



Escuela superior de ciencias experimentales y tecnología

**GRADO EN INGENIERÍA
DE TECNOLOGÍAS INDUSTRIALES**

Trabajo fin de grado

**AUTOMATIZACIÓN DE UN ENSAYO DE MODELO
ANÁLOGO GEOLÓGICO UTILIZANDO HARDWARE
LIBRE**

Adrián Zeus Román García

Director: Felipe Machado Sánchez

Curso académico 2019/2020

[Escriba aquí]

[Escriba aquí]

[Escriba aquí]

[Escriba aquí]



Grado en ingeniería en tecnologías industriales

Trabajo de Fin de Grado (TFG)

El presente trabajo, titulado “**AUTOMATIZACIÓN DE UN ENSAYO DE MODELO ANÁLOGO GEOLÓGICO UTILIZANDO HARDWARE LIBRE**”, constituye la memoria correspondiente a la asignatura Trabajo de Fin de Grado de D. Adrián Zeus Román García como parte de su formación para obtener el título de graduado en ingeniería en Tecnologías Industriales. Este trabajo ha sido realizado en el **Campus de Móstoles de la URJC** en el **departamento de electrónica** bajo la dirección de **Felipe Machado Sánchez**

Móstoles 18 junio de 2020

Índice

| | | |
|--------|---|--------------------------------------|
| 1. | Resumen | 7 |
| 2. | Introducción | 8 |
| 2.1. | Ensayo de modelo análogo..... | 8 |
| 2.2. | Hardware libre, software libre y la filosofía DIY | 10 |
| 2.3. | GitHub..... | 11 |
| 3. | Objetivos | 12 |
| 4. | Resultados | ;Error! Marcador no definido. |
| 5. | Accesibilidad y utilización del proyecto | ;Error! Marcador no definido. |
| 6. | Análisis y montaje electrónico | 26 |
| 6.1. | Componentes del circuito eléctrico | 26 |
| 6.1.1. | ARDUINO MEGA 2560..... | 26 |
| 6.1.2. | Controlador DRV8825 | 30 |
| 6.1.3. | Motor NEMA23 | 31 |
| 6.1.4. | Pantalla LCD | 32 |
| 6.2. | Trucos para la programación con Arduino..... | 34 |
| 6.2.1. | Programar una pantalla LCD..... | 34 |
| 6.2.2. | Programar un Encoder..... | 34 |
| 6.2.3. | Programación de estados de forma optima..... | 38 |
| 6.2.4. | Error en la medida de la velocidad..... | 40 |
| 6.2.5. | Optimización del programa..... | 40 |
| 6.2.6. | Complemento a 2 y representación de variables de gran valor | 42 |
| 7. | Análisis mecánico del diseño | 44 |
| 7.1. | Partes del diseño..... | 44 |
| 7.1.1. | Sistema de transmisión..... | 44 |
| 7.1.2. | Sistema de apoyo..... | 44 |
| 7.1.3. | Puente móvil..... | 45 |
| 7.1.4. | Estructura de la base de la maquina | 45 |
| 7.2. | Análisis mecánico de la estructura | 46 |
| 7.2.1. | Análisis de las velocidades transmitidas | 46 |
| 7.2.2. | Análisis de la fuerza transmitida | 47 |
| 7.3. | Diseño de piezas para imprimir 3D en Freecad | 48 |
| 7.4. | Diseño de planos con Freecad | 50 |
| 7.5. | Uso mods para Freecad para facilitar el diseño..... | 51 |
| 8. | Presupuesto | 53 |
| 8.1. | Mediciones | 53 |

| | |
|--|----|
| Figura 1: caja de modelado | 9 |
| Figura 2 sistema de modelado | 10 |
| Figura 3: modelo CAD del diseño del ensayo de modelo análogo geológico simple | 13 |
| Figura 4: modelo CAD del prototipo | 13 |
| Figura 5 modelo CAD del soporte del fin de carrera | 13 |
| Figura 6: modelo CAD del soporte de la tuerca del husillo | 13 |
| Figura 7: modelo CAD del soporte del cojinete | 13 |
| Figura 8: modelo CAD del soporte del motor | 13 |
| Figura 9: pantalla LCD con encoder y buzzer | 14 |
| Figura 10: Diagrama de estados del programa del ensayo | 14 |
| Figura 11: pantalla LCD en el estado 0 | 15 |
| Figura 12: pantalla LCD en el estado 1 | 15 |
| Figura 13: pantalla LCD en el estado 2 | 16 |
| Figura 14: pantalla LCD en estado 3 | 16 |
| Figura 15: pantalla LCD en estado 4 | 17 |
| Figura 16: pantalla LCD en estado 5 | 17 |
| Figura 17: modelo CAD del diseño del ensayo análogo geológico doble | 18 |
| figura 18: Diagrama de pantallas en el programa de ensayo doble | 19 |
| figura 19: Paginas 1 y 2 del estado de definición de variables | 19 |
| figura 20: Pantalla LCD durante el movimiento de motores | 20 |
| figura 21: comparativa entre la pantalla de pausa del diseño simple y el doble | 20 |
| figura 22: Interfaz del programa Ultimaker Cura | 21 |
| figura 23: Ilustración de como girar una figura en Ultimaker Cura | 22 |
| figura 24: modelo listo para imprimir | 22 |
| figura 25: lista de piezas necesarias para construir el ensayo | 23 |
| figura 26: Etapas del montaje | 23 |
| figura 27: Ejemplo de planos de este proyecto | 24 |
| figura 28: ilustración de como cargar un programa en el Arduino | 25 |
| figura 29: Esquema del circuito eléctrico | 26 |
| figura 30: Arduino MEGA 2560 | 27 |
| figura 31: Correspondencia entre los pines de la CNC shield y los del Arduino MEGA 2560 .. | 28 |
| figura 32: Controlador DRV8825 | 30 |
| figura 33: Circuito de instalación de un DRV8825 | 30 |
| figura 34: Interior de un motor paso a paso | 31 |
| figura 35: Medidas del motor NEMA 23 | 31 |
| figura 36: curva Torque frente a frecuencia del motor NEMA 23 | 32 |
| figura 37: Pantalla LCD DISCOUNT | 33 |
| figura 38: Funcionamiento de un encoder | 35 |
| figura 39: señales de un encoder | 35 |
| figura 40: Código para discriminar entre el giro horario o antihorario | 39 |
| figura 41: Código para la selección de estados | 39 |
| figura 42: Pantalla LCD en el estado de definición de variables del modelo simple | 41 |
| figura 43: ejemplo Código para escribir el texto constante del estado siguiente | 41 |
| figura 44: Código para la actualización de las variables de estado | 41 |
| figura 45: sistema de transmisión | 44 |
| figura 46: sistema de apoyo | 44 |
| figura 47: Puente móvil | 45 |
| figura 48: estructura de la base de la maquina | 45 |
| figura 49: Ejemplo de datos técnicos insuficiente para un diseño CAD | 48 |

[Escriba aquí]

[Escriba aquí]

| | |
|--|----|
| figura 51: Enlace de descarga del archivo CAD de la pieza | 49 |
| figura 50: Pieza CAD en el contexto del resto de piezas del ensayo | 49 |
| figura 52: construcción de un modelo Freecad | 49 |
| figura 53: mayado de una figura Freecad..... | 50 |

[Escriba aquí]

[Escriba aquí]

1. Resumen

El ensayo de modelo análogo geológico es ensayo muy útil en la predicción de eventos geológicos que no son fáciles de observar por su gran magnitud o larga duración. Consiste en introducir un modelo semejante a lo que se quiere estudiar en una caja con una o más caras móviles, después se procede a desplazar las caras para deformar el modelo. De esta manera se puede ver cómo reaccionaría el objeto de estudio, pero para poder conseguir una predicción fiable se debe conocer con precisión las variables del experimento como la velocidad o la distancia.

Este proyecto busca solucionar la petición del departamento de Geología de fabricar una maquina capaz de realizar un experimento de modelo análogo, controlada por un sistema electromecánico y debe tener unas dimensiones parecidas a las de un prototipo que fue dado al alumno en formato CAD. Además, este proyecto busca ser accesible desde cualquier parte del mundo utilizando plataformas open source y replicable con software y hardware libre.

Este trabajo se estructura para conseguir 4 objetivos principales: conseguir la accesibilidad global del proyecto; diseñar un apartado electrónico eficiente; diseñar un apartado mecánico resistente y funcional y presupuestar el proyecto.

Para conseguir la **accesibilidad global**, el alumno deberá ser autodidacta y familiarizarse con plataformas de open source como GitHub, para lo que necesitará aprender un nuevo lenguaje de programación; y facilitar a través de esta plataforma instrucciones suficientes para poder replicar el proyecto siguiendo la filosofía “do it yourself” (DIY).

Para el diseño de la **parte electrónica** del proyecto, alumno realizará un circuito eléctrico basado en Arduino y un programa de estados capaz de pedir variables al usuario a través de una sencilla interfaz y, en función de estas variables y de los datos obtenidos por los sensores, realizar el experimento y reiniciarse al acabar sin ayuda humana. De esta manera, el alumno deberá familiarizarse con el uso de hojas de datos técnicos, diseñar circuitos basados en hardware libre relacionado con Arduino y aplicar los conocimientos obtenidos en el grado en las asignaturas de la rama de electrónica y programación

El trabajo relacionado con la **parte mecánica** del proyecto consistirá en analizar el prototipo dado, corregir cualquier parte que considere importante cambiar, diseñar las piezas que falten para poder imprimirlas en 3D y realizar planos e indicaciones suficientes para la comprensión y montaje del proyecto. Para ello el alumno deberá aprender a utilizar programas CAD (freeCAD) y de impresión 3D (ULTIMAKER CURA en este caso) y aplicar sus conocimientos obtenidos en las asignaturas del grado de la rama de mecánica y dibujo técnico.

[Escriba aquí]

[Escriba aquí]

Para realizar el **presupuesto del proyecto**, el alumno debe realizar mediciones para saber cuántas unidades necesita de cada material, investigar proveedores y calcular el coste material que tiene el proyecto, para lo que deberá el alumno utilizar sus capacidades de búsqueda y sus conocimientos obtenidos en el grado en la asignatura de proyectos de ingeniería.

2. Introducción

2.1. Ensayo de modelo análogo

Un modelo es una forma de representar la realidad, se pueden distinguir 3 tipos: modelos icónicos, análogos, y simbólicos

Los modelos icónicos son aquellos que representan un evento o elemento real a través de propiedades morfológicas, cambiando la escala, pero conservando las propiedades topológicas.

Un ejemplo de estos modelos es una maqueta donde se ha aplicado un factor de escala, este tipo de modelado tienen unas implicaciones menos obvias que pueden ser importantes en algunos casos, como que la relación con la rugosidad no se conserve porque depende de la escala, y esto puede afectar a los resultados.

Los modelos análogos son aquellos que poseen algunas propiedades similares al objeto real sin ser una regla morfológica del mismo. Normalmente se utilizan ciertas convenciones que codifican las propiedades del objeto para su lectura.

El modelo análogo o analógico utiliza los principios de escalado para reproducir una situación real que debido a su tamaño y/o duración no es posible observar fácilmente.

Un ejemplo de modelo análogo son los mapas impresos donde se consigue un resultado distinto al objeto real pero que facilita la lectura de algunas propiedades en concreto.

Los modelos simbólicos se construyen mediante reglas más abstractas, representando en muchos casos el objeto real de forma matemática.

[Escriba aquí]

[Escriba aquí]

Un ejemplo es la representación matemática de un edificio permite aplicar algoritmos para calcular esfuerzos.

Los procesos geológicos son especialmente duraderos y grandes. Por eso, los ensayos en modelos análogos son muy utilizados en esta rama de la ciencia. Un ejemplo es el proceso de orogenia que puede durar millones de años y afectar a continentes enteros, pero con este ensayo se puede representar en unas pocas horas con un modelo de un par de metros cúbicos o menos.

Este ensayo consiste en colocar unos materiales con propiedades parecidas a los que se quiere emular (el modelo análogo) en un entorno controlado y aplicar fuerzas y desplazamientos regulados para observar los efectos que producen. De esta manera, y conociendo las ecuaciones que relacionan el modelo análogo con el elemento real, se pueden predecir con precisión eventos de grandes magnitudes.

La maquina para realizar este experimento suele estar compuesta por una caja con 1 o 2 paredes móviles y en algunos casos, 1 o 2 caras transparentes para poder ver el interior de la caja (aunque puede que no tenga caras laterales o que las tenga opacas), en el interior de esta caja se depositan los materiales del experimento, en ocasiones formando capas para representar la interacción entre diferentes materiales y se desplazan las caras móviles controlando la velocidad y duración del experimento para observar los cambios.

El ensayo ha evolucionado desde cuando en 1812 creado por Marion Hall

La caja de modelado es el modelo más antiguo y básico, se acciona a mano y esto hace que se encuentre en muchos centros educativos actuales para que se vean los efectos de forma cualitativa y barata pero no es muy práctico porque no es sencillo controlar la velocidad y el desplazamiento de forma precisa y por tanto no se pueden dar predicciones precisas.

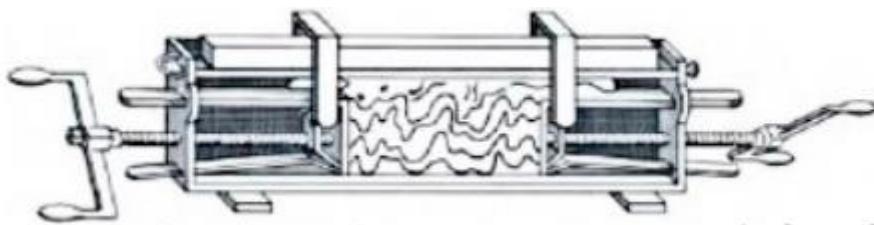


Figura 1: caja de modelado

[Escriba aquí]

[Escriba aquí]

El sistema de modelado es una actualización de la caja de modelado a las nuevas tecnologías de computación y mecánica, consiguiendo mucha más precisión y trazabilidad al realizar el experimento mediante un sistema electromecánico, consiguiendo velocidades del orden de micras por segundo constantes (nota: $1\mu\text{m}/\text{s} = 3,6 \text{ mm}/\text{h}$).

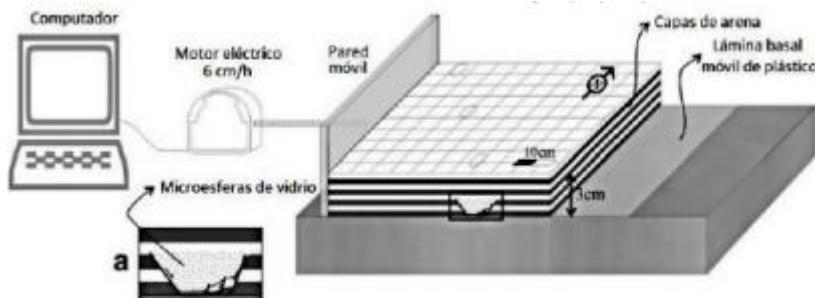


Figura 2 sistema de modelado

2.2. Hardware libre, software libre y la filosofía DIY

El concepto de hardware y software libre es el conjunto de programas y piezas de electrónica o máquinas que son de libre acceso a la población, tanto la pieza en si como sus diagramas y datos técnicos, ya sea de forma gratuita o mediante algún tipo de pago haciendo mucho más accesible la electrónica fuera de las empresas.

Un ejemplo de hardware y software libre son los dispositivos y programas de la familia Arduino, los cuales tienen sus esquemas y datos técnicos disponibles en internet, es descargable libremente su editor de código (que también permite fácilmente cargar los códigos en los dispositivos Arduino) y subir y descargar paquetes de programas compatibles con estos dispositivos.

Debido a que en los últimos años el libre intercambio de diseños y las impresoras 3D se han normalizado, ha nacido la corriente *Do It Yourself* (DIY). Esta corriente incita a sus usuarios a no comprar diseños terminados, sino a aprender y compartir el desarrollo tecnológico y construir ellos mismos sus diseños. Esto permite que afloren muchos equipos de desarrollo científico sin un alto presupuesto.

En particular este TFG utiliza estos conceptos para ser accesible, con bajo coste, a nivel internacional, consiguiendo que cualquier universidad, instituto o equipo de personas pueda acceder a esta tecnología.

[Escriba aquí]

[Escriba aquí]

2.3. GitHub

GitHub es un portal “open source” de desarrollo colaborativo con control de versiones donde cualquiera puede abrir una cuenta y subir diseños propios a repositorios de forma libre y gratuita.

Open Source es una expresión de la lengua inglesa que significa fuente abierta, se suele utilizar en el ámbito de la informática para referirse a programas que te dan su código de programación para que cualquiera lo descargue y edite.

GitHub permite el desarrollo colaborativo a través de repositorios de autor en los que se puede permitir que otros usuarios editen su contenido, facilitando así el trabajo en equipo, ya que mucha gente puede trabajar desde diferentes perspectivas en un mismo trabajo de forma sencilla, además, esta plataforma permite trabajar en el propio ordenador sin conexión en una copia del repositorio y avisa si otro usuario sube información para no trabajar con material desfasado.

Del mismo modo, también permite ver que ha cambiado en cada actualización y recuperar información de las versiones anteriores si fuese necesario, de manera que no es necesario conservar muchos programas por miedo a que al realizar cambios dejen de funcionar.

[Escriba aquí]

[Escriba aquí]

3. Objetivos

El proyecto busca crear una nueva máquina para realizar un ensayo de modelo análogo para el área de geología de la universidad Rey Juan Carlos (que actualmente carece de un método para realizar esta práctica) a partir de un diseño CAD donde se especifican las dimensiones que debe tener.

Durante el desarrollo de este proyecto se deben conseguir los siguientes resultados:

- El diseño debe ser **funcional** que permita la realización del experimento y la fácil medición de la distancia y velocidad a la que se ha realizado el experimento.
- El diseño debe ser **interactivo**, permitiendo cambios en los parámetros del experimento por el usuario.
- El diseño debe ser **intuitivo** y fácilmente utilizable, permitiendo introducir las variables y entender los resultados sin grandes complicaciones.
- El diseño debe ser **resistente**. Durante la ejecución del experimento ninguna pieza debe sufrir daños irreversibles (deformaciones plásticas de los componentes mecánicos, quema de los componentes electrónicos, etc....).
- El diseño debe ser **seguro y didáctico**. Dado que se utilizará para la enseñanza debe poder pararse fácilmente durante el funcionamiento del motor para dar una explicación o prevenir un riesgo y después poder continuar el experimento o terminarlo y que vuelva a su posición de inicio.
- Se deben calcular los límites del diseño como la precisión de las variables del ensayo o la fuerza máxima que puede ejercer.
- Este proyecto debe ser autoexplicativo y accesible a través de GitHub. Los contenidos en el repositorio de GitHub deben ser suficientes para que cualquier persona pueda adquirir todo el material necesario y construir el diseño sin tener conocimientos especiales.
- El diseño se debe utilizar los materiales disponibles en el departamento de electrónica, siempre que sea posible, para reducir al máximo el coste de compra de piezas para la universidad.

[Escriba aquí]

[Escriba aquí]

4. Solución técnica

Este proyecto consigue diseñar y construir una máquina para realizar un ensayo de modelo análogo con 1 cara móvil tras analizar el prototipo, corregir los fallos que se han encontrado en él e incluido las piezas impresas y un circuito eléctrico programable para la automatización del experimento

El diseño final puede ejercer una fuerza con su puente móvil de 2460 N (unos 246 kg) en movimiento si consideramos un factor de trabajo del 1 y despreciamos las perdidas por rozamiento, aunque para conseguir un movimiento fluido se recomienda no superar el 80% de su capacidad de carga (aproximadamente 2000 N), el sistema puede conseguir una precisión de 3,2 micras, pero el programa solo consigue una precisión de 15 micras y velocidades regulables entre las 11 micras/s hasta las 117 micras /s.

Entre el diseño final y el prototipo dado se cambia la longitud de los ejes de acero de los lados de 1100mm a solo 500 mm, porque no es necesaria tanta longitud porque solo se requiere que el puente móvil pueda desplazarse 400mm,

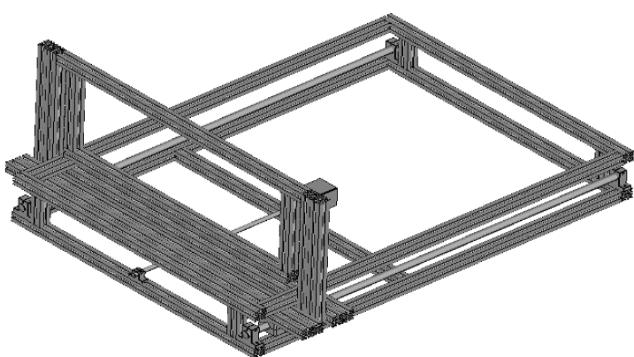


Figura 4: modelo CAD del prototipo

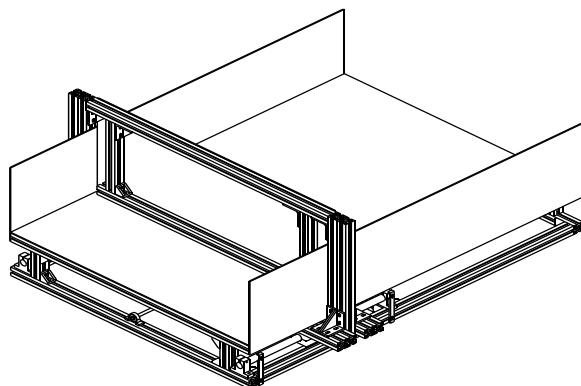


Figura 3: modelo CAD del diseño del ensayo de modelo análogo geológico simple

Por otro lado, también se han incluido las piezas impresas en 3D que se han considerado necesarias, por regla general se han utilizado para unir 2 o más piezas que de otra manera sería difícil unir

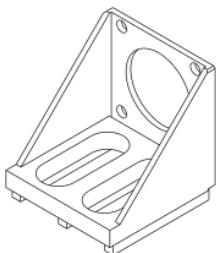


Figura 5: modelo CAD del soporte del motor

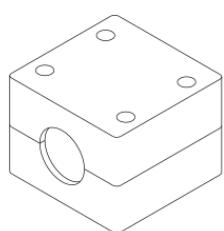


Figura 6: modelo CAD del soporte del cojinete

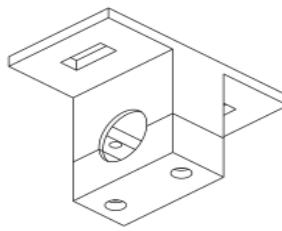


Figura 7: modelo CAD del soporte de la tuerca del husillo



Figura 8: modelo CAD del soporte del fin de carrera

[Escriba aquí]

[Escriba aquí]

El usuario puede controlar este sistema desde una pantalla LCD con un encoder y un buzzer, solo puede interactuar girando el encoder en sentido horario o antihorario o presionándolo, y a modo de feedback para que el usuario sienta que la pantalla funciona, cada vez que pulse el encoder sonará el buzzer.



Figura 9: pantalla LCD con encoder y buzzer

El programa que rige el experimento se divide en 6 estados:

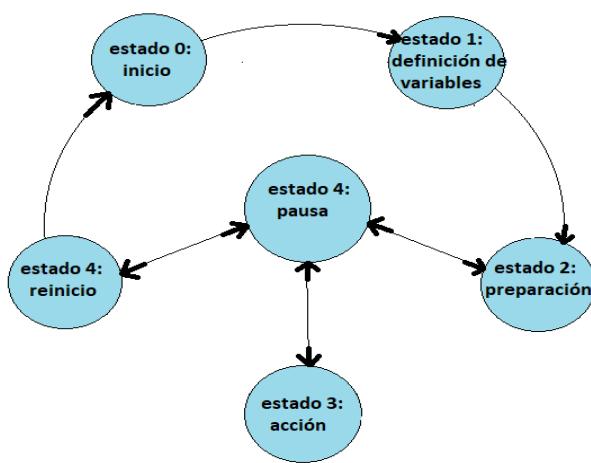


Figura 10: Diagrama de estados del programa del ensayo

Estado 0 inicio: Sirve de portada para esperar hasta que alguien quiera empezar el experimento y para reiniciar todas las variables para que no se arrastre información basura de un experimento al siguiente. Para salir de este estado e ir al estado 1 solo hay que pulsar el encoder.

[Escriba aquí]

[Escriba aquí]



Figura 11: pantalla LCD en el estado 0

Estado 1 Definición de variables: En esta etapa el usuario tiene la oportunidad de escoger 3 variables del experimento: “distancia inicial”, que regula hasta donde se mueve el puente móvil antes de empezar el experimento; “distancia final”, que regula el desplazamiento máximo durante el experimento y “velocidad”, que regula a que velocidad se mueve el puente móvil durante el experimento.

Para moverse por la pantalla el usuario puede girar la ruleta del encoder, una vez haya llegado a la variable que desea puede pulsar el encoder y después al girar otra vez el encoder en vez de moverse entre variables empezará a sumar o restar valor a la variable deseada. En el caso de “distancia inicial” y “distancia final” si se pulsa 1 vez se sumará de 100 en 100, si se pulsan 2 se sumarán de 10 en 10 y si se pulsan 3 de 1 en 1 (el incremento mínimo en esta variable es 1 mm). En el caso de la velocidad tiene valores limitados, y cuando giramos avanzamos entre ellos, y al volver a pulsar volvemos a la selección de variables.

Cuando se han editado todas las variables podemos clicar en “iniciar experimento” para avanzar al siguiente estado



Figura 12: pantalla LCD en el estado 1

Estado 2 Preparación: En este estado la maquina desplaza el puente móvil tan rápido como puede hasta la distancia inicial, supuestamente en esta etapa aun no se debe colocar el modelo analógico en el sistema de modelado porque esta etapa aun no forma parte del ensayo.

Si surge cualquier problema se puede detener el motor pulsando el encoder, e iremos al estado de pausa, desde ahí se puede elegir entre continuar con el experimento como estaba escogiendo “continuar” o reiniciarlo escogiendo “terminar”.

[Escriba aquí]

[Escriba aquí]

Si el puente móvil alcanza su posición para empezar el experimento, el programa se ira a pausa para dar tiempo a montar el modelo análogo, y se reanudara cuando alguien seleccione “continuar”



Figura 13: pantalla LCD en el estado 2

Estado 3 Acción: En este estado la maquina se mueve a la velocidad indicada hasta la posición final, durante este proceso se presenta por pantalla la distancia recorrida respecto del objetivo final.

Si surge cualquier problema se puede detener el motor pulsando el encoder, e iremos al estado de pausa, desde ahí se puede elegir entre continuar con el experimento como estaba escogiendo “continuar” o reiniciarlo escogiendo “terminar”.

Si el puente móvil alcanza su posición final o se activa el fin de carrera del final del recorrido, el programa se ira a pausa para dar tiempo tomar las notas que sean necesarias, después cuando se quiera terminar solo se deberá pulsar en “terminar”



Figura 14: pantalla LCD en estado 3

Estado 4 Pausa: Este es el estado por el que mas veces se pasa durante el experimento y que más cambia porque su interfaz cambia sutilmente en función de donde venga y si ese estado ha terminado o no; caso afirmativo si se selecciona continuar pasamos al estado siguiente y en caso negativo volvemos al estado anterior, en caso de seleccionar “terminar” siempre se va al estado 5 (reinicio).

+++++
+++++

[Escriba aquí]

[Escriba aquí]



Figura 15: pantalla LCD en estado 4

Estado 5 Reinicio: Este estado hace funcionar los motores al revés, haciendo que el puente móvil llegue a su posición inicial, solo se para si alguien pulsa el encoder para ir a pausa (por si hubiese algún peligro) o si se llega a la posición de inicio, en cuyo caso volvemos al estado 0 (inicio)



Figura 16:pantalla LCD en estado 5

[Escriba aquí]

[Escriba aquí]

A modo de extra el alumno ofrece una propuesta de mejora capaz de controlar dos puentes móviles a la vez para conseguir mayor variedad en el experimento.

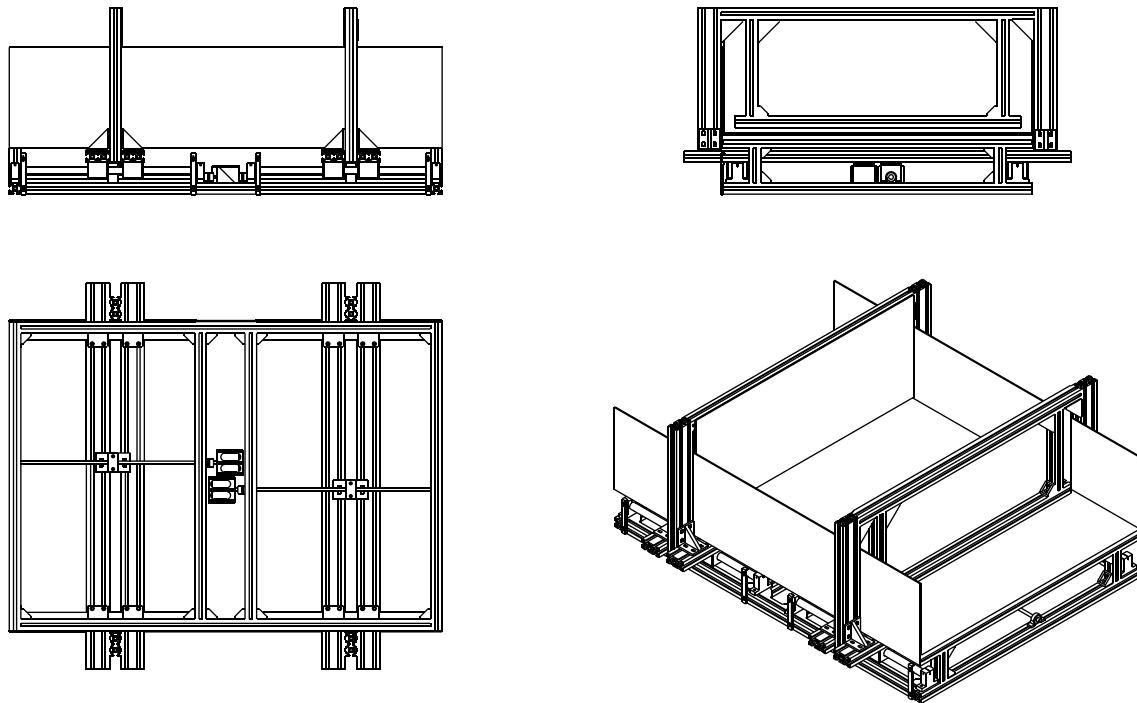


Figura 17: modelo CAD del diseño del ensayo análogo geológico doble

Este diseño no presenta cambios técnicos importantes en el aspecto mecánico, solo instala un segundo puente móvil con todos los sistemas necesarios asociados, haciendo que se pueda montar sobre el primero sin tener que empezar un diseño nuevo de 0 si se decide montar este más adelante.

Sin embargo, obliga a desplazar el sistema de transmisión de fuerza lateralmente 35mm, haciendo que no empuje el puente móvil en su punto medio sino desviado, con las implicaciones que eso tiene para las tensiones.

[Escriba aquí]

[Escriba aquí]

Sin embargo, los cambios más significativos están en el programa de Arduino, que debe adaptarse a trabajar con 2 motores nombrados X e Y.

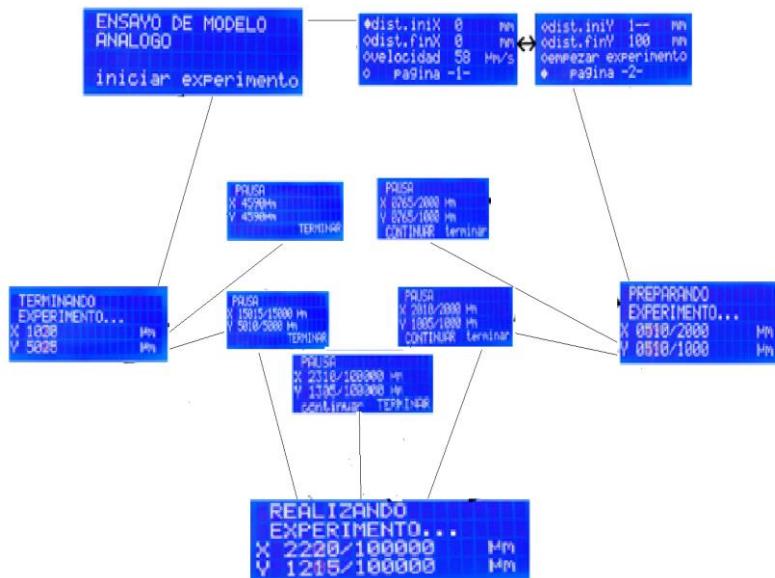


figura 18: Diagrama de pantallas de estados en el programa de ensayo doble

para afrontar este cambio el **estado 1: Definición de variables** debe crear diferentes páginas para hacer cómoda la navegación (porque las variables prácticamente se duplican) e implementar un método de avanzar y retroceder entre ellas sin perder información en el proceso.

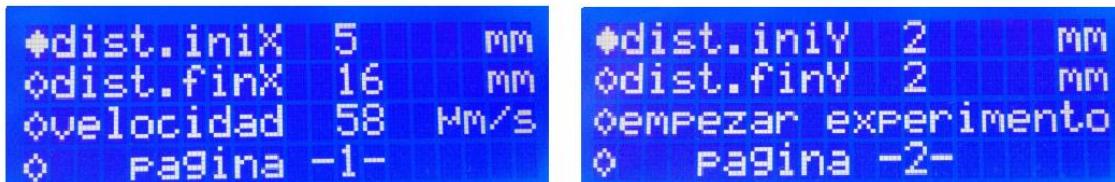


figura 19: Páginas 1 y 2 del estado de definición de variables

En los **estados donde los motores trabajan**, se indica la posición de estos durante el experimento; se ha considerado que no siempre se quieren utilizar los dos motores, de esta manera solo aparecen en pantalla el desplazamiento de los puentes móviles que se estén moviendo o se hayan movido durante el experimento.

Por ejemplo, si se quieren mover ambos motores en la fase de preparación en ese momento aparecerá por pantalla la posición de ambos puentes móviles, pero si al llegar al estado 3 Acción, donde tiene lugar el experimento en sí, solo se quiere utilizar el puente X solo aparecerá este por pantalla.

[Escriba aquí]

[Escriba aquí]

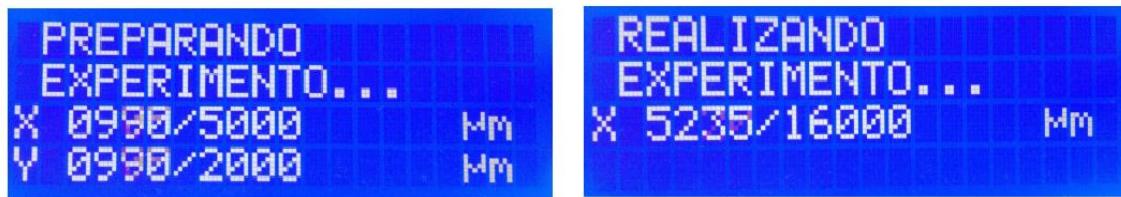


figura 20: Pantalla LCD durante el movimiento de motores

El estado de pausa ahora debe presentar 2 variables, y al poder escribir solo 4 líneas eso obliga a optimizar como se presenta las opciones de continuar y terminar porque no se puede mantener el diseño anterior de selección en vertical (que necesita 2 filas para escoger) con un indicador en forma de “o” llena o vacía.

Se aprovecho esta oportunidad para implementar un diseño más limpio donde se sabe cuál variable se ha escogido porque está en mayúsculas y al cambiar de variable, la no seleccionada, cambia automáticamente a minúsculas

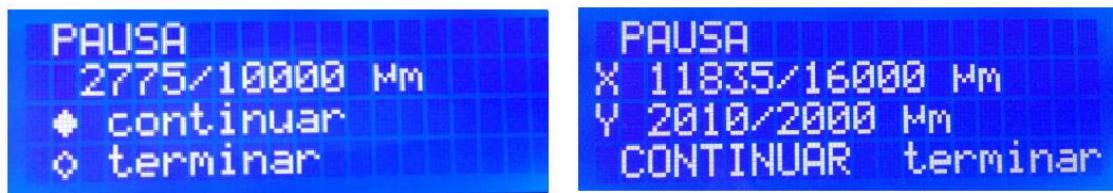


figura 21: comparativa entre la pantalla de pausa del diseño simple y el dobl

5. Uso de los archivos de este proyecto

En este apartado se analiza como se han subido los archivos de este TFG a internet, donde encontrar los archivos necesarios para llevar a cabo el montaje del diseño y que se debe hacer una vez se tengan esos archivos.

El TFG se encuentra en internet en GitHub, un portal de libre acceso para que creadores de todo el mundo intercambien libremente diseños. Para poder utilizarlo es necesario crear una cuenta en GitHub, una vez se tenga podrás crear tus propios repositorios y utilizar los de los demás

Todos los documentos se encuentran accesibles en el repositorio “tfg” del usuario “zeus97roman” (<https://github.com/zeus97roman/tfg>), clasificados en 4 carpetas “archivosparaimprimir3D”, “fcad”, “memoria y anexos” y “programas Arduino”

[Escriba aquí]

[Escriba aquí]

| Branch: master | | New pull request | Find file | Clone or download |
|---|--|------------------|--------------------------------------|-------------------|
|  zeus97roman | En esta actualizacion se han reorganizado las carpetas para facilitar... | ... | Latest commit 6d35fec 11 minutes ago | |
|  archivosparsaimprimir3D | En esta actualizacion se han reorganizado las carpetas para facilitar... | | 11 minutes ago | |
|  fcad | En esta actualizacion se han reorganizado las carpetas para facilitar... | | 11 minutes ago | |
|  memoria y anexos | En esta actualizacion se han reorganizado las carpetas para facilitar... | | 11 minutes ago | |
|  programas arduino | En esta actualizacion se han reorganizado las carpetas para facilitar... | | 11 minutes ago | |
|  .gitignore | añadido el .gitignore y borrados archivos temporales | | 7 months ago | |
|  readme.md | Anado el fichero readme | | 8 months ago | |
|  ~\$MORIATFG.docx | En esta actualizacion se han reorganizado las carpetas para facilitar... | | 11 minutes ago | |

En la carpeta “**archivosparsaimprimir3D**” se encuentran los archivos para utilizar en la impresora de forma directa.

Para imprimir estas piezas con el programa, de software libre, Ultimaker Cura únicamente hay que abrir el programa y clicar en la esquina superior izquierda en la opción “file” y después a “open file”, finalmente solo hay que seleccionar el archivo que queremos abrir y nos aparecerá por pantalla una representación de cómo se vería la pieza en la impresora.

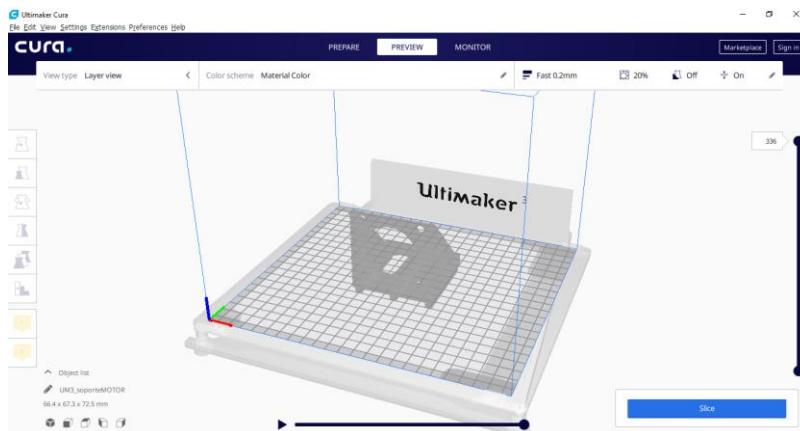


figura 22: Interfaz del programa Ultimaker Cura

Se recomienda buscar que la posición final que facilite la impresión teniendo en cuenta que esta se construirá en capas paralelas al suelo, de no ser así, se puede reorientar utilizando los comandos que nos aparecerán a la izquierda.

Un ejemplo de posición poco práctica para su impresión se da en la ilustración de la izquierda, porque no es una posición estable y obliga a crear un soporte para el voladizo muy costoso o arrriesgarse a que se desmorone. Mientras en la posición de la derecha la pieza no tiene que crear estructuras de apoyo porque no tiene voladizos muy pronunciados.

[Escriba aquí]

[Escriba aquí]

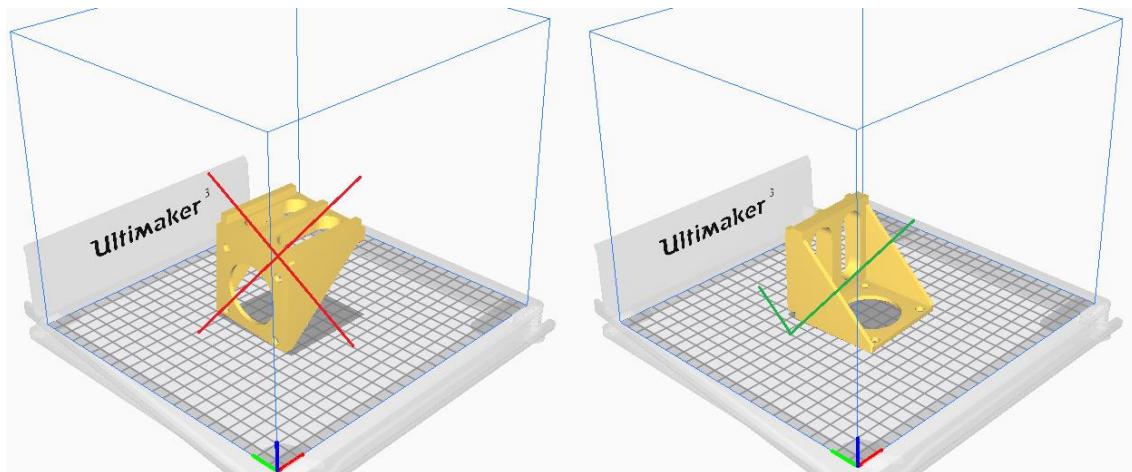


figura 23: Ilustración de como girar una figura en Ultimaker Cura

Utilizando la barra blanca que esta sobre la ilustración definiremos el grosor del trazo, el espesor de la pieza y si queremos o no que la impresora cree un soporte para la figura. En este caso hemos seleccionado para el soporte del motor un trazo rápido de 0,2mm, un 20% de densidad en la pieza y con apoyo.

Después solo se debe clicar en “slice” para que el programa diseñe los pasos necesarios para imprimir la pieza y estime la cantidad de hilo de plástico y el tiempo necesario para realizar esta impresión. En este caso utilizará 5,44 m de hilo de plástico y tardaría 3 horas y 48 min (aunque después se recomienda dejar un tiempo enfriar la pieza para poder sacarla sin dificultad y sin romperla).

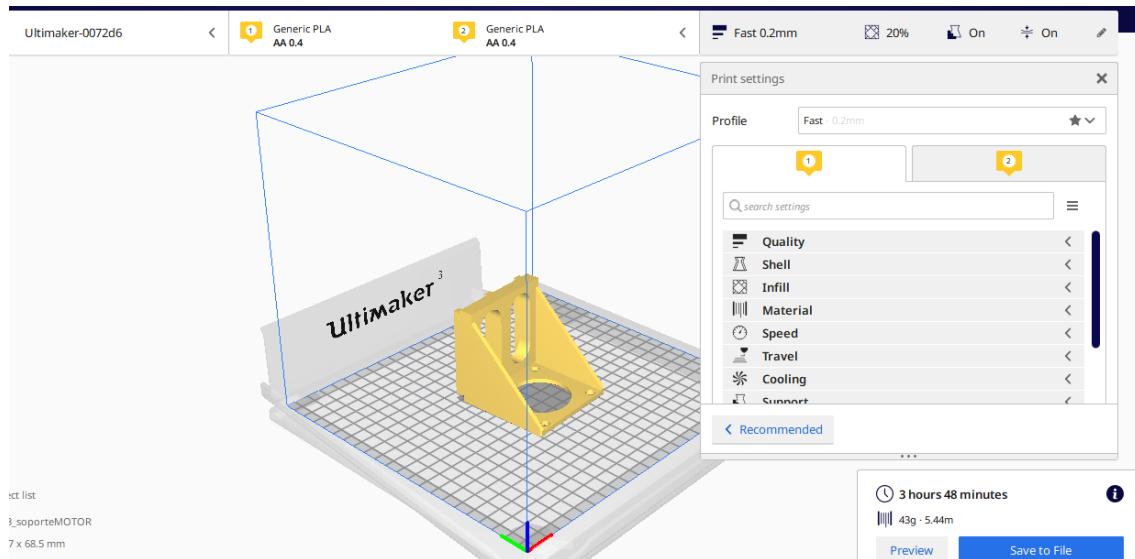


figura 24: modelo listo para imprimir

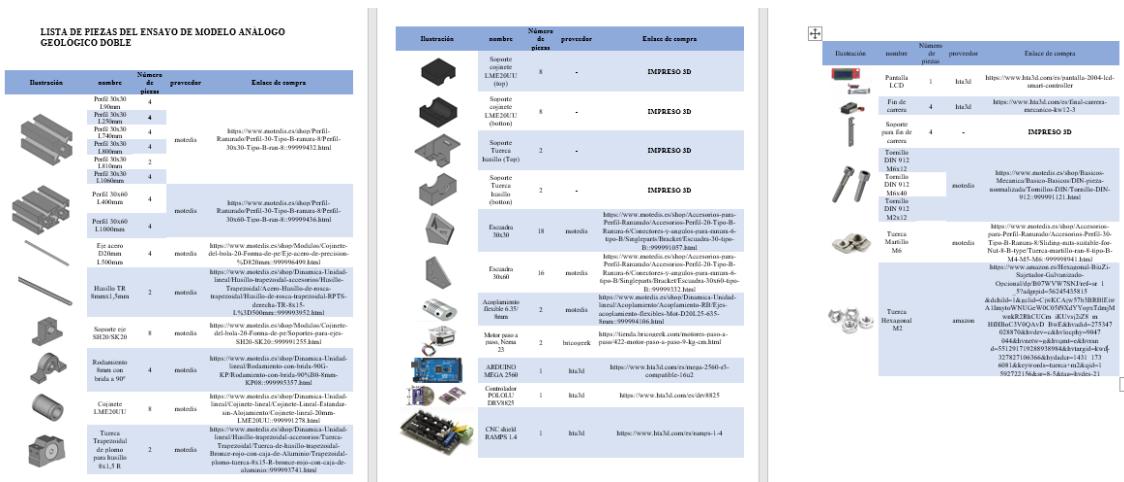
[Escriba aquí]

[Escriba aquí]

En la carpeta de “**fcad**” hay modelos CAD que pueden servir como apoyo en caso de tener cualquier complicación en entender el diseño, desde los diseños de los modelos enteros que se trabajan en este proyecto hasta material complementario a los planos (por ejemplo, diseños detallados de las uniones roscadas).

En la carpeta “**memoria y anexos**” se encuentran las instrucciones esenciales para poder conseguir los materiales necesarios y montar el ensayo.

Entre los documentos que se pueden encontrar aquí tenemos la lista de piezas necesarias para el ensayo (tanto para el ensayo simple como para el ensayo doble)



| LISTA DE PIEZAS DEL ENSAYO DE MODELO ANALOGO GEOLÓGICO DOBLE | | | | |
|---|---------------------|------------------------|---|-------------------|
| Ilustración | número de piezas | Número de proveedor | Efecto de compra | |
| Perfil 30x30 L 100mm | 4 | | | |
| Perfil 30x30 L 100mm | 4 | | | |
| Perfil 30x30 L 100mm | 4 | | | |
| Perfil 30x30 L 100mm | 4 | motola | https://www.motola.es/shop/Perfil-Ramado-Perfil-30-Tipo-B-estanca/Perfil-30x30-Tipo-B-estanca-30-999943.html | |
| Perfil 30x30 L 100mm | 4 | motola | https://www.motola.es/shop/Perfil-Ramado-Perfil-30-Tipo-B-estanca/Perfil-30x30-Tipo-B-estanca-30-999943.html | |
| Eje acero D25mm L 100mm | 4 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Eje-Acero-D25mm-L100mm | |
| Haste TR Rama 1,5mm | 2 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Haste-TR-Rama-1-5mm | |
| Sensor de SH90-SK20 | 8 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Sensor-de-SH90-SK20 | |
| Resistencias Rama con brida a 90° | 4 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Resistencia-Rama-con-brida-a-90 | |
| Conejito LME20/U | 8 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Conejito-LME20/U | |
| Tuerca Trapezoidal de aluminio para tornillo 8x1,5 mm | 2 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Tuerca-Trapezoidal-de-aluminio-para-tornillo-8x1-5-mm | |
| IMPRESO 3D | | | | |
| Sensor estacion LME20/U (top) | 8 | - | | IMPRESO 3D |
| Sensor estacion LME20/U (bottom) | 8 | - | | IMPRESO 3D |
| Sensor Tuerca lateral (top) | 2 | - | | IMPRESO 3D |
| Sensor Tuerca lateral (bottom) | 2 | - | | IMPRESO 3D |
| FÍSICO | | | | |
| Pantalla LCD | 1 | Invi | https://www.invi.es/pantalla-2004-lcd-smart-controller | FÍSICO |
| Fín de carrera | 4 | Invi | https://www.invi.es/final-carrera-mecanico-kw12-3 | FÍSICO |
| Sensor para fin de carrera | 4 | - | | FÍSICO |
| Tornillo DIN 912 M6x12 | 1 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Tornillo-DIN-912-M6x12 | |
| Tornillo DIN 912 M6x14 | 1 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Tornillo-DIN-912-M6x14 | |
| Tornillo DIN 912 M6x12 | 1 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Tornillo-DIN-912-M6x12 | |
| Tornillo Mártill M6 | 1 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Tornillo-Mártill-M6 | |
| ACCESORIOS | | | | |
| Base para el sensor | 1 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Base-para-el-sensor | ACCESORIOS |
| IMPRESO 3D | | | | |
| Escuadra M6x10 | 18 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Escuadra-M6x10 | IMPRESO 3D |
| Escuadra M6x10 | 16 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Escuadra-M6x10 | IMPRESO 3D |
| Ajustamiento final de 6/35 mm | 2 | motola | https://www.motola.es/shop/Modulos-Componentes-de-accionamiento/Ajustamiento-final-de-6-35-mm | IMPRESO 3D |
| Motor para pasos Nema 23 | 2 | motocraft | https://www.motocraft.com/motor-pasos-nema-23-42-mm-paso-a-paso-9-kg-cm.html | IMPRESO 3D |
| ARCDUINO Mega 2560 | 1 | Invi | https://www.invi.es/arduino-mega-2560-v3-compatible-f6a2 | IMPRESO 3D |
| Controlador POLolu DRV8825 | 1 | Invi | https://www.invi.es/drive8825 | IMPRESO 3D |
| CNC shield RAMPS 1.4 | 1 | Invi | https://www.invi.es/ramps-1-4 | IMPRESO 3D |

figura 25: lista de piezas necesarias para construir el ensayo

También se han realizado unas instrucciones sencillas donde resume el montaje, los cuales vienen ilustrados de manera que se pueda entender sin conocimientos técnicos. Cada paso indica el tiempo que se estima que dure, las piezas que se necesitas y las herramientas utilizadas.

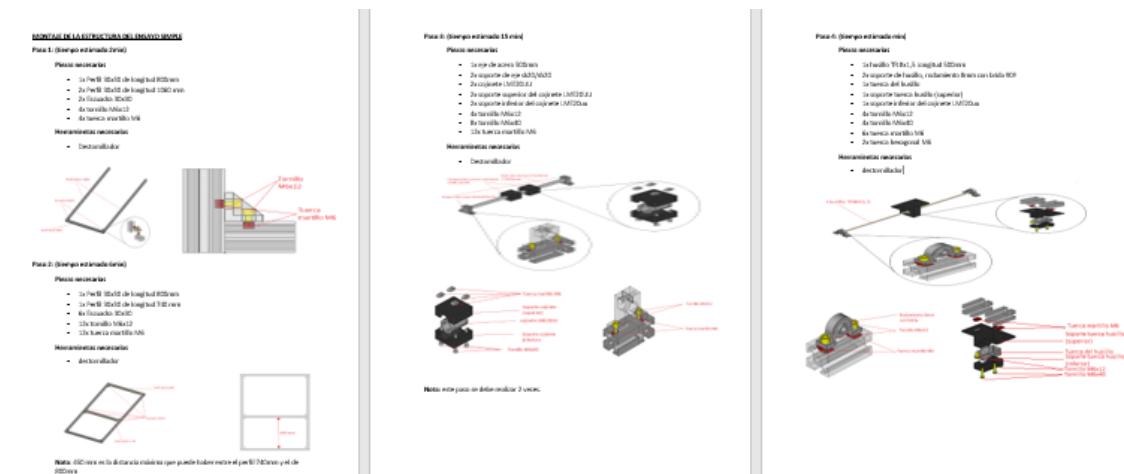


figura 26: Etapas del montaje

[Escriba aquí]

[Escriba aquí]

En esta carpeta también se pueden encontrar los planos en formato A3, los cuales son más técnicos y precisos que las instrucciones de montaje, y permiten ver en detalle y de forma técnica muchos aspectos del diseño.

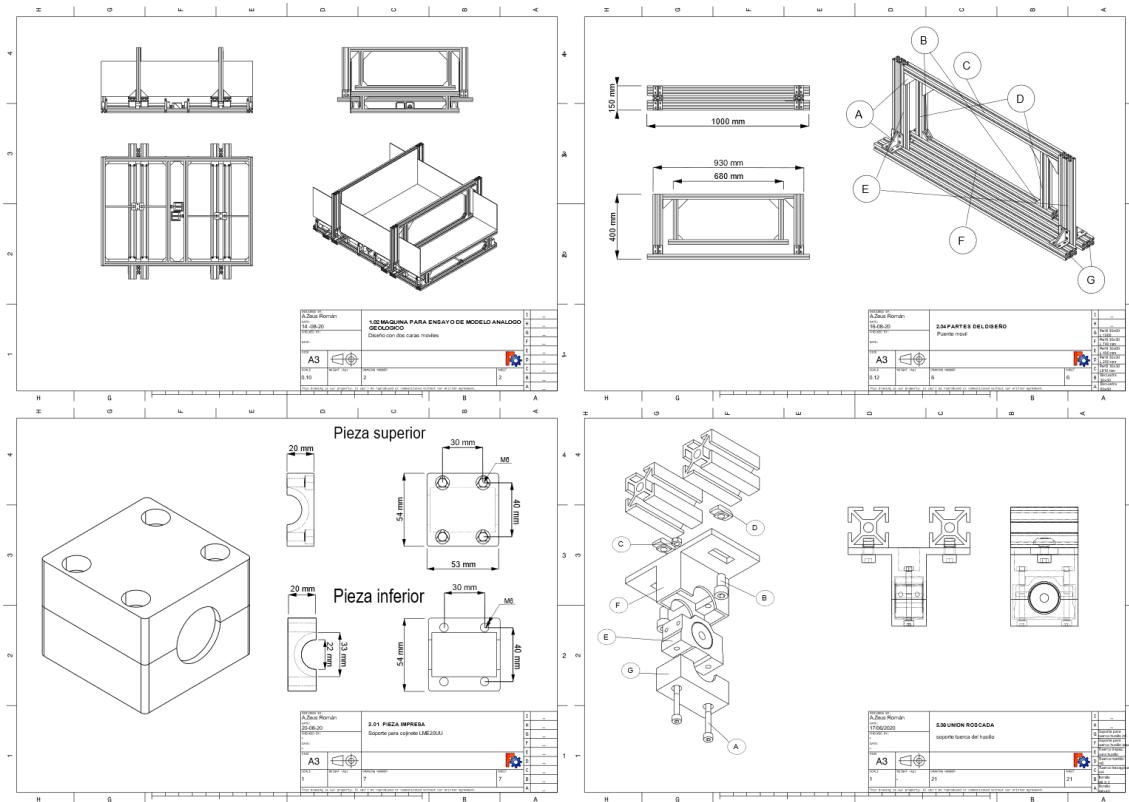


figura 27: Ejemplo de planos de este proyecto

Finalmente, en la carpeta “**programas Arduino**” hay los dos archivos con los programas para realizar el ensayo.

Para poder usarlos, el usuario debe descargarse el archivo que quiera utilizar y abrirlo con el editor de Arduino, después solo hay que conectar el Arduino al ordenador y pulsar a “subir” y el programa se instalará en el dispositivo.

[Escriba aquí]

[Escriba aquí]

The screenshot shows the Arduino IDE interface. The title bar reads "ProgramaEnsayoMAdoble Arduino 1.8.13 (Windows Store 1.8.39.0)". The menu bar includes "Archivo", "Editar", "Programa", "Herramientas", and "Ayuda". Below the menu is a toolbar with icons for save, undo, redo, open, upload, and download. The main window has a teal header bar with the text "ProgramaEnsayoMAdoble". The code editor contains the following C++ code:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(16,17,23,25,27,29);
//pins de la shield
#define BEEPER 37 // Beeper conectado a GADGETS3D shield MEGA_18BEEPER
#define LCD_PINS_RS 16 // LCD control conectado a GADGETS3D shield LCDRS
#define LCD_PINS_ENABLE 17 // LCD enable pin conectado a GADGETS3D shield LCDE
#define LCD_PINS_D4 23 // LCD signal pin, conectado a GADGETS3D shield LCD4
#define LCD_PINS_D5 25 // LCD signal pin, conectado a GADGETS3D shield LCD5
#define LCD_PINS_D6 27 // LCD signal pin, conectado a GADGETS3D shield LCD6
#define LCD_PINS_D7 29 // LCD signal pin, conectado a GADGETS3D shield LCD7
#define BTN_EN1 31 // Encoder, conectado a GADGETS3D shield S_E1
#define BTN_EN2 33 // Encoder, conectado a GADGETS3D shield S_E2
#define BTN_ENC 35 // Encoder Click, connected to Gadgets3D shield S_EC
#define X_STEP_PIN 54 // PIN de pulsos para avanzar el motor
#define X_DIR_PIN 55 // PIN para indicar la dirección en la que debe avanzar
#define X_MIN_PIN 3 // PIN para el fin de carrera colocado al inicio del recorrido
#define X_MAX_PIN 2 // PIN para el fin de carrera colocado al final del recorrido
#define Y_STEP_PIN 60
#define Y_DIR_PIN 61
#define Y_ENABLE_PIN 56
#define Y_MIN_PIN 14
#define Y_MAX_PIN 15
// signos
```

figura 28: ilustración de como cargar un programa en el Arduino

6. Análisis y montaje electrónico

6.1. Componentes del circuito eléctrico

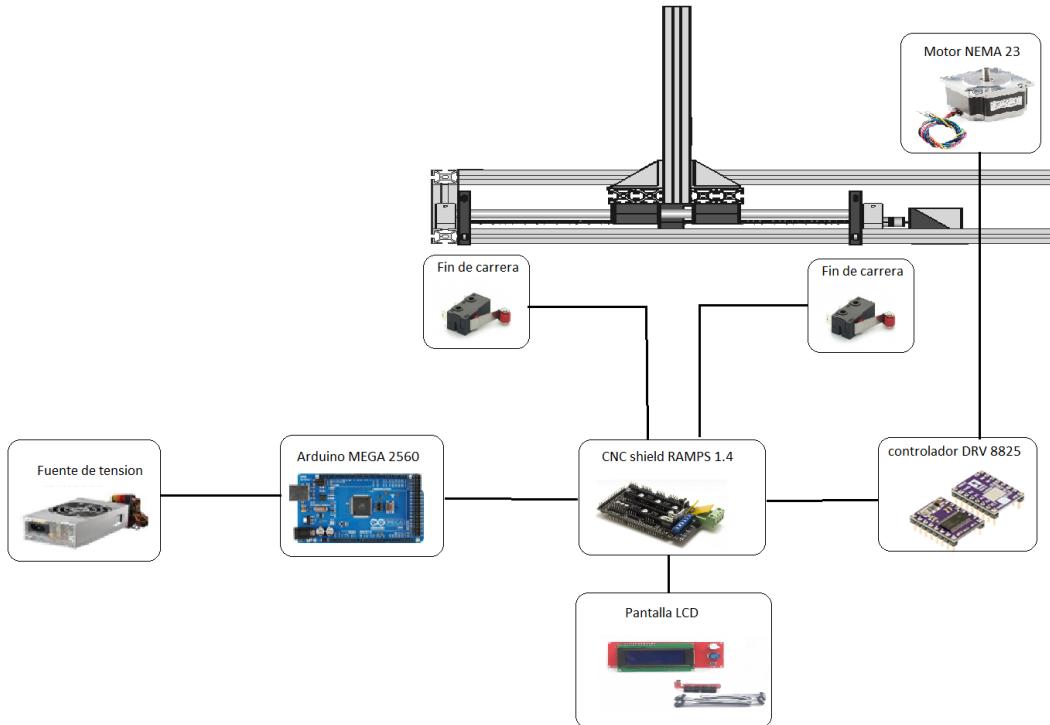


figura 29: Esquema del circuito eléctrico

6.1.1. ARDUINO MEGA 2560

Arduino es una plataforma de hardware y software abierto, es muy popular para diseños en la enseñanza o para proyectos a pequeña escala (en la industria no es muy usual) por su simplicidad de programación y fácil acceso.

Uno de los principales puntos a favor de la plataforma Arduino es que tiene muchos complementos electrónicos compatibles, por esto se ha seleccionado en este proyecto, de modo que en la mayoría de los casos no habrá que adaptar voltajes e intensidades para que dos elementos se comuniquen entre en el proyecto.

Arduino tiene varios modelos de placas con microcontrolador, en este proyecto se utiliza un Arduino MEGA 2560 por su potencia y numero de pines.

[Escriba aquí]

[Escriba aquí]

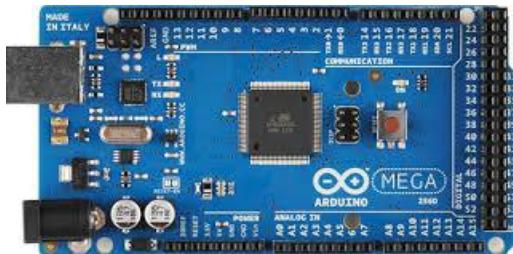


figura 30: Arduino MEGA 2560

Este modelo, en concreto, cuenta con estos datos técnicos:

- Microcontrolador: ATmega2560
- Clock speed: 16 MHz
- Voltaje operativo: 5V
- Voltaje de entrada: 7-12V
- Pines INPUT/OUTPUT: 54
- Pines analógicos INPUT: 16
- Corriente DC por cada pin INPUT/OUTPUT: 40mA

Esto significa que la fuente de tensión deberá alimentarlo a un voltaje de entre 7 y 12 V, y que el Arduino podrá enviar por sus puertos de salida 5V y 40mA de DC, con estos datos valoraremos si es necesario realizar ajustes al circuito.

CNC SHIELD

Es una placa que sirve para montar los drivers y simplificar el circuito para la comunicación entre el Arduino y los motores.

Este modelo, el RAMPS 1.4, puede ayudar a controlar con un solo Arduino hasta un máximo de 5 motores, por eso se suele usar en impresoras 3D y en este diseño facilitara mucho la ampliación a un modelo con dos puentes móviles.

[Escriba aquí]

[Escriba aquí]



figura 31: Correspondencia entre los pines de la CNC shield y los del Arduino MEGA 2560

Esta placa puede ser utilizada por muchos modelos de Arduino uniendo los pines de Arduino con el pin en cuestión para controlar los pasos o dirección de cada motor y para abastecer de energía a la placa.

El voltaje de trabajo del motor (12V) se introduce por unos pines especiales, para permitir abastecer los motores sin quemar los circuitos de la placa ni de los controladores.

Sobre la placa se deben montar tantos controladores de motores paso a paso como motores haya, en este caso utilizaremos el Pololu Driver DRV8825.

Bajo ellos se encuentran los jumpers de cada controlador, con ellos se puede regular cuantos pulsos hacen falta para avanzar un paso (con cada pulso avanza una fracción de paso, en el caso de nuestro motor cada paso entero es un giro de 1, 8º) de forma manual.

| MS0 | MS1 | MS2 | RESOLUCION |
|------|------|------|------------|
| bajo | bajo | bajo | 1 |
| alto | bajo | bajo | $1/2$ |
| bajo | alto | bajo | $1/4$ |
| alto | alto | bajo | $1/8$ |
| bajo | bajo | alto | $1/16$ |
| alto | bajo | alto | $1/32$ |
| bajo | alto | alto | $1/32$ |
| alto | alto | alto | $1/32$ |

NOTA: Siendo ALTO cuando se encuentran los jumpers conectados en las posiciones determinadas y BAJO en el caso contrario.

[Escriba aquí]

[Escriba aquí]

Como este modelo se puede montar directamente en el Arduino es muy importante conocer a que pines va conectados cada puerto de la placa.

Para controlar el primer motor:

- **Pin para el control de pulsos del motor** X_STEP_PIN 54
- **Pin para el control de la dirección del motor** X_DIR_PIN 55
- **Pin para el fin de carrera del punto mínimo** X_MIN_PIN 3
- **Pin para el fin de carrera del punto máximo** X_MAX_PIN 2

Para controlar el segundo motor:

- **Pin para el control de pulsos del motor** Y_STEP_PIN 60
- **Pin para el control de la dirección del motor** Y_DIR_PIN 61
- **Pin para el fin de carrera del punto mínimo** Y_MIN_PIN 14
- **Pin para el fin de carrera del punto máximo** Y_MAX_PIN 15
- **Pin para activar el motor** Y_ENABLE 56

Esta placa tiene también unos pines libres donde podemos colocar otros periféricos, en este caso esos pines irán a la pantalla DISCOUNT y su encoder.

- **Pin para el encoder** BTN_EN1 31
 - BTN_EN2 33
 - BTN_ENC 35
- **Pin para el beeper** BEEPER 37
- **Pin para la pantalla LSD** LCD_PINS_RS 16
 - LCD_PINS_ENABLE 17
 - LCD_PINS_D4 23
 - LCD_PINS_D5 25
 - LCD_PINS_D6 27
 - LCD_PINS_D7 29

[Escriba aquí]

[Escriba aquí]

6.1.2. Controlador DRV8825

Un motor paso a paso se mueve cambiando el voltaje de las bobinas que lo forman siguiendo un determinado orden, si se quisiera controlar directamente con el Arduino requeriría un complicado circuito, que sería caro y aparatoso. Por eso existen controladores o drivers como este para poderlo controlar usando 2 pines nada más del Arduino y un circuito bastante pequeño por un precio reducido.

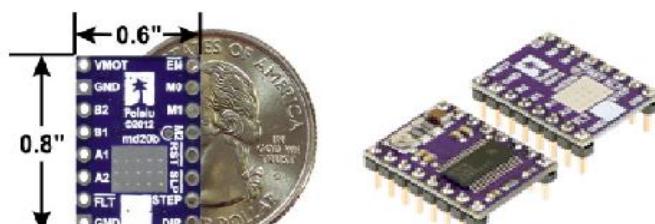


figura 32: Controlador DRV8825

Este controlador tiene unas dimensiones de 0,6 x 0,8 pulgadas (14,25 x 20,32 mm) y un peso de 1,6 g, es compatible con la CNC shield RAMPS 1.4 y tiene estas propiedades:

- **Voltaje mínimo /máximo de operación.....** 8,2 / 45 V
- **Corriente máxima por fase.....** 2,2 A
- **Mínimo/máximo voltaje lógico.....** 2,5 / 5,25 V
- **Resolución de micropaso.....** $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$

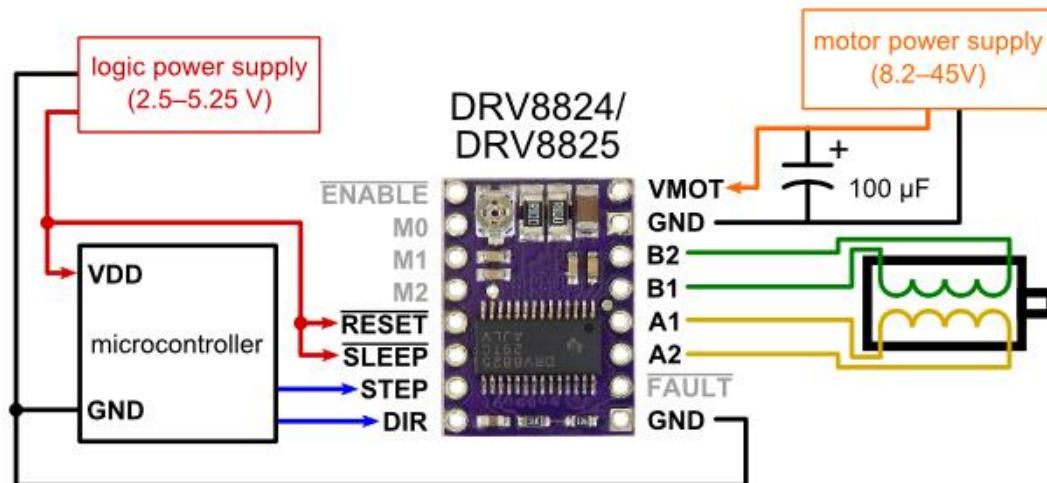


figura 33: Circuito de instalación de un DRV8825

Este complicado circuito de montaje entre el microcontrolador, el controlador y el motor está incorporado en el CNC shield, simplificando aún mas el circuito.

Por lo que, puede operar directamente con el voltaje lógico de Arduino ya que esta entre 2,5 y 5,25 V y el motor podrá funcionar entre 8,2 y 45 V y a una tensión máxima de 2,2 A.

[Escriba aquí]

[Escriba aquí]

6.1.3. Motor NEMA23

El motor es la principal fuente de fuerza en el diseño. Por eso será uno de los factores principales para limitar las dimensiones del modelo análogo



figura 35: Interior de un motor paso a paso

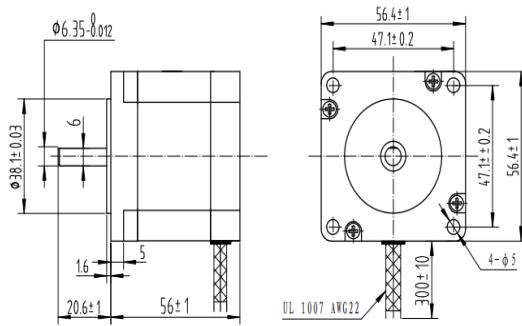


figura 34: Medidas del motor NEMA 23

| Especificación | parámetro | unidades | Motor NEMA23 | comentario |
|----------------------------|----------------------|----------|--------------|------------------------|
| Angulo de paso | | ° | 1.8 | |
| Precisión del paso | | % | 5 | |
| Voltaje nominal | V _{NOM} | V | 7.4 | |
| Corriente nominal por fase | I _{RMS NOM} | A | 2 | 1A por bobina |
| Inductancia por fase | R _{COIL} | Ω | 7.4 | 1.6Ω + 5.5 mH por fase |
| Torque (cortocircuito) | | N*cm | 90 | |

Estos son los valores de referencia del motor, pueden cambiar si el motor no trabaja en su estado nominal de corriente y tensión o al cambiar la velocidad de giro del motor.

Cada x pulsos (x dependerá del modo del controlador, puede ser 1, 2, 4, 8, 16 o 32) el motor gira el ángulo de paso, 1.8°. Si se para el motor mientras esta en movimiento avanzará o retrocederá a una posición estable que son las que son múltiplo de $\frac{1}{2}$ de paso (0, 9°).

[Escriba aquí]

[Escriba aquí]

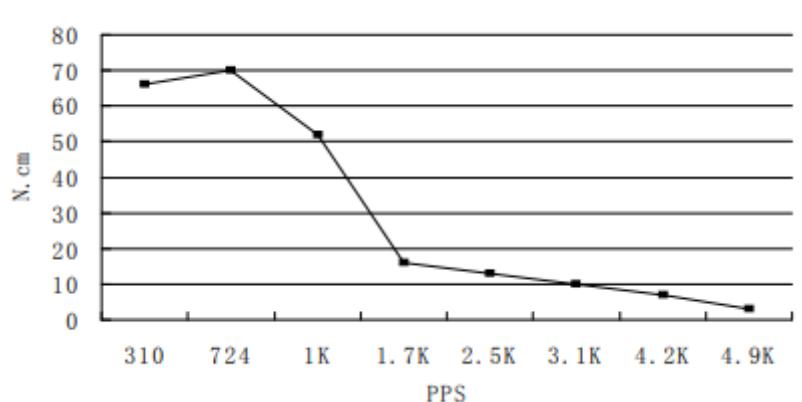


figura 36: curva Torque frente a frecuencia del motor NEMA 23

El torque se mantiene estable entre los 310-724 pps en 65-70 N*cm, desde ese punto cae lentamente hasta las 1000 pps donde ejerce poco más de 50 N*cm, finalmente se reduce hasta prácticamente 0 en los 5000 pps.

Por eso consideraremos que el valor mínimo de velocidad de trabajo para los cálculos de fuerza será 300pps y el máximo 720 porque en este intervalo consigue una velocidad uniforme y máxima. Podría trabajar a otras velocidades, pero la fuerza ejercida no sería la calculada.

A 300 pps permitiría dar entre 16,88°/s (o lo que es lo mismo, 1 giro cada 213,3 segundos), si el controlador avanza un $\frac{1}{32}$ de paso por pulso, y 540°/s (o lo que es lo mismo, 1,5 giros/s), si el controlador avanza 1 paso por pulso.

6.1.4. Pantalla LCD

La pantalla LCD se utilizará como interfaz con el usuario para introducir variables o presentar información. La pantalla LCD modelo DISCOUNT está especialmente diseñada para conectar con la CNC shield que se utiliza en este diseño y ser controlada con I2C (un programa para controlar con muy pocos pines una pantalla LCD) o MARLIN (un programa muy útil y versátil para controlar varios modelos de estas pantallas y de CNC shield enfocado principalmente a impresoras 3D, aunque se puede adaptar a otros diseños) u otros programas de Arduino especiales para estas pantallas.

[Escriba aquí]

[Escriba aquí]

Este modelo facilita mucho la conexión porque conecta directamente cada pin con su pin correspondiente de las RAMPS a través de un periférico como viene en la ilustración, y en la data sheet de la pantalla viene indicado que pines del Arduino MEGA son (se puede ver en este documento entre la información de la CNC SHIELD)



figura 37: Pantalla LCD DISCOUNT

Esta pantalla tiene espacio para 20 caracteres por fila y 4 por columna, y lleva incorporado un BEEPER, y un ENCODER.

6.2. Trucos para la programación con Arduino

6.2.1. Programar una pantalla LCD

La pantalla es donde se presentará la interfaz para que el usuario pueda trabajar sobre ella, por eso es uno de los puntos más importantes del proyecto.

Para utilizarla se propusieron varias alternativas como el programa Marlin que está muy utilizado en la industria de las impresoras 3D y es muy eficaz, pero adaptarlo a este proyecto aumentaba notablemente la dificultad y resultaba más sencillo y didáctico realizar un código propio; o la librería I2C que permite reducir los pines necesarios, pero resultaba más fácil para la instalación de la pantalla utilizar el adaptador de montaje en la placa CNC que viene pensado para utilizar 7 pines.

Para realizar el código, se debe incluir la librería “LiquidCrystal.h” que facilita el escribir directamente en la pantalla, de esta manera solo se necesitan utilizar los comandos “lcd.print (“ y lo que se dese escribir para que aparezca por pantalla

Para definir que pantalla esta conectada y a que pines; esta librería tiene el comando” LiquidCrystal lcd(RS,ENABLE,D4,D5,D6,D7)” para indicar los pines donde va conectada la pantalla y el comando ”lcd.begin(20,4);” para indicar las dimensiones de la pantalla (en este caso 20 columnas x 4 filas).

6.2.2. Programar un Encoder

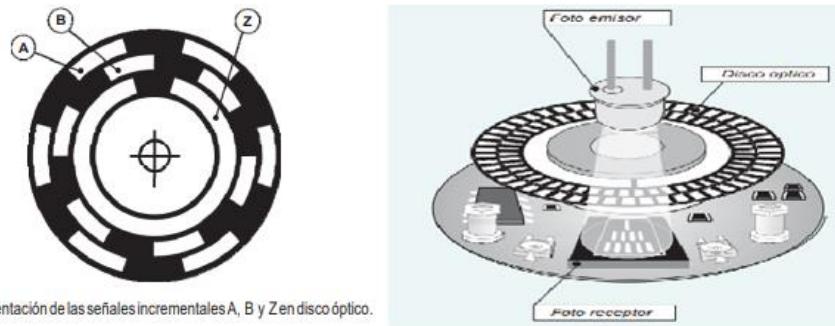
Los encoder aquí se utiliza como herramienta del usuario para navegar por la interfaz.

Un encoder es un aparato que codifica el giro de una ruleta (en el caso del utilizado para el diseño también puede detectar si se presiona la ruleta, pero esta aplicación se programa como si fuese un botón normalmente abierto) y lo transmite a través de señales digitales LOW (0) o HIGH (1).

Este encoder controla el giro con un disco ranurado con dos filas (aunque pueden ser más) de huecos que se denomina *disco óptico* que se lee a través de un fotoemisor y un fotorreceptor , de esta manera al girar los huecos dejan pasar la luz siguiendo un orden y el fotorreceptor convierte eso en una señal digital.

[Escriba aquí]

[Escriba aquí]



Representación de las señales incrementales A, B y Z en disco óptico.

figura 38: Funcionamiento de un encoder

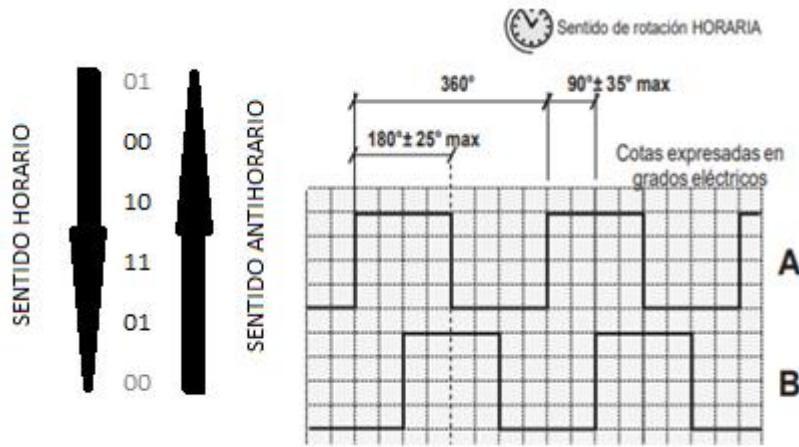


figura 39: señales de un encoder

Como los huecos en las dos filas del disco óptico no están alineados, las señales de las filas van desfasadas, en función de cual vaya adelantada podemos saber hacia dónde gira la ruleta (si no cambian es que o el giro ha sido muy pequeño o que está quieto).

[Escriba aquí]

[Escriba aquí]

Sabiendo esto podemos realizar una tabla de verdad para relacionar el estado anterior y siguiente con el giro que se esté tomando ya sea horario o antihorario (marcamos como 1 los casos favorables, como 0 los desfavorables y como x los imposibles).

Tabla 1: Tabla de la verdad del giro del encoder

| ESTADO | | ESTADO ANT | | horario | antihorario |
|---------|---------|--------------|--------------|---------|-------------|
| btn_en2 | btn_en1 | btn_en2_prev | btn_en1_prev | | |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | x | x |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | x | x |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | x | x |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | x | x |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

[Escriba aquí]

[Escriba aquí]

Así sacamos las siguientes tablas de Karnaugh donde en el resultado 1 indica que la variable tiene que ser HIGH, 0 indica que tiene que ser LOW y x que es indiferente, siguiendo el orden Q1, Q2, Q1', Q2':

Tabla 2: Tablas de Karnaugh del giro del encoder

| horario | | | | | antihorario | | | | |
|----------|---------|---------|---------|---------|---------------|---------|---------|---------|---------|
| Q\Q' | 0.0 | 0.1 | 1.1 | 1.0 | Q\Q' | 0.0 | 0.1 | 1.1 | 1.0 |
| 0.0 | 0 | 0 | x | 1 | 0.0 | 0 | 1 | x | 0 |
| 0.1 | 1 | 0 | 0 | x | 0.1 | 0 | 0 | 1 | x |
| 1.1 | x | 1 | 0 | 0 | 1.1 | x | 0 | 0 | 1 |
| 1.0 | 0 | x | 1 | 0 | 1.0 | 1 | x | 0 | 0 |
| horario= | x.1.0.0 | 1.1.0.x | 1.0.x.1 | 0.x.1.0 | antihorario = | 1.x.0.0 | x.0.0.1 | 0.x.1.1 | 1.1.x.0 |

Así podemos programar si el usuario está girando hacia la derecha (horario) o a la izquierda (antihorario)

Nota 1: las conexiones del encoder necesitan una resistencia de pull up (se puede usar la del propio Arduino). Si no se usan, el Arduino leerá valores aleatorios.

Nota 2: Si el programa se desarrolla lento y se gira rápidamente la ruleta puede que se salte la lectura de algún valor, llevando a interpretaciones erróneas. por eso es particularmente importante que el programa se ejecute siempre rápido

A la hora de programar la lectura del encoder, es importante programar un detector de flanco para evitar que avance por el interfaz muy rápido y resulte difícil de controlar, si no, una alternativa es no escribir todas las combinaciones del estado actual y el anterior para cada tipo de giro (horario o antihorario); si ponemos menos, avanzara más lento. Pero si queremos controlar bien cuanto giro hace falta para avanzar 1 valor lo mejor es un contador que cada cierto número de avances en un sentido envíe un pulso, aunque habrá que hacer un balance entre el incremento de precisión y el tiempo de más que tardara el programa en ejecutarlo.

[Escriba aquí]

[Escriba aquí]

6.2.3. Programación de estados de forma optima

Por regla general, un programa tiene 2 partes, el SETUP que es donde se inician las variables y se definen los pines del programa y el LOOP, la parte que se repite constantemente mientras dure el programa.

La estructura de este programa se divide en 3 partes: el SETUP, el LOOP y los ESTADOS

En el SETUP definiremos los pines que utilizaremos en el programa e iniciaremos las variables (esto solo se lleva a cabo 1 vez al iniciar el programa)

PINs de entradas:

Fin de carrera de inicio
Fin de carrera del final
Encoder (x3 pines)

PINs de salidas:

Pasos del motor
Dirección del motor
BEEPER
Pantalla DISCOUNT

En el LOOP es donde leeremos los comandos de la interfaz (a través de los pines del encoder y los fines de carrera) y escribiremos algunos pines de salida, pero esta parte del programa se reduce al mínimo porque esta parte se repetirá siempre pase lo que pase y recortar aquí es altamente eficiente. Las funciones más notables del LOOP son:

-Discriminación entre girar a derecha (sentido horario) o izquierda (antihorario):

Utilizando lo aprendido en el apartado anterior “6.2.2 Programar un encoder” se comparan las variables actuales con las del tiempo anterior, hay 4 combinaciones favorables para decir que la ruleta gire en sentido horario y otras 4 para decir que gira antihorario y si no coinciden se considera que no se mueve la ruleta.

[Escriba aquí]

[Escriba aquí]

```
//////////  
if (btn_en1 != btn_en1_prev || btn_en2 != btn_en2_prev)  
{  
    if ( btn_en2 == false & btn_en1 == false & btn_en2_prev == true & btn_en1_prev == false)  
    {  
        derecha = true;  
        izquierda = false;  
    }  
    else if( btn_en2 == false & btn_en1 == false & btn_en2_prev == false & btn_en1_prev == true )  
    {  
        derecha = false;  
        izquierda = true;  
    }  
    else  
    {  
  
        derecha = false;  
        izquierda = false;  
    }  
}
```

figura 40: Código para discriminar entre el giro horario o antihorario

Sin embargo, al ver que esto hacía difícil de controlar se redujeron las 4 opciones favorables a solo 1, reduciendo la velocidad a una cuarta parte, y ahora resulta cómodo de controlar. Otra opción sería crear un contador para tener control más preciso de cuanto hay que girar el encoder para considerar que ha girado y enviar 1 pulso, sin embargo, reducir las combinaciones favorables es el método más eficiente reduciendo el código y es suficiente para hacer cómoda la navegación por la interfaz.

-Definición de estados

Utilizamos una variable que se llama “estado” para escoger a que ESTADO debemos ir, de esta manera solo tenemos que cambiar el valor de la variable estado cuando se haya terminado esta

```
/////////  
switch (estado) // gestion de estados  
{  
    case 1:  
        DefinicionDeVariables();  
        break;  
    case 2:  
        preparacion ();  
        break;  
    case 3:  
        accion ();  
        break;  
    case 4:  
        pausa();  
        break;  
    case 5:  
        reinicio();  
        break;  
    default:  
        inicio();  
        break;  
}/////////
```

figura 41: Código para la selección de estados

parte del programa para pasar al siguiente ESTADO.

En el apartado de ESTADOS es donde se lleva el grueso del programa, funcionan como subprogramas independientes que pueden ser llamados desde el LOOP.

Cada estado se escribe utilizando “void ESTADO () ;” y escribiendo lo que queremos que se haga en ese estado como si fuese el LOOP. Después solo hay que escribir “ESTADO () ;” como un comando cualquiera cuando queramos que se ejecute esa parte del código.

[Escriba aquí]

[Escriba aquí]

6.2.4. Error en la medida de la velocidad

La velocidad es una de las variables que el usuario puede escoger, sin embargo, es una variable que por su manera de calcularse comete un error sistemáticamente entre lo que se da a elegir y lo que se obtiene debido al tiempo que tarda en ejecutarse el programa.

La velocidad en este programa, por el sistema mecánico que lo acompaña se calcula como:

$$234 / 2*T(\mu s) [\mu m/s]$$

Donde T es el tiempo de un ciclo en milisegundos (para explicar esta fórmula se puede ver el apartado de cálculos mecánicos). Idealmente T es igual a un tiempo “t” que se pide que espere el programa a través del comando **DELAY()**, pero se desvía de la idealidad cuando consideramos que el programa tarda un tiempo “ t_0 ” en llevar a cabo todos los comandos.

$$T = t + t_0$$

Para solucionar este problema se debe conseguir el programa tarde el mínimo tiempo posible, es decir que $t_0 \approx 0$ y crear un circuito cerrado, de manera que mida el tiempo que tarda el programa en realizarse y se reste ese tiempo a t para que el tiempo real de espera sea lo más parecido posible, de esta manera:

$$t' = t - t_0$$

$$T = t' + t_0 = t - t_0 + t_0 = t$$

El problema con el circuito cerrado es que el comando **DELAY()** solo puede realizar esperas múltiples de 1 milisegundo, y para cómo funciona el programa un milisegundo es una unidad muy grande, con lo que tendríamos que medir t_0 en μs y utilizar el comando **DELAYMICROSECONDS()** que permitiría aumentar la precisión pero solo se podrían hacer esperas de hasta 16383 μs (16,38 ms) y habría que considerar la posibilidad de que t_0 fuese más grande que t y para que no se intente una espera menor que 0.

6.2.5. Optimización del programa

El programa necesita, para ser funcional, ser capaz de realizarse de la forma más rápida posible aun siendo complejo. Esto significa que en cada ciclo pase por el número mínimo de comandos (especialmente evitando los comandos que más tiempo tardan).

De los comandos, los más complejos y que más tardan son los de escribir la pantalla LCD, sin embargo, para evitarlos se puede jugar con que si no se reescribe una celda de la pantalla LCD se

[Escriba aquí]

[Escriba aquí]

mantiene como estaba en el ciclo anterior. Esto se aplica escribiendo al cambiar de estado todo lo que saldrá por pantalla en el estado siguiente y no cambiará, como pueden ser los rótulos de distancia inicial y final del estado de definición de variables ya que, aunque cambia el valor que va después el título se mantiene fijo.

```
void inicio() ///////////////INICIO, estado 0/////////////////
{
    if (pulsador == true)
    {
        estado=1;
        estado_ant = 0;
        x_k_posicion=0;
        dist=0;
        dif= 0;
        v = 0;
        s=0;
        x_mm= 0;

        lcd.clear();
        lcd.setCursor(1, 0);
        lcd.print("dist. inic");
        lcd.setCursor(18, 0);
        lcd.print("mm");
        lcd.setCursor(1, 1);
        lcd.print("dist. final");
        lcd.setCursor(18, 1);
        lcd.print("mm");
        lcd.setCursor(1, 2);
        lcd.print("velocidad");
        lcd.setCursor(16, 2);
        lcd.write(byte(2));
        lcd.setCursor(17, 2);
        lcd.print("MM/s");
        lcd.setCursor(1, 3);
        lcd.print("iniciar experimento");
    }
}
```



figura 43: ejemplo Código para escribir el texto constante del estado siguiente

figura 42: Pantalla LCD en el estado de definición de variables del modelo simple

Otro método que se ha usado consiste en no escribir las variables que aparecen en pantalla en los estados en los que los motores funcionan en todos los ciclos, sino solo cuando cambia, esto se ve directamente afectado por la precisión del programa, a mayor precisión más veces se debe actualizar la medida.

Este método, sin embargo, permite una mayor variación ya que como se debe reescribir el código dentro de cada condicional para que solo se actualicen las casillas cuando se cumplan concretas condiciones, eso permite que cada condicional tenga sus propios rótulos, y que como ocurre con el estado de pausa tenga hasta 5 variaciones pero un código muy sencillo, porque de forma general solo envía al reinicio si eliges “terminar” o al estado anterior si eliges “continuar” que esta almacenado mediante una variable llamada “estado_ant” que guarda el estado anterior

En los casos donde el estado actual ha terminado y se debe pasar por pausa para ir al estado siguiente al pasar a pausa no se guarda en “estado_ant” el estado en el que nos encontramos sino al que el programa ira después de la pausa.

```
if ( x_mm >= di_X and y_mm >= di_Y)
{
    estado = 4;
    estado_ant = 3;
    lcd.clear();
    lcd.setCursor(1, 0);
    lcd.print("PAUSA");
    lcd.setCursor(0, 1);
    lcd.print("X");
    lcd.setCursor(0, 2);
    lcd.print("Y");
    lcd.setCursor(0, 3);
    lcd.print(" CONTINUAR");
    lcd.setCursor(12, 3);
    lcd.print("terminar");
    lcd.setCursor(2, 1);
    lcd.print(x_mm);
```

figura 44: Código para la actualización de las variables de estado

[Escriba aquí]

[Escriba aquí]

6.2.6. Complemento a 2 y representación de variables de gran valor

Arduino a pesar de presentar en todo momento números de base 10 (en el código o al presentar números por pantalla) trabaja internamente en complemento a 2.

El complemento a 2 (C2) es un método para representar números positivos y negativos en un rango que dependerá de los bits de la variable. Los números positivos se escriben en binario de la forma habitual solo que su valor máximo de 2^{N-1} siendo N el número de bits totales, de manera que los números positivos siempre tienen un 0 en la primera posición. Los números negativos, sin embargo, no se representan directamente igual que su contraparte positiva; se debe conocer el numero binario positivo, restarle 1 y sustituir los 1 por 0 y los 0 por 1. De esta manera para N= 4 bits se pueden escribir números en un rango de entre -8 y 7

Tabla 3: Rango de valores en C2 con 4 bits

| Binario (positivo) Complemento 2(negativo) | a | Decimal |
|--|---|---------|
| 0111 | | 7 |
| 0110 | | 6 |
| 0101 | | 5 |
| 0100 | | 4 |
| 0011 | | 3 |
| 0010 | | 2 |
| 0001 | | 1 |
| 0000 | | 0 |
| 1111 | | -1 |
| 1110 | | -2 |
| 1101 | | -3 |
| 1100 | | -4 |
| 1011 | | -5 |
| 1010 | | -6 |
| 1001 | | -7 |
| 1000 | | -8 |

[Escriba aquí]

[Escriba aquí]

Esto se debe a que para la maquina es más fácil realizar operaciones matemáticas utilizando este sistema.

Al presentar una variable debemos indicar de qué tipo de variable se trata, en este programa para las variables numéricas utilizamos los comandos INT y LONG de esta manera Arduino asigna un numero de bits a la memoria de esa variable. En el caso de INT 16 bits y en el caso de LONG 32 bits.

Este concepto puede resultar trivial si trabajamos con números pequeños, pero al alcanzar cierto valor las operaciones dan respuestas inesperadas. Por ejemplo, si sumamos $7 + 3$ podemos esperar que la respuesta sea 10, pero si trabajamos en 4 bits en C2 al realizar esa cuenta $0111 + 0011 = 1010$, o lo que es lo mismo en base 10, $7 + 3 = -6$. Por eso es importantes que todas las operaciones se mantengan dentro del rango de la variable que estemos escribiendo.

Este programa tiene variables como el número de micras que se desplaza un puente móvil (x) que pueden llegar a 400 000, el comando INT tiene un rango de -2^{15} a $2^{15}-1$ (-32 768 a 32 767) y el comando LONG tiene un rango de -2^{31} a $2^{31}-1$ (-2 147 483 648 a 2 147 483 647).

En este caso es fácil ver que la variable x no se puede considerar directamente una variable INT porque se saldría del rango. Por lo que tendríamos 2 opciones: Primera opción, utilizar 2 variables tipo INT para medir x, por ejemplo, una que almacene los valore por encima de 1000 (el valor del desplazamiento en mm) y otra con los valores por debajo de 1000 y ajustar la interacción entre ellas. Y segunda opción, utilizar una variable tipo LONG y que, si se hacen operaciones matemáticas con enteros, al menos uno de los números debe estar seguido de una L, forzando a este número a ser un LONG.

Cada opción tiene sus ventajas y desventajas, la primera opción es más compleja, necesita utilizar más variables y requiere un cuidado extra para regular la interacción entre las variables que componen un número, pero permite que se realicen las operaciones matemáticas de forma más simple (interviniendo menos bits) y al actualizar una variable en la pantalla LCD habrá que reescribir menos casillas.

[Escriba aquí]

[Escriba aquí]

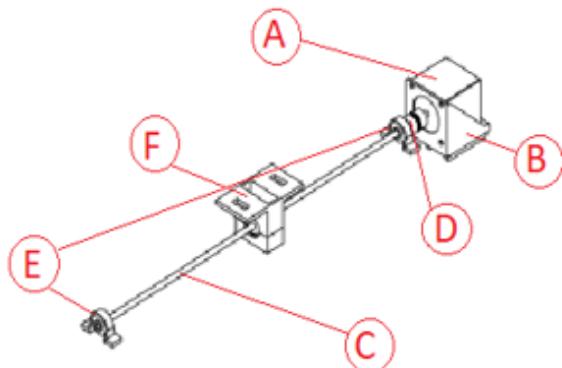
7. Análisis mecánico del diseño

7.1. Partes del diseño

Este diseño se compone de 4 partes principales: sistema de transmisión, sistema de apoyo, puente móvil y estructura de la base de la máquina.

7.1.1. Sistema de transmisión

El sistema de transmisión es el encargado de conectar el motor al resto del diseño y transmitir su fuerza al puente móvil.

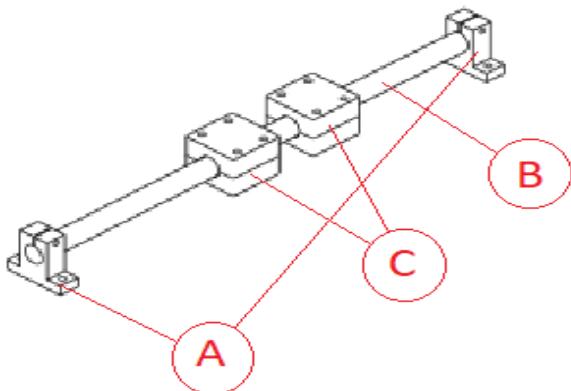


| | |
|---|----------------------------|
| A | Motor NEMA 23 |
| B | Soporte para motor |
| C | Husillo TR 8x1.5 |
| D | Acoplamiento Flexible |
| E | Rodamiento con brida a 90° |
| F | Tuerca para husillo |

figura 45: sistema de transmisión

7.1.2. Sistema de apoyo

Este sistema se encarga de soportar y equilibrar el peso del puente móvil para que este no rote ni se caiga.



| | |
|---|---------------------------|
| A | Soporte para eje de acero |
| B | Eje de acero D 20mm |
| C | Cojinete LME20UU |

figura 46: sistema de apoyo

[Escriba aquí]

[Escriba aquí]

7.1.3. Puente móvil

Esta es la parte del sistema que absorbe la fuerza del motor y la transmite al modelo análogo para realizar el experimento.

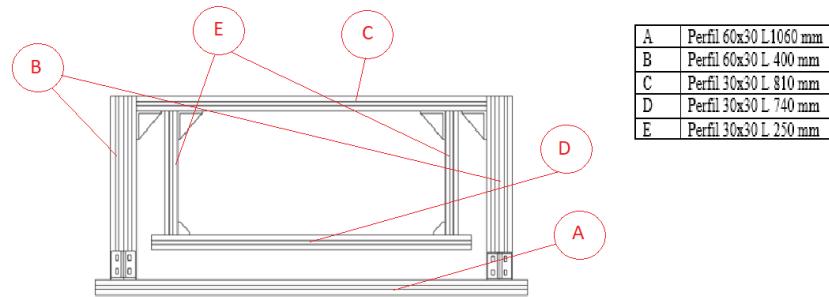


figura 47: Puente móvil

7.1.4. Estructura de la base de la maquina

Soporta el peso de la maquina y el modelo análogo, y contiene el sistema de transmisión y el sistema de apoyo

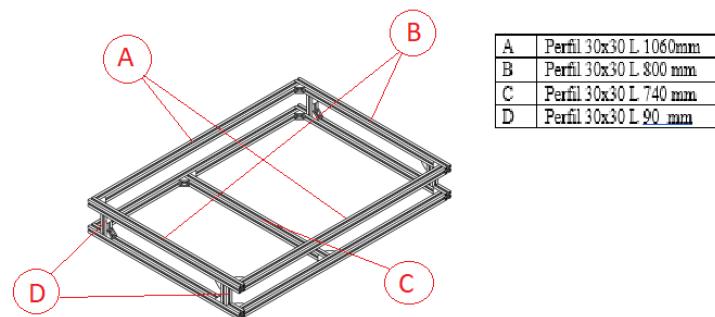


figura 48: estructura de la base de la maquina

[Escriba aquí]

[Escriba aquí]

7.2. Análisis mecánico de la estructura

7.2.1. Análisis de las velocidades transmitidas

El motor NEMA 23 mantiene un torque más o menos constante de 65-70 N/cm entre los 310pps y los 724pps y un paso de 1,8°; el husillo tiene un paso de 1,5 mm de paso y el controlador utilizará una resolución de 32 pulsos por paso.

Con estos datos es posible calcular la velocidad a la que se movería puente móvil:

$$V_{max} = pps_{max} * pasohusillo * pasomotor * resolución =$$

$$724pps * 1500\mu\text{m} * \frac{1,8^\circ}{360^\circ} * \frac{1}{32} = 169,69 \mu\text{m/s}$$

$$V_{min} = pps_{min} * pasohusillo * pasomotor * resolución =$$

$$310pps * 1500\mu\text{m} * \frac{1,8^\circ}{360^\circ} * \frac{1}{32} = 72,65 \mu\text{m/s}$$

Fuera de estas velocidades la fuerza puede reducirse notablemente por lo que no se consideran recomendables para realizar el experimento con precisión.

Por otro lado, para la programación se puede escribir esta fórmula de otra manera considerando que los pulsos por segundo se consiguen enviando durante el tiempo de un ciclo una señal HIGH y durante el siguiente una señal LOW:

$$PPS = \frac{1000 \text{ (ms)}}{2 * t(\text{ms})}$$

$$V = \frac{1000 * pasohusillo * pasomotor * resolucion}{2 * Tiempo \text{ de cada ciclo}} = \frac{117,18}{t(\text{ms})}$$

[Escriba aquí]

[Escriba aquí]

7.2.2. Análisis de la fuerza transmitida

El motor es la principal fuente de fuerza en este diseño. En las tablas de datos del motor se extrae el torque generado en función de las pps, en este caso tomaremos como referencia el valor máximo en movimiento de 70 N*cm a 724 pps.

La fuerza se transmite mediante el husillo, cuyos datos técnicos son el paso $P=1.5$ mm, un diámetro $D_m=6.5$ mm ($R_m = 0.325$ cm) y un ángulo de filete $\alpha=3.89^\circ$ (0.068 rad). Con estos datos el motor ejerce una fuerza radial:

$$Fr=Mt/Rm=70 [N*cm]/0.325[cm]=215.38 [N]$$

El valor máximo de esa fuerza en movimiento $F_{r\max}$ 215.38 N

Si consideramos que las fuerzas se equilibran en el plano inclinado del filete durante el movimiento, la normal ejercida por el filete es igual a la suma de la fuerza tangencial (F_t) y a la fuerza de empuje (F_z) menos la fuerza de rozamiento ($N*\mu$) (vectorialmente).

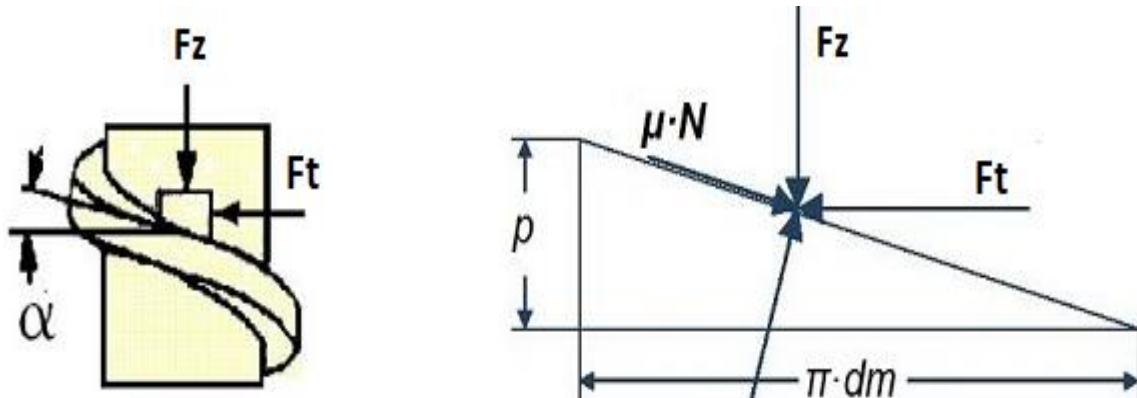


figura 49: Descomposición vectorial de fuerzas en el husillo

De modo que se obtiene el sistema de ecuaciones:

$$F_z = N * [\cos(\alpha) - \mu \sin(\alpha)]$$

$$F_t = N * [\sin(\alpha) + \mu \cos(\alpha)]$$

[Escriba aquí]

[Escriba aquí]

De forma general al despejar Fz y se consigue esta fórmula:

$$F_z = F_t * \frac{[\cos(\alpha) - \mu \sin(\alpha)]}{[\sin(\alpha) + \mu \cos(\alpha)]} = \frac{M_t}{R} * \frac{[\cos(\alpha) - \mu \sin(\alpha)]}{[\sin(\alpha) + \mu \cos(\alpha)]}$$

Nota: el ángulo α es lo más pequeño posible, para obtener más precisión, de modo que podemos suponer que $\cos(\alpha) \sim 1 \gg \sin(\alpha) \sim \alpha$, con estas simplificaciones la formula se reduce a:

$$F_z \approx \frac{M_t}{R} * \left[\frac{1 - \mu\alpha}{\alpha + \mu} \right] \sim \frac{M_t}{R} * \left[\frac{1}{\alpha + \mu} \right]$$

A la vista de esta fórmula, podemos concluir que el empuje será mayor cuanto mayor sea el momento torsor (M_t), el radio del husillo ($R = D_m/2$) y cuanto menor el rozamiento (μ).

Si tomamos $\mu = 0.05$ (en los husillos de bolas los valores típicos para el rozamiento están entre el 0.01 y 0.05, tomamos 0.05 porque es el que dará valores más pequeños) obtenemos una fuerza de 1825.29 N.

La fuerza F_z es la que obtenemos suponiendo que el sistema es ideal, pero si consideramos un rendimiento del 80% la fuerza máxima se queda en **Fmax= 1460.23 N** que son aproximadamente 146 kg de fuerza.

7.3. Diseño de piezas para imprimir 3D en Freecad

Todas las piezas impresas 3D sirven para unir piezas al resto del sistema, como ejemplo para este apartado se hablará del ejemplo de la tuerca del husillo, que necesita una pieza que une el husillo al puente móvil para transmitir el movimiento del motor.

El primer paso es conocer bien la pieza con todas sus dimensiones, para ello lo mejor es conseguir un diseño CAD de esta, porque en muchos casos las hojas de datos técnicos no dan información suficiente para realizar estos diseños.

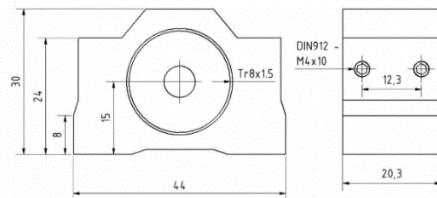


figura 50: Ejemplo de datos técnicos insuficiente para un diseño CAD

[Escriba aquí]

[Escriba aquí]

Para conseguir el diseño CAD muchas páginas ofrecen entre la descripción de la pieza archivos que se pueden abrir con cualquier programa de modelado como Freecad. Luego solo habrá que colocarlo en el lugar que ocupa en diseño para conocer la distancia que tiene con el resto de los componentes de su entorno.

| |
|--|
| Detalles: Trapezoidal plomo tuerca 8x1,5 R bronce rojo con caja de aluminio |
| Consejos |
| Más imágenes |
| 3D files |
| Motedit_Trapezoidal_lead_screw_nut_8x1.5_R_red_bronze_with_aluminium_housing.zip |
| Compras relacionadas |

figura 51: Enlace de descarga del archivo CAD de la pieza

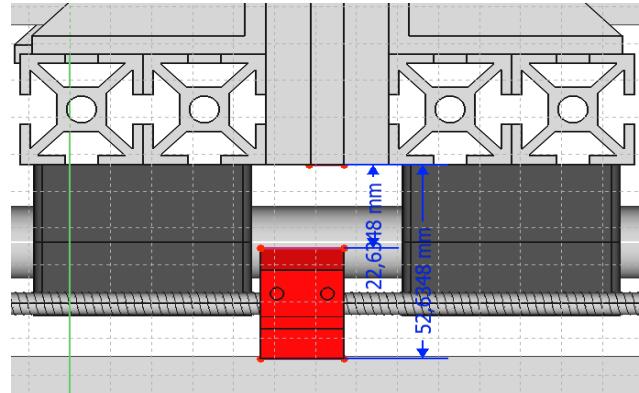


figura 52: Pieza CAD en el contexto del resto de piezas del ensayo

Después es recomendable abrir un documento nuevo con la pieza y utilizar los datos de su entorno para diseñar la pieza/s para imprimir. En esta etapa también es importante tener en cuenta las uniones roscadas que vayan asociadas a estas.

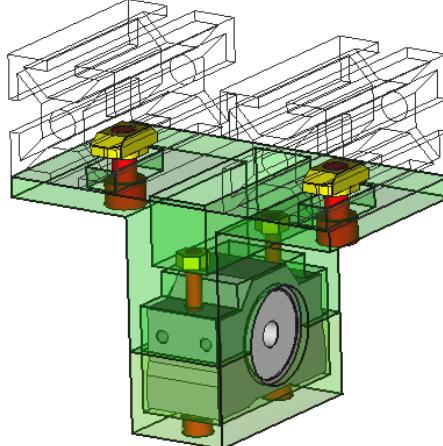


figura 53: construcción de un modelo Freecad de la pieza a imprimir

Una vez diseñada la pieza para imprimir en 3D solo hay que seleccionar la pieza e irnos al apartado de “archivo” en la esquina superior izquierda, clicar en exportar y guardarla como archivo STL MESH, consiguiendo así un mayado útil para imprimir directamente.

[Escriba aquí]

[Escriba aquí]

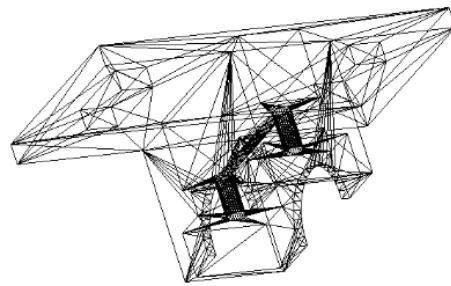


figura 54: mayado de una figura Freecad

7.4. Diseño de planos con Freecad

Para transmitir con la máxima precisión los detalles del proyecto de forma técnica se pueden utilizar planos. Freecad permite realizar planos de las piezas diseñadas de forma sencilla. Primero se debe tener el diseño tipo CAD, que puede tener una pieza o varias, si tiene varias se deben unir en una sola con los comandos lógicos de unión booleana (en el marco de “parts”)

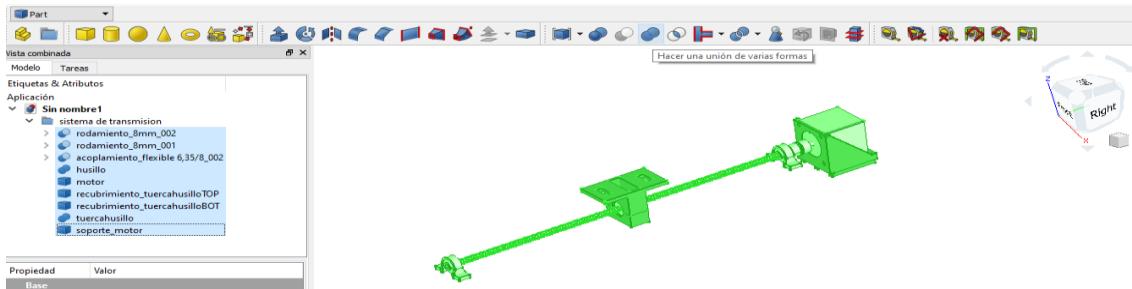


figura 55: Interfaz de Freecad para unir piezas

En el marco de “draft” se encuentra un comando llamado “cotas”, una herramienta que permite crear cotas de forma sencilla, solo se tiene que escoger los extremos de la medida que queremos tomar y nos aparecerá como otro objeto con nombre “dimensión” que se puede editar para delimitarla con flechas, precisar el tamaño de la letra o precisar el tamaño del “overshoot” de la medida.

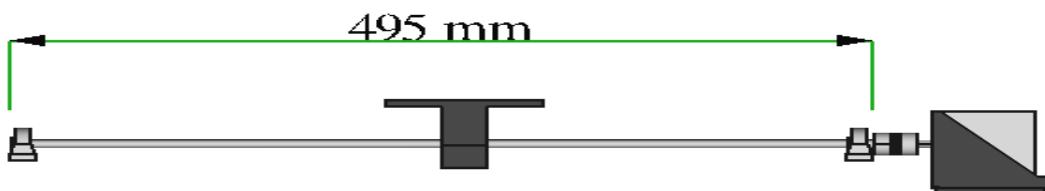


figura 56: Figura con cotas en Freecad

[Escriba aquí]

[Escriba aquí]

Después se tiene que abrir un documento A3 (se puede conseguir en el marco de “drawing”) y seleccionar en el marco de “drawing” la opción de “insertar proyecciones ortogonales”. De esta manera aparecerán en el documento A3 una de las vistas de la figura y una cruceta en la ventana de comandos con cuadros seleccionables, al clicar sobre ellos aparecerán otras vistas que de forma automática se escalarán para caber todas en el plano.

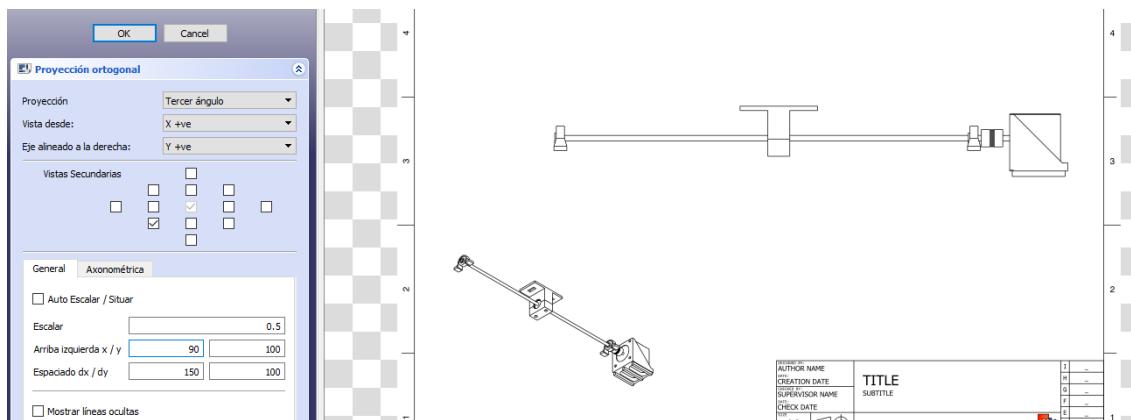


figura 57: Herramienta de planos en Freecad

Sin embargo, conseguir un buen plano con todas las ilustraciones, cotas y anotaciones correctamente indicadas solo utilizando Freecad puede resultar extremadamente tedioso especialmente lo relacionado con anotaciones y colocación de cotas.

Por eso en este proyecto se ha utilizado Freecad para conseguir las vistas y las cotas del diseño, y luego se ha exportado a “*inkscape*”, una aplicación de diseño gráfico donde se puede descomponer el plano en sus elementos y editarlos.

Con esta aplicación se pueden colocar las vistas en la parte del plano que resulte más visual, editar y mover las cotas para que el plano quede limpio, y crear anotaciones que no se pueden hacer en Freecad de forma directa.

7.5. Uso mods para Freecad para facilitar el diseño.

GitHub es una plataforma para compartir diseños, de este modo además de poder subir diseños podemos descargar los de otra gente.

En este caso para el diseño electrónico hay ciertas piezas que se usan repetidamente, como pueden ser los perfiles 30x30 o los tornillos y tuercas entre otros, y rediseñarlos cada vez que se quieran usar ralentizaría muchísimo el trabajo con Freecad. Por eso se instaló el paquete de “mecatrónica” de otro compañero que permite crear piezas normalizadas utilizando unos pocos comandos.

[Escriba aquí]

[Escriba aquí]

Para utilizarlo solo tenemos que acceder al repositorio <https://github.com/davidmubernal/Mechatronic> y descargar el mod en formato .zip, después solo tenemos que seguir las instrucciones de instalación.

De este modo encontraremos la próxima vez que abramos FreeCAD un marco nuevo llamado "mecatrónica" donde tendremos a nuestro alcance muchas piezas fácilmente editables.

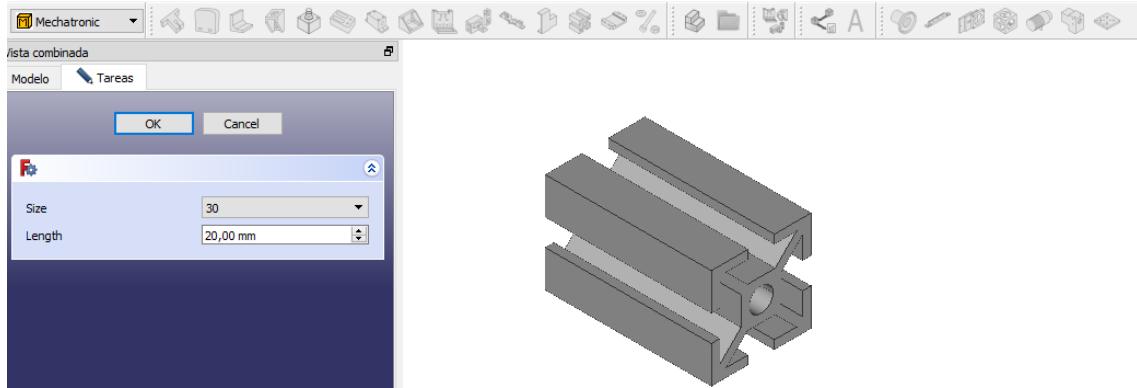


figura 58: interfaz del marco de mecatrónica

Mods como este son herramientas comunes muy potentes, capaces de reducir sensiblemente el trabajo necesario para realizar un trabajo. Por eso es interesante seguir el trabajo de algunos diseñadores para poder encontrar atajos.

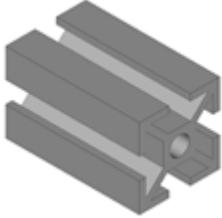
Si el trabajo de algún diseñador interesa particularmente se pueden clonar sus repositorios, así cada vez que se acceda a la página personal de GitHub aparecerán ahí, la plataforma avisa de si hay nuevas actualizaciones en estos repositorios y facilita el actualizar las carpetas clonadas.

[Escriba aquí]

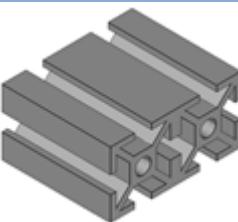
[Escriba aquí]

8. Presupuesto

8.1. Mediciones

| Ilustración | Nombre | medición | Unidades |
|---|--------------|----------|------------|
|  | Perfil 30x30 | 10,59 | [m] metros |

| nombre | Número de piezas | medición | unidades |
|----------------------|------------------|----------|----------|
| Perfil 30x30 L90mm | 4 | 0,36 | [m] |
| Perfil 30x30 L250mm | 2 | 0,50 | [m] |
| Perfil 30x30 L740mm | 2 | 1,48 | [m] |
| Perfil 30x30 L800mm | 4 | 3,20 | [m] |
| Perfil 30x30 L810mm | 1 | 0,81 | [m] |
| Perfil 30x30 L1060mm | 4 | 4,24 | [m] |
| TOTAL | | 10,59 | [m] |

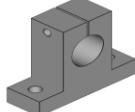
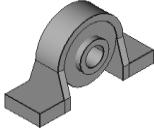
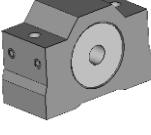
| Ilustración | Nombre | medición | Unidades |
|---|--------------|----------|------------|
|  | Perfil 30x60 | 2,80 | [m] metros |

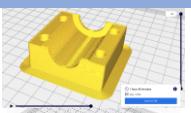
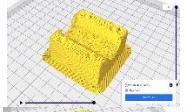
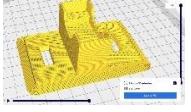
| nombre | Número de piezas | medición | unidades |
|----------------------|------------------|----------|----------|
| Perfil 30x60 L400mm | 2 | 0,80 | [m] |
| Perfil 30x60 L1000mm | 2 | 2,00 | [m] |
| TOTAL | | 2,80 | [m] |

| Ilustración | Nombre | medición | Unidades |
|---|------------------|----------|---------------|
|  | Husillo TR 8X1,5 | 1 | [Ud] unidades |

[Escriba aquí]

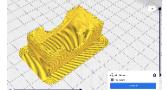
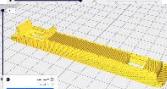
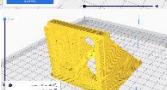
[Escriba aquí]

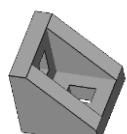
| Ilustración | Nombre | medición | Unidades |
|---|--|----------|---------------|
|  | Eje de acero D20mm | 2 | [ud] unidades |
|  | Soporte eje SH20/SK20 | 4 | [ud] unidades |
|  | Rodamiento 8mm con brida a 90° | 2 | [ud] unidades |
|  | Cojinete LME20UU | 4 | [ud] unidades |
|  | Tuerca Trapezoidal de plomo para husillo 8x1,5 R | 1 | [ud] unidades |
|  | Bobina de Filamento PLA 3D850 1.75mm | 276 | [g] gramos |

| Ilustración | nombre | Número de piezas | medición | unidades |
|---|-----------------------------------|------------------|----------|----------|
|  | Soporte cojinete LME20UU (top) | 4 | 23 | [g] |
|  | Soporte cojinete LME20UU (bottom) | 4 | 23 | [g] |
|  | Soporte Tuerca husillo (Top) | 1 | 31 | [g] |

[Escriba aquí]

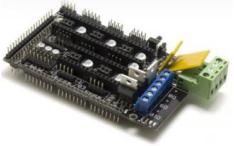
[Escriba aquí]

| Ilustración | nombre | Número de piezas | medición | unidades |
|---|---------------------------------|------------------|----------|----------|
|  | Soporte Tuerca husillo (button) | 1 | 10 | [g] |
|  | Soporte para fin de carrera | 2 | 6 | [g] |
|  | Soporte para el motor NEMA 23 | 1 | 39 | [g] |
| TOTAL | | | 276 | [g] |

| Ilustración | Nombre | medición | Unidades |
|---|---------------------------------|----------|---------------|
|  | Acoplamiento flexible 6.35/ 8mm | 1 | [ud] unidades |
|  | Escuadra 30x30 | 14 | [ud] unidades |
|  | Escuadra 30x60 | 8 | [ud] unidades |
|  | Motor paso a paso, Nema 23 | 1 | [ud] unidades |
|  | ARDUINO MEGA 2560 | 1 | [ud] unidades |
|  | Controlador POLOLU DRV8825 | 1 | [ud] unidades |

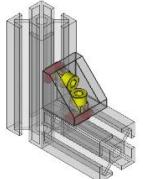
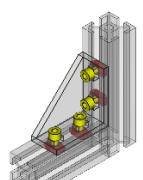
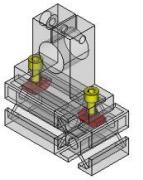
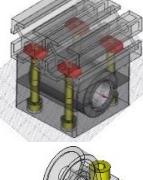
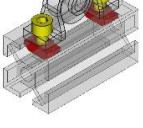
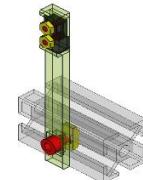
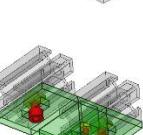
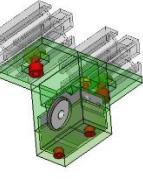
[Escriba aquí]

[Escriba aquí]

| Ilustración | Nombre | medición | Unidades |
|---|------------------------|----------|---------------|
|  | CNC shield RAMPS 1.4 | 1 | [ud] unidades |
|  | Pantalla LCD | 1 | [ud] unidades |
|  | Fin de carrera | 2 | [ud] unidades |
| | Tornillo DIN 912 M6x12 | 76 | [ud] unidades |
|  | Tornillo DIN 912 M6x40 | 16 | [ud] unidades |
| | Tornillo DIN 912 M4x40 | 2 | [ud] unidades |
| | Tornillo DIN 912 M2x12 | 4 | [ud] unidades |
|  | Tuerca Martillo M6 | 90 | [ud] unidades |
|  | Tuerca Hexagonal M2 | 2 | [ud] unidades |
| | Tuerca Hexagonal M4 | 2 | [ud] unidades |
| | Tuerca Hexagonal M6 | 2 | [ud] unidades |

[Escriba aquí]

[Escriba aquí]

| | Tornillo Modelo | Tornillo numero | Tuerca modelo | Tuerca numero | Ilustración de montaje |
|--|--------------------|--------------------|------------------|------------------|---|
| Escuadras 30x30 | M6x12 | 2 | M6 | 2 |  |
| Escuadras 30x60 | M6x12 | 4 | M6 | 4 |  |
| Soportes de ejes | M6x12 | 2 | M6 | 2 |  |
| Correderas | M6x40 | 4 | M6 | 4 |  |
| rodamientos | M6X12 | 2 | M6 | 2 |  |
| Soporte para los fines de carrera | M6x12 | 1 | M6 | 1 |  |
| | M2x12 | 2 | M2 | 2 |  |
| Soporte para la tuerca del husillo | M6x12 | 2 | M6 | 2 |  |
| | M4X40 | 2 | M4 | 2 |  |

[Escriba aquí]

[Escriba aquí]

| NOMBRE | MEDIDA | UNIDADES | PRECIO UNITARIO | PRECIO TOTAL |
|---|--------|---------------|-----------------|-----------------|
| Perfil 30x30 | 10,59 | [m] metros | 4,77 | 50,51 € |
| Perfil 30x60 | 2,80 | [m] metros | 10,45 | 29,26 € |
| Husillo TR 8mmx1,5mm | 1 | [ud] unidades | 2,64 | 2,64 € |
| Eje de acero D20mm | 1 | [m] metros | 8,47 | 8,47 € |
| Soporte eje SH20/SK20 | 4 | [ud] unidades | 2,34 | 9,36 € |
| Rodamiento 8mm con brida a 90° | 2 | [ud] unidades | 2,20 | 4,40 € |
| Cojinete LME20UU | 4 | [ud] unidades | 22,49 | 89,96 € |
| Tuerca Trapezoidal para husillo 8x1,5 R | 1 | [ud] unidades | 15,49 | 15,49 € |
| Bobina de Filamento PLA 3D850 1.75mm | 276 | [g] gramos | 0,02 | 4,56 € |
| Acoplamiento flexible 6.35/ 8mm | 1 | [ud] unidades | 5,16 | 5,16 € |
| Escuadra 30x30 | 14 | [ud] unidades | 0,60 | 8,40 € |
| Escuadra 30x60 | 8 | [ud] unidades | 1,68 | 13,44 € |
| Motor paso a paso, Nema 23 | 1 | [ud] unidades | 35,90 | 35,90 € |
| ARDUINO MEGA 2560 | 1 | [ud] unidades | 14,95 | 14,95 € |
| Controlador POLOLU DRV8825 | 1 | [ud] unidades | 2,33 | 2,33 € |
| CNC shield RAMPS 1.4 | 1 | [ud] unidades | 5,99 | 5,99 € |
| Pantalla LCD | 1 | [ud] unidades | 7,99 | 7,99 € |
| Fin de carrera | 2 | [ud] unidades | 0,7 | 1,40 € |
| Tornillo DIN 912 M6x12 | 76 | [ud] unidades | 0,39 | 29,64 € |
| Tornillo DIN 912 M6x40 | 16 | [ud] unidades | 0,39 | 6,24 € |
| Tornillo DIN 912 M4x40 | 2 | [ud] unidades | 0,39 | 0,78 € |
| Tornillo DIN 912 M2x12 | 4 | [ud] unidades | 0,39 | 1,56 € |
| Tuerca Martillo M6 | 90 | [ud] unidades | 0,116 | 10,44 € |
| Tuerca Hexagonal M2 | 2 | [ud] unidades | 0,067 | 0,13 € |
| Tuerca Hexagonal M4 | 2 | [ud] unidades | 0,067 | 0,13 € |
| Tuerca Hexagonal M6 | 2 | [ud] unidades | 0,067 | 0,13 € |
| TOTAL BRUTO | | | | 359,28 € |
| 21% IVA | | | | 75,45 € |
| TOTAL NETO | | | | 434,72 € |

[Escriba aquí]

[Escriba aquí]

[Escriba aquí]

[Escriba aquí]

ANEXOS

ANEXO 1

PLANOS 60

ANEXO 2

Lista de piezas de la maquina con una cara movil 72

ANEXO 3

Lista de piezas de la maquina con dos caras moviles..... 75

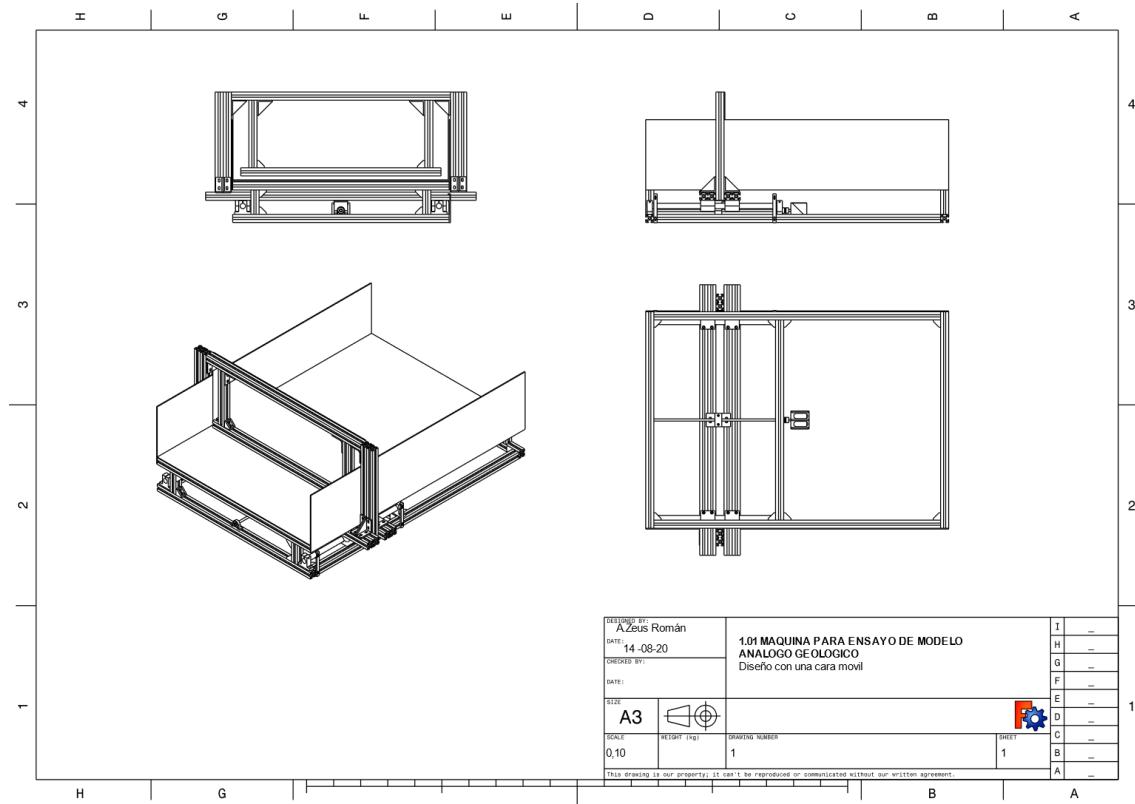
1. PLANOS

| | |
|--|----|
| Plano 1: MAQUINA PARA ENSAYO: Diseño con una cara móvil..... | 62 |
| Plano 2: MAQUINA PARA ENSAYO: Diseño con dos caras móviles | 62 |
| Plano 3: PARTES DEL DISEÑO: Base | 63 |
| Plano 4: PARTES DEL DISEÑO: Sistema de apoyo | 63 |
| Plano 5: PARTES DEL DISEÑO: Sistema de transmisión | 64 |
| Plano 6: PARTES DEL DISEÑO: Puente móvil | 64 |
| Plano 7: PIEZA IMPRESA: Soporte corredera | 65 |
| Plano 8: PIEZA IMPRESA: Soporte para tuerca trapezoidal para husillo | 65 |
| Plano 9: PIEZA IMPRESA: Soporte para final de carrera | 66 |
| Plano 10: PIEZA IMPRESA: Soporte para motor NEMA 23 | 66 |
| Plano 11: ETAPAS DEL MONTAJE: Montaje de la base | 67 |
| Plano 12: ETAPAS DEL MONTAJE: Inserción de los sistemas de apoyo y transmisión | 67 |
| Plano 13: ETAPAS DEL MONTAJE: Inserción del puente móvil | 68 |
| Plano 14: ETAPAS DEL MONTAJE: Finalización de la base..... | 68 |
| Plano 15: ETAPAS DEL MONTAJE: Inserción de los paneles laterales..... | 69 |
| Plano 16: UNIÓN ROSCADA: Escuadra 30x30 | 69 |
| Plano 17: UNIÓN ROSCADA: Escuadra 60x60 | 70 |
| Plano 18: UNIÓN ROSCADA: Soporte eje de acero | 70 |
| Plano 19: UNIÓN ROSCADA: Soporte para cojinete LME20UU | 71 |
| Plano 20: UNIÓN ROSCADA: Soporte para tuerca del husillo..... | 71 |
| Plano 21: UNIÓN ROSCADA: Soporte para fin de carrera | 72 |

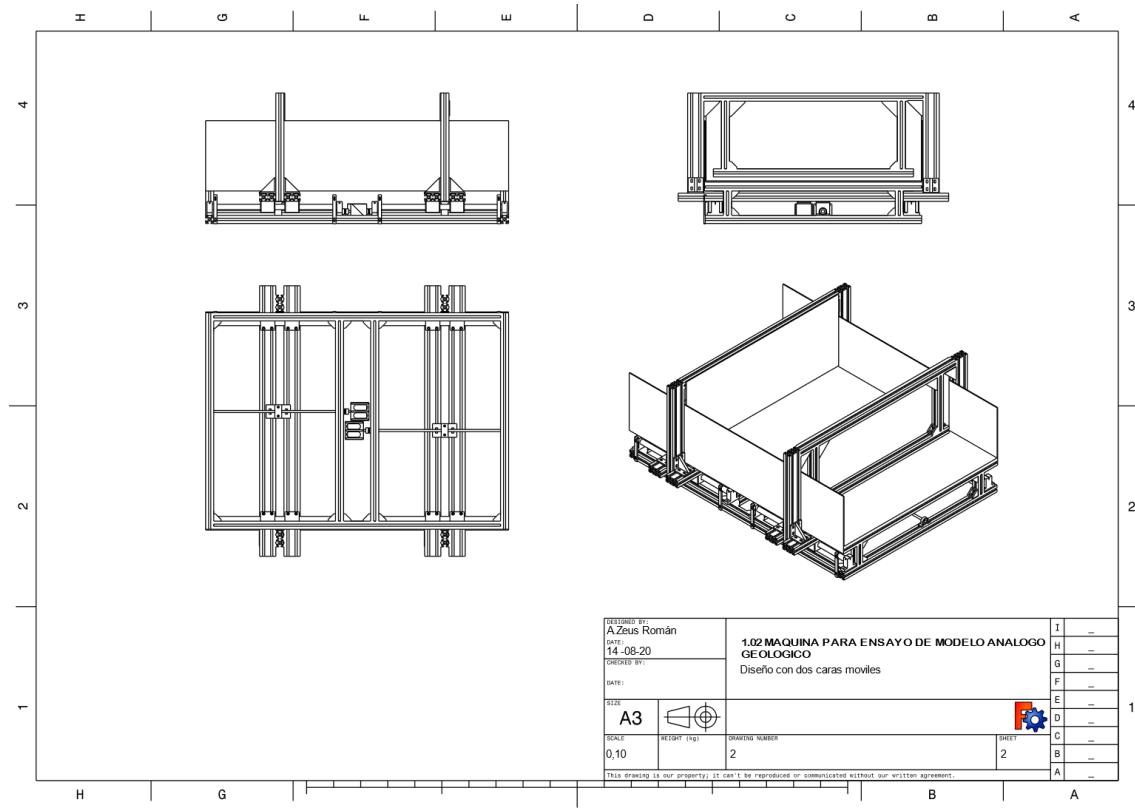
[Escriba aquí]

[Escriba aquí]

Plano 1: MAQUINA PARA ENSAYO: Diseño con una cara móvil



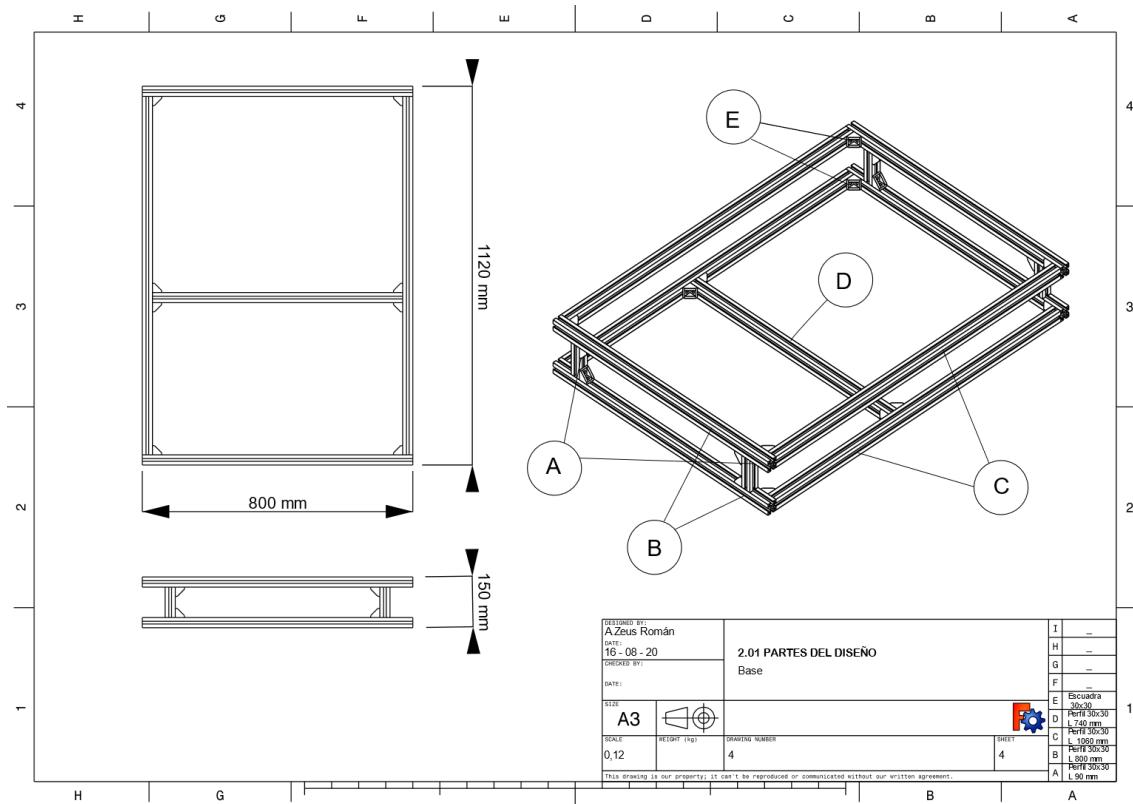
Plano 2: MAQUINA PARA ENSAYO: Diseño con dos caras móviles



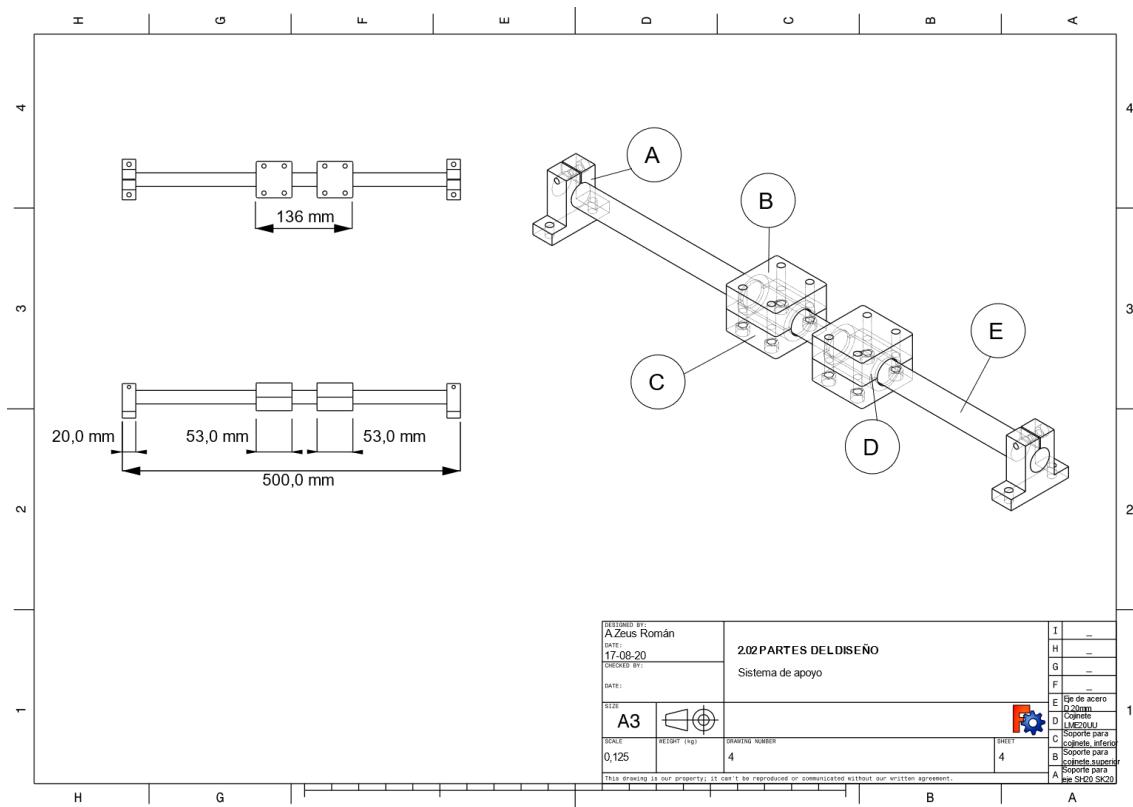
[Escriba aquí]

[Escriba aquí]

Plano 3: PARTES DEL DISEÑO: Base



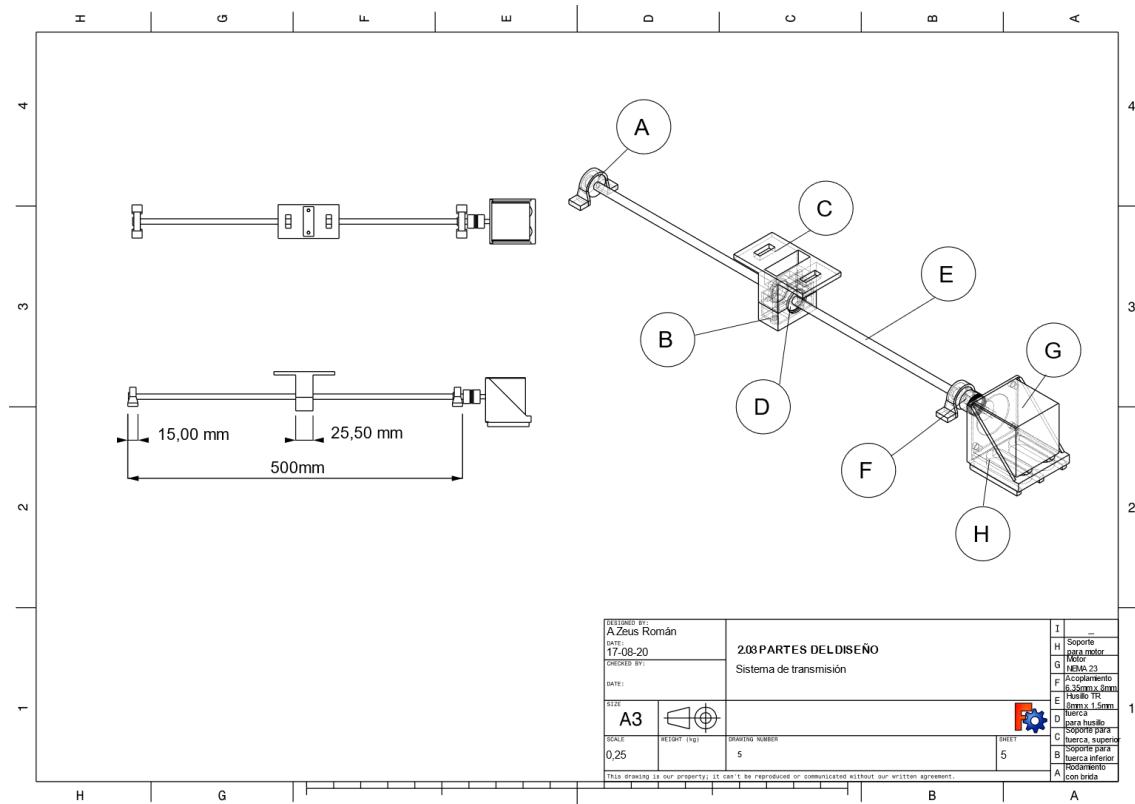
Plano 4: PARTES DEL DISEÑO: Sistema de apoyo



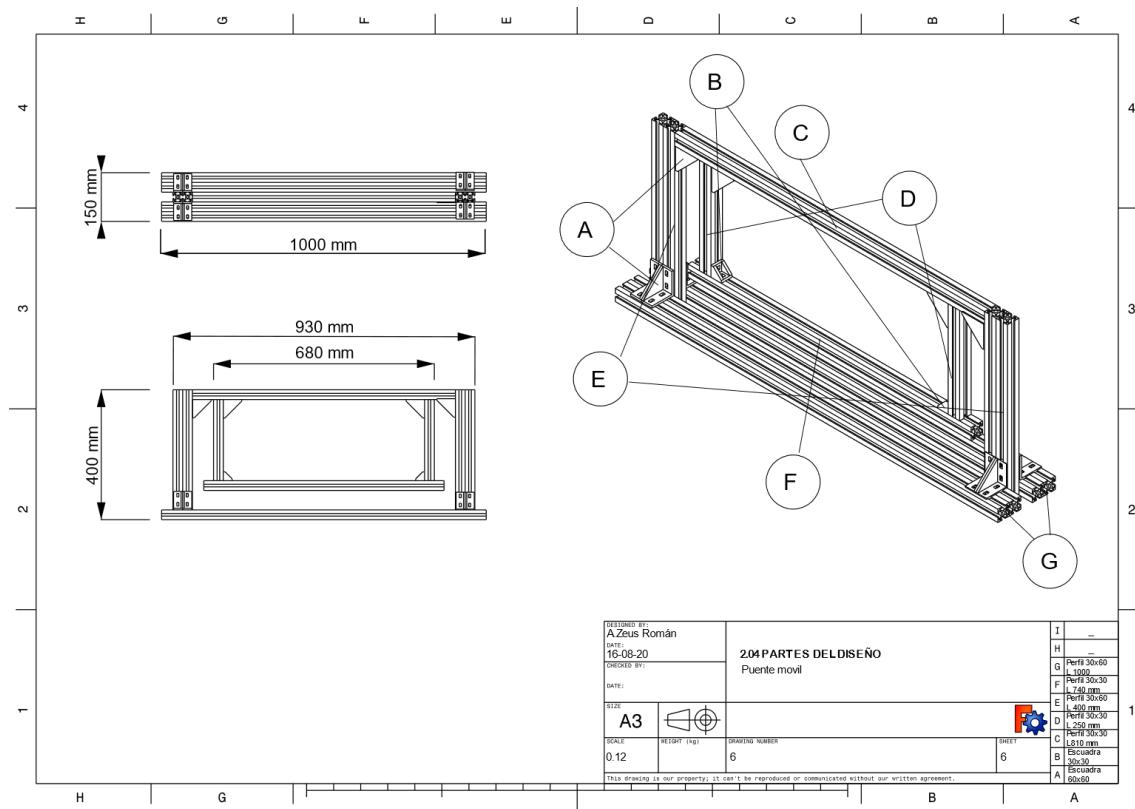
[Escriba aquí]

[Escriba aquí]

Plano 5: PARTES DEL DISEÑO: Sistema de transmisión



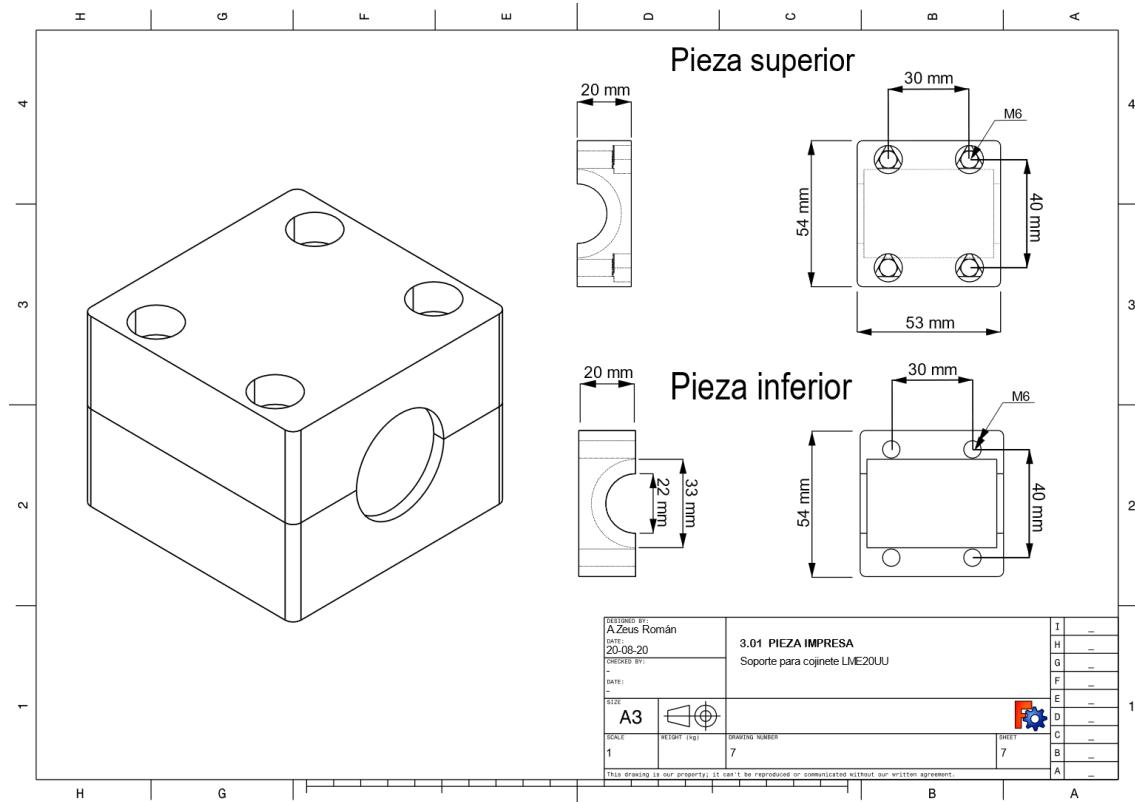
Plano 6: PARTES DEL DISEÑO: Puente móvil



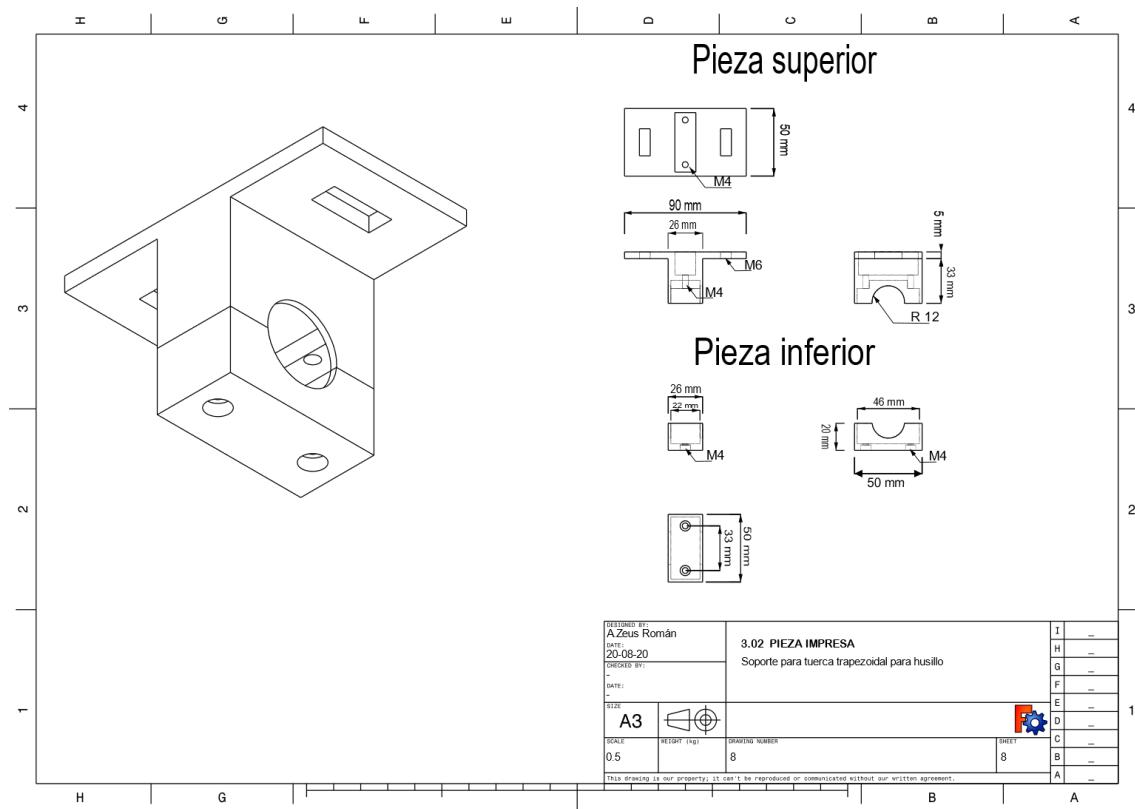
[Escriba aquí]

[Escriba aquí]

Plano 7: PIEZA IMPRESA: Soporte corredera



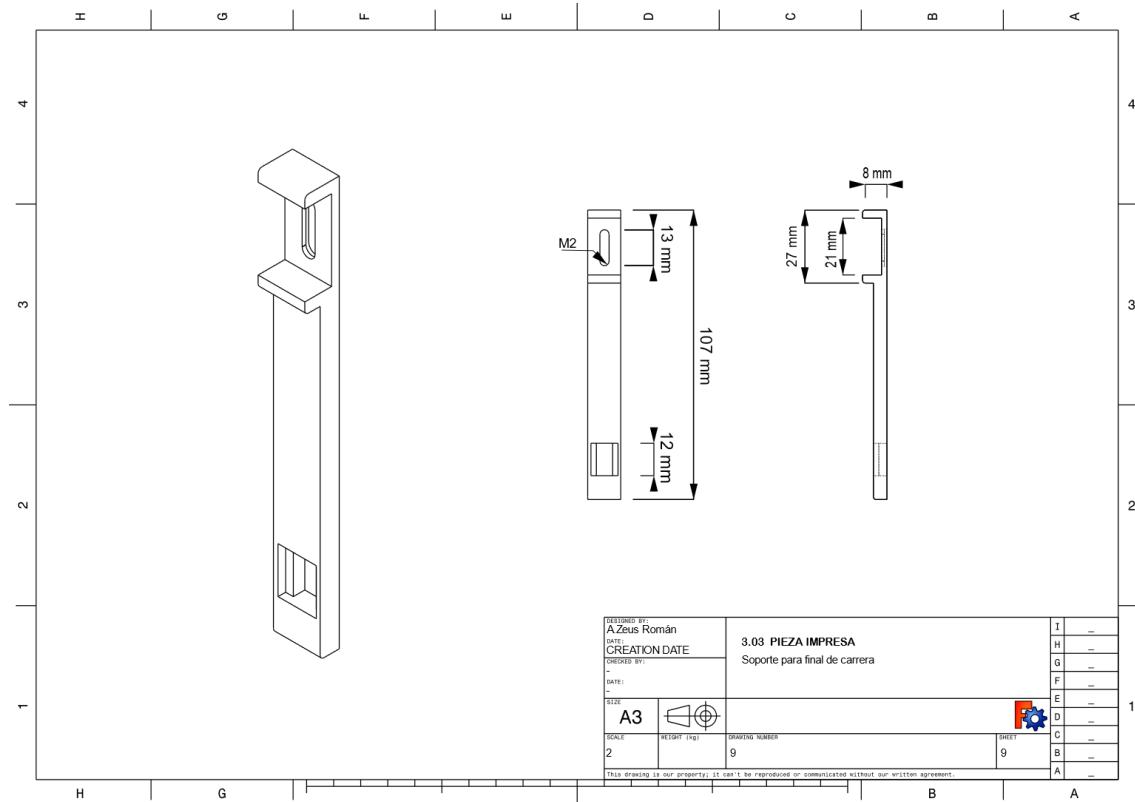
Plano 8: PIEZA IMPRESA: Soporte para tuerca trapezoidal para husillo



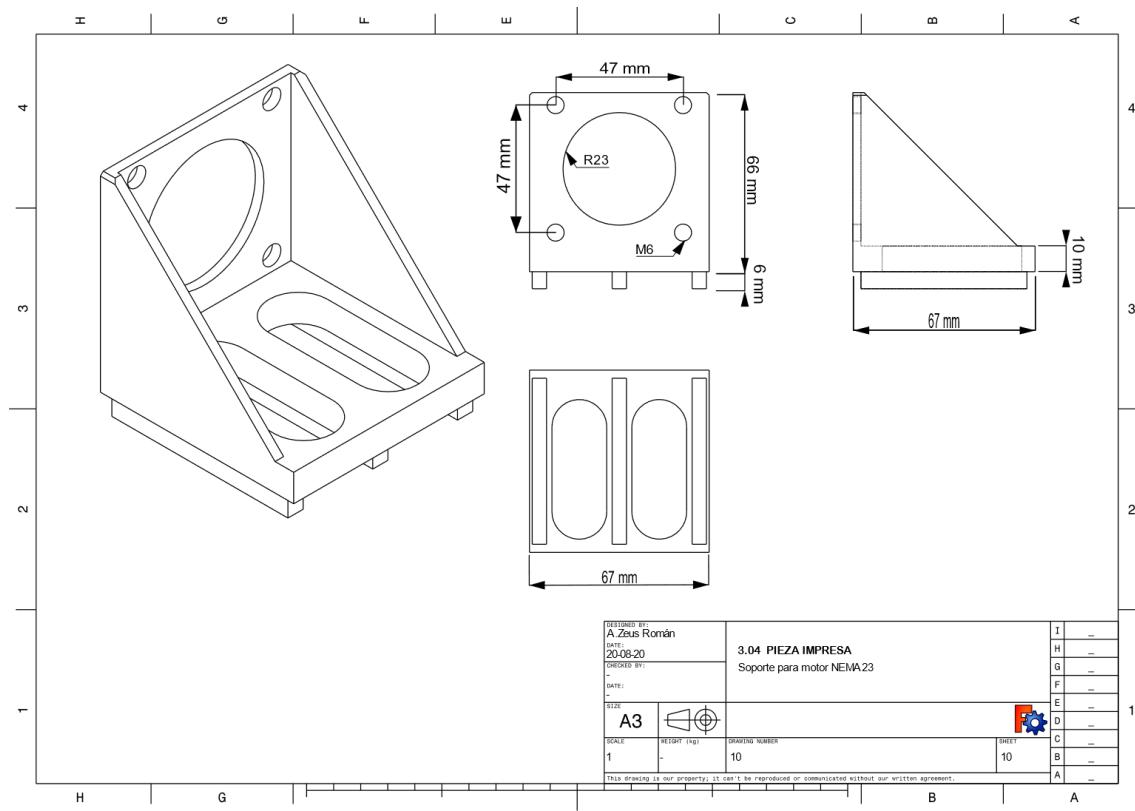
[Escriba aquí]

[Escriba aquí]

Plano 9: PIEZA IMPRESA: Soporte para final de carrera



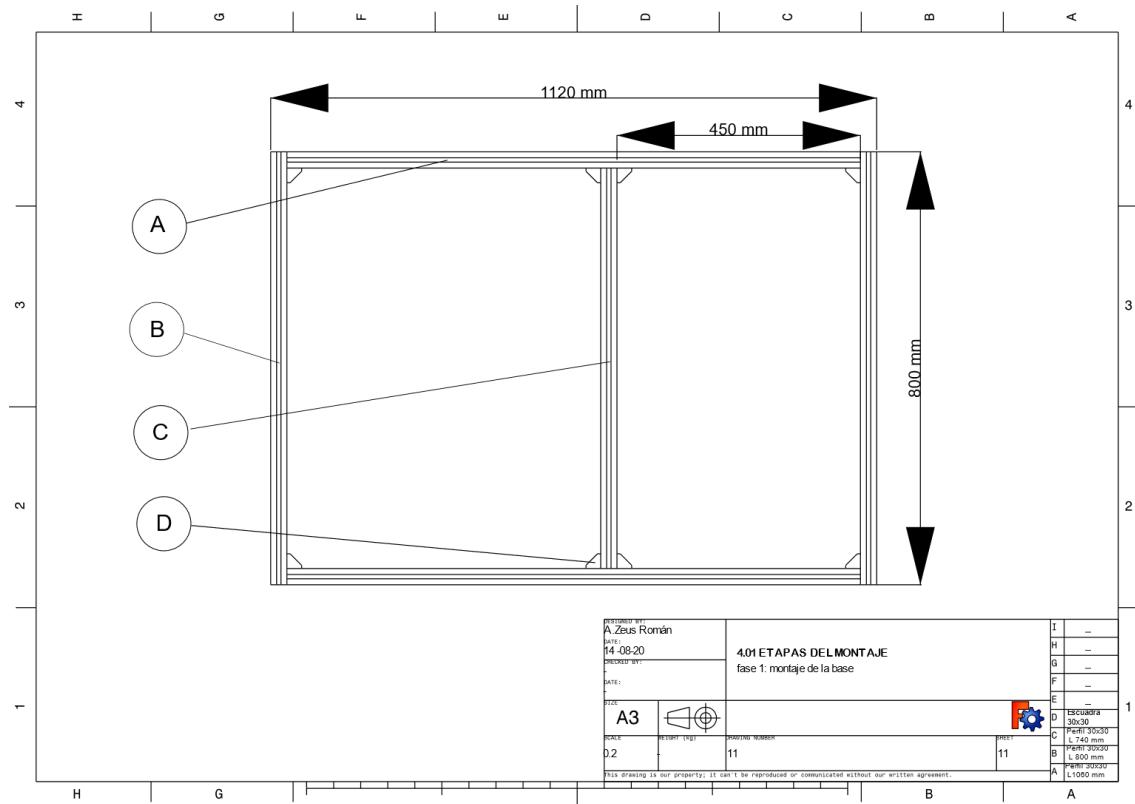
Plano 10: PIEZA IMPRESA: Soporte para motor NEMA 23



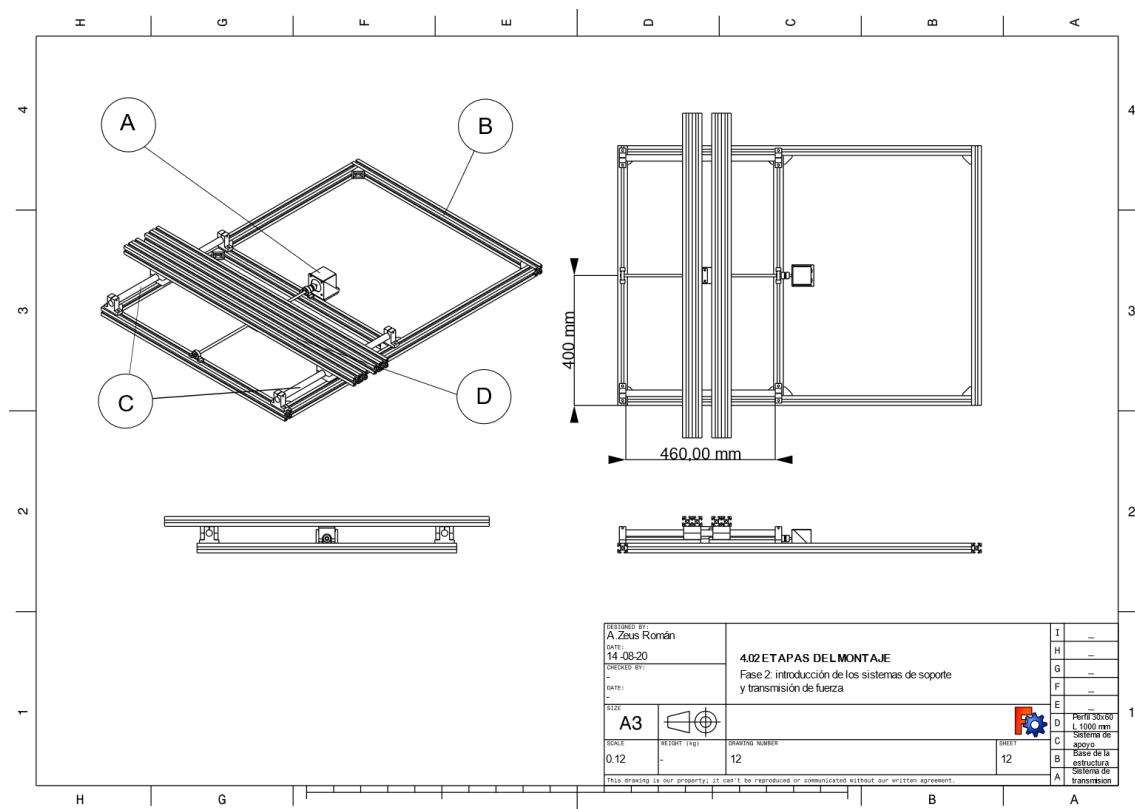
[Escriba aquí]

[Escriba aquí]

Plano 11: ETAPAS DEL MONTAJE: Montaje de la base



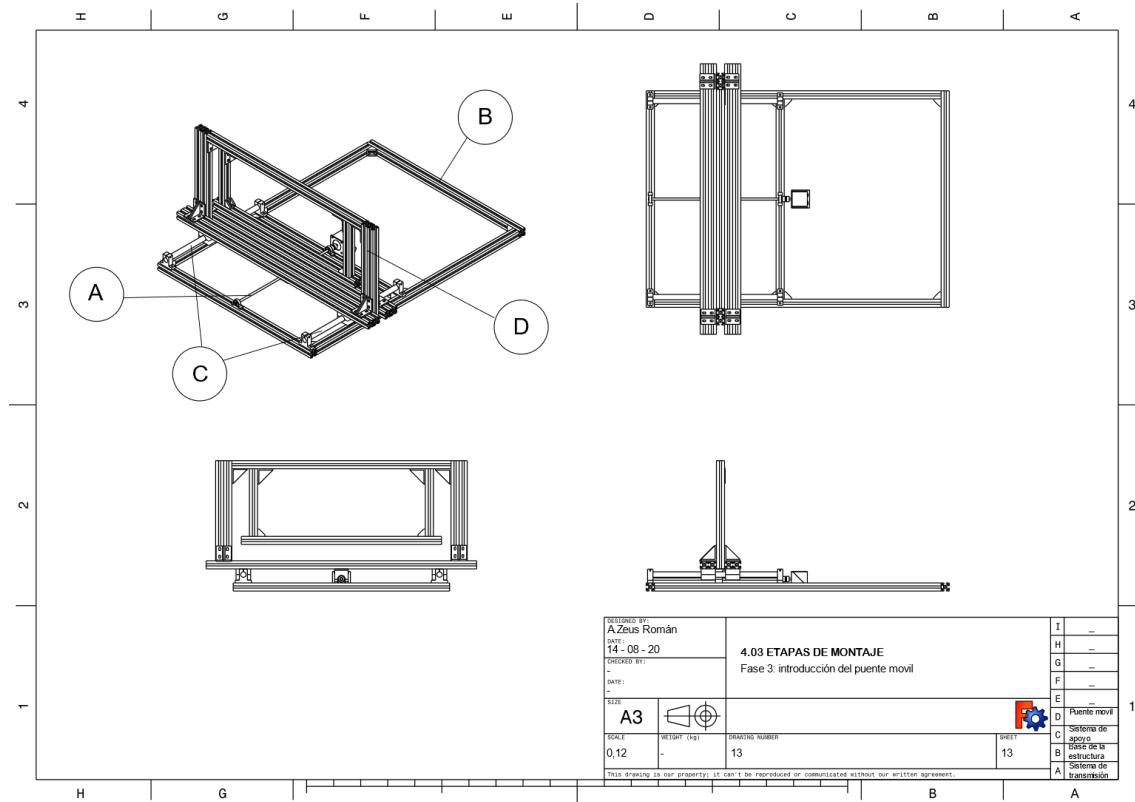
Plano 12: ETAPAS DEL MONTAJE: Inserción de los sistemas de apoyo y transmisión



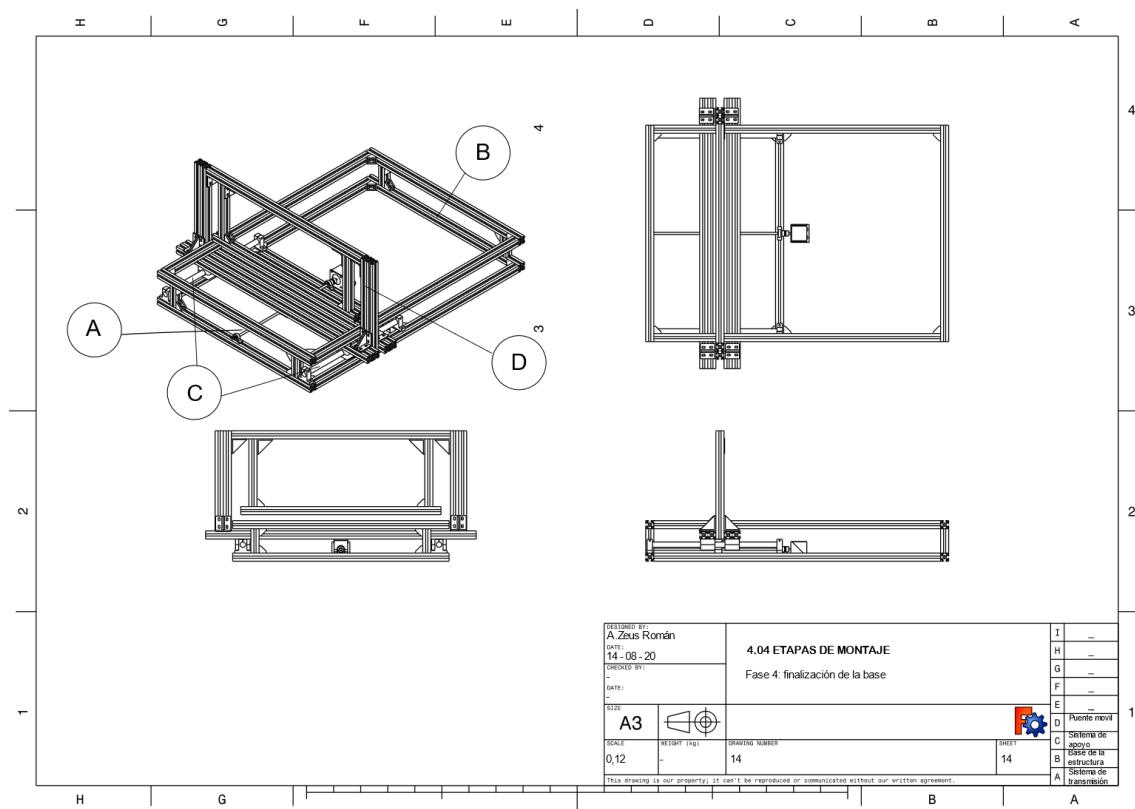
[Escriba aquí]

[Escriba aquí]

Plano 13: ETAPAS DEL MONTAJE: Inserción del puente móvil



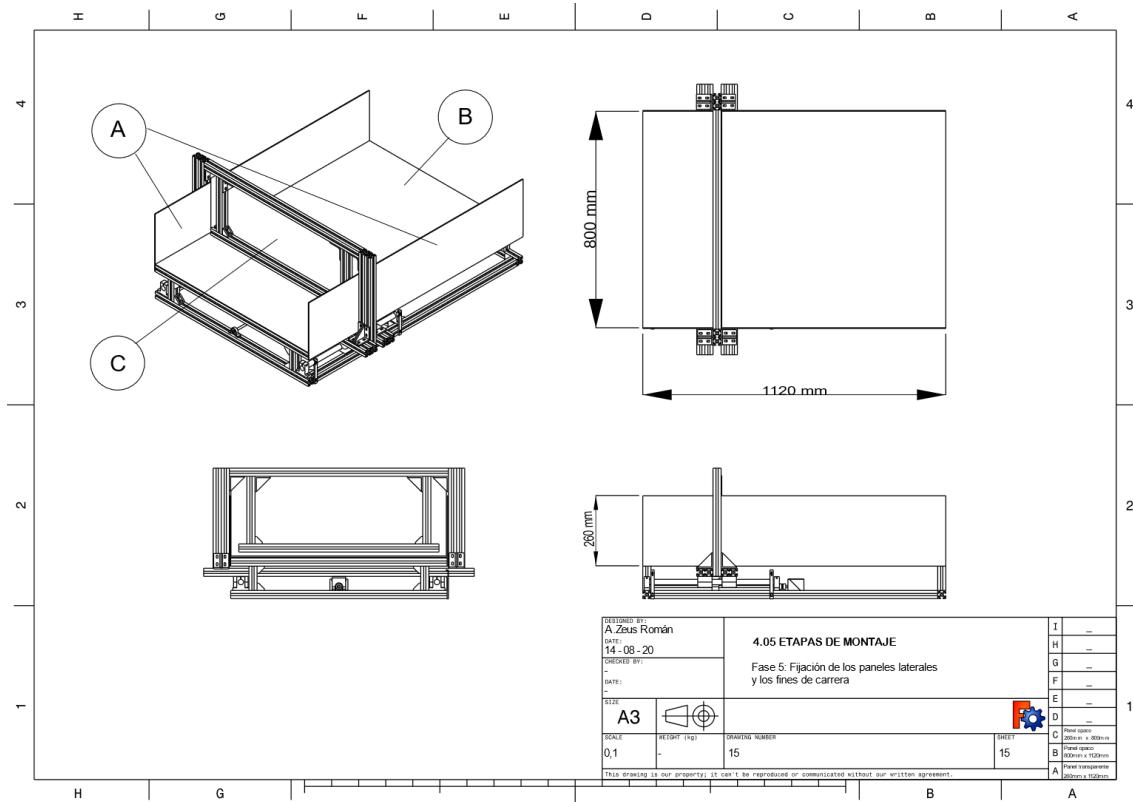
Plano 14: ETAPAS DEL MONTAJE: Finalización de la base



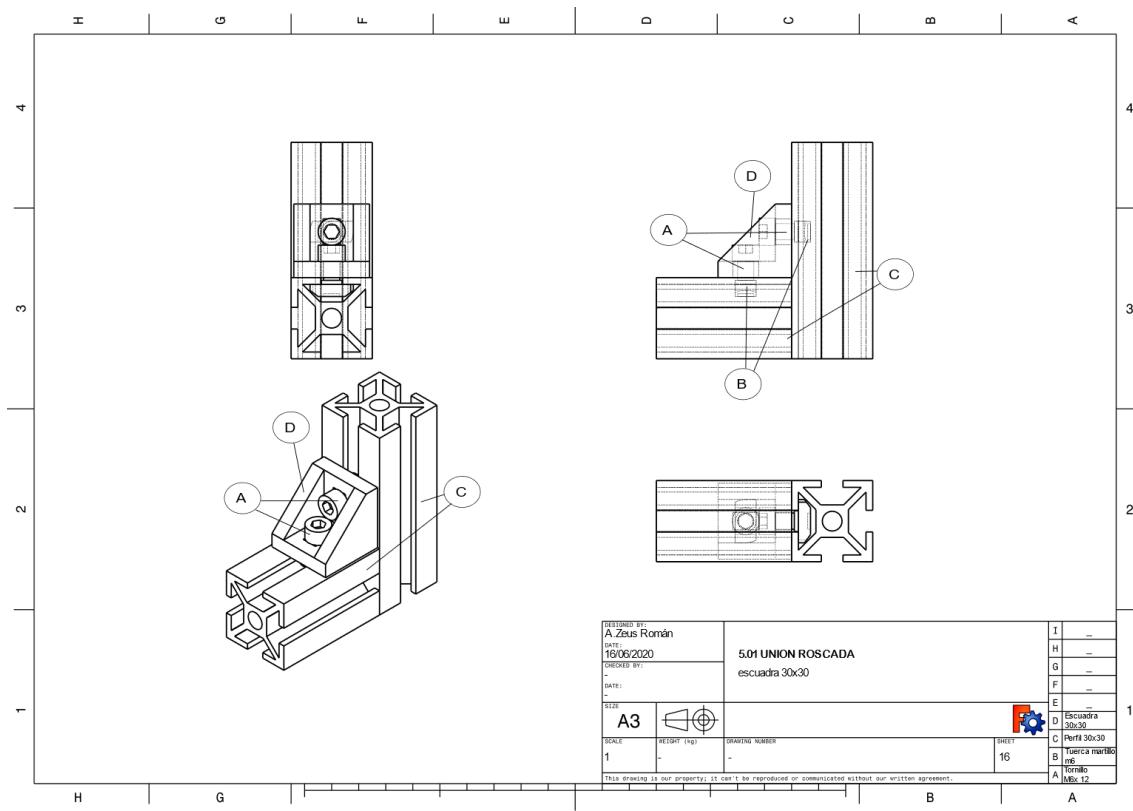
[Escriba aquí]

[Escriba aquí]

Plano 15: ETAPAS DEL MONTAJE: Inserción de los paneles laterales



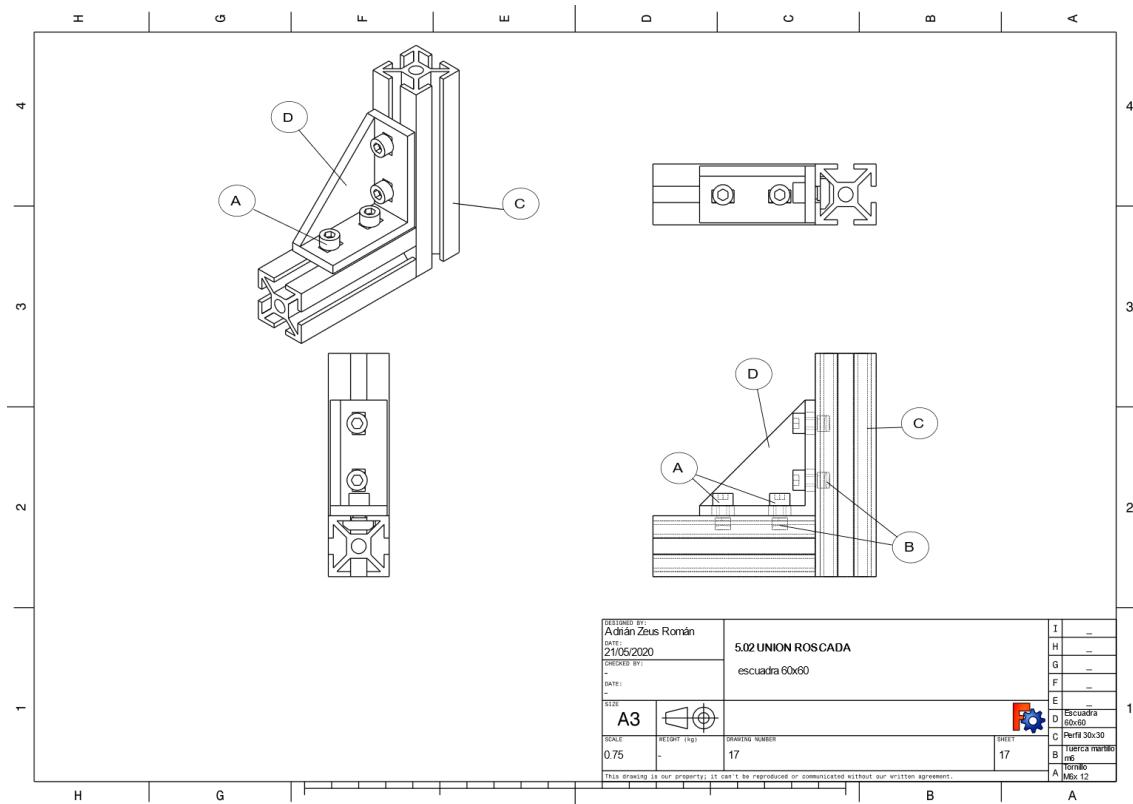
Plano 16: UNIÓN ROSCADA: Escuadra 30x30



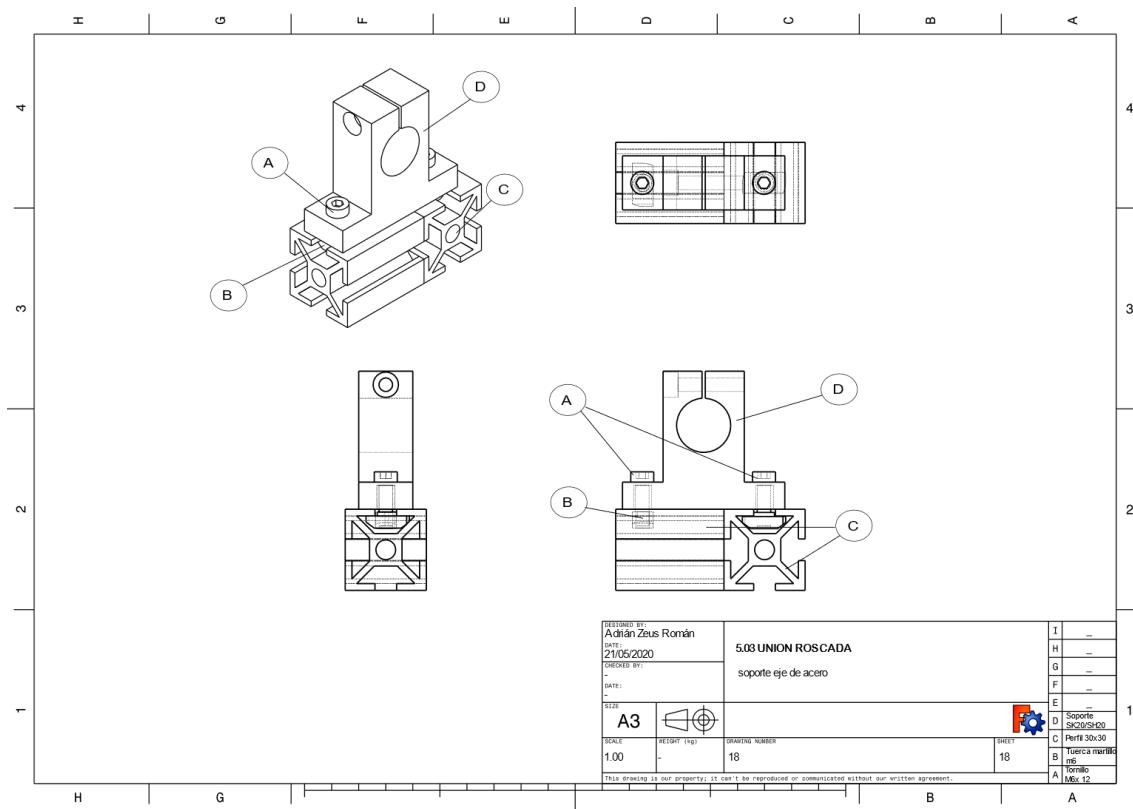
[Escriba aquí]

[Escriba aquí]

Plano 17: UNIÓN ROSCADA: Escuadra 60x60



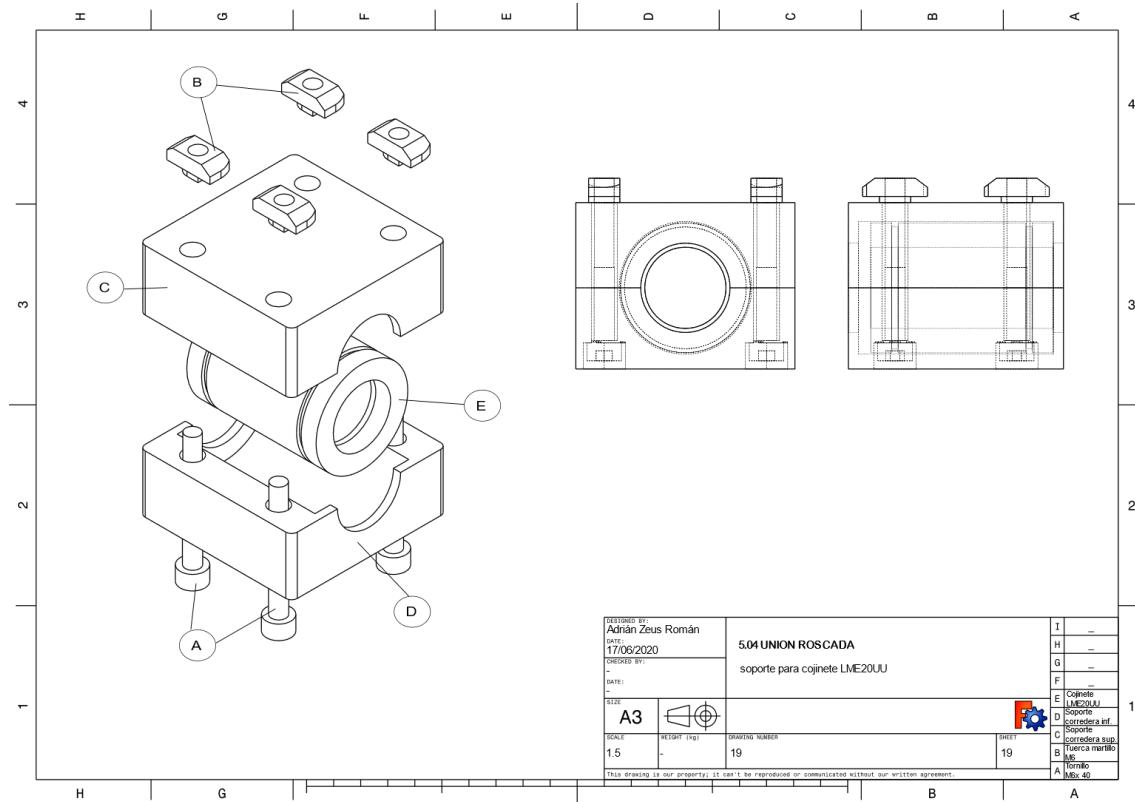
Plano 18: UNIÓN ROSCADA: Soporte eje de acero



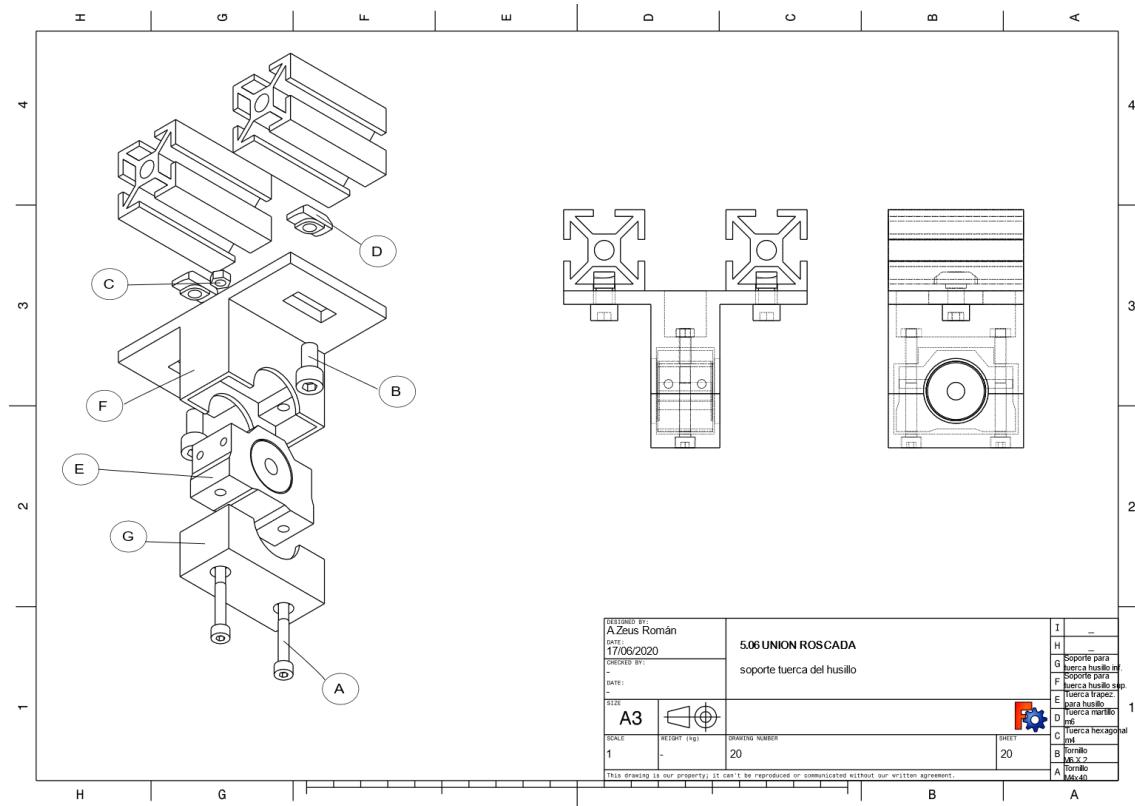
[Escriba aquí]

[Escriba aquí]

Plano 19: UNIÓN ROSCADA: Soporte para cojinete LME20UU



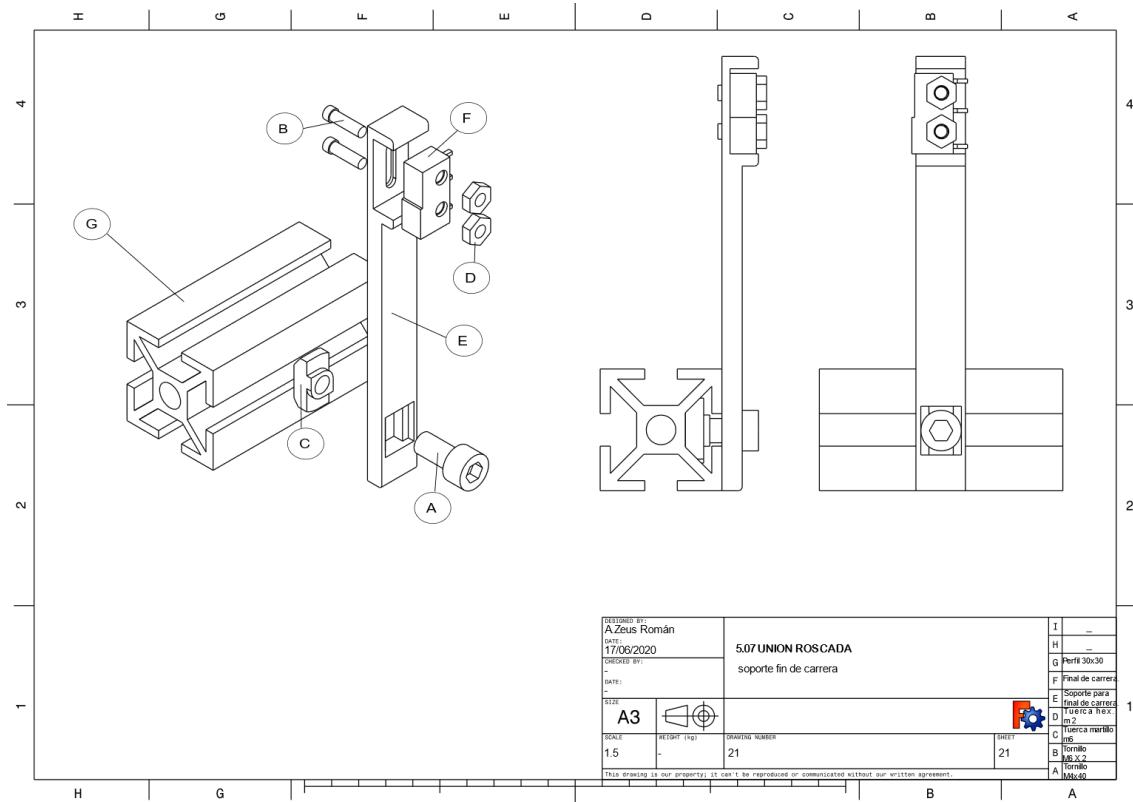
Plano 20: UNIÓN ROSCADA: Soporte para tuerca del husillo



[Escriba aquí]

[Escriba aquí]

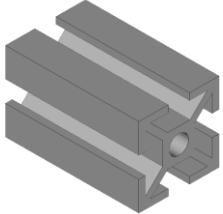
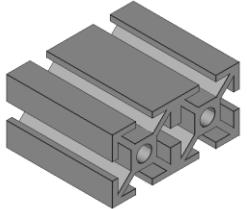
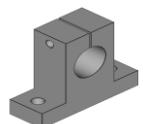
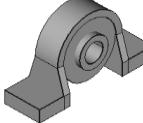
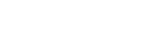
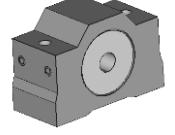
Plano 21: UNIÓN ROSCADA: Soporte para fin de carrera



[Escriba aquí]

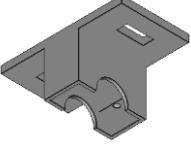
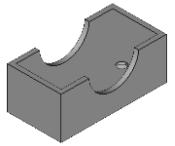
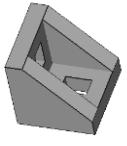
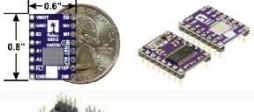
[Escriba aquí]

2. LISTA DE PIEZAS DE LA MAQUINA CON UNA CARA MOVIL

| Ilustración | nombre | Número de piezas | proveedor | Enlace de compra |
|---|--|------------------|-----------|---|
|  | Perfil 30x30 L90mm | 4 | motedis | https://www.motedis.es/shop/Perfil-Ranurado/Perfil-30-Tipo-B-ranura-8/Perfil-30x30-Tipo-B-ran-8::99999432.html |
| | Perfil 30x30 L250mm | 2 | | |
| | Perfil 30x30 L740mm | 2 | | |
| | Perfil 30x30 L800mm | 4 | | |
| | Perfil 30x30 L810mm | 1 | | |
| | Perfil 30x30 L1060mm | 4 | | |
|  | Perfil 30x60 L400mm | 2 | | https://www.motedis.es/shop/Perfil-Ranurado/Perfil-30-Tipo-B-ranura-8/Perfil-30x60-Tipo-B-ran-8::99999436.html |
|  | Perfil 30x60 L1000mm | 2 | | |
|  | Eje acero D20mm L500mm | 2 | motedis | https://www.motedis.es/shop/Modulos/Cojinete-del-bola-20-Forma-de-pe/Eje-acero-de-precision-%D820mm::999996499.html |
|  | Husillo TR 8mmx1,5mm | 1 | motedis | https://www.motedis.es/shop/Dinamica-Unidad-lineal/Husillo-trapezoidal-accesorios/Husillo-Trapezoidal/Acero-Husillo-de-rosca-trapezoidal/Husillo-de-rosca-trapezoidal-RPTS-derecha-TR-8x15-L%3D500mm::999993952.html |
|  | Soporte eje SH20/SK20 | 4 | motedis | https://www.motedis.es/shop/Modulos/Cojinete-del-bola-20-Forma-de-pe/Soportes-para-ejes-SH20-SK20::999991255.html |
|  | Rodamiento 8mm con brida a 90° | 2 | motedis | https://www.motedis.es/shop/Dinamica-Unidad-lineal/Rodamiento-con-brida-90G-KP/Rodamiento-con-brida-90% B0-8mm-KP08::999995357.html |
|  | Cojinete LME20UU | 4 | motedis | https://www.motedis.es/shop/Dinamica-Unidad-lineal/Cojinete-lineal/Cojinete-Lineal-Estandar-sin-Alojamiento/Cojinete-lineal-20mm-LME20UU::999991278.html |
|  | Tuerca Trapezoidal de plomo para husillo 8x1,5 R | 1 | motedis | https://www.motedis.es/shop/Dinamica-Unidad-lineal/Husillo-trapezoidal-accesorios/Tuerca-Trapezoidal/Tuerca-de-husillo-trapezoidal-Bronce-rojo-con-caja-de-Aluminio/Trapezoidal-plomo-tuerca-8x15-R-bronce-rojo-con-caja-de-aluminio::999993741.html |

[Escriba aquí]

[Escriba aquí]

| Ilustración | nombre | Número de piezas | proveedor | Enlace de compra |
|---|-----------------------------------|------------------|-----------|---|
|  | Soporte cojinete LME20UU (top) | 4 | - | IMPRESO 3D |
|  | Soporte cojinete LME20UU (bottom) | 4 | - | IMPRESO 3D |
|  | Soporte Tuerca husillo (Top) | 1 | - | IMPRESO 3D |
|  | Soporte Tuerca husillo (bottom) | 1 | - | IMPRESO 3D |
|  | Escuadra 30x30 | 14 | motedis | https://www.motedis.es/shop/Accesorios-para-Perfil-Ranurado/Accesorios-Perfil-20-Tipo-B-Ranura-6/Conectores-y-angulos-para-ranura-6-tipo-B/Singleparts/Bracket/Escuadra-30-tipo-B::999991057.html |
|  | Escuadra 30x60 | 8 | motedis | https://www.motedis.es/shop/Accesorios-para-Perfil-Ranurado/Accesorios-Perfil-20-Tipo-B-Ranura-6/Conectores-y-angulos-para-ranura-6-tipo-B/Singleparts/Bracket/Escuadra-30x60-tipo-B::99999332.html |
|  | Acoplamiento flexible 6.35/8mm | 1 | motedis | https://www.motedis.es/shop/Dinamica-Unidad-lineal/Acoplamiento/Acoplamiento-RB/Ejes-acoplamiento-flexibles-Mot-D20L25-635-8mm::999994106.html |
|  | Motor paso a paso, Nema 23 | 1 | bricogeek | https://tienda.bricogeek.com/motores-paso-a-paso/422-motor-paso-a-paso-9-kg-cm.html |
|  | ARDUINO MEGA 2560 | 1 | hta3d | https://www.hta3d.com/es/mega-2560-r3-compatible-16u2 |
|  | Controlador POLOLU DRV8825 | 1 | hta3d | https://www.hta3d.com/es/drv8825 |
|  | CNC shield RAMPS 1.4 | 1 | hta3d | https://www.hta3d.com/es/ramps-1-4 |

[Escriba aquí]

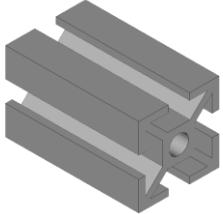
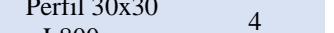
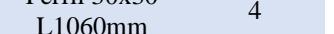
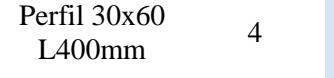
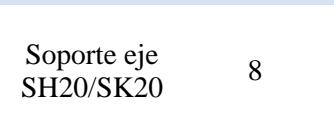
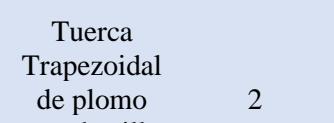
[Escriba aquí]

| Ilustración | nombre | Número de piezas | proveedor | Enlace de compra |
|---|--|------------------|-----------|--|
|  | Pantalla LCD | 1 | hta3d | https://www.hta3d.com/es/pantalla-2004-lcd-smart-controller |
|  | Fin de carrera | 2 | hta3d | https://www.hta3d.com/es/final-carrera-mecanico-kw12-3 |
|  | Soporte para fin de carrera | 2 | - | IMPRESO 3D |
|  | Tornillo DIN 912 M6x12 Tornillo DIN 912 M6x40 Tornillo DIN 912 M2x12 | | motedis | https://www.motedis.es/shop/Basicos-Mecanica/Basico-Basicos/DIN-pieza-normalizada/Tornillos-DIN/Tornillo-DIN-912::999991121.html |
|  | Tuerca Martillo M6 | | motedis | https://www.motedis.es/shop/Accesorios-para-Perfil-Ranurado/Accesorios-Perfil-30-Tipo-B-Ranura-8/Sliding-nuts-suitable-for-Nut-8-B-type/Tuerca-martillo-ran-8-tipo-B-M4-M5-M6::999998941.html https://www.amazon.es/Hexagonal-BiuZi-Sujetador-Galvanizado-Opcional/dp/B07WVW7SNJ/ref=sr_1_5?adgrpid=56245435815&dchild=1&gclid=CjwKCAjw57b3BRBI_EiwA1ImytoWNUGeW0C05f9XdYYopxTdmjMwnkR2RhCUCm_iKUvsj2iZ8_mHfHBoC3V0QAvD_BwE&hvadid=275347028870&hvdev=c&hvlocphy=9047044&hvnetw=g&hvqmt=e&hvran_d=551291719288938984&hvtargid=kwd-327827106366&hydadcr=1431_1736081&keywords=tuerca+m2&qid=1592722156&sr=8-5&tag=hydes-21 |
|  | Tuerca Hexagonal M2 | | amazon | |

[Escriba aquí]

[Escriba aquí]

3. LISTA DE PIEZAS DEL ENSAYO DE LA MAQUINA CON DOS CARAS MOVILES

| Ilustración | nombre | Número de piezas | proveedor | Enlace de compra |
|---|--|------------------|-----------|---|
|  | Perfil 30x30 L90mm | 4 | motedis | https://www.motedis.es/shop/Perfil-Ranurado/Perfil-30-Tipo-B-ranura-8/Perfil-30x30-Tipo-B-ran-8::99999432.html |
|  | Perfil 30x30 L250mm | 4 | | |
|  | Perfil 30x30 L740mm | 4 | | |
|  | Perfil 30x30 L800mm | 4 | | |
|  | Perfil 30x30 L810mm | 2 | | |
|  | Perfil 30x30 L1060mm | 4 | | |
|  | Perfil 30x60 L400mm | 4 | | https://www.motedis.es/shop/Perfil-Ranurado/Perfil-30-Tipo-B-ranura-8/Perfil-30x60-Tipo-B-ran-8::99999436.html |
|  | Perfil 30x60 L1000mm | 4 | | |
|  | Eje acero D20mm L500mm | 4 | motedis | https://www.motedis.es/shop/Modulos/Cojinete-del-bola-20-Forma-de-pe/Eje-acero-de-precision-%D820mm::999996499.html |
|  | Husillo TR 8mmx1,5mm | 2 | motedis | https://www.motedis.es/shop/Dinamica-Unidad-lineal/Husillo-trapezoidal-accesorios/Husillo-Trapezoidal/Acero-Husillo-de-roscas-trapezoidal/Husillo-de-roscas-trapezoidal-RPTS-derecha-TR-8x15-L%3D500mm::999993952.html |
|  | Soporte eje SH20/SK20 | 8 | motedis | https://www.motedis.es/shop/Modulos/Cojinete-del-bola-20-Forma-de-pe/Soportes-para-ejes-SH20-SK20::999991255.html |
|  | Rodamiento 8mm con brida a 90° | 4 | motedis | https://www.motedis.es/shop/Dinamica-Unidad-lineal/Rodamiento-con-brida-90G-KP/Rodamiento-con-brida-90% B0-8mm-KP08::999995357.html |
|  | Cojinete LME20UU | 8 | motedis | https://www.motedis.es/shop/Dinamica-Unidad-lineal/Cojinete-lineal/Cojinete-Lineal-Estandar-sin-Alojamiento/Cojinete-lineal-20mm-LME20UU::999991278.html |
|  | Tuerca Trapezoidal de plomo para husillo 8x1,5 R | 2 | motedis | https://www.motedis.es/shop/Dinamica-Unidad-lineal/Husillo-trapezoidal-accesorios/Tuerca-Trapezoidal/Tuerca-de-husillo-trapezoidal-Bronce-rojo-con-caja-de-Aluminio/Trapezoidal-plomo-tuerca-8x15-R-bronce-rojo-con-caja-de-aluminio::999993741.html |

[Escriba aquí]

[Escriba aquí]

| Ilustración | nombre | Número de piezas | proveedor | Enlace de compra |
|-------------|-----------------------------------|------------------|-----------|---|
| | Soporte cojinete LME20UU (top) | 8 | - | IMPRESO 3D |
| | Soporte cojinete LME20UU (bottom) | 8 | - | IMPRESO 3D |
| | Soporte Tuerca husillo (Top) | 2 | - | IMPRESO 3D |
| | Soporte Tuerca husillo (bottom) | 2 | - | IMPRESO 3D |
| | Escuadra 30x30 | 18 | motedis | https://www.motedis.es/shop/Accesorios-para-Perfil-Ranurado/Accesorios-Perfil-20-Tipo-B-Ranura-6/Conectores-y-angulos-para-ranura-6-tipo-B/Singleparts/Bracket/Escuadra-30-tipo-B::999991057.html |
| | Escuadra 30x60 | 16 | motedis | https://www.motedis.es/shop/Accesorios-para-Perfil-Ranurado/Accesorios-Perfil-20-Tipo-B-Ranura-6/Conectores-y-angulos-para-ranura-6-tipo-B/Singleparts/Bracket/Escuadra-30x60-tipo-B::99999332.html |
| | Acoplamiento flexible 6.35/8mm | 2 | motedis | https://www.motedis.es/shop/Dinamica-Unidad-lineal/Acoplamiento/Acoplamiento-RB/Ejes-acoplamiento-flexibles-Mot-D20L25-635-8mm::999994106.html |
| | Motor paso a paso, Nema 23 | 2 | bricogeek | https://tienda.bricogeek.com/motores-paso-a-paso/422-motor-paso-a-paso-9-kg-cm.html |
| | ARDUINO MEGA 2560 | 1 | hta3d | https://www.hta3d.com/es/mega-2560-r3-compatible-16u2 |
| | Controlador POLOLU DRV8825 | 1 | hta3d | https://www.hta3d.com/es/drv8825 |
| | CNC shield RAMPS 1.4 | 1 | hta3d | https://www.hta3d.com/es/ramps-1-4 |

[Escriba aquí]

[Escriba aquí]

| Ilustración | nombre | Número de piezas | proveedor | Enlace de compra |
|---|--|------------------|-----------|--|
|  | Pantalla LCD | 1 | hta3d | https://www.hta3d.com/es/pantalla-2004-lcd-smart-controller |
|  | Fin de carrera | 4 | hta3d | https://www.hta3d.com/es/final-carrera-mecanico-kw12-3 |
|  | Soporte para fin de carrera | 4 | - | IMPRESO 3D |
|  | Tornillo DIN 912 M6x12 Tornillo DIN 912 M6x40 Tornillo DIN 912 M2x12 | | motedis | https://www.motedis.es/shop/Basicos-Mecanica/Basico-Basicos/DIN-pieza-normalizada/Tornillos-DIN/Tornillo-DIN-912::999991121.html |
|  | Tuerca Martillo M6 | | motedis | https://www.motedis.es/shop/Accesorios-para-Perfil-Ranurado/Accesorios-Perfil-30-Tipo-B-Ranura-8/Sliding-nuts-suitable-for-Nut-8-B-type/Tuerca-martillo-ran-8-tipo-B-M4-M5-M6::999998941.html https://www.amazon.es/Hexagonal-BiuZi-Sujetador-Galvanizado-Opcional/dp/B07WVW7SNJ/ref=sr_1_5?adgrpid=56245435815&dchild=1&gclid=CjwKCAjw57b3BRBI_EiwA1ImytoWNUGeW0C05f9XdYYopxTdmjMwnkR2RhCUCm_iKUvsj2iZ8_mHfHBoC3V0QAvD_BwE&hvadid=275347028870&hvdev=c&hvlocphy=9047044&hvnetw=g&hvqmt=e&hvran_d=551291719288938984&hvtargid=kwd-327827106366&hydadcr=1431_1736081&keywords=tuerca+m2&qid=1592722156&sr=8-5&tag=hydes-21 |
|  | Tuerca Hexagonal M2 | | amazon | |

[Escriba aquí]

[Escriba aquí]