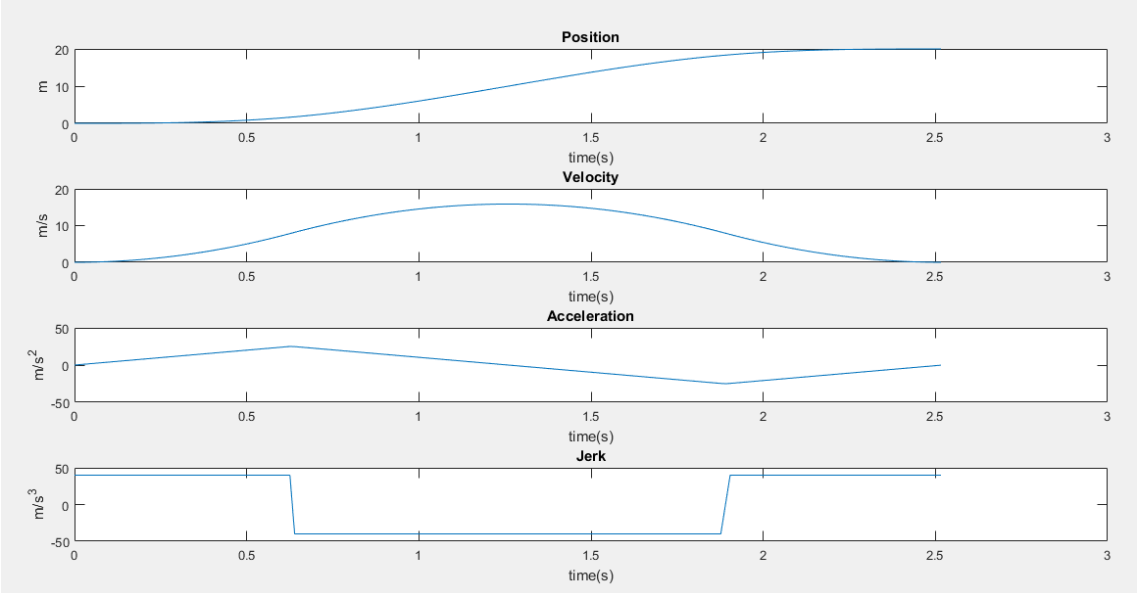


Figura 12. Representación posición, velocidad, aceleración y *jerk* de una S-Curve. [Fuente propia].

En la *Figura 13* se muestra un perfil de velocidades trapezoidal y en la *Figura 12*, se trata de una *S-Curve*. Si se observa ambas aceleraciones, en el perfil trapezoidal se aprecia una aceleración discontinua en el tiempo mientras que en el *S-Curve* se observa un triángulo o, en muchos otros casos, un trapecio. A causa de las discontinuidades de la aceleración en el perfil trapezoidal, la velocidad tiene forma de trapecio, mientras que en el otro se observa mucho más suave.

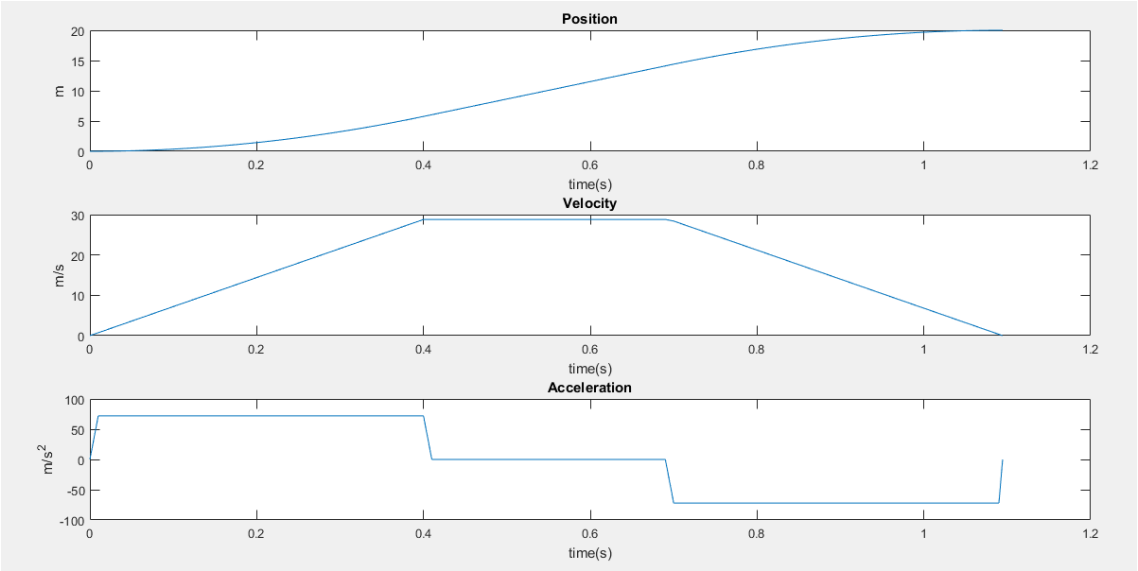


Figura 13. Representación posición, velocidad y aceleración perfil trapezoidal. [Fuente propia].

Estas discontinuidades en la aceleración son capaces de generar vibraciones que pueden incrementar la inercia de nuestro sistema. Esto supone la pérdida de micro pasos ya que saldrán de la zona de *Slew Range*.

Para calcular la trayectoria de un punto a otro, se necesita pasar el punto inicial, el punto final, su velocidad inicial y su velocidad final. Este hecho es de gran importancia ya que, si se quiere juntar con otra trayectoria, se tiene que adaptar bien y garantizar que no haya cambios bruscos, pérdida de control, etc. El *software* también presenta una conexión directa con el *hardware*, ya que se necesita ajustar la *S-Curve* y, para que no pierda micro pasos, hace falta disponer de unas condiciones límite en cuanto a velocidad, aceleración y *jerk*, para controlar que los motores trabajan en la zona del *Slew Range*.

Un punto queda definido por tres coordenadas, en el caso de un espacio tridimensional cartesiano. Si se tienen dos puntos y se quiere formar una trayectoria, el algoritmo tiene que ser capaz de obtener el punto inicial y final de cada coordenada y calcular sus correspondientes *S-Curves* y el tiempo total que le cuesta ir de un punto al otro. De este modo, se obtendrán tres *S-Curves* con distintos tiempos.

Para garantizar que las tres *S-Curves* tengan el mismo tiempo de duración, se ha implementado un promediador. Este promediador, va disminuyendo la velocidad y aceleración de cada S-Curve hasta cumplir con el tiempo deseado. Al disminuir estas dos magnitudes provoca un incremento del tiempo total,

Se ha construido una clase en MATLAB llamada *SCurve_a.m*. En esta clase existen dos constructores, dependiendo de los argumentos de entrada:

- *SCurve_a(q0,q1,v0,v1,vmax,amax, jmax)*: En esta función se le pasan las coordenadas y las velocidades de los dos puntos, el inicial (q0 y v0) y el final (q1 y v1); y las condiciones límite en cuanto a velocidad (vmax), aceleración (amax) y jerk máximo

(jmax). Esta función calcula las *S-Curves* para cada coordenada y devuelve los distintos tiempos para cada perfil de velocidad.

- *SCurve_a(q0,q1,v0,v1,vmax,amax, jmax, time)*: Además de heredar lo anterior, se le indica el tiempo que se quiere que duren las tres *S-Curves* para que, mediante el promediador, se ajusten los tiempos al tiempo deseado (time). Un ejemplo del ajuste de este tiempo se muestra en las *Figuras 14*, *15* y *16*, donde se ha configurado:

-SCurve_a(0,20,5,0,50,50,90,2);

-SCurve_a(0,20,5,0,50,50,90,4);

-SCurve_a(0,20,5,0,50,50,90,8);

Para encapsular un poco más el problema se ha diseñado una clase llamada *Planner_a.m*, que es el encargado de gestionar cada una de las *S-Curves* y fijar el tiempo deseado para cada una de ellas. Este encapsulamiento tiene una función que nos calcula las tres *S-Curves* (una para cada punto) con el tiempo deseado.

Esta clase primeramente calcula las tres *S-Curves* de cada trayectoria, así obtiene el tiempo que le cuesta a cada una. Una vez calculadas las *S-Curves*, se busca la *S-Curve* que tiene un tiempo mayor al resto y donde calcula este tiempo mayor al resto de *S-Curves*.

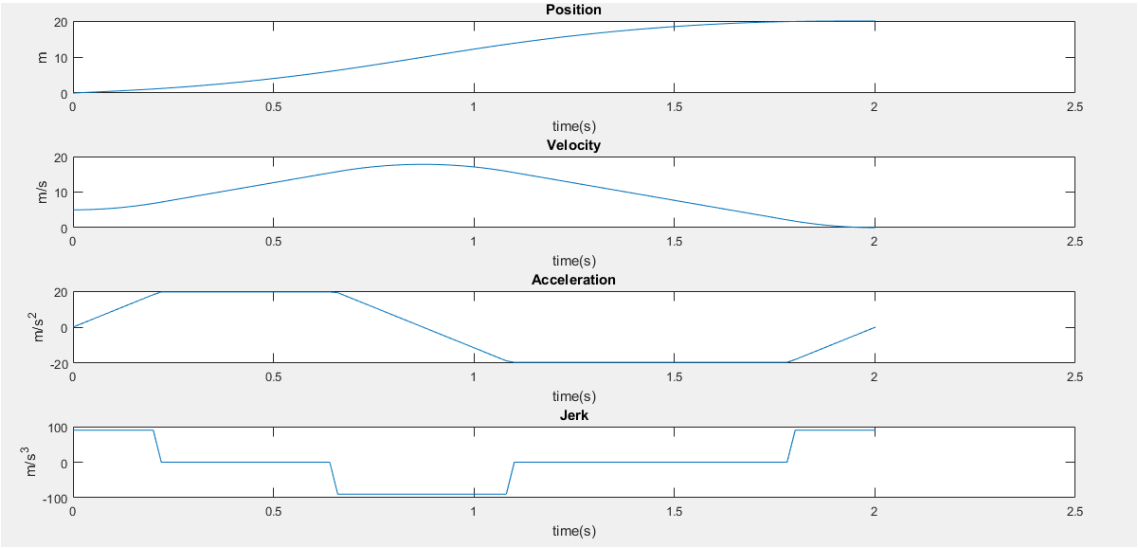


Figura 14. S-Curve con un tiempo de finalización dos segundos. [Fuente propia].

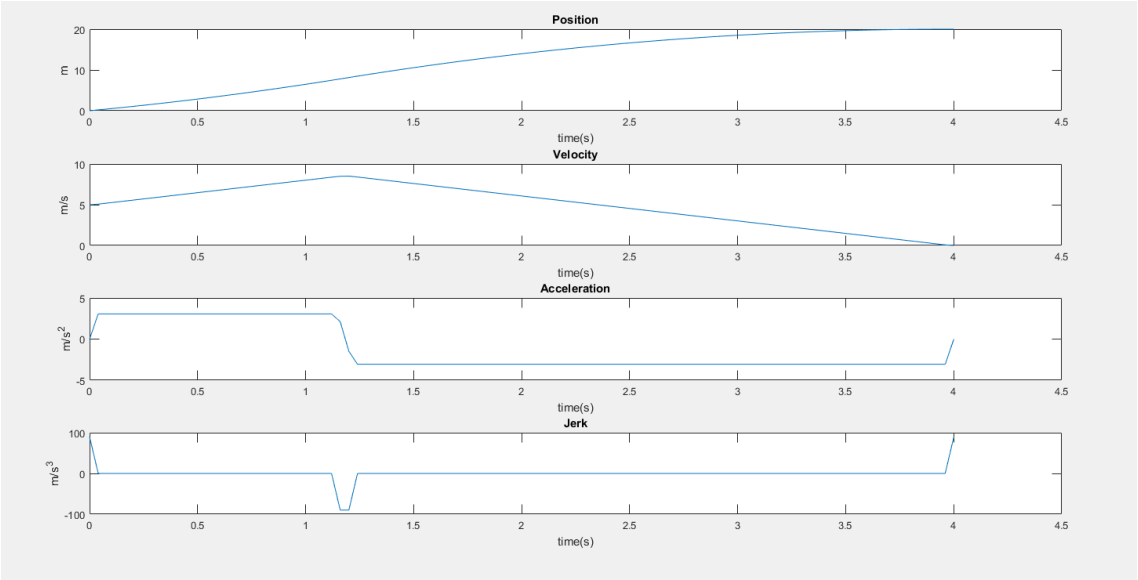


Figura 15. S-Curve con un tiempo de finalización cuatro segundos. [Fuente propia].

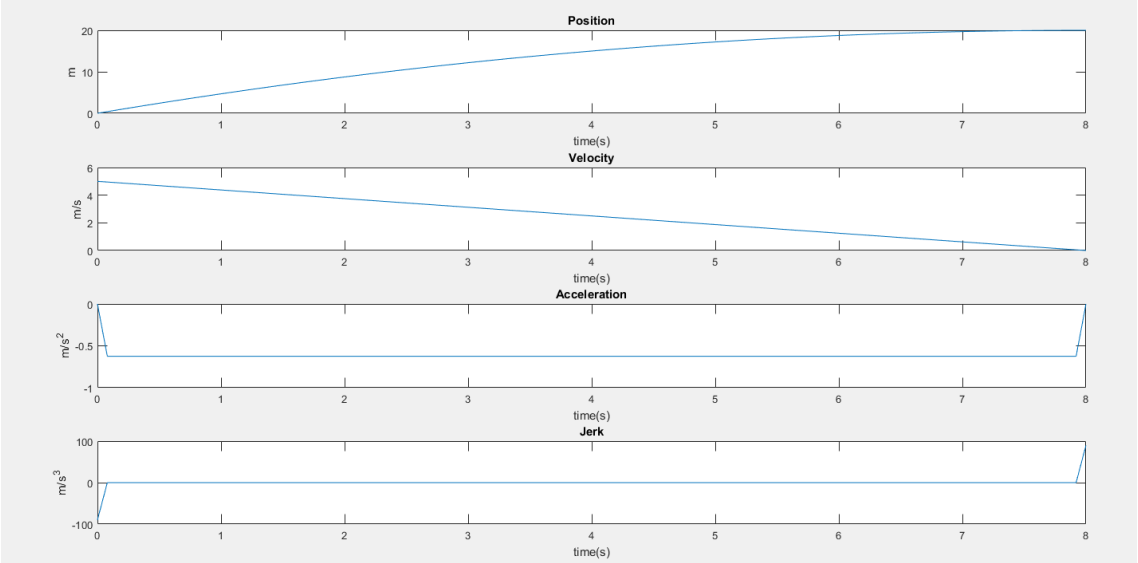


Figura 16. S-Curve con un tiempo de finalización ocho segundos. [Fuente propia].

4.2.1.2 Bézier

Las curvas de Bézier fueron creadas por el ingeniero Pierre Bézier, en Francia el 1962, a causa de un problema de diseño en la industria automovilística. La curva de Bézier es un método matemático que se empezó a utilizar en el diseño asistido por ordenador, como por ejemplo en programas de CAD, superficies de Bézier, etc.

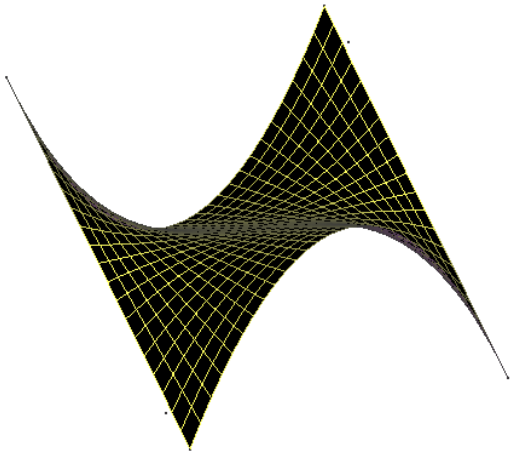


Figura 17. Generación superficie de Bézier. [Fuente propia].

Para la construcción de la curva de Bézier se utiliza el algoritmo de Casteljau. La idea de este algoritmo surge de los requisitos gráficos de informática, por el ingeniero Paul de Casteljau. Hay otra forma más eficiente, en la cual se ha implementado en este proyecto, que es mediante el método de Bézier. Para entender el concepto, este se explicará mediante el algoritmo de Casteljau.

Por motivos de propiedad intelectual, Paul de Casteljau no pudo publicar su algoritmo durante un tiempo, mientras que Pierre Bézier pudo divulgar y promover sus investigaciones. Por este motivo, existen dos métodos, y actualmente esas curvas llevan el nombre de curvas de Bézier.

Un punto del plano puede definirse por sus coordenadas cartesianas. Si por ejemplo se tiene un punto A con unas coordenadas (x0, y0, z0) y un punto (x1, y1, z1), se puede trazar una línea recta conociendo su posición. Si se desea unir estos dos puntos mediante una curva y no con una recta, se utilizará una curva de Bézier. La forma de curva se define por unos puntos imaginarios llamados puntos de control. Estos puntos de control se representan en la *Figura 18*.

Se puede construir de forma matemática una curva de Bézier, dados los puntos de control y los puntos donde empieza y acaba la curva que se quiere diseñar. En este caso, si se requiere mayor precisión en la forma de la curva se requerirán un mayor número de puntos de control. El punto donde empieza la curva corresponderá al primer punto de control y donde termina será el último. Para la explicación de este algoritmo se utilizarán cuatro puntos de control.

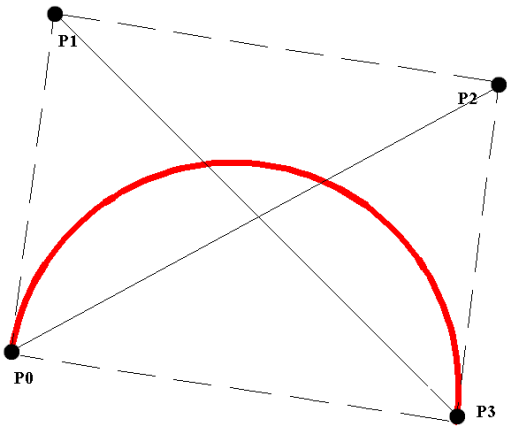


Figura 18. Definición de la curva de Bézier con cuatro puntos de control. [Fuente propia].

Si se supone que se dispone de cuatro puntos de control en el plano, según el algoritmo de Casteljau, se pueden hacer combinaciones convexas entre todas las parejas de los puntos, como se representa en la *Figura 19*.

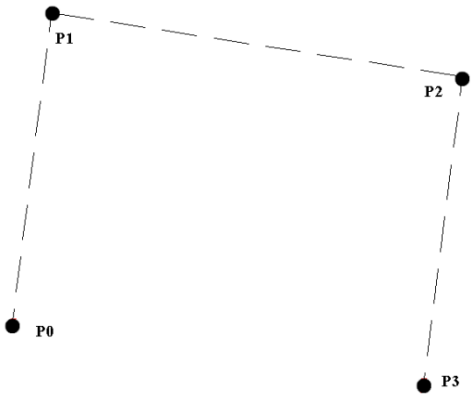


Figura 19. Combinación convexa. [Fuente propia].

En la *Figura 20* se muestra la combinación convexa que nos interesa, de este modo si se modifica la distancia entre P1 y P2 se podrá tener una curva más o menos estrecha.

Para el desarrollo de este algoritmo se construye a partir de un polinomio en forma de Bernstein. Ya que este algoritmo es un método recursivo para calcular polinomios de Bernstein.

En el *Anexo 8.1* aparece el desarrollo del polinomio de Casteljau dando como resultado la curva de Bézier.

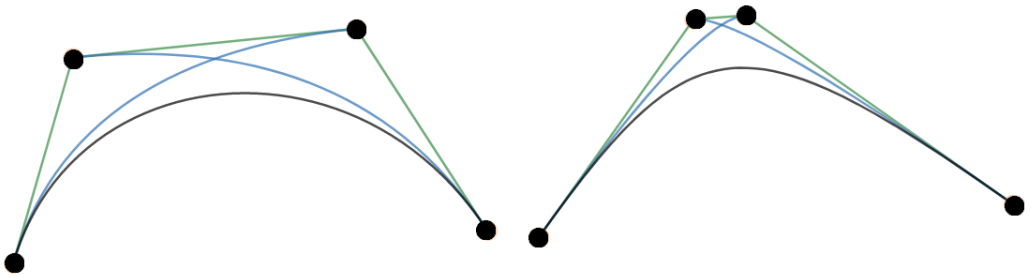


Figura 20. Distancias entre puntos de control. [Fuente propia].

El algoritmo de Casteljau ha resultado de gran utilidad para el diseño por ordenador, pero el problema es que, si se tiene una curva, este algoritmo no determinará qué puntos de control se necesitan para aproximarse lo mejor posible a la curva de Bézier.

Una vez entendido el funcionamiento de las curvas de Bézier, se utilizará el método de Bézier, ya que en este proyecto el propósito es conseguir una interpolación sin el conocimiento de los puntos de control.

Uno de los puntos clave para la generación de la trayectoria del robot se basa en la aproximación a un punto, llamado punto de paso. En este caso concreto, se ha implementado la interpolación lineal de un punto de paso, para hacer la trayectoria de forma suave.

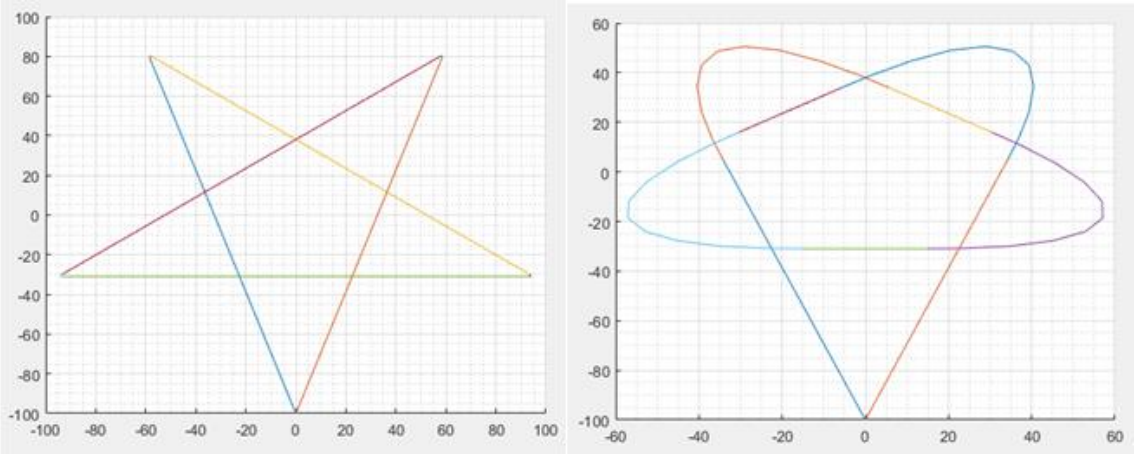


Figura 21. Representación de dos tipos de tolerancia para la interpolación. [Fuente propia].

Se puede hacer una aproximación al punto de paso a partir de una tolerancia definida como δ. Esta aproximación o interpolación lineal se ha hecho a partir de la curva de Bézier.

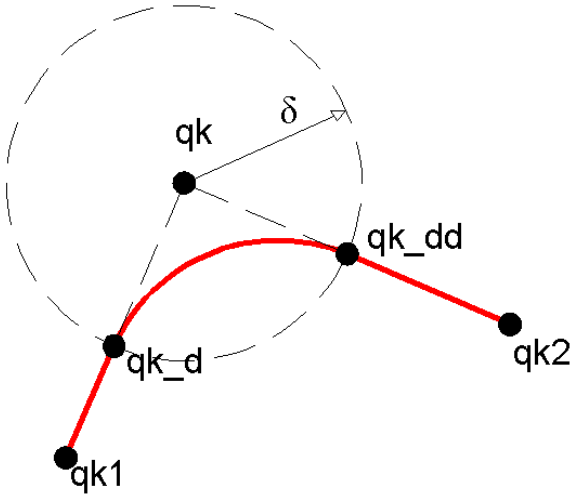


Figura 22. Representación de puntos y tolerancia. [Fuente propia].

La intersección entre la tolerancia δ (en forma de circunferencia) y las rectas que unen el punto anterior (qk1) con el actual (qk), y el actual con el siguiente (qk2), de los cuales estos puntos son obtenidos de las condiciones iniciales, dará lugar a los puntos de control de Bézier (qk_d y qk_dd). En la *Figura 22*. se representan dichos puntos.

Haciendo una comparación de la *Figura 18* y con la *Figura 22* se observa que el primer punto de control (P0) será qk_d y el último punto de control (P4) será qk_dd.

En el algoritmo de este proyecto se han utilizado cinco puntos de control para definir la curva de Bézier, ya que utilizando más puntos de control se tiene un mayor control sobre la forma de la curva. En el *Anexo 8.2* aparece el desarrollo de este método para su programación posterior.

Para encapsular el problema, se ha diseñado una clase en MATLAB llamada bezier_a.m. En esta clase se destacan dos constructores:

- *bezier_a(qk1,qk,qk2,r)*: En esta función solo se le pasa el punto anterior (qk1), el punto actual (qk), el punto siguiente (qk2) y r que es la tolerancia. Estos parámetros de entrada sirven para calcular los puntos de control y, a su vez, la curva de Bézier.

- *bezier_a(qk1,qk,qk2,r,vmax,amax,jmax):* Además de las características de la anterior función, esta tiene de condiciones no superar una velocidad (vmax), aceleración (amax) y *jerk* (jmax) máximo.

4.2.2 Cinemática inversa

Una vez se ha obtenido los puntos cartesianos de cada trayectoria, se necesita pasarlos a coordenadas articulares. Este procedimiento se llama cinemática inversa y se puede obtener con métodos geométricos o a partir de la matriz de transformación homogénea.

La matriz de transformación homogénea utiliza el algoritmo de Denavit-Hartenberg, cuyos cálculos se ven expuestos en el *Anexo 8.3*. Este método de resolución

no se ha implementado finalmente ya que daba errores en la solución de las articulaciones, debido al hecho que trabaja con métodos numéricos.

Finalmente, se ha implementado las tres primeras articulaciones que describen la posición del robot, mediante métodos geométricos. Estos cálculos se detallan en el *Anexo 8.4*.

Para ello se ha programado una clase llamada *CinematicaInversa(mx,my,mz)* donde se pasa la posición de cada punto cartesiano y devuelve las coordenadas articulares.

En este proyecto se ha calculado la posición del brazo robot, sin embargo, la orientación no. Esto es debido a que la orientación viene determinada mediante la articulación cuatro y cinco; y sería actuar directamente sobre estas articulaciones sin necesidad de realizar cálculos para obtener esa orientación deseada.

4.2.3 Control de pasos

Después de obtener las coordenadas articulares, es necesario pasar estos grados de cada articulación a pasos, siguiendo la relación que cada paso son 0.1125 grados.

Como los pasos tienen que ser enteros, y en muchos casos dan números decimales, se redondean al valor entero más próximo. El signo del paso, determinará si los motores de cada articulación van hacia un sentido o hacia el otro.

A continuación, se hace una relación entre el tiempo total que le cuesta alcanzar cada punto y el máximo valor de pasos para en un punto determinado. Con eso se puede controlar la velocidad para alcanzar un punto.

4.2.4 Envío de datos TCP/IP

Finalmente, estos pasos y el tiempo, se convierta a cadena de caracteres. El propio MATLAB crea un servidor donde el ESP32 se conecta, y así le envía el paquete de cadena de caracteres, compuesto por todos los pasos de las articulaciones.

4.3 ESP32

Para poder actuar sobre los motores y para así lograr que se mueva a tiempo real (RT), se ha descompuesto el algoritmo del ESP32 en dos partes. Ya que el propio ESP32 dispone de dos núcleos o *cores*. El segundo núcleo o CPU 1 ha sido utilizado para la conexión TCP/IP y poder recibir el paquete de cadena de caracteres proveniente de MATLAB y poder interpretarlo. Se ha utilizado una lwIP API para conectar el ESP32 como cliente [2].

Mientras que el primer núcleo o CPU 0 es el encargado de coordinar todas las articulaciones y poder controlar en cuanto a pasos y a tiempo, para garantizar su adecuada posición y velocidad de cada uno de los distintos puntos que conforman la trayectoria.

Para poder pasar los pasos de la CPU 1 a la CPU 0, se ha utilizado una cola. Al principio, se barajó la posibilidad de utilizar un acceso directo a memoria (DMA), dónde

la CPU 1 dejaba los pasos de los motores en unas direcciones de memoria, y la CPU 0 accedía a estas memorias, pero finalmente se descartó, ya que eso suponía un cierto retardo a causa de los permisos de lectura y escritura de ambas CPUs.

Las colas normalmente se utilizan en sistemas FIFO (First In First Out), donde los datos se escriben en el final de la cola y se van eliminando en la parte frontal.

Si se observa la *Figura 23*, primeramente, se crea una cola que permite la comunicación entre la CPU 0 y la CPU 1 (A). A continuación, la CPU 0 le envía la variable a=2 al final de la cola (B). Como la cola al principio está vacía solo aparecerá la variable 2. Seguidamente se cambia el valor de la variable local de la CPU 0 con a=55 y finalmente se envía a la cola (C). El primer valor escrito permanece en el frente de la cola mientras que el nuevo valor se inserta al final de la cola (D). El *core* uno lee el primer valor que el *core* cero escribió en la cola. La CPU 1 elimina el elemento que ha leído, dejando solo el segundo valor que queda en la cola. Este valor lo recibirá el *core* uno si se hace una lectura de la cola posteriormente.

La utilización de este sistema de colas se ha implementado mediante la API FreeRTOS dónde están incluidas dentro del ESP32. Esta información ha sido utilizada a partir de las siguientes fuentes [3] [4].

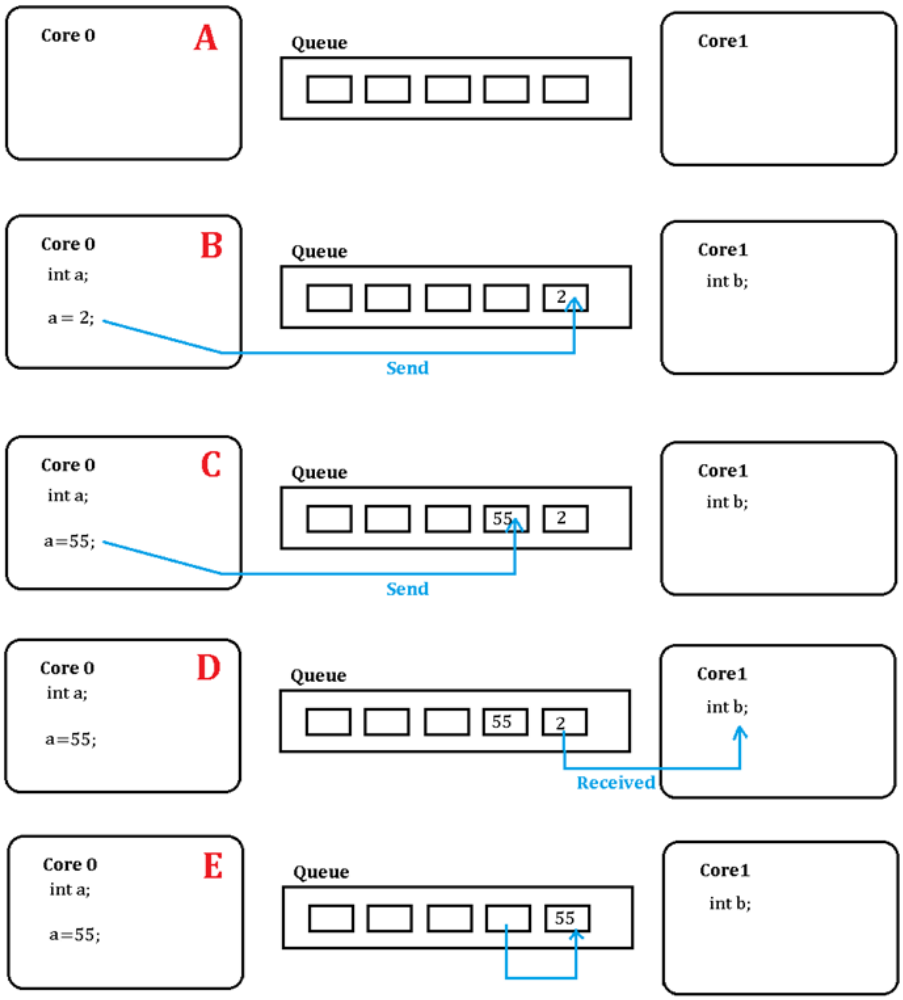


Figura 23. Ejemplo de cola. [Fuente propia].

Para la programación del ESP32 en este proyecto se ha utilizado la IDE de Arduino. Por defecto, Arduino, ejecuta el *setup* y el *loop* en la CPU 1, por lo que se necesita crear una tarea que se ejecute en la CPU0. Esta tarea se crea de acuerdo a la API FreeRTOS donde se obtenido información a partir de las siguientes fuentes [3] [4].

También en el *setup* se conecta la CPU 1 con el WIFI. Donde se requiere de una serie de procedimientos para garantizar esta conexión.

Se necesita inicializar un *event handler* (o controlador de eventos) para que el *core* correspondiente pueda comunicarse con el *WIFI driver*. El WIFI *driver* se comunica con el código de usuario a través de eventos y devoluciones de llamada. En la *Figura 24* se muestra de forma gráfica el procedimiento de inicialización del WIFI.

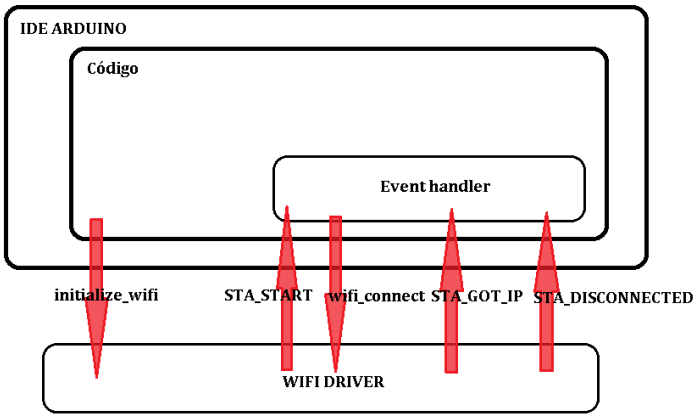


Figura 24. Procedimiento WIFI ESP32. [Fuente propia].

En la CPU 1 se conecta al servidor de MATLAB y recibe la cadena de caracteres que MATLAB le envía. Esta cadena de caracteres se transforma a valor entero, donde se almacena cada una de ellas en una clase llamada POINTS. Posteriormente, se añade a la cola para que la CPU 0 pueda obtener cada uno de los pasos.

Para poder hacer de forma sincronizada cada una de las articulaciones del robot se ha utilizado el algoritmo de Bresenham [5]. Este algoritmo se ha utilizado generalmente para el trazado de líneas en pantallas gráficas. Si se tiene dos puntos A (x0, y0) y B (x1, y1) y se necesita obtener todos los puntos intermedios para dibujar el segmento AB en una pantalla en pixeles, se recurrirá a este método, teniendo en cuenta que los pixeles de la pantalla son enteros. De igual manera, como se tiene un numero de pasos enteros para que actúen sobre dos motores, es necesario actúen de forma coordinada entre ambos, en el caso de un sistema dos dimensiones. En la *Figura 25* se muestra los pasos que debe hacer cada motor de forma coordinada para pasar del punto A al punto B. Cada incremento o decremento será un paso que uno de los actuadores deberá hacer. Este paso o *step* descrito en los apartados anteriores equivaldrá a 0.1125 grados.

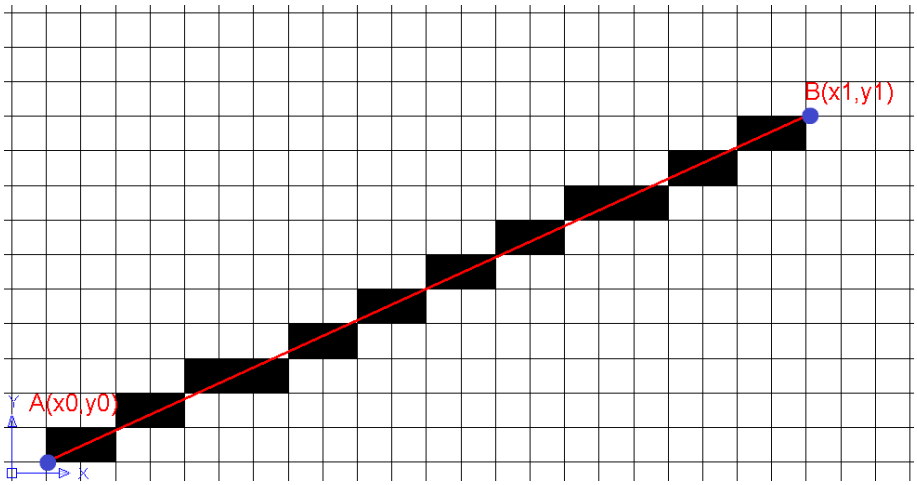


Figura 25. Algoritmo de Bresenham de dos dimensiones [Fuente propia].

Como vienen dados un número de *steps* para cada articulación, para lograr de ir de un punto a otro, se generará algoritmo de Bresenham de cinco dimensiones. El valor

del punto inicial (A) será cero mientras que el punto final (B) será el número de *steps* para cada articulación. En la siguiente tabla se muestra un ejemplo donde para pasar de un punto cartesiano a otro se requiere una serie de coordenadas articulares en forma de pasos para poder actuar sobre los motores.

Tabla 5. Ejemplo resultado de algoritmo de Bresenham de cinco dimensiones.

Puntos	X	Y	Z	W	U
Punto A	0	0	0	0	0
	0	1	0	1	0
	1	2	1	2	1
	1	3	1	2	1
	2	4	2	3	2
	2	5	2	4	2
	3	6	3	5	2
	3	7	3	6	3
	4	8	4	6	3
	4	9	4	7	4
Punto B	5	10	5	8	4

Se ha implementado una interrupción de tiempo en la CPU 0, donde se controla la velocidad en función del tiempo y del algoritmo de Bresenham, para así poder obtener un perfil S-Curve. Cada *step* irá asociado a un tiempo determinado y del algoritmo comentado.

5. Conclusiones

En este proyecto se ha verificado mediante simulación en MATLAB tanto la creación de trayectorias como el funcionamiento de las mismas.

Por lo que hace referencia al propio robot, se ha podido observar y comparar las trayectorias con las gráficas obtenidas con MATLAB. La no utilización de *encoders* dificulta la validación de la prueba experimental, ya que se desconoce la posición inicial donde el robot se encuentra.

Como posibles mejoras, se podría utilizar *encoders* para poder corregir las posibles inercias experimentadas por el brazo del robot, las cuales producen una afectación a su posición.

6. Glosario

- RAE Real Academia Española
- CEA Comité Español de Automática
- PLA Poliácido láctico
- SoC System On a Chip
- TCP/IP Transmission Control Protocol/Internet Protocol
- WIFI Wirelesdd Fidelity
- E/S Entradas/Salidas
- GPIO General Purpose Input/Output
- TSMC Taiwan Semiconductor Manufacturing Company
- PAP Motor paso a paso
- EN Enable
- CLK Pulso
- CW Dirección motor
- SW Switch o interruptor
- IEEE Institute of Electrical and Electronics Engineers
- AP Access Point
- BSS Conjunto de Servicios Básicos
- MAC Media Access Control
- SSID Service Set IDentifier
- ARPANET Advanced Research Projects Agency Network
- NOS Network Operating System
- RT Real Time
- CPU Central Processing Unit
- DMA Direct Memory Access
- FIFO First In First Out
- API Application Programming Interface

7. Bibliografía

[1] «BCN3D,» [En línea]. Available: https://www.bcn3dtechnologies.com/en/bcn3d-moveo-the-future-of-learning/.

[2] «lwIP,» [En línea]. Available: http://www.nongnu.org/lwip/2_0_x/raw_api.html.

[3] «Espressif FreeRTOS,» [En línea]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/system/freertos.html#.

[4] «FreeRTOS,» [En línea]. Available: https://www.freertos.org/.

[5] A. Zingl, A Rasterizing Algorithm for Drawing Curves, 2012.

[6] L. Biagiotti y C. Melchiorri, Trajectory Planning for Automatic Machines and Robots, Springer, 2008.

[7] A. Barrientos, Fundamentos de robótica, McGrawHill, 1997.

[8] L. Piegl y W. Tiller, The NURBS Book, Springer, 1997.

8. Anexos

8.1 Desarrollo algoritmo de Casteljau

En este apartado, de acuerdo con el polinomio base de Bernstein, se hará una recursión de cuatro puntos de control para representar una curva de Bézier. Se ha descrito este método de resolución para hacer más entendible la generación de la curva de Bézier y al mismo tiempo se han representado de forma gráfica los resultados matemáticos, ya que ayudan a su comprensión.

Primeramente, se hace una combinación convexa entre cada uno de los puntos de control. Por lo tanto, se calculará el valor del polinomio de Bernstein de grado tres a partir de los coeficientes, en este caso serán los puntos de control.

$$\beta_0^0 = P_0$$

$$\beta_1^0 = P_1$$

$$\beta_2^0 = P_2$$

$$\beta_3^0 = P_3$$

$$t \in [0,1]$$

Como se sabe que los puntos de control son puntos cartesianos que pertenecen en un plano se pueden substituir por sus coordenadas cartesianas.

$$\beta_0^0(t) = (x_0, y_0)$$

$$\beta_1^0(t) = (x_1, y_1)$$

$$\beta_2^0(t) = (x_2, y_2)$$

$$\beta_3^0(t) = (x_3, y_3)$$

A continuación, se realiza una combinación convexa para poder obtener los tres segmentos detallados en la *Figura 26*.

$$\beta_0^1(t) = \beta_0^0 \cdot (1-t) + \beta_1^0 \cdot t = P_0 \cdot (1-t) + P_1 \cdot t$$

$$\beta_1^1(t) = \beta_1^0 \cdot (1-t) + \beta_2^0 \cdot t = P_1 \cdot (1-t) + P_2 \cdot t$$

$$\beta_2^1(t) = \beta_2^0 \cdot (1-t) + \beta_3^0 \cdot t = P_2 \cdot (1-t) + P_3 \cdot t$$

$$(Ec. 1)$$

Seguidamente, se hará una combinación convexa de las líneas anteriores, en este caso se conseguirá aumentar el grado del polinomio.

$$\beta_0^2(t) = \beta_0^1 \cdot (1-t) + \beta_1^1 \cdot t =$$
$$= (P_0 \cdot (1-t) + P_1 \cdot t) \cdot (1-t) + (P_1 \cdot (1-t) + P_2 \cdot t) \cdot t =$$
$$= P_0 \cdot (1-t)^2 + 2 \cdot P_1 \cdot t \cdot (1-t) + P_2 \cdot t^2$$

$$\beta_1^2(t) = \beta_1^1 \cdot (1-t) + \beta_2^1 \cdot t =$$
$$= (P_1 \cdot (1-t) + P_2 \cdot t) \cdot (1-t) + (P_2 \cdot (1-t) + P_3 \cdot t) \cdot t =$$
$$= P_1 \cdot (1-t)^2 + 2 \cdot P_2 \cdot t \cdot (1-t) + P_3 \cdot t^2$$

$(Ec. 2)$

Finalmente, se obtiene la curva deseada mediante la combinación de las otras β_0^2 y β_1^2.

$$\beta_0^3(t) = \beta_0^2 \cdot (1-t) + \beta_1^1 \cdot t =$$
$$= (P_0 \cdot (1-t)^2 + 2 \cdot P_1 \cdot t \cdot (1-t) + P_2 \cdot t^2) \cdot (1-t)$$
$$+ (P_1 \cdot (1-t)^2 + 2 \cdot P_2 \cdot t \cdot (1-t) + P_3 \cdot t^2) \cdot t$$
$$= P_0 \cdot (1-t)^3 + 3 \cdot P_1 \cdot t \cdot (1-t)^2 + 3 \cdot P_2 \cdot t^2 \cdot (1-t) + P_3 \cdot t^3$$

$(Ec. 3)$

Una vez desarrollado el algoritmo de Casteljau, se puede visualizar la formación de esta curva en la *Figura 26*.

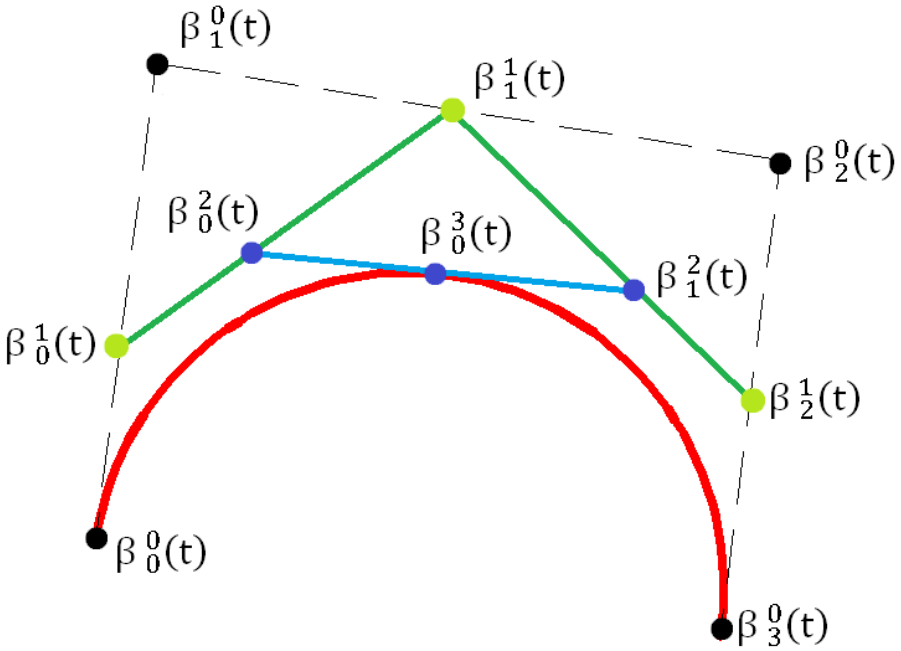


Figura 26. Ilustración algoritmo de Casteljau. [Fuente propia].

8.2 Desarrollo método de Bézier

En este apartado se detallará el método de Bézier para encontrar los puntos de control y, de esta forma, llegar a representar la curva de forma correcta. En la ecuación (4) se detalla una curva de Bézier con grado n.

$$C(u) = \sum_{i=0}^{n} B_{i,n}(u) \cdot P_i \qquad u \in [0,1]$$

$$B_{i,n}(u) = \frac{n!}{i! \cdot (n-i)!} \cdot u^n \cdot (1-u)^{n-i}$$

$$(Ec.4)$$

Donde P_i son los puntos de control de la curva y $B_{i,n}$ son los polinomios de Bernstein.

En este proyecto se ha diseñado una curva de cinco grados y con 5 puntos de control. Si se desarrolla la expresión (4), se puede obtener el polinomio que describe la curva de Bézier, tal como se muestra en la ecuación (5).

$$C(u) = (\mathrm{p0} \cdot (1-\mathrm{u})^5) + (5 \cdot \mathrm{u} \cdot \mathrm{p1} \cdot (1-\mathrm{u})^4) + (10 \cdot \mathrm{p2} \cdot (\mathrm{u}^2) \cdot (1-\mathrm{u})^3) + (10 \cdot \mathrm{p3} \cdot (\mathrm{u}^3) \cdot (1-\mathrm{u})^2) + (5 \cdot \mathrm{p4} \cdot (\mathrm{u}^4) \cdot (1-\mathrm{u})) + (\mathrm{p5} \cdot (\mathrm{u}^5) \cdot (1-\mathrm{u})^0$$

$$C(u) = (5 \cdot p1 - p0 - 10 \cdot p2 + 10 \cdot p3 - 5 \cdot p4 + p5) \cdot u^5 + (5 \cdot p0 - 20 \cdot p1 + 30 \cdot p2 - 20 \cdot p3 + 5 \cdot p4) \cdot u^4 + (30 \cdot p1 - 10 \cdot p0 - 30 \cdot p2 + 10 \cdot p3) \cdot u^3 + (10 \cdot p0 - 20 \cdot p1 + 10 \cdot p2) \cdot u^2 + (5 \cdot p1 - 5 \cdot p0) \cdot u + p0$$

$$(Ec.5)$$

Los puntos de control p0 y p5 están dados, mientras que los otros puntos deben ser determinados mediante la imposición de las tangentes preinscritas en dicho polinomio. En el primer punto de control, p0, la tangente tendrá el mismo valor que en el punto medio de la curva C'(1/2) y que en el punto de control final, p5. En este caso su valor queda definido por: ∝.

$$\begin{aligned} C'(\mathrm{u}) = {} & (-5 \cdot \mathrm{p0} + 25 \cdot \mathrm{p1} - 50 \cdot \mathrm{p2} + 50 \cdot \mathrm{p3} - 25 \cdot \mathrm{p4} + 5 \cdot \mathrm{p5}) \cdot \mathrm{u}^4 \\ & + (20 \cdot \mathrm{p0} - 80 \cdot \mathrm{p1} + 120 \cdot \mathrm{p2} - 80 \cdot \mathrm{p3} + 20 \cdot \mathrm{p4}) \cdot \mathrm{u}^3 \\ & + (-30 \cdot \mathrm{p0} + 90 \cdot \mathrm{p1} - 90 \cdot \mathrm{p2} + 30 \cdot \mathrm{p3}) \cdot \mathrm{u}^2 \\ & + (20 \cdot \mathrm{p0} - 40 \cdot \mathrm{p1} + 20 \cdot \mathrm{p2}) \cdot \mathrm{u} - 5 \cdot \mathrm{p0} + 5 \cdot \mathrm{p1} \end{aligned}$$

$$(Ec.6)$$

Para poder encontrar los otros puntos de control en función de los ya dados, hace falta calcular la segunda derivada del polinomio, tal y como se muestra en la ecuación (7).

$$C''(\mathrm{u}) = (-20\cdot \mathrm{p0} + 100\cdot \mathrm{p1} - 200\cdot \mathrm{p2} + 200\cdot \mathrm{p3} - 100\cdot \mathrm{p4} + 20\cdot \mathrm{p5})\cdot \mathrm{u}^3 + (60\cdot \mathrm{p0} - 240\cdot \mathrm{p1} + 360\cdot \mathrm{p2} - 240\cdot \mathrm{p3} + 60\cdot \mathrm{p4})\cdot \mathrm{u}^2 + (-60\cdot \mathrm{p0} + 180\cdot \mathrm{p1} - 180\cdot \mathrm{p2} + 60\cdot \mathrm{p3})\cdot \mathrm{u} + 20\cdot \mathrm{p0} - 40\cdot \mathrm{p1} + 20\cdot \mathrm{p2}$$

$(Ec. 7)$

$$C(0) = qk\_d$$

$$C(1) = qk\_dd$$

$$C'(0) = 5\cdot p1 - 5\cdot p0 = \propto \cdot T_0$$

$$p1 = p0 + \propto \cdot \frac{T_0}{5}$$

$$C'(1) = 5\cdot p5 - 5\cdot p4 = \propto \cdot T_1$$

$$p4 = p5 - \propto \cdot \frac{T_1}{5}$$

$$C''(0) = 20\cdot p0 - 40\cdot p1 + 20\cdot p2 = 0$$

$$p2 = 2\cdot p1 - p0 = p0 + 2\cdot \propto \cdot \frac{T_0}{5}$$

$$C''(1) = 20\cdot p3 - 40\cdot p4 + 20\cdot p5 = 0$$

$$p3 = 2\cdot p4 - p5 = p5 - 2\cdot \propto \cdot \frac{T_1}{5}$$

$(Ec. 8)$

Como la magnitud de la tangente es la misma que en los extremos y en el medio de la curva, su valor se podrá detallar mediante la resolución de una ecuación de orden dos. Ver ecuación (9).

$$C'\left(\frac{1}{2}\right) = \frac{5\cdot p3}{8} - \frac{15\cdot p1}{16} - \frac{5\cdot p2}{8} - \frac{5\cdot p0}{16} + \frac{15\cdot p4}{16} + \frac{5\cdot p5}{16} = \alpha$$

$$256\cdot \alpha^2 = (49\cdot t0^2 + 98\cdot t0\cdot t1 + 49\cdot t1^2)\cdot \alpha^2 + (420\cdot p5\cdot t0 - 420\cdot p0\cdot t1 - 420\cdot p0\cdot t0 + 420\cdot p5\cdot t1)\cdot \alpha + 900\cdot p0^2 - 1800\cdot p0\cdot p5 + 900\cdot p5^2$$

$$(256 - 49\cdot (T_1 + T_0)^2)\cdot \alpha^2 + \big(420\cdot (T_1 + T_0)\cdot (p5 - p0)\big)\alpha - 900\cdot (p5 - p0)^2 = = 0$$

$(Ec. 9)$

A continuación, se hace una reparametrización de la curva de Bézier para pasar el polinomio en función $u \in [0,1]$ a $u \in [0,\lambda]$ donde $\lambda = 5 \cdot |p_1 - p_0|$, como se plantea por Biagiotti, Luigi y Melchiorri, Claudio [6].

Para no sobrepasar unas condiciones límite en cuanto a velocidad, aceleración y *jerk* máximo, se debe implementar un escalado. Esta constante de escalado va a recibir la nomenclatura de $\dot{\lambda}$. Por lo tanto:

$$C'(t) = C'(u) \cdot \dot{\lambda}$$

$$C''(t) = C''(u) \cdot \dot{\lambda}$$

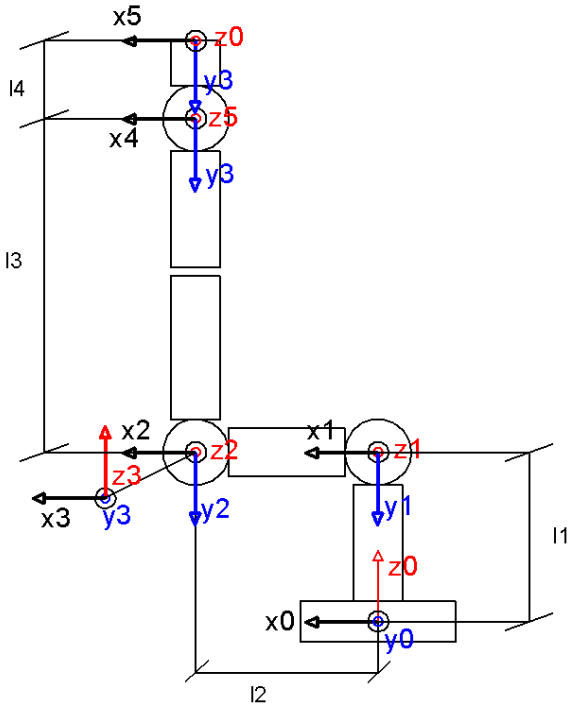
$$C'''(t) = C'''(u) \cdot \dot{\lambda}$$

Con el propósito de cumplir estas condiciones, se necesita calcular la relación entre las condiciones máximas y el máximo del polinomio para cada condición. Para ello, se desarrolla según el planteamiento presente en el libro de Biagiotti, Luigi y Melchiorri, Claudio [6].

$$\dot{\lambda} = \min\left(\frac{vmax}{C'(u)_{max}}, \sqrt{\frac{amax}{C''(u)_{max}}}, \sqrt[3]{\frac{jmax}{C'''(u)_{max}}}\right)$$

8.3 Desarrollo de matrices homogéneas

Para el cálculo de cinemática inversa, primeramente, se necesitará un modelo simplificado de nuestro robot para definir los ejes y ángulos de articulación que correspondan a su eje. Este modelo simplificado se muestra en la *Figura 27*.



[3]

Figura 27. Modelo simplificado del robot. [Fuente propia].

Con la identificación de los sólidos y ejes que conforman el robot se obtendrá la siguiente tabla con los parámetros de Denavit-Hatenberg correspondientes.

Tabla 6. Tabla de Denavit-Hatenberg.

Articulación	θ	d	a	α
1	θ_1	l_1	0	-90
2	θ_2	0	l_2	0
3	θ_3	0	0	90
4	θ_4	l_3	0	-90
5	θ_5	0	0	0

Se obtienen a partir de la tabla, las matrices de transformación

$$T01=\begin{pmatrix} C1 & 0 & -S1 & 0 \\ S1 & 0 & C1 & 0 \\ 0 & -1 & 0 & l1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T12=\begin{pmatrix} C2 & -S2 & 0 & l2{\cdot}C2 \\ S2 & C2 & 0 & l2{\cdot}S2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T23=\begin{pmatrix} C3 & 0 & S3 & 0 \\ S3 & 0 & -C3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T34=\begin{pmatrix} C4 & 0 & -S4 & 0 \\ S4 & 0 & C4 & 0 \\ 0 & -1 & 0 & l3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T45=\begin{pmatrix} C5 & -S5 & 0 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Según la solución de Pieper la posición de un punto queda determinado por las tres primeras articulaciones, si $a_4=a_5=d_5=0$, para ello se construye una matriz homogénea que define el brazo. En esta matriz aparecerá la posición (m) y la rotación (R03) del brazo.

$$T03 = T01 \cdot T12 \cdot T23 = \begin{pmatrix} R03 & m \\ 0 & 1 \end{pmatrix}$$

$$(Ec.10)$$

Para el cálculo, se determinará la posición de la muñeca donde vendrá definida como: posición muñeca= T03 · (0,0, l3).

$$l_3 \cdot (C1 \cdot C2 \cdot S3 + C1 \cdot S2 \cdot C3) + l_2 \cdot C1 \cdot C2 = mx$$

$$(Ec.11)$$

$$l_3 \cdot (S1 \cdot C2 \cdot S3 + S1 \cdot S2 \cdot C3) + l_2 \cdot S1 \cdot C2 = my$$

$$(Ec.12)$$

$$l_1 + l_3 \cdot (C2 \cdot C3 - S2 \cdot S3) - l_2 \cdot S2 = mz$$

$$(Ec.13)$$

A continuación, se procede al cálculo de la primera articulación mediante la sustitución de las ecuaciones (11) y (12).

$$C1 \cdot (l_3 \cdot C2 \cdot S3 + l_3 \cdot S2 \cdot C3 + l_2 \cdot C2) = mx$$

$$C1 = \frac{mx}{(l_3 \cdot C2 \cdot S3 + l_3 \cdot S2 \cdot C3 + l_2 \cdot C2)}$$

$$S1 \cdot (l_3 \cdot C2 \cdot S3 + l_3 \cdot S2 \cdot C3 + l_2 \cdot C2) = my$$

$$S2 = \frac{my}{(l_3 \cdot C2 \cdot S3 + l_3 \cdot S2 \cdot C3 + l_2 \cdot C2)}$$

$$\theta_1 = \arctan 2(my, mx)$$

Para el cálculo de theta uno, se utiliza el arco tangente dos ya que no se sabe el valor de las incógnitas (se desconocen θ2 y θ3), y este determinará el cuadrante en el que estará la theta uno, por lo que será posible obtener dos tipos de soluciones.

Haciendo uso ecuaciones (11) y (12) y de las razones trigonométricas de suma de ángulos (14) se obtiene la ecuación (15) donde se querrá determinar el valor de theta dos y tres.

$$C1^2 \cdot (l_3 \cdot C2 \cdot S3 + l_3 \cdot S2 \cdot C3 + l_2 \cdot C2) = mx \cdot C1$$

$$S1^2 \cdot (l_3 \cdot C2 \cdot S3 + l_3 \cdot S2 \cdot C3 + l_2 \cdot C2) = my \cdot S1$$

$$(C1^2 + S2^2) \cdot (l_3 \cdot C2 \cdot S3 + l_3 \cdot S2 \cdot C3 + l_2 \cdot C2) = mx \cdot C1 + my \cdot S1$$

$$C23 = C2 \cdot C3 - S2 \cdot S3$$

$$S23 = C2 \cdot S3 + C3 \cdot S2$$

$$(Ec. 14)$$

$$l_3 \cdot (C2 \cdot S3 + S2 \cdot C3) + l_2 \cdot C2 = mx \cdot C1 + my \cdot S1$$

$$l_3 \cdot S23 + l_2 \cdot C2 = mx \cdot C1 + my \cdot S1 = a$$

$$l_3 \cdot S23 + l_2 \cdot C2 = a$$

$$l_2 \cdot C2 = a - l_3 \cdot S23$$

$$l_1 + l_3 \cdot (C2 \cdot C3 - S2 \cdot S3) - l_2 \cdot S2 = mz = b$$

$$l_1 + l_3 \cdot (C2 \cdot C3 - S2 \cdot S3) - l_2 \cdot S2 = b$$

$$l_2 \cdot S2 = l_1 + l_3 \cdot C23 - b$$

$$l_2^2 \cdot C2^2 = (a - l_3 \cdot S23)^2$$

$$l_2^2 \cdot S2^2 = (l_1 + l_3 \cdot C23 - b)^2$$

$$l_2^2 \cdot C2^2 + l_2^2 \cdot S2^2 = (a - l_3 \cdot S23)^2 + (l_1 + l_3 \cdot C23 - b)^2$$

$$C23X + S23Y = Z$$

$$\begin{cases} X = 2 \cdot l_3 \cdot B - 2 \cdot l_1 \cdot l_3 \\ Y = 2 \cdot a \cdot l_3 \\ Z = -l_2^2 + l_3^2 + a^2 + b^2 - 2 \cdot l_1 \cdot b + l_1^2 \end{cases}$$

$$\emptyset = \arctan(Y, X)$$

$$r = \sqrt{X^2 + Y^2}$$

$$\theta_{23} = -\emptyset \pm arcos\left(\frac{Z}{r}\right)$$

$(Ec. 15)$

$$l_2 \cdot S2 = l_1 + l_3 \cdot C23 - b = c$$

$$l_2 \cdot C2 = a - l_3 \cdot S23 = d$$

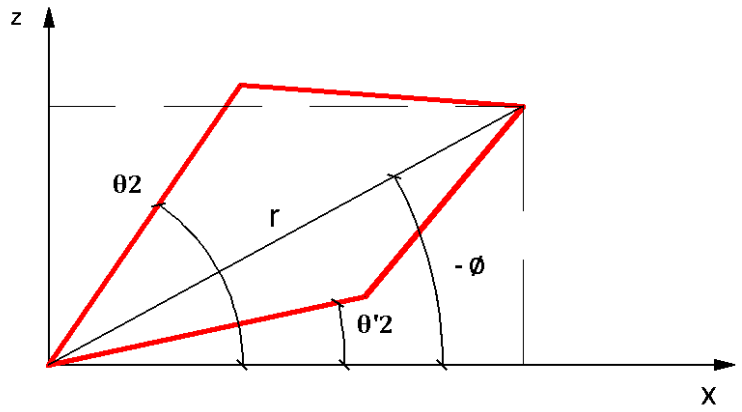


Figura 28. Posibles soluciones segunda articulación. [Fuente propia].

La articulación dos tendrá dos soluciones o "el codo arriba" o "el codo abajo", tal como se muestra en la *Figura 28*.

$$mx \cdot C1 + my \cdot S1 = a$$

$$mz = b$$

$$\theta_2 = arctan2\left(\frac{c}{d}\right)$$

Una vez obtenida la articulación dos se procederá al cálculo de la articulación tres.

$$\theta_3 = \theta_{23} - \theta_2$$

Para la obtención de la theta cuatro y cinco se hará mediante el cálculo de la matriz de rotación R03, según ecuación (16). Los resultados que se necesita obtener muestran reflejado en 17.

$$R05 = R03 \cdot R35 \rightarrow R35 = (R03)^{-1} \cdot R05$$

$$\begin{pmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{pmatrix} = (R03)^{-1} \cdot \begin{pmatrix} nx & ox & ax \\ ny & oy & ay \\ nz & oz & az \end{pmatrix}$$

$$(Ec. 16)$$

$$r33 = az \cdot C23 + ax \cdot S23 \cdot C1 + ay \cdot S23 \cdot S1$$

$$r13 = ax \cdot C23 \cdot C1 - az \cdot S23 + ay \cdot C23 \cdot S1$$

$$r32 = oz \cdot C23 + ox \cdot S23 \cdot C1 + oy \cdot S23 \cdot S1$$

$$r31 = nz \cdot C23 + nx \cdot S23 \cdot C1 + ny \cdot S23 \cdot S1$$

$$(Ec. 17)$$

También se hace uso de la matriz de rotación R35 obtenida a partir de las matrices de transformación T34 y T45.

$$T35 = T34 \cdot T45 = \begin{pmatrix} R35 & m \\ 0 & 1 \end{pmatrix}$$

$$(Ec. 18)$$

Si se despejan las ecuaciones (17) y (18) se obtendrá la articulación cuatro y lo mismo pasará con la articulación cinco.

$$r33 = C4$$

$$r33 = az \cdot C23 + ax \cdot S23 \cdot C1 + ay \cdot S23 \cdot S1$$

$$\theta_4 = \pm \arccos(r33)$$

$$C5 \cdot S4 = r31$$

$$S4 \cdot S5 = -r32$$

$$\theta_5 = \arctan\left(-\frac{r32}{r31}\right)$$

8.4 Desarrollo métodos geométricos

En este apartado se calcula la cinemática inversa a partir de métodos geométricos. Utilizando el teorema de Pitágoras y la ley del coseno se puede obtener theta dos y theta tres.

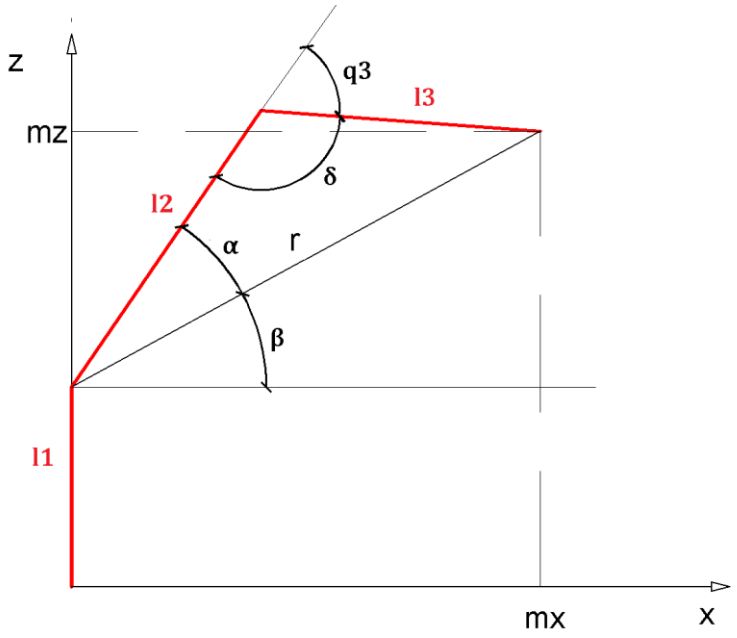


Figura 29. Planteamiento mediante métodos geométricos. [Fuente propia].

$$r = mx^2 + my^2 + (mz - l_1)^2$$

$$r = l_2^2 + l_3^2 - 2 \cdot l_2 \cdot l_3 \cdot \cos(\delta) \rightarrow r = l_2^2 + l_3^2 + 2 \cdot l_2 \cdot l_3 \cdot \cos(q3)$$

$$\cos(\theta 3) = \frac{x^2 + my^2 + (mz - l_1)^2}{l_2^2 + l_3^2 + 2 \cdot l_2 \cdot l_3}$$

$$q3 = arctan2\left(\frac{-\sqrt{1 - \cos(\theta 3)^2}}{\cos(\theta 3)}\right)$$

$$\propto = \text{atan}\left(\frac{l_3 \cdot sen(q3)}{l_2 + l_3 \cdot cos(q3)}\right)$$

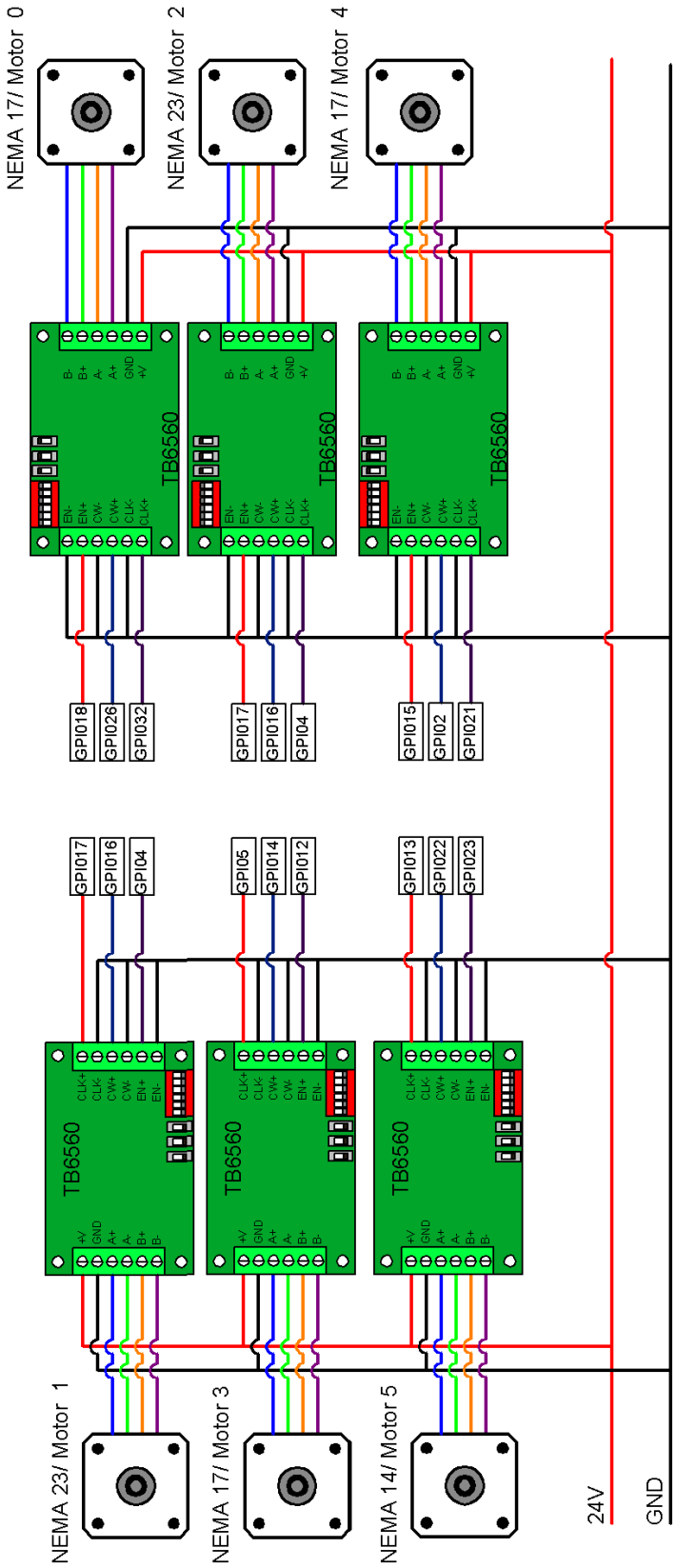
$$\beta = -\frac{mz - l_1}{\sqrt{mx^2 + my^2}}$$

$$\theta 2 = \propto + \beta$$

Para la obtención de theta uno se calculará a partir del arco tangente dos.

$$\theta 1 = arctan2\left(\frac{my}{mx}\right)$$

8.5 Esquema eléctrico



8.6 Presupuesto

Descripción	Cantidad	Precio unitario	Total
Caja de conectores	1,0	7,9	7,9
Ventilador monofásico 230V	1,0	20,0	20,0
Acoplamiento flexible D19/L25/5mm*8mm	1,0	1,5	1,5
Nema 23	2,0	34,1	68,2
Caja de empalmes	1,0	32,0	32,0
Chapa 900*350mm	1,0	15,0	15,0
Cojinete 624ZZ	10,0	0,1	0,9
Cojinete 623ZZ	10,0	0,1	1,1
Cojinete 608ZZ	10,0	0,2	2,2
Cojinete 625ZZ	10,0	0,1	1,4
Driver TB6560	6,0	5,2	31,1
Fuente de alimentación 230AC/24DC 20A	1,0	20,2	20,2
Nema 14	1,0	21,4	21,4
Nema 17 5:1	1,0	40,0	40,0
Polea T5 14 dientes 16mm	1,0	30,0	30,0
Polea T5 10 dientes 5mm	2,0	16,2	32,4
Tornillería	1,0	50,0	50,0
Caja de empalmes	1,0	30,0	30,0
Bobina PLA amarillo	3,0	24,8	74,4
Bobina PLA negro	2,0	24,8	49,6
Canaleta	1,0	3,2	3,2
***Protoboard* 165*55mm**	2,0	2,3	4,6
Nema 17	2,0	16,9	33,9
Servomotor 20KG·cm	1,0	19,0	19,0
Chapa	1,0	20,0	20,0
Pequeño material	1,0	18,3	18,3
		Total	620,2