UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y Programación de un Robot Delta basado en Arduino con fines académicos

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: López Tomás, Héctor

Tutor/a: Armesto Ángel, Leopoldo

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Diseño y programación de un Robot Delta basado en Arduino con fines académicos

Trabajo final de grado en Ingeniería Electrónica Industrial y Automática

Documentos:

1- Memoria
2- Documentación Técnica
3- Pliego de condiciones
4- Planos
5- Presupuesto
6- Código fuente

Autor: Héctor López Tomás
Tutor: Leopoldo Armesto Ángel

Curso académico 2021-2022

RESUMEN

Este proyecto consiste en realizar un Robot Delta de tres grados de libertad, de bajo coste, con fines académicos, y que sea cómodo de manipular y sencillo de fabricar. Asimismo, debe tener un tamaño que garantice un mínimo espacio de trabajo sin confeccionarlo a una escala excesivamente grande.

Para ello, se realiza un estudio de todas las alternativas y factores a considerar para establecer finalmente unas propiedades sobre las que trabajar. Habiendo decidido los componentes y características del proyecto, se pasa al planteamiento de diferentes diseños que puedan acoplarse a las mismas, y que garanticen un correcto funcionamiento del robot.

Una vez se ha establecido el diseño del robot, se procede a realizar las simulaciones y cálculos pertinentes para comprobar si cinemáticamente es realizable, y si los actuadores escogidos cumplen con el par necesario para el diseño. También se calcula su volumen de trabajo obtenido.

Finalmente, con todas las simulaciones y los cálculos realizados y corroborados, se implementa de forma física el robot, para analizar su comportamiento dinámico y estudiar la posible aparición de defectos que no se hubieran tenido en consideración, con el objetivo de corregirlos a continuación.

ABSTRACT

This project consists of making a low-cost, three-degree-of-freedom delta robot for academic purposes that is easy to handle and simple to manufacture. Likewise, it must have a size that guarantees a minimum working space without making it on an excessively large scale.

To this end, a study of all the alternatives and factors to be considered is carried out to finally establish the properties on which to work. Having decided on the components and characteristics of the project, the next step is to propose different designs that can be coupled to them, guaranteeing the correct operation of the robot.

Once the robot design has been established, we proceed to perform the relevant simulations and calculations to check if it is kinematically feasible and if the chosen actuators comply with the torque required for the design. Its obtained working volume is also calculated.

Finally, with all the simulations and calculations performed and corroborated, the robot is physically implemented to analyze its dynamic behavior and study the possible appearance of defects that had not been taken into consideration, with the aim of correcting them later.

RESUM

Aquest projecte consisteix a realitzar un Robot Delta de tres graus de llibertat, de baix cost, amb finalitat acadèmica, i que siga còmode de manipular i senzill de fabricar. Així mateix, ha de tindre una grandària que garantisca un mínim espai de treball sense confeccionar-lo a una escala excessivament gran.

Per a això, es realitza un estudi de totes les alternatives i factors a considerar per a establir finalment unes propietats sobre les quals treballar. Havent decidit els components i característiques del projecte, es passa al plantejament de diferents dissenys que puguen acoblar-se a aquestes, i que garantisquen un correcte funcionament del robot.

Una vegada s'ha establit el disseny del robot, es procedeix a realitzar les simulacions i càlculs pertinents per a comprovar si cinemáticament és realitzable, i si els actuadors triats compleixen amb el parell necessari per al disseny. També es calcula el seu volum de treball obtingut.

Finalment, amb totes les simulacions i els càlculs realitzats i corroborats, s'implementa de manera física el robot, per a analitzar el seu comportament dinàmic i estudiar la possible aparició de defectes que no s'hagueren tingut en consideració, amb l'objectiu de corregir-los a continuació.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Documento número 1:

Memoria técnica

Diseño y programación de un Robot Delta basado en Arduino con fines académicos

Trabajo final de grado en Ingeniería Electrónica Industrial y Automática

Autor: Héctor López Tomás

Tutor: Leopoldo Armesto Ángel

Curso académico 2021-2022

ÍNDICE

1 OBJETIVO Y MOTIVACIÓN DEL TFG

1.1 Justificación del TFG

Actualmente, la existencia en el mercado de productos académicos dentro del sector de sistemas robotizados es limitada y tienen una difícil accesibilidad.

La mejora, tanto del aprendizaje como de la enseñanza de este tipo de sistemas, en concreto, el de los robots de tipo paralelo, es la razón principal que justifica el objeto de este proyecto.

El diseño final pretende ofrecer un producto de bajo coste que permite ser utilizado con fines académicos, además de incluir la posibilidad de implementar mejoras posteriores. Así pues, tiene un valor práctico, académico y funcional.

La consecución de un resultado que combina a la vez la utilidad en las aulas y un reto técnico, ha supuesto una motivación añadida.

1.2 Objetivo del TFG

El objetivo del presente Trabajo de Fin de Grado (en adelante TFG), tiene carácter tanto académico como práctico.

El propósito de este proyecto es realizar un Robot Delta como producto académico, para centrarse en su aprendizaje, cuya principal misión es favorecer los procesos formativos de los estudiantes, a través del diseño de recursos pedagógicos que tienen directo impacto en sus procesos formativos y en sus prácticas docentes.

El robot paralelo tiene las propiedades y características de un Robot Delta. No obstante, algunas de sus propiedades deberán ser modificadas para cumplir con la finalidad en cuestión.

Este proyecto abarca el proceso completo de la elaboración del producto. Se desarrollan sus conceptos teóricos, indagando en los cálculos necesarios elaborados y en su programación para su correcto funcionamiento.

1.3 Destinatario del proyecto

El destinatario final de este producto es cualquier público que esté dispuesto a aprender sobre sistemas robotizados, sin embargo, los usuarios deberán poseer unos conocimientos básicos, entre los que destaquen: trigonometría, matemática avanzada y lenguaje de programación básico. Por tanto, el producto está orientado principalmente a estudiantes de cualquier curso universitario, o a aquellos que se encuentren cursando estudios tanto de grado medio como de grado superior o máster, dentro del ámbito de la electrónica.

Se ofrecerá el producto a instituciones educativas para su empleo dentro de la enseñanza, y también podrán acceder al mismo de forma particular, quienes tengan un interés personal.

2 INTRODUCCIÓN

2.1 Antecedentes

2.1.1 Automatización industrial

La automatización industrial es el fenómeno mediante el que las máquinas, la informática y la tecnología son utilizadas con fines industriales. Mediante la automatización, los procesos que, a priori, eran procesos manuales, pasan a ser procesos automatizados, lo que dota de gran autonomía a los procesos industriales. Para algunos pensadores, la automatización ha sido, ya no solo el origen de la Tercera Revolución Industrial, sino componente fundamental para el desarrollo de lo que denominan Cuarta Revolución Industrial, donde se dota a la automatización de inteligencia artificial e interconexión total para la completa autonomía de los procesos industriales.

La automatización industrial forma parte de la Tercera Revolución Industrial, al mecanizarse muchos procesos y dando lugar a una nueva industria a la que muchos autores denominan “industria 4.0”.

La automatización, aunque no en el sentido que se conoce habitualmente, ha existido a lo largo de la historia. Ya en la época griega o romana, los ciudadanos trataron de implementar la automatización en la sociedad, con objetos mecánicos que trataban de imitar determinados movimientos de seres animados. También los egipcios tuvieron experiencias con los principios de automatización, pues, al igual que los griegos, utilizaron los procesos mecánicos para la animación de las estatuas que trataban de representar la figura de sus dioses.

La automatización en el principio de línea no se limita solo a un sistema de control, tiene en ella muchos instrumentos industriales, entre los cuales puede haber sensores y transmisores de campo, sistemas de control y supervisión, sistemas de recopilación de datos y transmisión y aplicaciones de software en tiempo real para supervisión y control de operaciones de procesos y plantas industriales.

La automatización, como todo, tiene sus ventajas y sus desventajas. Con el surgimiento de las máquinas automatizadas, la industria experimentó una revolución que desencadenó una nueva metodología.

Entre las ventajas que podemos destacar de la automatización, se encuentran:

- Incremento de la productividad.
- Ahorro de costes
- Menor esfuerzo físico.
- Mayor calidad de vida.
- Mejora de la salud de los trabajadores.
- Mejoras de las condiciones de trabajo.
- Mayor ventaja competitiva.
- En algunos casos, reducción de los riesgos laborales.

Por otro lado, la automatización también tiene sus desventajas. Entre ellas cabría destacar las siguientes:

- Destrucción de empleo.
- Posible disminución de la recaudación fiscal en el impuesto sobre la renta.
- Dependencia tecnológica.
- Obsolescencia tecnológica.
- Incremento en los costes de inversión.
- Incremento en los costes de mantenimiento.
- Dependencia de personal más cualificado.

Las mejoras de la salud y seguridad derivadas de la implantación de sistemas robotizados son entre otros, una disminución de los accidentes laborales, mejora y optimización de las condiciones de trabajo, eliminando riegos laborales asociados a entornos peligrosos y la eliminación de trabajos rutinarios y fatigosos.

2.1.2 Robótica

La robótica es la ciencia y la técnica que está involucrada en el diseño, la fabricación y la utilización de robots. Un robot es, por otra parte, una máquina que puede programarse para que interactúe con objetos y lograr que realice el comportamiento humano o animal. La informática, la electrónica, la mecánica y la ingeniería son solo algunas de las disciplinas que se combinan en la robótica. El objetivo principal de la robótica es la construcción de dispositivos que funcionen de manera automática y que realicen trabajos dificultosos o imposibles para los seres humanos.

En la década de los años 30 del siglo XX (1937 para ser más exactos) el estudiante británico Bill Taylor creó el robot Gargantua, un robot con forma de grúa para el que utilizó componentes que se usaban en los juguetes de Meccano. Este robot se utilizó para la colocación de productos, lo que en lenguaje industrial se conoce como pick & place. Se programó con cinta de papel y motorizado con un único motor eléctrico. Gargantua era capaz de apilar bloques de madera en patrones programados. Por su parte, en el año 1938, la compañía Devilviss construye el primer robot con forma de brazo articulado para pintura en spray. En la feria anual de Nueva York de 1939, ya se presentaron robots para uso industrial y para ocio. Esto supuso el inicio de una de las industrias más interesantes de la historia y de una nueva era.

Dentro de esta nueva era surgirían nuevos tipos de robots industriales en los que variarán características como: su grado de libertad, zona de trabajo, carga a sostener, velocidad de acción o nivel de programación, cuyas aplicaciones específicas más frecuentes son: manipulación, aportación de materiales, ensamblado y desensamblado, empaquetado y paletizado, logística, e inspección y control de calidad.

Dentro de los robots manipuladores se encuentran: robots Antropomórficos, robots Scaras, robots Cartesianos, robots Paralelogramos, robots Scara Dual, robots Redundantes, robots Colaborativos y robots Paralelos. En este último se enfoca este proyecto.

El principal ejemplo del robot Paralelo es el Delta, que fue inventado a principios de los noventa por un equipo de investigación liderado por el profesor Reymond Clavel en la École Polytechnique Fédérale de Lausanne (EPFL, Suiza). El propósito de este robot no era otro que el de manipular objetos pequeños y ligeros a una velocidad muy alta (pick and place).

Diferentes empresas empezaron a sacar al mercado modelos de robot Delta, destacando: robot Delta de KUKA, robot Delta de Fanuc, robot Delta de Omron, robot Delta de ABB, robot Delta de Yaskawa, robots Delta Maxon para microcomponentes… Este tipo de robots industriales destacan por abarcar una amplia superficie de trabajo y por ofrecer una gran velocidad y aceleración. Ofrecen la seguridad de realizar trabajos repetitivos y posicionados con alta precisión. Integran tecnologías avanzadas como la Visión Artificial y un software controlado por Deep Learning. Los fabricantes han desarrollado softwares que permiten configuraciones para que puedan interaccionar hasta 8 robots en una misma línea de producción. La gran mayoría de ellos dispone de una categoría de protección IP65 e IP67 que les permite trabajar en el sector alimentario.

Imagen 1. Robot Delta MPP3H YASKAWA

La motivación para realizar este proyecto viene de la sugerencia de mi tutor de TFG, que es el mismo que impartía la asignatura de Sistemas Robotizados. En esta asignatura, había que realizar un proyecto que consistía en la elaboración y programación de un brazo robótico de 3 grados de libertad, más la herramienta, cuyos costes de fabricación y montaje debían ser menores de 80 € (*Imagen 2*).

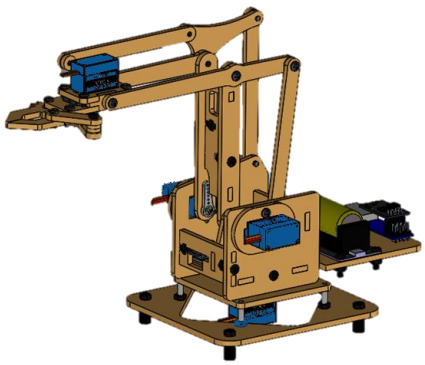


Imagen 2. Brazo Robot meArm (versión UPV) con ESP-01S y SG90

Partiendo de esta base, cuando mi tutor supo que quería realizar un brazo robótico como proyecto para mi TFG, me propuso la idea de realizar un Robot Delta de bajo coste, que también pudiera ser utilizado como proyecto de la asignatura y ofrecido al alumnado como ejemplo práctico de robots manipuladores, fuera de los ya comercializados en tiendas de electrónica.

2.2 Programas utilizados

Para la realización de este proyecto ha sido necesario emplear diversos programas informáticos para la correcta resolución de los objetivos que se citarán posteriormente. A continuación, se detallan los diferentes softwares exponiendo sus características y su utilidad dentro del proyecto. Cabe destacar que los programas Matlab y Arduino IDE, han sido utilizados durante el curso universitario, mientras que SolidWorks, CoppeliaSim y Meshmixer se han manejado de forma autodidacta.

2.2.1 SolidWorks

SolidWorks es un software de diseño CAD 3D (diseño asistido por computadora), para modelar piezas y ensamblajes en 3D y planos en 2D. Este programa ofrece un abanico de soluciones para cubrir los aspectos implicados en el proceso de desarrollo del producto, dando la posibilidad de crear, diseñar, simular, fabricar, publicar y gestionar los datos del proceso de diseño.

La labor de SolidWorks en el proceso de desarrollo del producto es muy específica, las soluciones ayudan a acelerar el proceso ahorrando tiempo y dinero dando paso a la innovación de los productos.

SolidWorks ofrece soluciones intuitivas para cada fase de diseño. Cuenta con un completo conjunto de herramientas que le ayudan a ser más eficaz y productivo en el desarrollo de sus productos en todos los pasos del proceso de diseño. La sencillez, que es parte de su propuesta de valor, incluye cinco líneas de productos diferentes:

- Herramientas de diseño para crear modelos y ensamblajes.
- Herramientas de diseño para la fabricación mecánica, que automatiza documentos de inspección y genera documentación sin planos 2D.
- Herramientas de simulación para evaluar el diseño y garantizar que es el mejor posible
- Herramientas que evalúan el impacto medioambiental del diseño durante su ciclo de vida.
- Herramientas que reutilizan los datos de CAD en 3D para simplificar el modo en que las empresas crean, conservan y utilizan contenidos para la comunicación técnica.

Además, presenta una interfaz intuitiva y sencilla lo cual ayuda a transformar rápidamente las ideas en productos reales.

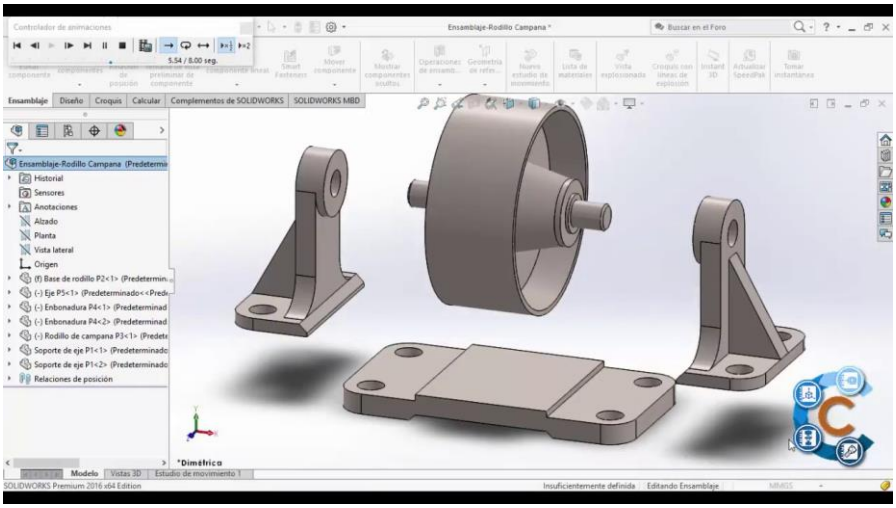


Imagen 3. Interfaz usuario SolidWorks

Se ha elegido emplear este programa para el diseño de los componentes del producto por su facilidad de manejo, y como se ha dicho anteriormente, por su cómoda interfaz, dando la posibilidad de crear piezas tridimensionales realizando croquis en los diferentes planos de la pieza, para posteriormente aplicar operaciones de extrusión. También cabe destacar la compatibilidad que tiene este software para poder convertir sus documentos a datos CAD para su fabricación rápida.

2.2.2 Matlab

MATLAB es una plataforma de programación y cálculo numérico utilizada por millones de ingenieros y científicos para analizar datos, desarrollar algoritmos y crear modelos.

La plataforma de MATLAB está optimizada para resolver problemas científicos y de ingeniería. El lenguaje de MATLAB, basado en matrices, es la forma más natural para expresar las matemáticas computacionales. Las gráficas integradas facilitan la visualización de los datos y la obtención de información a partir de ellos. Una vasta biblioteca de herramientas (Toolboxes) integradas le permite empezar a trabajar inmediatamente con algoritmos esenciales para su dominio. Todas estas herramientas y funciones de MATLAB están probadas rigurosamente y diseñadas para trabajar juntas. Sus características principales son:

- Lenguaje de alto nivel para cálculos científicos y de ingeniería.
- Entorno de escritorio optimizado para la exploración iterativa, el diseño y la solución de problemas.
- Gráficas para visualizar datos y herramientas para crear diagramas personalizados.
- Aplicaciones para ajustar curvas, clasificar datos, analizar señales, ajustar sistemas de control y muchas otras tareas.
- Toolboxes complementarias para una amplia variedad de aplicaciones científicas y de ingeniería.
- Herramientas para crear aplicaciones con interfaces de usuario personalizadas.
- Interfaces para C/C++, Java®, .NET, Python, SQL, Hadoop y Microsoft® Excel®.
- Opciones de implementación libres de derechos para compartir programas de MATLAB con los usuarios finales.

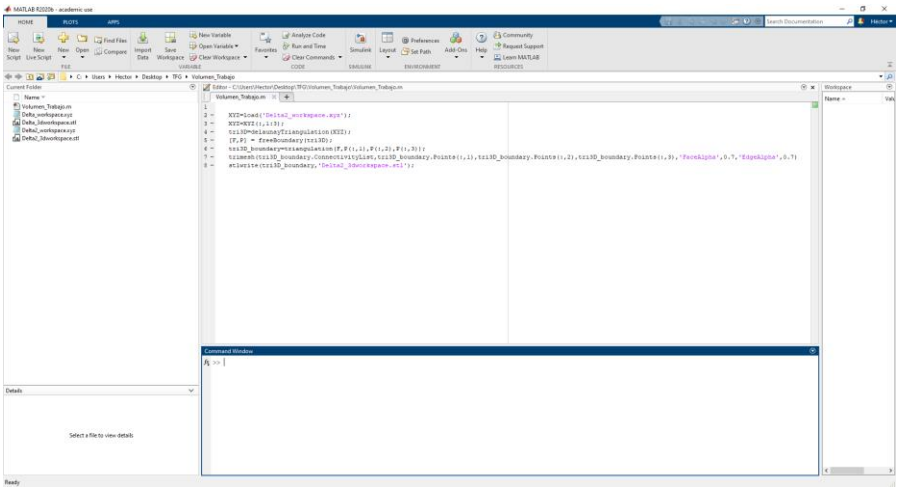


Imagen 4. Script ejemplo de Matlab

A pesar de las múltiples capacidades que tiene este programa, en este proyecto es necesario emplearlo para realizar un mapeado tridimensional de una sucesión de puntos que formarán un sólido, donde posteriormente se aplica un vaciado para obtener una malla (superficie del bloque creado). Este procedimiento se emplea para obtener la malla de volumen de trabajo del robot a diseñar, que se explicará a lo largo del proyecto.

Cabe destacar que este programa no se encuentra de forma gratuita y para su utilización la UPV ha facilitado su licencia gratuita.

2.2.3 CoppeliaSim

Está construido alrededor de una arquitectura de control distribuida que tiene scripts de Python y Lua, o complementos C / C++ que actúan como controladores síncronos individuales. Los controladores asincrónicos adicionales pueden ejecutarse en otro proceso, subproceso o máquina a través de varias soluciones de middleware (ROS, API remota, ZeroMQ), con lenguajes de programación como C/C++, Python, Java y Matlab.

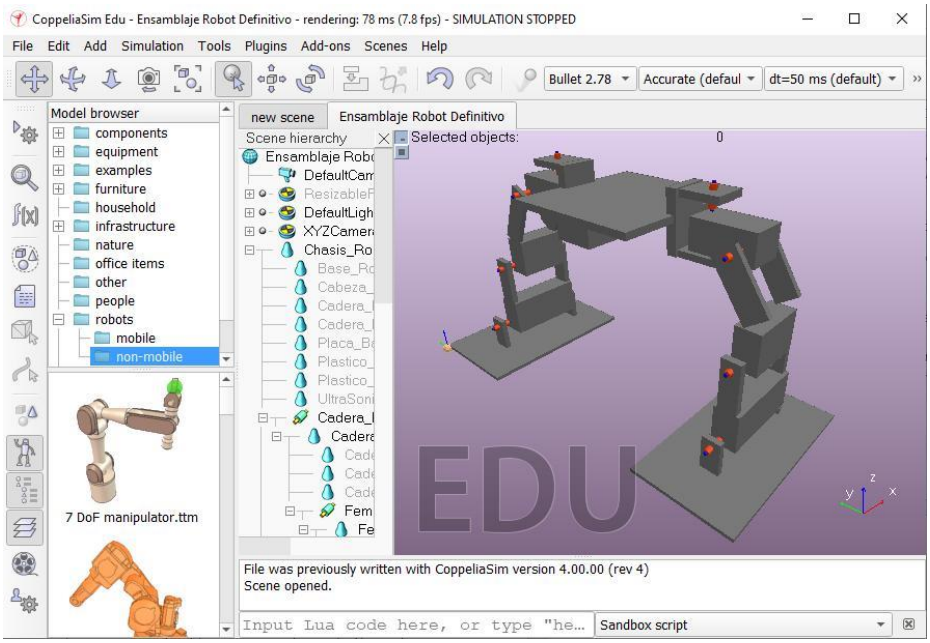


Imagen 5. Interfaz usuario CoppeliaSim

CoppeliaSim utiliza un motor de cinemática para los cálculos de cinemática directa e inversa, y varias bibliotecas de simulación de física (Bullet, ODE, Vortex, Newton Game Dynamics), para realizar la simulación de cuerpo rígido. Los modelos y escenas se construyen ensamblando varios objetos (mallas, uniones, varios sensores, nubes de puntos, árboles OC, etc.), en una estructura jerárquica. La funcionalidad adicional, proporcionada por complementos, incluye: planificación de movimiento (a través de OMPL), visión sintética y procesamiento de imágenes (por ejemplo, a través de OpenCV), detección de colisiones, cálculo de distancia mínima, interfaces gráficas de usuario personalizadas y visualización de datos (por ejemplo, a través de gráficos).

Los principales campos de aplicación de CoppeliaSim son la investigación en robótica y educación, por ello se utiliza principalmente para el desarrollo rápido de algoritmos, simulaciones de automatización de fábricas, creación rápida de prototipos y verificación, monitoreo remoto y educación relacionada con la robótica.

Este programa tiene especial importancia en la parte de simulación del robot, ya que se utiliza para comprobar el correcto funcionamiento cinemático del diseño que se emplee y para calcular su espacio de trabajo. También, una vez corroborados los procesos anteriores, se puede implementar la simulación de trayectorias y procesos del robot para su estudio y comprobación.

2.2.4 Arduino IDE

Arduino IDE es un software de código abierto que se utiliza para escribir y cargar código en las placas Arduino. La aplicación IDE es adecuada para diferentes sistemas operativos como Windows, Mac OS X y Linux. Soporta los lenguajes de programación C y C++. Aquí, IDE significa Entorno de desarrollo integrado.

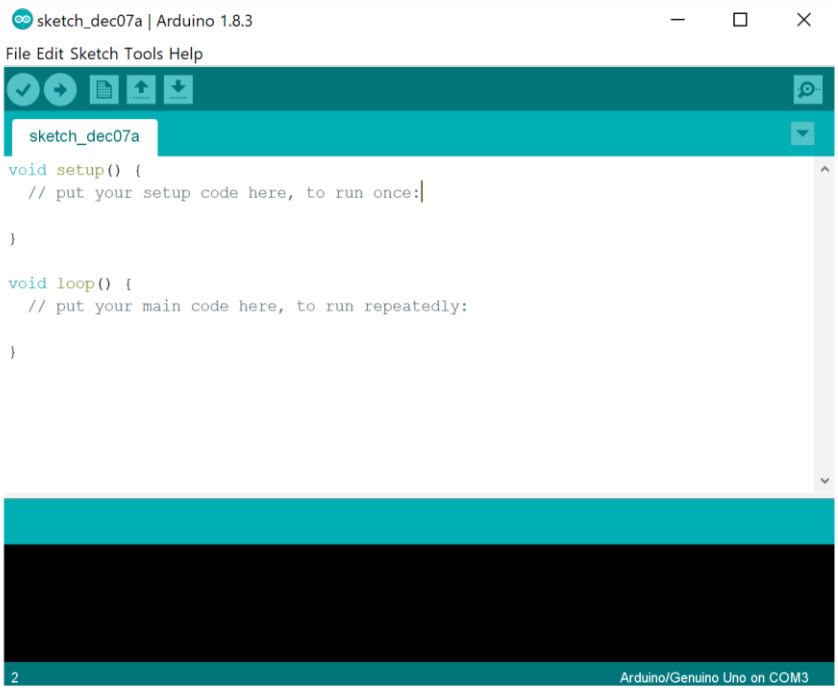


Imagen 6. Interfaz usuario Arduino IDE

La implementación de este software tiene la propiedad de crear un entorno de programación para que el usuario pueda desarrollar el código necesario para implementar un proceso cualquiera en el producto. No obstante, en este proyecto se desarrolla un código obtenido de los cálculos realizados, para habilitar la posibilidad de diferentes procesos a diseñar.

También cabe señalar que es un software libre y con capacidad suficiente para realizar el código de control del robot.

2.2.5 Meshmixer

Meshmixer es una aplicación de modelado en 3D con la cual se puede crear, reconstruir y esculpir diseños 3D de forma sencilla. Ofrece todos los servicios para mejorar las superficies de un objeto (modelar) o generar un modelo 3D desde el comienzo. El entorno es una muy buena alternativa gratuita que ofrece una gran cantidad de servicios para poder modelar en 3D.

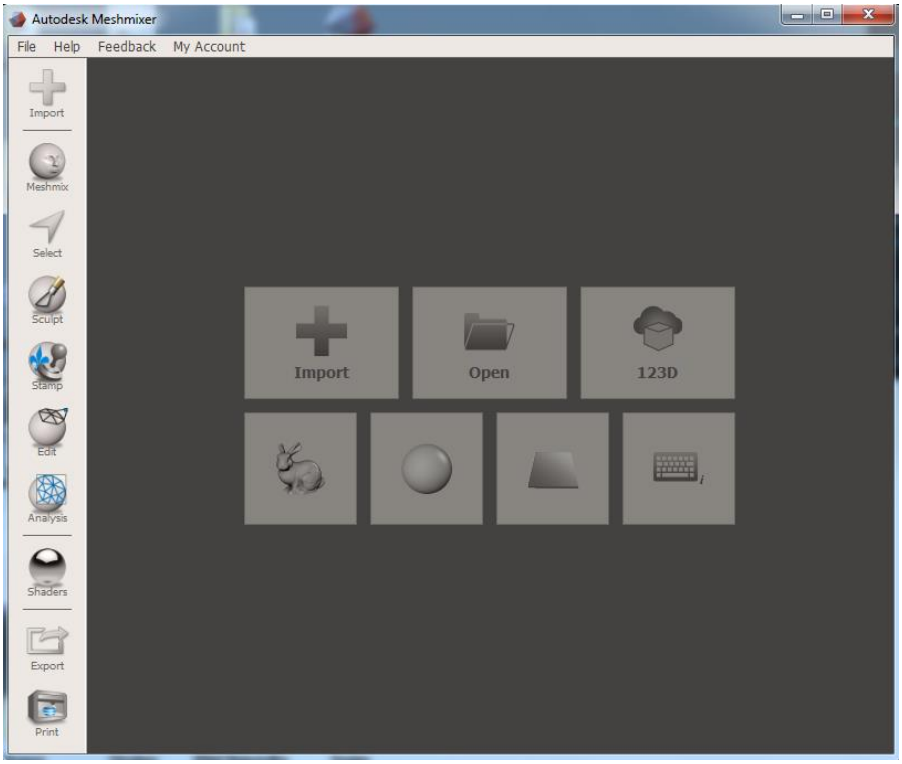


Imagen 7. Interfaz usuario Meshmixer

La finalidad de este programa es reducir el número de vértices (triángulos) de una superficie o malla, obteniendo un elemento más ligero y cómodo de trabajar. Este procedimiento se emplea para la simplificación de la malla de trabajo del robot, que se explicará con mayor detalle a lo largo del proyecto.

3 FACTORES A CONSIDERAR

3.1 Especificaciones

El presupuesto para elaborar este producto es la principal condición a la que se somete este proyecto para que se pueda considerar con fines académicos y sea asequible para cualquier público o gran parte de él. Debido a esto, se deberán plantear diferentes opciones o alternativas para cada uno de los aspectos tanto del hardware como del software del robot. La combinación entre ellos conformará un producto que satisfaga todas las especificaciones planteadas.

Como ya se ha destacado, reducir lo máximo posible el coste económico es primordial, pero también hay otros requerimientos que se han de cumplir:

- Emplear un material que sea fácil de manipular.
- Realizar un diseño de montaje que no requiera de una complejidad elevada, o el uso de herramientas complejas fuera de lo habitual dentro del uso doméstico.
- Que el robot pueda manipular objetos o herramientas de un máximo de 200 gramos.
- Que el conjunto total del robot sea cómodo de transportar y de interactuar con él.
- Adaptación para producción en masa.

3.2 Alternativas

3.2.1 Alternativas de microcontrolador

Hay diferentes tipos de microcontroladores que se podrían emplear en el proyecto, los cuales serán de tipo EEPROM, para que pueda ser programable y borrable electrónicamente, proporcionando un control y aprendizaje cómodo y total.

El microprocesador está diseñado para simplificar electrónicos complejos, reducir el coste y disminuir el coste energético. Los microprocesadores pueden ser de varios tipos, cuyas características vienen dadas por el número de bits necesarios, el tipo de memoria a emplear, o su arquitectura.

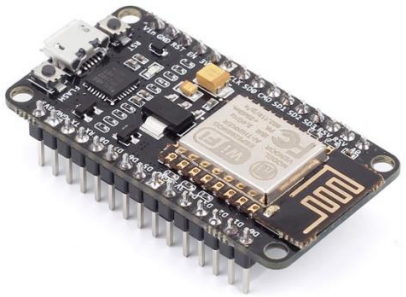


Imagen 8. NodeMCU V3 ESP8266

Las dos opciones principales que se plantea emplear son la versión Atmega128P con la placa Arduino, o el procesador ESP8266 con la placa de desarrollo ESP-01S. Para este proyecto se emplea una placa Arduino ya que su programación y lenguaje es más cómodo e intuitivo.

Dentro de la placa escogida se puede optar entre 28 tipos diferentes, siendo el más genérico y empleado el perteneciente a la familia clásica, Arduino UNO. Con este tipo de arduino se podría realizar el proyecto perfectamente sin añadirle ninguna complicación adicional. No obstante, se considera que la opción más adecuada es emplear un arduino perteneciente a la familia Nano. Este tipo de placa permite ser conectada a un adaptador que facilite la conexión con diferentes tipos de dispositivos electrónicos que requieran de entrada digital o de salida digital o analógica (sin necesidad de realizar ningún proceso de soldadura con estaño). Esto hace posible una conexión más cómoda de los actuadores consiguiendo así un esquema eléctrico más organizado, sin embargo, emplear esta opción conlleva un aumento en el coste del robot.

Finalmente, se emplea un Arduino Nano acoplado a un adaptador, priorizando una estructura eléctrica clara y simple, además de que hay diferentes marcas de acopladores (fuera de la marca oficial de Arduino) que pueden reducir el incremento de este coste.

3.2.2 Alternativas material del robot

La elección del material repercute en gran medida en el precio de coste final del producto pues constituye más del 80% del robot, por ello para su elección se precisa un estudio minucioso.

Dentro del mercado de los robots de tipo Delta, el principal material que se emplea es el plástico (para la carcasa y la plataforma móvil o fija), que puede ser plástico ABS o metacrilato, junto a algunas piezas (brazos del robot), hechas de una aleación de aluminio. El hecho de emplear aleaciones de diferentes metales conlleva un aumento considerable de peso en los componentes del robot, obligando a utilizar unos actuadores más potentes que garanticen la correcta funcionalidad del producto. Además, el coste de dichos materiales incrementa excesivamente el precio final del producto.

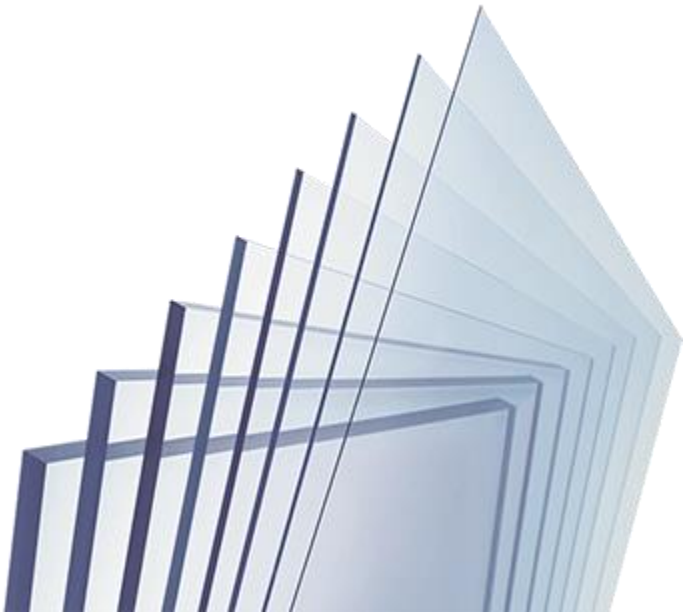


Imagen 9. Planchas de metacrilato transparentes

Otra alternativa a considerar es la utilización únicamente de plástico. Es un material ligero con un precio de coste no muy alto y es cómodo de manipular, sin embargo, emplearlo conlleva un tratamiento de elaboración más complejo. La obtención de las diferentes piezas que forman el robot se puede realizar empleando una impresora 3D o una inyectora de plásticos. Ambas alternativas acarrean inconvenientes. Emplear una impresora 3D implica una producción de piezas lenta dado que su proceso de elaboración es complejo y, en caso de acelerar la producción, se debería aumentar el número de impresoras, arrastrando así un coste final mayor. El inconveniente de utilizar una inyectora de plásticos es el alto coste de producción, aunque empleando este método se obtendría mejor calidad de pieza y tendría una mayor velocidad de elaboración.

La madera es otra posible opción. Al igual que el plástico, cumple con los requerimientos físicos necesarios para no poner en riesgo la integridad del robot, tiene un precio asequible, es fácil trabajar con ella y empleando una cortadora láser, la velocidad de elaboración de las

piezas es adecuada. Sin embargo, emplear este material conlleva limitaciones importantes a la hora del diseño del robot.

Para este proyecto se opta finalmente por la madera pues, a pesar del inconveniente de limitar y dificultar el diseño del robot, se prioriza su reducido coste y la velocidad y facilidad de manipulación.

3.2.3 Alternativas tipo de actuador

La elección del actuador es fundamental para asegurar el correcto funcionamiento del robot, a mayor par, mayor será el rendimiento y la rentabilidad del producto, pero también aumentará su coste económico.

El proyecto se puede llevar a cabo con motores paso a paso (como el Wantai 42BY), y con servomotores. En este caso se emplean actuadores de este último por su programación más sencilla, porque no requiere de ningún tipo de driver y por tener mayor precisión angular.

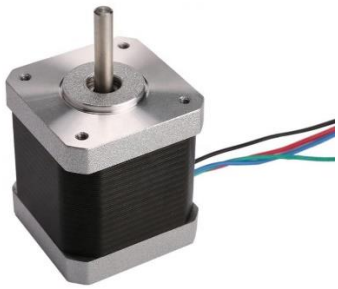


Imagen 10. Motor paso a paso Wantai 42BYGH Nema 17

A través de simulaciones se ha podido comprobar que el rango de trabajo de los actuadores es menor de 180 grados, por lo que la elección más intuitiva sería emplear servomotores con un rango máximo de 180 grados, aunque con rangos completos de 360 grados también se podría realizar el proyecto perfectamente sin necesidad de modificar el código de programación.

Actualmente, en el mercado, la variedad y disponibilidad de los servomotores está al alcance de cualquier usuario. Para este proyecto se deben emplear servomotores de uso académico de bajo coste, los cuales, en comparación con motores de uso industrial, tienen una diferencia de calidad y precisión excesiva, pero para la finalidad que se le quiere dar en este proyecto es más que suficiente.

Los servomotores de engranaje metálico, con un par de 11Kg/cm. podrían considerarse como otra opción. La razón por la que se considera emplear este tipo de servomotores es el público al que se orienta este proyecto, ya que, si no se manipulan de forma correcta los motores con engranajes de plástico, se puede dañar alguno de los dientes de sus engranajes, dejándolo

prácticamente inutilizable. Emplear este tipo de servomotores conlleva un incremento en el coste del producto y una fuente de alimentación mayor.

Imagen 11. Micro servomotor SG90 de 1Kg/cm

También se contempla la posibilidad de emplear servomotores con engranajes de plástico con un par de 1Kg/cm. Esta alternativa limita considerablemente la productividad del robot pues no se podrán emplear herramientas pesadas, pero también reduce notablemente el coste del producto.

Asimismo, se pueden emplear servomotores con un par entre 1Kg/cm y 11Kg/cm, que se pueden encontrar con engranajes de plástico o metálicos, por lo que la variedad de elección es amplia.

Para este proyecto se opta finalmente por los servomotores de 11Kg/cm, ya que, a pesar de tener el precio más alto en el mercado, dentro del abanico de posibilidades que se ha planteado anteriormente, este precio se puede asumir perfectamente y no aumentaría en exceso el coste. De este modo se puede ofrecer un producto con una calidad de trabajo alta, con la posibilidad de añadir el peso de una herramienta al efector final y la independencia de no estar altamente condicionado a minimizar el peso del robot a la hora de realizar su diseño.

Cabe destacar que la elección final del servomotor condiciona la elección final de la fuente de alimentación y una vez realizado el diseño final del producto se realizarán los cálculos pertinentes para corroborar su correcto funcionamiento.

3.2.4 Alternativas fuente de alimentación

La elección de la fuente de alimentación depende del uso que se le quiera dar al producto. Por lo general, los robots Delta tienen finalidades industriales para agilizar procesos que requieran de precisión y poco peso y su ubicación en dichos procesos suele ser en un punto fijo, conectado a una toma de corriente eléctrica.

Imagen 12. Fuente de alimentación fija de Voltaje CC Variable XP-752A

Dado que una de las especificaciones del proyecto es ofrecer un producto cómodo de transportar y cargar, emplear una fuente de alimentación fija lo dificulta. Además, tanto el microcontrolador como los actuadores que se emplean no necesitan más de 5V de entrada, por lo que haría falta emplear un transformador para obtener de él la salida deseada, suponiendo esto un incremento considerable en el coste económico del robot.

Se ha de tener en cuenta también que tal vez el posible usuario no tenga una toma de red eléctrica donde desee utilizar el producto, por lo que se podrían emplear pilas como fuente de alimentación, lo cual permitiría usar el artículo en el lugar que se crea oportuno.

Por otra parte, la batería de litio recargable podría considerarse una mejor opción dado que con el tiempo, el coste económico de esta será inferior al de las pilas. Además, la durabilidad de las pilas es corta y pueden dejar de funcionar en cualquier momento mientras que la batería recargable, como su nombre indica, puede utilizarse de manera portátil y cargarla en una toma eléctrica cuando se necesite.

Sin embargo, las baterías recargables presentan un problema relacionado con la compatibilidad pues tienen un número de recargas limitada y presentarían una dificultad relacionada con la reemplazabilidad, acortando así la vida útil del dispositivo. También crean la necesidad de emplear un porta-baterías donde cargarlas y obtener la salida de tensión deseada, aumentando ligeramente el coste del producto.

Finalmente se escoge emplear baterías de litio, ya que como se ha mencionado anteriormente, garantizan obtener un producto portátil, reduce su coste económico y ayuda al medio ambiente no fomentando el uso y desecho de pilas.

4 SOLUCIÓN ADOPTADA

A continuación, se detalla la elaboración final del producto de este proyecto, atendiendo a las alternativas y decisiones ya tomadas en el apartado anterior e indagando en otros campos

más prácticos, como los diferentes cálculos y simulaciones, que son necesarios para contrastar el correcto funcionamiento del robot.

4.1 Hardware

4.1.1 Materiales

Dentro de este apartado se especifica el material que constituirá el robot al completo quitando los componentes electrónicos y los diferentes elementos de sujeción (tornillos).

El material empleado es madera mdf, que es producto de madera reconstruida que se obtiene descomponiendo residuos de madera dura o blanda en fibras de madera y combinándolo con cera y un aglutinante de resina. Se forman paneles mediante la aplicación de alta temperatura y presión. Es un material más denso que el contrachapado y es más fuerte y denso que el aglomerado.

Para manipular este material y obtener las piezas deseadas se emplea una cortadora láser Trotec Láser, con una superficie de trabajo de 600x300 mm, pudiendo trabajar con superficies de espesor de 8 mm.

4.1.2 Elementos de sujeción

Se requieren componentes de unión que garanticen la integridad del robot. A pesar de ser un producto académico el cual se puede considerar casero y cuyas piezas, en su gran mayoría, se podrían unir empleando cualquier tipo de adhesivo, en este proyecto se utilizan tornillos de nylon con cabeza redonda en cruz, de M3x10 mm, pues se pretende ofrecer un producto que pueda ser ensamblado y desensamblado a voluntad del usuario.

Imagen 13. Tornillos de nylon cabeza redonda con cruz

Dado que no todas las sujeciones se enroscan al material del robot (lo cual daña el material y dificulta su reutilización), también se emplean tuercas hexagonales M3 de plástico.

Imagen 14. Tuercas hexagonales M3 de plástico

Cabe señalar que emplear elementos de sujeción de un material tan moldeable y deformable podría poner en peligro la estabilidad del producto. No obstante, dada la finalidad del proyecto y de sus especificaciones, emplear dicho material no condiciona negativamente al robot y el precio de emplear estos elementos se reduce considerablemente, teniendo en cuenta, además, que emplear elementos de sujeción de otro material como el metal, aumentaría el peso de cada brazo del robot, exigiendo así un mayor par a los actuadores.

4.1.3 Batería

Como fuente de alimentación del sistema se emplea la batería 18650 Li-Ion recargable. Esta batería tiene una tensión normalizada de 3.7 V y puede alcanzar una capacidad de carga de 2600 mAh. Su voltaje de carga completa y descarga de corte es de 4.2 V y 2.75 V respectivamente y tiene un peso de 45.6 gramos. Sus dimensiones son 1.82 cm de diámetro y 6.9 cm de longitud, lo cual hace que sea cómoda de emplear y no supone ningún tipo de condicionante para el diseño del robot.

Imagen 15. Batería 3.7 V 18650 Li-Ion 2600 mAh

Respecto a la intensidad, aproximadamente cada micro servomotor consume 500 mA/h. Dado que el robot está formado por tres de estos y que el consumo del microprocesador es prácticamente despreciable, el consumo total del producto es de 1500 mA/h (cabe señalar que este valor puede variar en función de la fuerza que apliquen los micro servomotores o del funcionamiento que estén llevando a cabo).

Puesto que se desea que tenga una autonomía mínima de 1.5 horas, se emplea la ecuación (1), para obtener la capacidad que debe poseer la batería.

$$Capacidad = \frac{1500\ mA/h * 1.5\ h}{1\ h} = 2250\ mA/h \quad (1)$$

Viendo el resultado obtenido se puede apreciar que con la batería empleada alcanzamos con creces la capacidad mínima exigida, de hecho, como muestra la ecuación (2), se consigue una autonomía de 1.73 h.

$$Autonomía = \frac{2600\ mA/h * 1\ h}{1500\ mA/h} = 1.73\ h \quad (2)$$

4.1.3.1 Cargador

La batería de litio elegida es recargable, por lo que es necesario emplear un soporte donde poder recargar y obtener la salida de tensión deseada. Se emplea el Soporte Batería 18650 Simple Powerbank, el cual proporciona 5 V a través de un conector USB de salida, y también puede proporcionar 5 V y 3.3 V a través de unos pines accesibles.

Imagen 16. Soporte Batería 18650 Simple Powerbank

Tiene unas dimensiones de 98x38.6 mm, que al igual que la batería, dado su pequeño tamaño, no supone ningún impedimento para este proyecto. También dispone de una entrada micro-USB por donde se alimenta la batería para recargarla, e incluye un Switch para encender y apagar la fuente de alimentación a voluntad del usuario. Además, un indicador led informa de si la carga está completa o en proceso.

Evidentemente, para poder realizar la carga de la batería se dispone de un cable genérico USB 2.0 tipo A a mini-USB tipo B, con una velocidad de transferencia de 480 Mbps.

Imagen 17. Cable mini-USB

4.1.4 Servomotores

Como actuador se emplean Micro servomotores MG995R Tower Pro. Es un servo de reducidas dimensiones y económico que tiene un conector universal tipo “S. Los cables en el conector están distribuidos de forma que el rojo es alimentación, el marrón es toma a tierra y el naranja pertenece a la señal PPM.2

Imagen 18. Micro servo MG996R

Tiene unas dimensiones de 40x19x43 mm y pesa 55 gramos, ya que este producto está orientado a una primera experiencia de aprendizaje y no como un uso industrializado. Para una alimentación de 4.8 V (~5 V) tiene un par de aproximado de 9 kg/cm y una velocidad angular de 0.17 seg/60º, lo cual es suficiente para garantizar un funcionamiento óptimo. Su rango de movimiento es de 180º.

4.1.5 Microprocesador

Un componente fundamental en el sistema eléctrico es el microprocesador, por ello, la correcta elección de este es muy importante para el buen funcionamiento del producto.

Como se ha mencionado anteriormente, se emplea un microprocesador Arduino Nano v3.0 con controlador ATMEGA328 acoplado a un Extension Shield para este tipo de microprocesador, cuya finalidad es la de facilitar la conexión del Arduino con el robot.

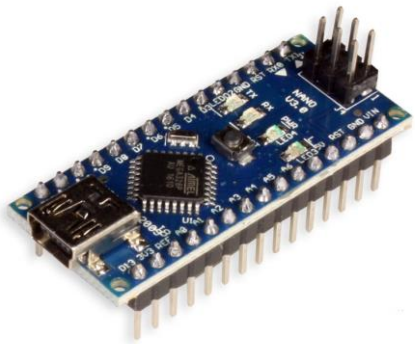


Imagen 19. Arduino Nano v3.0

El Extension Shield está constituido por 14 entradas/salidas digitales, 6 pines de salida PWM, 8 pines de salidas analógicas, pines para la interfaz I2C, conector de alimentación power jack y botón de reset. La superficie que ocupa es de 56x53 mm aproximadamente, lo cual hace que el microprocesador tenga las mismas dimensiones que un arduino UNO estándar.
A pesar de todas las posibilidades que ofrece este acoplador, para este proyecto solo se hará uso de tres pines digitales como salida.

Imagen 20. I/O Extension Shield para Arduino Nano

Como se ha justificado previamente, la razón principal por la que se decide emplear esta alternativa en lugar de utilizar únicamente un Arduino UNO, es porque aporta una conexión entre Arduino y robot más cómoda e intuitiva, ya que como se puede apreciar en la *Imagen 20*, tanto los pines digitales como los analógicos están distribuidos de forma que cada pin tiene su propia conexión de alimentación y de masa, en lugar de compartir una única conexión para todos. La *Imagen 18* muestra a su vez su perfecta adaptación para conectarse a la entrada del micro servomotor. También capacita para emplear cualquier otro tipo de actuador o sensor analógico para completar este proyecto, lo que da una infinidad de posibilidades para el usuario en su aprendizaje.

4.1.6 Esquema eléctrico

Como ya se ha destacado, gracias a emplear el Extension Shield (*Imagen 20*), el esquema eléctrico del robot es muy sencillo y únicamente se precisan dos cables auxiliares de 10 cm Macho-Hembra para poder alimentar el microprocesador.

Dado que solo se emplean tres micro servomotores para este proyecto, la elección de tres pines entre los catorce que hay, es libre, aunque se recomienda que la conexión de los motores sea consecutiva para no crear confusión a la hora de realizar la programación.

Respecto a la fuente de alimentación, se debe emplear un cable Macho-Hembra para conectar el Powerbank a la entrada de alimentación del Shield. Se puede escoger entre cualquiera de los pines de salida de 5 V que ofrece el soporte. En cuanto al acoplador, se puede hace uso de cualquiera de las dos entradas de alimentación para tensiones de 5 V.

Un ejemplo de una posible conexión eléctrica es el siguiente:

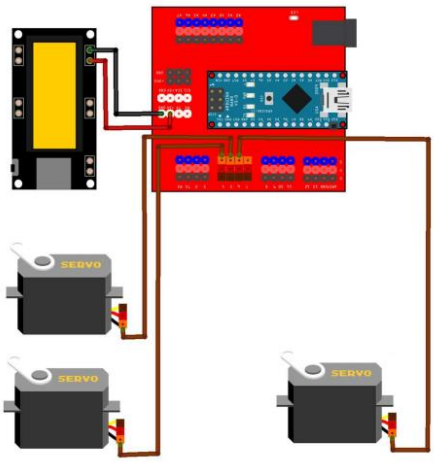


Imagen 21. Esquema eléctrico propuesto

4.2 Diseño

A continuación, se redacta el procedimiento a seguir para el diseño del robot con el objetivo de tener una vista preliminar con la que trabajar y hacer cálculos. La finalidad es obtener el mejor resultado posible del producto. Cabe indicar que el software a emplear para este procedimiento es SolidWorks.

Puesto que el diseño es un concepto libre y se pueden emplear diferentes opciones para obtener un mismo resultado, su elaboración se divide en dos apartados. El primero corresponde al planteamiento, donde se consideran todas las opciones posibles, y un segundo en el que se escoge y se define el diseño final del robot.

4.2.1 Planteamiento

El diseño a plantear no difiere en exceso del de un Robot Delta existente. El robot está formado por tres actuadores, que forman tres cadenas cinemáticas basadas en el uso del paralelogramo, las cuales unen dos bases, donde la base superior se encuentra fija, mientras

que la base inferior, donde se ubica el efector final, es móvil y siempre está paralela a la base fija.

Las partes fundamentales que componen el robot son: la base, el brazo, las juntas o uniones, el paralelogramo y el efector. Planteando diferentes alternativas para estos componentes, se llega a la elaboración de un diseño sólido y eficiente.

4.2.1.1 Base

Es el eslabón que está compuesto por las piezas que dan soporte a los actuadores y las demás piezas móviles del robot (los brazos). Dado que debe poder posicionar los brazos manipuladores a una distancia exacta de 120º para cumplir con el comportamiento cinemático del manipulador, el diseño que constituye esta pieza puede adoptar diferentes formas.

En la primera alternativa, la base consta de una placa central de forma circular donde, junto a tres superficies salientes situadas a una distancia equidistante entre sí y orientadas a 120º respecto las otras, se sitúan las piezas de sujeción de los actuadores.

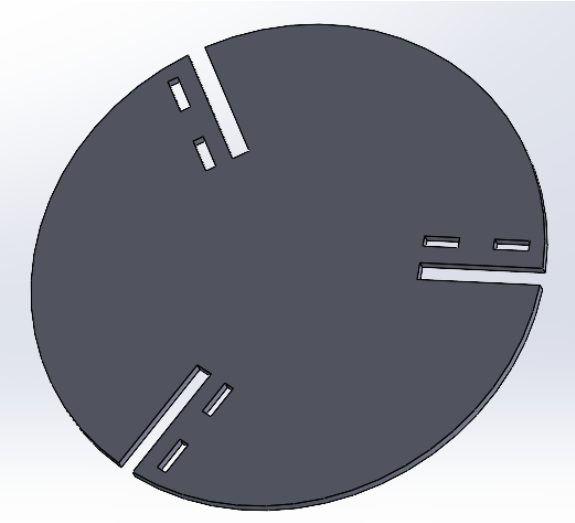


Imagen 22. Base circular

Hay que indicar que al situar los actuadores sobre la superficie de la base y no debajo, hace falta realizar un recorte para que pueda circular libremente el brazo sin colisionar con la base, aunque también se podrían situar debajo de la base para evitar hacer dicho ajuste.

Otra alternativa sería emplear el mismo procedimiento y metodología, pero dándole a la base una forma triangular, permitiendo así una mayor superficie de trabajo. Asimismo, al igual que en la otra alternativa, se puede realizar un pequeño orificio para dar la posibilidad de situar un eje que actúe directamente sobre la herramienta a través de un cuarto actuador, o para darle alguna otra finalidad, la cual no se detalla en este proyecto.

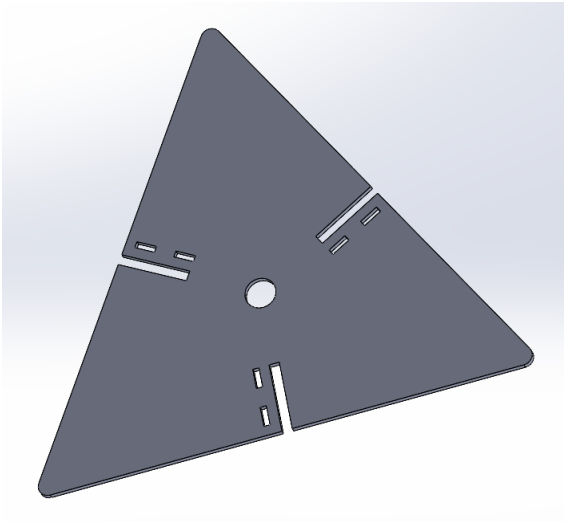


Imagen 23. Base triangular

4.2.1.2 Juntas Paralelogramo

La función del paralelogramo dentro de la cinemática del robot es poder transmitir los movimientos del brazo-actuador hacia el efector por medio de la conversión de las rotaciones en desplazamientos traslacionales. Para realizar esta acción, la junta del paralelogramo debe pivotar respecto al eje horizontal del brazo, que permite realizar las rotaciones que no son posibles para el motor y así complementar el mecanismo del robot de arquitectura paralela. Dentro de las rotaciones que debe realizar este sistema de junta-paralelogramo está el movimiento angular sobre el eje horizontal (en adelante eje Y), y el movimiento sobre el eje perpendicular a este y paralelo a la base, que se nombra como eje X.

Una alternativa posible consiste en emplear dos ejes de revolución, uno situado al extremo opuesto del brazo, que permite las variaciones de altura del efector final, y otro perpendicular al anterior, permitiendo las traslaciones sobre el plano donde se sitúa la base móvil (efector), consiguiendo así los grados de libertad pertinentes.

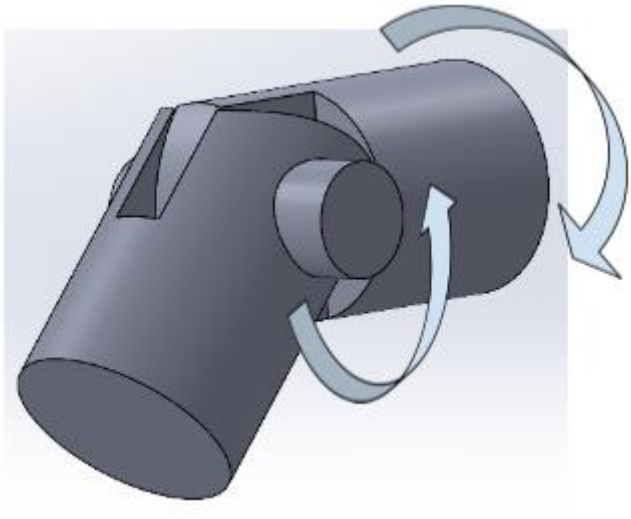


Imagen 24. Junta de revolución en Y

Otra posibilidad es emplear juntas esféricas que se emplean ampliamente en robots de arquitectura paralela y ofrecen un gran rango de movimientos posibles o grados de libertad, permitiendo traslación y rotación en los tres ejes, y que además son de gran precisión.

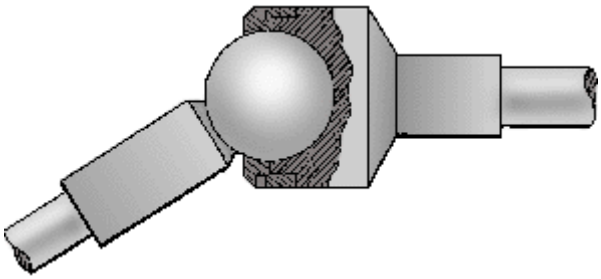


Imagen 25. Junta esférica

No obstante, emplear este tipo de articulación requiere un gran coste de adquisición, lo que aumenta considerablemente el coste de producción del robot y requeriría la utilización de un material de mayor calidad como el aluminio, el cual añade un peso considerable al brazo del robot, exigiendo un mayor par a los micro servomotores.

También existe la opción de combinar una articulación de revolución con otra articulación prismática para así conseguir los grados de libertad requeridos.

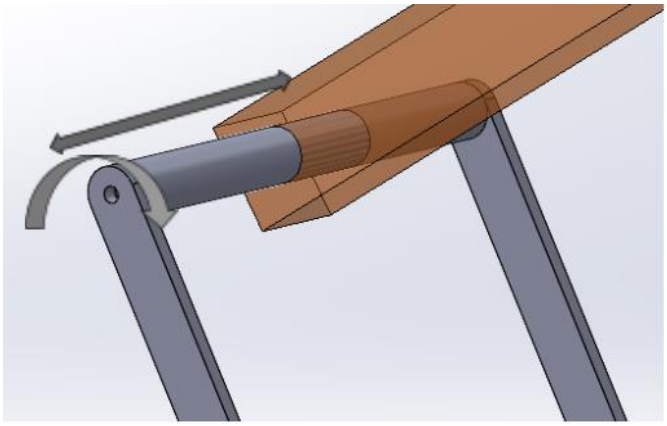


Imagen 26. Junta prismática

Empleando este método se consigue reducir el coste de producción del robot y garantiza una estabilidad aceptable, ya que el paralelogramo está formado por el conjunto de cuatro piezas que se unifican para formar un único eslabón sólido. Sin embargo, la superficie de rozamiento entre sus eslabones es mayor, lo que implica mayor resistencia de desplazamiento con respecto a las otras alternativas, aunque dado la fuerza negativa que esto genera, se podría despreciar pues apenas es cuantificable.

4.2.1.3 Efector

Este componente tiene la finalidad de sostener la herramienta del robot dando cierre a la cadena cinemática del Delta.

Las juntas que se emplean en esta pieza son las mismas que se han utilizado para la unión del brazo y del paralelogramo explicado anteriormente, por lo que hay que especificar el

diseño de este componente de forma correcta, ya que es el eslabón que se mueve libremente dentro del volumen de trabajo del robot.

En este proyecto no se emplea ningún tipo de herramienta a colocar en el efector final, por lo tanto, no se precisa de condición de elaboración, pudiendo diseñarse un modelo más genérico para que posteriormente el usuario, de forma independiente, decida darle alguna utilidad.

Existe la posible alternativa de emplear la misma forma geométrica que la base fija, pero reduciendo su tamaño y utilizando una geometría para ensamblar diferentes sistemas de agarres, haciendo uso de las piezas pertinentes (agarre revolución en Y, agarre esférico, o agarre prismático). Un ejemplo, considerando la opción de realizar la base fija triangular, es el siguiente:

Imagen 27. Efector triangular

El posible inconveniente que podemos encontrar en esta opción es el escaso grosor de la pieza ya que puede presentar problemas de estabilidad para el uso de una herramienta. Por ello, consideraríamos la opción de aumentar el espesor de este eslabón, lo cual acarrearía más problemas al diseño del robot. También podríamos duplicar la pieza, aumentando así su volumen, pero dando una mayor estabilidad práctica al robot y ofreciendo la posibilidad de interactuar mejor con ella. Un ejemplo de esta última opción aplicado a una posible junta prismática es:

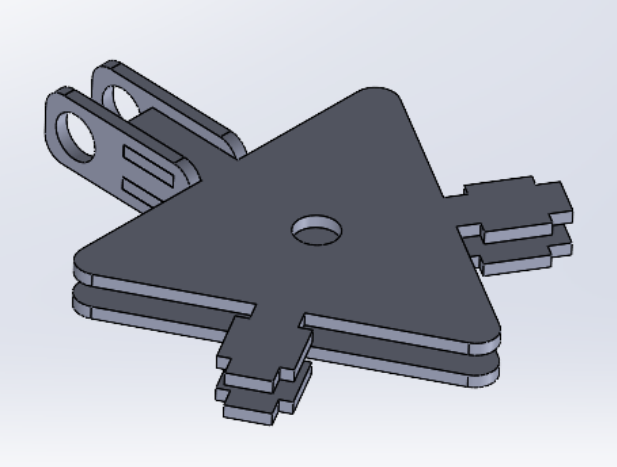


Imagen 28. Efector triangular de doble superficie con junta prismática

El diseño de los eslabones que forman la junta prismática se adapta para el perfecto acople con ambos efectores finales, un diseño práctico el cual también se puede utilizar para los otros métodos de sujeción entre efector y paralelogramo. En esta opción cabe señalar que los orificios céntricos se realizan en ambas superficies para obtener una buena comunicación de la herramienta con el espacio de trabajo.

Para implementar un servomotor como herramienta, ambas opciones serían aceptables para el diseño. De hecho, emplear una única superficie es más rentable dado que quita peso al efector final del robot y reduce su coste de elaboración, pero la razón de plantear un efector con dos superficies es para darle un uso más académico y básico donde, en lugar de emplear otro actuador como herramienta, se puede utilizar algo más básico con lo que empezar.

Un ejemplo es emplear un boli, un rotulador o cualquier otro tipo de objeto similar, donde usando ambas superficies como punto de apoyo y realizando el orificio inferior de un diámetro más pequeño para aguantar el objeto, se puede hacer uso de él. Además, como se ha mencionado anteriormente, este planteamiento no imposibilita el uso de actuadores en el efector final, en caso de que el usuario quiera realizarlo por su propia cuenta.

4.2.1.4 Brazo

El brazo se considera el primer elemento dentro de la cadena cinemática ya que es aquel que recibe la potencia del actuador para transmitirla a los otros elementos móviles del robot. Este eslabón debe tener un rango de movimiento de aproximadamente 180º aunque de forma práctica nunca vaya a completar dicho rango, por ello, se debe permitir un grado de rotación en el plano perpendicular al eje del actuador.

Su complejidad de elaboración en casos industriales es muy alta, ya que se deben considerar las fuerzas de torsión aplicadas en él y los diferentes métodos para minimizar los esfuerzos generados por los cambios de movimientos en las diferentes trayectorias. No obstante, dadas las especificaciones y propiedades de este proyecto, no se entrará a realizar cálculos en estos campos.

El brazo de este proyecto únicamente queda condicionado a tener dos puntos opuestos que permitan la sujeción tanto del eje del actuador, que hará pivotar el brazo, como del elemento que conecta a este con las juntas del paralelogramo. Dado que el robot no está planteado para que soporte elevadas cantidades de peso, el grosor de esta pieza se puede reducir considerablemente, manteniendo el grosor del resto del conjunto del robot (3 mm).

Se empieza por el diseño de sujeción entre el brazo y el eje del micro servomotor, donde empleando los accesorios de brazos que viene con el servo (*Imagen 29*), se hace más fácil su anclaje.

Imagen 29. Accesorios brazos para servo

El planteamiento de esta sujeción es muy parecido al de cualquiera de los accesorios que se quiera emplear, pues para el procedimiento que se va a realizar solo se modifican ligeramente entre ellos.

Los brazos para servo que se pueden observar en la *Imagen 29* se anclan y atornillan directamente al actuador y, como se puede apreciar, a lo largo de la pieza hay unos orificios de reducido tamaño donde, empleando los tornillos en punta de hierro que vienen con él, se puede sujetar otro elemento que en este caso es al brazo del robot. Señalemos que estas piezas son de un material plástico poco resistente por lo que una vez se haya deformado el material introduciendo un tornillo o cualquier otro medio de sujeción, este no volverá a su estado original cuando se quiera extraer la pieza y sacar los tornillos.

Como vemos, las alternativas son numerosas, dadas las especificaciones del proyecto, pues tanto el diseño más simple como el más complejo serían perfectamente aptos para este proyecto, siempre que no incremente de forma excesiva el peso del robot. Lo único que varía en esta pieza es la implementación del tipo de sujeción a emplear, la cual depende de la elección del tipo de juntas del paralelogramo que se quiere implementar y que se verá directamente en el diseño final del producto.

4.2.2 Primer diseño de ensamblaje

En este apartado se desarrolla el primer ensamblaje que se implementa en el proyecto, empleando las diferentes alternativas explicadas anteriormente y realizando algunas modificaciones que se creen pertinentes.

Se empieza por el diseño de la base, para el que finalmente se ha escogido emplear una superficie triangular. La razón de esta decisión viene dada por la necesidad de emplear pilares para la correcta sujeción del robot sobre una superficie plana. Aunque el diseño tradicional del Robot Delta esté elaborado para anclarlo al techo o a alguna otra superficie, en este proyecto se quiere dar más movilidad al robot, pudiendo ser desplazado cómodamente por el usuario. Por ello, emplear el diseño circular hubiera dificultado la implementación de los pilares por la reducida superficie de trabajo, provocando posibles colisiones del efector final con los pilares.

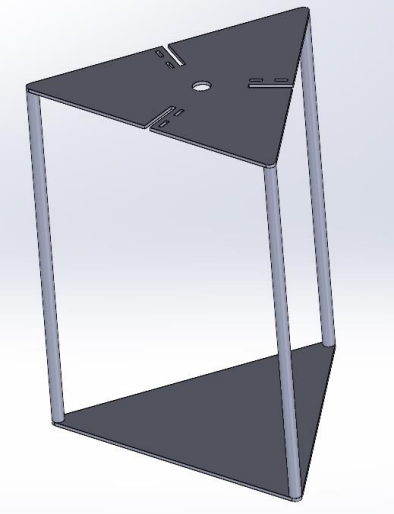


Imagen 30. Base fija del Robot Delta

Por otra parte, como se puede apreciar en la *Imagen 30*, se ha decidido añadir una segunda superficie donde anclar los pilares, para incrementar la superficie de apoyo al robot y conseguir así una mayor estabilidad y rigidez del producto.

Para la sujeción de los actuadores se han diseñado dos piezas, ambas ancladas a la base fija. La pieza de mayor tamaño se encarga de dar cabida al servomotor, cuyos orificios a los lados concuerdan con los orificios del MG996 (ver *Imagen 18*) para poder emplear un tornillo con su arandela y tuerca para la sujeción del mismo.

La segunda pieza, de un tamaño más reducido, tiene la finalidad de evitar el desplazamiento vertical del actuador. Al tener un peso elevado, la primera pieza no puede soportar todo su peso por la ubicación en que se encuentra respecto el servo. Por ello se realiza una pequeña plataforma que se fija a la base del robot para impedir que el extremo opuesto del servomotor ceda a la fuerza de la gravedad y se incline.

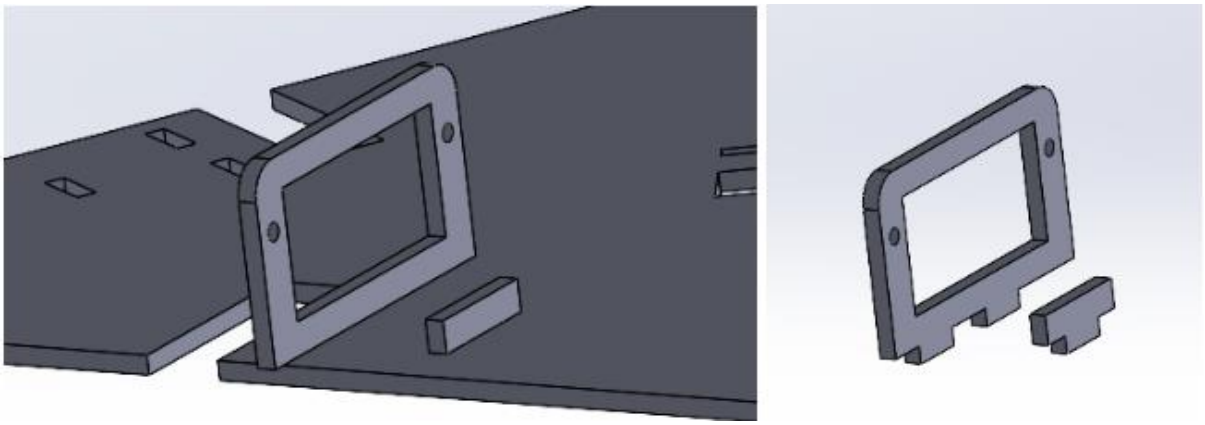


Imagen 31. Sujeción adaptada para servomotores MG996

Para realizar el diseño del brazo se ha decidido emplear como junta entre brazo y paralelogramo, una articulación de revolución, siendo el diseño inicial como el que se puede observar en la *Imagen 32*.

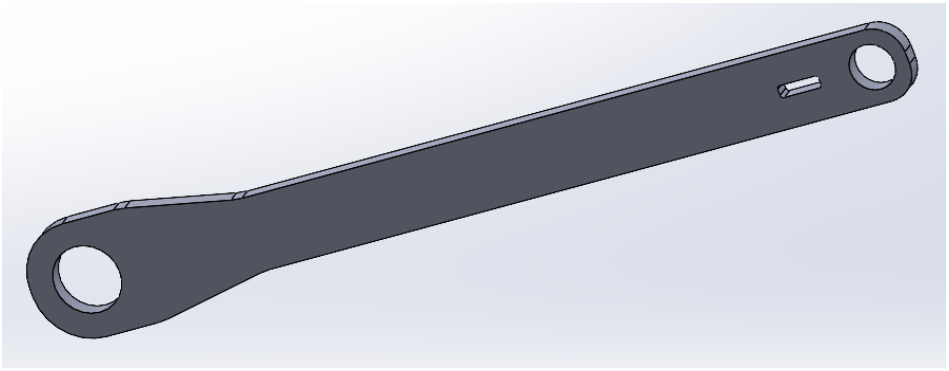


Imagen 32. Brazo Robot Delta

El orificio situado en el extremo estrecho de la pieza sería el encargado de la sujeción del brazo junto al eje del actuador, y el orificio de mayor diámetro, situado en el extremo más ancho de la pieza, sería la sujeción entre el brazo y en paralelogramo.

No obstante, emplear únicamente una pieza de grosor 3 mm como brazo del robot suponía un incremento en la inestabilidad ya que podía ceder ante fuerzas laterales creando así torsión y comprometiendo la integridad física del producto. Por ello, para aumentar su rigidez y evitar la aparición de la situación anterior, se ha decidido utilizar una segunda pieza idéntica, con el objetivo de aumentar al doble el ancho de esta. Para mantener ambas piezas unidas y constituirlas como un único eslabón, se emplea cola de contacto para superficies de madera, considerado el mejor adhesivo para trabajar con estos materiales y que no requiere modificar el diseño inicial de la pieza ni exige una alta complejidad de utilización.

Con esto, se pasaría al diseño del paralelogramo, al que también se puede denominar “brazo pasivo”, que está constituido por cuatro piezas parejas.

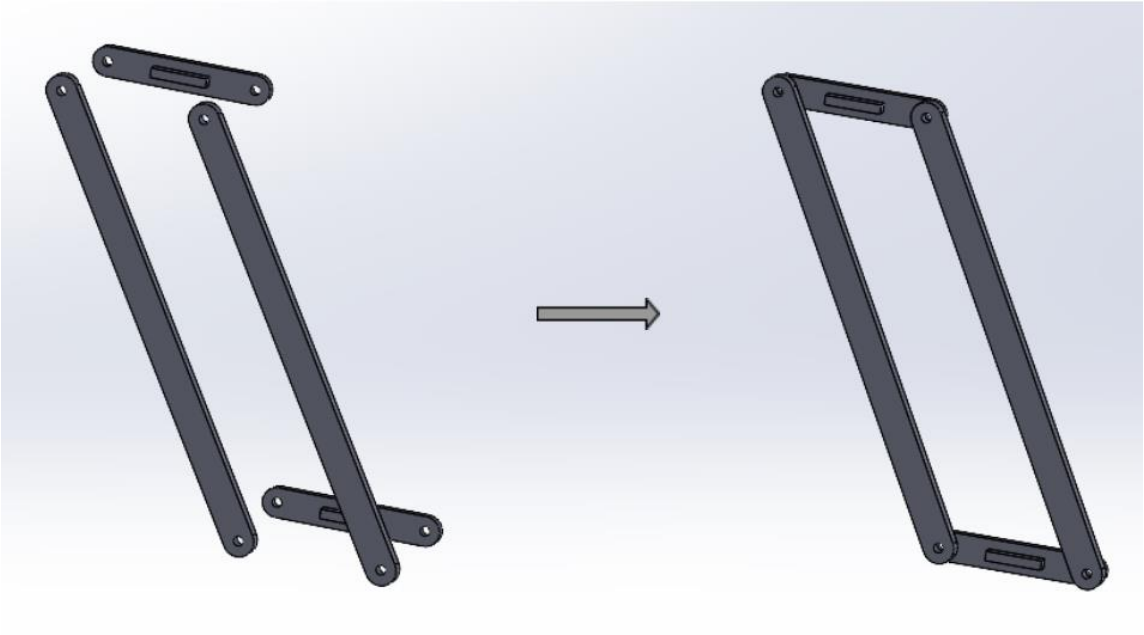


Imagen 33. Paralelogramo Robot Delta

La complejidad de este diseño es mínima, únicamente debe permitir que los ejes situados en los extremos de esta pieza puedan desplazarse de forma angular. La solución a dicha especificación consiste en emplear elementos cilíndricos que permitan esos grados de libertad.

Unos de los requisitos de este proyecto es hacer que su elaboración sea lo más cómoda y rápida posible, así pues, no se pueden emplear piezas cilíndricas que no se pueden obtener ni manipular en una cortadora láser. Para sustituir la utilización de cilindros como eje de revolución entre el lado más corto del paralelogramo y el del efector final (o brazo del robot), se emplea un eje en cruz constituido por dos piezas, mejorando además la dinámica del robot al reducir la superficie de rozamiento en la junta de revolución.

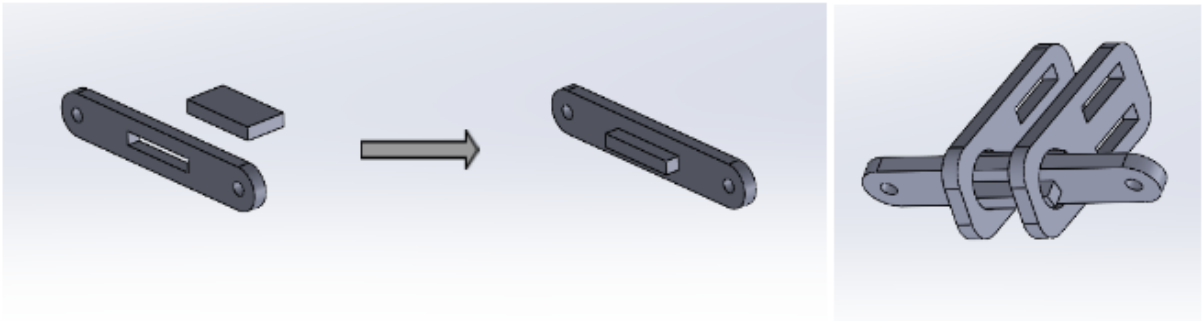


Imagen 34. Lado corto paralelogramo Robot Delta

Por último, se llega al efector final del robot haciendo uso del último diseño aportado en el planteamiento de este eslabón (*Imagen 28*). Finalmente se opta por un efector final constituido por dos superficies separadas entre sí 5 mm, donde la junta que une el paralelogramo con el efector también tiene la finalidad de mantener ambas superficies constantemente paralelas entre ellas y equidistantes.

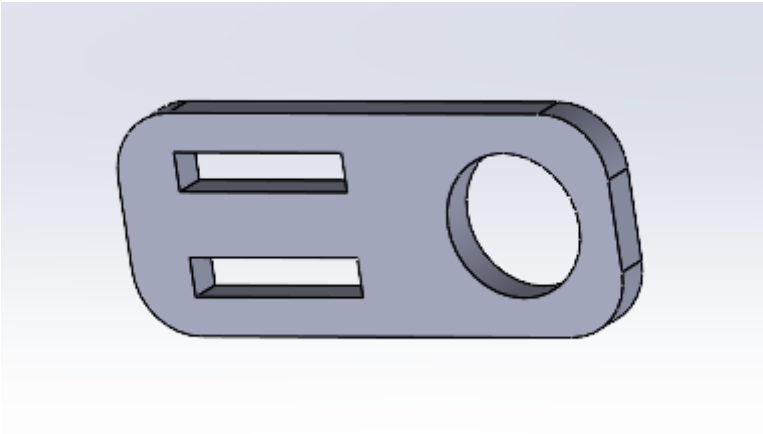


Imagen 35. Diseño sujeción paralelogramo-efector

Como se puede apreciar en la *Imagen 35*, la pieza presenta tres cavidades. Una con forma cilíndrica para albergar el extremo cilíndrico del paralelogramo y dos rectangulares para anclarse junto a las dos superficies que forman el efector final.

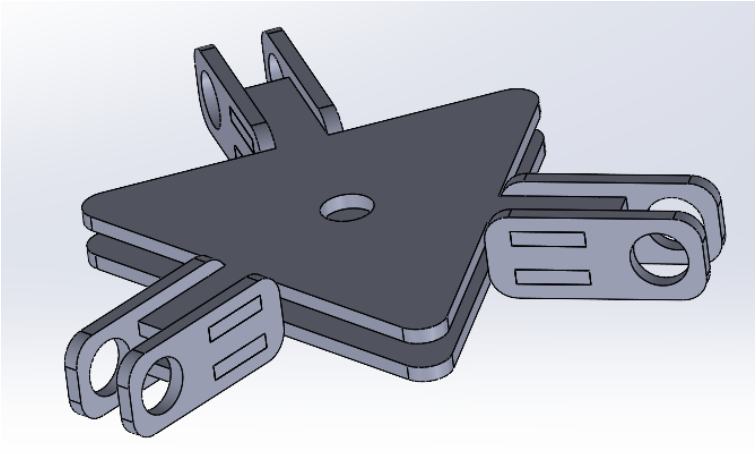


Imagen 36. Efector final Robot Delta

El efector final se podría haber realizado de forma que solo hiciera falta emplear una pieza de sujeción para cada cadena cinemática que constituye el robot, sin embargo, se ha decidido implementar un planteamiento similar al del brazo, añadiendo una segunda pieza y manteniendo una separación de 1 cm para evitar oscilaciones, dejando un primer diseño como el que se puede ver en la *Imagen 36*.

La *Imagen 37* presenta el diseño completo del producto que se implementa en este proyecto, una vez unidos los diferentes diseños elaborados a lo largo de este apartado.

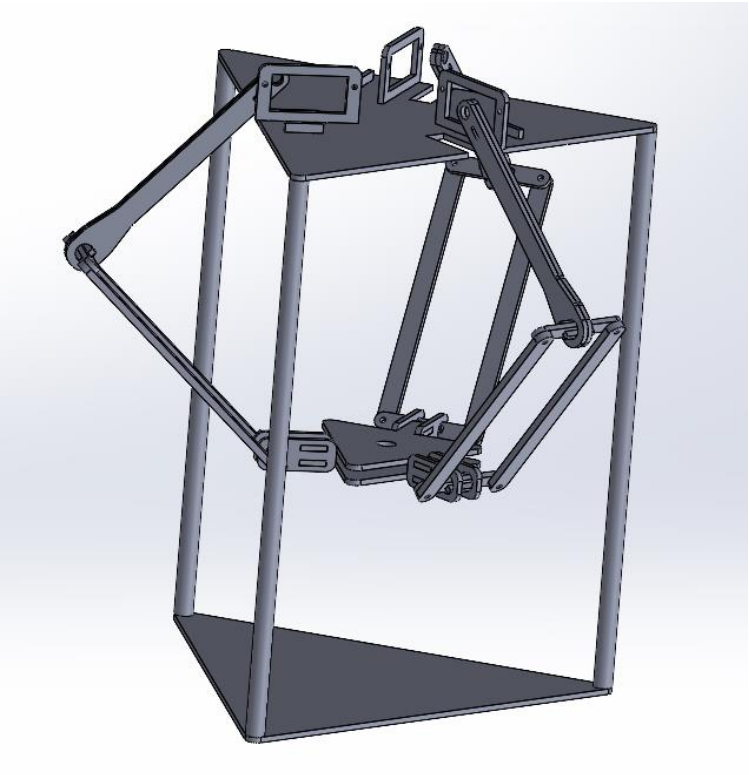


Imagen 37. Primer diseño de ensamblaje Robot Delta

4.2.3 Montaje físico y diseño final

En este apartado se redactan los resultados obtenidos de la elaboración física del producto de este proyecto. Dado que la utilización de los servomotores MG90s puede comprometer la estabilidad y correcto funcionamiento de los diferentes procesos del robot debido a su par, se ha decidido realizar el montaje empleándolos, ya que la otra opción no genera las incertidumbres mencionadas anteriormente (servomotores MG996R).

Cabe destacar que la posibilidad de emplear ambos servomotores se explica en el apartado de cálculos más adelante.

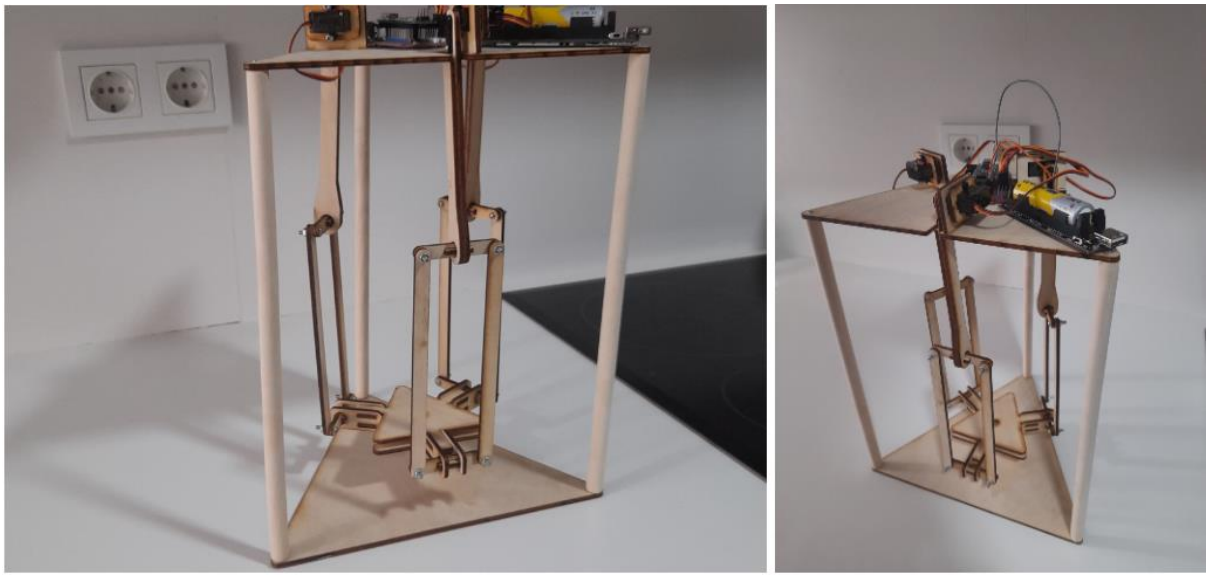


Imagen 38. Primer montaje Robot Delta

En este primer montaje (*Imagen 38)*, surgieron varios errores los cuales se van a citar a continuación, junto a su solución.

El primer error se detectó al realizar el montaje. Las piezas que van encajadas con otras no entraban con facilidad, ya que el área de la superficie saliente con el área del hueco donde encajaban era exactamente la misma. Para realizar el montaje fue necesario lijar ligeramente algunos de los componentes del robot. La medida tomada fue aumentar una décima de milímetro el hueco donde encajaban las piezas. Con eso, más el pequeño error que puede hacer la cortadora (centésimas de milímetro), las piezas encajarían perfectamente y se quedarían fijas.

Otro problema importante fue que, a pesar de estrechar lo máximo permitido (para que no se viera comprometida la dinámica del robot) la unión entre el lado corto del paralelogramo y la muñeca del efector final, aparecía una pequeña holgura, la cual, sumada a las holguras del resto de las juntas, perjudicaba la estabilidad y precisión del efector final.

Este problema era grave pues perjudicaba al correcto funcionamiento del robot. Se plantearon diferentes opciones como la de aumentar el grosor del brazo del robot para tener mayor superficie de apoyo y evitar dicha oscilación, pero dadas las propiedades de la madera, reducir la holgura entre esas dos piezas suponía comprometer la eficiencia de la junta de revolución.

Imagen 39. Rodamiento metálico

Por ello, la alternativa que se escogió fue emplear rodamientos metálicos de diámetro exterior de 15 mm e interior de 10 mm (*Imagen 39*), cuya superficie exterior va fijada al brazo del robot y la interior va fijada al eje en cruz que constituye el lado corto del paralelogramo. Así se consigue reducir prácticamente a cero la holgura entre las uniones, evitando el error de oscilación producido. Cabe destacar que solo se emplean rodamientos en el codo y muñeca del Robot Delta, en el resto de las juntas de revolución no es necesario.

Una vez recalculados todos los parámetros del robot con las modificaciones realizadas, se llega a la conclusión de que emplear rodamientos reduce ligeramente la capacidad de peso de la herramienta que puede manipular el efector final, pero mejora considerablemente la estabilidad de este, obteniendo un resultado tal y como se muestra en la *Imagen 40*:

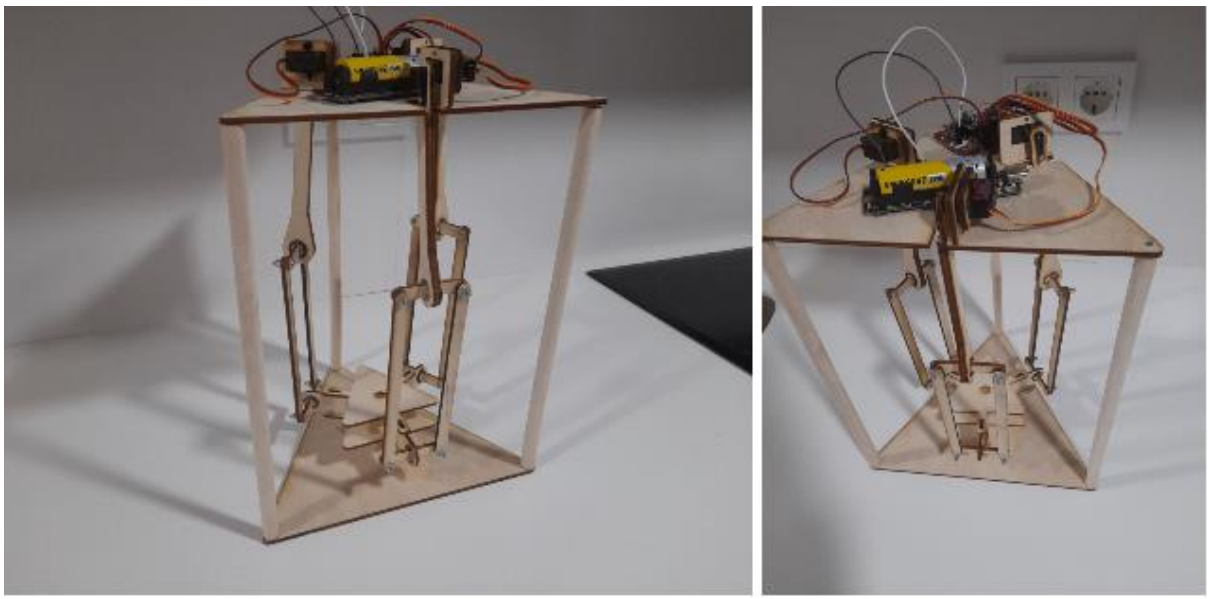


Imagen 40. Montaje final Robot Delta

Como se puede observar, dada la implementación de los rodamientos, ha sido necesario rediseñar algunos de los componentes del robot, como el efector final, y el lado corto del paralelogramo.

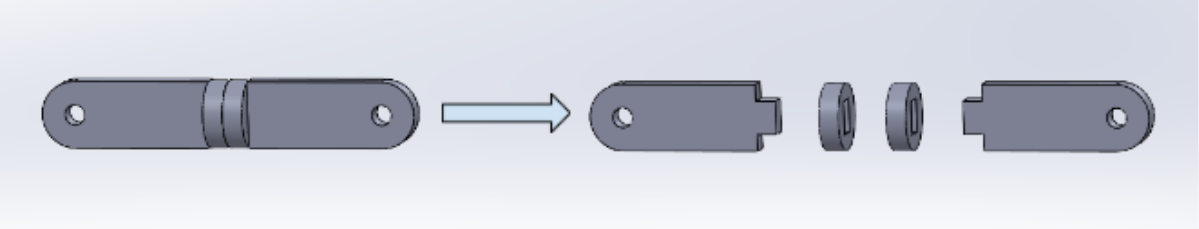


Imagen 41. Eslabón corto paralelogramo diseño final

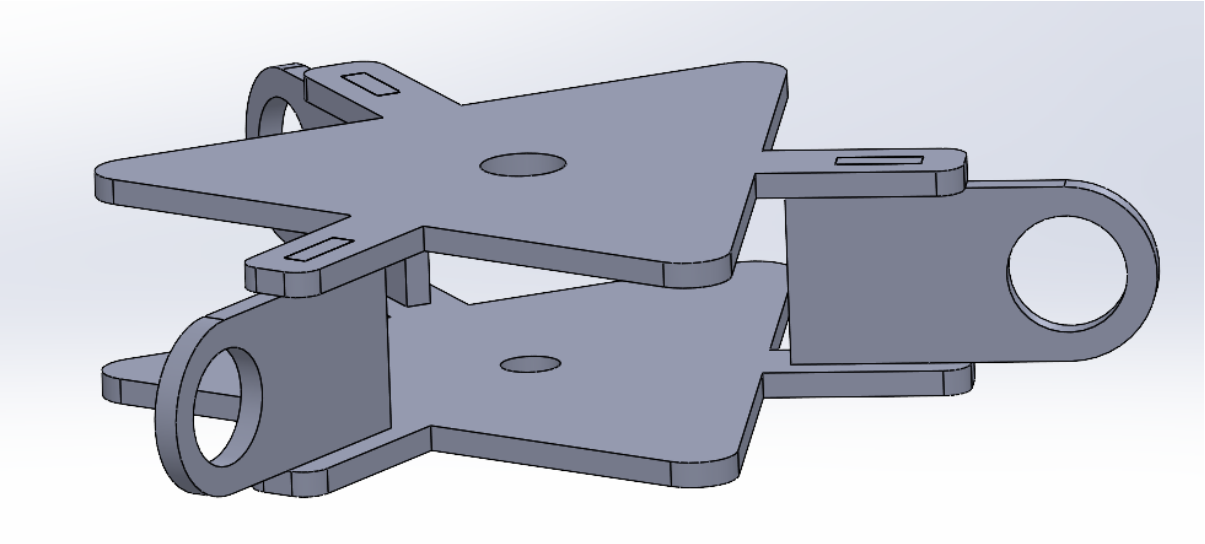


Imagen 42. Efector final diseño final

Con las modificaciones realizadas anteriormente se consigue obtener una mayor estabilidad en el efector final. No obstante, al alejar el efector final del centro de la base del robot (dando valores a Z e Y), si dicha distancia es demasiado grande, las plataformas que lo forman giran ligeramente, al igual que en el montaje anterior, pero en menor grado. Dada la finalidad de este proyecto, donde se prioriza el aprendizaje sobre la precisión, el resultado final obtenido es satisfactorio, finalizando así el diseño final del producto.

4.3 Simulación

A continuación, una vez se tiene el diseño final de todas las piezas que constituyen el robot junto a un diseño de su ensamblaje, se procede a realizar su estudio dentro del software de CoppeliaSim, con el fin de comprobar si el planteamiento es cinemáticamente factible.

4.3.1 Creación Robot Delta en CoppeliaSim

El primer paso a llevar a cabo es transformar el formato de los documentos creados en la plataforma SolidWorks para emplear uno en el cual se puede trabajar con CoppeliaSim. Se pasan los documentos de formato ".SLDPRT" a formato ".STL", que convierte las piezas en una malla tridimensional formada por la unión de un gran número de figuras triangulares.

Para el correcto y fluido funcionamiento del CoppeliaSim es recomendable emplear el programa Meshmixer, donde importando las piezas en formato ".STL" y realizando la

operación de *Remesh,* se disminuye, en caso de ser necesario, el número de triángulos que constituye la malla. El fin es obtener un elemento más cómodo y menos pesado de procesar.

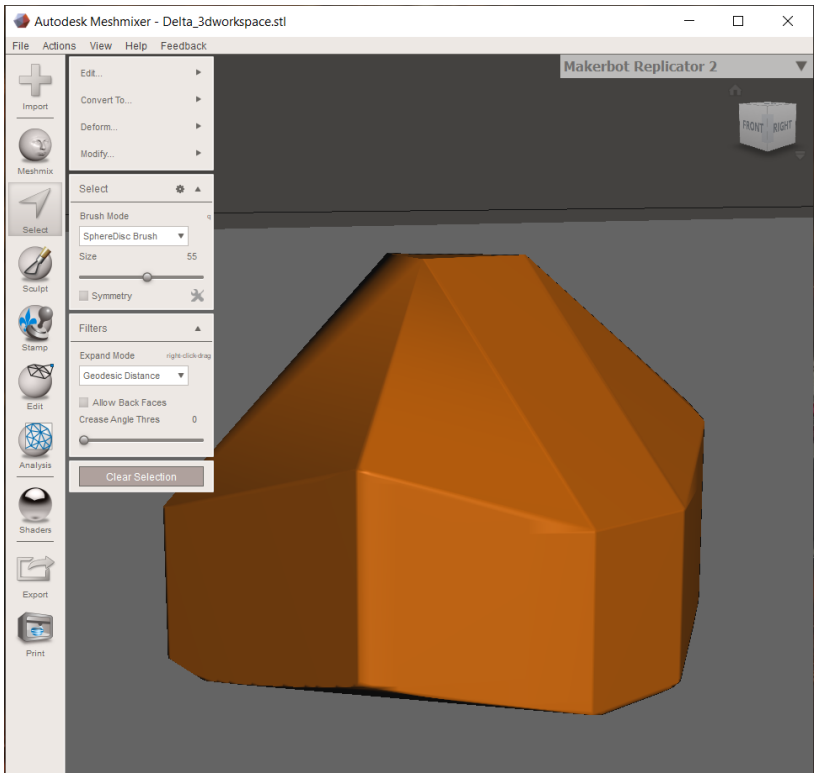


Imagen 43. Interfaz usuario Meshmixer

Realizar este proceso hace que baje el nivel de detalle de la pieza, pero dado que se le va a dar un uso de simulación, las mínimas modificaciones que se puedan crear durante este proceso son despreciables.

Una vez se tienen todas las piezas en el formato correcto y se han adaptado a una forma menos compleja, se importan a CoppeliaSim para poder trabajar con ellas.

4.3.2 Jerarquía de ensamblaje Robot Delta en CoppeliaSim

Para la simulación del robot no hace falta emplear todas las piezas que lo forman, únicamente aquellas que vayan a afectar a su movilidad o funcionamiento. Por ejemplo, las piezas cuyo propósito es el de mantener fijo los micro servomotores, las piezas que forman el lado corto del paralelogramo y la doble pieza del brazo, son prescindibles. Así se obtiene un espacio de trabajo tal y como se ve en la *Imagen 44*.

Aunque anteriormente se haya especificado que no se emplean eslabones cilíndricos para los ejes de revolución, dado que esto es una simulación cinemática del robot y realizan la misma función ambas alternativas, estos se pueden utilizar en este apartado consiguiendo una manipulación más cómoda e intuitiva.

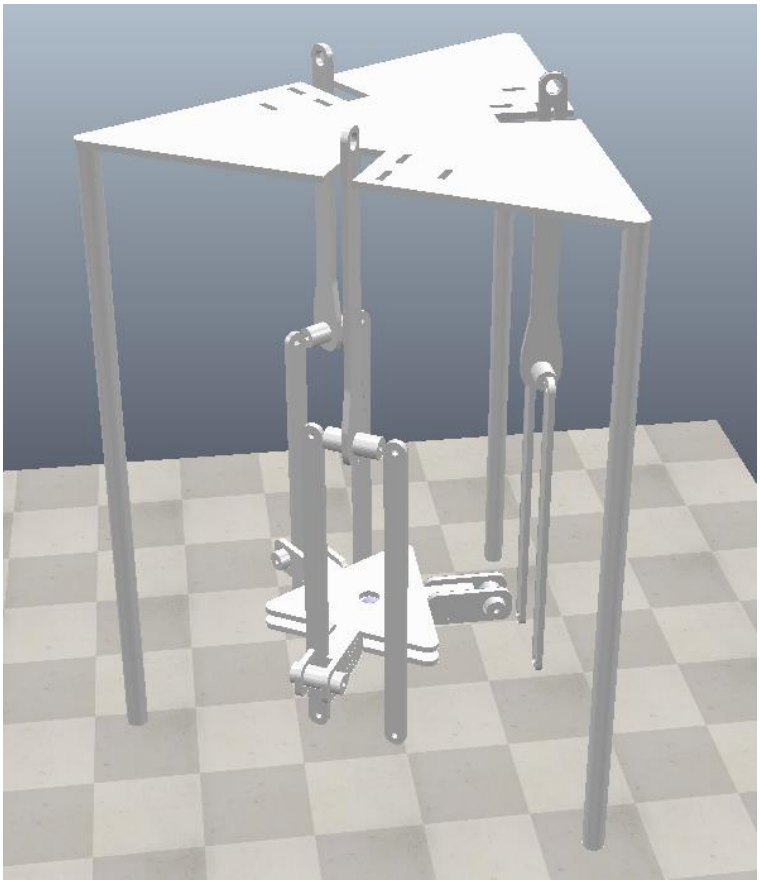


Imagen 44. Conjunto formas empleadas en CoppeliaSim

A pesar de que los pilares del robot son piezas fijas ancladas a su base, es necesario emplearlas ya que pueden implicar colisión con el efector final, modificando así su espacio de trabajo.

Posteriormente, se deben declarar y posicionar las diferentes juntas que forman el producto, para darle la movilidad con la que supuestamente ha sido diseñado. Analizando los brazos de forma individual, cada uno está formado por siete juntas de revolución. La primera de ellas es la que se encarga de mover toda la cadena cinemática, por lo que debe coincidir con el eje del actuador y situarla coherentemente dentro del espacio de estudio, aunque esta tenga libertad de posicionamiento a lo largo del eje donde se encuentra.

Las juntas de revolución restantes, se sitúan cada una en el eje céntrico de las cavidades cilíndricas que constituyen el paralelogramo y el efector final.

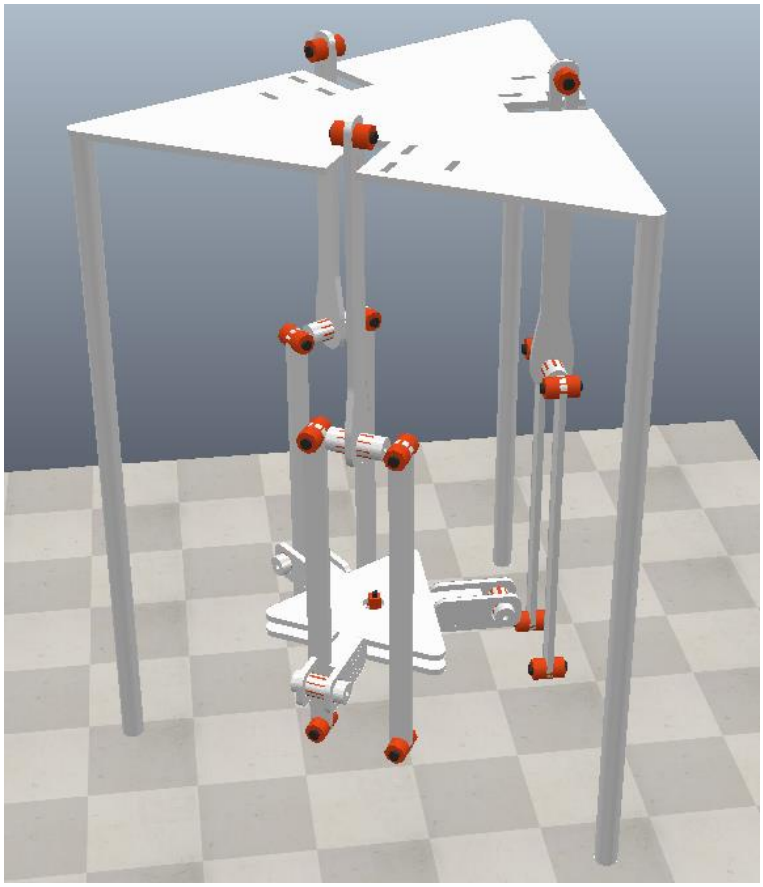


Imagen 45. Posicionamiento juntas Robot Delta en CoppeliaSim

Siguiendo con el proceso de ensamblaje, para cerrar la cadena cinemática de cada brazo, es necesario emplear “dummies”. La función de los “dummies” consiste en añadir restricciones a la cinemática del robot para que se simulen correctamente sus movimientos. Puesto que en CoppeliaSim el ensamblaje de las piezas y juntas debe hacerse de forma escalonada dando mayor prioridad a los elementos que se encuentren por encima de él (método de jerarquía), el hecho de que una pieza o eslabón se emplee como nexo final de diferentes piezas es algo que directamente no se puede interpretar en el programa.

En el caso de este proyecto, el efector final es la pieza común que cierra la cadena cinemática, lo que se puede entender como un nexo entre los tres brazos. Por ello, se deben crear seis pares de “dummies”, dos para cada brazo.

Uno de los “dummies” de cada pareja deberá situarse en la posición exacta donde finaliza la última junta que forma el paralelogramo, mientras que el otro deberá situarse en la posición donde se quiere ubicar la junta de la muñeca del robot en el efector final.

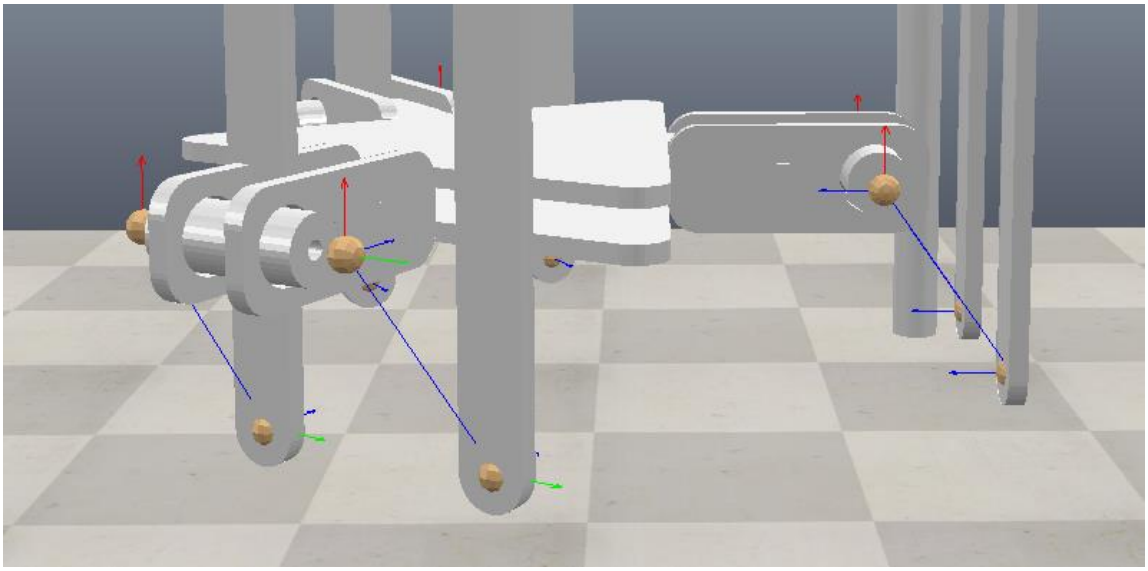


Imagen 46. Colocación dummies efector final en CoppeliaSim

Cabe destacar que es importante que cada pareja esté orientada de la misma forma respecto al mundo, ya que al empezar la simulación los dummies se sitúan en la posición y orientación del dummy de referencia.

Una vez se hayan realizado los pasos anteriores, se dispone la jerarquización de los componentes.

La pieza que está en el nivel más alto es la Base, de donde posteriormente surgen Pilar1, Pilar2 y Pilar3, que están fijados a ella. También se anclan a Base las juntas de revolución de cada brazo, de donde sale la pieza Brazo. A esta se le ancla una junta de revolución y a continuación se fija uno de los dos lados cortos del paralelogramo, para posteriormente seguir de forma independiente con los dos eslabones largos que finalizan con su respectiva junta de revolución.

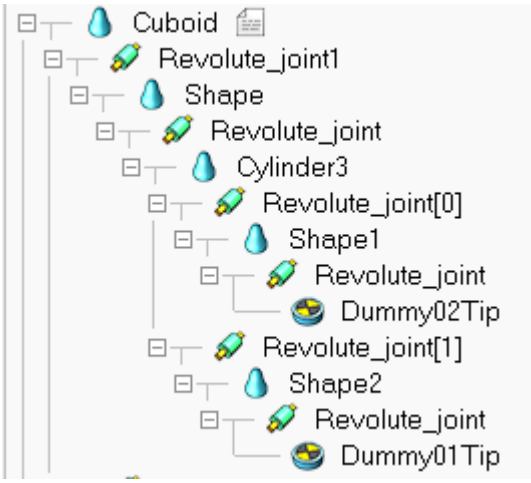


Imagen 47. Jerarquía brazo robot

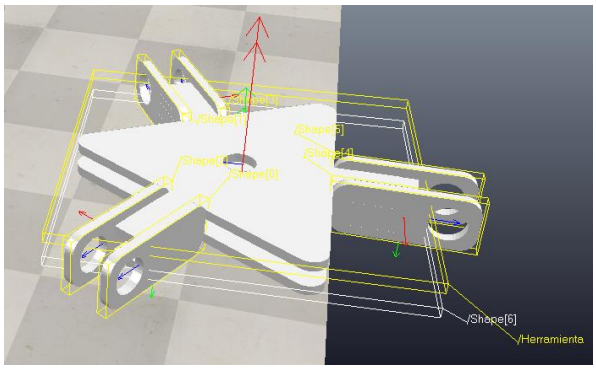


Imagen 48. Conjunto piezas efector final

Importante resaltar la obtención del efector final como un único elemento, ya que, a la hora de realizar el montaje, se agruparon el conjunto de piezas que lo componen, se seleccionaron y se fusionan para formar una pieza unitaria, como se puede apreciar en la *Imagen 48*.

Para finalizar con la jerarquización del robot, al eslabón que se ha creado como efector final, se le añaden las tres juntas de revolución restantes (una para cada brazo) y el otro par de dummies por brazo restantes.

Una vez se ha completado la jerarquización del robot hay que configurar las características dinámicas tanto de las piezas como de sus juntas. Dado que esta simulación tiene fines teóricos y no prácticos, ni las juntas ni las piezas tienen propiedades dinámicas y será necesario poner todas las juntas en un modo de funcionamiento pasivo.

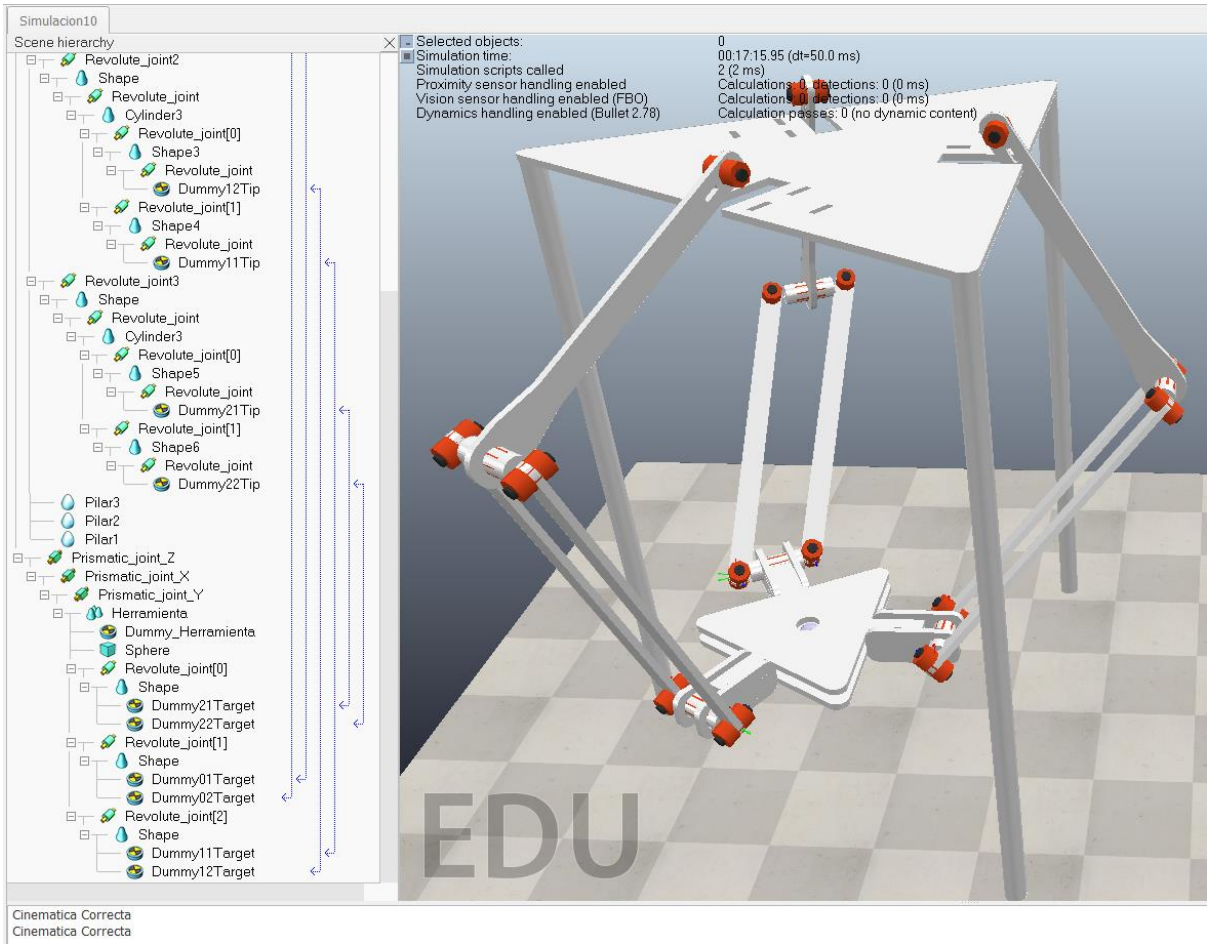


Imagen 49. Simulación cinemática Robot Delta

Finalizada la jerarquización y configuración del robot y viendo la simulación obtenida (*Imagen 49*), moviendo libremente el eslabón donde iría situada la herramienta, se puede confirmar que la cinemática del diseño realizado es perfectamente válida, por lo que se podría continuar realizando los cálculos pertinentes con el diseño establecido.

Cabe destacar que algunos de los elementos que aparecen en la jerarquía del efector final no han sido mencionados. Se explicará su funcionalidad en los siguientes apartados del proyecto.

4.3.3 Volumen trabajo

En este apartado se explica el procedimiento a emplear para calcular el volumen de trabajo del robot. Para ello se utilizan CoppeliaSim, Matlab y Meshmixer.

La base para obtener el espacio de trabajo es el archivo de simulación ya creado, en el que realizando pequeñas modificaciones y aportando más contenido y programación, se puede realizar el primer paso que consiste en un barrido de puntos para comprobar si el efector final se puede situar en cada punto.

Se comienza añadiendo tres juntas prismáticas en las que cada una tiene la dirección de uno de los ejes sobre el mundo en el que se está trabajando. Se colocan de forma escalonada entre ellas (el orden es independiente), para finalmente anclar la herramienta a la última junta.

También se debe añadir una pequeña esfera donde supuestamente iría la herramienta del robot, y en esa misma posición un dummie, dejándolo tal y como se muestra en la *Imagen 50*.

A continuación, hace falta crear un script dentro de la base fija del robot (ya que desde ese componente se ramifica el resto de elementos que lo componen), para elaborar un código que se encargue de realizar un barrido de puntos y crear un fichero con los puntos válidos del mismo.

Imagen 50. Jerarquía herramienta

La extensión completa del código está adjunta en el Documento 6 de este proyecto. No obstante, para la correcta explicación del proceso de este apartado se aportan fragmentos de texto que permiten poder avanzar junto a su explicación.

Primero se han obtenido todos los nombres de los objetos que son necesarios para guardarlos en parámetros empleando el comando “sim.getObject” (*Imagen 51*) y el comando “io.open”. Se crea un fichero el cual por el momento está vacío. Para este apartado, en lugar de emparejar los dummies desde su propia configuración, su restricción de posicionamiento se realiza empleando código, el cual se puede obtener en la página oficial de CoppeliaSim.

Gracias a emplear este código, se crean funciones “if” para comprobar que ha sido posible realizar dicha restricción y, en caso contrario, mostrar un aviso de error en la ventana de notificaciones del programa.

```
jointHandles[1]=sim.getObject('/Prismatic_joint_X')
jointHandles[2]=sim.getObject('/Prismatic_joint_Y')
jointHandles[3]=sim.getObject('/Prismatic_joint_Z')

Objt1=sim.getObject('/Shape1')
Objt2=sim.getObject('/Shape2')
Objt3=sim.getObject('/Shape3')
Objt4=sim.getObject('/Shape4')
Objt5=sim.getObject('/Shape5')
Objt6=sim.getObject('/Shape6')

Objt7=sim.getObject('/Pilar1')
Objt8=sim.getObject('/Pilar2')
Objt9=sim.getObject('/Pilar3')

Objt10=sim.getObject('/Herramienta')
```

Imagen 51. Obtención elementos simulación en CoppeliaSim

También es necesario crear una función que devuelva el barrido de puntos que se emplea para el estudio del efector final. Este volumen de puntos se limita por el rango de las juntas prismáticas que se han añadido previamente al efector final.

Obtenido el conjunto de puntos donde se quiere estudiar el estado de la herramienta, con el comando “set.JointPosition”, esta se va desplazando libremente hasta llegar al último punto obtenido, el cual se deduce cuando el contador de posiciones realizadas (inicializado a 1), llega al máximo.

```
if (counterQHer<maxPositions) then
    for j=1,N,1 do
        sim.setJointPosition(jointHandles[j],QHer[counterQHer][j])
    end
```

Imagen 52. Bucle puntos de posicionamiento

Importante realizar una serie de comprobaciones para cada punto donde se va desplazando. Hay que asegurarse de que la cinemática de los tres brazos se ha resuelto con éxito y que no existe ningún tipo de colisión entre el efector final y los distintos componentes que constituyen el robot (*Imagen 53*).

```
p18=sim.checkCollision(Objt1,Objt8)
p48=sim.checkCollision(Objt4,Objt8)
p39=sim.checkCollision(Objt3,Objt9)
p59=sim.checkCollision(Objt5,Objt9)
p67=sim.checkCollision(Objt6,Objt7)
p27=sim.checkCollision(Objt2,Objt7)
p810=sim.checkCollision(Objt8,Objt10)
p910=sim.checkCollision(Objt9,Objt10)
p710=sim.checkCollision(Objt7,Objt10)
```

Imagen 53. Comprobación colisión robot

Finalmente, si el punto que se estudia cumple con todos los requerimientos, este se guarda en el fichero previamente creado, obteniendo así el conjunto de puntos físicamente alcanzables del robot.

Una vez se obtiene el fichero con todos los puntos guardados de la simulación, se pasa a Matlab, para crear una malla que posteriormente se añadirá como un elemento más en el espacio de simulación.

Dado que su código es muy escueto y simple, únicamente hace falta comentar que los pasos que realiza este fichero matlab son crear una figura cuyo volumen es el espacio de trabajo del robot y, posteriormente, aplicar un proceso de vaciado para disponer únicamente la superficie de dicha figura, obteniendo un resultado como el que se aprecia en la *Imagen 54*.

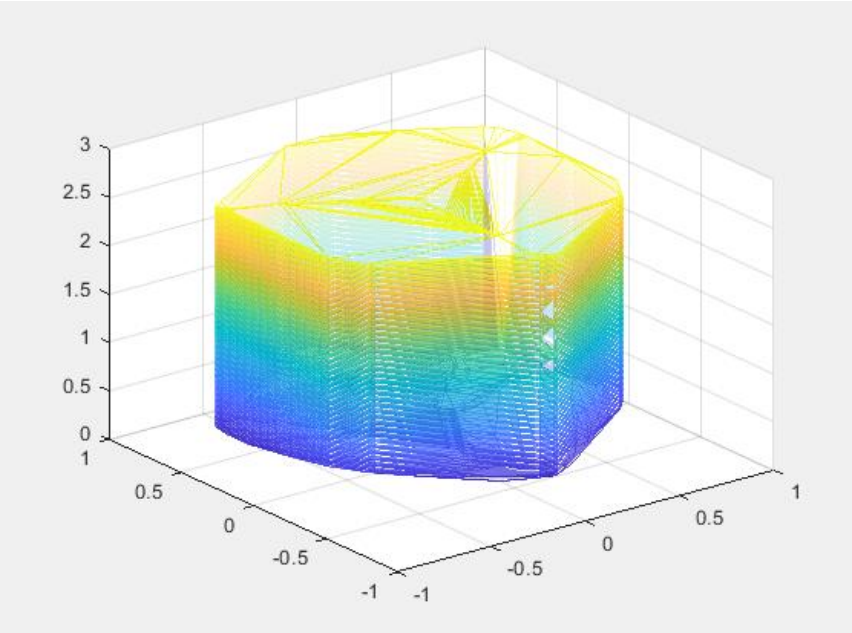


Imagen 54. Malla volumen de trabajo del robot

El resultado obtenido, al igual que todos los elementos que se han importado a CoppeliaSim, pasa por el software de Meshmixer, para reducir el número de triángulos que lo forman y poder emplearlo más cómodamente en la simulación.

Una vez se tiene la malla dentro del espacio de simulación se pueden cambiar sus propiedades para obtener una figura semitransparente. Además, añadiendo líneas de código, se consigue que cuando la esfera del efector final (la cual se ha creado previamente en el punto de la herramienta) detecte colisión con la malla, haga que esta cambie al color rojo para indicar que la posición que se quiere alcanzar está al límite de las capacidades del robot (*Imagen 55*).

Con todo esto, se da por concluido el estudio de su volumen de trabajo.

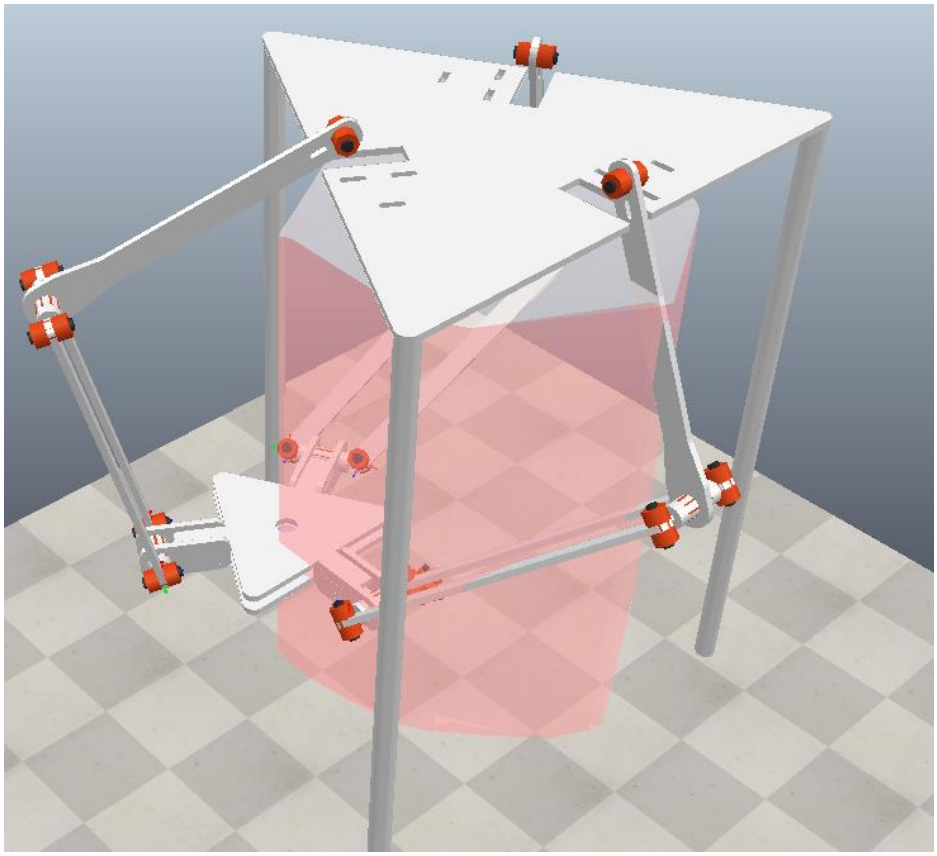


Imagen 55. Ejemplo límite de posición del volumen de trabajo

5 CÁLCULOS

Una vez se ha confirmado la realización del diseño del proyecto a través de su simulación, se procede a elaborar los cálculos pertinentes tanto para seguir comprobando su correcto funcionamiento, como para obtener propiedades y características del producto diseñado.

Los cálculos a realizar servirán para obtener diferentes aspectos relevantes como la cinemática del robot, su volumen de trabajo y la comprobación de la correcta elección de los actuadores para el producto.

5.1 Cinemática inversa

En robótica, la cinemática inversa es la técnica que permite determinar el movimiento de una cadena de articulaciones para conseguir que un actuador final se ubique en una posición concreta, es decir, consiste en calcular las transformaciones necesarias en las articulaciones de una estructura (para este proyecto los parámetros a calcular son las posiciones de los actuadores), de modo que su extremo se coloque en una posición determinada. Obtener las ecuaciones de este estudio es fundamental para poder implementar trayectorias o posicionamientos dentro del volumen de trabajo del robot.

Primero se debe comprobar la movilidad del robot (el número de grados de libertad), usando la ecuación de movilidad espacial de Kutzbach:

$$M = 6(N-1) - 5J_1 - 4J_2 - 3J_3 \quad (3)$$

Donde:

M es la movilidad o número de grados de libertad
N es el número total de eslabones, incluyendo tierra
J1 es el número de juntas de revolución o prismática
J2 es el número de juntas universales
J3 es el número de juntas esféricas

Para el diseño del robot realizado:

N=17
J1=21
J2=0
J3=0

$$M = 6(17-1) - 5 * 21 = -9 \quad (4)$$

Como sucede a menudo, la ecuación de Kutzbach falla porque el resultado obviamente debe ser 3 grados de libertad. Este resultado predice que Delta es una estructura estáticamente indeterminada y severamente restringida, lo cual es incorrecto.

La ecuación de Kutzbach no comprende la geometría especial; en el caso del Robot Delta, hay tres mecanismos paralelos de cuatro barras. El robot completo trabajaría cinemáticamente de manera idéntica al Robot Delta original si quitamos uno de los largos enlaces paralelos del mecanismo de cuatro barras, junto con dos articulaciones giratorias cada uno. En este caso equivalente, la ecuación de Kutzbach es:

N=14
J1=15
J2=0
J3=0

$$M = 6(14-1) - 5 * 15 = 3 \quad (5)$$

Como se muestra en la ecuación (5), así se consigue obtener los grados de libertad deseados.

Como ya se ha especificado, las tres cadenas cinemáticas están unidas entre sí mediante el efector final, por lo que los ángulos de los servomotores están relacionados entre ellos. Esto es debido a que en los robots paralelos las cadenas cinemáticas son cerradas en un mismo punto, creando dependencia entre ellas.

El primer paso consiste en hallar la matriz de Denavit-Hartenberg, para describir la estructura cinemática de una cadena articulada constituida por articulaciones con un solo grado de libertad.

Para hallar dicha matriz, hay que situar los ejes de cada una de las juntas que forman el brazo del robot, sin contar con la base y el efector final de este.

Del Eje 1 se obtiene el valor de q1, que corresponde a la posición del servomotor. Los Ejes 2 y 3 comparten ubicación ya que representan las dos juntas de revolución entre la unión del brazo y del paralelogramo, donde una tiene el eje z perpendicular a la hoja saliendo de él, y otro perpendicular a este último eje y al eje x2.

Finalmente, se sitúa el Eje 4 en el punto de unión entre el paralelogramo y el efector final, compartiendo la misma orientación que el eje anterior.

Una vez planteados los ejes de referencia del brazo, se emplea la herramienta Matlab (versión 2020b), para realizar todos los cálculos y despejar y simplificar las ecuaciones requeridas.

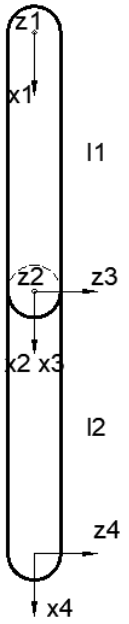


Imagen 56. Esquema 1

Declarando la matriz de transformación DH y calculando la transformación necesaria entre el Eje 1 y el Eje 4, se obtienen las ecuaciones de las coordenadas x y z, en función de los valores de q1, q2, q3 y los parámetros l1 y l2 del robot. Dado que las matrices obtenidas tienen una gran envergadura como para mostrarlas en este documento, únicamente se redactan los procedimientos empleados, sin mostrar los resultados obtenidos.

El valor q3 se puede despejar directamente ya que únicamente depende del parámetro z y del l2:

$$q3 = asin(-z/l2) \quad (6)$$

Las ecuaciones x e y son dependientes de los parámetros q1 y q2. Por ello, para despejar una ecuación donde se quiera obtener el valor de q1 en función de los parámetros x e y, se emplea el comando *solve* donde se declaran ambas ecuaciones resultantes de la matriz DH anterior y se despeja en función de q1 y q2.

$$eq1 = l2 * cos(q1 + q2) * cq3 + l1 * cos(q1) == x; \quad (7)$$
$$eq2 = l2 * sin(q1 + q2) * cq3 + l1 * sin(q1) == y; \quad (8)$$
$$S = solve([eq1\ eq2], [q1\ q2]) \quad (9)$$

De la función anterior se obtiene que para q1 (que es el parámetro que interesa), hay dos posibles soluciones, esto es debido a la posición del hombro que puede ser positiva o negativa. Se escoge el valor que corresponda con el rango de movilidad del servomotor.

q1 = 2*atan((2*l1*y + (- cq3^4*l2^4 + 2*cq3^2*l1^2*l2^2 + 2*cq3^2*l2^2*x^2 + 2*cq3^2*l2^2*y^2 - l1^4 + 2*l1^2*x^2 + 2*l1^2*y^2 - x^4 - 2*x^2*y^2 - y^4)^(1/2)) / (- cq3^2*l2^2 + l1^2 + 2*l1*x + x^2 + y^2));

Una vez despejada la ecuación de q1, hay que calcular los valores de x', y', z', para cada uno de los Ejes 1 de cada brazo respecto el Eje 0 que se encuentra en el centro de la base del robot.

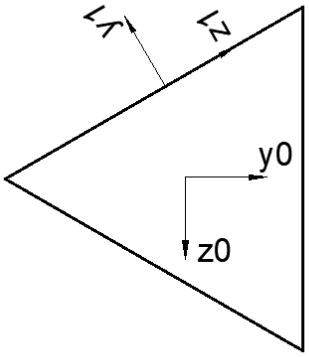


Imagen 57. Esquema 2

Como se observa en la *Imagen 57*, es necesario rotar 0º, 120º y -120º (o 240º), para orientar el Eje 0 a cada una de las posiciones de los ejes de los actuadores, desplazándose posteriormente en el eje y, obteniendo las siguientes ecuaciones:

$$x' = x \quad (10)$$
$$y' = y * cos(alpha) - z * sin(alpha) + l0 - l3 \quad (11)$$
$$z' = y * sin(alpha) + z * cos(alpha) \quad (12)$$

Donde l0 y l3 son la distancia entre el Eje 0 y el eje del actuador y la distancia entre el centro del efector final y el eje de la junta con el paralelogramo, respectivamente, siendo alpha el valor del ángulo a rotar respecto cada brazo (0º, 120º o -120º).

Cabe destacar que en la ecuación (11) se resta l3 para obtener el punto central de la herramienta, ya que con el planteamiento empleado del Esquema 1 de la *Imagen 56*, se obtenía el punto de la junta.

Empleando este desarrollo se obtiene la cinemática inversa del robot para el diseño desarrollado anteriormente.

5.2 Par servomotores

A continuación, una vez se ha establecido el diseño final del robot y se ha confirmado que es un diseño apto, hay que realizar los cálculos pertinentes para corroborar la correcta elección de los servomotores. Para ello, empleando la herramienta de SolidWorks con la que se han

creado previamente todas las piezas, se obtiene el volumen de cada una y junto a la densidad del material se adquiere el peso de la pieza.

Aunque el peso esté repartido entre los tres servomotores, el cálculo se realizará para que un solo servo soporte el total de los tres brazos, con el objetivo de ponerse en la peor situación y asegurar así un correcto funcionamiento.

La densidad de la madera mdf es de 450 Kg/cm^3, lo que equivale a 0.450 Kg/cm^3.

Brazo: $5.46\ cm^3 * 0.450 * 6 = 14.742\ g$ (16)
Codo: $(1.37\ cm^3 + 0.51\ cm^3) * 0.450 * 6 = 5.076\ g$ (17)
Paralelogramo: $6.26\ cm^3 * 0.45 * 6 = 16.902\ g$ (18)
Muñeca: $1.53\ cm^3 * 0.45 * 6 = 4.131\ g$ (19)
Herramienta: $14.25\ cm^3 * 0.45 * 2 = 12.825\ g$ (20)

Total = 53.676 g

Añadiendo el peso de tornillos, arandelas y tuercas, el peso se aproxima a los 80 g, por lo que para redondear al alza se tomará dicho valor.

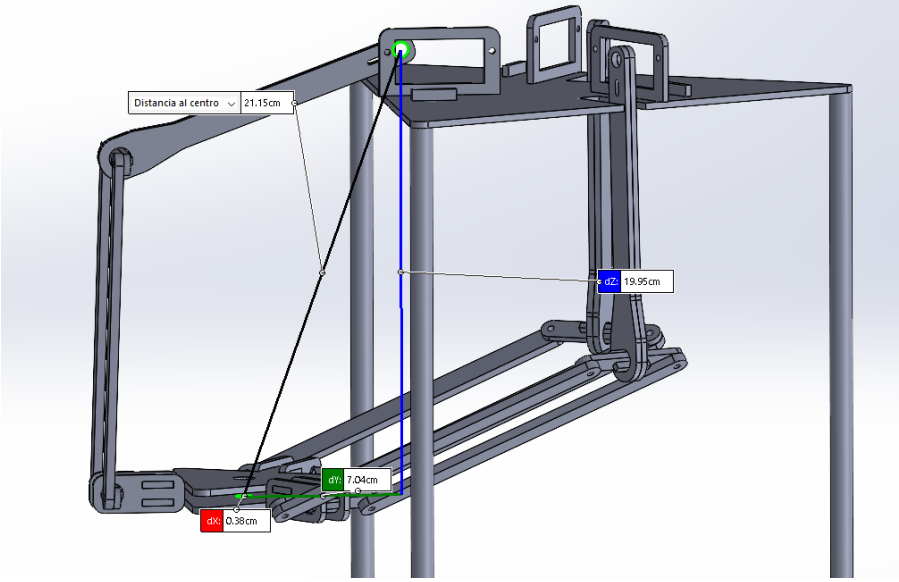


Imagen 58. Punto más lejos del eje del actuador al eje de la herramienta

Sabiendo que la distancia máxima entre el centro del efector final y del eje del actuador es de 21.15 cm, se puede calcular el par necesario de los actuadores:

$$par = 0.08\ Kg * 21.15\ cm = 1.692\ Kg/cm \quad (21)$$

Como se puede apreciar, el par obtenido está muy por debajo del par de los actuadores que se han planteado inicialmente (MG996r con un par mínimo de 9 Kg/cm), por lo que el robot funciona perfectamente. No obstante, se puede apreciar que también es aceptable el uso de los micro servomotores MG90s, los cuales, con la fuente de alimentación planteada, tienen un par de 1.8 Kg/cm.

Ambos servomotores son aptos para este proyecto, por ello, la decisión final la tendrá el usuario, teniendo en cuenta las propiedades y características que aportan cada uno. Ambas opciones no se encuentran muy lejos en el coste económico y se considera que dar elección a los usuarios aumenta la calidad del producto.

Emplear el servomotor MG996r conlleva una disminución del tiempo de trabajo del robot pues esto motores, al tener una gran fuerza, consumen una cantidad muy elevada de energía. Sin embargo, se obtiene mayor rentabilidad en los procesos del robot, dada su mayor estabilidad. Además, permite perfectamente el uso de herramientas en el efector final con el fin de seguir completando este proyecto.

Imagen 59. Servomotor MG90s

En caso de no querer emplear herramientas y buscando una productividad del robot más académica y teórica, la elección del servomotor MG90s es perfectamente apto, consiguiendo a su vez un aumento en la duración de la batería.

Cabe destacar que en el Documento 4 se adjuntan los planos de las piezas que se encarguen de la correcta sujeción de este último servomotor en la base ya diseñada.

6 PROGRAMACIÓN ROBOT DELTA

Como se ha dicho anteriormente, el objetivo de este proyecto es darle un uso académico al producto, por ello, su parte de programación, al igual que el cálculo de la cinemática inversa, debería ser desarrollada por el usuario, no obstante, en este proyecto se implementa un código base con el que poder realizar movimientos básicos empleando coordenadas en x, y, z, para desplazar el efector final dentro del volumen de trabajo del robot.

Dicho código se entiende como una demo donde se muestran las posibilidades del proyecto. Para este caso, la programación realizada tiene la función de desplazar el efector final del brazo a cuatro puntos diferentes, partiendo de una posición de reposo la cual establece el usuario a voluntad, y una vez se haya finalizado el cumplimiento del posicionamiento de todos los puntos, se vuelve a su posición de reposo inicial.

En este apartado no se detalla cada línea de código que constituye la demo, únicamente se explica la utilidad de las diferentes funciones declaradas.

6.1 Declarar actuadores

El primer paso a realizar es declarar el número de actuadores que se emplean, asignar a qué pin van ligados y si tienen offset, y determinar la posición máxima y mínima en la que el actuador deba trabajar.

```
Servo servos[12];

typedef struct
{
  uint8_t pin;
  int offset;
  int min_pos;
  int max_pos;
} RobotServo_t;

#define JOINTS 3

RobotServo_t robotServos[JOINTS] = {{4, 0, 0, 180}, {5, 0, 0, 180}, {6, 0, 0, 180}};
```

Dado que para este diseño se trabaja con el rango completo de los servomotores escogidos, y ninguno de ellos requieren de offset, se crea la definición de estructura que se muestra anteriormente.

6.2 Función “writeServo”

Esta función se encarga de asignar la posición de ángulo determinada para el servomotor deseado, por ello, es necesario darle a esta función la definición de estructura (explicada anteriormente), del servo que se quiera emplear, y el valor del ángulo.

```
void writeServo(const RobotServo_t&servo, int angle)
{
  angle = constrain(angle + servo.offset, servo.min_pos, servo.max_pos);
  servos[servo.pin].attach(servo.pin);
  servos[servo.pin].write(angle);
}
```

Dado que no es necesario que devuelva ningún valor, esta función es de tipo “void”, donde empleando el valor del ángulo y los datos de la estructura del servomotor escogido, posiciona el servomotor en la posición correcta.

6.3 Función “moveAbsJ”

Consiste en el control eje a eje con trayectoria polinomial. El robot debe moverse a una configuración articular determinada por los valores de referencia en un tiempo dado. En este modo, los ejes están coordinados, en el sentido de que cada uno de ellos trata de alcanzar una posición destino, pero todos al mismo tiempo.

Esta función tiene la finalidad de implementar una trayectoria polinomial y controlar la velocidad de traslación del efector final entre los diferentes valores de ángulos que se les dé a los actuadores. Para ello, es necesario darle a la función los datos de estructura de cada servomotor (ya que dentro de ella se hace llamar a la función “writeServo”), los valores de los ángulos iniciales y finales de los servomotores, y el tiempo en segundas que se quiere que dure el proceso.

```
void moveAbsJ(const RobotServo_t servos[JOINTS], const double q0[JOINTS], const double qT[JOINTS], const double T);
```

Con los anteriores datos y realizando los cálculos pertinentes, se consigo realizar el desplazamiento en el tiempo asignado.

6.4 Función “inverseKin”

Para poder asignar posiciones dentro de un espacio de coordenadas alcanzable para el robot, es necesario implementar esta función, donde empleando las ecuaciones desarrolladas en el apartado 5.1, se consigue obtener los valores de los ángulos para los puntos dados.

```
typedef struct
{
  double l0;
  double l1;
  double l2;
  double l3;
} RobotParams_t;

RobotParams_t params = {6, 13.45, 15, 6};

void inverseKin(double pT[], const RobotParams_t &params, double *q);
```

Esta función requiere de las dimensiones del robot, las cuales están definidas dentro de la estructura “RobtParams_t”, el vector punto que se quiere estudiar (el cual está formado por las coordenadas en x, y, z), y una variable donde se almacena los valores obtenidos.

Cabe destacar que para que dicha variable pueda emplearse fuera de la función, se le asigna delante un asterisco.

6.5 Función “moveJ”

Esta función se encarga de agrupar las dos funciones anteriores, donde aportando el punto de inicio y el punto final, junto al tiempo de traslación entre ellos, se consigue realizar el proceso correctamente.

```
void moveJ(double p0[], double pT[], float T)
{
  double qT[JOINTS], q0[JOINTS];

  inverseKin(p0, params, q0);
  inverseKin(pT, params, qT);
  moveAbsJ(robotServos, q0, qT, T);
}
```

Cabe destacar que la función “inverseKin” es necesaria llamarla dos veces para ambos puntos.

6.6 Función “moveL”

Esta función es una mejora de la función anterior, la cual no es estrictamente necesaria pero muy empleada en procesos industriales. Tiene la misma finalidad que la función anterior, que consiste en desplazar el efector final entre dos coordenadas en un tiempo determinado. No obstante, la mejora que se aplica en esta función es la de asegurar que el traslado entre ambos puntos sea totalmente lineal, algo que “moveJ” no puede garantizar, ya que únicamente se encarga de establecer la posición final de los actuadores, sin preocuparse del recorrido empleado.

Se plantea como un problema en el que el avance lineal se implementa con una trayectoria polinomial, mientras que la dirección de avance se determina por un vector unitario a partir de las posiciones de inicio y destino. De esta forma, el avance de la trayectoria tendrá una velocidad nula al principio y al final de la misma, mientras que en los puntos intermedios, la velocidad será mayor.

```
X=sqrt(pow(pT[0]-p0[0],2)+pow(pT[1]-p0[1],2)+pow(pT[2]-p0[2],2));
a=-(2*X)/(T*T*T);
b=(3*X)/(T*T);
t=0;

for(int k=0;k<3;k++)
  V[k]=(pT[k]-p0[k])/X;

 for(int j=0;j<T*50;j++)
 {
  df = millis();
   t=t+0.02;
   s=a*t*t*t+b*t*t;

 for(int k=0;k<3;k++)
  P[k]=s*V[k]+p0[k];
```

En esta función se implementa un cálculo en el cual se obtienen un número determinado de puntos, que constituyen la recta que forma l aposición de salida y la final, haciendo que el servomotor se posicione en cada punto obtenido, consiguiendo así el resultado buscado.

6.7 Función “detachServo”

Por último, se declara esta función para que una vez se haya finalizado el proceso, se pueda manipular el robot con la seguridad de no dañar los servomotores.

```
void detachServo(const RobotServo_t &servo)
{
  servos[servo.pin].detach();
}
```

7 CONCLUSIONES Y TRABAJO FUTURO

Dados los resultados tanto teóricos como prácticos del proyecto, se puede confirmar que se han cumplido todas las especificaciones y requerimientos planteados, obteniendo así el producto deseado.

En el desarrollo de este proyecto se ha profundizado en diferentes campos como su cinemática inversa, diseño de los componentes, simulaciones, cálculos de volumen de trabajo y de servomotores, con el fin de elaborar un robot paralelo que tuviera tres grados de libertad.

Durante la elaboración de cada etapa surgieron impedimentos a los cuales se les buscó una solución, llegando así el diseño final del producto. Pese a tener todos los cálculos correctos y simulaciones comprobadas surgen problemas los cuales solo se aprecian al llevar el diseño al terreno físico.

Podemos confirmar que el producto obtenido en este proyecto es el deseado. No obstante, siempre se pueden hacer modificaciones o aportaciones futuras para mejorarlo y completarlo como por ejemplo:

- Emplear otro material ligero y barato como el metacrilato, el cual se piensa que podría reducir los problemas surgidos de la holgura con la madera, dada su mayor precisión de elaboración y por tratarse además de un material que presenta poca deformación ante fuerzas externas.

- Acoplar una herramienta en el efector final. La opción lógica es emplear un micro servomotor que simule el movimiento angular de una herramienta, pero existe otra posibilidad. Esta consiste en emplear un láser ky-008 que proyecte un punto rojo en la base del robot. El esfuerzo físico de los servomotores para manipularlo es mínimo dado su reducido peso. Con esto, se podría comprobar si realiza correctamente la trayectoria del efector final, programando que haga diferentes figuras geométricas (cuadrado, circunferencia, triángulo...).

- Implementación de visión artificial para poder detectar objetos y con ello calcular la posición del efector final y realizar el movimiento hasta dicho punto. Este complemento aporta mucho valor y conocimiento al proyecto y no supone un gran incremento en el coste final.

El proceso de realización de este TFG ha resultado enriquecedor y estimulante. Enfrentarse a los problemas reales de un diseño de estas características supone un aprendizaje en sí mismo. Se considera muy satisfactorio el hecho de haber alcanzado el objetivo final por lo que implica de esfuerzo y como cierre a los 4 años del Grado de Ingeniería cursados. Además, se confirma la decisión de continuar la formación en este campo, optando al máster de Mecatrónica que oferta esta Universidad.

9 BIBLIOGRAFÍA

[1] Artículo robots Delta https://es.wikipedia.org/wiki/Robot_Delta

[2] Artículo comparativa modelos robots Delta https://revistaderobots.com/robots-y-robotica/robot-delta-aplicaciones-y-precios/

[3] Página oficial de Arduino https://www.arduino.cc/

[4] Manual de usuario Coppelia Sim https://www.coppeliarobotics.com/helpFiles/

[5] Empresa cortadora láser https://www.troteclaser.com/en/

[6] Artículo madera mdf https://es.wikipedia.org/wiki/Tablero_de_fibra_de_densidad_media

[7] Artículo Automatización industrial https://economipedia.com/definiciones/automatizacion-industrial.html

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Documento número 2:

Documentación Técnica

Diseño y programación de un Robot Delta basado en Arduino con fines académicos

Trabajo final de grado en Ingeniería Electrónica Industrial y Automática

Autor: Héctor López Tomás

Tutor: Leopoldo Armesto Ángel

Curso académico 2021-2022

ÍNDICE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

BATERÍA Litio 18650

Documento número 2: Documentación Técnica

Tenergy Corporation
436 Kato Terrace
Fremont CA 94539
Tel: 510-687-0388 Fax: 510-687-0328

TENERGY 18650 2200mAh Li-Ion Cell

Product Name:	Tenergy Lithium Ion 18650 Cell	
Product Number:	30003	
Battery Model:	18650 2200mAh	
Battery Chemistry:	Lithium Ion Rechargeable	
Dimension:	Max Diameter (ϕ): 18.3mm	
	Max Height (H): 65.0mm	

1. Scope

The specification describes the technology parameters and testing standard for the lithium ion rechargeable cell supplied by TENERGY CORPORATION.

2. References

This specification is referenced GB/T18287-2000, UL1642, IEC61960-1:2000.

3. Basic characteristics

3.1 Capacity	Nominal Capacity : 2200mAh (0.2C_A Discharge)
	Minimum Capacity: 2100mAh (0.2C_A Discharge)
3.2 Nominal Voltage	3.7V
3.3 Internal impedance	≤ 80mΩ(with PTC)
3.4 Discharge Cut-off Voltage	3.0V
3.5 Max Charge Voltage	4.20±0.02V
3.6 Standard Charge Current	0.5C_A
3.7 Rapid Charge Current	1C_A
3.8 Standard Discharge Current	0.5C_A
3.9 Rapid Discharge Current	1C_A

Tenergy Corporation
436 Kato Terrace
Fremont CA 94539
Tel: 510-687-0388 Fax: 510-687-0328

3.10 Max Discharge Current	2.0 C_A	
3.11 Weight	45±1g	
3.12 Max. Dimension	Diameter(ϕ):	18.3mm
	Height (H):	65.0mm
3.13 Operating Temperature	Charge	0 ~ 45℃
	Discharge	-20 ~ 60℃
3.14 Storage Temperature	Within 1 month	-5 ~ 35℃
	Within 6 months	0 ~ 35℃

4. Standard Conditions for Test

Unless specified, all tests should be conducted within one month after the delivery under the following conditions: Ambient Temperature: 25±5℃; Relative Humidity: 65±20%

4.1 Standard Charge:	Constant Current and Constant Voltage (CC/CV) Current = 1100mA End-up Voltage = 4.2 V End Current = 22mA
4.2 Standard Discharge:	Constant Current (CC) Current = 1100mA End Voltage = 3.0V

5. Characteristics

※In this section, the Standard Conditions of Tests see the part 4.

5.1 Electrical Performances

Item	Test procedure	Requirements
5.1.1 Nominal Voltage	The average value of the working voltage in the whole discharge progress.	3.7V
5.1.2 Discharge Performance	The discharge capacity of the cell, which is measured at $1C_5A$ (or 0.5CA) current discharge to 3.0V within 1 hour after completely charge.	≥57(or 120)min

5.1.3 Capacity Retention	After 28 days storage at 25±5℃ after completed charge, the residual capacity is above 90%.	Capacity≥1980mAh
5.1.4 Cycle Life	After 300 cycles in 100% DOD charge and discharge at 0.5CA current, the residual discharge capacity is above 60% of nominal capacity.	≥300 cycles
5.1.5 Storage	(Within 3 months after manufactured) after standard charged 40-50% capacity and stored at ambient temperature 25±5℃、65±20%RH for 12 months, the storage expiry and the cell completely charged, the cell is discharged at 0.2 CA current discharge to 3.0V.	Discharge time≥4h

5.2 safety Performances

5.2.1 Short Circuit	The cell is to be short-circuited by connecting the positive and negative terminals of the cell directly with copper wire with a resistance less than 0.05Ω.	No fire, no explosion.
5.2.2 Impact Test	Impacting of a cell on a hard surface following a hammer of 10 kilograms free fall from 1m height.	No fire, no explosion.

5.2.3 Overcharge (3C/10V)	The cell that connect with the thermocouple is put in the fume hood, the positive and negative terminals are connected by a permanent constant electrical source, regulate current to 3 CA and voltage to 10 V. Then charge the cell until voltage is 10 V, current about 0A. Monitor the temperature change of cell when the temperature of cell is about lower 10℃than peak value, the test is over.	No fire, no explosion.
5.2.4 Thermal shock	After standard charging, heat cell to 150±2℃at rate of 5±2℃/min and keep 10 minutes.	No fire, no explosion.

5.3 Environmental tests

5.3.1 High temperature performance	The fully charged cell is put in the surroundings of 55±2℃for 2 hours, and then it is discharged to the 2.75V at 1CA current rate.	Capacity≥2160mAh
5.3.2 Low temperature performance	The charged cell is put 16-24 hours at -20±2℃and then discharge to 2.75V at 0.2 CA current rate.	Capacity≥1680mAh
5.3.3 Vibration Test	After standard charging, the cell is fixed on the platform and be subjected to vibrate on following frequency 10~55Hz and amplitude vibration for 30 minutes with direction of X, Y. Vibration Frequency: 10~30Hz, vibration amplitude 0.38mm. Vibration Frequency: 30~55Hz, vibration amplitude 0.19mm	
5.3.4 Drop Test	The cell is to be dropped from a height of 1m to hard board in X、Y、Z directions for twice respectively. Then discharge the cell at 1CA	No fire, no explosion.

	current rate to 3.0V, and undertake more than three circles of standard charge and discharge at 1CA current rate.	

6. Packing

Keep the cells at the half-fully charged state before packing.

7. Transportation:

Violent shaking, bumping, rain and flaring sun shall be forbidden during the transportation. Keep the cells at the half-fully charged state.

8. Storage

Please keep the cell in the cool and dry environment: Within 1 month -5 ~ 35℃ or Within 6 months, 0 ~ 35℃， Relative humidity≤75%, Keep the cells at the half-fully charged state.

9. Warranty period of this product is 6 months from leaving plant.

10. We will not guarantee against any accidents occurring due to usage against this specification.

11. The information in this specification subject to change without prior notice.

12. The information contained in this document is for reference only and should not be used as a basis for product guarantee or warranty. For applications other than those described here, please consult TENERGY CORP directly.

13. Caution:

13.1 Please read the specification carefully before testing or using the cell, as improper handling of Lithium-ion cell may result in loss of efficiency, heating ignition, electrolyte leakage or even explosion.

13.2 While testing the cell of charging and discharging, please use the testing equipment special for Li-ion cell. Do NOT use the ordinary source of constant current and constant voltage, which fails to restrict charge and discharge to cell in order to

prevent the cell from being overcharged and over-discharged, triggering cell malfunction or explosion.

13.3 When charging and discharging to the cell or packing it into the equipment, do NOT reverse the terminals of cathode and anode or it will make the cell overcharging and over-discharging, causing the cell to lose efficiency seriously and even explode.

13.4 Do NOT weld the cell directly, do not disassembly the cell.

13.5 Do NOT put the cell together with such metal products as necklace, hairpin, coin or screw in the pocket or in the bag; neither store them together. Do NOT connect the positive and negative electrode directly with such conductive materials as metal, or it may make the cell short-circuit.

13.6 Do NOT beat, throw or trample the cell. Do NOT put the cell into the washing machine or the high-pressure container.

13.7 Do NOT put the cell close to heat source, for instance, fire, heater etc. Do NOT use the cell under the circumstance of burning sun or the temperature exceeding 60℃, or it may cause the cell to generate heat, heating ignition and loss of efficiency.

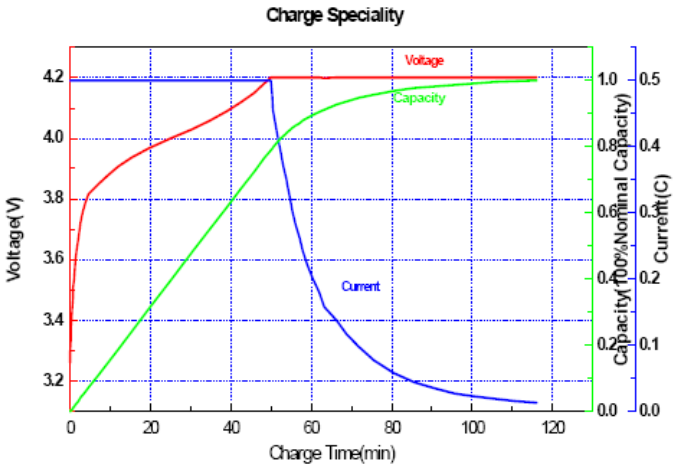
13.8 Do NOT get the cell wet or throw the cell into water. When not use, it should be placed in the dry and low temperature environment.

13.9 While using, testing or preserving the cell, if you find the battery become hot , distribute smell , change color, deform or any other abnormality, please stop using or testing immediately, and attempt to isolate and keep away from the cell.

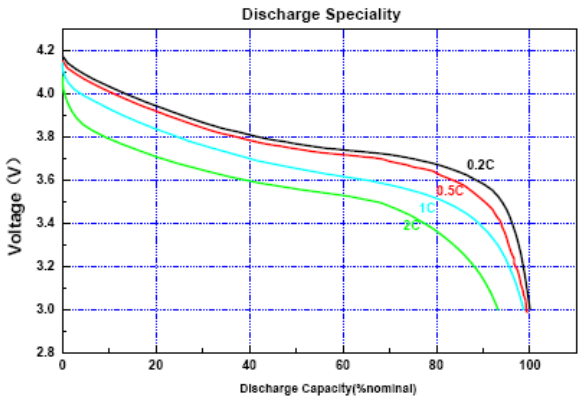
13.10 If the cell leaks, the electrolyte gets into the eyes, do not rub eyes, instead, rinse the eyes with plenty of water, and seek medical service. If the electrolyte gets onto the skin or clothe, wash it with plenty of water immediately.

Tenergy Corporation
436 Kato Terrace
Fremont CA 94539
Tel: 510-687-0388 Fax: 510-687-0328

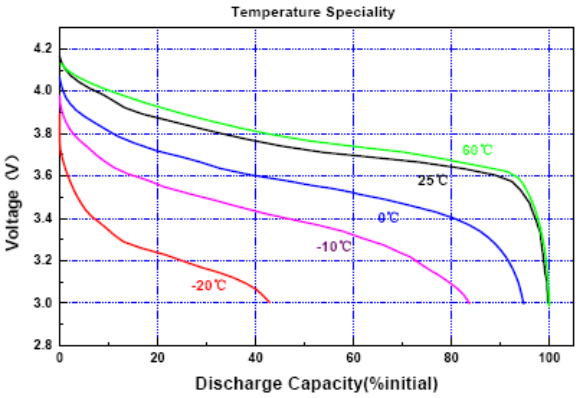
Performance Curve



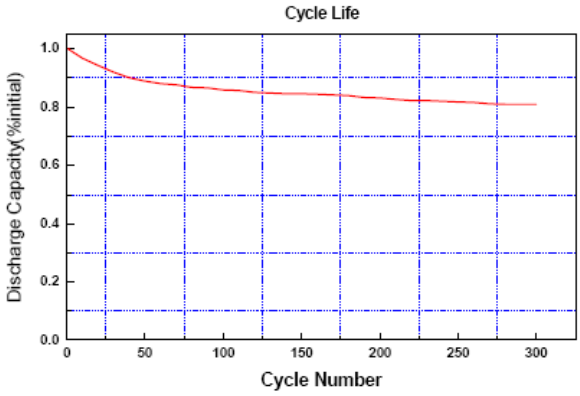
Charge: CC/CV 0.5CmA, 4.2V, 20mA cut off at RT.



Charge: CC/CV 0.5CmA, 4.2V, 20mA cut off at RT
Discharge: 3.0V cut off at RT.



Charge: CC/CV 0.5CmA, 4.2V, 20mA cut off
Discharge: 3.0V cut off



Charge: CC/CV 0.5CmA, 4.2V, 20mA cut off at RT
Discharge: CC 0.5CmA, 3.0 V cut off at RT.

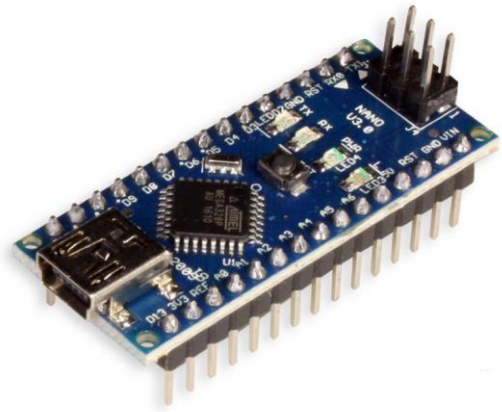
Note:
CC represent constant current
CV represent constant voltage
1C represent multiple current
RT represent room temperature

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

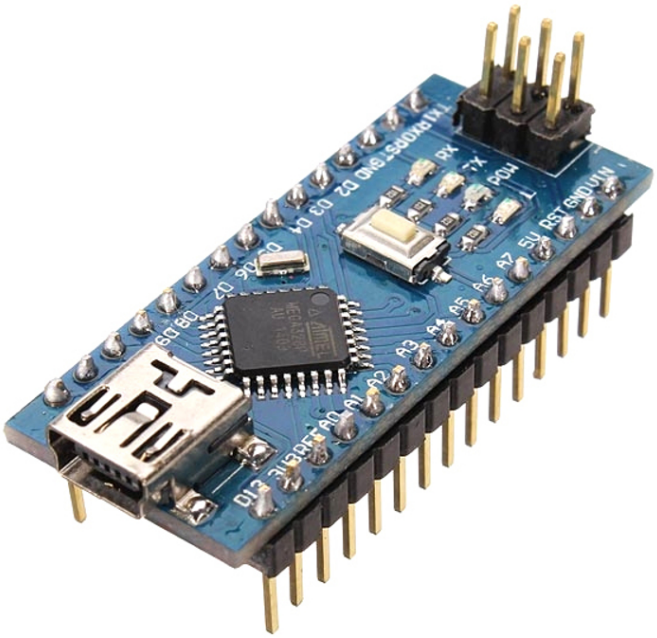
Escuela Técnica Superior de Ingeniería del Diseño

ARDUINO NANO

Documento número 2: Documentación Técnica



Nano v.3



Product Overview

The Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Nano 3.0) or ATmega168 (Nano 2.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one. The Nano was designed and is being produced by Gravitech.

Index

HK Shan Hai Group Limited
Room 620 , Yutian building , songling road , Futian district , Shenzhen

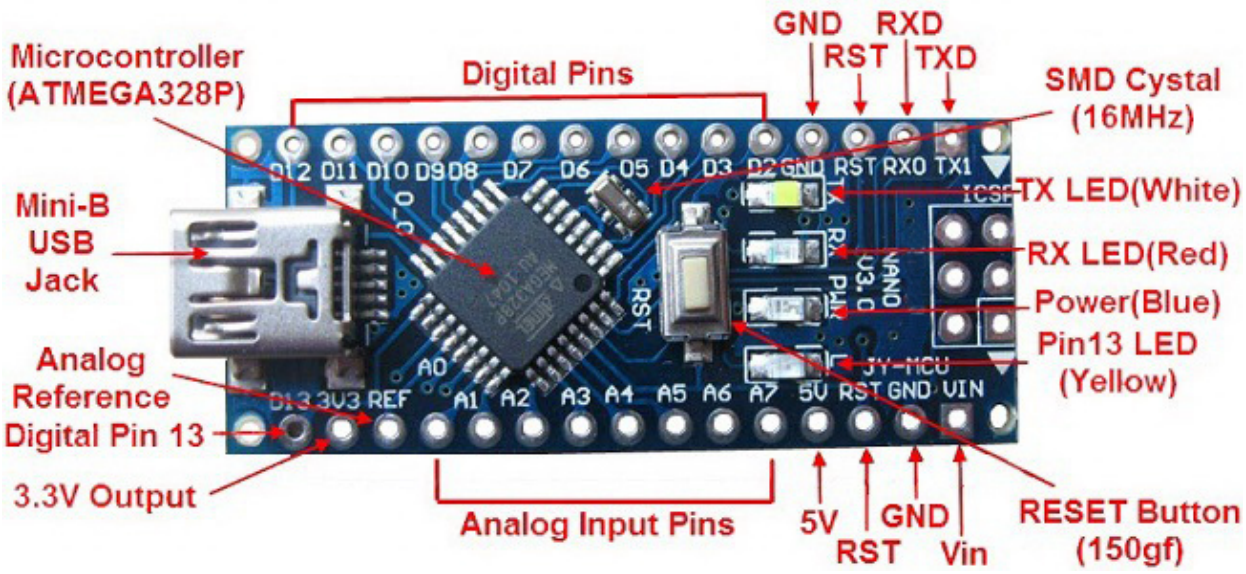
Nano 3.0 (ATmega328): schematic, Eagle files.

Nano 2.3 (ATmega168): manual (pdf), Eagle files. *Note:* since the free version of Eagle does not handle more than 2 layers, and this version of the Nano is 4 layers, it is published here unrouted, so users can open and use it in the free version of Eagle.

Summary

Microcontroller	Atmel ATmega328
Operating Voltage (logic level)	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (ATmega168) or 32 KB (ATmega328) of which 2 KB used by bootloader
SRAM	1 KB (ATmega168) or 2 KB (ATmega328)
EEPROM	512 bytes (ATmega168) or 1 KB (ATmega328)
Clock Speed	16 MHz
Dimensions	0.73" x 1.70"

the board



Power

The Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

The FTDI FT232RL chip on the Nano is only powered if the board is being powered over USB. As a result, when running on external (non-USB) power, the 3.3V output (which is supplied by the FTDI chip) is not available and the RX and TX LEDs will flicker if digital pins 0 or 1 are high.

Memory

The ATmega168 has 16 KB of flash memory for storing code (of which 2 KB is used for the bootloader); the ATmega328 has 32 KB, (also with 2 KB used for the bootloader). The ATmega168 has 1 KB of SRAM and 512 bytes of EEPROM (which can be read and written with the EEPROM library); the ATmega328 has 2 KB of SRAM and 1 KB of EEPROM.

Input and Output

Each of the 14 digital pins on the Nano can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Nano has 8 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the analogReference() function. Additionally, some pins have specialized functionality:

- **I^2C: 4 (SDA) and 5 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website).

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega168 ports.

Communication

The Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega168 and ATmega328 provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with the Arduino software) provide a virtual com port to software on the computer. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Nano's digital pins.

The ATmega168 and ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega168 or ATmega328 datasheet.

Programming

The Nano can be programmed with the Arduino software (download). Select "Arduino Diecimila, Duemilanove, or Nano w/ ATmega168" or "Arduino Duemilanove or Nano w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega168 or ATmega328 on the Arduino Nano comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Nano is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the FT232RL is connected to the reset line of the ATmega168 or ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Nano is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Nano. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you'll need to install the FTDI Drivers to let your PC talk to the board. First **Plug the Arduino to your PC via USB cable.**

Blink led

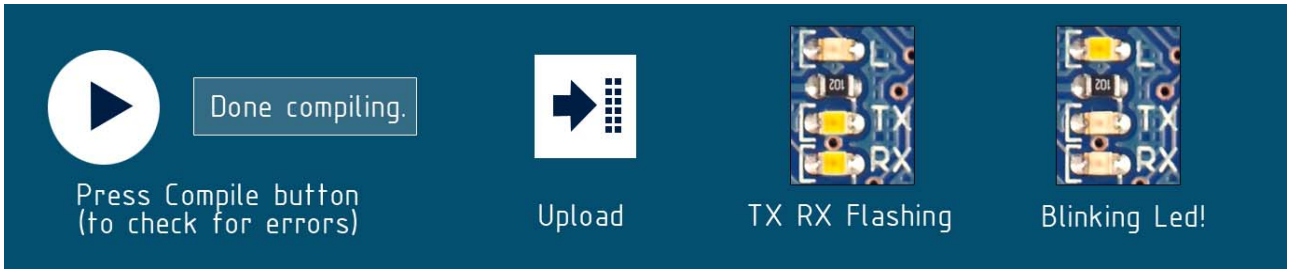
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

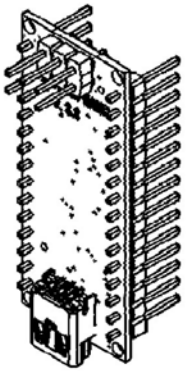
Once you have your skecth you'll see something very close to the screenshot on the right.

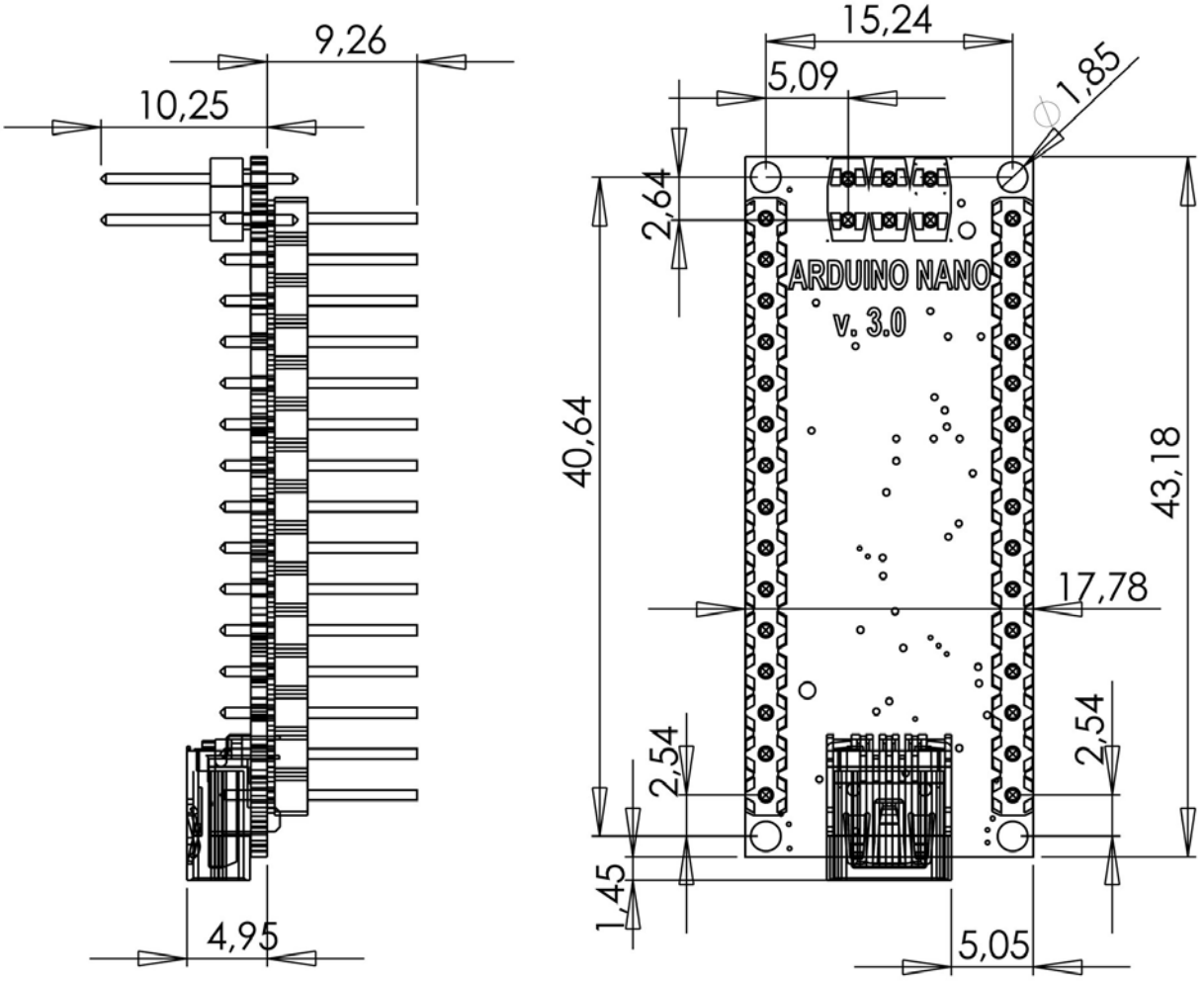
In **Tools>Board** select Arduino NANO and with the AtMEGA you're using (probably 328)

Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.

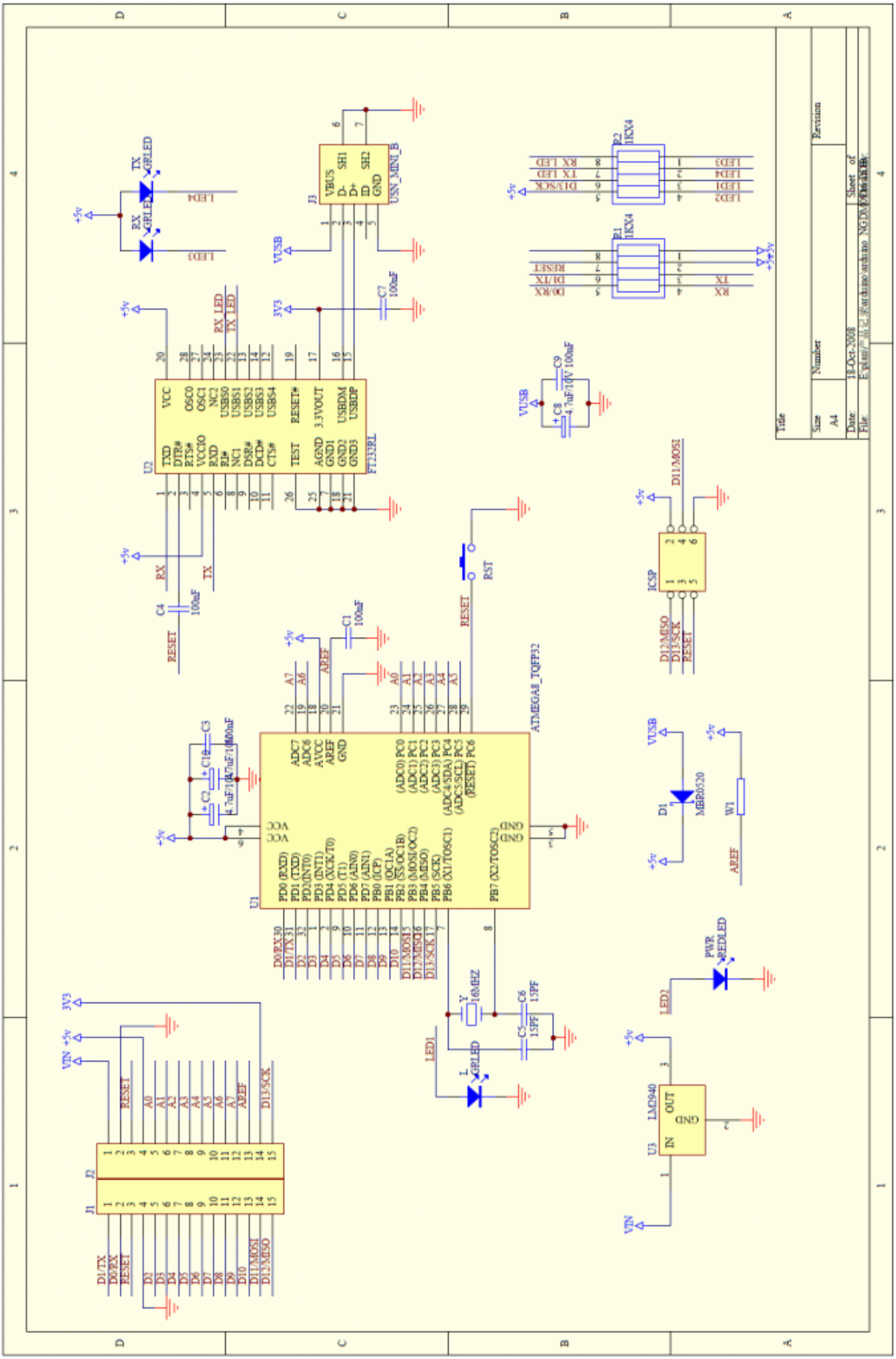


Dimensioned Drawing





HK Shan Hai Group Limited
Room 620 , Yutian building , songling road , Futian district , Shenzhen



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

EXPANSION SHIELD

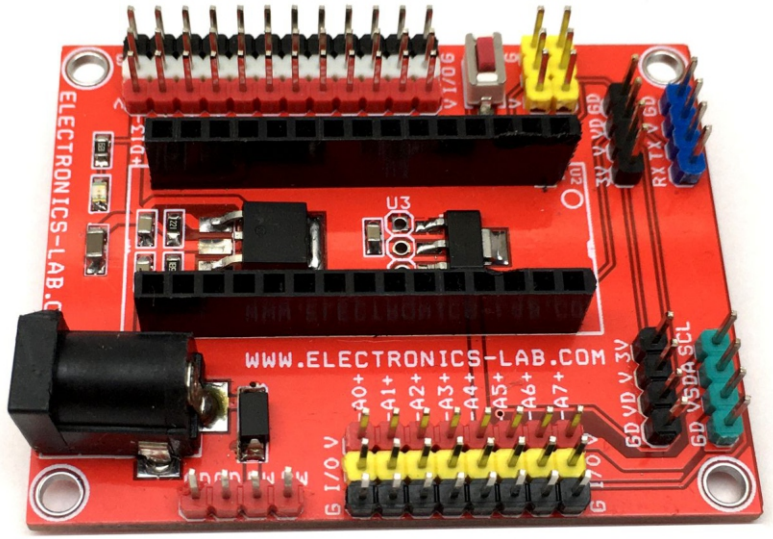
Documento número 2: Documentación Técnica

Expansion Shield-Breakout Board for Arduino Nano

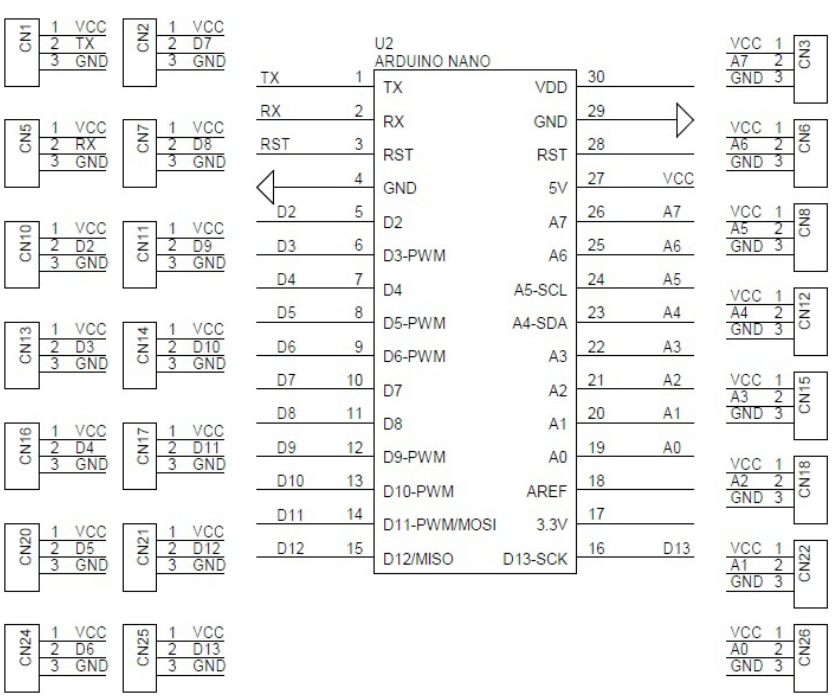
If you need to interface many devices and sensors to Arduino Nano, then, this project is for you. This is a Nano expansion I/O shield (breakout board) for the Arduino Nano. The board facilitates the easy connection between Arduino Nano and other devices. Each Arduino (I/O) Pin including the 5V DC and GND pins are available for easy connection to the sensors and other devices. The board enables the easy interface of many devices and sensors which includes various power voltage options. It provides several different options for power outputs and wide range of operating power supply input.

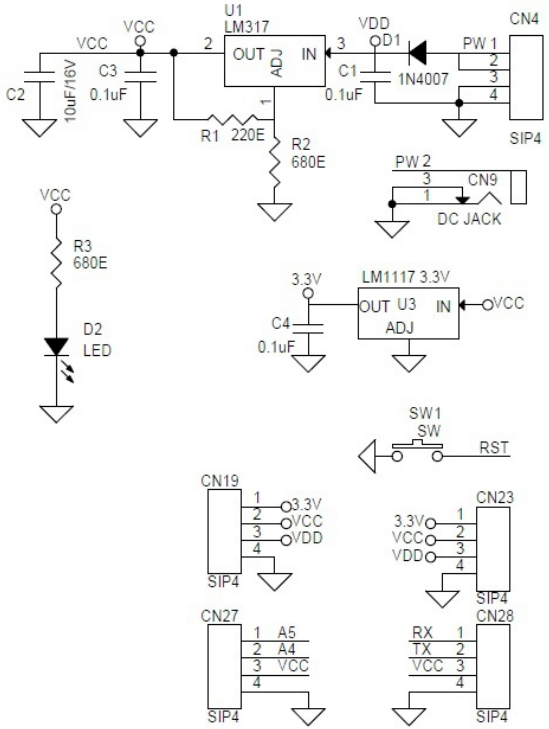
Features

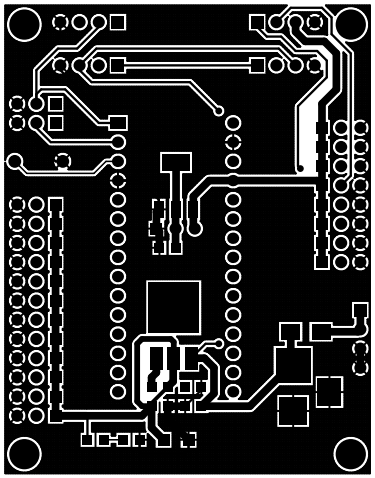
- All I/O pins including 5V DC and GND Pins (5V-I/0-GND)
- On Board TX/RX/5V/GND Connector for Serial Interface
- On Board I2C Connector (A5SCL- A4SDA-5V-GND)
- 2x Connector for Power Output: VD (Input Supply After Diode), 5V DC, 3.3V DC)
- Supply Input 7V to 24V DC Using DC Jack or Header Connector
- On Board 5V Regulator L317-Adj (Output Set to 5V)
- On Board 3.3V Regulator LM1117-3.3V
- On Board Power LED
- On Board DC Jack for Power Input 7V-24V DC
- Reverse Supply Protection Diode on Power Input
- Optional 4 Pin Header Connector for Power Input
- On Board Reset Switch
- PCB Dimensions 62.55MM X 48.90MM



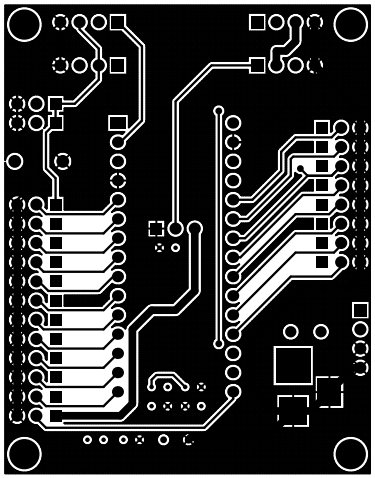
Author Rajkumar Sharma


CN1 1 VCC 2 TX 3 GND
CN2 1 VCC 2 D7 3 GND
CN5 1 VCC 2 RX 3 GND
CN7 1 VCC 2 D8 3 GND
CN10 1 VCC 2 D2 3 GND
CN11 1 VCC 2 D9 3 GND
CN13 1 VCC 2 D3 3 GND
CN14 1 VCC 2 D10 3 GND
CN16 1 VCC 2 D4 3 GND
CN17 1 VCC 2 D11 3 GND
CN20 1 VCC 2 D5 3 GND
CN21 1 VCC 2 D12 3 GND
CN24 1 VCC 2 D6 3 GND
CN25 1 VCC 2 D13 3 GND
U2
ARDUINO NANO
TX 1 TX
RX 2 RX
RST 3 RST
4 GND
D2 5 D2
D3 6 D3-PWM
D4 7 D4
D5 8 D5-PWM
D6 9 D6-PWM
D7 10 D7
D8 11 D8
D9 12 D9-PWM
D10 13 D10-PWM
D11 14 D11-PWM/MOSI
D12 15 D12/MISO
VDD 30
GND 29
RST 28
5V 27 VCC
A7 26 A7
A6 25 A6
A5-SCL 24 A5
A4-SDA 23 A4
A3 22 A3
A2 21 A2
A1 20 A1
A0 19 A0
AREF 18
3.3V 17
D13-SCK 16 D13
CN3 VCC 1 A7 2 GND 3
CN6 VCC 1 A6 2 GND 3
CN8 VCC 1 A5 2 GND 3
CN12 VCC 1 A4 2 GND 3
CN15 VCC 1 A3 2 GND 3
CN18 VCC 1 A2 2 GND 3
CN22 VCC 1 A1 2 GND 3
CN26 VCC 1 A0 2 GND 3

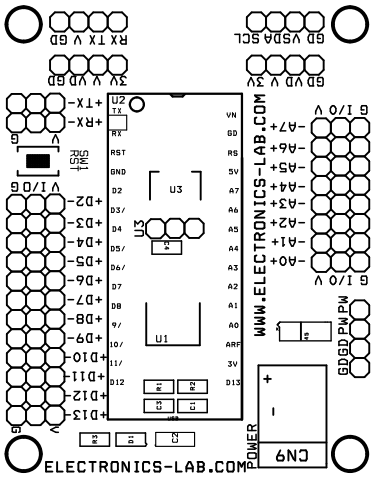

U1
LM317
VCC
OUT
ADJ
IN
VDD
D1
1N4007
CN4
PW 1
SIP4
C2
10uF/16V
C3
0.1uF
C1
0.1uF
R1 220E
R2
680E
PW 2
CN9
DC JACK
R3
680E
D2
LED
3.3V
LM1117 3.3V
U3
C4
0.1uF
SW1
SW
RST
CN19
SIP4
CN23
CN27
A5
A4
CN28
RX
TX



TOP LAYER

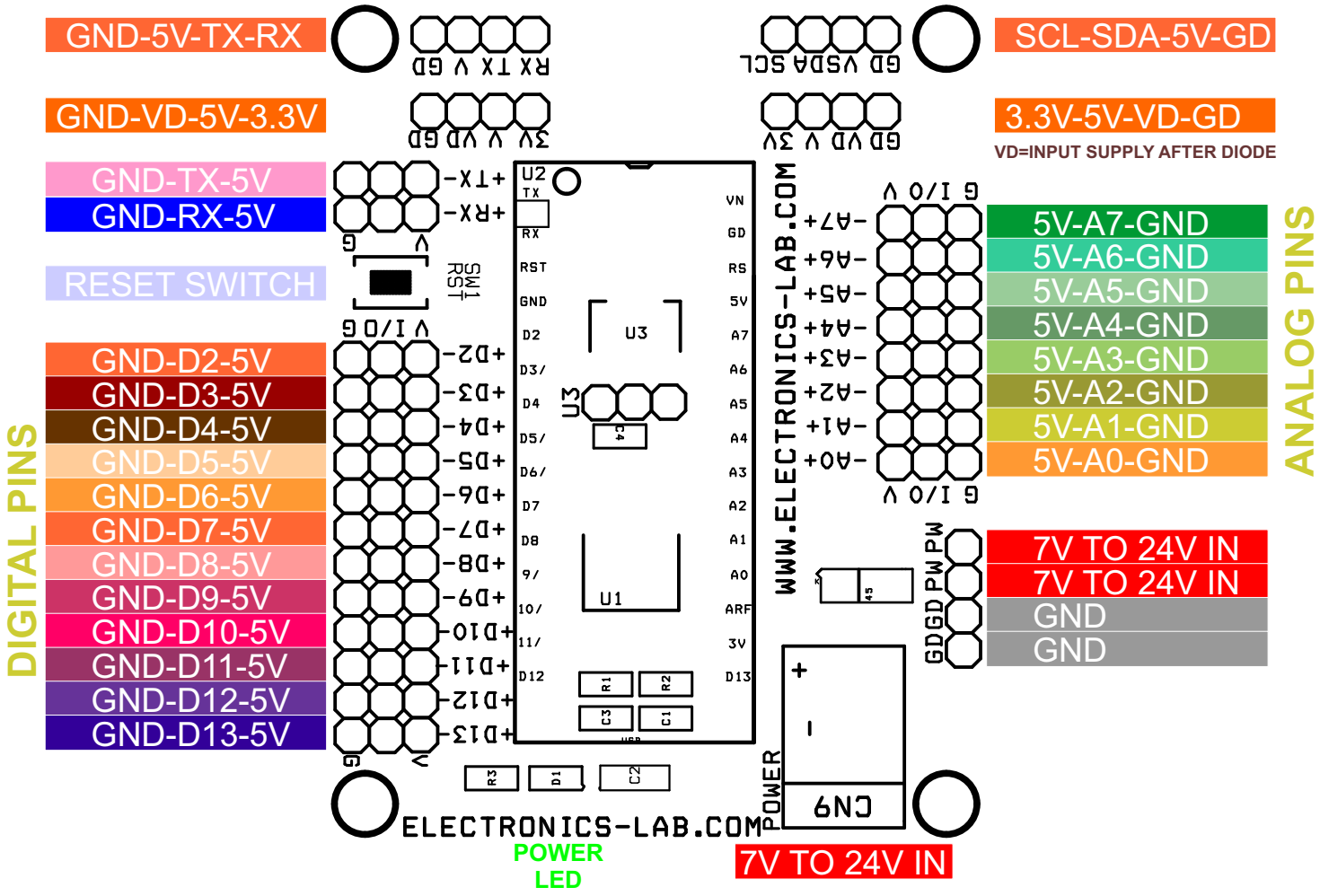


BOTTOM LAYER



SILK SCREEN TOP

BOM				
SR.	**QNTY.**	**REF.**	**DESC.**	**VENDOR/DIGIKEY/MOUSER**
1	22	CN1 TO CN8,CN10 TO CN18,CN20,CN21,CN22,CN24,CN25,CN26	3 PIN MALE HEADER 2.54MM PITCH	DIGIKEY S1011EC-40-ND
2	5	CN4,CN19,CN23,CN27,CN28	4 PIN MALE HEADER 2.54MM PICTH	DIGIKEY S1011EC-40-ND
3	1	CN9	DC JACK	DIGIKEY 486-4186-ND
4	3	C1,C3,C4	0.1uF/50V SMD SIZE 0805	YAGEO
5	1	C2	10uF/16V SMD SIZE 1206	YAGEO
6	1	D1	1N4007 SMD	DIGIKEY 1655-1N4007FLDKR-ND
7	1	D2	LED SMD SIZE 0805	DIGIKEY 475-1415-2-ND
8	1	R1	220E 5% SMD SIZE 0805	YAGEO
9	2	R2,R3	680E 5% SMD SIZE 0805	YAGEO
10	1	SW1	TACTILE SWITCH 2PIN THT	DIGIKEY CKN12306-ND
11	1	U1	LM317 DPAK	DIGIKEY LM317MDTRKGOSCT-ND
12	1	U2	ARDUINO NANO	DIGIKEY 1050-1001-ND
13	1	U3	LM1117 3.3V	DIGIKEY LM1117MPX-3.3/NOPBCT-ND
14	2	SCKT	2X14 FEMALE HEADER 2.54MM PITCH	DIGIKEY S7012-ND

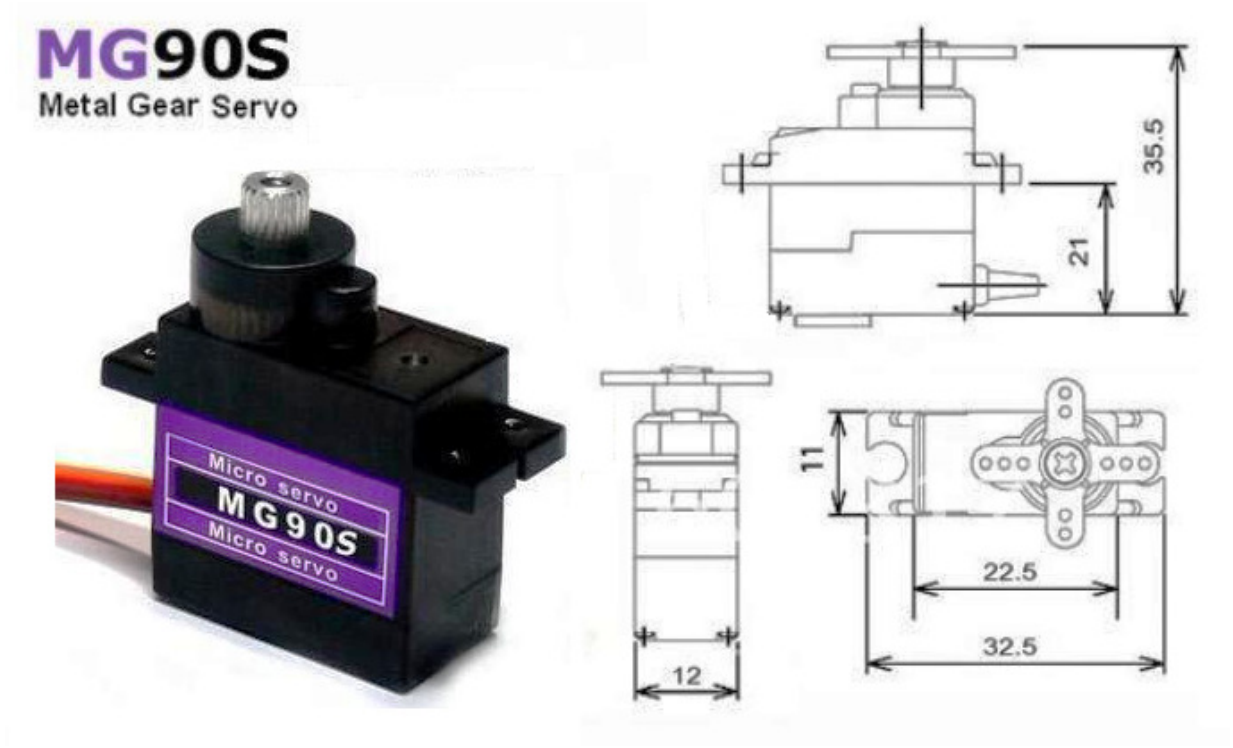

GND-5V-TX-RX
GND-VD-5V-3.3V
GND-TX-5V
GND-RX-5V
RESET SWITCH
GND-D2-5V
GND-D3-5V
GND-D4-5V
GND-D5-5V
GND-D6-5V
GND-D7-5V
GND-D8-5V
GND-D9-5V
GND-D10-5V
GND-D11-5V
GND-D12-5V
GND-D13-5V
DIGITAL PINS
SCL-SDA-5V-GD
3.3V-5V-VD-GD
VD=INPUT SUPPLY AFTER DIODE
5V-A7-GND
5V-A6-GND
5V-A5-GND
5V-A4-GND
5V-A3-GND
5V-A2-GND
5V-A1-GND
5V-A0-GND
ANALOG PINS
7V TO 24V IN
7V TO 24V IN
GND
GND
WWW.ELECTRONICS-LAB.COM
ELECTRONICS-LAB.COM
POWER
LED
7V TO 24V IN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SERVOMOTOR MG90s

Documento número 2: Documentación Técnica



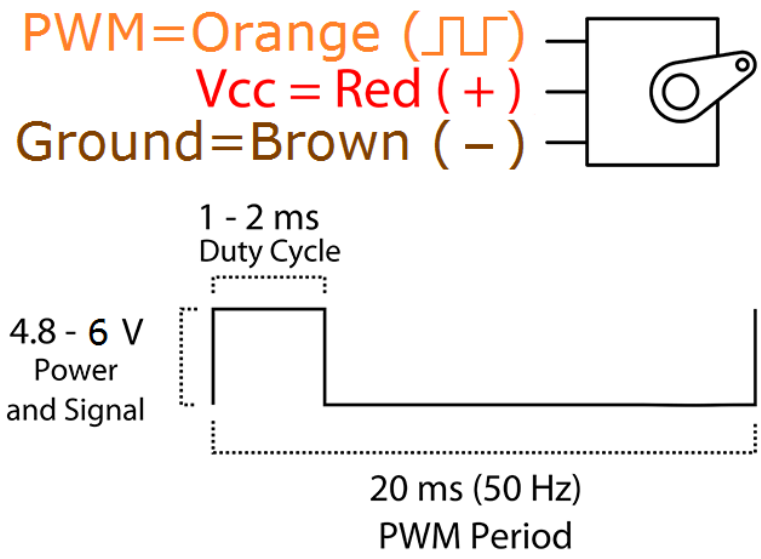
MG90S servo, Metal gear with one bearing

Tiny and lightweight with high output power, this tiny servo is perfect for RC Airplane, Helicopter, Quadcopter or Robot. This servo has *metal gears* for added strength and durability.

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: 13.4 g
- Dimension: 22.5 x 12 x 35.5 mm approx.
- Stall torque: 1.8 kgf·cm (4.8V), 2.2 kgf·cm (6 V)
- Operating speed: 0.1 s/60 degree (4.8 V), 0.08 s/60 degree (6 V)
- Operating voltage: 4.8 V - 6.0 V
- Dead band width: 5 µs



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

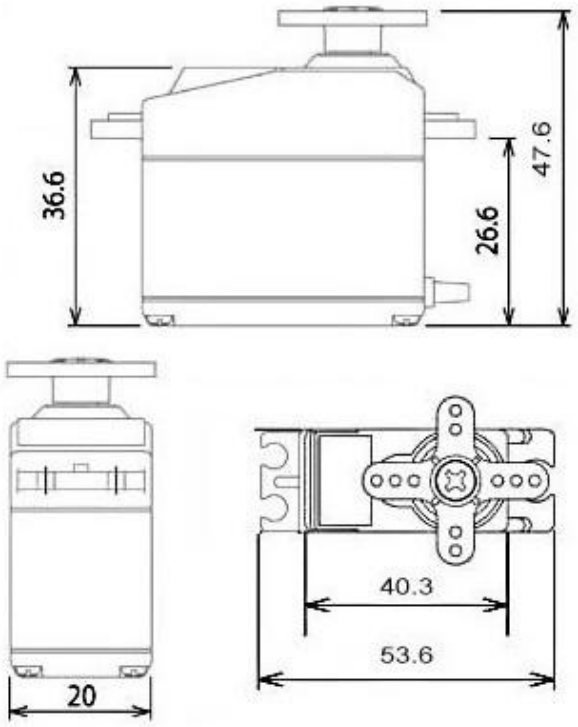
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SERVOMOTOR MG996R

Documento número 2: Documentación Técnica

MG996R High Torque
Metal Gear Dual Ball Bearing Servo



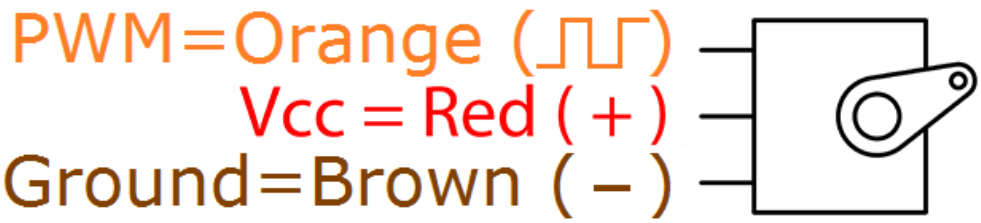
This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwith and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

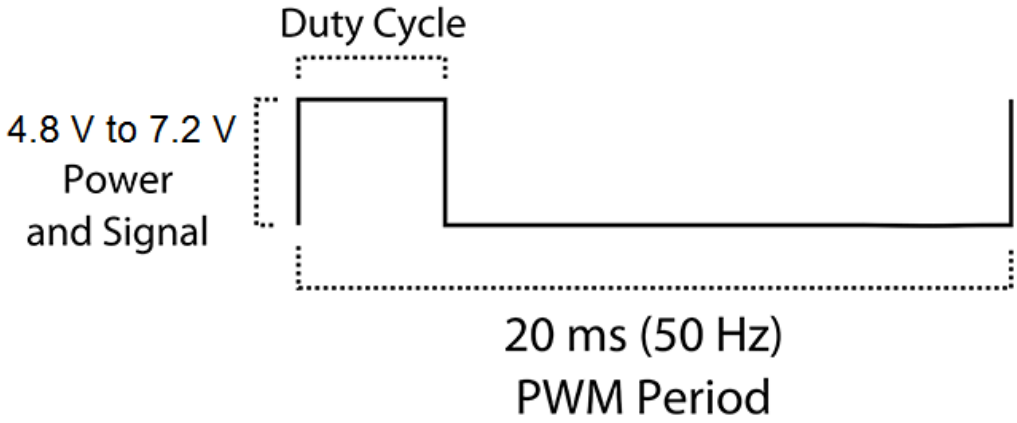
This high-torque standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Operating speed: 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)

- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC





UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SHIELD 18650 V3

Documento número 2: Documentación Técnica

Shield 18650 V3 cargador para 1 batería

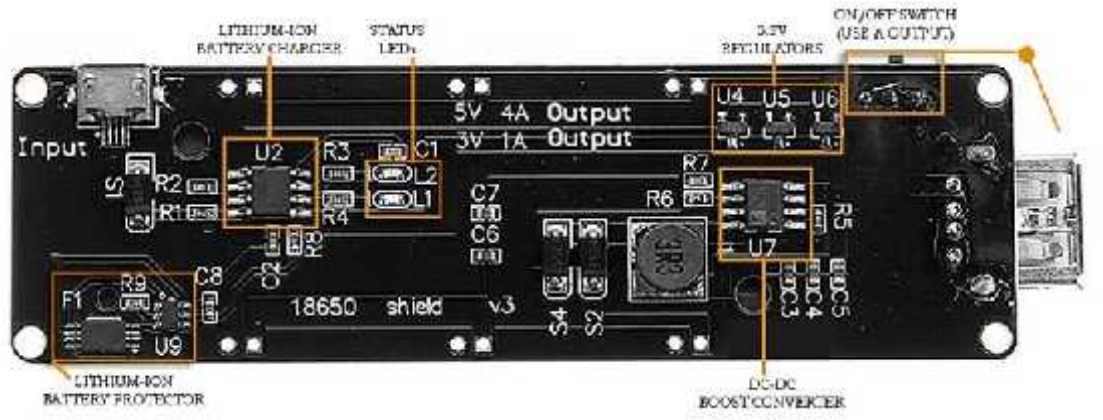
- Modelo: 18650 Shield V3
- Color de PCB: Negro o Azul
- Dimensiones: 10cm x 2.9cm x 3.8cm (largo x ancho x alto)
- Peso: 5g
- Salida USB tipo A
- Voltaje de entrada puerto MicroUSB: 5VDC
- Voltaje de salida pines:
    - 3.3V 1A
    - 5V 2A
- Puerto USB tipo A voltaje de salida: 5VDC
- Indicador LED (verde significa completo, rojo significa carga)
- Pines de voltaje de salida:
    - 3V x 3
    - 5V x 3
- Protección de la batería (sobrecarga o descarga excesiva)
- Incorpora interruptor para control de salida del puerto USB tipo A

El módulo 18650 Shield V3 también conocido como porta batería de tipo cilíndrica 18650 permite la correcta carga y descargar baterías 18650, el voltaje de entrada debe de ser de 5V a una corriente de 500mA.

Cuenta con un porta pilas para insertar de forma rápida y sencilla las baterías; tiene 3 pines de salidas de voltaje de 3.3V y 5V respectivamente en donde podrás soldar tu mismo Headers de tipo macho en estas salidas para facilitar las conexiones y alimentar otros dispositivos.

También incorpora un conector USB de tipo A Hembra para conectar cables USB de tipo “USB A Macho a MicroUSB Macho”, esto facilitara tener voltaje de salida y alimentar tarjetas de desarrollo como placas ESP32 ya sea su versión de 30 pines o la de 38 pines, de igual forma podrás alimentar otras placas que tengan puerto MicroUSB hembra.

Sirve para cargar y descargar baterías 18650 ya que incorpora un circuito de carga (TP4056A) con protección de batería 1S (DW01V) y un convertidor Boost DC-DC (FP6298) y reguladores de voltaje para alimentar placas electrónicas ya sea mediante los pines de voltaje de salida a 3.3V, 5V y mediante el conector USB. Este cargador de baterías es de gran ayuda en proyectos de electrónica donde se requiera una fuente de voltaje externa con batería recargable.



COMPONENTE DESCRIPCIÓN

U9 = DW01V	Chip de protección de batería de iones de litio 1S
F1 = 8205A	MOSFET de doble canal N
U2 = TC4056A	Chip cargador de batería Li-Ion 1S
U7 = FP6298	Chip convertidor boost dc-dc de modo actual 4.5A
U4, U5, U6 = 662K (XC6206xxxx)	Chip regulador de voltaje positivo (3,3 V)
L1 = LED verde	Indicador CHGD de batería
L2 = LED rojo	Indicador CHRG de batería
S1, S2, S4 = diodos Schottky	S1 = SS14 y S2, S4 = SS24

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Documento número 3:

Pliego de Condiciones Técnicas

Diseño y programación de un Robot Delta basado en Arduino con fines académicos

Trabajo final de grado en Ingeniería Electrónica Industrial y Automática

Autor: Héctor López Tomás

Tutor: Leopoldo Armesto Ángel

Curso académico 2021-2022

ÍNDICE

1 OBJETO

El presente Pliego de Condiciones Técnicas constituye el conjunto de especificaciones, condiciones y criterios que definen todos los requisitos técnicos del proyecto Diseño de un Robot Delta basado en Arduino con fines académicos.

El pliego contiene además, la descripción general del proyecto, las condiciones que han de cumplir los materiales y las instrucciones para la fabricación.

2 ESPECIFICACIONES

2.1 Especificaciones del software

El software del proyecto debe desarrollarse cumpliendo con las siguientes especificaciones:

- El software debe permitir ser compilado por distintos sistemas operativos.
- El código será desarrollado mediante software de código libre.
- Su complejidad de lenguaje de programación ha de ser estándar (C, C++).

Se decide emplear Arduino IDE como software del proyecto, necesario para la programación del producto y que permite escribir, depurar, editar y grabar el programa (llamado “sketch” en el mundo Arduino) de una manera sumamente sencilla.

2.2 Especificaciones de hardware

El hardware del proyecto debe cumplir las siguientes especificaciones:

- El material que conforme el producto ha de ser ligero y cómodo de manipular, además de tener una accesibilidad sencilla para el usuario.
- El microprocesador debe tener un lenguaje de programación estándar y ha de ser capaz de acoplarse a un adaptador Extension Shield.
- Los actuadores deben tener un par mínimo de 1.7 Kg/cm, y que no superen los 10 € de coste por unidad.
- La batería debe ser capaz de ofrecer una salida de tensión de 5 V constantes, con un amperaje mínimo de 2000 mAh.

El material del robot es madera mdf, que se obtiene aplicando presión y calor a fibras de madera a las que se ha añadido previamente un adhesivo. Se caracteriza por tener una densidad de 450 kg/m3. La denominación exacta recogida en la normativa es: tableros de fibras fabricados por el proceso seco. Son extremadamente resistentes, con una superficie menos porosa, creando mayores posibilidades en el diseño de los entornos.

El microprocesador del robot es Arduino Nano. Es una placa de desarrollo de tamaño compacto, completa y compatible con protoboards, basada en el microcontrolador

ATmega328P. La placa está equipada con conjuntos de pines de E/S digitales y analógicos que pueden conectarse a varias placas de expansión y otros circuitos. Se trata de una placa de desarrollo y pruebas completa y fácil de usar que mide sólo 45 x 18 mm. Es posible alimentar la tarjeta Arduino Nano por el puerto mini-USB, por el conector 5V o por el conector de entrada de alimentación VIN. La fuente de alimentación que proporcione el mejor voltaje es la que seleccionará la tarjeta como fuente de alimentación.

Los actuadores, dependiendo del uso que se le quiera dar al producto, son dos:

1) El servomotor SG90, que es un pequeño actuador rotativo o bien motor que permite un control preciso en posición angular, que puede rotar de 0° hasta 180°, cuyo voltaje de operación va desde los 4.8 a 6 VDC y tiene un par entorno 1.8 y 2.2 Kg/cm.

2) El servomotor MG996R, que presenta unas dimensiones mayores (54.2 x 45 x 20 mm), el rango de rotación es de 180º, su par de torsión oscila entre 9.4 y 11 Kg/cm, se puede alimentar entre 4.8 y 6 V y tiene un consumo en torno a 500 y 600 mAh.

La fuente de alimentación del robot es una batería 18650. Se trata de una batería recargable Li-ion, que se parece mucho a la pila tipo AA convencional, la cual tiene 1.5/1.2V, pero esta tiene en la salida el voltaje 3.7 V y capacidad de 2600 mAh. Son baterías muy ligeras, pero con una elevada capacidad energética y resistencia a la descarga, y con poco efecto memoria. Además, aguantan un elevado número de ciclos. Para su correcta utilización es necesario emplear un porta-baterías 18650 para realizar la carga de esta, obtener los pines de salida de 5 V, además de un interruptor para encender o apagar la alimentación.

2.3 Especificaciones de desarrollo

Para poder desarrollar este proyecto desde cero, tal y como se redacta en la memoria, es necesario el uso de diferentes softwares como Arduino IDE, Matlab, CoppeliaSim, SolidWorks y Meshmixer.

También es necesario un equipo de ordenador capaz de soportar los requerimientos de dichas aplicaciones, los cuales son:

- Windows 7 o superior (de 64 bits)
- CPU de doble núcleo; se recomienda cuatro núcleos
- 2 GB de espacio libre en disco; se recomiendan 5 GB
- 8 GB de RAM; se recomiendan 16 GB
- 2 GB o más de RAM de GPU; se recomiendan 4 GB
- Tarjeta gráfica NVIDIA®: NVIDIA Quadro®/NVIDIA GeForce®/Tesla™ con chip NVIDIA Kepler™ como mínimo; se recomienda una configuración de doble GPU con tarjetas NVIDIA Maxwell™ como mínimo para disfrutar de una mejor experiencia.
- Se recomienda la versión del controlador NVIDIA 385.41 o superior
- Se necesita un controlador NVIDIA compatible con CUDA® 9.0 o superior
- Conexión de HDR Light Studio: HDR Light Studio v5.3.3 o superior, excepto v5.4
- Se necesita un mínimo de 4 GB de memoria de vídeo para poder utilizar la función Denoiser.

3 ROBOT DELTA

3.1 Enumeración de las piezas

El robot está compuesto por las siguientes piezas:

- 1x Base superior
- 3x Pilar
- 1x Base inferior
- 3x Agarre servomotor MG996R
- 3x Soporte servomotor MG996R
- 6x Brazo actuador
- 12x Eslabón corto paralelogramo 1
- 12x Eslabón corto paralelogramo 2
- 6x Eslabón largo paralelogramo
- 3x Sujeción efector final
- 1x Efector final plataforma inferior
- 1x Efector final plataforma superior
- 3x Tornillos con punta hierro
- 12x Tornillos plástico
- 12x Tuercas
- 12x Arandelas
- 3x Elementos de unión actuadores
- 6x Rodamientos

3.2 Montaje base

Para el montaje de la base se emplean la “Base superior”, la “Base inferior” y los “Pilares”, donde empleando los tornillos metálicos capaces de perforar la madera, se asegura la sujeción de toda la estructura. Después se encajan “Agarre servomotor MG996R” y “Soporte servomotor MG996R” a la “Base superior”.

Cabe destacar que es importante orientar la “Base superior” de forma correcta, dependiendo de dónde se quieran ubicar los actuadores.

3.3 Montaje efector final

Para el montaje del efector final únicamente se emplean las piezas “Efector final plataforma inferior”, “Efector final plataforma superior” y la “Sujeción efector final”, donde no es necesario utilizar ningún tipo de elemento de unión o tornillo, ya que las piezas están diseñadas para que encajen perfectamente entre ellas siendo el proceso muy intuitivo. Después, hay que situar uno de los rodamientos en las piezas de “Sujeción efector final”.

3.4 Montaje brazo

Para el montaje del brazo se empieza por adherir dos piezas “Brazo actuador” entre sí, conformando el mismo elemento, pero con doble grosor, teniendo así un total de tres pares.

Una vez los brazos obtenidos se han secado, se procede a unirlos con los servomotores empleando las cabezas y tornillos que vienen con ellos (importante realizar esta unión de forma que el brazo se quede perpendicular a la base del robot y mirando hacia abajo, con los servomotores colocados en la posición 0º), y después encajar uno de los rodamientos en el extremo grueso del brazo.

Posteriormente se pasa al montaje del paralelogramo del brazo, que está formado por cuatro lados: dos largos y dos cortos.

Para los lados cortos se emplean las piezas “Eslabón corto paralelogramo 1” y “Eslabón corto paralelogramo 2”, las cuales están diseñadas para que encajen entre sí, formando un semieje con una base cilíndrica, que deberá encajar en el rodamiento del brazo de forma que quede lo más centrado posible, y este proceso se repite nuevamente para situarlo al otro lado del mismo rodamiento. Hay que realizar lo anteriormente dicho para cada rodamiento del robot.

Una vez completados los pasos anteriores, para cerrar la estructura del robot, se emplean las piezas “Eslabón largo paralelogramo”, que se encargan de unir los ejes del brazo con el del efector final, usando los tornillos de plásticos, las tuercas, y unas arandelas que separen el contacto entre las superficies de madera, cerrando la cinemática del robot.

Este proceso hay que repetirlo para cada uno de los brazos que constituyen al robot.

4 NORMAS DE EJECUCIÓN

La única norma de ejecución con la que cuenta este proyecto es la elaboración del conjunto de piezas. Las piezas se realizan empleando una cortadora láser con una superficie mínima de trabajo de 600 x 300 mm, para poder agrupar todas las piezas en un único plano y conseguir así realizar un único corte para cada elaboración del producto.

También debe poder trabajar con espesores superiores a 5 mm, a pesar de que este proyecto esté diseñado para emplear piezas de grosor 3 mm. Además, debe tener una precisión de centésima de milímetro para asegurar las medidas de las piezas que cuentan con valores de décimas de milímetro.

Hay varios modelos que concuerdan con las características y propiedades mencionadas, un ejemplo es la Grabadora y cortadora láser de alto rendimiento: serie Speedy, que pertenece a la compañía Troteclaser, donde emplea un tipo de láser CO2, Flexx o láser de fibra, cuya potencia láser oscila entre 20 y 120 vatios. Este modelo tarda en realizar el corte del conjunto total de las piezas del robot entre 6 y 7 minutos.

5 PRUEBAS DE SERVICIO

Los procesos de pruebas de funcionamiento y pruebas físicas no se llevan a cabo para todos los productos elaborados. Se realiza en un producto de cada lote de veinte.

5.1 Pruebas de funcionamiento

Para las pruebas de funcionamiento se realizan dos procesos diferentes. En el primero se emplea únicamente la parte eléctrica del robot, haciendo uso de la fuente de alimentación, el microprocesador con su adaptador y los servomotores, donde se emplea un fichero de Arduino básico el cual únicamente otorga valores de ángulos separados entre sí 30º (por ejemplo: 0º, 30º, 90º, 120º, 150º y 180º), se comprueba el correcto funcionamiento de los actuadores, del microprocesador, y del rendimiento de la batería. Se emplea la velocidad máxima que pueda ofrecer el servomotor y aguantará 2 segundos para cada una de las posiciones que se le asignen.

El segundo proceso comienza una vez el montaje estructural y las conexiones eléctricas se han realizado, para así comprobar la movilidad del robot y la fuerza de los servomotores al añadirle el peso de los brazos y del efector final. Para ello, dado que el proyecto ofrece al usuario la posibilidad de emplear dos tipos diferentes de servomotores, este procedimiento se realiza para ambos. A través de un fichero de Arduino, en el cual hay programado un recorrido de 5 puntos realizando una traslación lineal del efector final entre cada punto, a una velocidad de movimiento de dos segundos entre puntos, se consigue comprobar el funcionamiento dinámico del robot, ya que al tratar con un material que puede presentar pequeñas oscilaciones y holguras en el producto, es necesario comprobar su correcto funcionamiento.

5.2 Pruebas físicas

Las pruebas físicas se realizan únicamente en las piezas de madera del robot, ya que al realizar su corte, o por defecto de fábrica, pueden presentar micro roturas, las cuales pueden astillar la pieza.

Estas pruebas consisten únicamente en una inspección visual, seguida de una comprobación manual de la torsión de la pieza, dado que si presenta algún tipo de defecto, será detectado por el ojo humano, y en caso de no ser muy perceptible, al aplicar una ligera torsión sobre la pieza, esta cederá rompiéndose e inutilizándose.

6 CONDICIONES DE ENTREGA

6.1 Embalaje

El producto cumplirá todas las exigencias en lo relativo a toxicidad, condiciones de envasado, transporte y etiquetado.

Cada actuador va dentro de una pequeña bolsa de plástico que a su vez está dentro de otra bolsa de plástico con los componentes que vienen con el actuador (cabezas y tornillos). El Arduino Nano y el Extension Shield vienen separados, cada uno envuelto en papel de burbuja para proteger el producto contra vibraciones y golpes que puedan causar daños. Los tornillos, tuercas, arandelas y rodamientos comparten bolsa de plástico en uno de cuyos lados dispone de un cierre adhesivo.

Todo este conjunto está embalado con una caja de cartón 100% reciclado de peso 140 g/m^2 automontable, junto a las piezas del robot. Esta caja garantiza una excelente calidad, resistencia y rigidez, asegurando el acondicionamiento y la protección eficaz del producto. El papel de burbuja tiene un grosor de 9 mm y la caja tiene una dimensión de 300 x 300 x 150 mm.

6.2 Transporte

Todos los productos se transportan en palets de madera que garantizan la inmovilidad transversal y longitudinal de la carga. Están protegidos con un plástico opaco fijado con flejes, de esta forma el plástico de protección no se podrá volar o soltar del producto y evitará la incidencia del sol en el mismo. Los flejes empleados en el embalaje son de poliéster reforzado.

La mercancía se descarga cerca del lugar donde tenga que ser entregado el pedido, de forma que pueda trasladarse con facilidad al destinatario. El producto debe estar protegido de acciones o elementos que puedan dañarlo, además se ha de evitar que quede apoyado sobre superficies estrechas y con posibilidad de sufrir caídas.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Documento número 4:

Planos

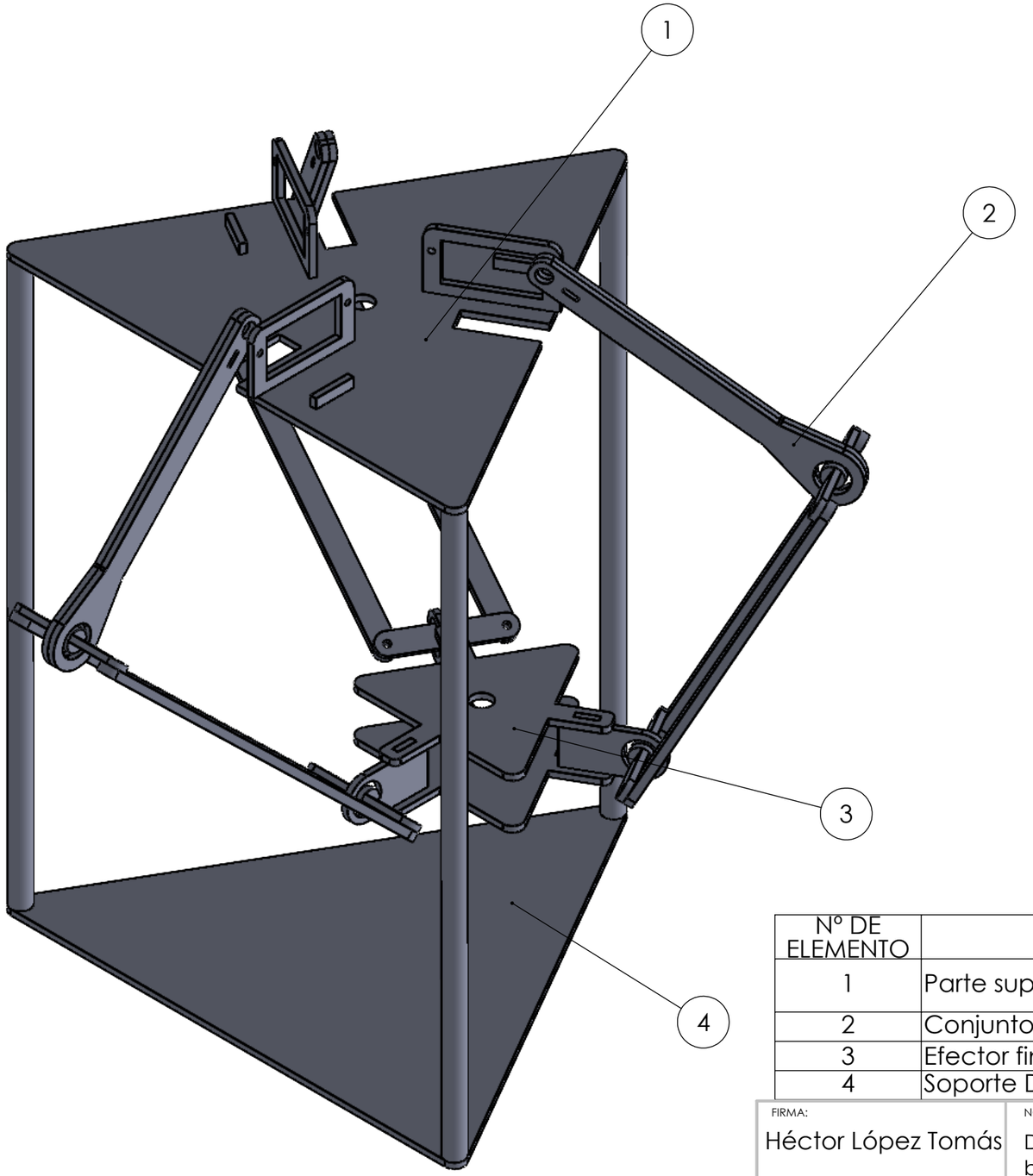
Diseño y programación de un Robot Delta basado en Arduino con fines académicos

Trabajo final de grado en Ingeniería Electrónica Industrial y Automática

Autor: Héctor López Tomás

Tutor: Leopoldo Armesto Ángel

Curso académico 2021-2022



Nº DE ELEMENTO	Nº DE PIEZA	MATERIAL	CANTIDAD
1	Parte superior para MG996R	Madera mdf	1
2	Conjunto brazo	Madera mdf	3
3	Efector final	Madera mdf	1
4	Soporte Delta	Madera mdf	1

FIRMA: Héctor López Tomás

NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos

MATERIAL: Madera mdf

ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño

TIPO DE DOCUMENTO: Dibujo ensamblaje

UNIDADES: cm

ORIENTACIÓN:

ESCALA: 1:2

REVISIÓN: A

UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA

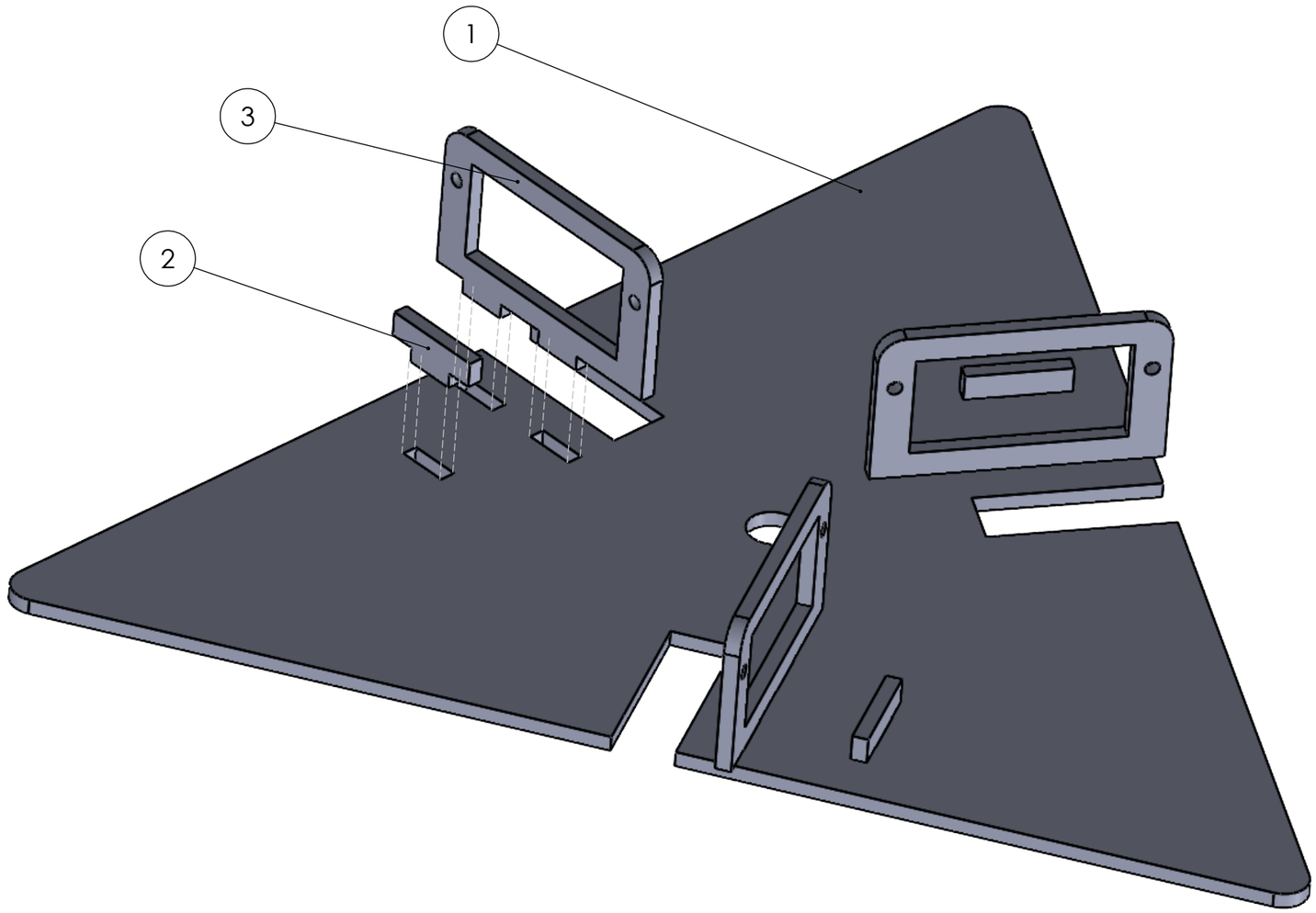
TÍTULO: Robot Delta 3 gdl

FORMATO: A3

Nº PLANO: 0

FECHA DE EDICIÓN: 10/06/22

HOJA: 1/1



N.º DE ELEMENTO	N.º DE PIEZA	MATERIAL	CANTIDAD
1	Base superior	Madera mdf	1
2	Soporte MG996R	Madera mdf	3
3	Sujeción MG996R	Madera mdf	3

FIRMA: Héctor López Tomás

ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA

NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos

TIPO DE DOCUMENTO: Dibujo ensamblaje

TÍTULO: Parte superior Delta para MG996R

MATERIAL: Madera mdf

UNIDADES: cm

ORIENTACIÓN:

ESCALA: 1:1

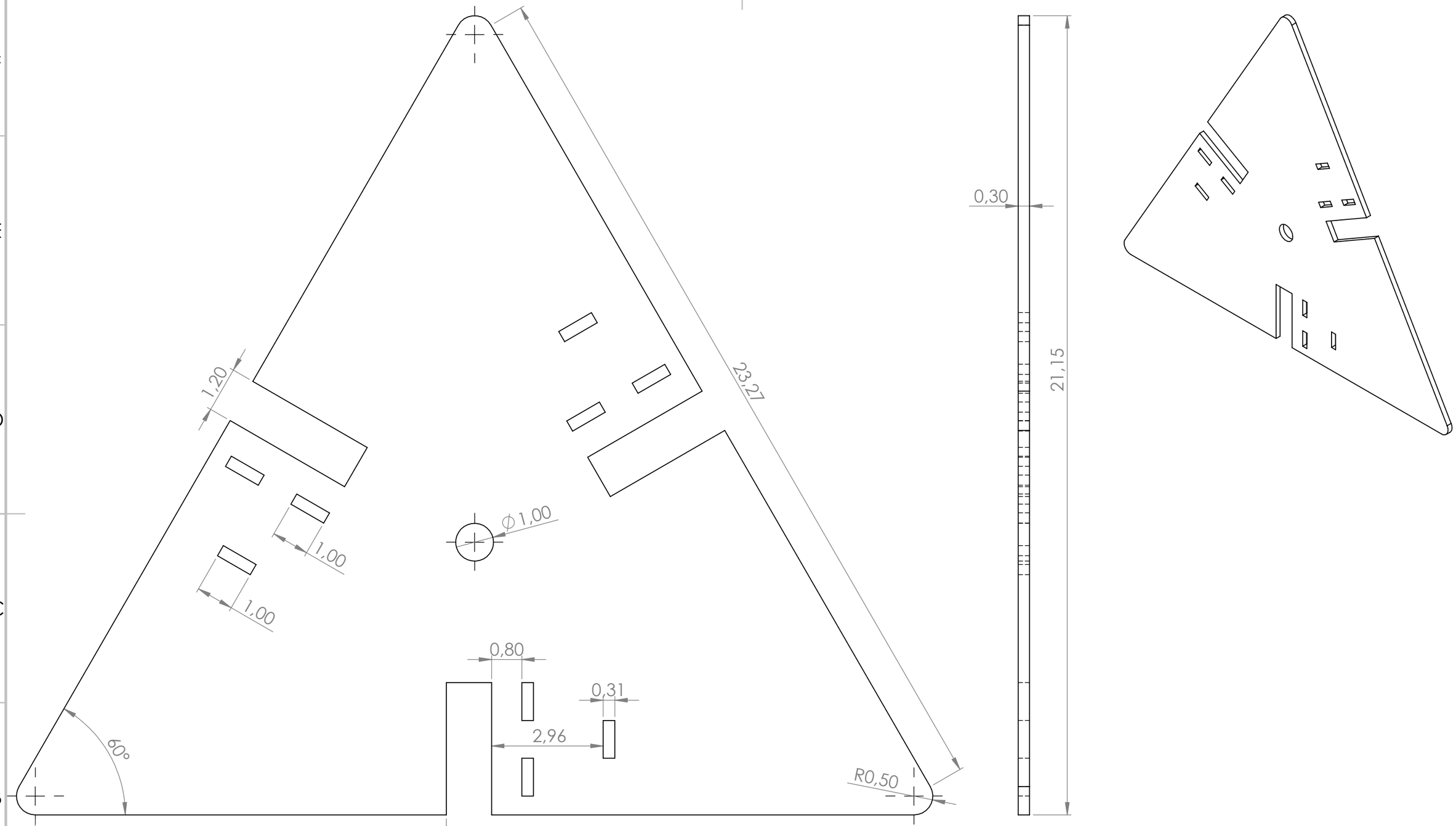
REVISIÓN: A

FORMATO: A3

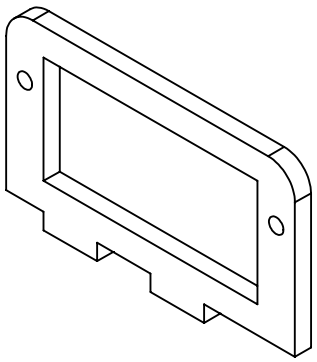
Nº PLANO: 1

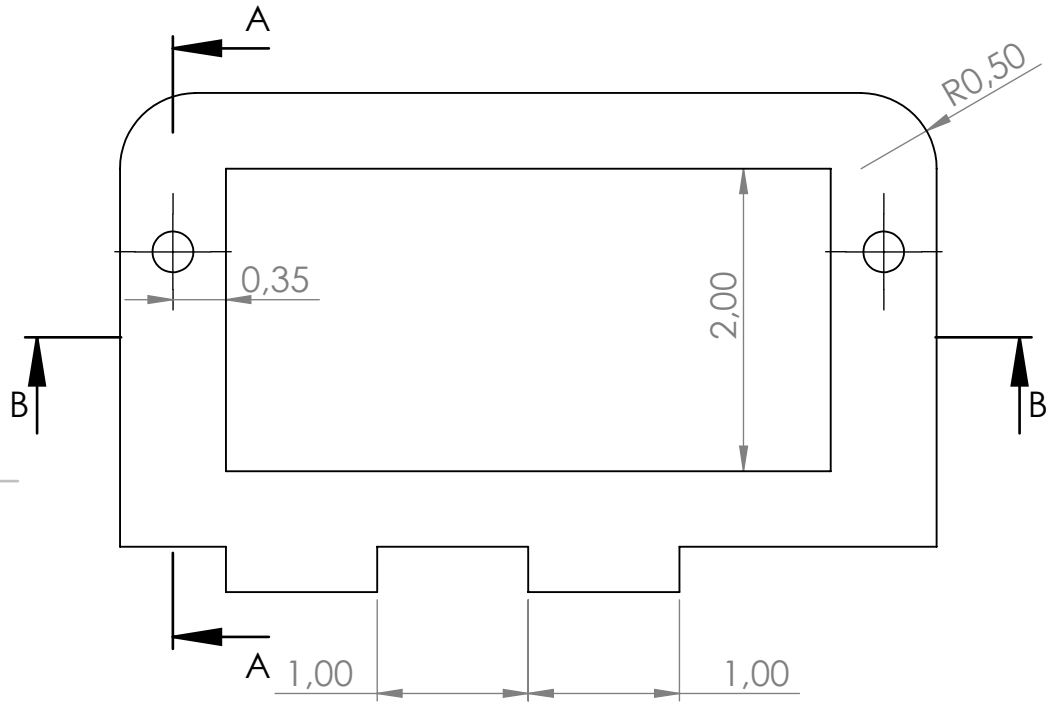
FECHA DE EDICIÓN: 10/06/22

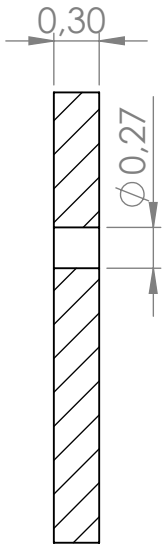
HOJA: 1/1



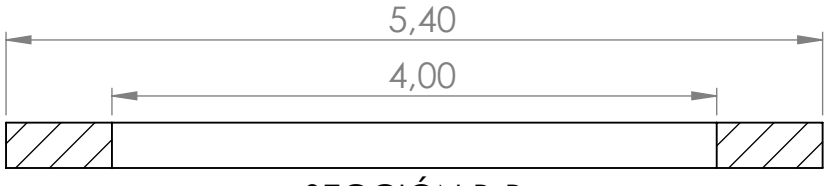
FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm / ORIENTACIÓN: 
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Base superior	ESCALA: 1:1 / REVISIÓN: A
		FORMATO: A3 / Nº PLANO: 1.1 / 2.1
		FECHA DE EDICIÓN: 10/06/22 / HOJA: 1/1





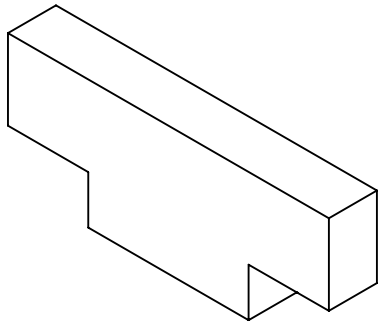


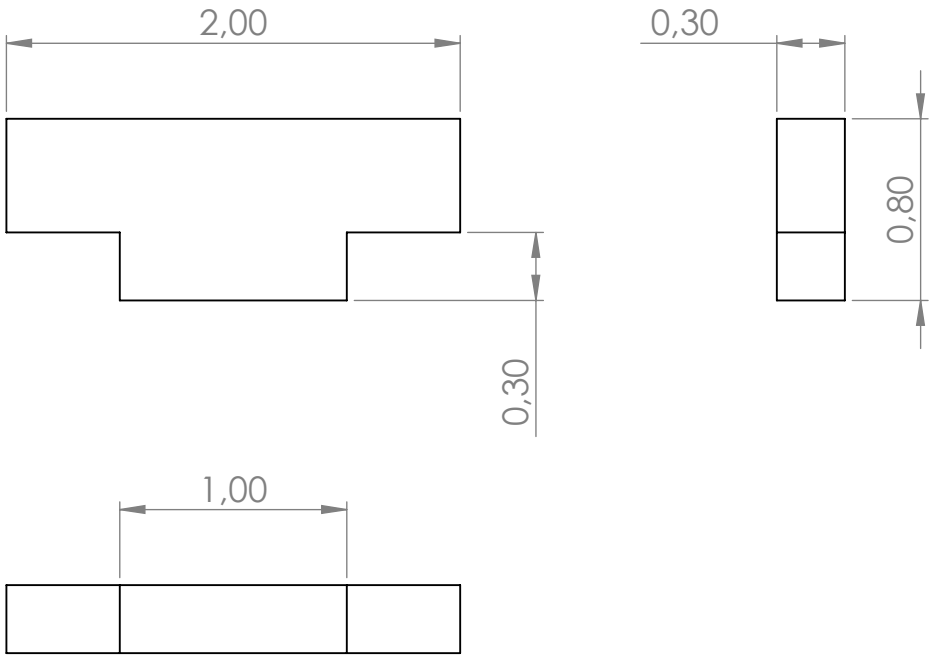
SECCIÓN A-A
ESCALA 2 : 1



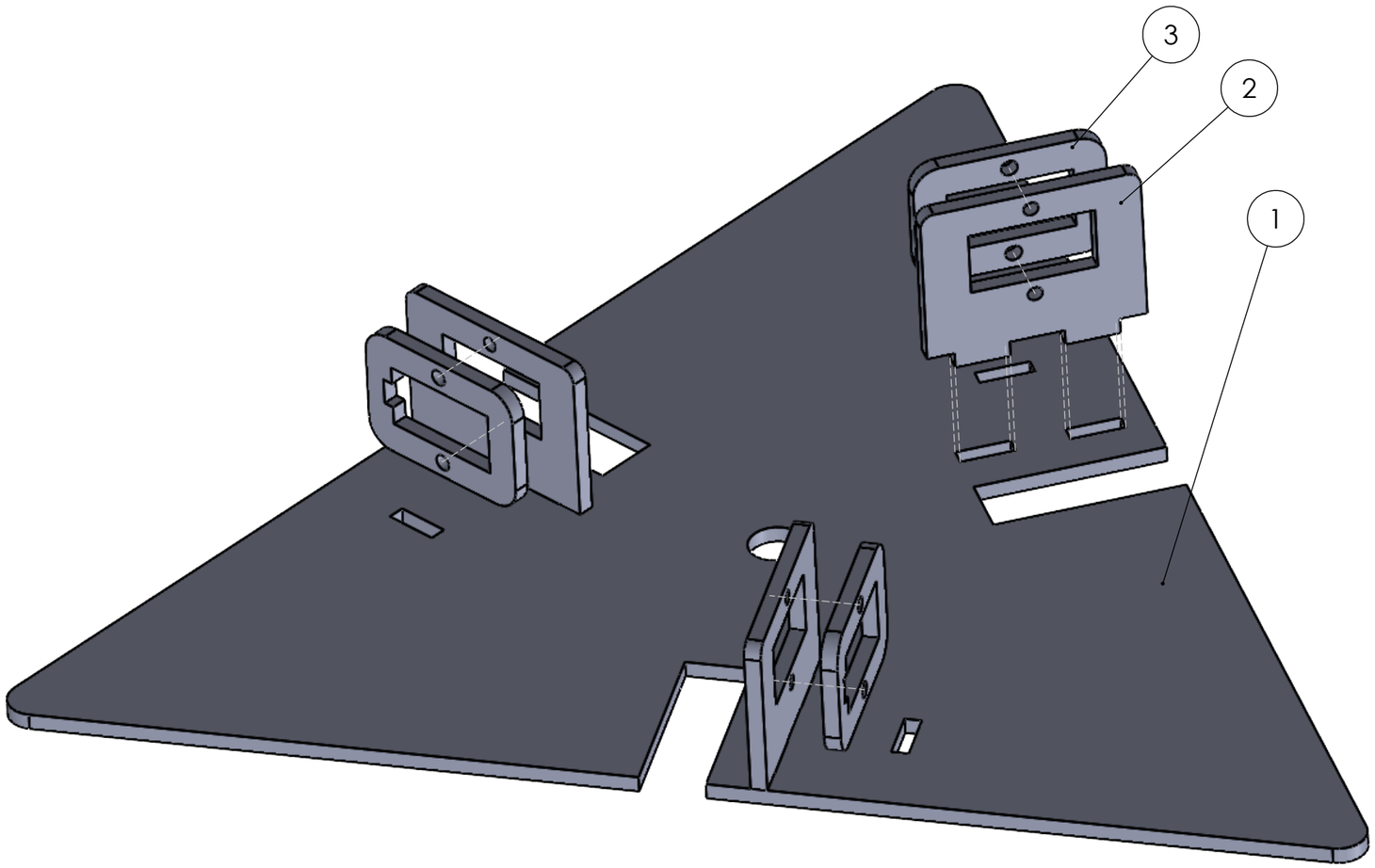
SECCIÓN B-B
ESCALA 2 : 1

FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
		ESCALA: 2:1	REVISIÓN: A
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Sujeción servomotor MG 996R	FORMATO: A4	Nº PLANO: 1.2
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1



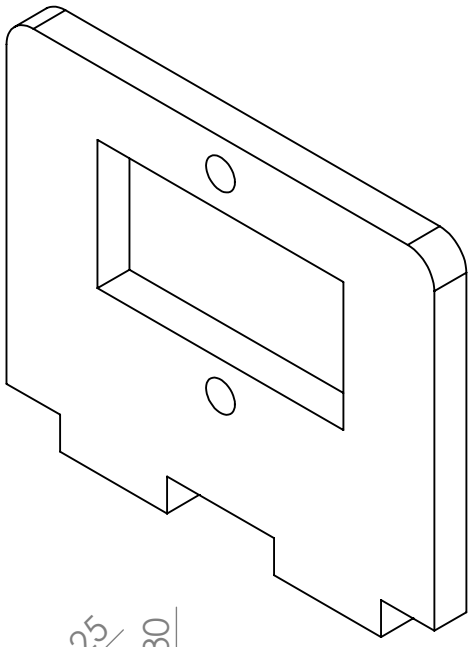


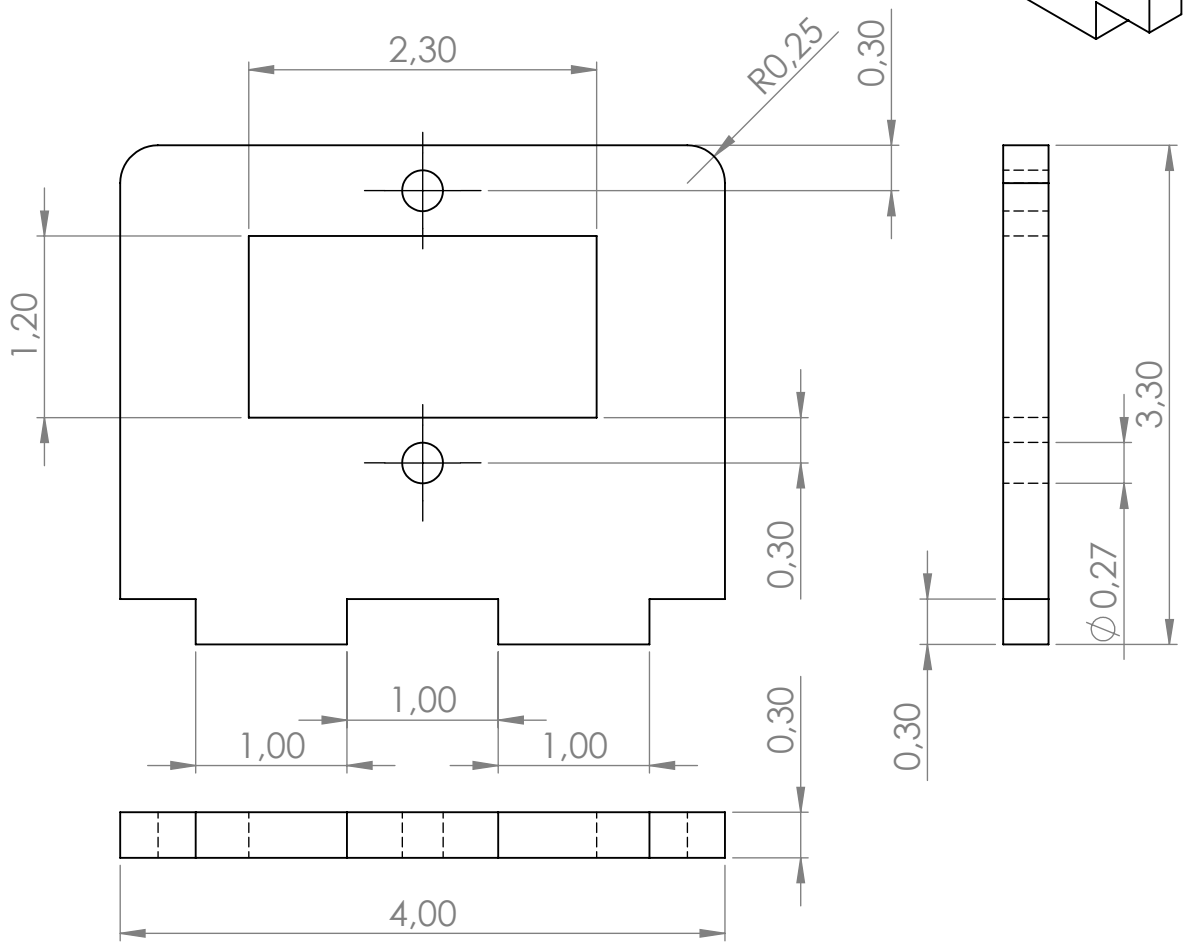
FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
		ESCALA: 3:1	REVISIÓN: A
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Soporte servomotor MG 996R	FORMATO: A3	Nº PLANO: 1.3
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1



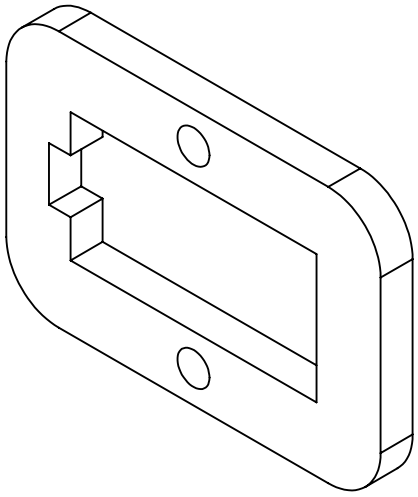
N.º DE ELEMENTO	N.º DE PIEZA	MATERIAL	CANTIDAD
1	Base superior	Madera mdf	1
2	Sujeción MG90s	Madera mdf	3
3	Agarre MG90s	Madera mdf	3

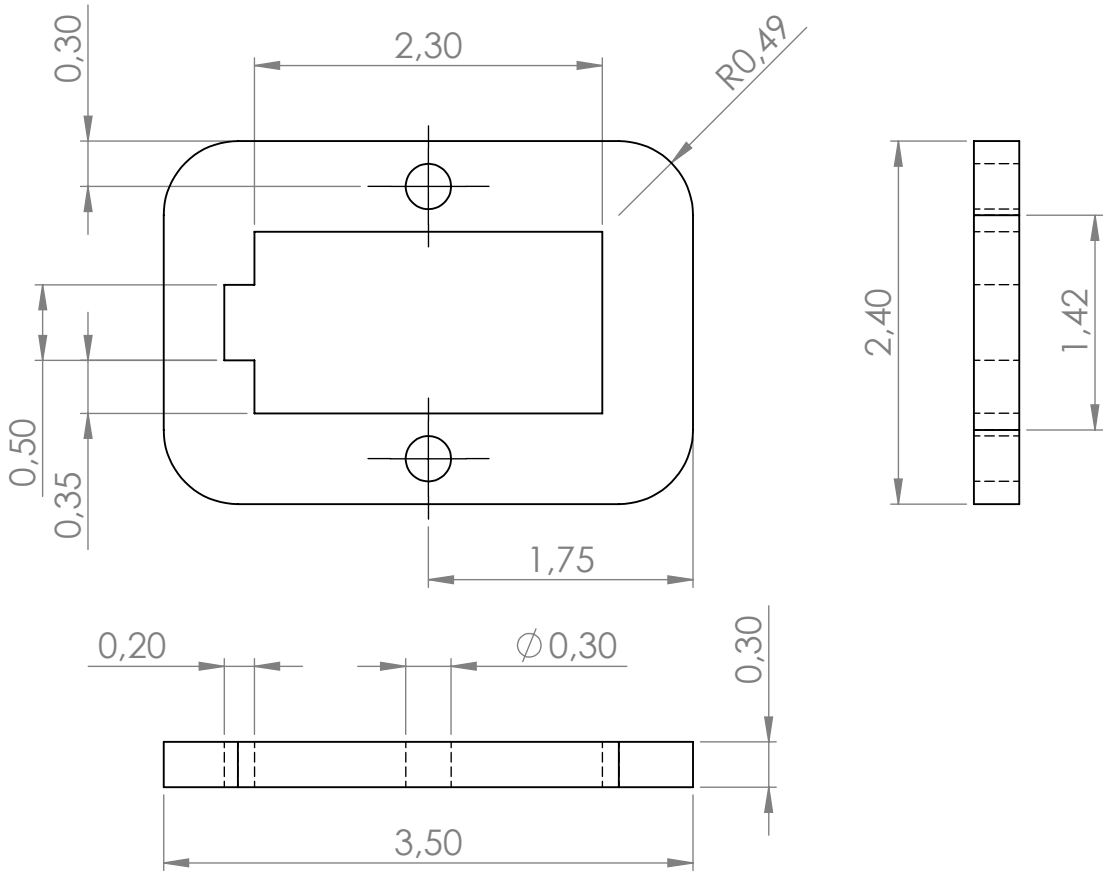
FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo ensambleje	UNIDADES: cm	ORIENTACIÓN:
		ESCALA: 1:1	REVISIÓN: A
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Parte superior Delta para MG90s	FORMATO: A3	Nº PLANO: 2
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1



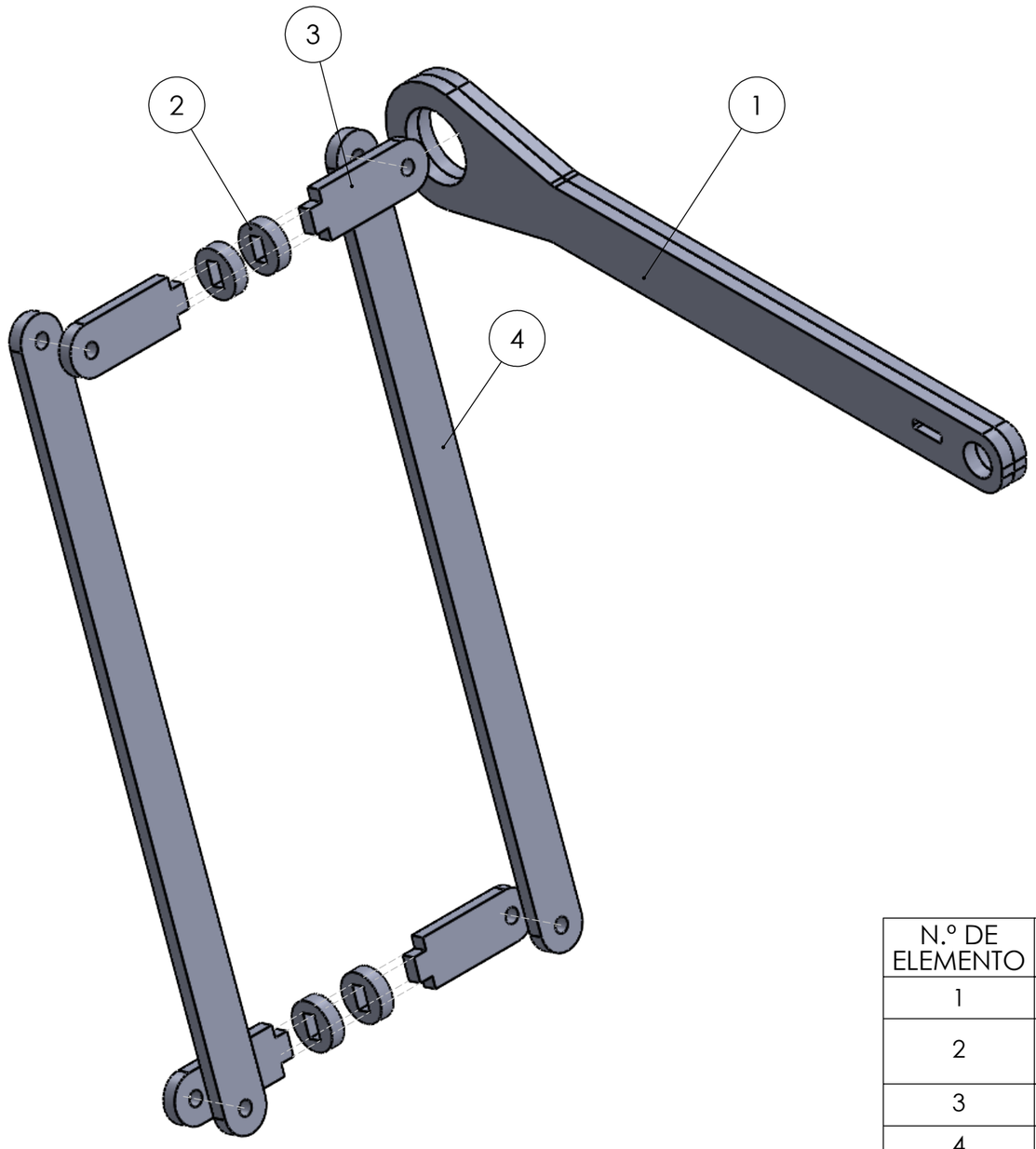


FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Sujeción servomotor MG 90s	ESCALA: 2:1	REVISIÓN: A
		FORMATO: A3	Nº PLANO: 2.2
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1





FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Agarre servomotor MG 90s	ESCALA: 2:1	REVISIÓN: A
		FORMATO: A3	Nº PLANO: 2.3
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1



N.º DE ELEMENTO	N.º DE PIEZA	MATERIAL	CANTIDAD
1	Brazo actuador	Madera mdf	2
2	Eslabón corto paralelogramo 1	Madera mdf	4
3	Eslabón corto paralelogramo 2	Madera mdf	4
4	Eslabón largo paralelogramo	Madera mdf	2

FIRMA: Héctor López Tomás

NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos

MATERIAL: Madera mdf

ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño

TIPO DE DOCUMENTO: Dinujo ensamblaje

UNIDADES: cm

ORIENTACIÓN:

ESCALA: 1:1

REVISIÓN: A

UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA

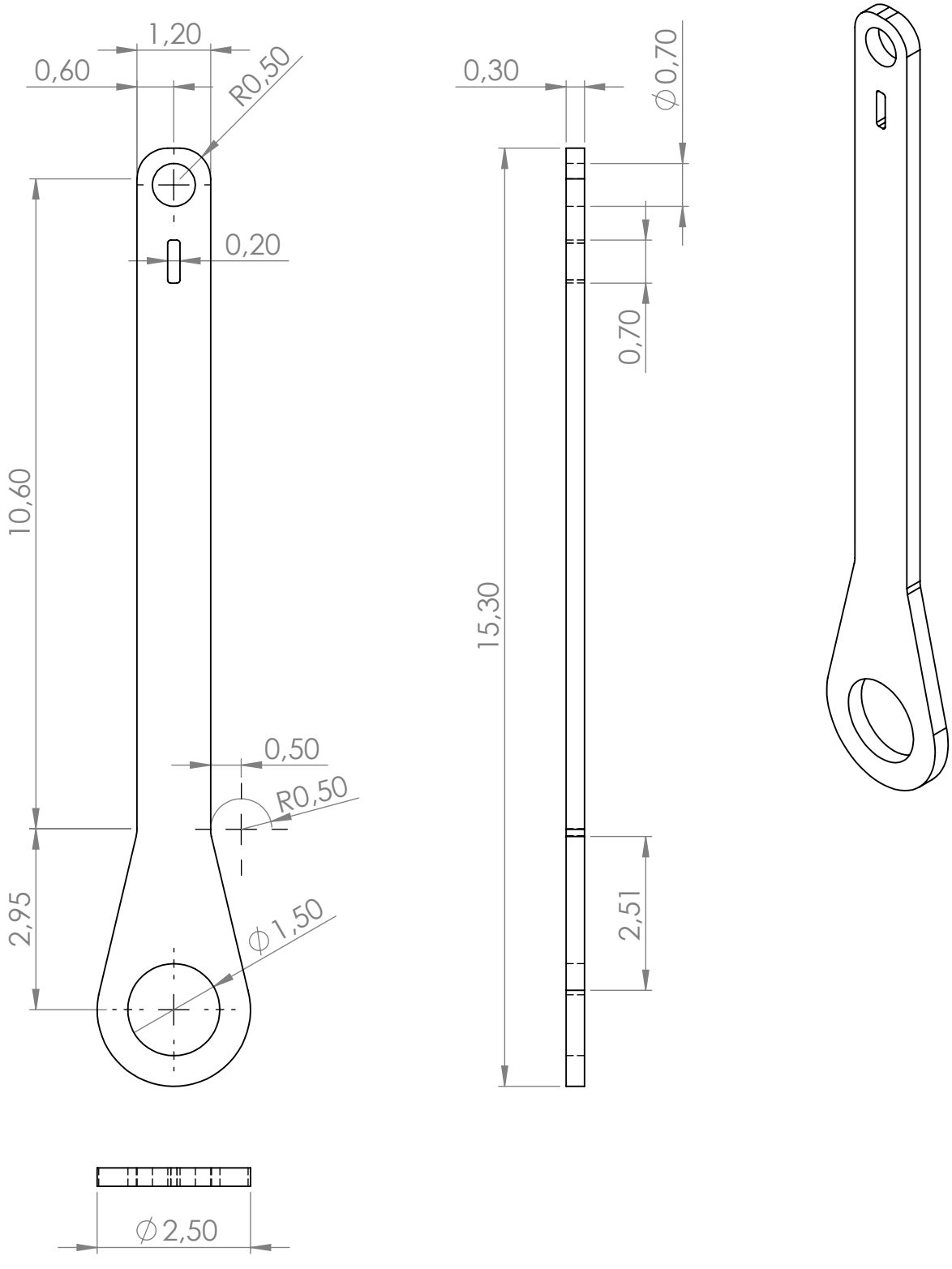
TÍTULO: Conjunto brazo

FORMATO: A3

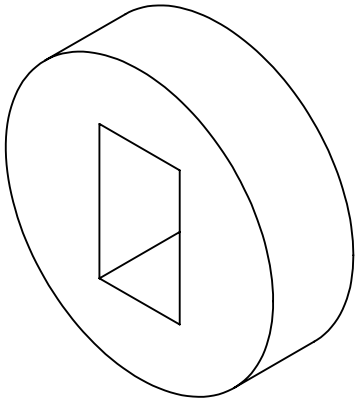
Nº PLANO: 3

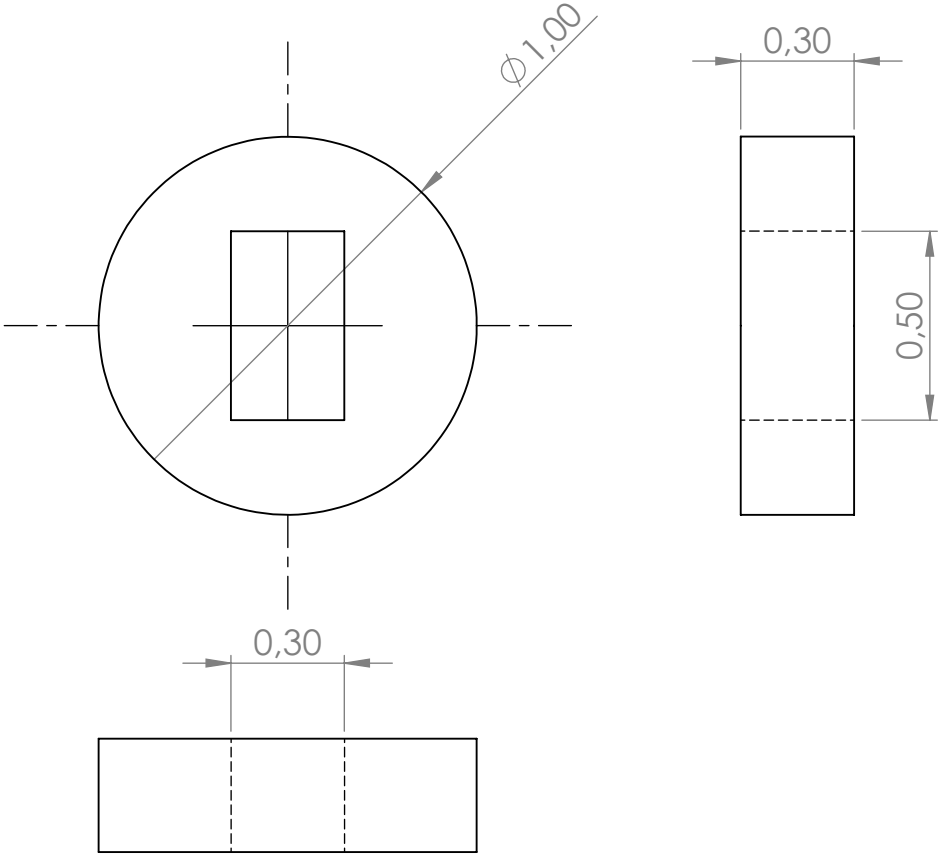
FECHA DE EDICIÓN: 10/06/22

HOJA: 1/1

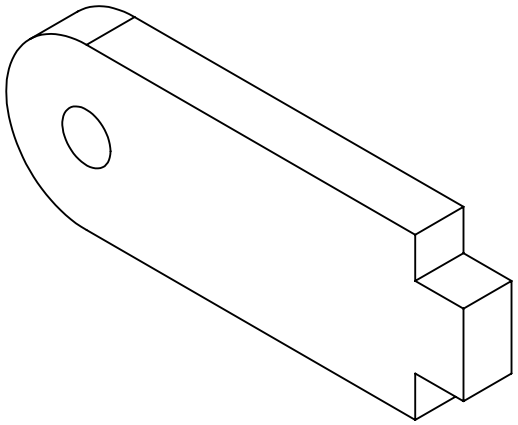


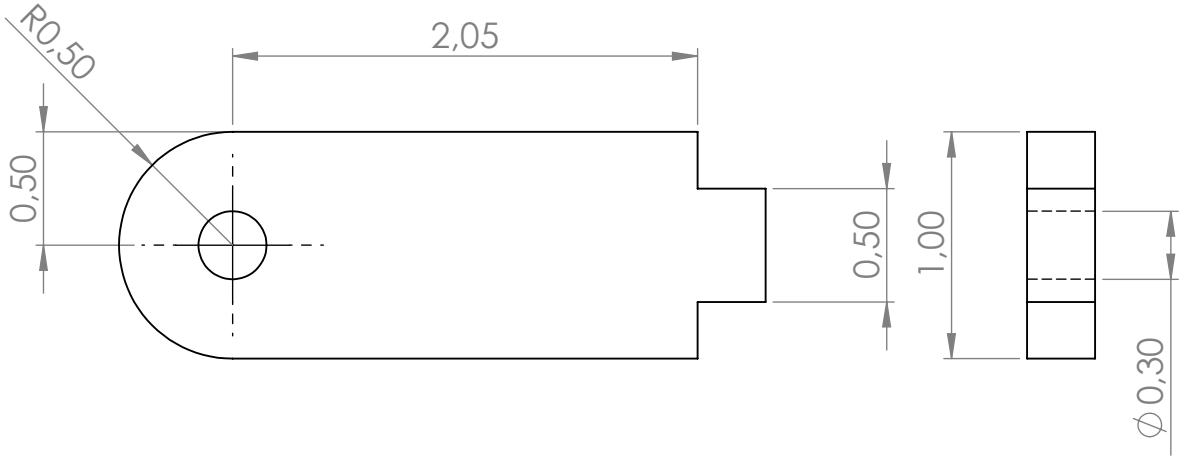
FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
		ESCALA: 1:1	REVISIÓN: A
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Brazo actuador	FORMATO: A3	Nº PLANO: 3.1
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1



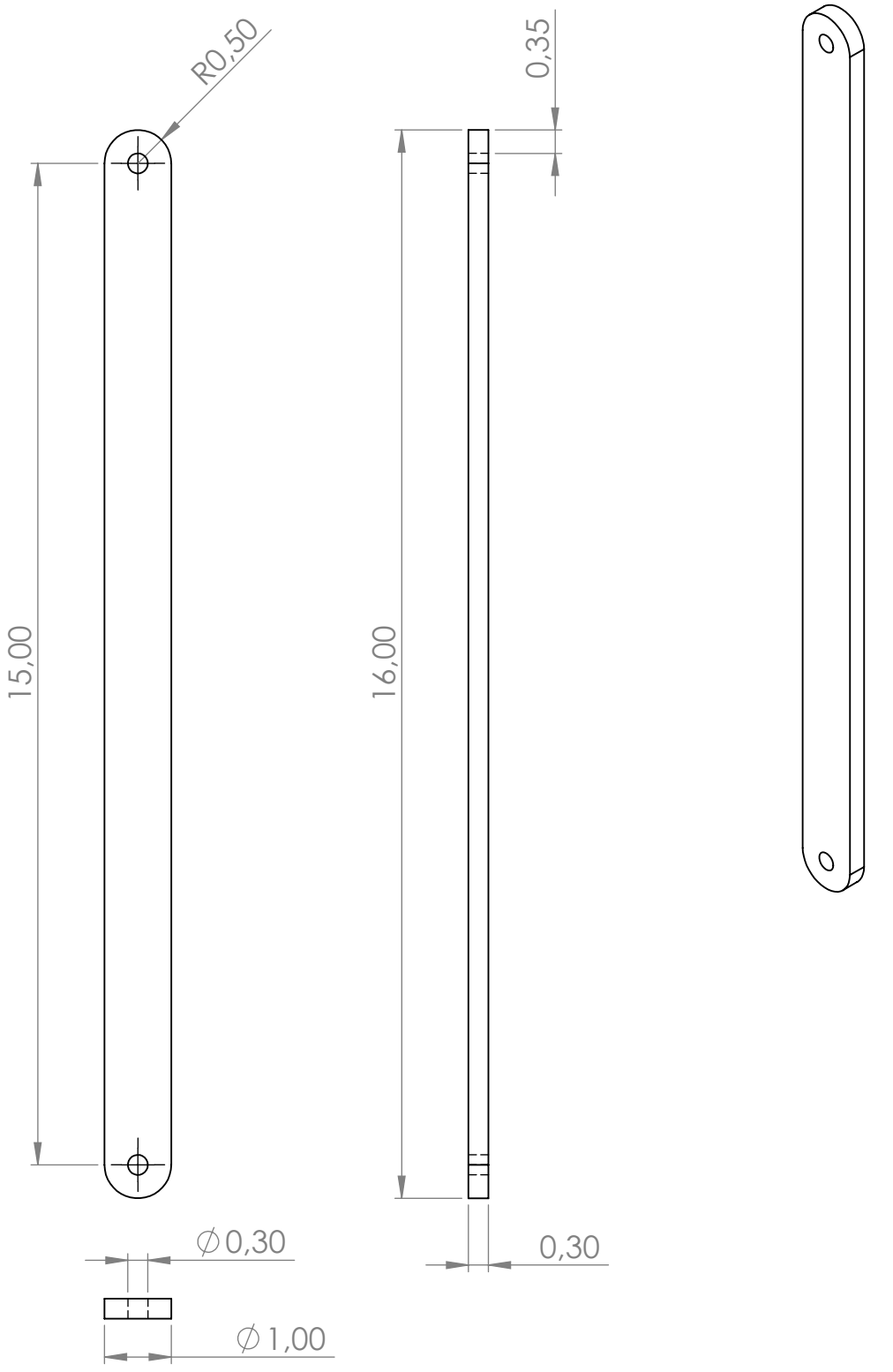


FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
		ESCALA: 5:1	REVISIÓN: A
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Eslabón corto paralelogramo 1	FORMATO: A3	Nº PLANO: 3.2
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1

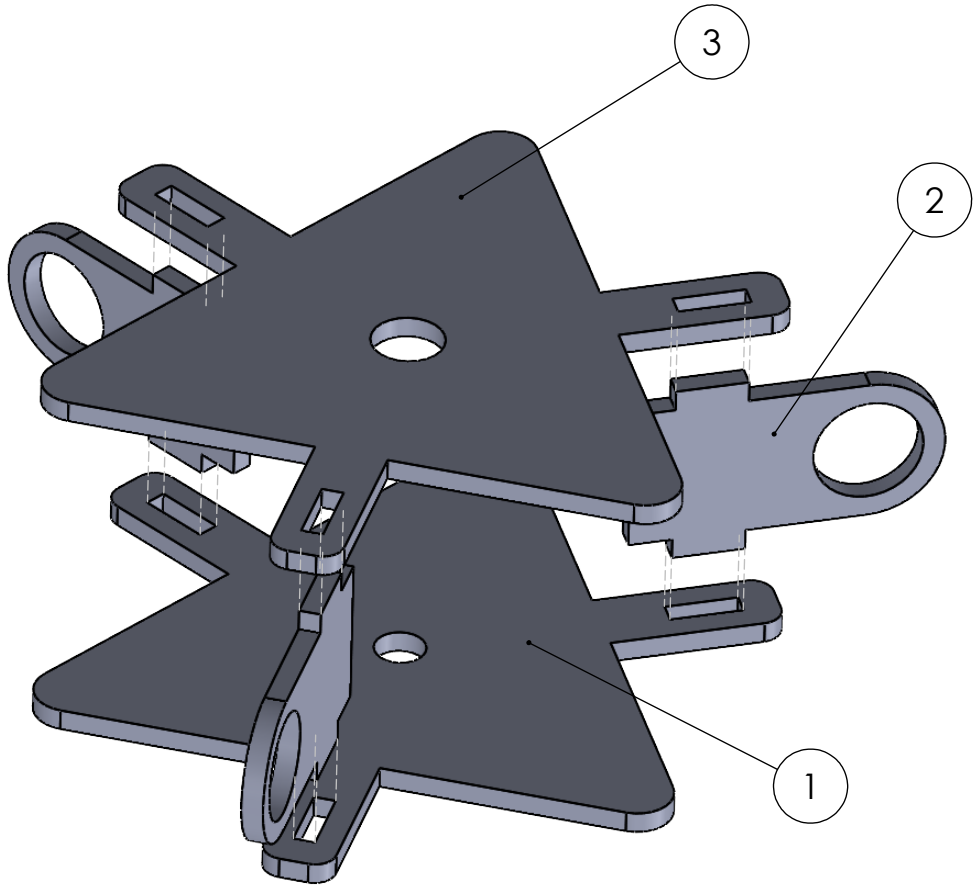




FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
		ESCALA: 3:1	REVISIÓN: A
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Eslabón corto paralelogramo 2	FORMATO: A3	Nº PLANO: 3.3
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1



FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
		ESCALA: 1:1	REVISIÓN: A
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Eslabón largo paralelogramo	FORMATO: A3	Nº PLANO: 3.4
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1



N.º DE ELEMENTO	N.º DE PIEZA	MATERIAL	CANTIDAD
1	Efector final plataforma inferior	Madera mdf	1
2	Sujeción efector final	Madera mdf	3
3	EFector final plataforma superioir	Madera mdf	1

FIRMA:
Héctor López Tomás

NOMBRE DEL PROYECTO:
Diseño y Programación de un Robot Delta basado en Arduino con fines académicos

MATERIAL:
Madera mdf

ESCUELA:
Escuela Técnica Superior de Ingeniería del Diseño

TIPO DE DOCUMENTO:
Dibujo ensamblaje

UNIDADES:
cm

ORIENTACIÓN:

ESCALA:
1:1

REVISIÓN:
A

UNIVERSIDAD:
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

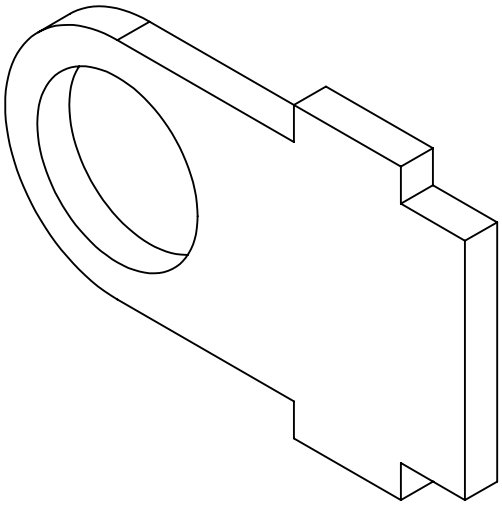
TÍTULO:
Efector final

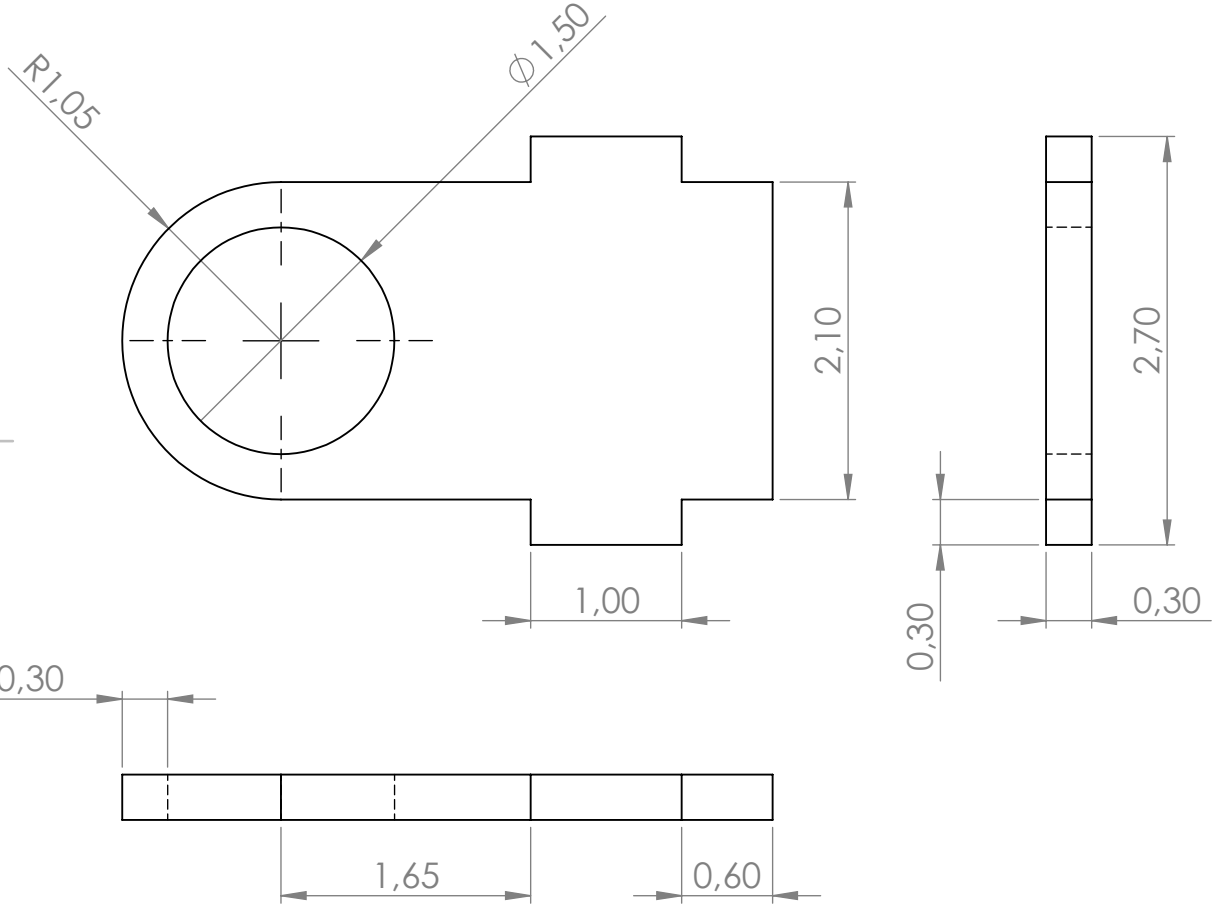
FORMATO:
A4

Nº PLANO:
4

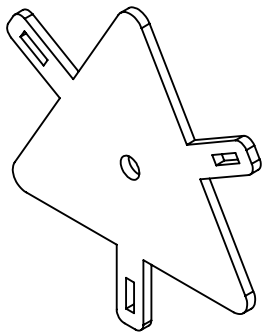
FECHA DE EDICIÓN:
10/06/22

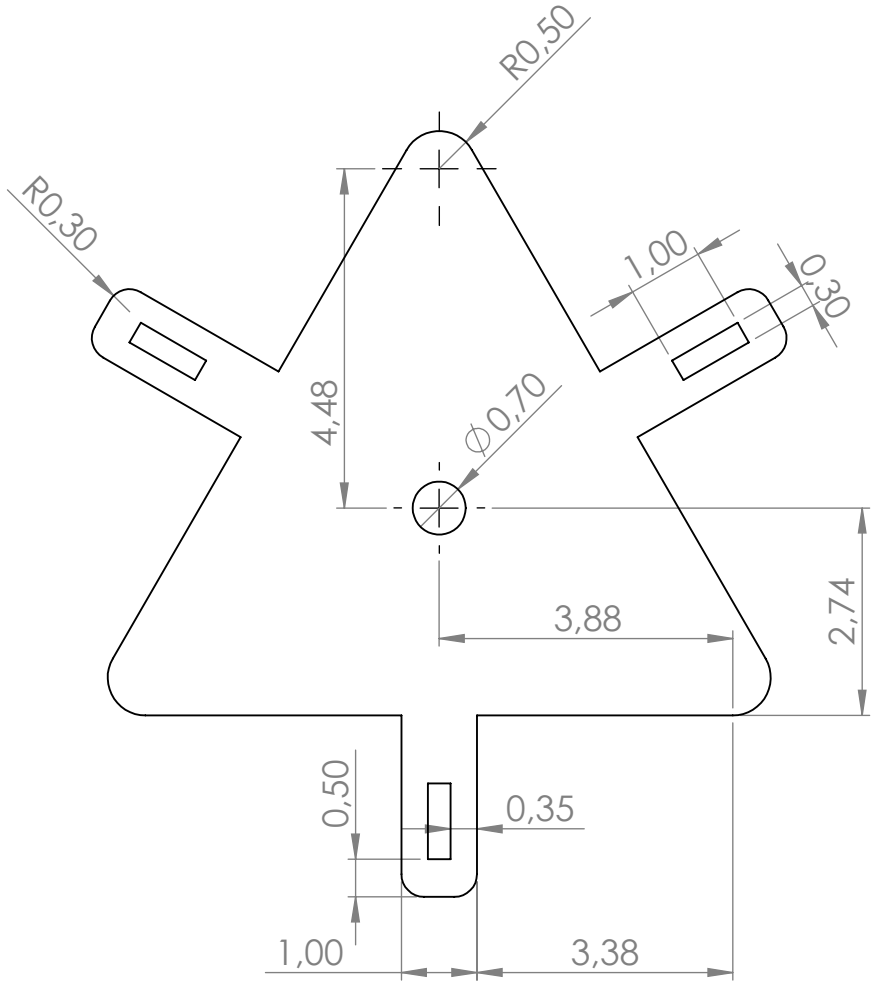
HOJA:
1/1

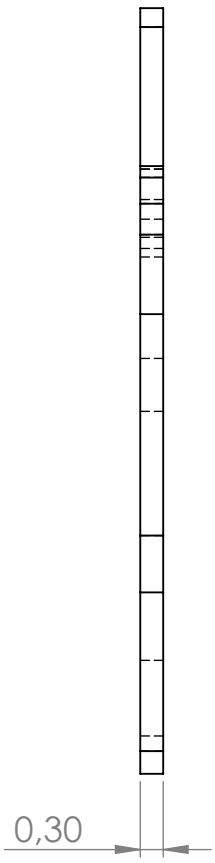


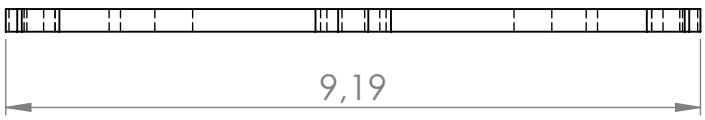


FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Sujeción efector final	ESCALA: 2:1	REVISIÓN: A
		FORMATO: A3	Nº PLANO: 4.1
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1

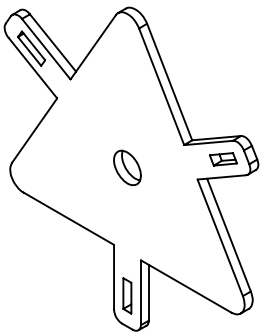


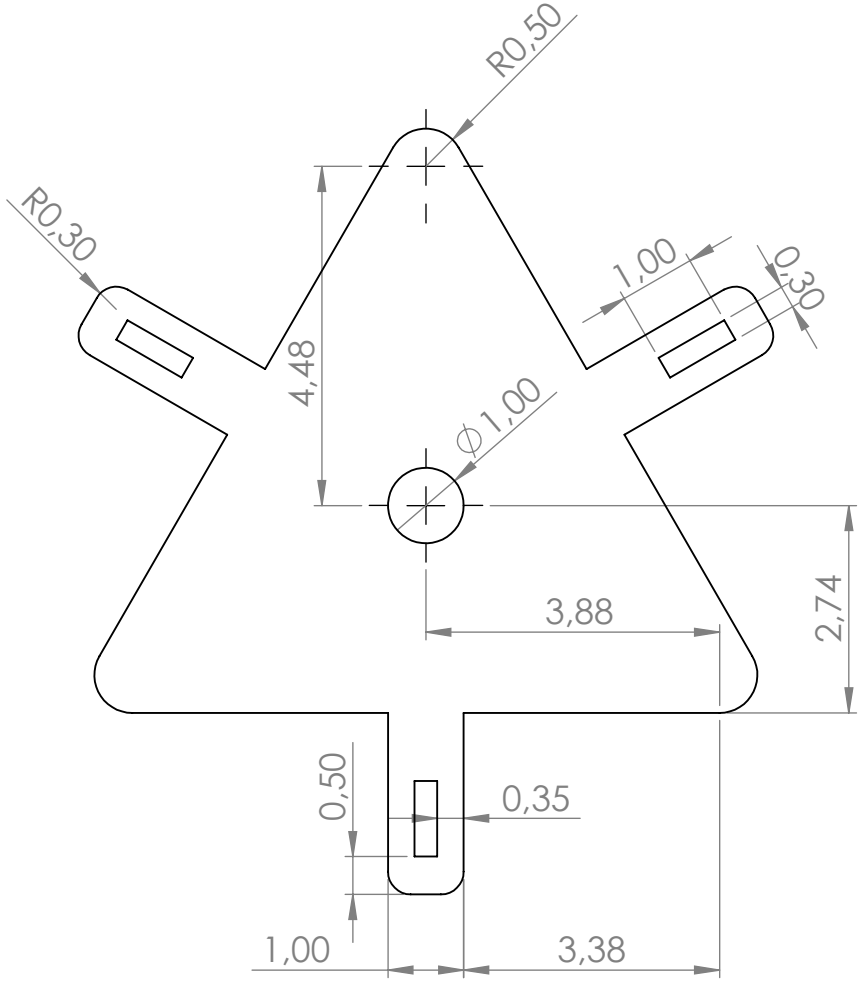


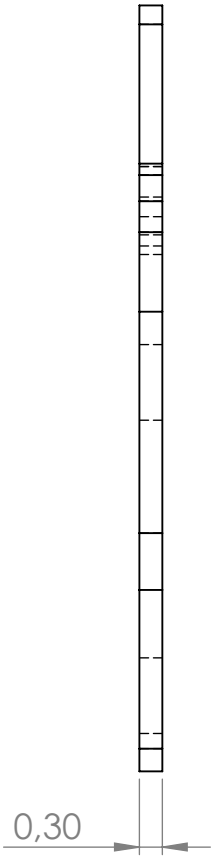


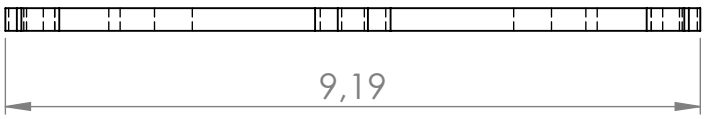


FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Efector final plataforma inferior	ESCALA: 1:1	REVISIÓN: A
		FORMATO: A3	Nº PLANO: 4.2
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1

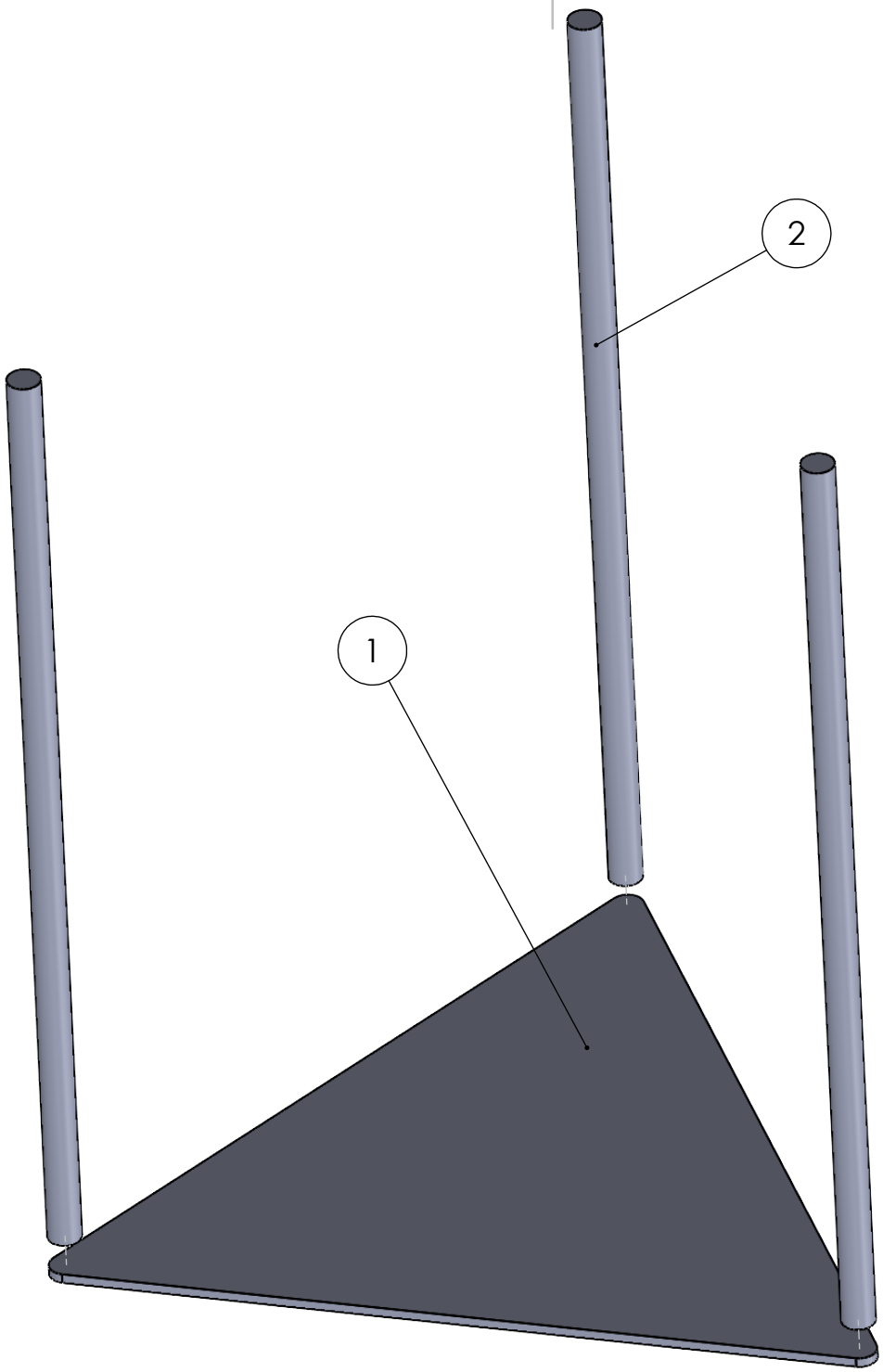








FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Efector final plataforma superior	ESCALA: 1:1	REVISIÓN: A
		FORMATO: A3	Nº PLANO: 4.3
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1



N.º DE ELEMENTO	N.º DE PIEZA	MATERIAL	CANTIDAD
1	Base inferior	Madera mdf	1
2	Pilar	Madera mdf	3

FIRMA:
Héctor López Tomás

NOMBRE DEL PROYECTO:
Diseño y Programación de un Robot Delta basado en Arduino con fines académicos

MATERIAL:
Madera mdf

ESCUELA:
Escuela Técnica Superior de Ingeniería del Diseño

TIPO DE DOCUMENTO:
Dibujo ensamblaje

UNIDADES:
cm

ORIENTACIÓN:

ESCALA:
1:2

REVISIÓN:
A

UNIVERSIDAD:
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

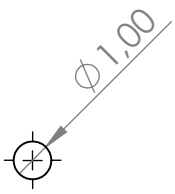
TÍTULO:
Soporte Delta

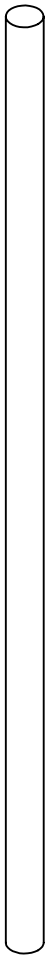
FORMATO:
A4

Nº PLANO:
5

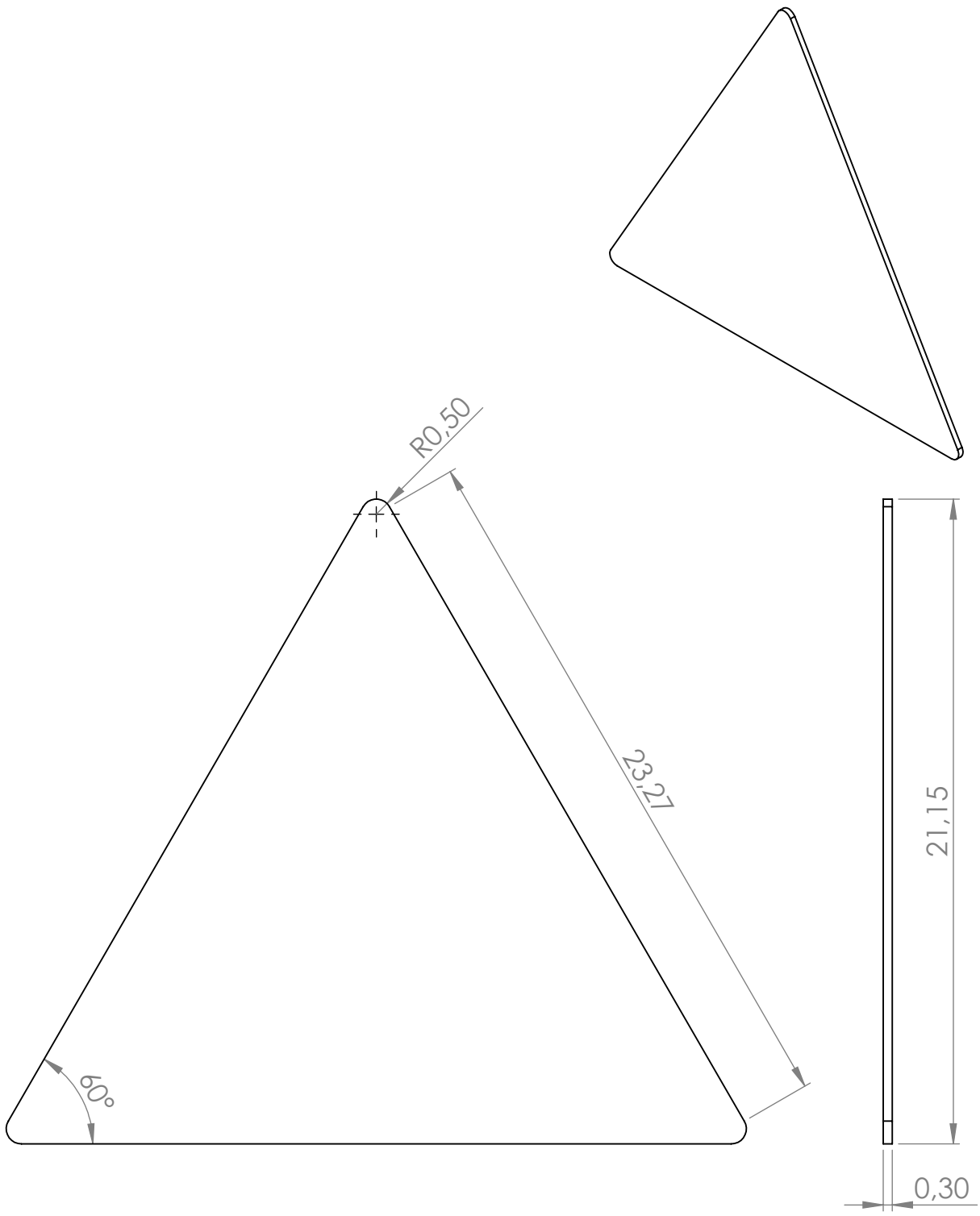
FECHA DE EDICIÓN:
10/06/22

HOJA:
1/1





FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Pilar	ESCALA: 1:2	REVISIÓN: A
		FORMATO: A4	Nº PLANO: 5.1
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1



FIRMA: Héctor López Tomás	NOMBRE DEL PROYECTO: Diseño y Programación de un Robot Delta basado en Arduino con fines académicos	MATERIAL: Madera mdf	
ESCUELA: Escuela Técnica Superior de Ingeniería del Diseño	TIPO DE DOCUMENTO: Dibujo de subconjunto	UNIDADES: cm	ORIENTACIÓN:
UNIVERSIDAD: UNIVERSITAT POLITÈCNICA DE VALÈNCIA	TÍTULO: Base inferior	ESCALA: 1:2	REVISIÓN: A
		FORMATO: A4	Nº PLANO: 5.2
		FECHA DE EDICIÓN: 10/06/22	HOJA: 1/1

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Documento número 4:

Presupuesto

Diseño y programación de un Robot Delta basado en Arduino con fines académicos

Trabajo final de grado en Ingeniería Electrónica Industrial y Automática

Autor: Héctor López Tomás

Tutor: Leopoldo Armesto Ángel

Curso académico 2021-2022

COMPOSICIÓN DEL PRODUCTO

MATERIALES				
Uds	Denominación	Cantidad	Precio(€)	Total(€)
MATERIAS PRIMAS				
m2	Madera mdf de grosor 3 mm	0,18	4,8	0,864
u	Bote pegamento	1	1,54	1,540
PRODUCTO INDUSTRIAL				0,000
u	Arduino nano	1	4,5	4,500
u	10 x Tornillas de nylon M3x10mm	2	0,05	0,100
u	20 x Tuercas de nylon M3	1	0,05	0,050
u	20 x Arandelas Acero M3	1	0,05	0,050
u	7 x Rodamientos	1	3	3,000
u	Batería Li-Ion 18650 2600 mAh	1	5,5	5,500
u	Simple Powerbank 18650	1	9,99	9,990
u	Servomotor MG995R Tower Pro	3	6,93	20,790
u	I/O Extension Shield para Arduino nano	1	4,5	4,500
u	5 x Tornillos metálicos para madera M2	1	0,05	0,050
u	Cable mini-USB (30cm)	1	2	2,000
u	Cable DuPont 10 cm Macho-Hembra	2	0,1	0,200
		SUBTOTAL DE MATERIALES		**52,934**

FABRICACIÓN				
Uds	Denominación	Cantidad	Precio(€)	Total(€)
CORTADORA LÁSER				
h	Corte del conjunto total de piezas	0,1	67	6,7
		SUBTOTAL DE FABRICACIÓN		**6,7**

COSTE ELABORACIÓN Y OBTENCIÓN	**59,634**

Para el proceso de cortadora láser se ha estimado dicho valor para situaciones en las que se realicen, a la compañía pertinente, pedidos superiores a 150 unidades, ya que, si se calculara el coste para la fabricación de una única unidad, el precio de coste ascendería 30 euros.

MATERIALES UTILIZADOS EN EL EMBALAJE FINAL DEL PRODUCTO

MATERIALES				
Uds	Denominación	Cantidad	Precio(€)	Total(€)
MATERIA PRIMA				
m2	Rollo de papel de burbujas	4	0,160	0,640
m	Rollo cinta adhesiva embalaje	5	0,040	0,200
u	Film transparente de embalaje	0,01	9,000	0,090
u	Material de relleno soft-fill	1	0,270	0,270
m	Cinta de fleje	4	0,420	1,680
PRODUCTO INDUSTRIAL				
u	Caja de cartón	1	1,700	1,700
		SUBTOTAL MATERIALES		**4,580**

MANO DE OBRA				
Uds	Denominación	Cantidad	Precio(€)	Total(€)
MANO DE OBRA DIRECTA				
h	Operario encargado del proceso de embalaje	0,2	12,000	2,400
h	Auxiliar de máquina en línea de producción	0,2	10,000	2,000
		SUBTOTAL MANO DE OBRA		**4,400**

COSTE EMBALAJE DEL PRODUCTO	**8,980**

RESUMEN PRESUPUESTO

Denominación	Coste(€)
COSTE ELABORACIÓN Y OBTENCIÓN PRODUCTO	59,634
COSTE EMBALAJE DEL PRODUCTO	8,98
COSTE TOTAL ELABORACIÓN PRODUCTO	**68,614**
IVA(21%)	14,409
PRECIO FINAL DE VENTA	**83,023**

PRESUPUESTO DISEÑO DEL PRODUCTO

CONCEPTO				
Uds	Denominación	Cantidad	Precio(€)	Total(€)
h	Servicio de ingeniero cualificado	250	55,00	13750,00
u	Software de diseño SolidWorks durante 3 meses	1	1010,00	1010,00
u	Software de simulación CoppeliaSim	1	0,00	0,00
u	Software aplicación Matlab 2020b	1	800,00	800,00
u	Software Arduino IDE	1	0,00	0,00
u	Software Meshmixer	1	0,00	0,00
		SUBTOTAL PRESUPUESTO		**15560,00**
			IVA(21%)	3267,60
		TOTAL PRESUPUESTO		**18827,60**

Sin contemplar el coste del diseño del producto, el presupuesto final asciende a un total de ochenta y tres euros y tres céntimos (83.03 €).

Junio 2022, Valencia.

Equipo redactor: Héctor López Tomás

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Documento número 6:

Código fuente

Diseño y programación de un Robot Delta basado en Arduino con fines académicos

Trabajo final de grado en Ingeniería Electrónica Industrial y Automática

Autor: Héctor López Tomás

Tutor: Leopoldo Armesto Ángel

Curso académico 2021-2022

ÍNDICE

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

MATLAB

Documento número 6: Código fuente

Cinemática inversa:

```
syms q1 q2 q3 l1 l2 real


%Basic transformations
trans=@(t) [eye(3) t;zeros(1,3) 1];
rotx=@(ang) [1 0 0 0;0 cos(ang) -sin(ang) 0;0 sin(ang) cos(ang) 0;0 0
0 1];
rotz=@(ang) [cos(ang) -sin(ang) 0 0;sin(ang) cos(ang) 0 0;0 0 1 0;0 0
0 1];

%DH Transformation (Revolute joint)
DHR=@(q,p)
rotz(q+p(1))*trans([0;0;p(2)])*trans([p(3);0;0])*rotx(p(4));

%RRR Robot
A01=DHR(q1,[0;0;l1;0]);
A12=DHR(q2,[0;0;0;-pi/2]);
A23=DHR(q3,[0;0;l2;0]);

A03=A01*A12*A23;

vpa(simplify(A03),3)

syms x y z real
q3=asin(-z/l2);

syms cq3 real
eq1=l2*cos(q1+q2)*cq3+l1*cos(q1)==x;
eq2=l2*sin(q1+q2)*cq3+l1*sin(q1)==y;
S=solve([eq1 eq2],[q1 q2])

subs(S.q1(1),{l1,l2,cq3,x,y},{150,150,1,150*sqrt(2),0})
simplify(S.q1(1))
```

Volumen trabajo:

```
XYZ=load('Delta4_workspace.xyz');
XYZ=XYZ(:,1:3);
tri3D=delaunayTriangulation(XYZ);
[F,P] = freeBoundary(tri3D);
tri3D_boundary=triangulation(F,P(:,1),P(:,2),P(:,3));
trimesh(tri3D_boundary.ConnectivityList,tri3D_boundary.Points(:,1),tri
3D_boundary.Points(:,2),tri3D_boundary.Points(:,3),'FaceAlpha',0.7,'Ed
geAlpha',0.7)
stlwrite(tri3D_boundary,'Delta4_3dworkspace.stl');
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

COPPELIASIM

Documento número 6: Código fuente

```
function jointSweep(Q,i,N,q)
    if (i<N) then
        if (steps[i]>0) then
            for ii=0,steps[i],1 do
                qi=qLim[i][2]*ii/steps[i]+qLim[i][1]
                if (cyclic[i]) then
                    qi=2*math.pi*ii/steps[i]-math.pi
                end
                q[i]=qi
                Q=jointSweep(Q,i+1,N,q)
            end
        else
            Q=jointSweep(Q,i+1,N,q)
        end
    else
        if (steps[i]>0) then
            for ii=0,steps[i],1 do
                qi=qLim[i][2]*ii/steps[i]+qLim[i][1]
                if (cyclic[i]) then
                    qi=2*math.pi*ii/steps[i]-math.pi
                end
                q[i]=qi
                qq={}
                for j=1,N,1 do
                    qq[j]=q[j]
                end
                table.insert(Q,qq)
            end
        else
            qq={}
            for j=1,N,1 do
                qq[j]=q[j]
            end
            table.insert(Q,qq)
        end
    end
    return Q
end

function sysCall_init()
    -- do some initialization here

    file = io.open('Delta4_workspace.xyz','w')
    io.output(file)

    jointHandles={-1,-1,-1}
    cyclic={-1,-1,-1}
    qLim={}
    N=3
    steps={130,130,130}
    q={0,0,0}
    revolution={-1,-1,-1}

    jointHandles[1]=sim.getObject('/Prismatic_joint_X')
    jointHandles[2]=sim.getObject('/Prismatic_joint_Y')
```

```
jointHandles[3]=sim.getObject('/Prismatic_joint_Z')

Objt1=sim.getObject('/Shape1')
Objt2=sim.getObject('/Shape2')
Objt3=sim.getObject('/Shape3')
Objt4=sim.getObject('/Shape4')
Objt5=sim.getObject('/Shape5')
Objt6=sim.getObject('/Shape6')

Objt7=sim.getObject('/Pilar1')
Objt8=sim.getObject('/Pilar2')
Objt9=sim.getObject('/Pilar3')

Objt10=sim.getObject('/Herramienta')

Herramienta=sim.getObject('/Sphere')
VT=sim.getObject('/Shape')
g_color1={250,1,1}
g_color2={1,1,1}

for i=1,N,1 do
    sim.setJointPosition(jointHandles[i],q[i])
    cyclic[i],qLim[i]=sim.getJointInterval(jointHandles[i])
end

Her=sim.getObject('/Dummy_Herramienta')
QHer={}
QHer=jointSweep(QHer,1,3,q)
maxPositions=#QHer
--print(maxPositions)
counterQHer=1

local simBase=sim.getObject('.')

local simTip01=sim.getObject('/Dummy01Tip')
local simTip02=sim.getObject('/Dummy02Tip')
local simTarget01=sim.getObject('/Dummy01Target')
local simTarget02=sim.getObject('/Dummy02Target')

local simTip11=sim.getObject('/Dummy11Tip')
local simTip12=sim.getObject('/Dummy12Tip')
local simTarget11=sim.getObject('/Dummy11Target')
local simTarget12=sim.getObject('/Dummy12Target')

local simTip21=sim.getObject('/Dummy21Tip')
local simTip22=sim.getObject('/Dummy22Tip')
local simTarget21=sim.getObject('/Dummy21Target')
local simTarget22=sim.getObject('/Dummy22Target')



-- create an IK environment:
ikEnv=simIK.createEnvironment()

-- create an IK group:
```

```
    ikGroup_undamped01=simIK.createIkGroup(ikEnv)
    -- set its resolution method to undamped:

simIK.setIkGroupCalculation(ikEnv,ikGroup_undamped01,simIK.method_pseudo_inver
se,0,6)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_undamped01,simBase,simTip01,simTa
rget01,simIK.constraint_position)
    -- create another IK group:
    ikGroup_damped01=simIK.createIkGroup(ikEnv)
    -- set its resolution method to damped:

simIK.setIkGroupCalculation(ikEnv,ikGroup_damped01,simIK.method_damped_least_
squares,1,99)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_damped01,simBase,simTip01,simTarg
et01,simIK.constraint_position)
    ----------------------------------------------------------------------------------------------------
    -- create an IK group:
    ikGroup_undamped02=simIK.createIkGroup(ikEnv)
    -- set its resolution method to undamped:

simIK.setIkGroupCalculation(ikEnv,ikGroup_undamped02,simIK.method_pseudo_inver
se,0,6)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_undamped02,simBase,simTip02,simTa
rget02,simIK.constraint_position)
    -- create another IK group:
    ikGroup_damped02=simIK.createIkGroup(ikEnv)
    -- set its resolution method to damped:

simIK.setIkGroupCalculation(ikEnv,ikGroup_damped02,simIK.method_damped_least_
squares,1,99)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_damped02,simBase,simTip02,simTarg
et02,simIK.constraint_position)
    ----------------------------------------------------------------------------------------------------
    -- create an IK group:
    ikGroup_undamped11=simIK.createIkGroup(ikEnv)
    -- set its resolution method to undamped:

simIK.setIkGroupCalculation(ikEnv,ikGroup_undamped11,simIK.method_pseudo_inver
se,0,6)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_undamped11,simBase,simTip11,simTa
rget11,simIK.constraint_position)
    -- create another IK group:
    ikGroup_damped11=simIK.createIkGroup(ikEnv)
    -- set its resolution method to damped:
```

```
simIK.setIkGroupCalculation(ikEnv,ikGroup_damped11,simIK.method_damped_least_
squares,1,99)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_damped11,simBase,simTip11,simTarg
et11,simIK.constraint_position)
    ----------------------------------------------------------------------------------------------------
    -- create an IK group:
    ikGroup_undamped12=simIK.createIkGroup(ikEnv)
    -- set its resolution method to undamped:

simIK.setIkGroupCalculation(ikEnv,ikGroup_undamped12,simIK.method_pseudo_inver
se,0,6)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_undamped12,simBase,simTip12,simTa
rget12,simIK.constraint_position)
    -- create another IK group:
    ikGroup_damped12=simIK.createIkGroup(ikEnv)
    -- set its resolution method to damped:

simIK.setIkGroupCalculation(ikEnv,ikGroup_damped12,simIK.method_damped_least_
squares,1,99)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_damped12,simBase,simTip12,simTarg
et12,simIK.constraint_position)
    ----------------------------------------------------------------------------------------------------
    -- create an IK group:
    ikGroup_undamped21=simIK.createIkGroup(ikEnv)
    -- set its resolution method to undamped:

simIK.setIkGroupCalculation(ikEnv,ikGroup_undamped21,simIK.method_pseudo_inver
se,0,6)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_undamped21,simBase,simTip21,simTa
rget21,simIK.constraint_position)
    -- create another IK group:
    ikGroup_damped21=simIK.createIkGroup(ikEnv)
    -- set its resolution method to damped:

simIK.setIkGroupCalculation(ikEnv,ikGroup_damped21,simIK.method_damped_least_
squares,1,99)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_damped21,simBase,simTip21,simTarg
et21,simIK.constraint_position)
    ----------------------------------------------------------------------------------------------------
    -- create an IK group:
    ikGroup_undamped22=simIK.createIkGroup(ikEnv)
    -- set its resolution method to undamped:
```

```
simIK.setIkGroupCalculation(ikEnv,ikGroup_undamped22,simIK.method_pseudo_inver
se,0,6)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_undamped22,simBase,simTip22,simTa
rget22,simIK.constraint_position)
    -- create another IK group:
    ikGroup_damped22=simIK.createIkGroup(ikEnv)
    -- set its resolution method to damped:

simIK.setIkGroupCalculation(ikEnv,ikGroup_damped22,simIK.method_damped_least_
squares,1,99)
    -- create an IK element based on the scene content:

simIK.addIkElementFromScene(ikEnv,ikGroup_damped22,simBase,simTip22,simTarg
et22,simIK.constraint_position)

end

function sysCall_actuation()

   P=sim.checkCollision(Herramienta,VT)

    if (P==1) then
        sim.setShapeColor(VT,nullptr,0,g_color1)
    else
        sim.setShapeColor(VT,nullptr,0,g_color2)
    end

    if (counterQHer<maxPositions) then
        for j=1,N,1 do
            sim.setJointPosition(jointHandles[j],QHer[counterQHer][j])
        end

   if
simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_undamped01,true)==simIK.result_fa
il then
        -- the position/orientation could not be reached.
        -- try to solve with the damped method:
        simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_damped01)
        -- We display a IK failure report message:
        sim.addLog(sim.verbosity_scriptwarnings,"IK solver failed.")
        c=1
    end
    ------------------------------------------------------------------------------------
   if
simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_undamped02,true)==simIK.result_fa
il then
        -- the position/orientation could not be reached.
        -- try to solve with the damped method:
        simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_damped02)
        -- We display a IK failure report message:
        sim.addLog(sim.verbosity_scriptwarnings,"IK solver failed.")
        c=2
```

```
    end
    ----------------------------------------------------------------------------------------
   if
simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_undamped11,true)==simIK.result_fa
il then
        -- the position/orientation could not be reached.
        -- try to solve with the damped method:
        simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_damped11)
        -- We display a IK failure report message:
        sim.addLog(sim.verbosity_scriptwarnings,"IK solver failed.")
        c=3
    end
    ----------------------------------------------------------------------------------------
   if
simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_undamped12,true)==simIK.result_fa
il then
        -- the position/orientation could not be reached.
        -- try to solve with the damped method:
        simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_damped12)
        -- We display a IK failure report message:
        sim.addLog(sim.verbosity_scriptwarnings,"IK solver failed.")
        c=4
    end
    ----------------------------------------------------------------------------------------
   if
simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_undamped21,true)==simIK.result_fa
il then
        -- the position/orientation could not be reached.
        -- try to solve with the damped method:
        simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_damped21)
        -- We display a IK failure report message:
        sim.addLog(sim.verbosity_scriptwarnings,"IK solver failed.")
        c=5
    end
    ----------------------------------------------------------------------------------------
   if
simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_undamped22,true)==simIK.result_fa
il then
        -- the position/orientation could not be reached.
        -- try to solve with the damped method:
        simIK.applyIkEnvironmentToScene(ikEnv,ikGroup_damped22)
        -- We display a IK failure report message:
        sim.addLog(sim.verbosity_scriptwarnings,"IK solver failed.")
        c=6
    end

   p18=sim.checkCollision(Objt1,Objt8)
   p48=sim.checkCollision(Objt4,Objt8)
   p39=sim.checkCollision(Objt3,Objt9)
   p59=sim.checkCollision(Objt5,Objt9)
   p67=sim.checkCollision(Objt6,Objt7)
   p27=sim.checkCollision(Objt2,Objt7)
   p810=sim.checkCollision(Objt8,Objt10)
   p910=sim.checkCollision(Objt9,Objt10)
   p710=sim.checkCollision(Objt7,Objt10)
```

```
    if ((c1==1) or (c2==1) or (c3==1) or (c4==1) or (c5==1) or (c6==1) or (p18==1) or (p48==1) or (p39==1) or (p59==1) or (p67==1) or (p27==1) or (p910==1) or (p810==1) or (p710==1)) then
        c1=0
        c2=0
        c3=0
        c4=0
        c5=0
        c6=0
        counterQHer=counterQHer+1
        sim.addLog(sim.verbosity_scriptwarnings,"Collision")
    else
        position=sim.getObjectPosition(Her,-1)
        io.write(table.concat(position,"\t"))
        io.write("\t")
        io.write(table.concat(QHer[counterQHer],"\t"))
        io.write("\n")
        print('Cinematica Correcta')
        counterQHer=counterQHer+1
        end
    else
        sim.stopSimulation()
    end

end

function sysCall_sensing()
    -- put your sensing code here
end

function sysCall_cleanup()
    -- do some clean-up here

    io.close(file)

    --io.close(file)
end

-- See the user manual or the available code snippets for additional callback functions and details
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ARDUINO IDE

Documento número 6: Código fuente

```
#include <Servo.h>

Servo servos[12];

typedef struct
{
  uint8_t pin;
  int offset;
  int min_pos;
  int max_pos;
} RobotServo_t;

#define JOINTS 3

RobotServo_t robotServos[JOINTS] = {{4, 0, 0, 180}, {5, 0, 0, 180}, {6, 0, 0, 180}};


void moveAbsJ(const RobotServo_t servos[JOINTS], const double q0[JOINTS], const
double qT[JOINTS], const double T);

double p0[JOINTS] = {28.45, 0, 0};
double q0[JOINTS] = {0, 0, 0};

void writeServo(const RobotServo_t&servo, int angle)
{
  angle = constrain(angle + servo.offset, servo.min_pos, servo.max_pos);
  servos[servo.pin].attach(servo.pin);
  servos[servo.pin].write(angle);
}

void detachServo(const RobotServo_t &servo)
{
  servos[servo.pin].detach();
}

void moveAbsJ(const RobotServo_t robotServos[], const double q0[], const double qT[],
const double T)
{
  double dt = 0, df = 0;
  double a[JOINTS];
  double b[JOINTS];
  double d[JOINTS];
  double q[JOINTS];
  double t = 0;

  double N = T * 50;

  for (int i = 0; i < JOINTS; i++)
  {
    a[i] = -2 * ((qT[i] - q0[i]) / (T * T * T));
    b[i] = 3 * ((qT[i] - q0[i]) / (T * T));
    d[i] = q0[i];
  }
  for (int j = 0; j < N; j++)
  {
```

```
    df = millis();
    q[0] = a[0] * t * t * t + b[0] * t * t + d[0];
    q[1] = a[1] * t * t * t + b[1] * t * t + d[1];
    q[2] = a[2] * t * t * t + b[2] * t * t + d[2];

    t = t + 0.02;

    writeServo(robotServos[0], q[0]);
    writeServo(robotServos[1], q[1]);
    writeServo(robotServos[2], q[2]);
    dt = millis();
    delay(20 - (dt - df));
  }
}


typedef struct
{
  double l0;
  double l1;
  double l2;
  double l3;
} RobotParams_t;

RobotParams_t params = {6, 13.45, 15, 6};

void inverseKin(double pT[], const RobotParams_t &params, double *q);


void inverseKin(double pT[], const RobotParams_t &params, double *q)
{
  double angulo, x, y, z, q3, cq3;

  for (int i = 0; i < JOINTS; i++)
  {
    angulo=0+2.094395102*i;

    x=pT[0];
    y=pT[1]*cos(angulo)-pT[2]*sin(angulo)+params.l0-params.l3;
    z=pT[1]*sin(angulo)+pT[2]*cos(angulo);

    q3=asin(-z/params.l2);
    cq3=cos(q3);

    q[i]=2*atan((2*params.l1*y + pow((- pow(cq3,4)*pow(params.l2,4) +
2*pow(cq3,2)*pow(params.l1,2)*pow(params.l2,2) +
2*pow(cq3,2)*pow(params.l2,2)*x*x + 2*pow(cq3,2)*pow(params.l2,2)*y*y -
pow(params.l1,4) + 2*pow(params.l1,2)*x*x + 2*pow(params.l1,2)*y*y - pow(x,4) -
2*x*x*y*y - pow(y,4)),0.5))/(- pow(cq3,2)*pow(params.l2,2) + pow(params.l1,2) +
2*params.l1*x + x*x + y*y));

    q[i]=q[i]*57.29577951;

  }
```

```
  Serial.println(q[0]);
  Serial.println(q[1]);
  Serial.println(q[2]);
}



void moveJ(double p0[], double pT[], float T)
{
  double qT[JOINTS], q0[JOINTS];

  inverseKin(p0, params, q0);
  inverseKin(pT, params, qT);
  moveAbsJ(robotServos, q0, qT, T);

}


void moveL(double p0[], double pT[], double T)
{
   double df, dt;
   double a;
   double b;
   double X;
   double s;
   double t;
   double V[JOINTS], P[JOINTS], q[JOINTS];


     X=sqrt(pow(pT[0]-p0[0],2)+pow(pT[1]-p0[1],2)+pow(pT[2]-p0[2],2));
     a=-(2*X)/(T*T*T);
     b=(3*X)/(T*T);
     t=0;

    for(int k=0;k<3;k++)
      V[k]=(pT[k]-p0[k])/X;

     for(int j=0;j<T*50;j++)
     {
      df = millis();
       t=t+0.02;
       s=a*t*t*t+b*t*t;

     for(int k=0;k<3;k++)
      P[k]=s*V[k]+p0[k];

     for(int l=0;l<3;l++)
      inverseKin(P,params,q0);

      writeServo(robotServos[0],q[0]);
      writeServo(robotServos[1],q[1]);
      writeServo(robotServos[2],q[2]);
      dt = millis();
       delay(20 - (dt - df));
     }
```

```
}


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  double p1[]={10,0,0};
  double p2[]={20,0,0};
  double p3[]={20,10,0};
  double p4[]={20,5,-10};

  for (int i = 0; i < JOINTS; i++)
    writeServo(robotServos[i], (int)q0[i]);
  delay(3000);

 moveJ(p0, p1, 2);
 delay(2000);
 moveJ(p1, p2, 2);
 delay(2000);
 moveJ(p2, p3, 2);
 delay(2000);
 moveJ(p3, p4, 2);
 delay(2000);
 moveJ(p4, p0, 2);
 delay(2000);

  for (int i = 0; i < JOINTS; i++)
    detachServo(robotServos[i]);

}

void loop() {
  // put your main code here, to run repeatedly:

}
```