Construcción de un brazo robótico con impresión en 3D

Memoria

Trabajo Final de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autor

Pablo Paniagua Herrero

Tutores

Vidal Moreno Rodilla

Juan Alberto García Esteban

Dr. Vidal Moreno Rodilla, Dr. Juan Alberto García Esteban Profesores del Departamento de Informática y Automática de la Universidad de Salamanca.

CERTIFICAN:

Que el trabajo titulado “Construcción de un brazo robótico con impresión en 3D” ha sido realizado por D. Pablo Paniagua Herrero, con DNI 70960903S y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado en Ingeniería Informática de esta Universidad.

Y para que conste a todos los efectos oportunos.

En Salamanca, 03 de septiembre 2022

MORENO RODILLA VIDAL - 07870365H
Firmado digitalmente por MORENO RODILLA VIDAL - 07870365H
Fecha: 2022.09.06 11:12:16 +02'00'

Dr. Vidal Moreno Rodilla
Dpto. de Informática y Automática
Universidad de Salamanca

GARCIA ESTEBAN JUAN ALBERTO - 70910863T
Firmado digitalmente por GARCIA ESTEBAN JUAN ALBERTO - 70910863T
Fecha: 2022.09.06 10:41:12 +02'00'

Dr. Juan Alberto García Esteban
Dpto. de Informática y Automática
Universidad de Salamanca

Resumen

La impresión 3D se ha abaratado mucho en los últimos 10 años, pasando de ser maquinas industriales a parecer kits DIY (Do It Yourself) relativamente baratos. Actualmente se puede conseguir una impresora 3D, con una calidad aceptable, por 100 €, véase "EASY THREED X1", que es la impresora 3D más barata del mercado.

Esto nos abre la puerta a poder crear prácticamente cualquier cosa a un precio muy asequible, por ejemplo, un brazo robótico.

¿Qué brazo robótico vamos a construir? En la enorme comunidad de impresión 3d thingiverse, donde miles de personas comparten sus diseños de forma gratuita, nos encontramos con Thor. Thor es un brazo robótico Open Source creado por AngelLM completamente imprimible en 3D, capaz de levantar 750 gramos. También nos permite cambiar el acople de la mano entre los que se encuentran diferentes pinzas o si lo quisiéramos podríamos crear nuestros propios acoples como pudiera ser una ventosa o un soldador.

Ilustración 1 Imagen de thor por angelLM fuente: http://thor.angel-lm.com/

Summary

3D printing has become cheaper in the last 10 years, going from industrial machines to relatively cheap DIY (do it yourself) kits. Currently you can get a 3D printer for 100 € a printer with an acceptable quality, see "EASY THREED X1", which is the cheapest 3D printer on the market.

This opens the door to create virtually anything at a very affordable price, for example a robotic arm.

What robotic arm are we going to build? In the huge 3d printing community thingiverse, where thousands of people share their designs for free, we came across Thor. Thor is an open source robotic arm created by AngelLM fully 3D printable, capable of lifting 750 grams. It also allows us to change the coupling of the hand among which are different grippers or if we wanted, we could create our own couplings as could be a suction cup or a soldering iron.

Ilustración 2 image of thor by angelLM source: http://thor.angel-lm.com/

Tabla de contenido

Tabla de Ilustraciones

Palabras clave

Impresión 3D: proceso de fabricación aditiva de objetos tridimensionales a base de capas de material.

Arduino: Plataforma open-source formada por hardware y software. Que permite de manera sencilla iniciar a cualquier persona en robótica e incluso crear proyectos avanzados.

Brazo robótico: Brazo artificial programable, que intenta imitar los brazos humanos a través de distintas articulaciones. Puede tener distintos propósitos, desde coger objetos a soldar u otras actividades.

Key words

3D printing: additive manufacturing process of three-dimensional objects based on layers of material.

Arduino: Open-source platform consisting of hardware and software. It allows in a simple way to initiate anyone in robotics and even create advanced projects.

Robotic arm: Programmable artificial arm, which tries to imitate human arms through different joints. It can have different purposes, from picking up objects to welding or other activities.

1. Introducción

Motivación del proyecto

La robótica es un área que crece cada vez más, en dirección a un mundo cada vez más automatizado donde cada vez menos trabajos requieran esfuerzos físicos. Esto conlleva una gran necesidad de personas formadas en robótica, para diseñar, construir o reparar estos robots. Este proyecto se plantea de forma que ayude a formar a otras personas en estos ámbitos. Obviamente un robot construido con impresora 3D en PLA no es la mejor opción para una gran fabrica, donde se necesita una gran fiabilidad, pero es de gran utilidad a la hora de formar a las personas debido a su bajo coste.

Este proyecto puede ayudar a las personas a formarse en robótica ya que se utilizan técnicas similares a las de robot industriales, tanto en los mecanismos, programación o electrónica. También proyectos como este pueden provocar en las personas nuevas ideas revolucionarias.

Los costes de la impresión 3D han bajado mucho desde sus orígenes donde era un hobby caro que ha pasado a ser para algunos un hobby relativamente barato. Pero no solo eso, muchas empresas han visto mucho potencial en esta técnica de fabricación. Principalmente para crear prototipos a bajo coste. Incluso poder fabricar piezas de repuesto de alguna herramienta cuyo soporte haya terminado y no existan recambios, para poder ahorrar mucho dinero sustituyendo toda la máquina o herramienta que solamente fallaba por una pequeña pieza.

Estructura de la memoria

La memoria se ha divido en este documento, que es el documento principal y otros 5 anexos.

En este documento se detallan aspectos más generales del proyecto y el los anexos aspectos más relacionados con la ingeniería del software y el manual de usuario.

En el anexo 1 se desarrolla el plan de proyecto.

En el anexo 2 se detalla la especificación de requisitos.

En el anexo 3 se muestra la especificación del diseño.

En el anexo 4 encontramos la documentación técnica de programación

En el anexo 5 tenemos un manual de usuario.

En el anexo 6 se ha detallado el montaje del robot Thor

Objetivos

- **Impresión de todas las piezas del brazo robótico:** primeros pasos para la creación de nuestro brazo robótico, que servirá como base de todo el proyecto
- **Elección de la placa controladora** que permitirá el control de motores paso a paso nema usados en este proyecto.
- **Elección de otras partes electrónicas**: como motores, servos, sensores y cables que nos permitirá el correcto funcionamiento de todo el brazo
- **Elección de fuente de la alimentación**: para el robot que nos permitirá suministrar corriente eléctrica tanto a los motores como al resto de componentes electrónicos
- **Montaje del robot** y cableado de todos los componentes electrónicos
- **Programación de la placa controladora:** de forma que sea capaz de recibir comandos, interpretarlos y actuar acode al comando.
- **Programación de un cliente en .net y una interfaz gráfica en wpf:** que permita el control del robot de una manera sencilla y fiable.
- **Mantener un estándar** con otros brazos robóticos del departamento usando llamadas a las funciones RW() e initR() en el cliente.

2. Conceptos teóricos

Impresión 3D

¿Qué es la impresión 3D?

La **impresión 3D** es un avance importante en cuanto a las tecnologías de fabricación por adición consiste en la creación de un objeto tridimensional mediante capas sucesivas de material. (Wikipedia)

¿Solo existe un tipo de impresión 3D?

No, existen muchos tipos se pueden imprimir en 3D muchos materiales, suelen ser materiales que se puede fundir fácilmente sin perder propiedades o líquidos que se puedan solidificar, como plásticos en el primer caso o resinas fotosensibles en el segundo.

¿Cuáles son los tipos de impresión 3D más comunes?

Los tipos de impresión 3D más comunes son FDM (**Fused Deposition Modeling**) y SLA (Stereo Litography Apparatus) o **Estereolitografía**.

La tecnología FDM utiliza termoplásticos para fabricar las piezas. Estos plásticos vienen en forma de filamento enrollado en una bobina, generalmente estos filamentos tienen 1.75 mm de diámetro. Son plásticos que se pueden fundir sin que pierdan sus propiedades

Ilustración 2 Ejemplo de una bobina de PLA (amazon basics)

Las impresoras 3D FDM consisten un fusor de plástico capaz de moverse en los tres ejes del espacio que va depositando el material para formar nuestra pieza.

Ilustración 3 Ejemplo de impresora 3D FDM Creality ender 3 V2 (Página oficial Creality)

Existen muchos plásticos que se pueden utilizar en este tipo de impresión, PLA, PETG, ABS, nylon, TPU… El más común es el PLA ya que es barato y fácil de imprimir, pero aguanta peor las temperaturas altas comparado con otros como PETG o ABS.

La tecnología SLA utiliza resinas fotosensibles para fabricar las piezas. Son resinas liquidas que cuando reciben luz ultravioleta se endurecen.

Ilustración 4 Ejemplo de bote de resina fotosensible (página oficial elegoo)

Las impresoras 3D SLA consisten un depósito de resina con una pantalla ultravioleta con una buena resolución que solidifica la resina para formar nuestra pieza.

Ilustración 5 Creality3D HALOT-ONE: CL-60 Resin 3D Printer (página oficial creality)

¿Cómo imprimir?

1. Para imprimir los primero que necesitamos es el modelo 3D de la pieza que vamos imprimir generalmente en formato *.stl.

   ¿Dónde lo puedo obtener? Hay varias formas, la primera es hacer tu propio diseño con cualquier programa de diseño, como freecad, blender o fusión 360. Otra opción es descargar modelos de internet, existen gran cantidad de diseños publicados en páginas como thingiverse.com, cults3d.com o www.myminifactory.com, donde se puede encontrar prácticamente cualquier diseño que puedas querer.

2. Necesitaremos un slicer, en mi caso utilizaré Ultimaker Cura. Un slicer es un programa que nos permite dividir los diseños 3D en capas. Aquí deberemos indicar que ajustes queremos (ver siguiente apartado). Al hacer el slice obtendremos un fichero gcode, que es un fichero con instrucciones para la impresora 3D para formar la pieza.

3. Pasaremos el fichero gcode a una tarjeta microSD y la introduciremos en la ranura de nuestra impresora 3D.

4. Pondremos a precalentar nuestra impresora a la temperatura necesaria para imprimir con nuestro material. Y mientras se precalienta cargamos la impresora con el filamento.

5. Desde la interfaz de la impresora seleccionamos el gcode que queremos imprimir e iniciará la impresión.

Parámetros más importantes en un slicer

Altura de capa y ancho de línea

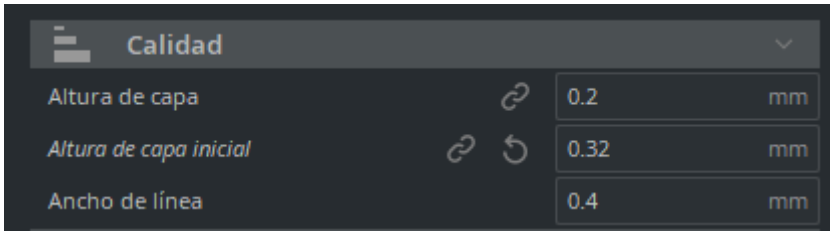


Ilustración 6 Impresión altura de capa

Podemos elegir la altura de capa, cuanto mayor sea menor calidad, pero más rápido imprimirá. En cuanto al ancho de línea debe coincidir con el ancho de la boquilla de nuestra impresora, si cambiamos de boquilla debemos cambiar este parámetro.

Recuento de líneas de pared

Ilustración 7 Impresión Líneas de pared

Nos, permite indicarle cuantas paredes queremos poner en nuestra pieza. Mayor número de paredes aumenta la resistencia de la pieza, pero consume más material y tarda más. Este es uno de los parámetros clave en la impresión del robot, ya que imprimiendo las piezas con un recuento de líneas elevado se ha conseguido que la resistencia de la piezas sea mayor.

Capas superiores e inferiores

Ilustración 8 Impresión capas superior e inferior

Nos permite cambiar el número de capas que queremos en la parte superior e inferior de la pieza. Mayor número de capas aumenta la resistencia de la pieza, pero consume más material y tarda más.

Densidad de relleno y patrón

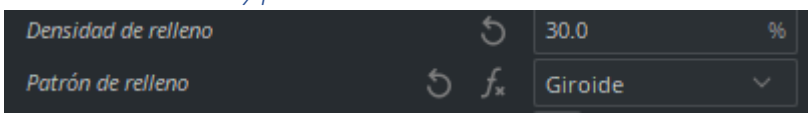


Ilustración 9 Impresión Relleno

Nos permite indicar cuanto relleno queremos en la pieza, mayor porcentaje de relleno aumenta la resistencia de la pieza, pero consume más material y tarda más. En cuanto el patrón tampoco afecta tanto, algunos patrones pueden dar mayor resistencia, pero no tiene tanto efecto como la densidad de relleno.

Temperaturas

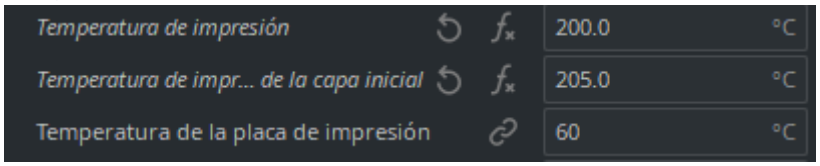


Ilustración 10 Impresión temperatura

Nos permite indicar que temperatura se usará para fundir el filamento, los fabricantes de filamento generalmente nos dan un rango de temperaturas de unos 20 grados, en cualquier temperatura de ese rango debería estar bien. La temperatura de la bandeja o cama de impresión también se puede modificar, generalmente usaremos un 30 o 40 % de la temperatura de impresión, esta nos permite que la pieza se adhiera mejor a la bandeja y no se despegue mientras imprimimos.

Flujo

Ilustración 11 Impresión flujo

Nos permite modificar el flujo de filamento que usará la impresora, generalmente un 100% estará bien, pero si por la mecánica de nuestra impresora añade mayor o menos caudal podemos corregirlo con este parámetro.

Velocidades

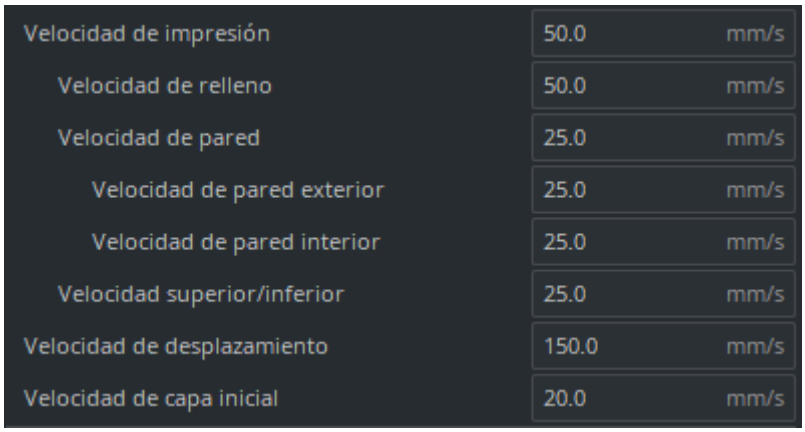


Ilustración 12 Impresión velocidades

Nos permite indicar que velocidades queremos para cada parte. Obviamente a mayor velocidad tarda menos, pero menor calidad. Por lo general la velocidad de relleno y soportes puedes poner mayor velocidad ya que la calidad de estas partes da más igual, pero la de paredes y superior/inferior se le pone menor velocidad.

Soportes

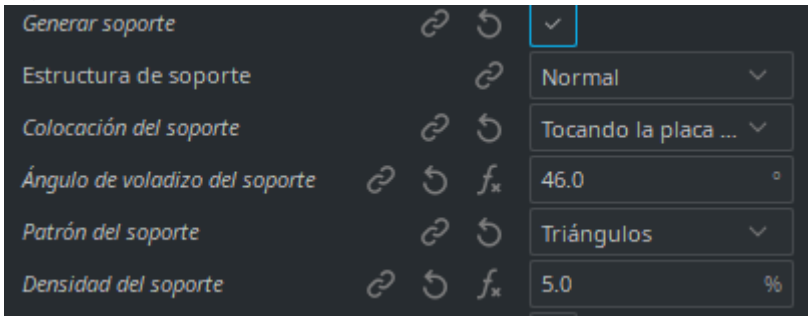


Ilustración 13 Impresión soportes

Los soportes nos permiten imprimir partes de nuestras piezas que de otra forma quedaría en el aire y se caería el material sin formar la pieza.

El parámetro más importante es la densidad, con un 5% es suficiente con un patrón de triángulos.

Brazo robótico

Un **brazo robótico** es un tipo de brazo mecánico, normalmente programable, con funciones parecidas a las de un brazo humano; este puede ser la suma total del mecanismo o puede ser parte de un robot más complejo. Las partes de estos manipuladores o brazos son interconectadas a través de articulaciones que permiten tanto un movimiento rotacional (tales como los de un robot articulado), como un movimiento traslacional o desplazamiento lineal. (Wikipedia)

Los brazos robóticos tienen en su extremo una herramienta, generalmente es una pinza como la de la imagen, que nos permite coger objetos:

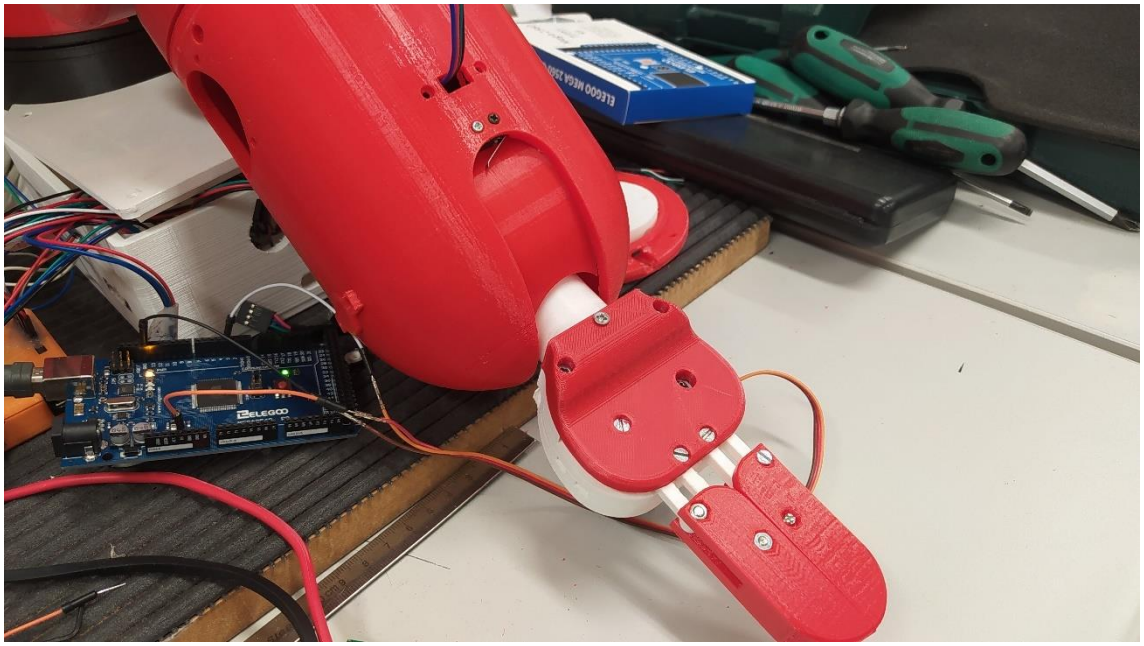


Ilustración 14 Imagen pinza ejemplo

Otra opción de herramientas son ventosas, soldadores, atornilladores o electroimanes. Las ventosas o electroimanes nos permiten coger objetos también. El soldador permite soldar con la precisión que nos proporciona estar montada en un brazo robótico. Con los atornilladores podríamos atornillar de forma que no fuera necesaria la intervención humana en esas partes del montaje de alguna pieza, maquina u objeto.

La capacidad de los brazos robóticos se mide de diferentes formas:

- Capacidad de levantamiento: Se mide en gramos o kilos, es el peso máximo que puede levantar el brazo. En el caso del robot Thor es de 750 gramos.

- Número de grados de libertad: Es el número de giros independientes que puede realizar el brazo robótico. En el caso del robot Thor es de 6 grados de libertad, como se puede apreciar en la siguiente ilustración.

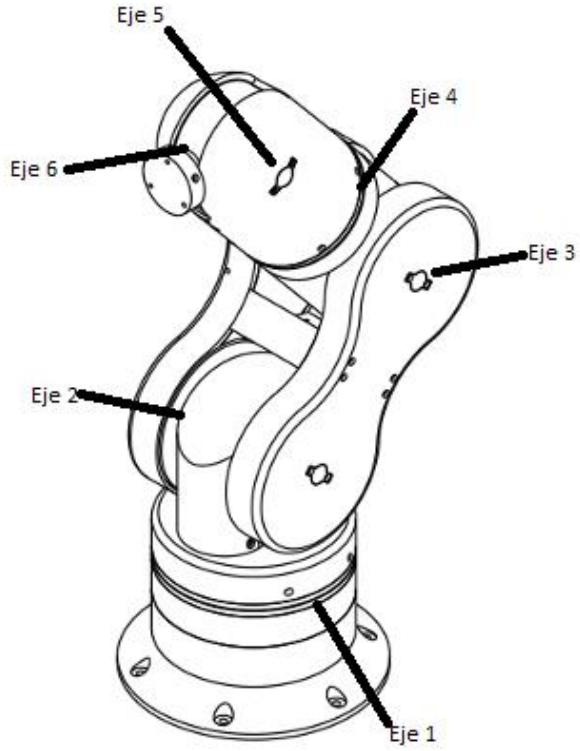


Ilustración 15 Ejes Thor

3. Técnicas y herramientas

Electrónica

Para el proyecto se han usado varios elementos electrónicos que se detallan aquí.

Arduino Mega:

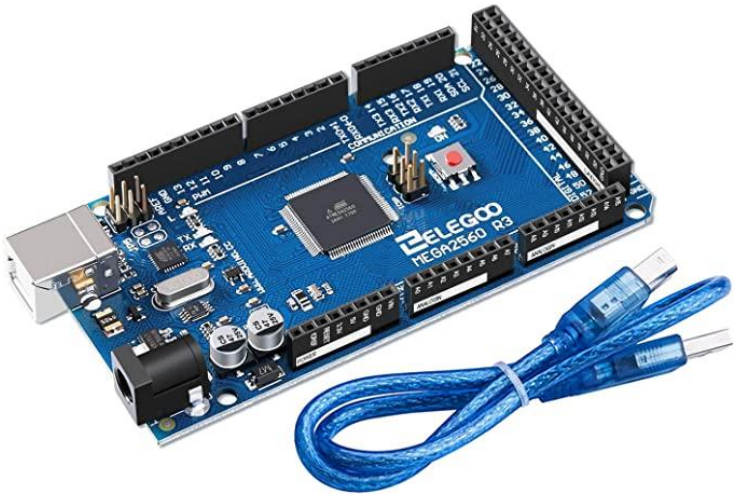


Ilustración 16 Arduino mega general

Arduino es una plataforma electrónica *open-source* basada en hardware y software fácil de usar. Pensado para que cualquiera pueda hacer proyectos interactivos. (arduino.cc).

Existen varias placas Arduino entre las que se encuentran: Arduino UNO, Arduino Nano, Arduino Mega y otras menos comunes. En este caso utilizaremos un Arduino Mega ya que usaremos una *shield* personalizada para él, que nos permitirá controlar los motores paso a paso.

Este Arduino mega será el corazón de nuestro brazo robótico, pues será en el encargado de decidir los movimientos del robot basándose en los comandos recibidos desde el cliente en el ordenador. También leerá los finales de carrera para conocer las posiciones de cada una de las articulaciones.

La placa controladora se programa mediante el uso de un lenguaje de programación homologo, Arduino. Este lenguaje es una variación de c++, pero adaptado a una placa controladora. Esto lo veremos en detalle en el anexo 4. Documentación técnica de programación.

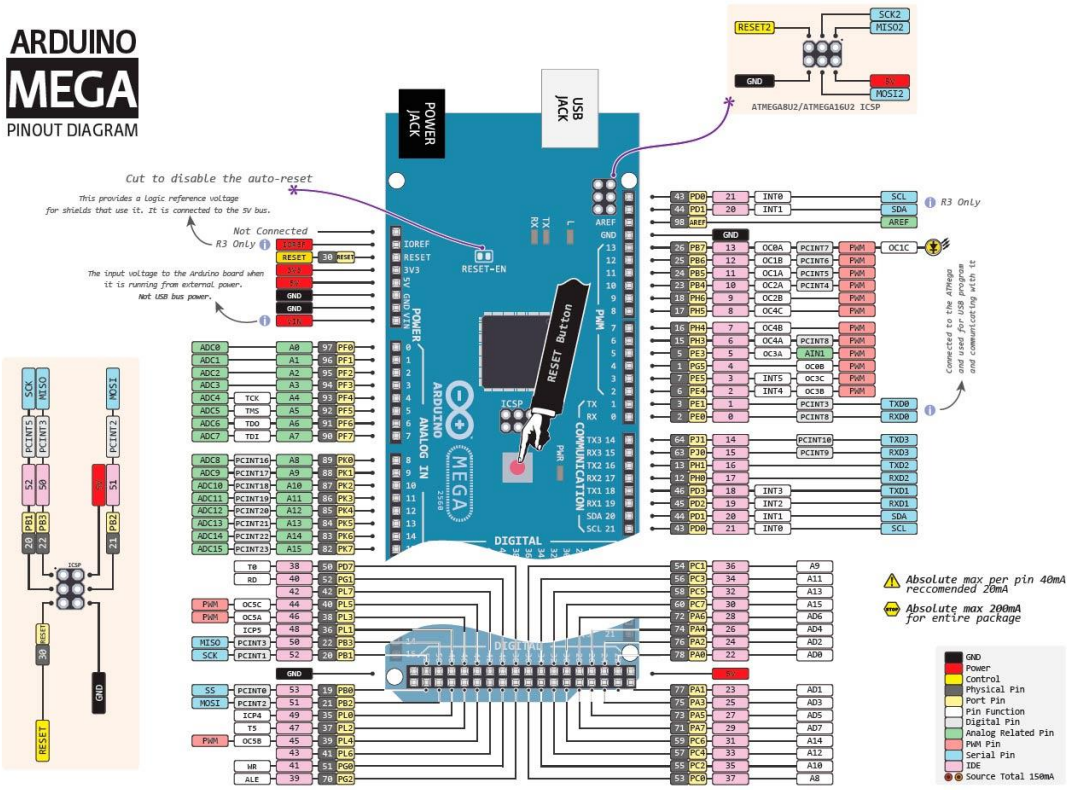


Ilustración 17 PinOut Arduino mega(https://www.hwlibre.com/wp-content/uploads/2020/03/pinout-arduino-mega.jpg)

En esta imagen podemos ver los pines que tiene esta placa controladora. Encontramos distintos pines. Los principales pines son los siguientes:

- Pines de 5V. Proporcionan corriente eléctrica a pequeños elementos electrónicos.
- Pines GND. Proporcionan GND donde conectar elementos electrónicos.
- Pines numerados. Son los pines utilizados para controlar elementos. Se pueden utilizar tanto de entradas como salidas. Existen unos pines especiales llamados PWM. Los pines PWM permiten enviar señales PWM que consisten en pequeños pulsos que permiten enviar a los componentes señales, por ejemplo, la posición a un servo.
- Pines de comunicación. Son pines que se pueden utilizar para comunicarse a través de un puerto serie.
- Pin VIN. Permite alimentar a la placa Arduino con entre 5V y 12V.

También encontramos otros elementos, como una conexión USB para poder conectarse a un ordenador, un puerto de alimentación tipo Jack y un botón de reset.

Microsensor final de carrera físico:

Ilustración 18 Microsensor final de carrera

Es un sensor muy común en este tipo de proyectos ya que nos permite saber cuándo una articulación ha llegado al final de su recorrido. Su funcionamiento es un simple pulsador, pero con la parte activable con forma de lámina, que facilita la colocación para ser activado con piezas mecánicas.

Se puede conectar fácilmente a un Arduino siguiendo el siguiente esquema y, utilizando las resistencias pull-up que tiene integrada la placa controladora, podemos leer si está pulsado o no:

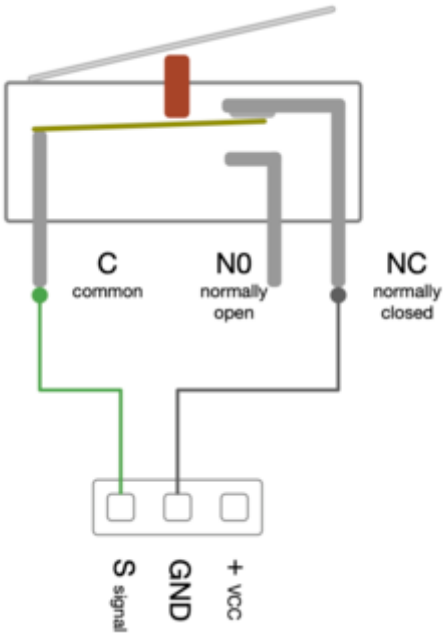


Ilustración 19 micro sensor final de carrera esquema

Fuente: reprap.org

Comprado en: es.rs-online.com

Optointerruptor:

Los optointerruptores son sensores que permiten conocer la posición de piezas. Están formados por dos partes. Un diodo emisor de rayos infrarrojos y otro que lo detecta. Cuando algo se interpone, corta la luz infrarroja y el sensor se activa.

Tiene tres pines. 5v, GND y señal. La señal varía dependiendo si hay algo que corta el flujo de luz o no. Si hay algo envía 0V, sino hay nada envía 5V.

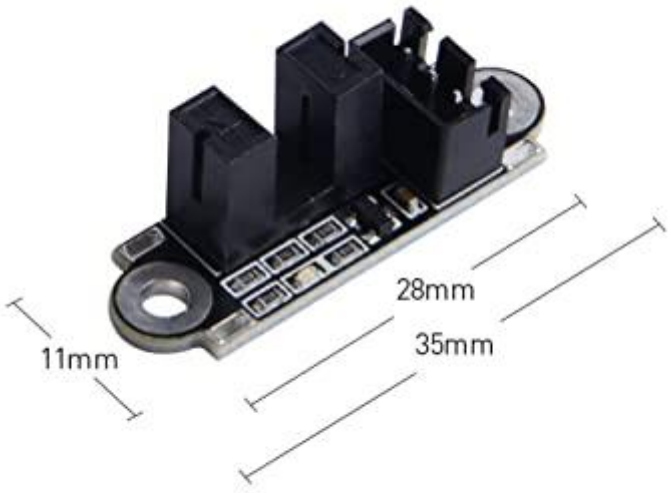


Ilustración 20 Optointerruptor

Se ha usado este tipo de interruptores para marcar la posición inicial del robot en los ejes 1, 2, 3 y 4.

Motor Nema 17:

Los motores Nema 17, son motores de tipo paso a paso muy utilizados en robótica e impresión 3D por su precio y características.

Motor paso a paso significa

Ilustración 21 nema 17 general

Motor paso a paso funciona de la siguiente manera. Tiene conectado al eje un imán fijo. Y alrededor de él tenemos unas bobinas. Dependiendo como polaricemos las bobinas atraen el imán en esa dirección.

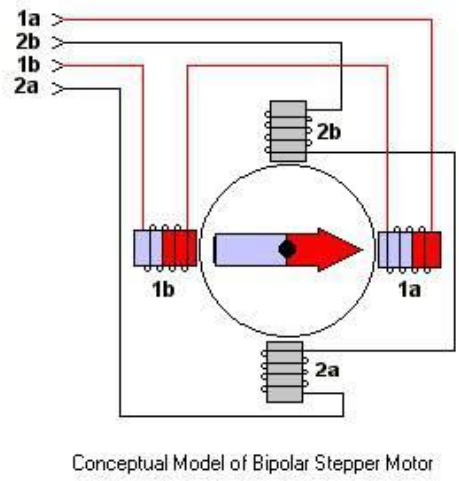


Ilustración 22 Motor paso a paso esquema

Los motores nema 17 tienen 200 pasos, aunque podemos utilizar micropasos, que consiste en dejar el imán central entre medias de dos de las bobinas. Se ha utilizado motores nema 17 para los ejes principales, algunos con distintos pares o reductoras. Ver apartado montaje.

Servo MG995:

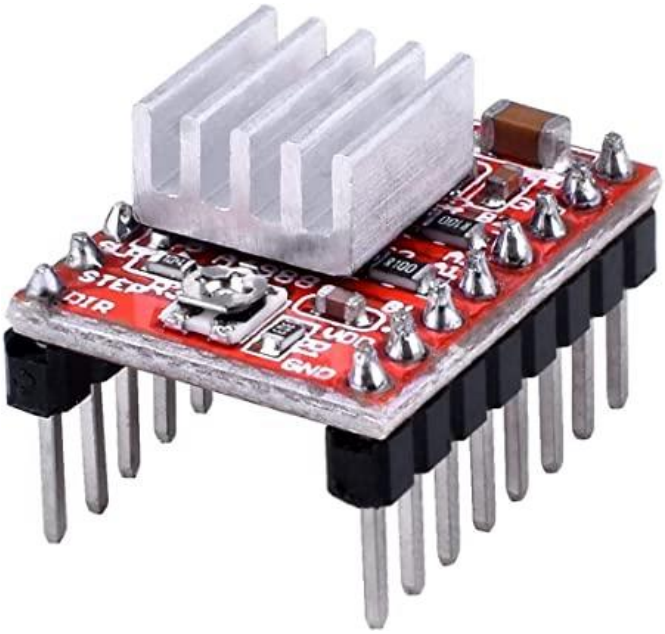
Los servos son un tipo de motor muy precisos y con bastante fuerza. Son capaces de ir a una posición exacta, indicada mediante una señal PWM.

Ilustración 23 Servo

Estos motores tienen 3 conexiones. El cable rojo para alimentación, entre 4.8 y 7 V. El cable negro para GND. Y naranja para la señal PWM, esta señal será la que nos marque la posición a la que debe ir el motor.

Controlador A4988:

Los motores paso a paso no se pueden controlar directamente desde un Arduino, necesitamos



controladores de motores.

Ilustración 24 Controlador A4899

Estos nos permiten controlar y alimentar los motores paso a paso de manera sencilla.

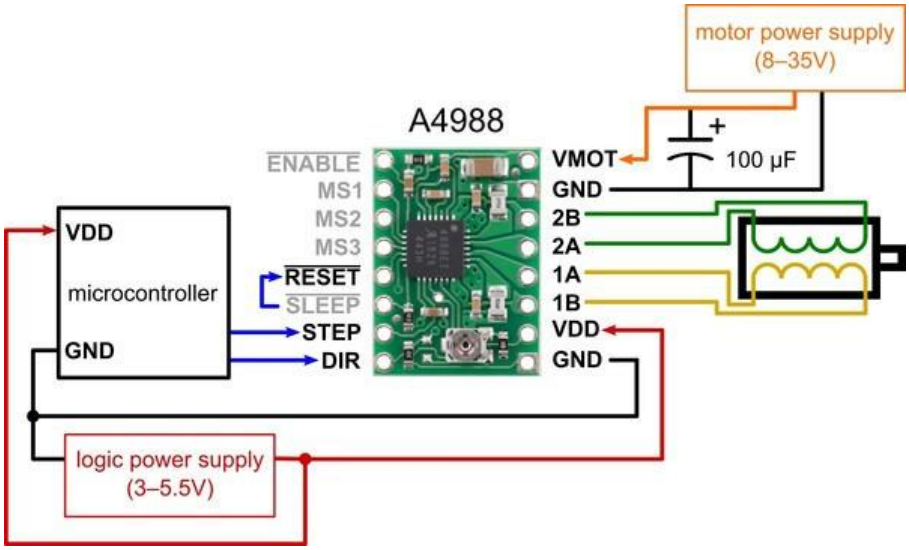


Ilustración 25 Controlador A4899 esquema

Se deben conectar los pines 1A, 1B, 2A, 2B, al motor. Los pines VDD y GND a la alimentación de la electrónica, VMOT a la alimentación del motor.

Para controlarlos se utilizan los pines STEP y DIR, con DIR indicamos la dirección, 5V hacia un lado y 0V hacia el otro lado, dependiendo como esté conectado al motor. Y con el pin step damos un paso cada vez que se pone en 5V. Se debe tener en 0V enable, sino el motor no se moverá, quedará como si no tuviera corriente.

PCB personalizada:

Para poder controlar motores paso a paso no se pueden controlar directamente desde un Arduino, necesitamos controladores de motores. Pero esos controladores tampoco se pueden conectar directamente al Arduino. Para ellos utilizaremos las PCB creado por AngelLM: http://thor.angel-lm.com/documentation/electronics/

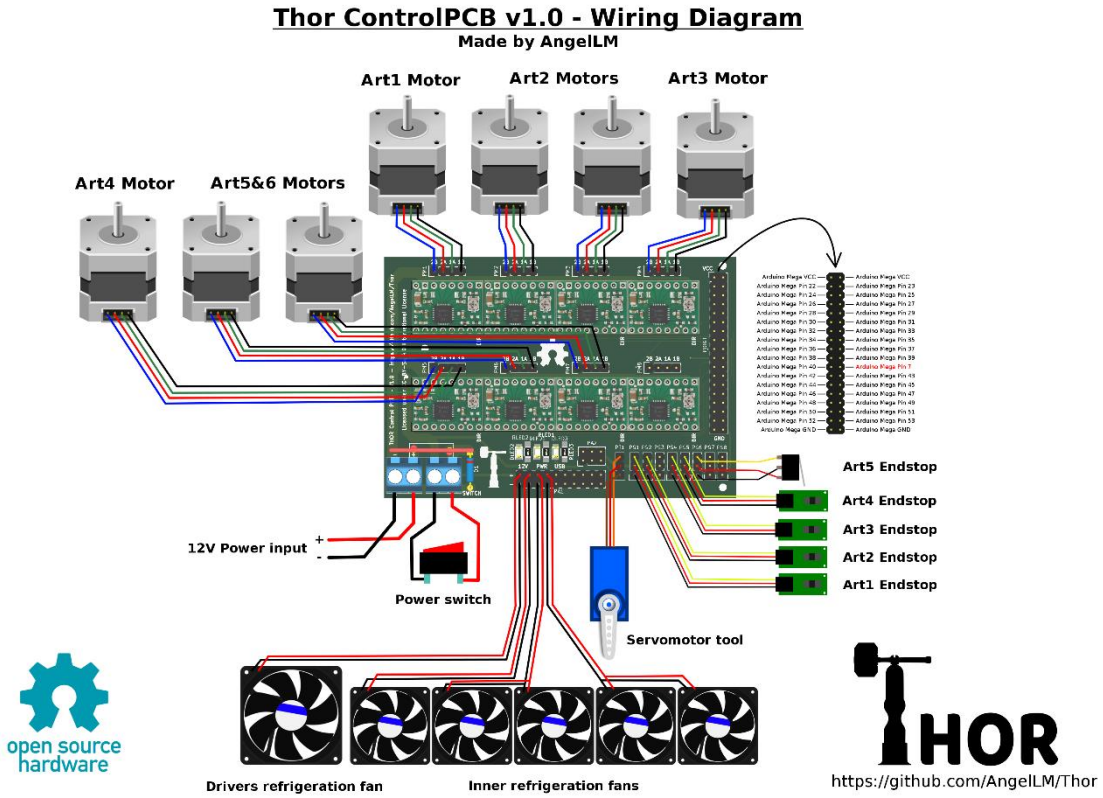


Ilustración 26 PCB esquema

En nuestro caso la encargamos a https://www.pcbway.es/ ellos montaron y soldaron todos los componentes. Esta placa actúa como una shield para Arduino Mega. Incluye soporte para conectar hasta 8 motores con sus 8 controladores correspondientes. También tiene conexión para los distintos finales de carrera y el servo.

Arduino IDE

Arduino IDE es un IDE para desarrollar proyectos de Arduino. Creada por Arduino y disponible de forma gratuita. Se utiliza para programar placas Arduino de forma sencilla. Utiliza un lenguaje de programación también llamado Arduino, que es una variación C++ adaptado para programar una placa Arduino.

Veamos la aplicación en detalle:

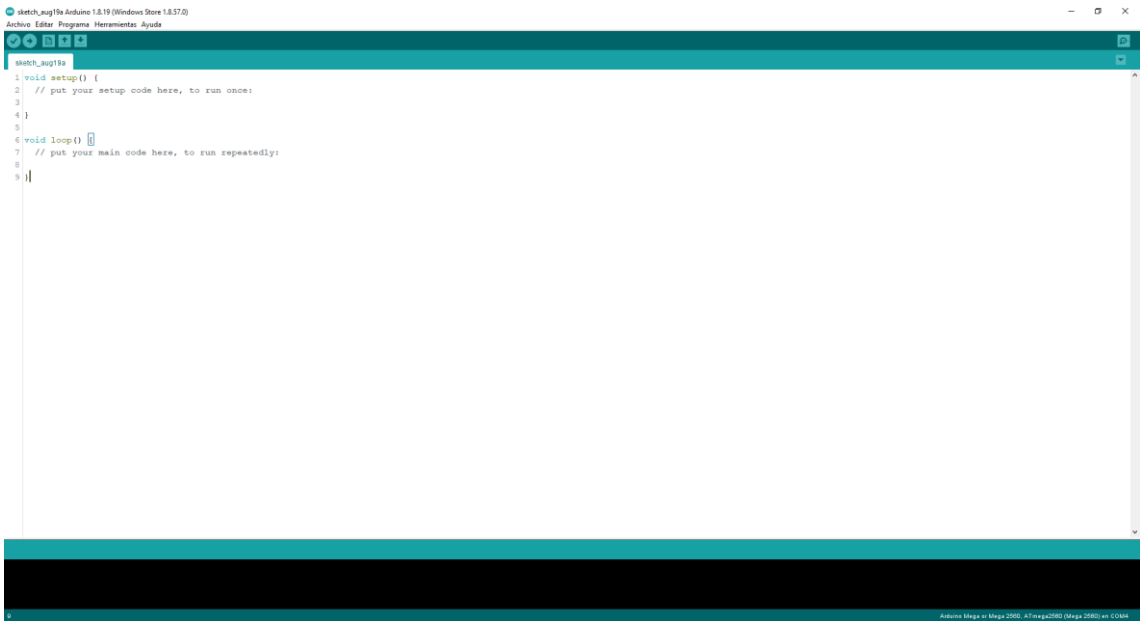


Ilustración 27 Arduino Vista General

Botones

 Botón del Verificar, se utiliza para compilar la aplicación sin subirla. Se utiliza para poder corregir errores de depuración sin ser necesario tener una placa Arduino.

 Botón Subir, se utiliza para compilar el código y subirlo a la placa Arduino. Previamente debemos elegir el modelo de placa.

 Botón Nuevo, abre un nuevo sketch para poder crear un nuevo programa.

 Botón Abrir, abre un sketch ya creado.

 Botón guardar, guarda nuestro sketch.

 Botón monitor Serie, abre la ventana de monitor serie.

Editor de texto

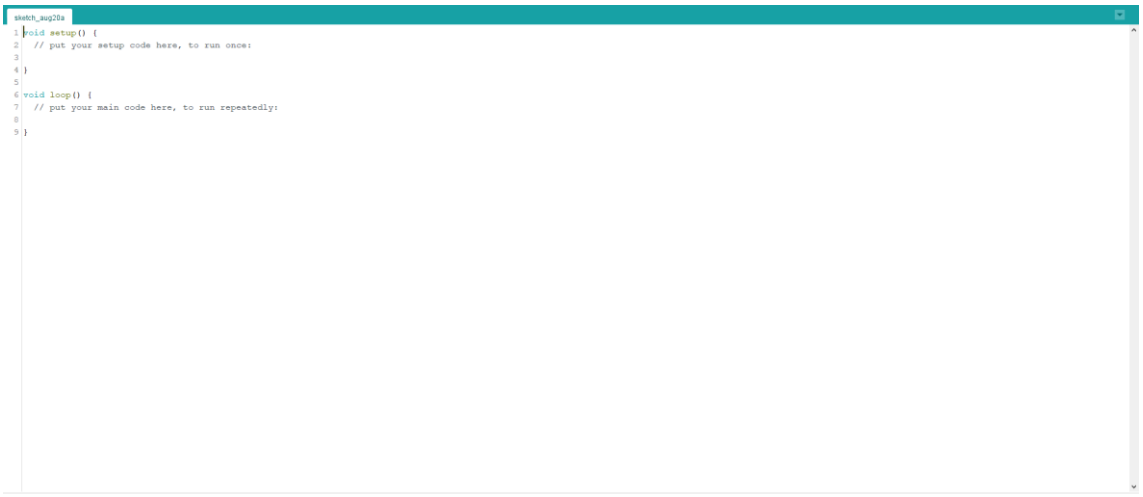


Ilustración 28 Editor de texto Arduino IDE

Aquí podremos modificar el texto de los distintos ficheros de nuestro sketch.

Consola de salida

Ilustración 29 Consola de salida Arduino Ide

Aquí el compilador nos mostrará mensajes sobre la compilación y subida del código a la placa Arduino.

Ventana Monitor serie

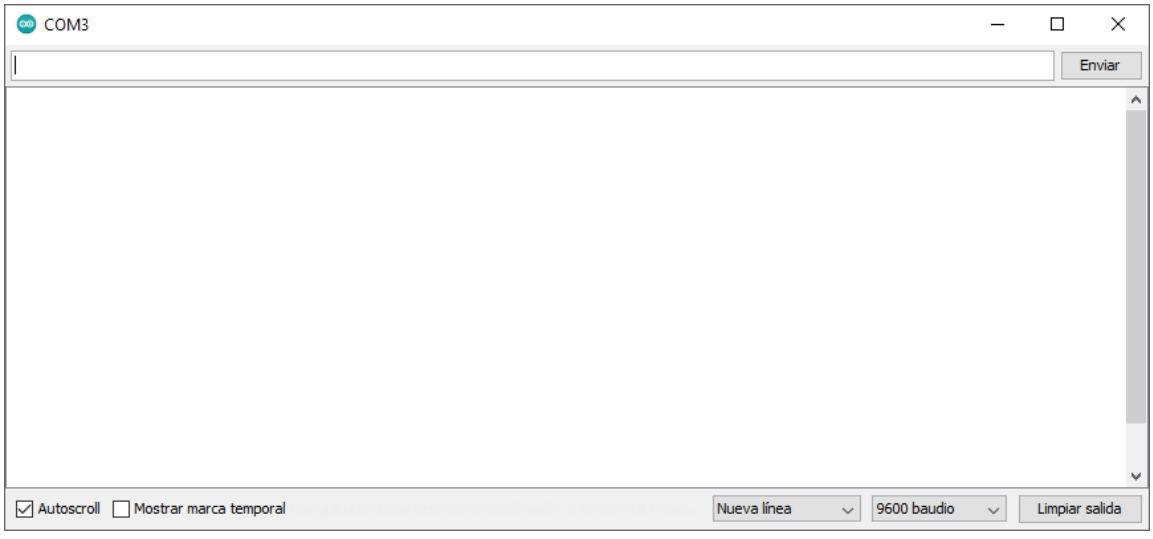


Ilustración 30 Ventana monitor serie Arduino IDE

Esta ventana nos permite comunicarnos por puerto serie con la placa Arduino. Nos muestra en la pantalla principal lo que nos envía el Arduino. En la parte superior tenemos un cuadro de texto donde podremos escribir lo que queramos mandar, la enviaremos con el botón enviar. Nos permite modificar la velocidad de comunicación mediante un combo.

Submenú archivo

Ilustración 31 Arduino submenú editar

Aquí encontramos las funciones típicas del menú archivo como guardar, abrir, cerrar o imprimir.

Submenú editar

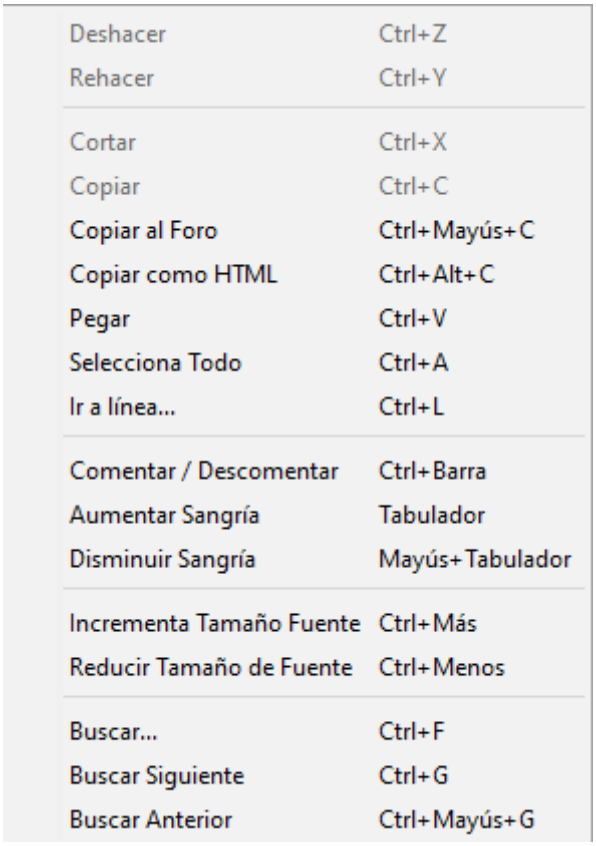
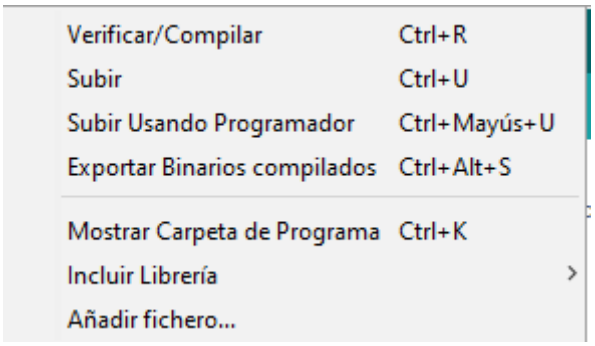


Ilustración 32 Arduino submenú editar

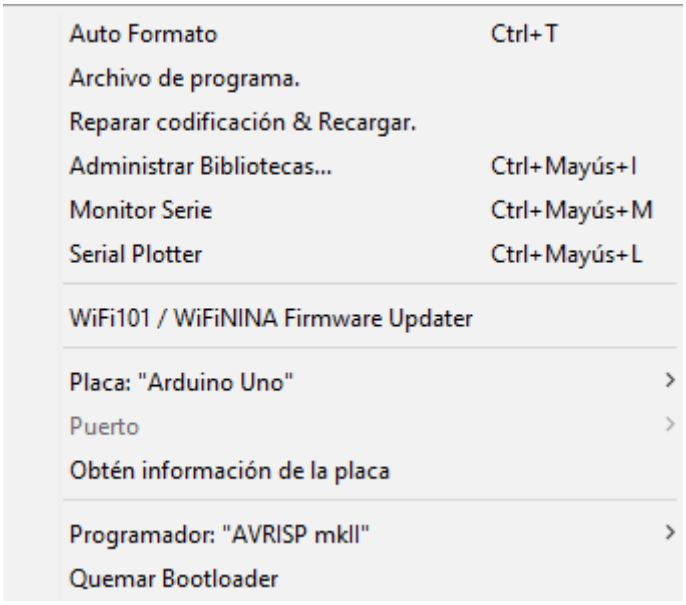
En este submenú encontramos funciones relacionadas con el texto de nuestros archivos, como puede ser copiar y pegar.

Submenú Programa



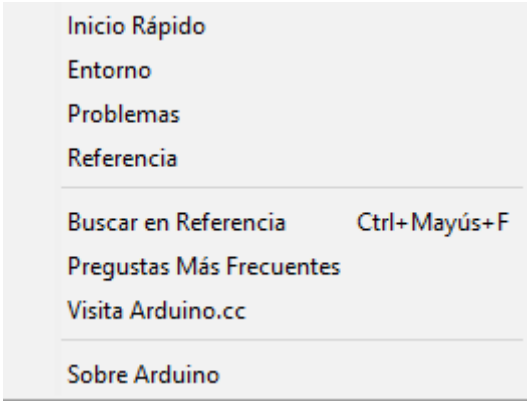
En este submenú encontramos funciones relacionadas con la programación, podremos compilar, subir programas. Desde aquí podremos administrar archivos y librerías.

Submenú Herramientas



Aquí tenemos funciones relacionadas con la placa Arduino. Aquí debemos elegir el tipo de placa y el puerto al que está conectada.

Submenú ayuda



Aquí encontraremos guía de iniciación, enlaces a foros y otras páginas para aprender a usar el entorno.

Visual Studio 2017

Visual Studio (VS) es un IDE para desarrollar .NET y C++ entre otros. Creada por Microsoft y disponible de forma gratuita. En nuestro caso utilizaremos la versión 2017.

Veamos la aplicación en detalle:

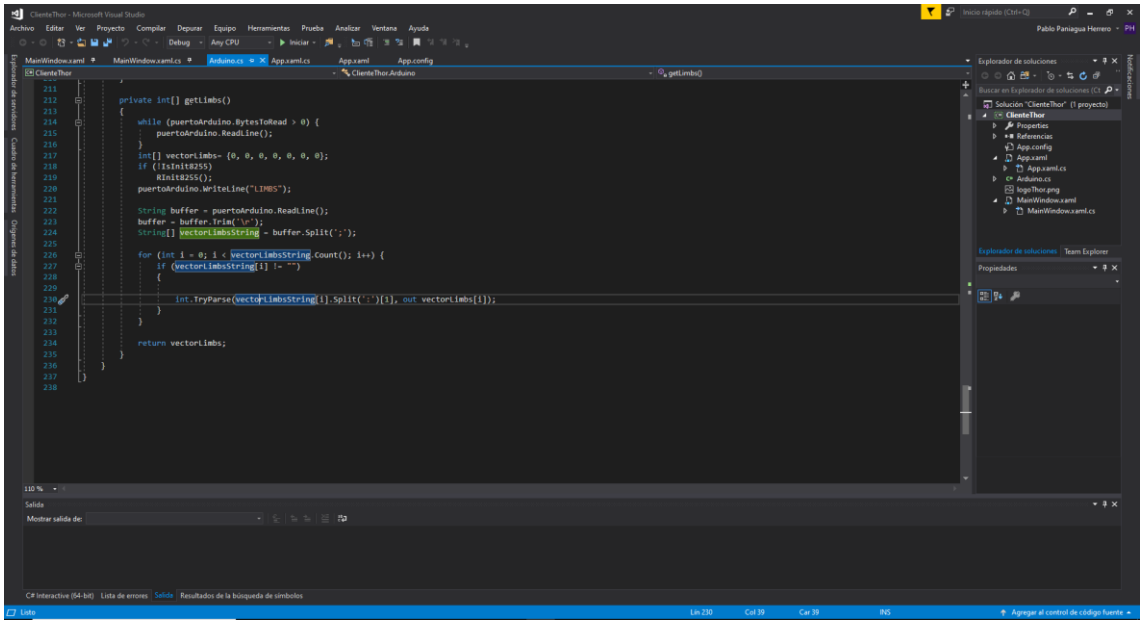


Ilustración 33 Vista general VS

Aquí tenemos una vista general de VS. Veamos en detalle cada uno de sus apartados.

Ventanas

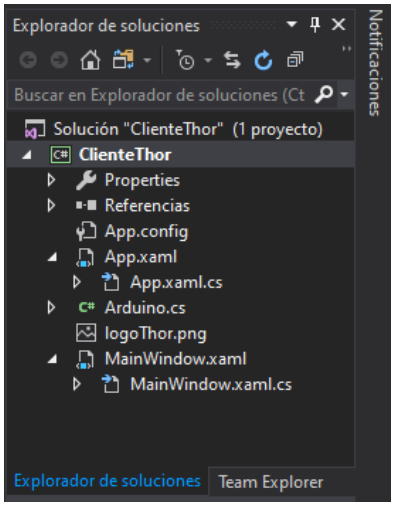


Ilustración 34 VS explorador de soluciones

En este apartado podemos ver todos los archivos que forman parte de nuestra solución, con un clic podremos abrir en el editor de texto el fichero que deseemos abrir.

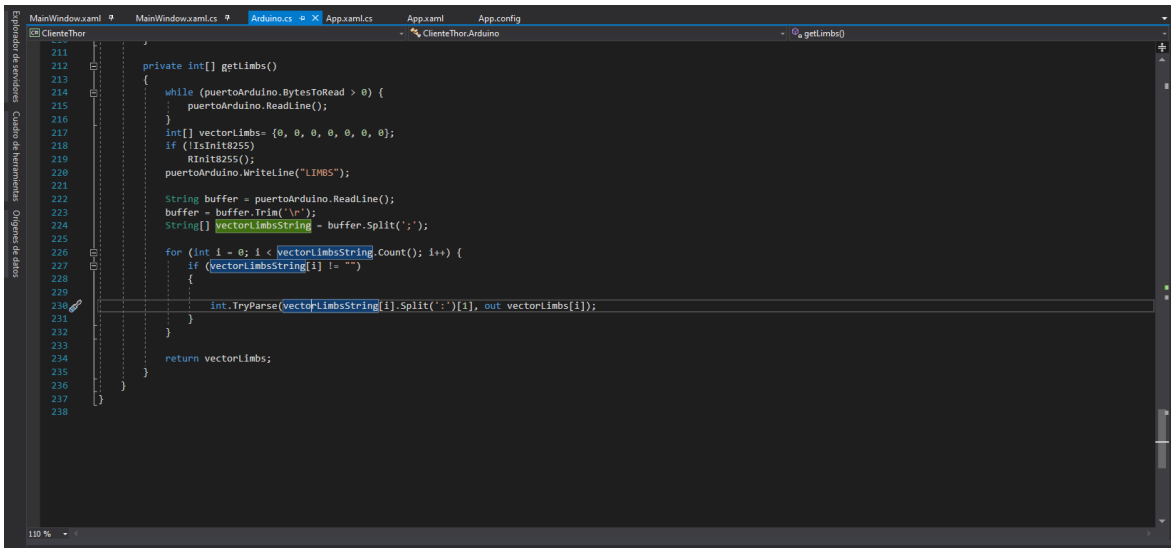


Ilustración 35 Editor de texto VS

En este apartado podemos ver el contenido de cada uno de los ficheros y modificarlos. Tiene una pestaña en la parte superior donde podremos ver que fichero tenemos abiertos y cambiar entre ellos rápidamente.

Ilustración 36 VS Consola de salida

En este apartado, la consola de salida, podemos ver los resultados que nos muestre la aplicación que estemos desarrollando, así como fallos de compilación, warnings, ficheros faltantes u otros. También permite enviar comandos.

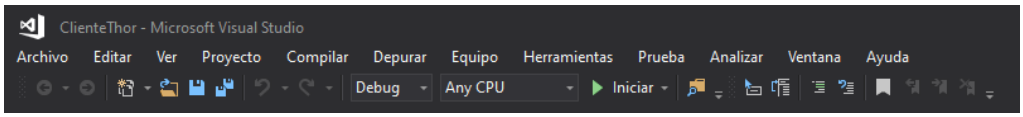


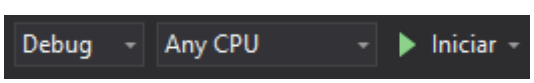
Ilustración 37 Barra de herramientas VS

En este apartado, barra de herramientas, tendremos herramientas útiles para el desarrollo de nuestro proyecto. Veamos las herramientas a las que tenemos acceso directamente y luego veremos en detalle cada uno de los menús.

 Nuevo proyecto, permite añadir un nuevo proyecto.

 Abrir, permite abrir otra solución.

 Guardar, permite guardar el fichero actual o todos los ficheros modificados.



Permite compilar e iniciar la aplicación. Se selecciona si se quiere la versión *debug* o la versión *release* y se pulsa el botón iniciar.

 Botón comentar, comenta todas las líneas seleccionadas.

 Botón descomentar, descomenta todas las líneas seleccionadas.

Submenú archivo

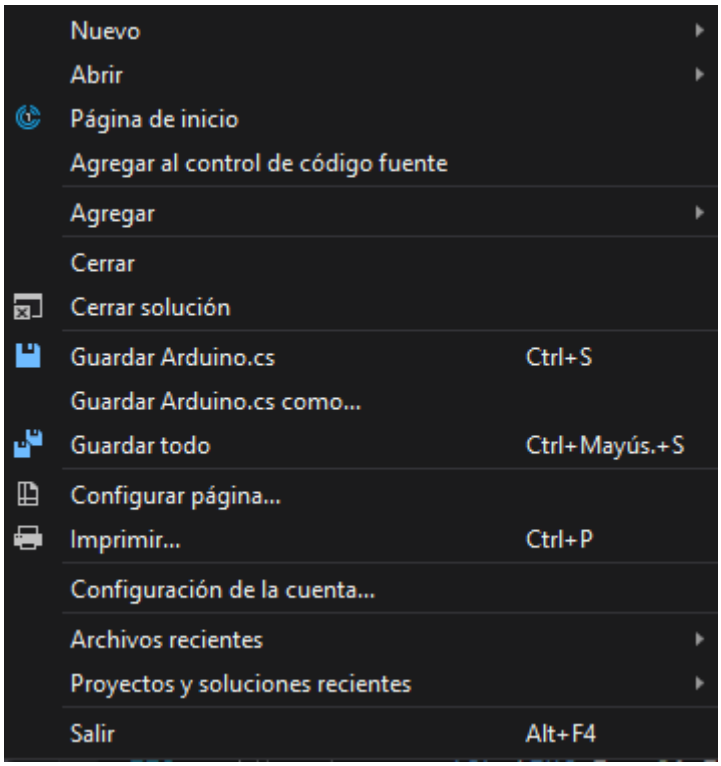


Ilustración 38 VS Submenú archivo

Aquí encontramos las funciones típicas del menú archivo como guardar, abrir, cerrar o imprimir.

Submenú Editar

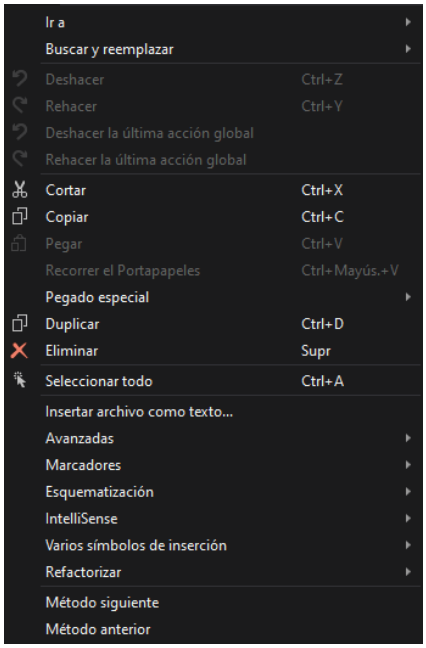


Ilustración 39 VS Submenú Editar

En este submenú encontramos funciones relacionadas con el texto de nuestros archivos, como puede ser copiar y pegar.

Submenú Ver

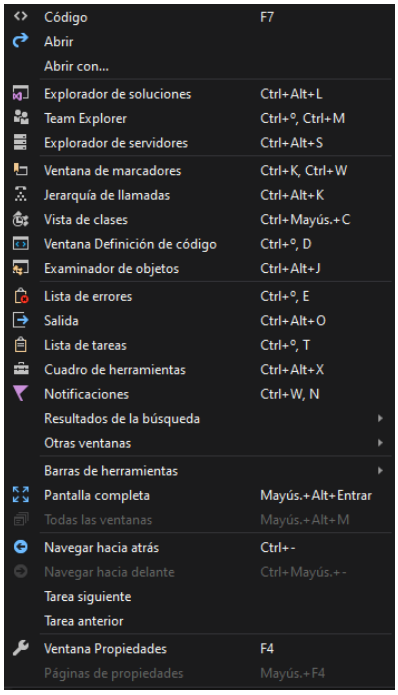


Ilustración 40 VS Submenú ver

Este submenú nos permite modificar que barras de herramientas y ventanas podemos ver.

Submenú proyecto

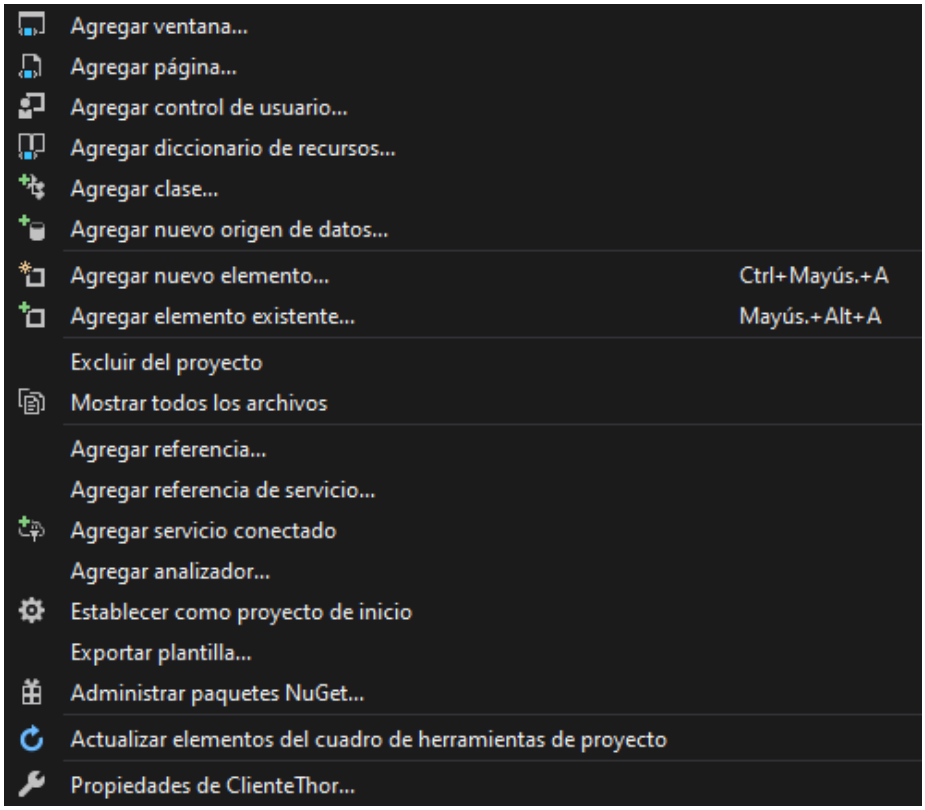


Ilustración 41 VS Submenú Proyecto

Este submenú nos permite modificar aspectos relacionados con el proyecto, como añadir nuevas clases, ventanas o administrar bibliotecas.

Submenú compilar

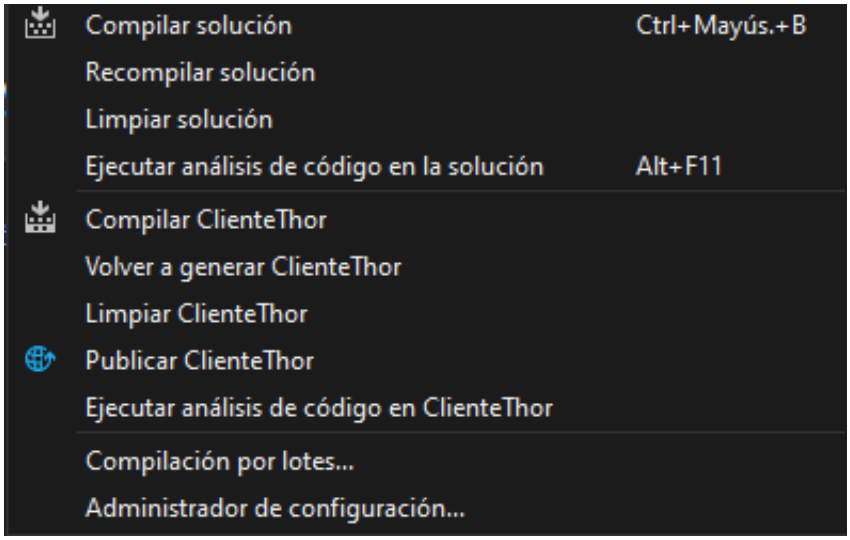


Ilustración 42 VS Submenú compilar

Este submenú nos muestra funciones relacionadas con la compilación del proyecto, así como compilar, limpiar o publicar.

Submenú Depurar

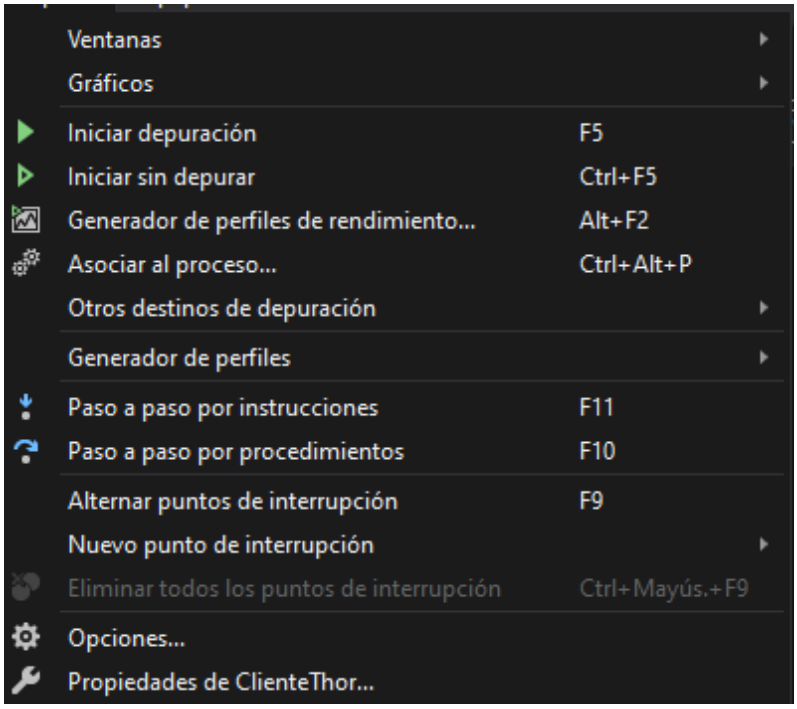


Ilustración 43 VS Submenú Depurar

En este menú encontramos aspectos relacionados con la depuración. Desde este menú podemos iniciar la depuración, alterar puntos de interrupción y manejar las opciones de depuración como paso a paso por instrucciones o paso a paso por procedimientos.

Submenú Equipo

Ilustración 44 VS Submenú Equipo

Aquí solamente encontramos “Administrar conexiones” donde podremos administras nuestras conexiones con bases de datos y con servidores de Azure para poder hacer despliegues de forma sencilla.

Submenú Herramientas

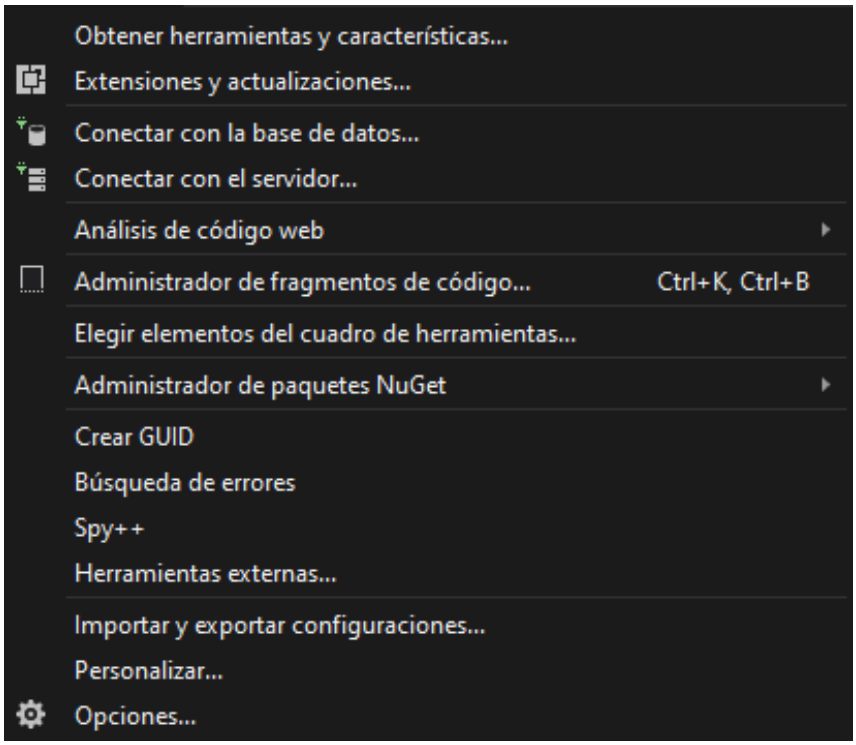


Ilustración 45 VS Submenú Herramientas

Desde este menú podemos acceder a funciones relacionadas con la administración de bibliotecas, extensiones y bases de datos.

Submenú prueba

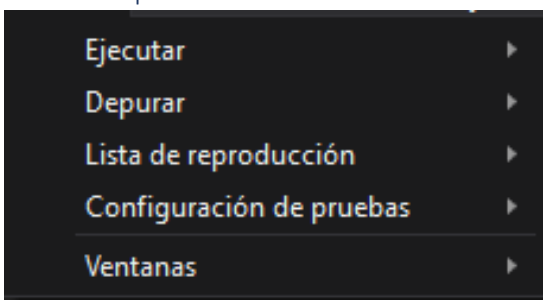


Ilustración 46 VS Submenú Prueba

Desde aquí podemos realizar pruebas de nuestro programa, y administrar pruebas automáticas.

Submenú Analizar

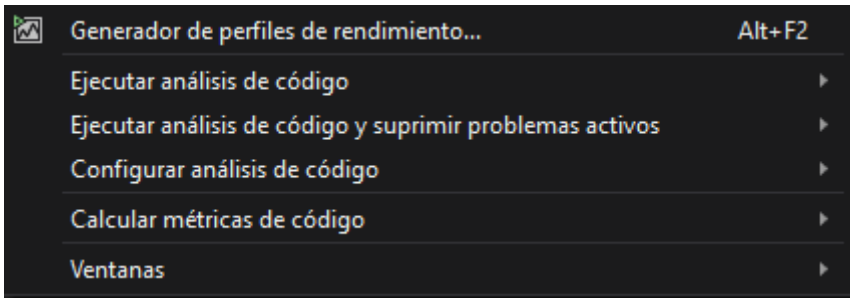


Ilustración 47 VS Submenú Analizar

Desde este submenú podemos utilizar herramientas relacionadas con la búsqueda de errores en nuestro código y calidad de código.

Submenú ventana

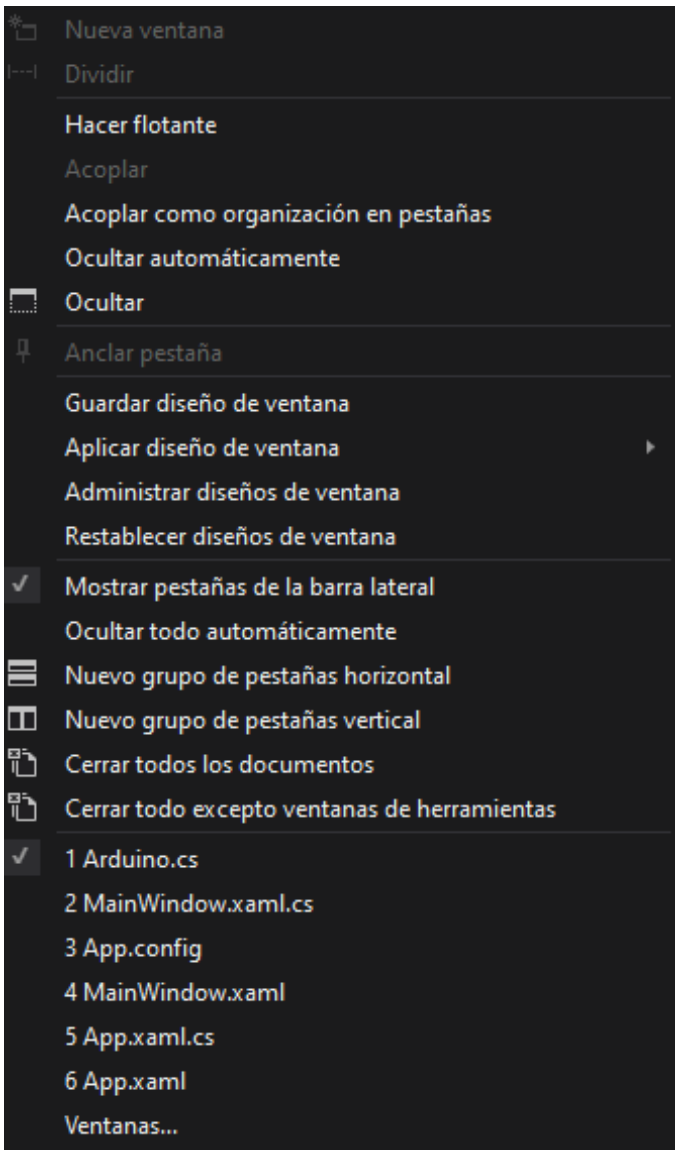


Ilustración 48 VS Submenú Ventana

Desde aquí podemos hacer uso de herramientas relacionadas con la gestión de las distintas ventanas y cambiar de una ventana a otra.

Submenú ayuda

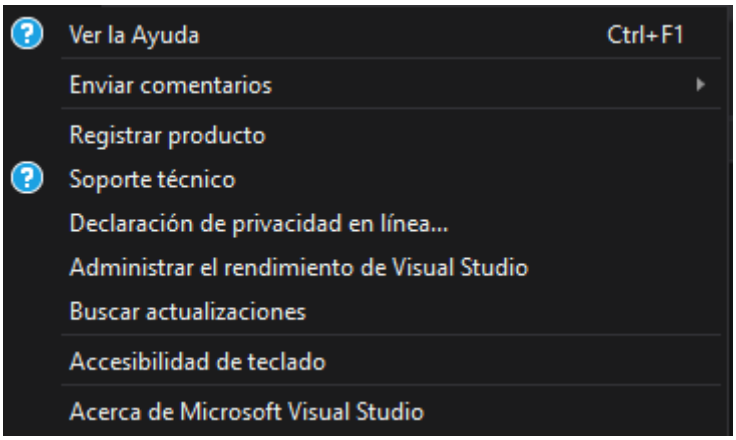


Ilustración 49 VS Submenú ayuda

Desde este menú podemos acceder a toda la ayuda de Microsoft.

4. Aspectos relevantes del desarrollo

En este apartado se recoge los aspectos más relevantes del desarrollo del sistema.

(terminar introducción)

4.1 Modelo de proceso

El Proceso Unificado, es el proceso por excelencia de desarrollo de software, pero debido a las características de este proyecto, que, por cierta sencillez del software implicado, y puesto que gran parte del proyecto se centra el electrónica y piezas físicas no sería una opción demasiado adecuada. Por eso se ha elegido un Proceso Unificado Ágil. Una versión simplificada del Proceso Unificado convencional. Combina conceptos del proceso unificado con técnicas agiles, con el objetivo de mejorar la productividad.

Las actividades de modelado realizadas para este proyecto se encuentran detalladas en los anexos II y III. Se ha prestado mayor atención en los apartados necesarios para la implementación y mantenimiento del sistema.

4.2 Planificación y estimación

En el documento Anexo I se detalla la planificación temporal del proyecto. Para ellos se ha utilizado un diagrama Gantt, creado en la herramienta Microsoft Project.

	Task Mode	Task Name	Duration	Start	Finish
1		Especificación Requisitos	18 days	Fri 11/02/22	Tue 08/03/22
2		Estudio del sistema Robotico	21 days	Sun 20/02/22	Sun 20/03/22
3		Impresion de las piezas	40 days	Mon 07/03/2	Sun 01/05/22
4		Elección Componentes electronica	19 days	Wed 16/03/22	Sat 09/04/22
5		Montaje del robot	36 days	Thu 31/03/22	Thu 19/05/22
6		Cableado del robot	15 days	Wed 27/04/2	Tue 17/05/22
7		programacion del robot	45 days	Tue 03/05/22	Sun 03/07/22
8		programacion del cliente	33,5 days	Sat 21/05/22	Wed 06/07/22
9		Desarrollo del protocolo de comunicación	8,5 days	Wed 08/06/22	Mon 20/06/22
10		creación de la interfaz gráfica del cliente	14,5 days	Sat 11/06/22	Thu 30/06/22
11		Pruebas	31,5 days	Mon 20/06/2	Tue 02/08/22

Ilustración 50 Tabla diagrama Gantt

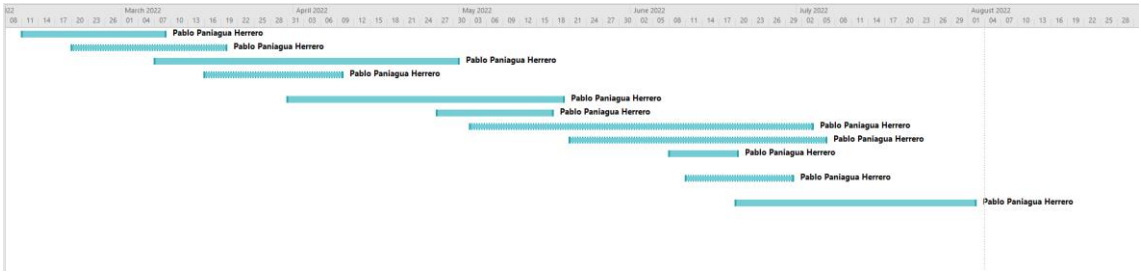


Ilustración 51 Gráfica diagrama Gantt

4.3 Modelado del sistema

El objetivo de este proceso es desarrollar el dominio del problema y de la solución, primero se detalló el dominio del problema, que lo encontramos en el Anexo 2.

El proyecto se ha dividido en dos subsistemas. ClienteThor, que será el encargado de interactuar con el usuario, leer las secuencias de movimientos de fichero y enviarlas al otro subsistema mediante un puerto serie. El otro subsistema, ThorRobot se encarga de recibir las órdenes del clienteThor y ejecutarlas moviendo al robot.

En este anexo se detallan los apartados más importantes del dominio del problema: objetivos, actores, requisitos funcionales, requisitos no funcionales, requisitos de información.

Diagramas de casos de uso:

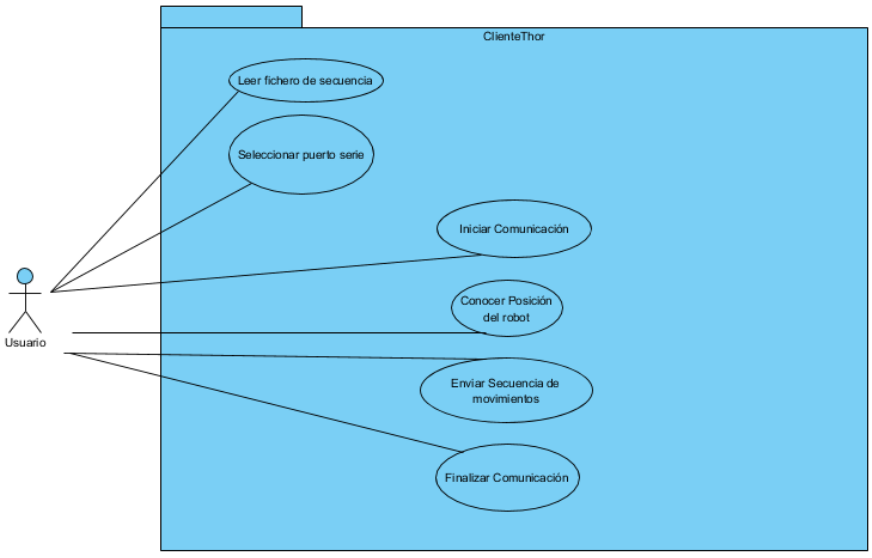


Ilustración 52 diagrama de UC cliente

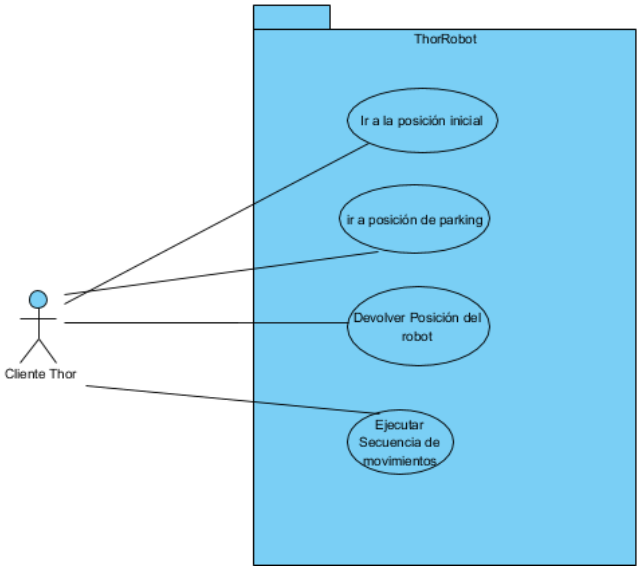


Ilustración 53 Diagrama de UC ThorRobot

En cuanto el dominio de la solución se encuentra detallado en el Anexo 3. En él se detalla cómo se va a resolver el dominio del problema.

Se divide en tres partes una para el cliente, otra para el controlador del robot y otra para la comunicación entre ellos.

En cuanto al cliente, se ha detallado la arquitectura del sistema

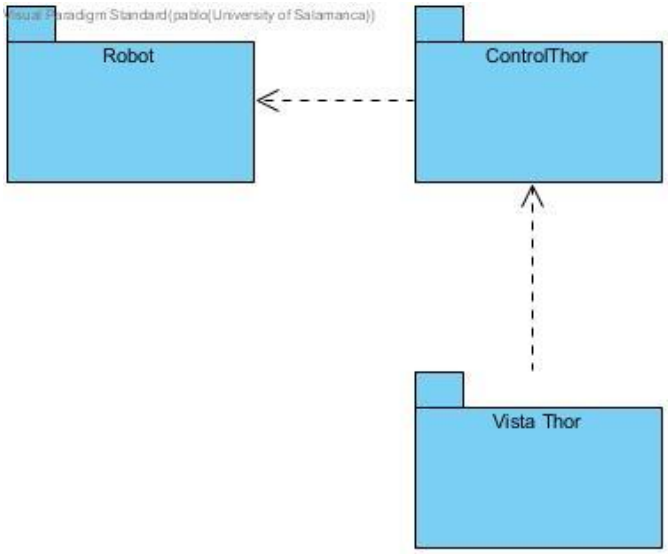


Ilustración 54 Arquitectura Cliente

También se ha detallado como se ha desarrollado la interfaz gráfica y se ha añadido un diagrama de clases.

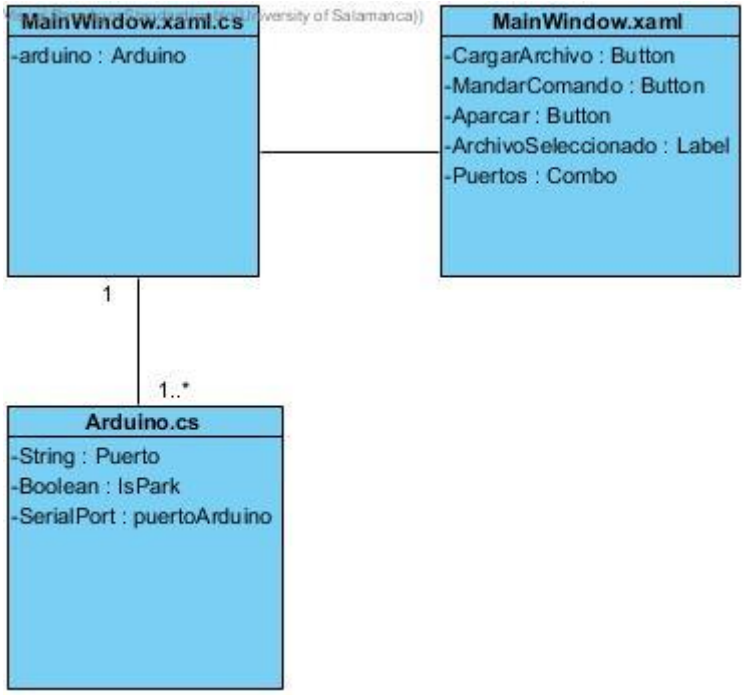


Ilustración 55 Diagrama de clases cliente

En cuanto a la comunicación, se ha creado un diagrama de despliegue donde podemos apreciar cómo se conecta un sistema con otro. También se ha creado un diagrama de secuencia de la comunicación entre sistemas.

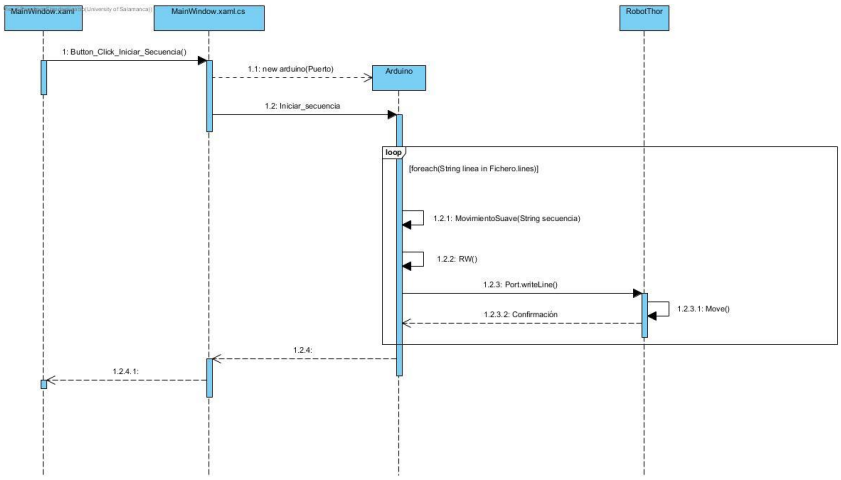


Ilustración 56 diagrama de secuencia de comunicación

4.4 Montaje del brazo robótico

Para montar el brazo necesitaremos las siguientes piezas impresas en 3D, todas están en Thingiverse y en la página web http://thor.angel-lm.com/ pero en Thingiverse solo encontraremos los archivos *.stl' y tendremos que descargar de 1 en 1 por lo que en recomendable descargarlo desde la web de Thor. Más de detalles en el apartado impresión 3D (añadir enlace).

También necesitaremos otros materiales como tornillos, varillas o tuercas y componentes electrónicos, así como motores o finales de carrera, en la propia página web de Thor también encontramos las siguientes tablas que nos los detallan.

Lista de materiales:

Motores paso a paso

Cantidad	Tamaño(mm)	Holding Torque (N x cm)	Otros	Lugar
1	40	39.22		Art 1
3	34	121.2	5:1 reducción mecánica	Art 2 y 3
3	34	21.57		Art 4, 5 y 6

Correas GT2

Cantidad	Tipo	Largo (mm)	Ancho (mm)
2	Correa cerrada GT2	208 (104 dientes)	6
1	Correa abierta GT2	1000 (500 dientes)	6

Poleas GT2

Cantidad	Tipo	Numero de dientes	Centro (mm)
3	Polea GT2	20	5
2	Polea GT2	40	5

Rodamientos

Cantidad	Tipo
1	16014zz
11	625ZZ
2	MF84ZZ
30-40	Bolas de plástico o metal de 6 mm

Varillas metálicas

Cantidad	Largo (mm)	Diámetro (mm)
2	14	4
1	14.5	5
1	32	5
1	102	5
1	128	5

Tornillos

Cantidad	Tamaño	Largo (mm)
2	M2	4

3	M2	6
2	M2	10
4	M3	6
60	M3	8
4	M3	10
35	M3	12
2	M3	14
37	M3	16
12	M3	20
16	M3	25
8	M3	30
15	M3	40
8	M3	46
4	M4	20
1	M5	16

Tuerca

Cantidad	Tamaño
4	M2
155	M3
4	M4
1	M5

Electrónica

Cantidad	Parte	Descripción
1	Arduino Mega	4
1	Micro Endstop	6
7	Ventilador	10
4	Final de carrera óptico	6
1	Botón on/off	8
1	Botón reset	10

Componentes de la PCB

Cantidad	Parte
1	PCB

Más detalles de la PCB en el apartado PCB (añadir enlace)

¿Qué tipo de impresión hemos elegido para este proyecto?

Para este proyecto hemos elegido la impresión FDM, ya que nos permite una mayor resistencia mecánica.

Dentro de FDM, hemos elegido el material PLA, porque nos proporciona una buena resistencia mecánica, es barato y fácil de imprimir, como aspecto negativo del PLA tenemos que poca resistencia a las altas temperaturas.

Como impresora utilizamos una BQ witbox

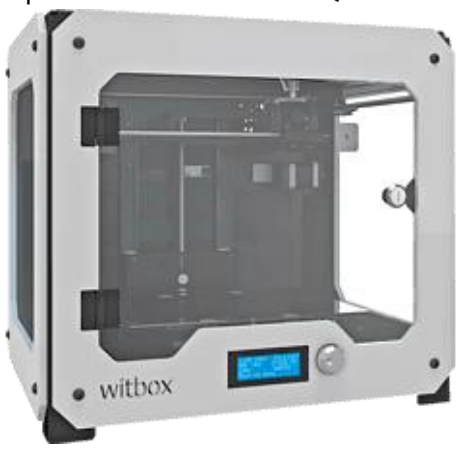
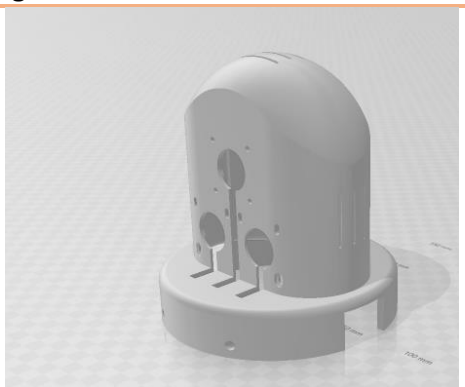
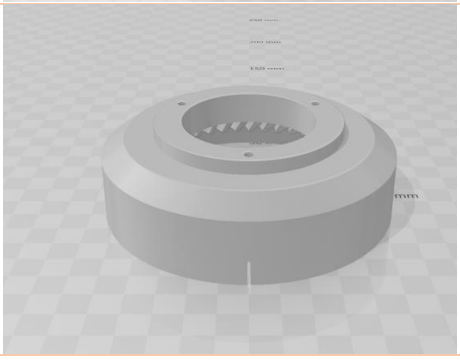


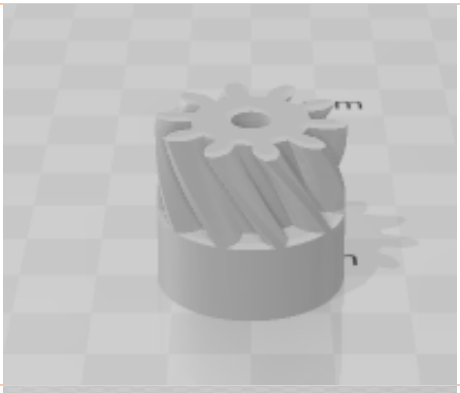
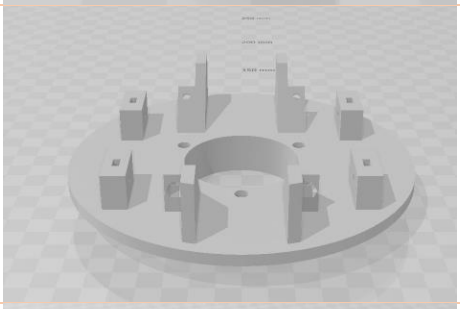
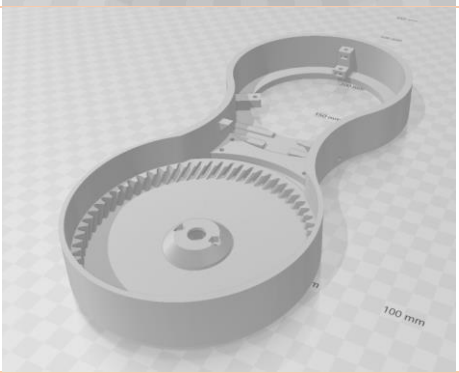
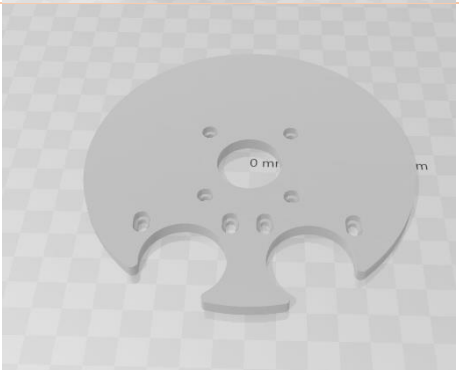
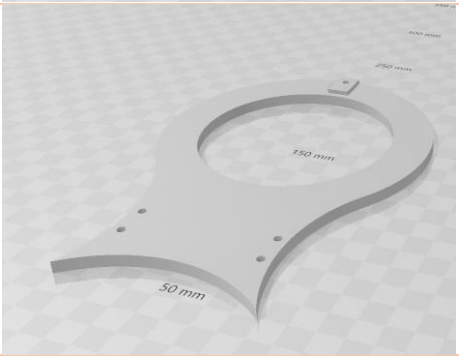
Ilustración 57 BQ witbox

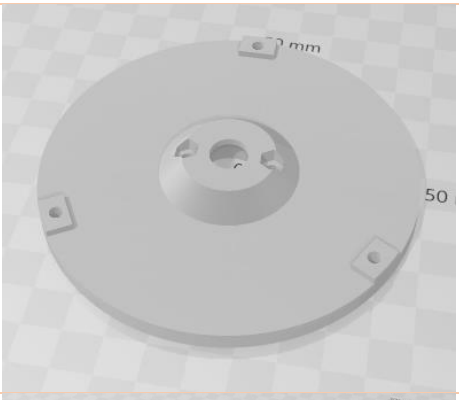
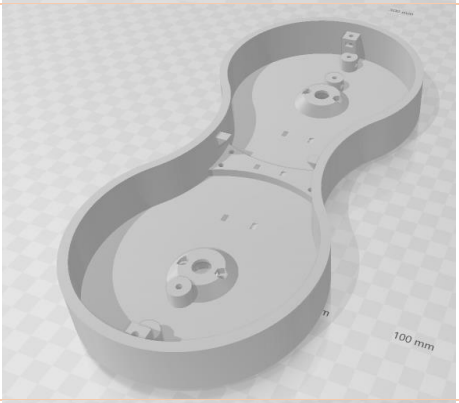
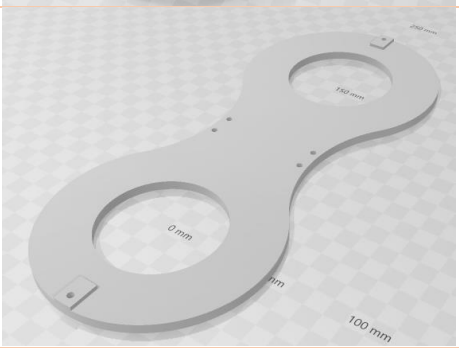
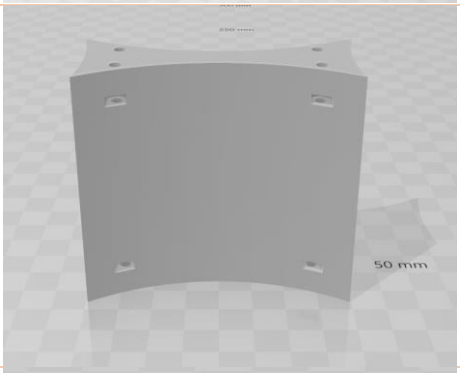
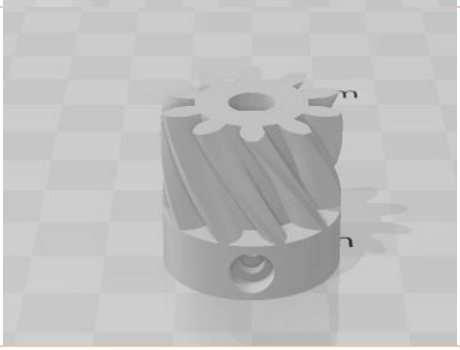
Esta impresora nos permite un volumen de impresión de 297 x 310 x 200 mm, lo que nos proporciona un gran volumen, ya que algunas piezas de nuestro robot en otras impresoras más pequeñas no son suficientemente grandes y sería necesario dividir las piezas para poder imprimirlas.

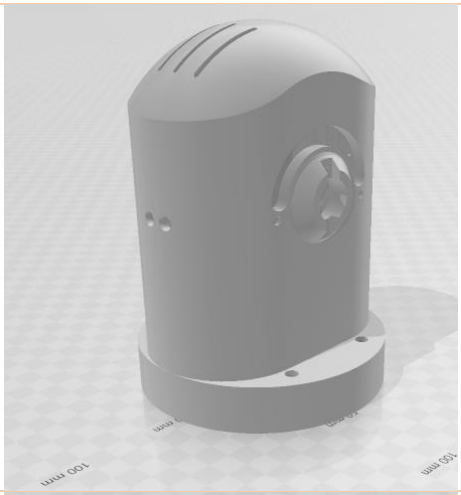
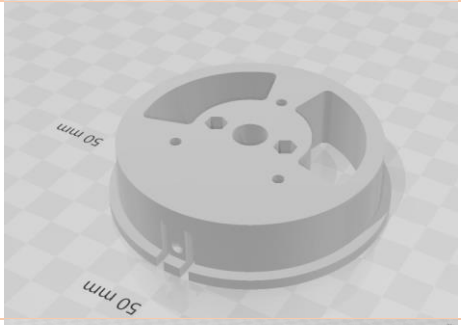
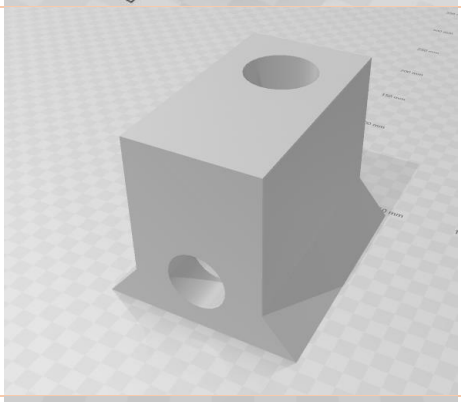
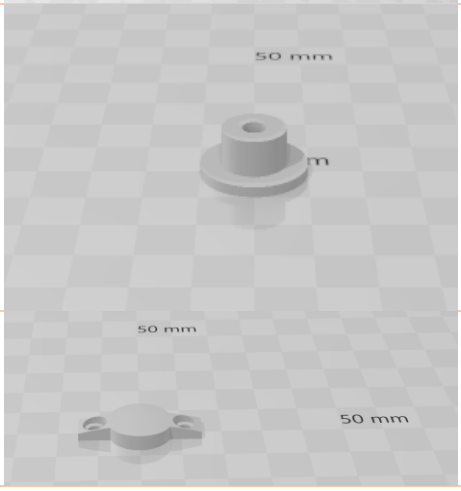
¿Qué hay que imprimir para Thor?

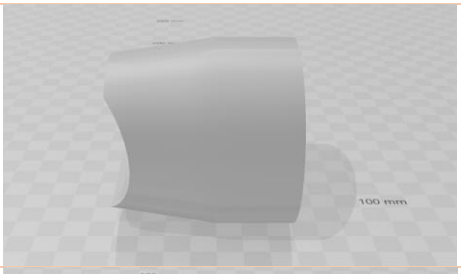
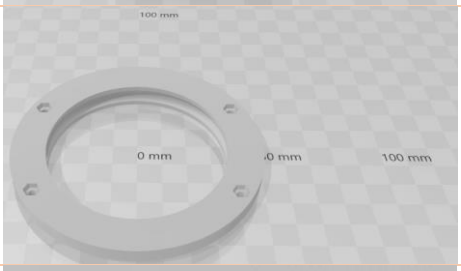
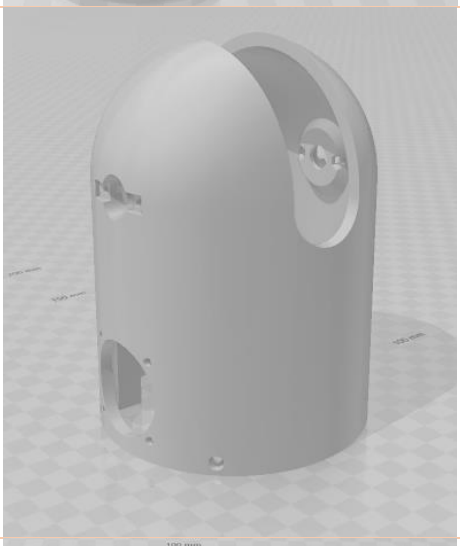
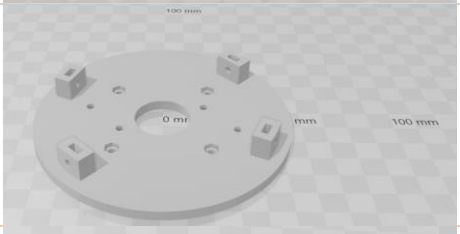
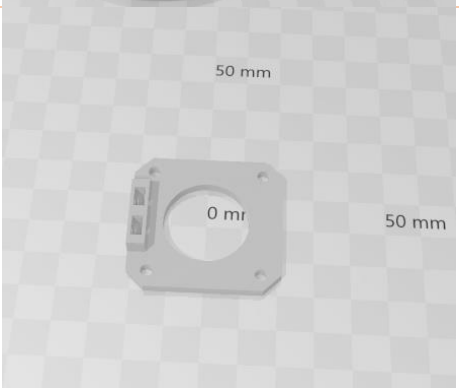
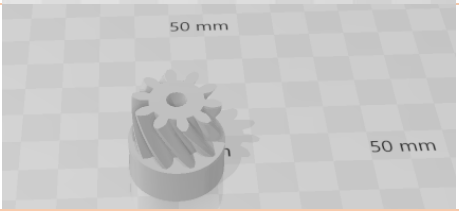
En este apartado se va a detallar qué partes hace falta imprimir para construir el robot, la página http://thor.angel-lm.com/ se encuentran los archivos .stl del robot a excepción de un par de piezas que hemos modificado para nuestro caso concreto, que veremos en detalle más adelante. En esta tabla se incluye las piezas y la cantidad que hay que imprimir de ellas.

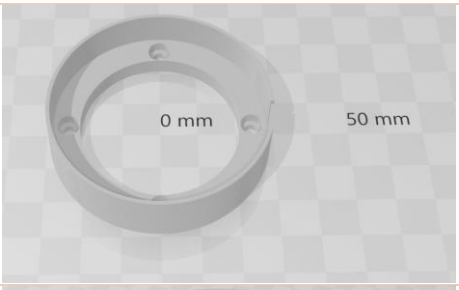
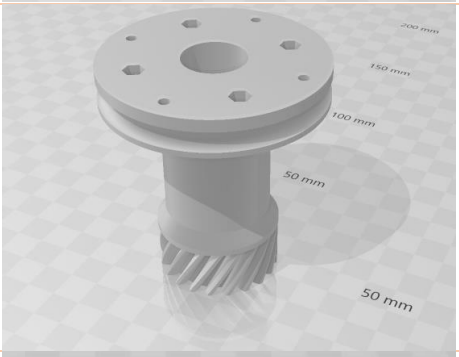
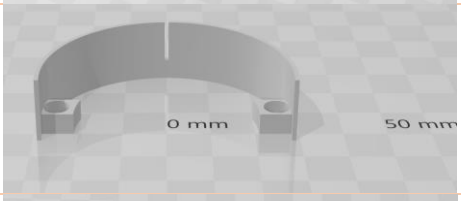
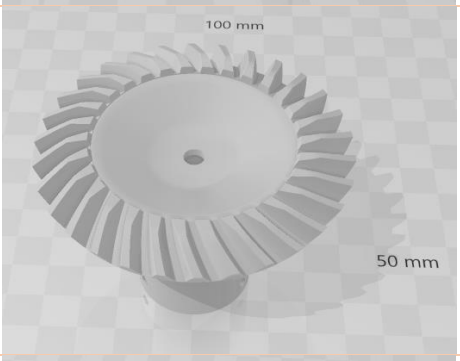
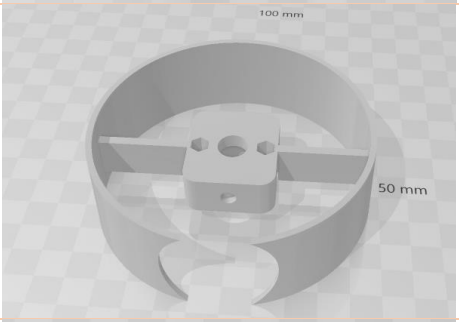
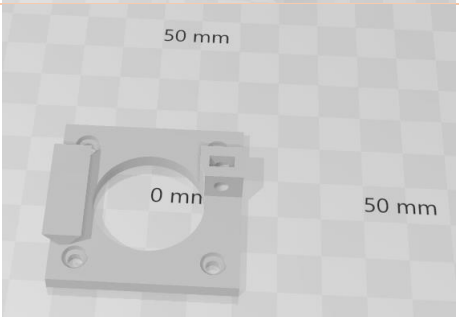
Pieza	Cantidad	Imagen
Art1Body	1	
Art1Bot	1	

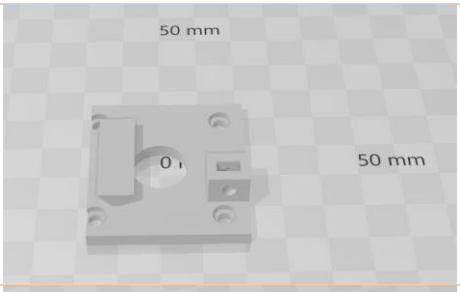
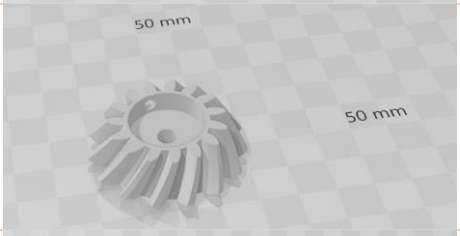
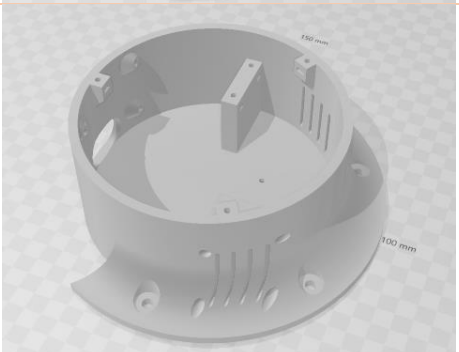
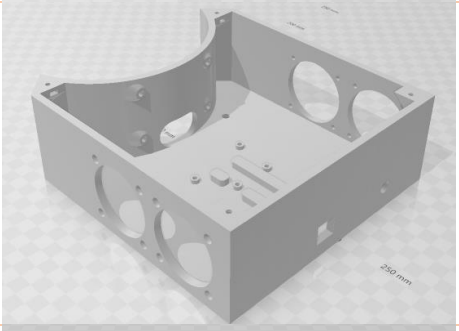
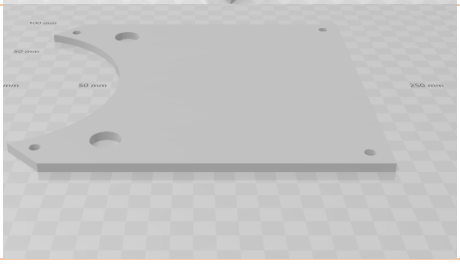
Art1GearMotor	1		
Art1Top	1		
Art2BodyA	1		
Art2BodyACover1	1		
Art2BodyACover2	1		

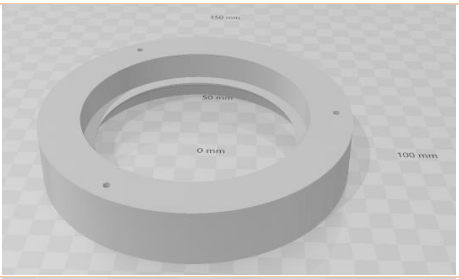
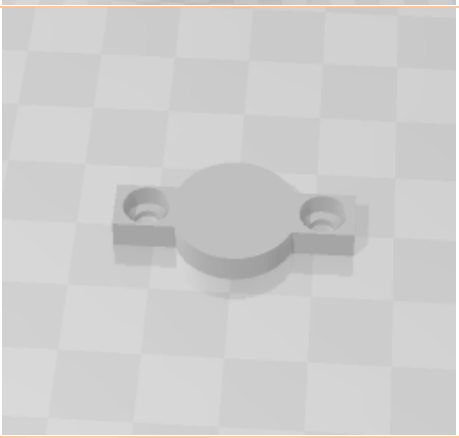
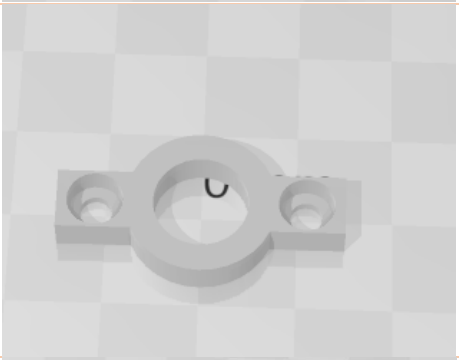
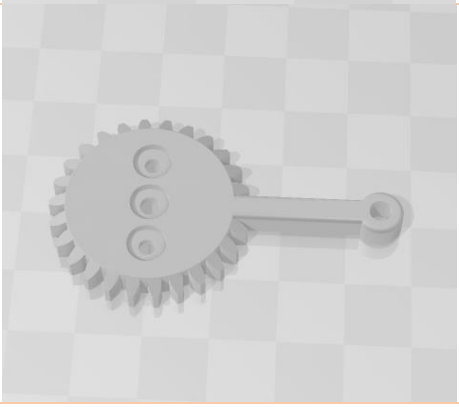
Art2BodyAWindow	1		
Art2BodyB	1		
Art2BodyBCover	1		
Art2BodyUnion	1		
Art2MotorGear	2		

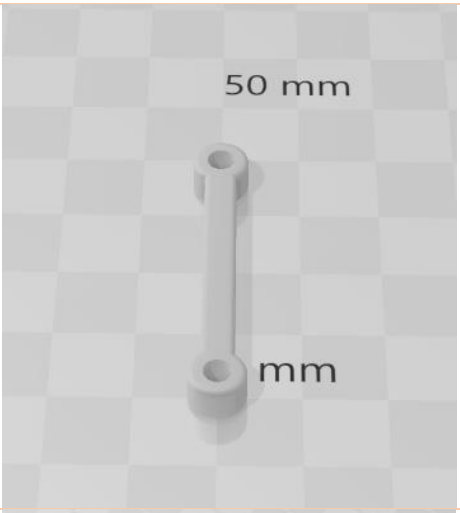
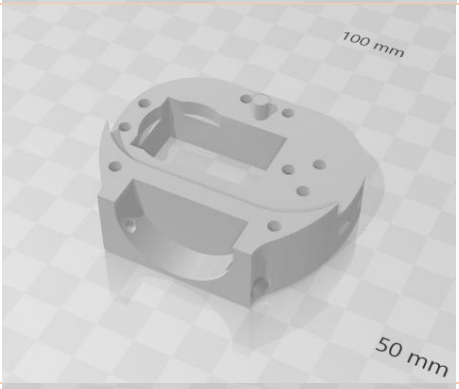
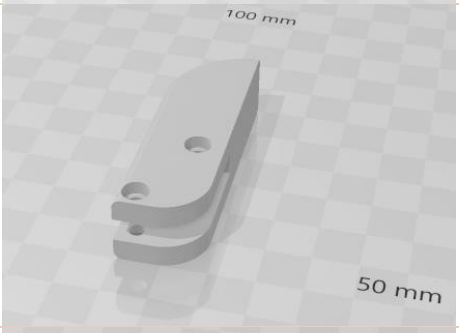
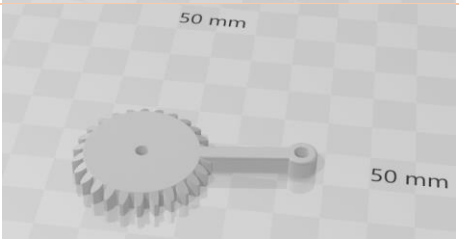
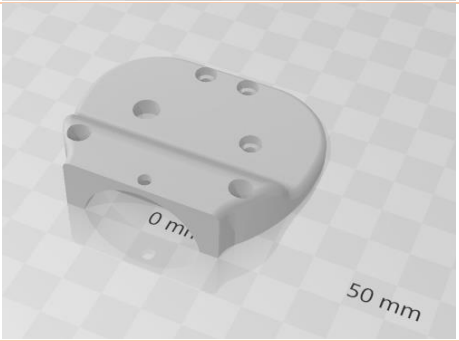
Art3Body	1		
Art3Pulley	1		
Art3TensionerBody	2		
Art3TensionerPulley	2		
Art4BearingFix	2		

Art4BearingPlug	1		
Art4BearingRing	1		
Art4Body	1		
Art4BodyBot	1		
Art4MotorFix	1		
Art4MotorGear	1		

Art4OptoDisk *	1		
Art4TransmissionColumn	1		
Art23Optodisk	2		
Art56GearPlate	1		
Art56MotorCoverRing	1		
Art56MotorHolderA	2		

Art56MotorHolderB	2		
Art56SmallGear	2		
BaseBearingFix	2		
BaseBot	1		
BaseBoxBody	1		
BaseBoxCover	1		

BaseTop	1		
CommonBearingFix	4		
CommonBearingFixThrough	4		
GripperActiveArm	1		

GripperArm	2		
GripperBot	1		
GripperFinger	2		
GripperPassiveArm	1		
GripperTop	1		

(*) Art4OptoDisk En nuestro caso no se ha impreso esta, sino que se ha modificado, como veremos en el siguiente apartado.

Piezas modificadas y nuevas

Art4OptoDisk se ha modificado reduciéndose 0.1mm de radio, pasando de 29.3mm a 29.2mm . Solo será necesario imprimir 1

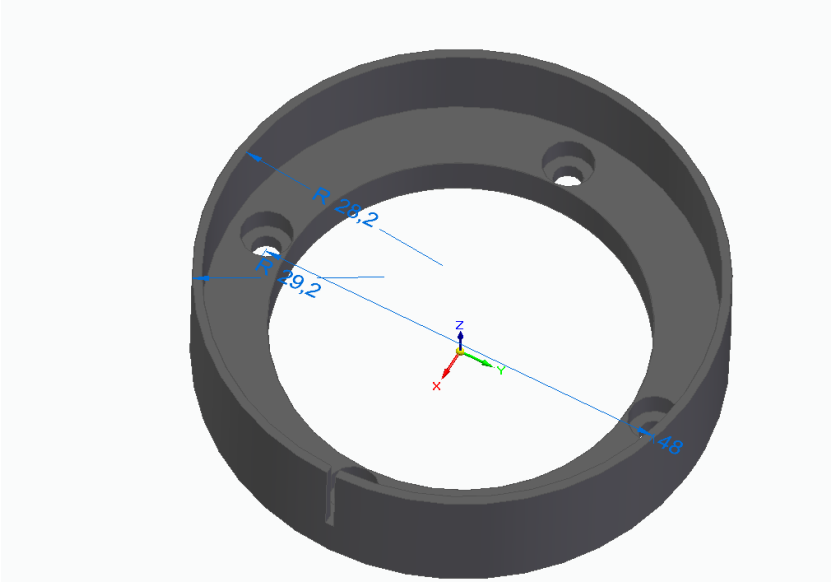


Ilustración 58 Art4OptoDisk modificación

SeparadorArt56 se utiliza para formar la articulación 56 de forma más cómoda. Ya que en la varilla hay un espacio libre y se mueven las piezas haciendo más complicado montarla. Para eliminar el hueco se ha diseñado un separador. Con la forma de la imagen, siendo 7mm de alto, 18 mm de radio total y 6mm de radio del agujero.

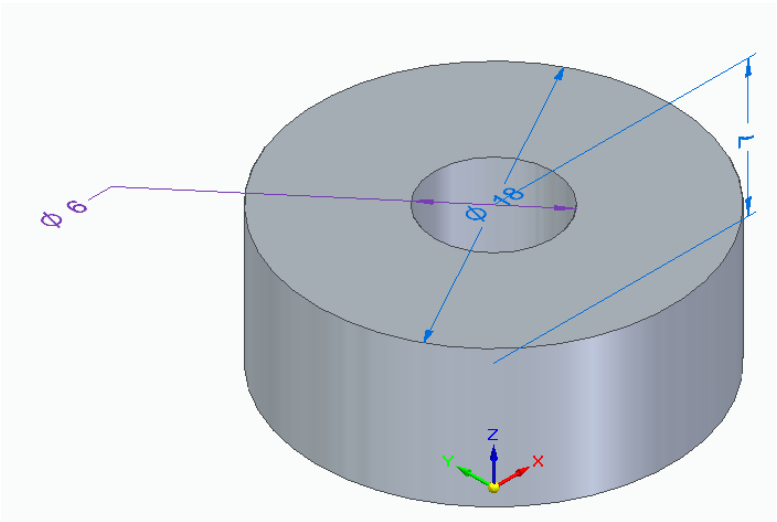


Ilustración 59 Art56 Separador

Montaje Paso a Paso

En este apartado veremos el montaje paso a paso del robot con imágenes para que se puede ver mejor que estamos explicando. Todo el montaje está más detallado en el anexo VI, montaje de Thor. Existen unos ficheros Freecad donde podemos ver todo el robot montado donde podremos ocultar partes poder ver otras partes interiores y poder saber dónde colocarlas.

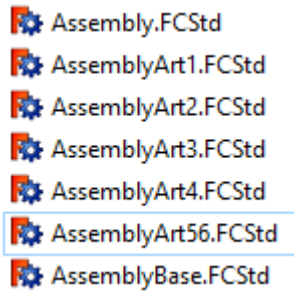


Ilustración 60 Archivos assembly freecad

Para poder abrir estos archivos necesitaremos un programa gratuito llamado Freecad. Se puede descargar aquí: https://www.freecadweb.org/

Aquí tenemos un ejemplo de cómo se ve la aplicación con el montaje de art4.

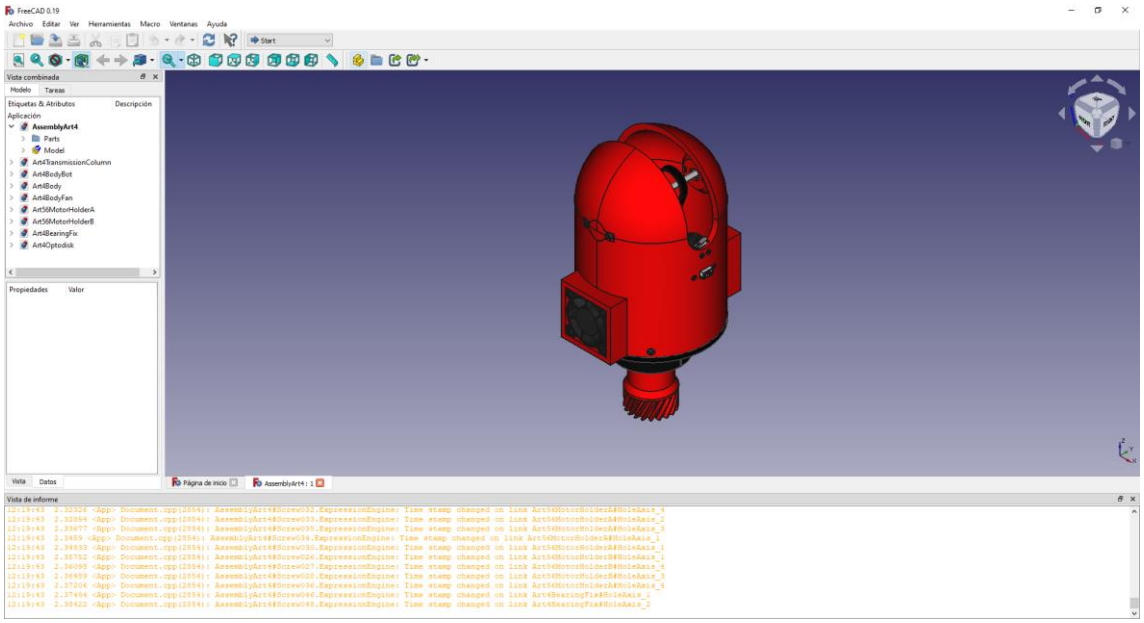


Ilustración 61 Vista freecad assemblyArt4

Articulaciones 56

Para empezar con el montaje montaremos la articulación56.

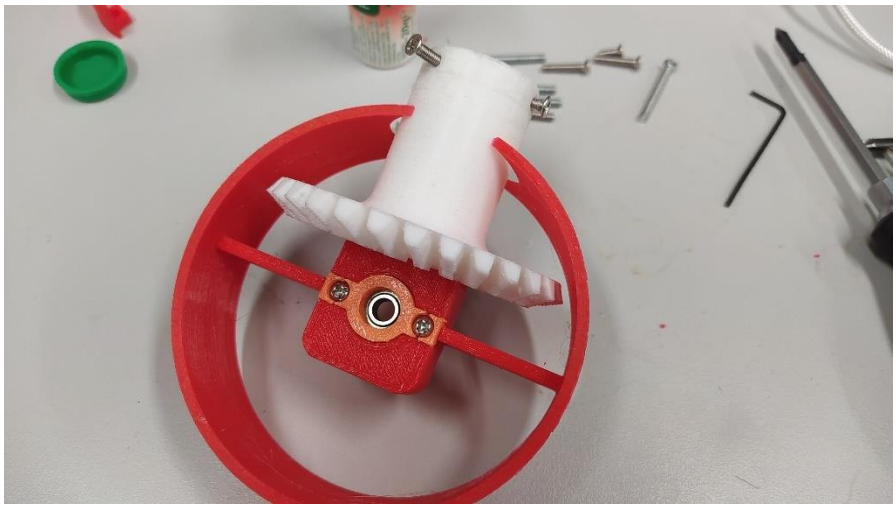


Ilustración 62 Art56MotorCoverRing

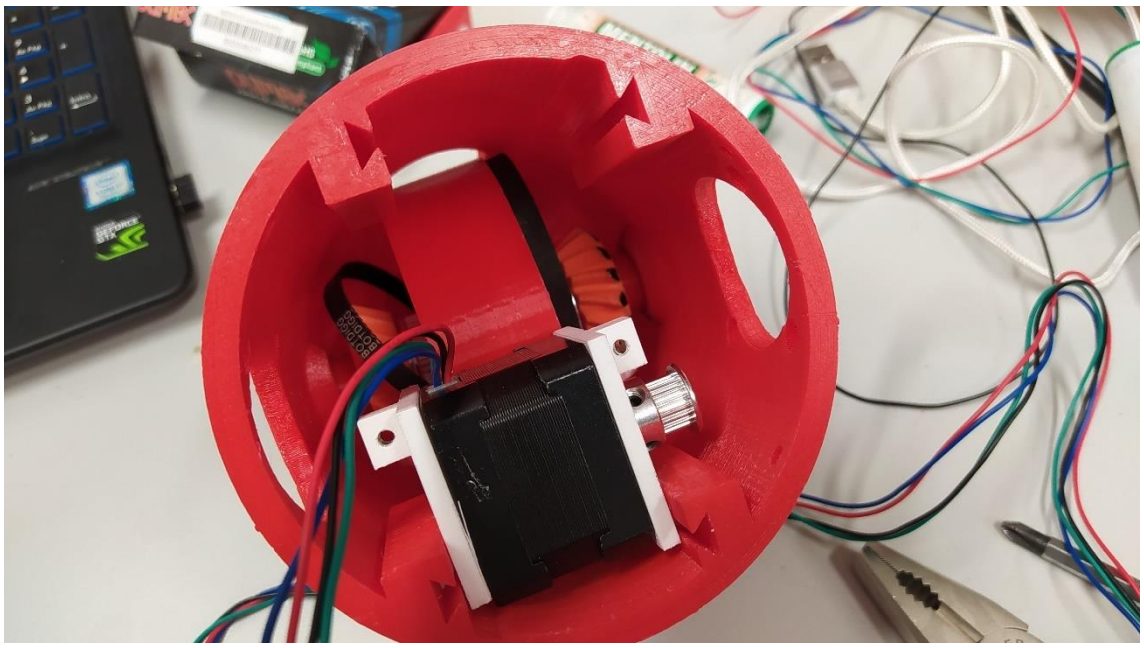


Ilustración 63 art56 con las piezas añadidas a la varilla

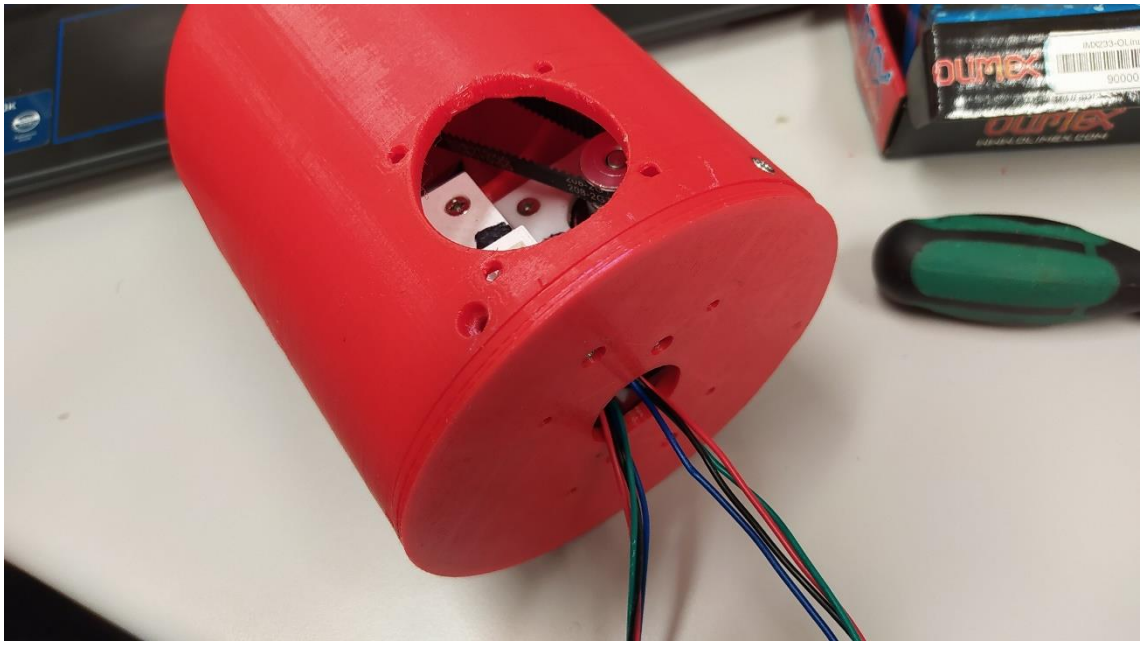


Ilustración 64 Art 56 finalizada

Añadiremos la tapa Art4BodyBot debajo para cerrar la articulación

Articulación 4

Ilustración 65 Art4 Columna de transmisión

Utilizando bolas de plástico o metal de 6 mm, Art4TransmissionColumn y Art4BearingRing y cerraremos el agujero de las bolas con Art4BearingPlug para que no se caigan. En nuestro caso Art4BearingPlug se ha impreso TPU ya que es un material flexible y nos permite que el tapón cierre a presión.

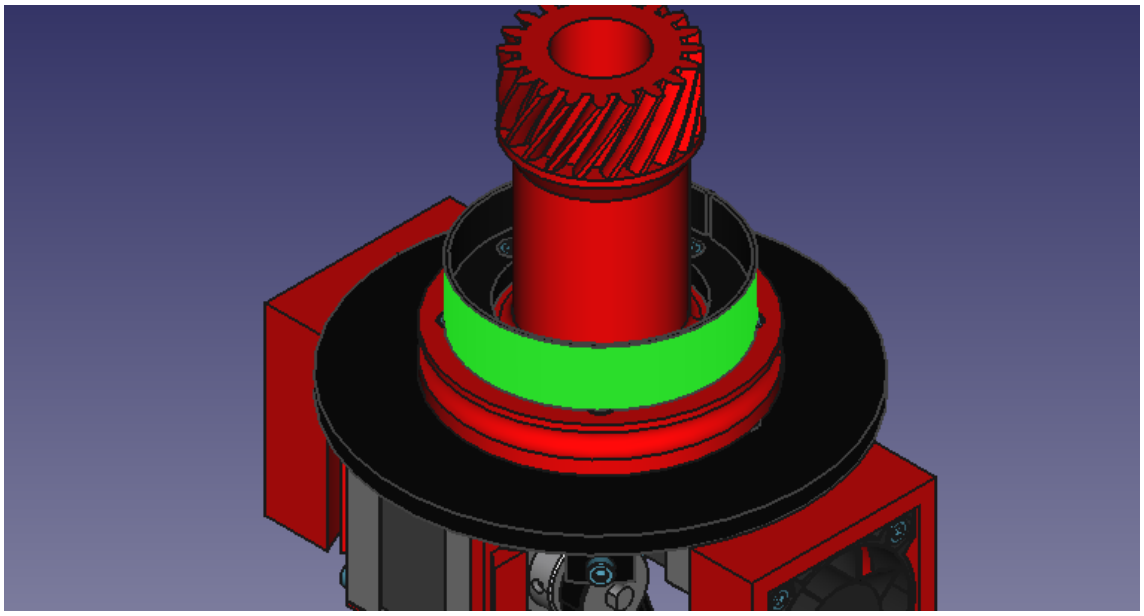


Ilustración 66 Optodisk

Se unen a la articulación 56 para formar la pieza de la imagen.

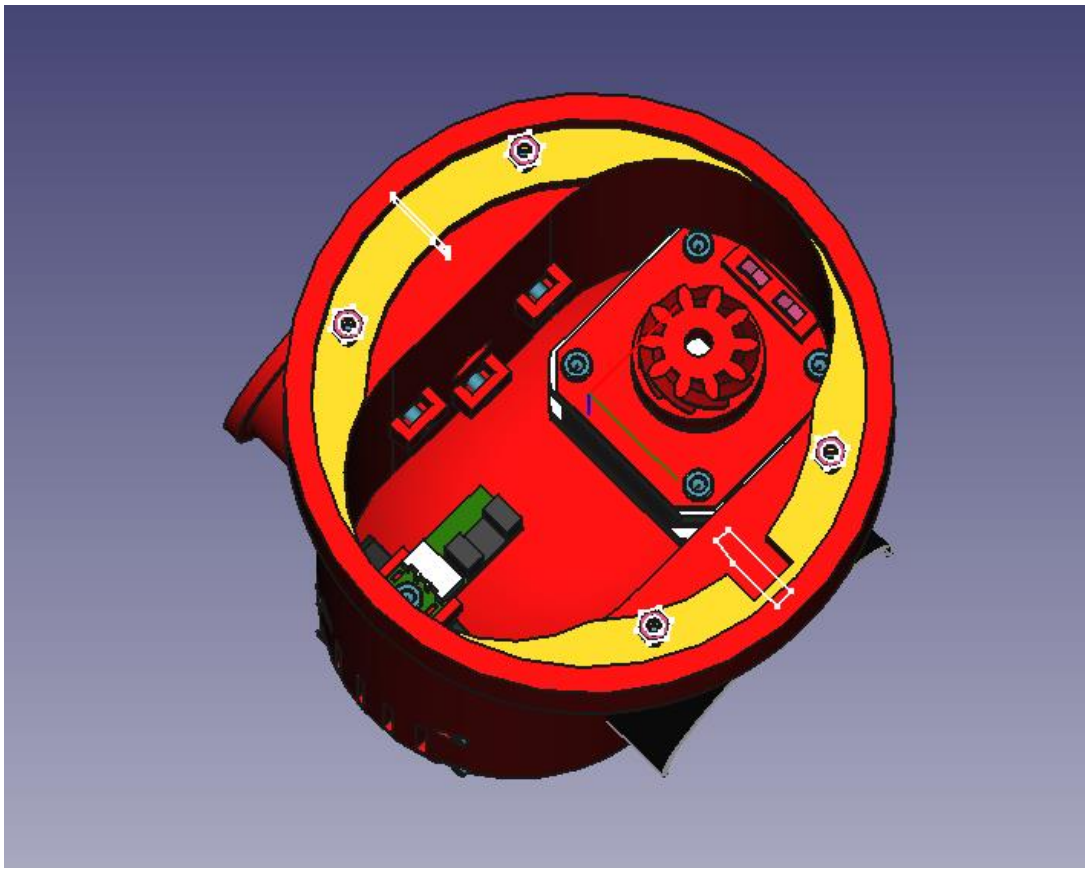


Ilustración 67Art4 parte de abajo

Usando la pieza Art3Body como base montaremos lo que se ve en la imagen. Con un motor, Art4MotorGear, Art4MotorFix y un sensor óptico de posición.

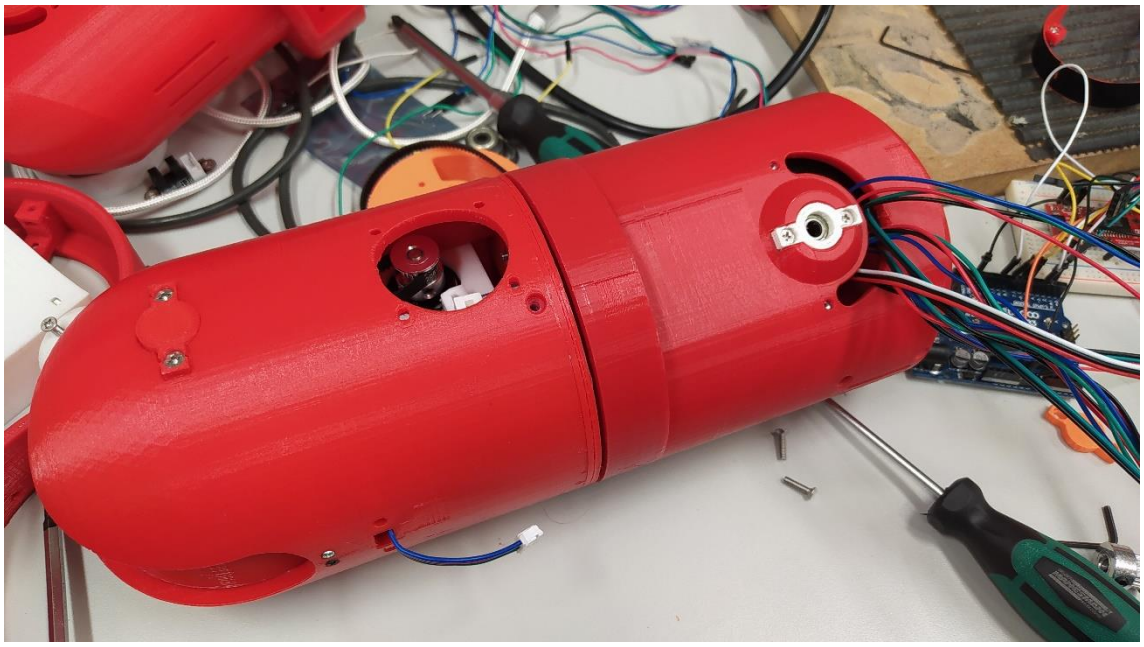


Ilustración 68 Art56 y art4 unidas

Se unen las dos partes.

Articulaciones 23

Ilustración 69 Añadido Art2BodyUnion

Utilizando las piezas de la parte derecha de la articulación 2-3 uniremos art1 body y art4

Se añaden los motores a la pieza Art1Body, también se le añade también un Art23Optodisk.

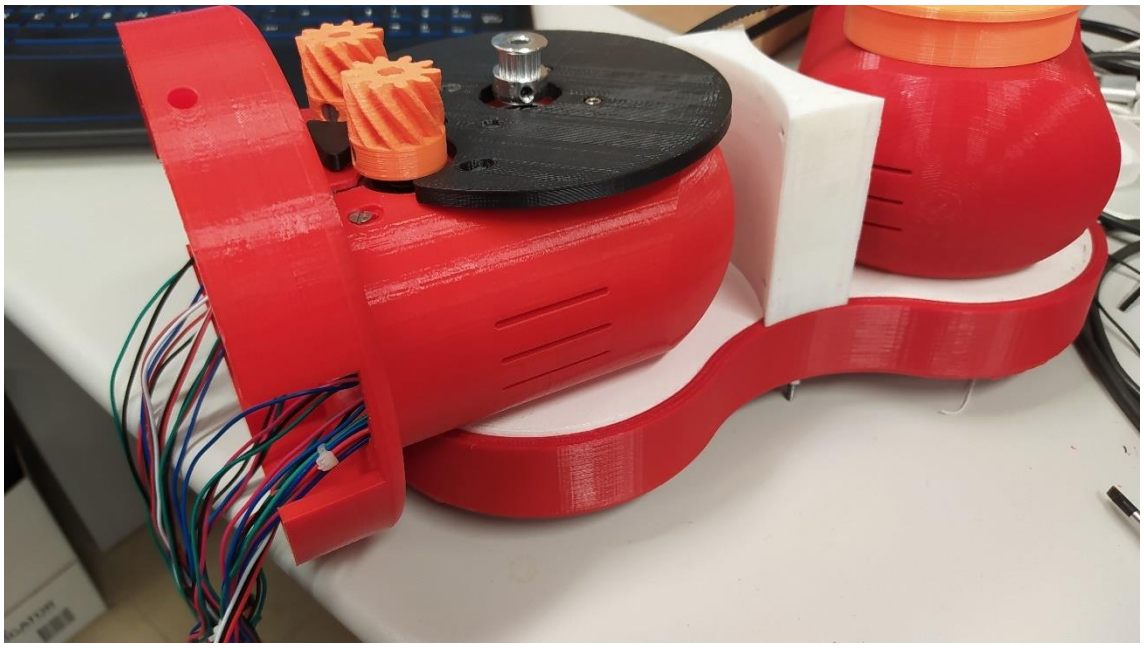


Ilustración 70 Art1Body sobre Art2BodyB

Se montan las piezas de la parte izquierda de art 23.

Ilustración 71 Añadir Art2BodyACover2 y Art3Pulley

Se añade Art3Pulley a Art1Body y Art2BodyACover2. A Art3Pulley se le une una correa gt2.

Ilustración 72 Unir la correa art3

Se une la correa gt2 a la polea.

Con esto tendremos la parte superior del robot montada y preparada para ser montada en la base.

Base

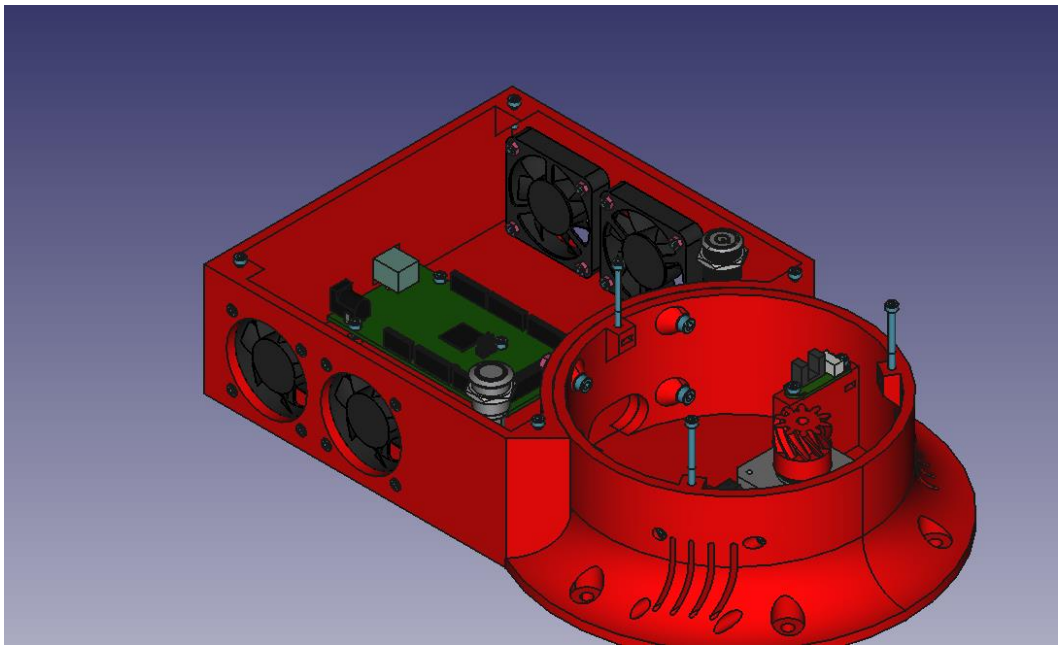


Ilustración 73 Base Primera parte

Para comenzar a montaremos las piezas de la base con el motor y la electrónica.

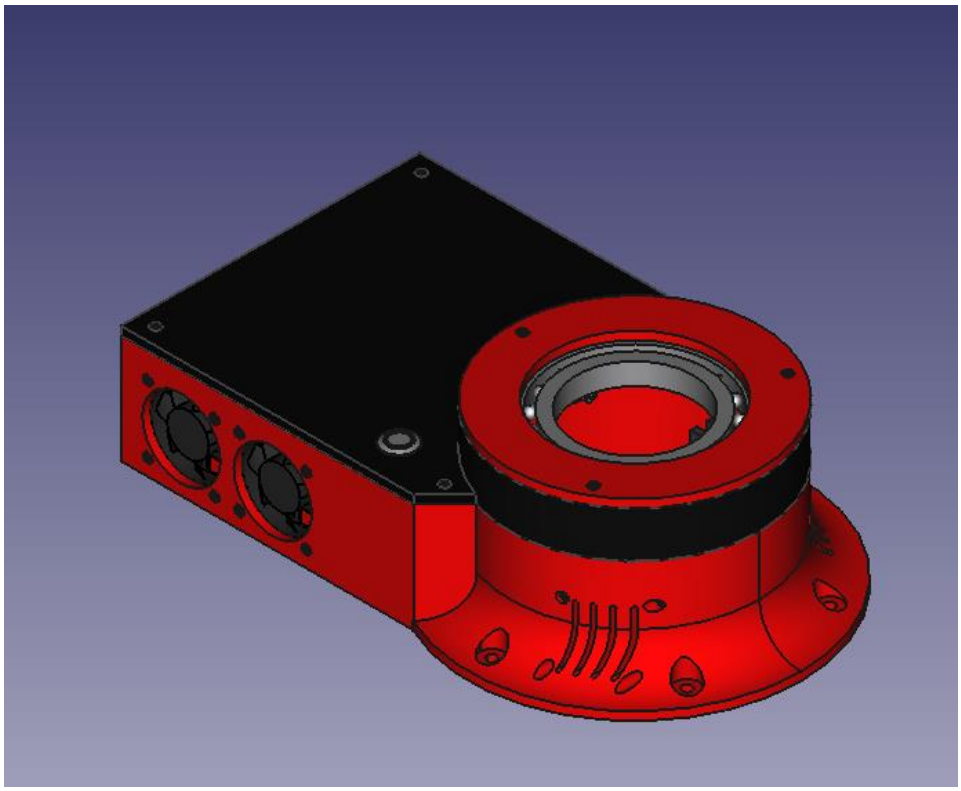


Ilustración 74 BaseBearingFix

Se añaden las piezas móviles de la articulación 1.

Uniremos las dos partes del robot.

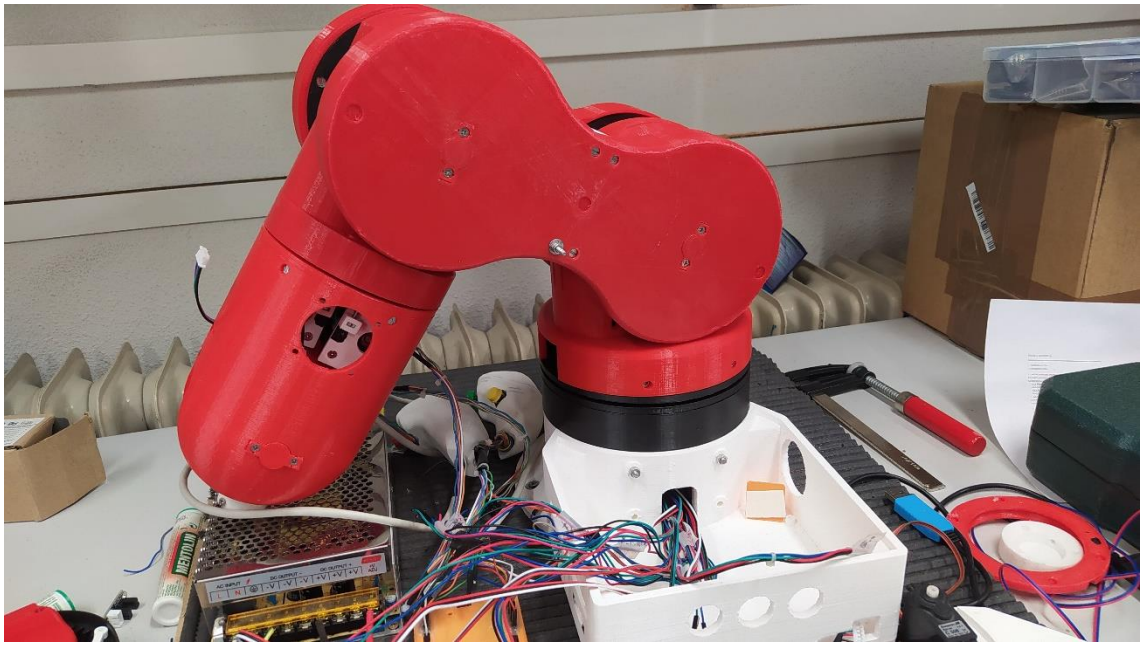


Ilustración 75 Brazo montado

Pinza

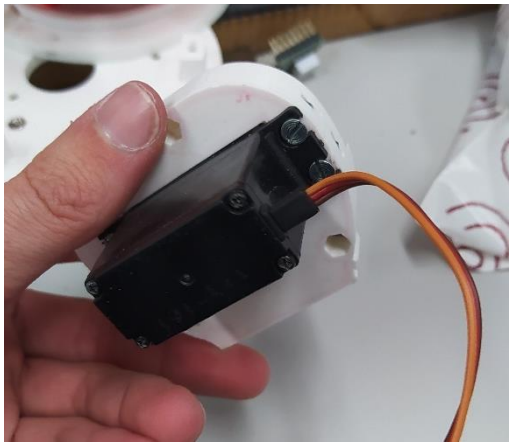


Ilustración 76 GripperBot y servo

Para montar la pinza empezaremos uniendo el servo con las piezas.

Ilustración 77 Pinza montada

Terminaremos el montaje de la pinza haciendo uso de GripperTop, que colocaremos encima de GripperBot. Así nuestra pinza estará lista para ser instalada en el brazo. Aquí un video de muestra de funcionamiento de la pinza.

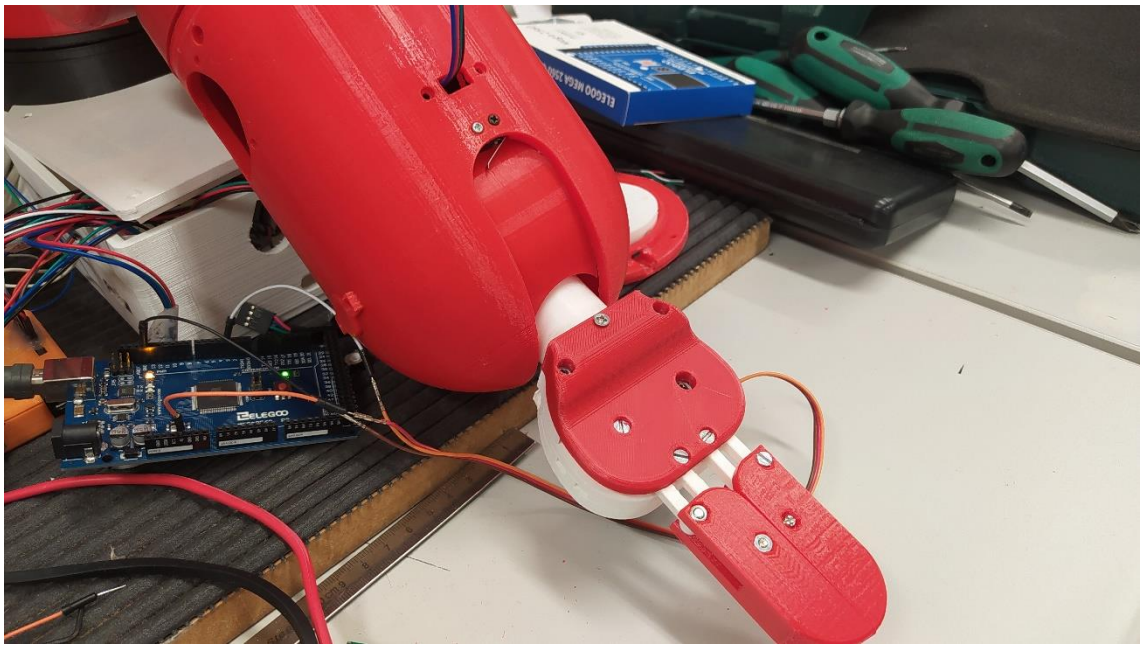


Ilustración 78 Pinza instalada

4.5 Implementación del sistema

Para la implementación de los dos sistemas se ha utilizado dos entornos completamente distintos. Para la implementación del ClienteThor se ha utilizado .net y se ha creado una interfaz de usuario con WPF. En cuanto a RobotThor se ha utilizado el lenguaje Arduino para programar una placa Arduino Mega. Se ha creado la documentación técnica de programación en el Anexo IV.

4.5.1 Cliente Thor

El cliente es un programa creado en Visual Studio 2017. Se ha utilizado el lenguaje de programación .net y se ha creado una interfaz gráfica con WPF. Es el encargado de leer ficheros de secuencia e interactuar con el usuario. En el Anexo 4 se detalla cuales con los métodos principales de las clases del programa y como se ha montado el proyecto.

4.5.2 Robot Thor

Robot Thor es el software utilizado en la placa controladora para controlar el robot. Se ha utilizado el lenguaje de programación de Arduino, para el desarrollo se ha utilizado el programa Arduino IDE. Recibe comandos del cliente Thor y mueve el robot dependiendo de que comandos reciba. En el anexo 4 se detalla como son las funciones principales y como se ha montado el proyecto.

4.5.3 Comunicación Cliente-Robot

Para la comunicación entre el cliente y el robot se ha utilizado una conexión USB. USB como puerto serie, se puede controlar mediante algunas funciones ya integradas tanto en .net como en Arduino. Se han definido distintos tipos de comandos, los principales son:

- Limbs: El cliente envía este comando al robot y la controladora devuelve las posiciones actuales de los ejes.
- Mover eje: El cliente envía este comando indicando el eje y la posición, el robot devuelve una confirmación.

Todo esto está mucho más detallado en el anexo 4.

4.5.4 Cálculo de pasos por cada grado de giro de cada eje

Para saber de forma precisa cuantos grados gira cada articulación se han hecho cálculos de cada eje teniendo en cuenta las reducciones por poleas, engranajes o cajas reductoras del propio motor. Como premisa sabemos que cada motor Nema 17 da una vuelta completa cada 200 pasos, aunque los motores con reductora usados en los ejes 2 y 3 tienen 1000 pasos por vuelta, gracias a sus reductoras 1/5.Definimos como dientesMotor a los dientes del engranaje acoplado al motor y dientesActuador a los dientes del engranaje conectado al actuador. A partir de ahí multiplicaremos por las reducciones. De la siguiente forma:

$$grados/paso = \frac{\text{gadros vuelta entera } (360) * \text{dientesMotor/dientesActuador}}{\text{numero de pasos para una vuelta del motor}}$$

Si no existen dientes en alguna polea se realizará el cálculo para obtener que numero de dientes sería equivalente teniendo en cuenta las medidas de la misma.

Cadera (Eje1)

Engranaje del motor 10 dientes.

Engranaje del actuador 50 dientes.

$$\frac{360 * \frac{10}{50}}{200} = 0.36\ grados/paso$$

Hombro (Eje2)

En este caso los motores 1000 pasos por a reductora 1/5.

Engranaje del motor 10 dientes.

Engranaje del actuador 60 dientes.

$$\frac{360 * \frac{10}{60}}{1000} = 0.06\ graados/paso$$

Codo (Eje3)

Este eje funciona con dos poleas, una dentada para el motor y otra impresa en 3D para el actuador por lo que no tenemos un número fijo de dientes que contar. Pero sabemos que la separación entre dientes en las poleas GT2 es de 2mm. Y al medir la polea del actuador sabemos que tiene 37 mm de radio.

Calculamos el perímetro de la polea del actuador con la siguiente formula: 2 πR, por lo que:

2 π 37 = 232.5 mm de perímetro.

Como la separación entre dientes de la polea GT2 es de 2mm, si dividimos el perímetro de la polea del actuador por 2, obtendremos el numero dientes equivalentes a si fuera una polea dentada. 232.5/2 = 116 dientes.

Ilustración 79 Polea GT2

Ahora ya podemos hacer el cálculo.

En este caso los motores 1000 pasos por a reductora 1/5.

Polea del motor 20 dientes.

Polea del actuador 116 dientes.

$$\frac{360 * \frac{20}{116}}{1000} = 0.062\ grados/paso$$

Muñeca (Eje4)

Engranaje del motor 10 dientes.

Engranaje del actuador 20 dientes.

$$\frac{360 * \frac{10}{20}}{200} = 0.9\ grados/paso$$

Muñeca 2 (Eje5)

Polea del motor 20 dientes.

Polea del actuador 40 dientes.

$$\frac{360 * \frac{20}{40}}{200} = 0.9\ grados/paso$$

Muñeca 3(Eje6)

Polea del motor 20 dientes.

Polea del actuador 40 dientes.

Engranaje del motor 15 dientes.

Engranaje del actuador 30 dientes.

A pesar de tener dos puntos de reducción se aplica de la misma forma:

$$\frac{360 * \frac{20}{40} * \frac{15}{30}}{200} = 0.45\ grados/paso$$

Aplicado a la implementación

Haciendo la simple división de los grados por el numero grados / paso, podemos obtener el número de pasos necesarios para mover cierta cantidad de grados cada eje. Como se puede apreciar en las funciones son super sencillas, esto está dentro del Arduino, puesto que el cliente solo trabaja con valores entre 0 y 1000.

```
/*funciones para la concersion de grados a pasos*/
/*eje1*/
int AngulosAPasosEje1(float grados) {
  return round(grados / 0.36f);
}

/*eje2*/
int AngulosAPasosEje2(float grados) {
  return round(grados / 0.06f);
}

/*eje3*/
int AngulosAPasosEje3(float grados) {
  return round(grados / 0.062f);
}

/*eje4*/
int AngulosAPasosEje4(float grados) {
  return round(grados / 0.9f);
}

/*eje5*/
int AngulosAPasosEje5(float grados) {
  return round(grados / 0.9f);
}

/*eje6*/
int AngulosAPasosEje6(float grados) {
  return round(grados / 0.45f);
}
```

Ilustración 80 Funciones de gados a pasos

4.5.1 Cálculo de posición para giro suave

Puesto que es imposible que el giro de dos ejes sea completamente simultaneo, en caso de que se necesite con este algoritmo se intenta conseguir que se alternen de la forma más suave posible. Esto se realiza desde el Cliente y es transparente al robot.

¿Como funciona el algoritmo?

Cuando el Cliente lee un fichero, puede que sea necesario realizar movimiento en uno o varios ejes. Cuando es necesario mover dos o más ejes se busca que el movimiento sea lo más coordinado posible, que ambos se muevan a la vez y no secuencialmente. Para esto dividiremos el movimiento de cada eje en varias fases de movimiento mucho más cortas. Realizaremos esos movimientos de forma secuencial intercalando de forma proporcional los ejes.

- Primera fase del algoritmo: Cuanto mover cada eje en total

Mediante la función limbs que devuelve la posición actual de los ejes podemos conocer la posición de cada eje. Restando esas posiciones a nuestras posiciones objetivo podemos saber cuánto se debe desplazar cada eje.

```
int movimientoSuave(int[] eje, int[] pos, int cantidad)
{
    byte h;
    byte l;
    int inc_ref = 5;
    int[] incrementos = { 0, 0, 0, 0, 0, 0, 0 };
    int[] get_limbs = getLimbs();

    for (int i = 0; i < cantidad; i++) {
        incrementos[i] = pos[i] - get_limbs[i];
    }
```

Ilustración 81 Movimiento suave fase 1

- Segunda fase del algoritmo:

En esta fase comprobaremos que eje tiene que moverse más y dividiendo cuanto se tenga que mover por un incremento de referencia, obtendremos el número de divisiones que haremos a cada eje.

-

```
int maxIndex = Array.IndexOf(incrementos, incrementos.Max());
int lessIndex = Array.IndexOf(incrementos, incrementos.Min());

int indiceMasDistante;
if (incrementos[maxIndex] > -incrementos[lessIndex])
{
    indiceMasDistante = maxIndex;
}
else {
    indiceMasDistante = lessIndex;
}

int num_steps = Math.Abs( incrementos[indiceMasDistante]) / inc_ref;
if ((num_steps == 0)) {
    return 1;
}
```

Ilustración 82 Movimiento suave fase 2

- Tercera fase del algoritmo

Calcular cada una de las fases de cada eje y enviar la petición de cada movimiento de forma secuencial al Robot Thor.

```
float[] posCadaStep = { 0, 0, 0, 0, 0, 0, 0};
float[] ultimostep = { 0, 0, 0, 0, 0, 0, 0 };
for (int i = 0; i < cantidad; i++)
{
    posCadaStep[i] =(float) incrementos[i] / num_steps;
    ultimostep[i] = get_limbs[i];
}
for (int i = 0; i < num_steps; i++) {
    for (byte j = 0; j < cantidad; j++)
    {
        if (get_limbs[j]!=pos[j]) {
            int last = get_limbs[j];
            ultimostep[j] = ultimostep[j] + posCadaStep[j];
            get_limbs[j] = (int)Math.Round(ultimostep[j]);
            if (last != get_limbs[j])
            {
                h = Convert.ToByte(get_limbs[j] / 256);
                l = Convert.ToByte(get_limbs[j] % 256);
                if (!(RW(j, l, h)))
                {
                    return -1;
                }
            }
        }
    }
}
```

Ilustración 83 Movimiento suave fase 3

- Cuarta fase del algoritmo

Corrección de la posición final. Es posible, al usar float para sumar que al final del algoritmo se haya perdido una pequeña fracción de posición de alguno de los ejes. Por lo que enviamos de nuevo las posiciones finales para que se corrijan.

```
for (byte j = 0; j < cantidad; j++)
{
    h = Convert.ToByte(pos[j] / 256);
    l = Convert.ToByte(pos[j] % 256);
    if (!(RW(j, l, h)))
    {
        return -1;
    }
}

return 0;
```

Ilustración 84 Movimiento suave fase 4

4.6 Funcionalidades de la aplicación

La principal funcionalidad de la aplicación es el movimiento de Thor. Para ello se requieren ciertos parámetros para que funcione correctamente. Thor robot debe estar conectado a la red eléctrica y por USB a un ordenador con clienteThor instalado. Al encender el robot, este

intentará buscar los finales de carrera. Debido a limitaciones de diseño del robot, este debe estar en una posición inicial aproximada. Sobre todo, los ejes 2 y 3 porque no pueden dar una vuelta completa. En cuanto al resto de ejes es recomendable tener cuidado porque si da muchas vueltas hacia el mismo lado se enredan los cables. Posición correcta de los ejes 2 y 3, para distinguir un lado del otro nos podemos fijar en la tapa blanca de lateral.

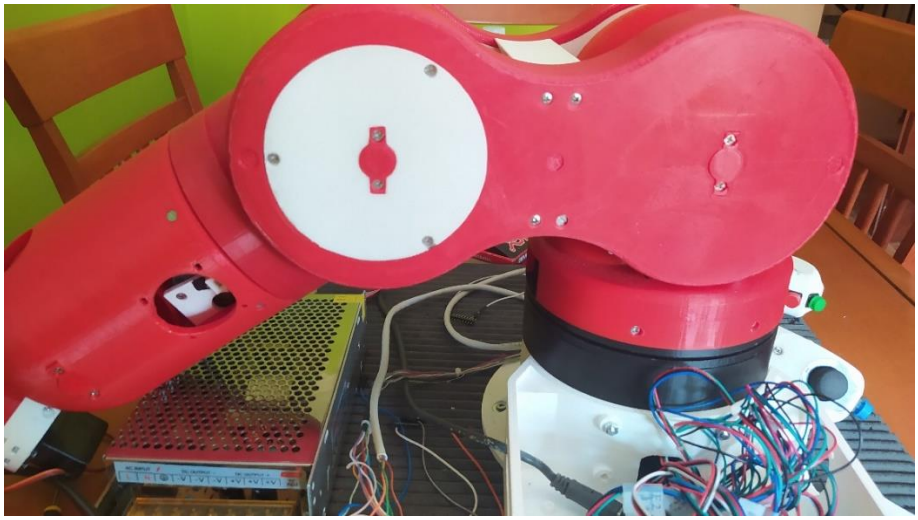


Ilustración 85 Posición inicial Thor

Aquí un video del robot moviéndose a la posición inicial

Una vez iniciado el robot y correctamente en su posición podemos ejecutar la aplicación Cliente Thor. Aquí debemos elegir antes de nada el puerto al que está conectado nuestro robot mediante este combo:

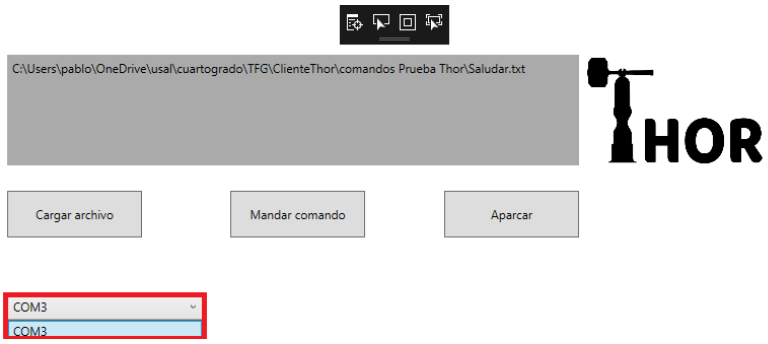


Ilustración 86 Combo Cliente thor

Después elegiremos que archivo de secuencia de pasos vamos a mandar a nuestro robot con el botón 'cargar archivo'. Nos mostrará un dialogo de selección de fichero.

Una vez elegido nuestro fichero mandaremos las secuencias con el botón 'Mandar comando'. Esto encadenará los movimientos en ThorRobot. Video aquí

Cuando hayamos finalizado, podremos utilizar el botón de aparcar para colocar a Thor en una posición segura para apagarlo: Video aquí

5 Conclusiones

Robot Thor cumple con los objetivos iniciales y requisitos que se plantearon en las fases iniciales del desarrollo del proyecto.

Se han conseguido los siguientes objetivos:

- **Impresión de las piezas del robot**
- **Montaje del robot**
- **Elección de componentes electrónicos**
- **Conectar componentes electrónicos**
- **Desarrollo de software para la placa controladora**
- **Desarrollo de software del cliente Thor**
- **Desarrollo de un protocolo de comunicación entre el cliente Thor y la placa controladora**

También se ha realizado toda la documentación asociada a todos los aspectos del proyecto.

Se ha buscado en la medida de lo posible que ambas partes del proyecto sean compatibles con otros sistemas. Por ejemplo, poder utilizar programas desarrollados para ser utilizados con otros robots o que nuestro robot sea fácil de controlar desde cualquier otro cliente.

Las mayores dificultades de este proyecto han sido principalmente relacionadas con el montaje del robot. Las piezas impresas en 3D son relativamente frágiles y se han roto algunas piezas que han tenido que ser impresas de nuevo con mejor calidad (más relleno y paredes más gruesas). También ha habido dificultades relacionadas con la electrónica, como anécdota, la placa controladora inicial con la que se pensó para este proyecto era una placa de impresión 3D, pero tras una mala conexión la placa quedó inutilizada y decidimos utilizar el sistema que planteó AngelLM.

Personalmente me ha gustado mucho hacer este proyecto. Conocía la tecnología de impresión 3D, pero nunca lo había probado por mí mismo y me ha parecido una tecnología tan interesante que me he comprado una impresora 3D para uso personal. En cuanto al desarrollo de Arduino, ya conocía la tecnología y la había usado, pero con este proyecto he aprendido muchas cosas sobre ella. Y en cuanto a la tecnología .net ya la conocía de la asignatura Interfaces Gráficas, pero aun así he aprendido nuevos aspectos como, por ejemplo, el uso del puerto serie a través de ella.

6 Líneas de trabajo futuras

Como líneas de trabajo futuras se abren varias opciones:

Añadir control con joysticks, mediante una placa Arduino uno. Podemos leer los movimientos de unos joysticks y sin tocar el código del robot Arduino podemos comunicarnos mediante puerto serie.

Control del robot a través de un módulo bluetooth HC-05. Puesto que estos módulos funcionan igual que un puerto serie, se puede conectar uno de ellos a nuestro Arduino Mega y utilizarlo como puerto de comunicación. Se podría utilizar los puertos 'RX' y 'TX' del Arduino Mega para poder conectarse sin tener que cambiar nada en el código. También sería interesante modificar el código para utilizar otros puertos como 'RX1' y 'TX1' y dejar ese libre para el USB. E incluso permitir leer comandos de ambas fuentes.

Control del robot mediante software creado para otros robots del departamento. El departamento tiene software para el control de otros brazos robóticos, ya que este software incluye control de robots otras funcionalidades como creación de secuencias de pasos de movimientos.

Referencias

https://github.com/AngelLM/Thor

https://es.wikipedia.org/wiki/Impresi%C3%B3n_3D

https://www.freecadweb.org/

https://ultimaker.com/es/software/ultimaker-cura

https://reprap.org/wiki/Mechanical_Endstop

https://www.arduino.cc/

https://es.wikipedia.org/wiki/Brazo_rob%C3%B3tico

https://www.hwlibre.com/wp-content/uploads/2020/03/pinout-arduino-mega.jpg

https://www.luisllamas.es/usar-un-optointerruptor-con-arduino/

https://www.diarioelectronicohoy.com/blog/descripcion-del-driver-a4988

http://thor.angel-lm.com/documentation/electronics/

https://www.pcbway.es/

Videos

https://www.youtube.com/watch?v=Cg-D_c0iNwI

https://youtube.com/shorts/PqVLH58eWAQ?feature=share

https://www.youtube.com/watch?v=zck94UHz4Fo