UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGIENERÍAS INDUSTRIALES

Departamento: Ingeniería de Sistemas y Automática

Área de conocimiento: Ingeniería de Sistemas y Automática

TRABAJO DE FIN DE MÁSTER

Diseño y construcción de una bobinadora de filamento de plástico reciclado para impresión 3D

Design and construction of a plastic filament winder machine for 3D printing.

Máster en Ingeniería Mecatrónica

Universidad de Málaga, UMA.

Autor: Francisco Luque del Castillo

Tutor: J. Manuel Gómez de Gabriel

Málaga Octubre, 2022

DECLARACIÓN DE ORIGINALIDAD DEL PROYECTO/TRABAJO FIN DE GRADO

D./ Dña.: Francisco Luque del Castillo

DNI/Pasaporte: 25610227A. Correo electrónico: pacoluque1999@gmail.com

Titulación: Máster en Mecatrónica

Título del Proyecto/Trabajo: Diseño y construcción de una bobinadora de filamento de plástico reciclado para impresión 3D.

DECLARA BAJO SU RESPONSABILIDAD

Ser autor/a del texto entregado y que no ha sido presentado con anterioridad, ni total ni parcialmente, para superar materias previamente cursadas en esta u otras titulaciones de la Universidad de Málaga o cualquier otra institución de educación superior u otro tipo de fin.

Así mismo, declara no haber trasgredido ninguna norma universitaria con respecto al plagio ni a las leyes establecidas que protegen la propiedad intelectual, así como que las fuentes utilizadas han sido citadas adecuadamente.

En Málaga, Septiembre de 2022

Fdo.: Don Francisco Luque del Castillo.

AGRADECIMIENTOS

A mi familia, por la motivación que me han dado día tras día, en especial a mi padre, por la gran ayuda que me ha dado.

A mis compañeros de clase, ya que nos hemos ayudado unos a otros y más con los tiempos de la crisis sanitaria.

A mi tutor J. Manuel Gómez por ayudarme en el desarrollo del trabajo fin de master y ofrecerme todos los recursos necesarios.

A Marea Plastic y a todos sus integrantes por la confianza y la ayuda que me han aportado en el desarrollo de la máquina.

RESUMEN

El objetivo de este proyecto es contribuir para reducir la contaminación de nuestros mares provocada por los residuos plásticos, a los cuales se les puede dar una segunda vida gracias al reciclaje. Por ejemplo: se puede convertir una botella de agua en una bobina de filamento para imprimir en 3D.

El reciclaje del residuo plástico pasa por diversas máquinas. Este proyecto se centra en el diseño y la construcción de una de ellas, llamada bobinadora 3D, que recicla el residuo plástico y lo convierte en filamento 3D para las impresoras.

La bobinadora se encarga de enfriar, moldear y bobinar un filamento de plástico ya reciclado de espesor variable. Por lo tanto, hay que utilizar diferentes tecnologías con el fin de que el filamento sea apto para la impresión 3D.

Un requerimiento importante es priorizar el uso de materiales reciclados para la construcción de dicha máquina, evitando lo máximo posible su compra. Así se consigue demostrar que gracias al reciclaje esta tecnología está al alcance de todo el mundo, ya que las bobinadoras de filamento 3D existen en el mercado, pero son demasiado costosas y no son accesibles para un uso doméstico

Una vez diseñada esta máquina, se construirá un prototipo funcional que, junto a las diferentes máquinas necesarias, deberán ser capaces de convertir cualquier residuo plástico en una bobina apta para la impresión 3D. Las otras máquinas necesarias serán diseñadas y construidas por compañeros de la universidad que pertenecen al grupo Marea Plastic. Este grupo es un startup de la universidad que lucha contra la contaminación que producen los residuos plásticos.

ABSTRACT

The aim of this project is to contribute to reducing the pollution of our seas as a result of plastic waste, giving it a second life through recycling. For example: a water bottle can be turned into a spool of filament for 3D printing.

The recycling of plastic waste goes through a number of different machines. This project focuses on the design and construction of one of them, called a 3D winder, which recycles plastic waste and turns it into 3D filament for the printers

The winder is in charge of cooling, moulding and winding a plastic filament of variable thickness. Therefore, different technologies have to be used in order to make the filament suitable for 3D printing.

An important request is to focus on the use of recycled materials for the construction of this machine and to avoid the purchase of recycled materials as much as possible. This proves that, by recycling, this technology is available to everyone, because 3D filament winders exist on the market but they are too expensive and not affordable for domestic use.

Once this machine has been designed, a prototype will be built, together with the different machines needed to be able to convert any plastic waste into a spool suitable for 3D printing. The other required.

Índice

Índice de Planos

Índice de Figuras

Índice de tablas

Índice de códigos

CAPÍTULO 1. INTRODUCCIÓN

1.1. Historia del movimiento Marea Plastic

Este proyecto proviene de un movimiento medioambiental y ecológico llamado Marea Plastic [1]. El movimiento quiere reducir la contaminación de los océanos. Actualmente se vierten más de 200Kg de basura en los océanos por segundo. [2]

Esto es un problema bastante serio ya que contamina el planeta y perjudica a las especies marinas. Incluso esta contaminación puede acabar con algunas especies marinas.

Este movimiento, Marea Plastic es un grupo de investigación financiado por la universidad de málaga, está formado por un grupo de alumnos y profesores, cuya función es diseñar y desarrollar máquinas y medios necesarios para el procesado de los plásticos para su reciclado. Las máquinas son diseñadas y desarrolladas por los alumnos con ayuda de profesores.

Reciclar el plástico no es una tarea sencilla, ya que se necesitan varios procesos y es un poco complejo. Pero la universidad y profesores están facilitando recursos y material necesario para diseñar y crear las diferentes máquinas necesarias para procesar el plástico.

Lo bueno del reciclado de plástico es que se puede reciclar hasta el plástico de una botella, fundas de móviles, etc. Que normalmente acaban tiradas en la basura. Lo único que hay que diferencias los tipos de plásticos para su reciclado. Ya que no todos los plásticos tienen las mismas características.

Actualmente el grupo Marea Plastic está recibiendo desechos plásticos de los alumnos, profesores y empresas que quieren colaborar con el movimiento, ayudando a reducir la contaminación en el planeta.

1.2. Objetivo

El objetivo principal es diseñar y crear una máquina capaz de bobinar un filamento de plástico para su posterior uso en impresión 3D.

La idea principal es crear una máquina capaz de reciclar el plástico que se usa en el día a día y darle un nuevo uso. Este uso es impresión 3D. Se quiere convertir material “basura” en un rollo de filamento para poder imprimir en 3D con impresoras de filamento. El problema principal de reciclar el plástico es que es un proceso algo complejo.

Se utilizan varias máquinas para poder reciclar el plástico. Se necesita una trituradora, que sea capaz de triturar el plástico y convertirlo en pequeñas escamas. Posteriormente, se necesita lavar el plástico, para eliminar residuos. Por último, se necesita una extrusora y bobinadora. La extrusora se encarga de derretir el plástico y expulsarlo con forma de filamento. La bobinadora se encarga de controlar el espesor del filamento y bobinarlo en un rollo vacío.

La bobinadora tiene que ser capaz de enfriar el filamento que sale de la extrusora hasta la temperatura de cristal. Esta es la temperatura en el que el plástico es sólido pero moldeable. Una vez conseguido esta temperatura, la bobinadora controla el espesor. Ya que el espesor tiene que ser de 1,75mm. Este es un espesor común a la mayoría de impresoras 3D. Si el filamento tiene un espesor distinto afecta a la calidad del rollo, provocando obstrucciones o deslizamientos del motor que introduce el filamento en el extrusor de la impresora, estropeando así una impresión.

Una vez el filamento tenga el espesor deseado, el filamento se bobina de manera ordenada y correcta para evitar nudos y enredos.

1.3. Plan de trabajo

La máquina se ha dividido en varias etapas o estaciones, facilitando así su diseño y construcción. El primer paso que se ha realizado es diseñar las etapas y se han dividido en tres.

La primera etapa, es la que se encarga de enfriar el plástico proveniente del extrusor a su temperatura de cristal. Por ejemplo, el PLA tiene una temperatura de cristal de 60º. La temperatura de cristal es la temperatura de reblandecimiento del plástico, es decir, es una temperatura en la que el plástico se puede moldear sin llegar a derretirse. Esto se consigue controlado varios ventiladores pequeños reciclados.

La segunda estación, se encarga de controlar el diámetro. Para realizar esta tarea se ha creado un sensor de espesor que funciona a través de rodamientos y un sistema de palancas. Según el espesor del hilo, la palanca mueve un imán, alejándolo o acercándolo a un sensor de efecto hall. Hay que añadir que el sensor de efecto hall es lineal, ya que si no lo fuera se complicaría mucho este sistema. A continuación, se controla el espesor del filamento, aplicando tensión según el diámetro medido por el sensor. A mayor diámetro más tensión se ha de ejercer para reducir el espesor del filamento.

Si el diámetro es correcto, se mantendrá la tensión actual y si el diámetro es menor se baja la tensión.

La última estación, se encarga de bobinar el filamento. El problema de esta tarea es que el radio del rollo varia a medida que se va enrollando el filamento, por lo que hay que estudiar alguna solución. Para bobinar el hilo sin ejercer mucha tensión al final y poca al principio, existen dos soluciones, variar la velocidad del motor para conseguir una velocidad tangencial constante o variar la tensión con la que se bobina.

Una vez desarrollado el planteamiento de la máquina, se debe de montar una maqueta funcional, que sirva de prueba. La idea es que esta maqueta sea capaz de bobinar diferentes clases de plásticos, como por ejemplo PLA, PETG, Filaflex, EDP, etc. La única diferencia entre estos plásticos es la temperatura de cristal y la tensión que hay que ejercer para controlar el espesor. La temperatura de cristal se puede conocer por los datasheets de los plásticos y las tensiones que hay que ejercer se pueden obtener de forma experimental.

Por último, una vez se desarrolle la maqueta, hay que crear una capa software que se encargue de controlar las tensiones, temperatura, espesor y bobinado. Para ello se usará un microcontrolador. La primera estación controla la temperatura, por lo que debe de encender o apagar los ventiladores para enfriar el filamento. La segunda estación, se encarga de que el espesor sea el óptimo, variando la tensión. La tercera, se encarga de bobinar el filamento de forma correcta para su posterior uso.

CAPÍTULO 2. DISEÑO

2. Diseño

En este capítulo se realizará un estudio de los posibles diseños de la bobinadora, diferenciándolos en las tres etapas definidas en el apartado 1.3.

El objetivo principal es diseñar y realizar un análisis de los materiales necesarios para el montaje de la bobinadora. Para ello, se analizará posibles soluciones para cada etapa por separado. Pero antes se analizarán los requerimientos necesarios para realizar el diseño de la bobinadora.

Por último, se concluirá la solución aportada y se diseñarán las piezas necesarias para la máquina. Estas piezas serán diseñadas con la herramienta SolidWorks.

2.1. Requerimientos.

Antes de comenzar con las soluciones posibles hay que analizar los requerimientos exigidos para realizar el diseño de la bobinadora.

1. Utilizar la mayor cantidad de componentes reciclados.
2. Tiene que ser una máquina “portátil”, que se pueda transportar.
3. Debe de ser capaz de enfriar un filamento de plástico a una temperatura deseada.
4. Debe de controlar el espesor del filamento.
5. Tiene que bobinar el filamento de forma ordenada.

2.2. Estudio y análisis de posibles soluciones.

En este subapartado se realizará un estudio y análisis de las posibles soluciones de cada etapa por separado.

Para realizar este estudio hay que conocer la función que debe desempeñar cada etapa de la máquina.

La primera etapa se llama etapa de refrigeración y esta se encarga de enfriar el filamento de plástico.

La segunda etapa se denomina etapa de control, se encarga de controlar el espesor.

La tercera y última etapa se llama etapa de bobinado y se encarga de bobinar el filamento en un rollo de impresión 3D.

Una vez conocido las funciones de cada etapa, se realiza el análisis de cada etapa por separado.

Se analizarán múltiples soluciones y se concluirá cuál es la solución más optima en el apartado 2.2.

2.2.1. Primera etapa, etapa de refrigeración

Esta etapa se encarga de enfriar el filamento hasta la temperatura de cristal, para luego poder moldear el filamento.

Existen varias soluciones para este problema, lo que se busca es refrigerar el filamento, y existen varios tipos de refrigeraciones. Las más comunes son refrigeración líquida y refrigeración por aire.

La refrigeración líquida es simple, solo hace falta un compartimento sellado donde se encuentra el líquido y el filamento entra en este compartimento así enfriándose. El problema principal, es que es un poco complicado de controlar. Se podría controlar la temperatura del líquido y con eso se controlaría la temperatura de salida del filamento.

La refrigeración líquida también es algo compleja de manejar, ya al que tener un compartimento lleno de líquido es algo que puede producir problemas a largo plazo. Se podría picar el material con el que está hecho el compartimento y tener fugas, esto puede provocar que no se controle bien la temperatura o incluso hacer algún cortocircuito con algún componente electrónico. Para evitar esto, se debe usar líquidos dieléctricos, es decir, líquidos que no conduzcan la electricidad. Como por ejemplo, el agua destilada o algún refrigerante de ordenador.

Figura 2.1. Refrigeración líquida para filamento 3D

La refrigeración por aire es más simple de manejar y controlar, ya que este tipo de refrigeración consta de uno o más ventiladores que se pueden controlar por independiente. El control de esta refrigeración es más simple, debido a que se puede controlar los ventiladores por separado, y según la temperatura de salida se pueden encender o apagar los ventiladores.

La refrigeración por aire se realiza colocando uno o varios ventiladores en una estructura por donde pasa el filamento, esta estructura puede ser un tubo de metal o un perfil donde se colocan los ventiladores.

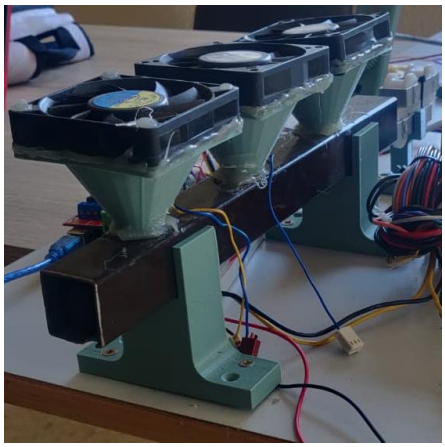


Figura 2.2. Refrigeración por aire para filamento 3D

2.2.2. Segunda etapa, etapa de control

En esta etapa se pretende controlar el espesor del filamento cuando este se encuentre a temperatura de cristal. Existen varias posibilidades y se van a analizar dos posibles soluciones. Ambas soluciones se basan en controlar la tensión del filamento. Al variar la tensión del filamento, se puede variar el espesor.

Existen varias formas de aplicar tensión al filamento, una de ellas es modificando la velocidad de los motores que mueven el filamento. Esto aplica una fuerza de tracción al filamento.

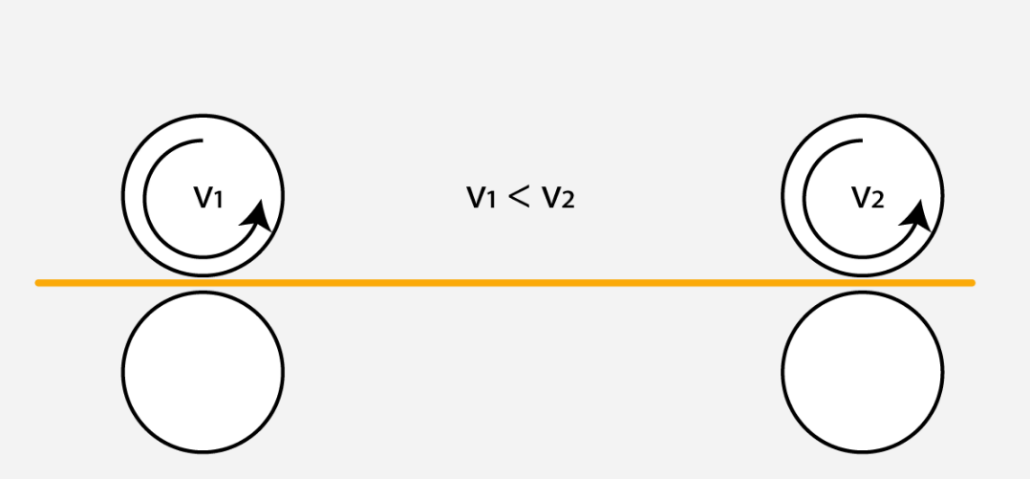


Figura 2.3. Tensión aplicando tracción

Siendo V_1 la velocidad del motor principal y V_2 la velocidad del motor que gira más rápido para ejercer una fuerza a tracción. Esta solución es simple pero compleja de controlar y además es costosa ya que se requieren dos motores.

Otra posible solución es aplicar una tensión directamente en el filamento, de forma similar al control de espesor de las planchas metálicas. Esto se consigue ejerciendo fuerza sobre el filamento moviendo un pistón con un rodamiento en la punta, como puede verse en la figura 2.4.

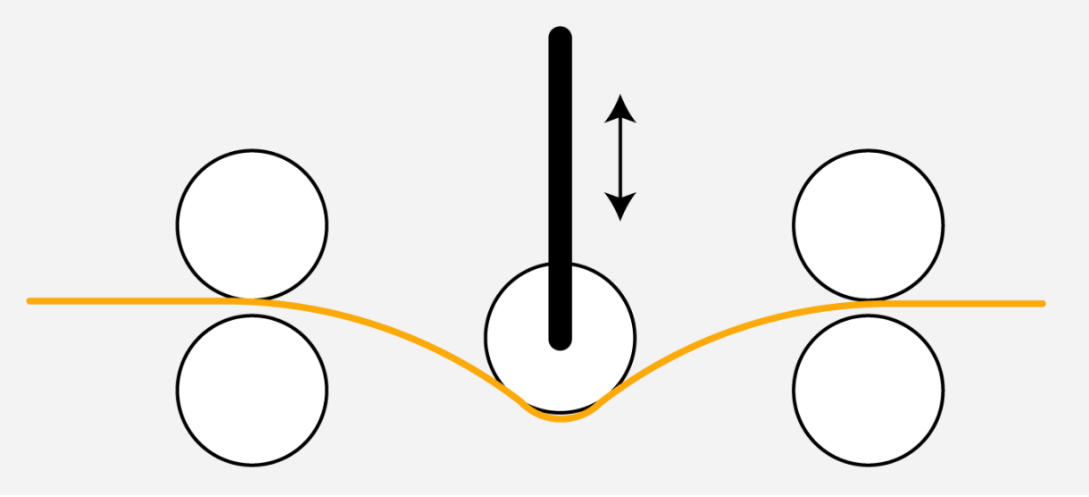


Figura 2.4. Tensión aplicando fuerza

También es posible controlar la tensión ejercida, con un motor paso a paso, variando el ángulo de un eje de pivote. Este sistema se usa en el sistema de laminación de planchas de acero. El modelo puede identificarse en la figura 2.5.

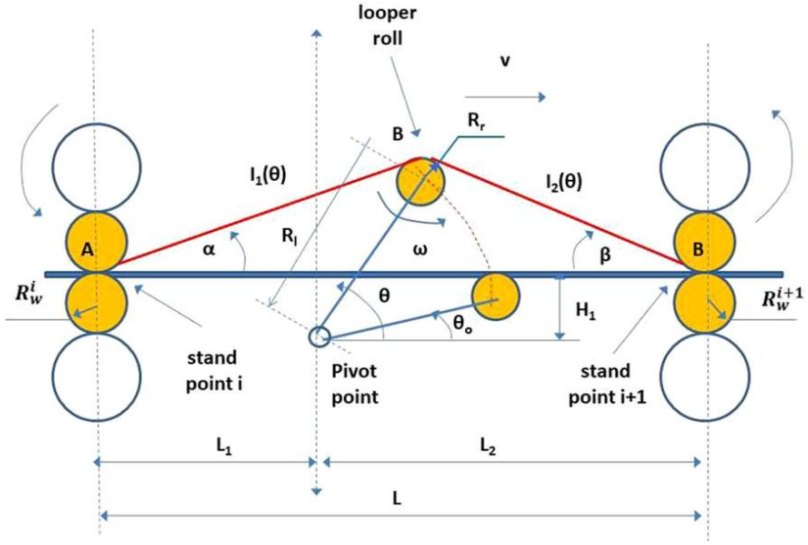


Figura 2.5. Sistema no lineal para el control del espesor de planchas de acero [3]

El sistema no lineal de control de espesor se usa para el control de espesor en láminas de acero. Un sensor preciso toma la medida del espesor de la plancha y un controlador calcula el ángulo que debe de tener el eje pivotante para moldear la plancha, ajustando su grosor a uno deseado. Este método se puede aplicar al control de espesor de filamento, adecuando los parámetros del control y modificando algún componente. Por ejemplo: en el control de las planchas de acero se usa un cilindro como "*looper rool*" (es el punto de apoyo de la plancha que se mueve para ejercer la tensión, Circunferencia B en la figura 2.5). En cambio, esto no es necesario para el filamento, no hace falta un cilindro alargado, solo hace falta un pequeño cilindro con unas guías para que no se salga el filamento del punto de apoyo.

El sistema no lineal de control de espesor es un sistema con retardo. El sensor es la entrada del sistema, que se encuentra alejado de la salida del sistema. Pero con la correcta calibración del sensor se puede obtener buen resultado. La calibración se debe de hacer de forma práctica, ajustando los parámetros del controlador PID que se describirá en el apartado de programación.

En el mercado existen varias opciones de sensores de espesor. La manera más sencilla es controlar el espesor con un sensor dial digital (Digital dial indicator, en inglés). Este sensor es simple de usar, ya que indica el recorrido que recorre la punta del sensor con una precisión de 0,01mm. Si se coloca en la punta del sensor el filamento, se podrá medir con facilidad la variación de espesor del filamento. Para colocarlo, se puede diseñar un soporte a partir de modelos ya creados en internet. Véase un ejemplo en la figura 2.6.

Figura 2.6. Soporte de dial digital por dy chen [4]

Otra solución sería crear el propio sensor a través de un sistema de palancas. Estaría formado por imán y un sensor lineal de efecto hall. Se coloca el filamento en un lado del sistema de palancas y en lado opuesto se coloca el imán. Según la distancia entre el sensor y el imán determinará el espesor del filamento. Véase el esquema en la figura 2.7.

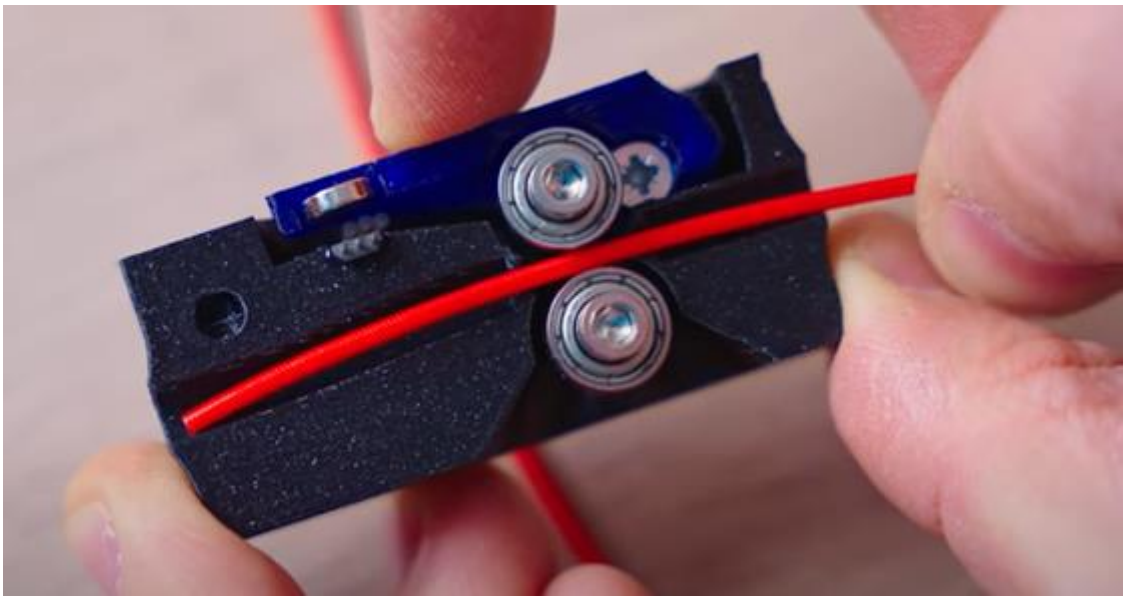


Figura 2.7. Sensor de espesor, efecto hall de Thomas Sanladerer [5]

Esta es una solución simple y barata de realizar. Debido a su bajo costo en materiales y a la eficacia que puede llegar a tener. Para realizar este modelo se ha partido de un modelo funcional realizado por Thomas Sanladerer [5].

2.2.3. Tercera etapa, etapa de bobinado

Esta etapa es la que se encarga de mover y almacenar de forma ordenada el filamento reciclado. El filamento debe de ser almacenado en una bobina, para su posterior uso en impresoras 3D.

Esta etapa es la más simple de diseñar, ya que solo consta de un soporte donde se coloque el rollo y este sea capaz de rotar con una velocidad controlable. Para ello se podría usar un motor paso a paso o un motor CC.

Se puede usar algunos modelos disponibles que se encuentran en internet. El problema de usar soportes de internet puede llegar a ser la compatibilidad de la bobina. La idea es que sea lo más universal posible y no todos los modelos son aptos para esta tarea. Por tanto, la mejor idea será desarrollar un soporte que sea universal a todas las bobinas que se encuentran en el laboratorio del grupo Marea Plastic.

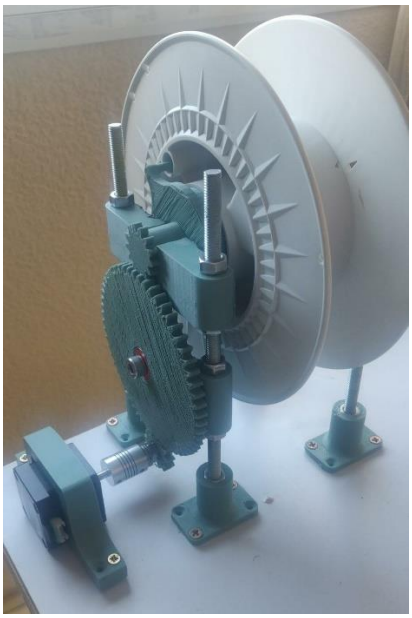


Figura 2.8. Soporte bobina, diseño propio

2.3. Soluciones concluidas

En este subapartado se van a concluir las soluciones aportadas para cada parte. Listando las piezas y materiales necesarias.

Las soluciones aportadas son aquellas que sean simples de controlar y no supongan un gran gasto. Ya que la idea de este movimiento es concienciar a la gente de que hay que reciclar y darle una segunda vida a los materiales que se tienen por casa.

Por tanto, para desarrollar la bobinadora se han usado diseños simples pero funcionales y se ha reciclado la mayor cantidad de piezas. Ya que la idea es el reciclado, la máquina también debía tener partes recicladas.

Como en el apartado 2.1, se realizará un análisis de las soluciones aportadas de cada etapa por separado, dando una explicación del motivo de la selección escogida.

También se realizará un listado de piezas que se han usado y se mostrará los diseños necesarios para las diferentes etapas.

2.3.1. Solución etapa de refrigeración

En la primera etapa se ha decidido escoger la refrigeración por aire, ya que es más fácil de controlar y se de usar. Además de que los ventiladores se pueden reciclar con facilidad.

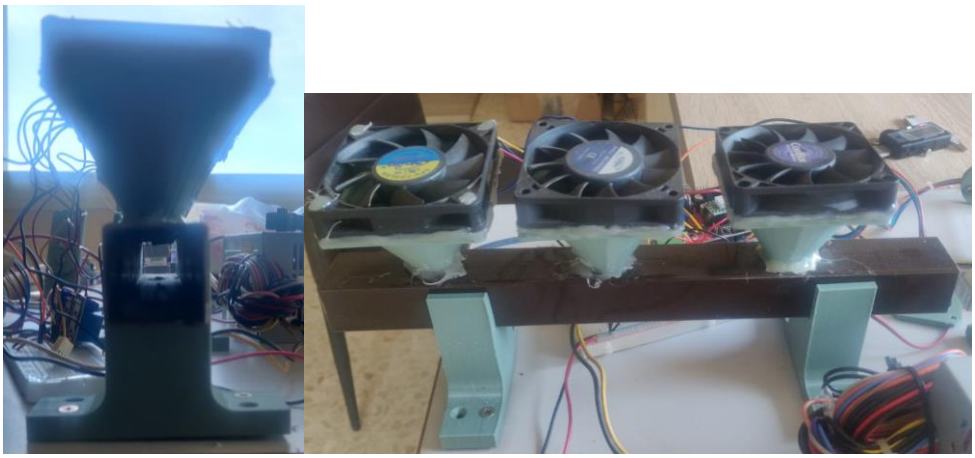
Se pueden usar ventiladores de alto flujo, turbinas o incluso ventiladores de ordenador pequeño.

Se ha seleccionado la refrigeración por aire debido a su sencillez y a la facilidad de encontrar los materiales necesarios.

En este caso se ha usado 3 ventiladores de corriente continua de ordenador de 70mm. Estos ventiladores pertenecían a la refrigeración de una CPU de un ordenador antiguo.

Otro material necesario para esta etapa es el soporte del ventilador, este se ha diseñado e impreso en 3D. Aunque otra solución sería usar embudos u otro material para sujetar los ventiladores y enviar el flujo de aire hacia el filamento.

Por último, se necesita un soporte de la refrigeración, por el que el filamento pueda circular libremente. Este puede ser un perfil metálico o de otro material. En este caso se ha seleccionado un perfil cuadrado. Este perfil ha sido facilitado por la universidad, debido a que no se iba a usar. Se le han realizado tres agujeros donde se colocarán los embudos con el ventilador. Véase el esquema en la figura 2.9.



a) Vista frontal *b) Vista lateral*

Figura 2.9. Sistema refrigeración, diseño propio

Para finalizar se ha diseñado un pequeño pie para apoyar el perfil metálico y poder atornillarlo a una mesa donde se está construyendo la máquina.

Véase una lista de los materiales necesarios en el apartado 2.3.

2.3.2. Solución etapa de control

En esta etapa se ha escogido la solución del control no lineal de espesor que se usa para controlar el espesor de las láminas de acero. Debido a que es un modelo que se puede encontrar bien detallado y no es complicado de implementar aparte de que los materiales necesarios no son muy costosos.

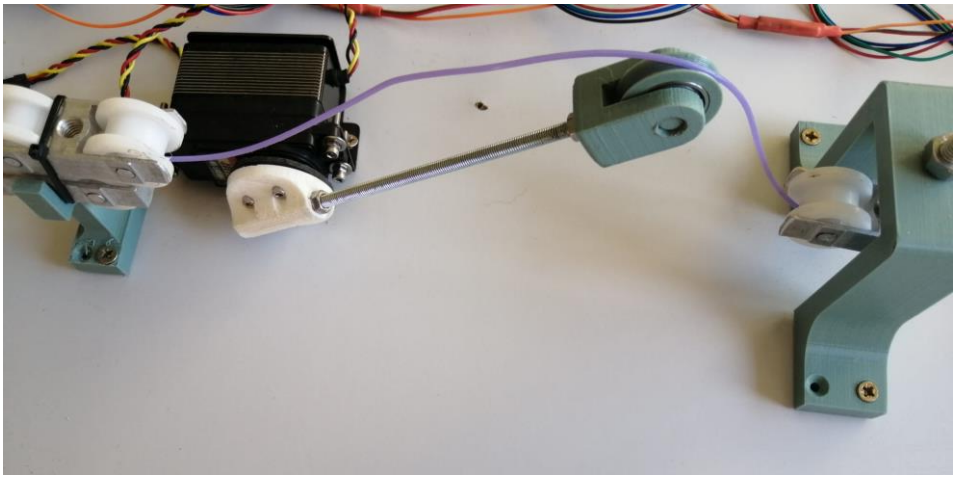


Figura 2.10. Etapa de control

El principal material necesario es un motor paso a paso, este realizará el control del ángulo, modificando así el espesor del filamento.

El motor paso a paso controla el ángulo de un pivote de rotación. Para realizar la función de pivote se ha usado una varilla roscada, para que sea luego más fácil de insertar un rodamiento en la punta.

Para colocar un rodamiento en la varilla se ha realizado un diseño e impreso en 3D de un soporte con un hueco hexagonal para introducir a presión una tuerca. Con esta tuerca se facilita el roscado a la varilla roscada. Se puede ver la pieza en la figura 2.11.

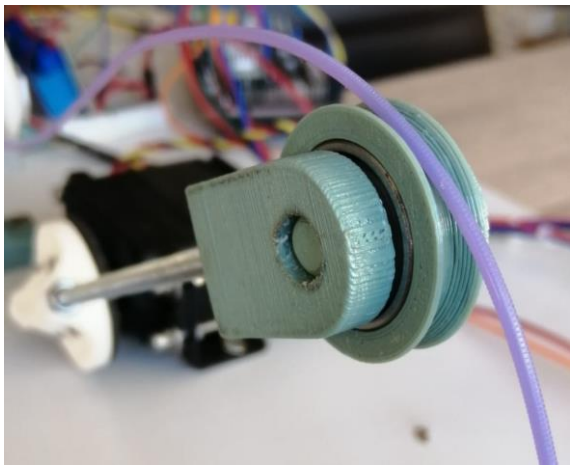


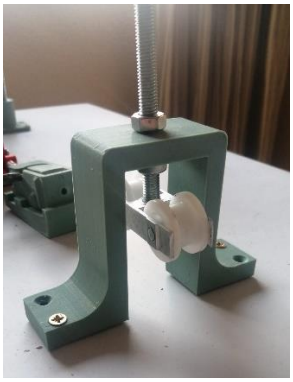
Figura 2.11. Soporte de rodamiento en el pivote de rotación, diseño propio.

El rodamiento consta de un soporte con una entrada hexagonal para una tuerca y además de un hueco para introducir un rodamiento, que tenía por casa de unos patines, que es un modelo estándar. Se ha diseñado una guía que encaja con el rodamiento y así mejora la sujeción del filamento a la hora de ejercer la fuerza. Todo esto se sujeta al soporte con un pasante que se ha diseñado e impreso en 3D.

Otro material necesario son los puntos de guía, es decir, puntos por donde pasa el filamento y esta ajustado y controlado. Restringiendo la libertad de movimiento del filamento. Solo tiene libertad de avanzar de forma horizontal.

Los puntos de guía constan de unos rodamientos reutilizados de una impresora de papel. Se han diseñado unos soportes para estos puntos guía. El primer punto se encuentra justo a la salida de la refrigeración de aire y el segundo se encuentra al final de la etapa. Siendo el motor la parte que se encuentra en el medio. Véase los dos soportes con los rodamientos en las figuras 2.12.

a) Soporte guía principio



b) Soporte guía final

Figura 2.12. Soporte guía, diseño propio.

Como se puede apreciar en la figura 2.12 a, el soporte es fijo, consta de dos rodamientos uno encima del otro que restringe el movimiento del filamento en todas las direcciones menos en la que interesa. El soporte es fijo ya que la altura de la salida de la primera etapa es fija y no hace falta ajustarla. En cambio, el soporte final es de altura variable, esto se debe a un mejor ajuste y así poder ajustar mejor las tensiones que se van a ejercer. Cuanto más arriba esté menos tensión ejercerá el motor, debido a que el ángulo que se genera es menor. Véase este ángulo (el ángulo del que se genera con el soporte final es el ángulo β) en la figura 2.5.

2.3.3. Solución etapa de bobinado

La solución para esta etapa es simple. Se han comparado varios modelos por internet, pero ninguno era estándar y daría un buen resultado. Ya que la mayoría de los soportes de bobinas que existen en internet es para desbobinar el filamento y colocarlas en una impresora 3D. Esto es incompatible con la solución que se está buscando.

Se busca un soporte que sea capaz de girar la bobina con la finalidad de bobinar de forma ordenada el filamento. Por tanto, se ha diseñado un soporte desde cero.

Como la velocidad tiene que ser controlable hace falta un motor paso a paso. El motor paso a paso se puede situar en la parte superior del soporte, ejerciendo la fuerza de rotación directamente en la bobina o situado en la mesa. Se ha optado por colocarlo en la mesa atornillado, para simplificar el modelo del soporte.

Para transmitir el movimiento de rotación existen varias opciones, una sería usar una cadena con dos engranajes o correa. Y la otra solución sería usar engranajes directamente. Se ha optado por la solución de los engranajes porque es más simple de implementar y así también se ahorra tiempo en buscar una correa adecuada.

Figura 2.13. Engranajes, diseño propio.

Los engranajes están diseñados para transmitir el mismo par y velocidad que transmite el motor. La relación de transmisión es 1:1. Existen dos engranajes pequeños de 13 dientes y otro engranaje en medio de 58 dientes. Si se usa la fórmula de transmisión se obtiene que la relación es 1:1.

$$Vi * Zi = Vo * Zo$$

Siendo V_i y Z_i la velocidad y el número de dientes de la rueda de entrada. V_o y Z_o son la velocidad y el número de dientes respectivamente del engranaje de salida. La relación que se obtiene es:

$$rt = \frac{Zi}{Zo}$$

Si se tiene 3 engranajes como es el caso quedaría:

$$rt = \frac{Z_{peq_{in}} * Z_{gr}}{Z_{gr} * Z_{peq_{out}}}$$

Si Z_{peqin} y Z_{peqout} son iguales, la relación de transmisión es 1.

A continuación, se modela el soporte. Para realizar esta tarea se necesita conocer las medidas del orificio de la bobina. Usando varios modelos de bobina, se obtiene que el orificio de media tiene 53mm. Ahora viene el principal problema. ¿Qué sistema hay que implementar para hacer girar la bobina? Hay diversas soluciones. Que el pasante donde se coloca la bobina este muy ajustado y por fricción es capaz de mover la bobina. Pero esta solución no es común a todas las bobinas e incluso puede fallar resbalando y no se bobinará de forma correcta.

Otra solución posible es usar algún punto de apoyo para impulsar la bobina. Si se observa varios modelos de bobinas (Véase en la figura 2.14) existe un hueco pequeño. Si se usara un pasante pequeño, este podría impulsar sin problemas a la bobina. El problema de este hueco es que no están situados en la misma posición en todas las bobinas ni tienen la misma dimensión. No obstante, se puede hacer una media y hacer un pasante que se pueda adaptar a todas las bobinas. Así se consigue una solución bastante buena y sin problemas de deslizamiento.

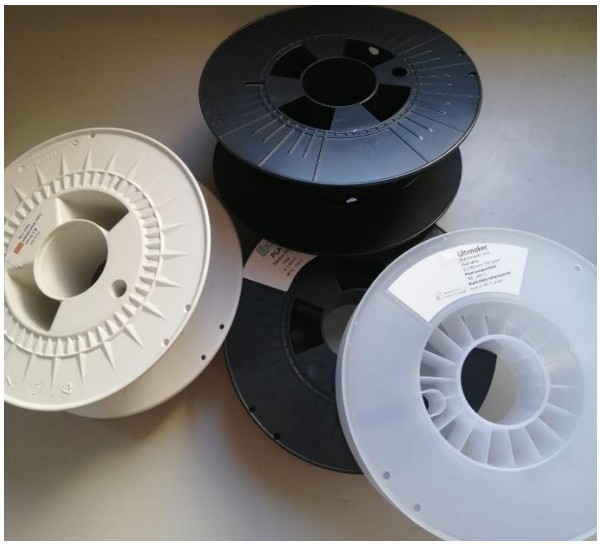


Figura 2.14. Diferentes bobinas.

Por último, se ha diseñado un soporte que mantiene y sujeta todo el sistema. El diseño consta de 4 pies con una tuerca incrustada, que va atornillada a la base (una mesa). En el pie se introduce una varilla roscada M8 con una altura de 150mm. La siguiente parte que se diseña es un punto de apoyo para el engranaje grande que se encuentra a mitad de altura. Por último, se diseña el soporte donde irá el pasante de la bobina que realiza el movimiento.

Para evitar rozamientos y problemas con la fricción, se ha usado un poco de grasa de litio, mejorando la efectividad, y disminuyendo el esfuerzo del motor.

Figura 2.15. Soporte completo, diseño propio.

2.4. Lista de materiales

En este apartado se recogen un resumen de los materiales necesarios para las diferentes partes.

2.4.1. Lista materiales etapa de refrigeración

En este punto se concluyen la lista de materiales necesarios para construir la primera etapa de la bobinadora.

Lista	Número de unidades	Medidas	Material	Impreso 3D
Ventiladores	**3**	**70mm**		
Soporte ventilador	**3**	**70mm**	**PETG**	**Si**
Tubo de enfriamiento	**1**	**310x30x30mm**	**Aluminio**	
Pie	**2**		**PETG**	**Si**
Tornillos	**4**	**M4**		

Tabla 2.1. Listado de materiales etapa refrigeración

Los diseños de las piezas impresas en 3D se pueden encontrar en el capítulo de planos.

2.4.2. Lista materiales etapa de control

En este punto se concluyen la lista de materiales necesarios para construir la segunda etapa de la bobinadora.

Lista	Número de unidades	Medidas	Material	Impreso 3D
Puntos guía	2		PVC	
Soporte primera guía	1		PETG	Si
Soporte segunda guía	1		PETG	Si
Tuercas	2	M8	Aluminio	
Varilla roscada	1	M8 x 120mm	Aluminio	
Sujeción motor paso a paso	1		PETG	Si
Soporte rodamiento pivotante	1		PETG	Si
Rodamiento patines	1	R_{ext} 22, R_{int} 8 mm		
Guía rodamiento	1	R_{ext} 24, R_{int} 22 mm	PETG	Si
Tuercas	2	M5	Aluminio	
Varilla roscada	1	M5 x 140 mm	Aluminio	
Extensión servo-varilla	1		PLA	Si
Sensor espesor	1		PETG	Si
Soporte sensor espesor	1		PETG	Si
Rodamientos pequeños	4	623ZZ		
Muelle de bolígrafo	1			
Imán	1	25 x 6 x 2 mm		
Tornillo	1	M6		
Tornillos	8	M4		

Tabla 2.2. Listado de materiales etapa de control

Los diseños de las piezas impresas en 3D se pueden encontrar en el capítulo de planos.

2.4.3. Lista materiales etapa bobinadora

En este punto se concluyen la lista de materiales necesarios para construir la tercera etapa de la bobinadora.

Lista	Número de unidades	Medidas	Material	Impreso 3D
Pies	4		PETG	Si
Pieza A soporte	1		PETG	Si
Pieza B soporte	1		PETG	Si
Pieza C soporte	1		PETG	Si
Extensión motor paso a paso	1	M5 a M8		
Sujeción motor paso a paso	1		PETG	Si
Engranaje	1	Z = 13	PETG	Si
Engranaje	1	Z = 58	PETG	Si
Tornillo	2	M8	Aluminio	

Pasante bobina	**1**		**PETG**	**Si**
Bobina vacía				
Varilla roscada	**4**	**M8 x 160mm**	**Aluminio**	
Tuercas	**19**	**M8**	**Aluminio**	
Tornillos	**10**	**M4**	**Aluminio**	
Arandelas		**M8**	**Aluminio**	

Tabla 2.3. Listado de materiales etapa bobinadora

Los diseños de las piezas impresas en 3D se pueden encontrar en el capítulo de planos.

.

CAPÍTULO 3.

ELECTRÓNICA Y PROGRAMACIÓN

3. Programación

En este capítulo se describe la programación de la bobinadora. Para realizar esta tarea es necesario conocer las herramientas necesarias. Es indispensable hacer un estudio de los materiales que se van a utilizar y si es posible reciclar el mayor número de componentes.

El equipo electrónico que se usará para la programación es un microcontrolador Arduino Mega. Una vez conocido el hardware hay que listar los componentes necesarios que se van a utilizar.

Una vez listado los componentes necesarios, hay que comprobar que librerías o herramientas hacen falta. Por ejemplo, no se puede usar un motor paso a paso sin un driver capaz de controlarlo. Por tanto, hay que saber que driver es necesario.

3.1. Conocimientos previos

En este apartado se repasarán los conocimientos previos necesarios para programar el microcontrolador Arduino.

Lo primero es conocer el Arduino, es un microcontrolador programable, que tiene varios puertos digitales y analógicos, tanto de entrada como de salida. Un dato importante es la alimentación del Arduino. Se puede alimentar por el cable tipo USB B o por una alimentación externa de máximo 20V, aunque lo recomendable es 12V. Si se alimenta por el cable tipo USB B el voltaje de entrada es 5V.

A continuación, hay que tener en cuenta que los motores paso a paso necesitan un driver para programarlos. Se podría usar un par de drivers sueltos en una protoboard, pero como se necesitan 3 motores paso a paso, es mejor usar una Ramp Shield. Esta es un módulo de expansión que se usa para impresoras 3D, CNC, máquinas de grabado laser, etc. Debido a su facilidad de conectar los drivers y su sencilla programación.[6]

3.1.1. Fuente alimentación.

En este subapartado se hablará de un punto importante, la fuente de alimentación.

No es solo la fuente de alimentación del Arduino, también es la fuente de poder de los ventiladores y motores. Los ventiladores y motores paso a paso funcionan a 12V. Por lo que se necesita una fuente capaz de suministrar 12V.

Se podría buscar una fuente de alimentación en el mercado y no son muy caras, pero la idea es reciclar lo máximo posible. Debido a este requerimiento, se ha usado una fuente de alimentación de un ordenador de sobremesa.

Los componentes de un ordenador de sobremesa funcionan a 3.3 , 5 y 12V, siendo 12V lo que se necesita. Hay que investigar que cables otorgan la potencia deseada. Se encuentran cables de diferentes colores, cada color corresponde a una tensión, se puede comprobar en la figura 3.1, que tensión tiene cada cable por su color. Se necesita el color amarillo, este otorga 12V. Otro dato necesario es seleccionar el cable amarillo correcto. Aunque se puede usar cualquier cable amarillo, no todos distribuyen la misma potencia. El cable SATA suministra 12V, pero puede ser que no otorgue la potencia necesaria. En cambio, los cables que salen de la extensión de la CPU son capaces de suministrar más potencia. Por tanto, se recomienda usar esos cables en vez de los cables de la extensión SATA.

Existe un problema al usar una fuente de alimentación de ordenador y es que si se enchufa a la corriente la fuente de alimentación no se encenderá automáticamente. Ya que es el ordenador quien manda la señal de encendido, hay que buscar la forma de puentear el ordenador y poder encenderla.

La señal de encendido la manda el ordenador a través de los cables de 24 pines que van conectados a la placa base. Suele ser un conector de 24 pines, aunque también existen de 20. Lo que hace la placa base es cerrar o abrir el circuito según si se quiere apagar o encender la fuente de alimentación. Si está cerrado el puente, la fuente de alimentación enciende el ordenador.

Los cables que hay que puentear es el cable "*PS_ON"* con un cable "*COM",* véase en la figura 3.1 que cables hay que puentear de forma gráfica.

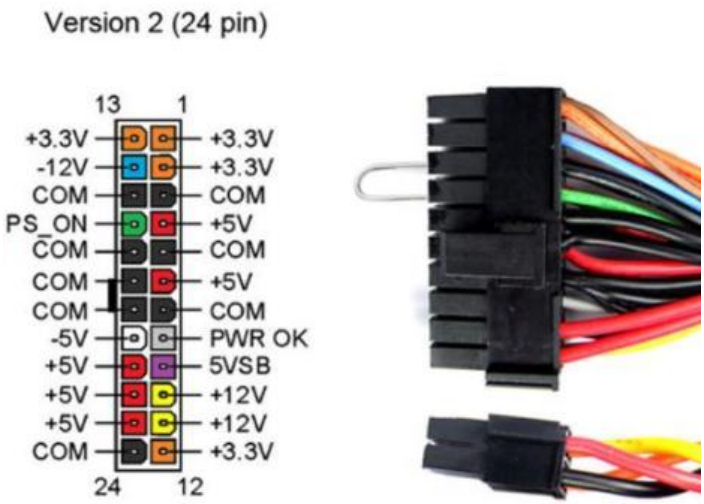


Figura 3.1. Detalle del puente fuente alimentación de un conector ATX.

Para puentear la fuente de alimentación se ha colocado un interruptor reciclado de una máquina antigua. Facilitando así el encendido y apagado de la máquina.

De los cables amarillos de la fuente de alimentación también se suministra energía a una línea de protoboard. Teniendo así 12V en la protoboard. Además de los 12V también se usa un cable rojo, para alimentar otra línea de la protoboard con 5V. Esta tensión se usará para suministrar la energía a los relés. Puede verse en la figura 3.2.

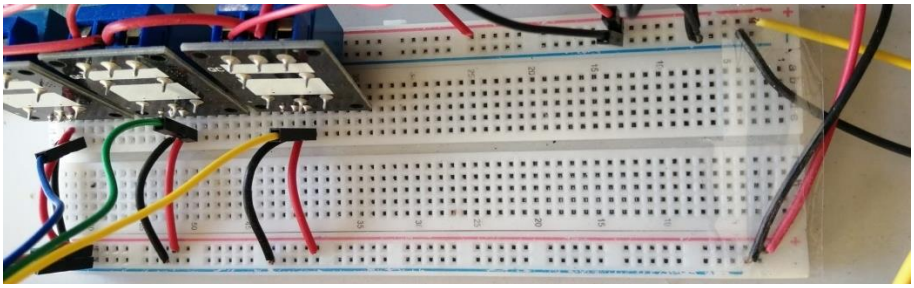


Figura 3.2. Conexiones protoboard.

Se conecta la línea superior a 12V que otorga el cable amarillo, se puede observar este cable en la esquina superior derecha. También se conecta la línea inferior a 5V. Esta potencia la suministra el cable rojo que se observa en la esquina inferior derecha.

En la figura también se aprecian los relés conectados a 5V y otros 3 cables (color amarillo, verde y azul) que son la señal de activación o apagado de los relés. Este cable está conectado al Arduino.

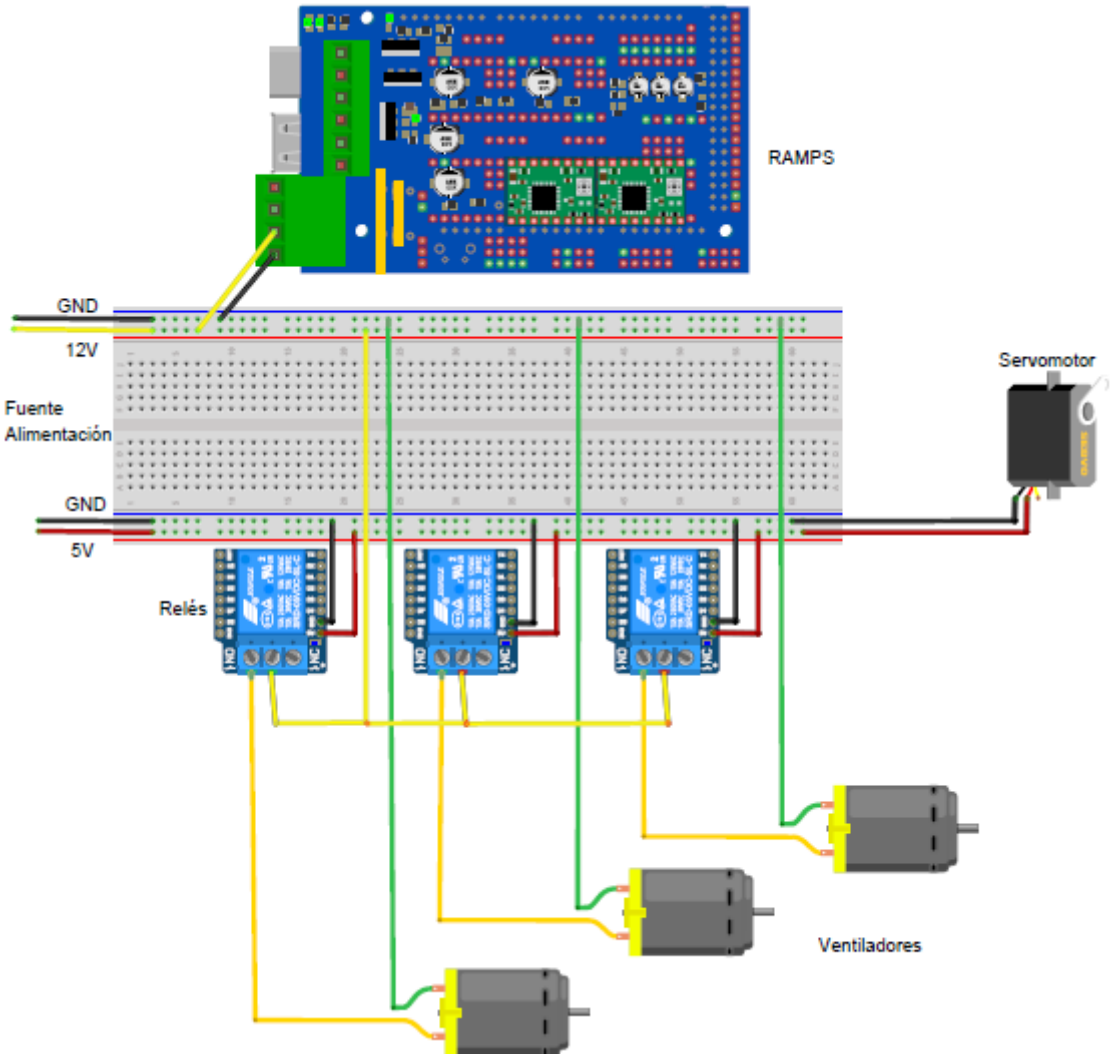


Figura 3.3. Esquema conexiones protoboard.

3.1.2. Ramp shield

RepRap Arduino Mega Pololu Shield (su acrónimo es Ramps) es una expansión del microcontrolador Arduino. La shield que se usará para este proyecto es la Ramps V1.4.

Hay que conocer con detalle el esquema electrónico de la ramps, debido a que la distribución de los pines y puertos es diferente a la del Arduino.

Los puertos en el Arduino se distribuyen de forma ordenada y con una enumeración. En cambio, en la ramps dos pines consecutivos no tienen porque coincidir con dos pines consecutivos en el Arduino. Por tanto, hay que mirar bien cada pin en la ramps y su

correspondencia en el Arduino. Puede verse una representación de las conexiones físicas simplificado en la figura 3.4.

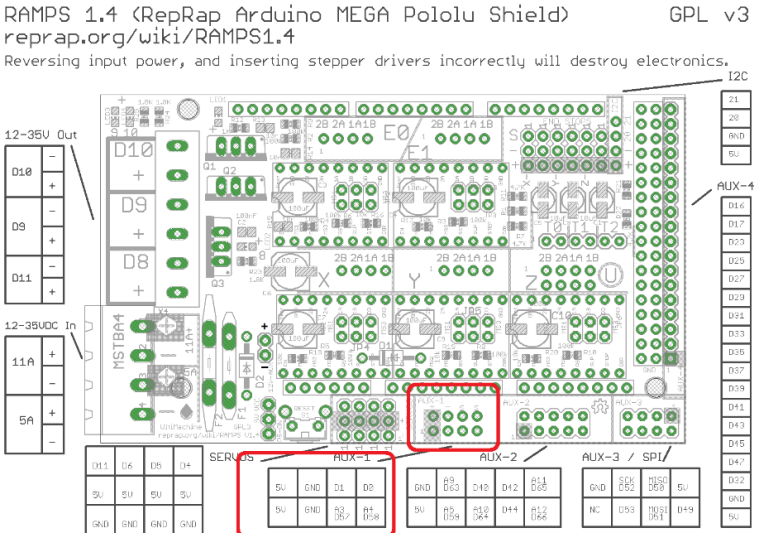


Figura 3.4. Representación de las conexiones físicas Ramps.[6]

Una vez conocido el esquema de la Ramps, se conecta al Arduino y se conectan los drivers de los motores paso a paso.

Los drivers seleccionados para este proyecto son los A4988, es un driver simple y sencillo de programar. Los drivers es el componente que se encarga de controlar los pasos de los motores paso a paso.

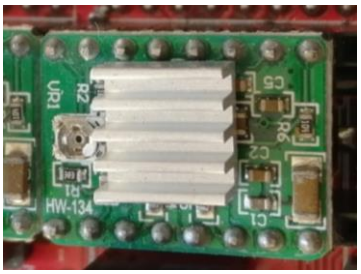


Figura 3.5. Driver A4988.

Este driver se caracteriza por: [7]

- Proporciona una interfaz de control y dirección simple.
- 6 resoluciones de pasos diferentes: paso completo (1 vuelta), 1/2 paso, 1/4 paso, 1/8 paso, 1/16 paso, 1/32 paso que se configuran según la cantidad de jumpers colocados en la ramps.
- Control de corte inteligente.
- Desconexión térmica.
- Control de corriente ajustable deforma física a través del potenciómetro.

3.2. Componentes electrónicos

En este punto se resumirá los componentes electrónicos necesarios para el correcto funcionamiento de cada etapa por separado.

3.2.1. Componentes etapa de refrigeración

La primera etapa es simple conceptualmente. Solo se requiere unos ventiladores y un interruptor que encienda y apague los ventiladores.

Para ello, se han seleccionado 3 ventiladores de CPU de un ordenador de sobremesa antiguo.

Los interruptores pueden ser mecánicos o electromecánicos, es decir, un interruptor normal, un switch o un relé de 5V que se pueda controlar con el Arduino.

La opción seleccionada son los relés. Usando relés se puede hacer un control de temperatura automático y se simplifica la tarea de enfriamiento bastante. La entrada de este control es un sensor de temperatura infrarrojo. Este sensor es capaz de medir la temperatura de un objeto a distancia, se va a usar el sensor MLX90614-DCI. Según la temperatura que mida el sensor, activa o desactiva los ventiladores, controlando así la temperatura del filamento.

Lista	Número de unidades	Precio/u	Precio total aproximado
Relés	**3**	**5**	**15**
Sensor óptico	**1**	**40**	**40**
Cables			**3,33**
Total			**58,33 €**

Tabla 3.1. Listado de componentes electrónicos etapa de refrigeración

3.2.2. Componentes etapa de control

La segunda etapa es la más compleja, esta se encarga de controlar el espesor del filamento, y no es tarea fácil.

Para esta tarea se puede usar un motor paso a paso, pero los motores que se han probado y usado no tienen la suficiente fuerza, por tanto, se diseñó una reductora orbital. Pero aun así seguía siendo insuficiente. A veces se saltaba un paso y esto puede ser un problema, se puede solucionar haciendo otro control de ángulo con un encoder. Esto es bastante complejo y se ha buscado otra solución.

Otra solución es usar un servo motor con un gran torque. Esto facilitaría el trabajo, ya que se ahorra el control de ángulo, es innecesario. Tiene mayor torque que el motor paso a paso, por lo que no se necesita reductoras.

Por último, hace falta un sensor de espesor, este se ha realizado con un sensor de efecto hall lineal. Con un sistema de palancas, según el espesor del filamento un imán se acerca o se aleja del sensor y esa medida es proporcional al espesor del filamento.

A continuación del sensor de espesor se coloca un motor, que mueve el filamento, este motor debe de ir sincronizado con la velocidad lineal del filamento a la hora de salir de la extrusora. Esta velocidad se ajustará experimentalmente.

Lista	Número de unidades	Precio/u	Precio total aproximado
Motor paso a paso	1	8	8
Sensor efecto hall lineal	1	0,5	0,5
Servo	1	8	8
Cables			3,33
Total			19,83 €

Tabla 3.2. Listado de componentes electrónicos etapa de control

3.2.3. Componentes tercera etapa

La tercera etapa es más simple que las etapas anteriores, ya que solo hace falta un motor que sea capaz de girar la bobina y pueda bobinar el filamento.

La parte más compleja de esta parte es sincronizar la velocidad lineal del filamento con la velocidad de rotación de la bobina. Este factor que relaciona las velocidades se puede obtener de forma experimental a la hora de ensamblar y calibrar la máquina.

Lista	Número de unidades	Precio/u	Precio total aproximado
Motor paso a paso	1	8	8
Cables			3,33
Total			11,33 €

Tabla 3.3. Listado de componentes electrónicos etapa 3

3.3. Programación

En este punto se resumirá los códigos usados para el correcto funcionamiento de la máquina. Se estructurará por separado para una mejor comprensión. Por último, se explicará cómo se ha unido los códigos y los cambios que se han realizado para la sincronización y su correcto funcionamiento.

El entorno que se ha usado para programar la máquina ha sido el propio entorno de Arduino, que es muy parecido a C++. El entorno propio de Arduino permite el uso y manejo de diferentes librerías que facilitarán el trabajo de la programación.

3.3.1. Programación etapa de refrigeración

Como se ha mencionado en capítulos anteriores apartados, el primer apartado trata de controlar la temperatura del filamento, para ello debe de controlar el encendido y apagado de los ventiladores.

Para realizar esta tarea se ha usado un relé por ventilador, el relé tiene 3 entradas, un positivo, un negativo y una señal que activa o desactiva el relé. Los pines positivo y negativo se conectan a 5V y a GND de la protoboard. (Véase los terminales positivos de 5v y GND en la figura 3.2)

La señal encargada de controlar el encendido se conecta a las salidas digitales del Arduino. Viendo el esquema de la Ramps (Figura 3.3), se concluye que los pines necesarios para esta tarea serán 16, 17 y 23.

En el código adjunto a continuación se muestra cómo se pueden activar y desactivar los ventiladores. De forma periódica y sin control. Solo para comprender su funcionamiento.

```
const int Fans[3] ={16,17,23};// the number of the FAN pin

// Variables will change:
int State = LOW;             // State used to set the FAN

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;        // will store last time FAN was updated

// constants won't change:
const long interval = 4000;           // interval at which to blink (milliseconds)

void setup() {
  // set the digital pin as output:
  pinMode(Fans[0], OUTPUT);
  pinMode(Fans[1], OUTPUT);
  pinMode(Fans[2], OUTPUT);
}


void loop() {
  // here is where you'd put code that needs to be running all the time.

  // check to see if it's time to blink the FAN; that is, if the difference
  // between the current time and last time you blinked the FAN is bigger than
  // the interval at which you want to blink the FAN.
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the FAN
    previousMillis = currentMillis;

    // if the FAN is off turn it on and vice-versa:
    if (State == LOW) {
      State = HIGH;
    } else {
      State = LOW;
    }

    // set the FAN with the State of the variable:
    digitalWrite(Fans[0], State);
    digitalWrite(Fans[1], State);
    digitalWrite(Fans[2], State);
  }
}
```

Código 3.1. Código de ejemplo ventiladores

La primera parte del código se declaran los pines como constantes enteras. La siguiente variable que se observa "*State*" sirve para declarar el estado de los ventiladores, encendido o apagado. Las dos siguientes variables que se encuentran sirven para controlar el periodo de la señal de encendido y apagado.

A continuación, se observa la función setup, donde se declara el modo del pin, en este caso se declara los pines como salidas digitales.

Por último, se encuentra la función loop, esta función es la función principal que está en constante repetición. Es decir, que esta función se ejecuta constantemente de forma cíclica, como un bucle.

Hay que tener en cuenta que la función loop es un bucle, si se quiere crear una señal cuadrada o de encendido y apagado se debe de evitar a toda costa el uso de delays, ya que es una mala práctica. Para realizar esta tarea se crea una condición que según el tiempo que ha pasado desde la ejecución anterior, es capaz de crear la señal deseada. Para eso sirven las variables que se encuentran al principio "*previousMillis*" y "*interval*". Siendo la marca de tiempo de cuando se actualizó la señal por última vez, y el intervalo de tiempo que debe de pasar para la actualización de la señal, respectivamente.

Dentro de esta condición, se puede observar cómo se cambia el estado de los ventiladores según el estado anterior. Esto es un ejemplo para ver como funcionan los ventiladores, posteriormente se añadirá un control PID que se explica detalladamente en el apartado 3.3.4.

El controlador PID necesita una entrada, esta entrada será la temperatura del filamento, para ello se usará un sensor de temperatura que se comunica por bus I2C (Inter-Integrated Circuit o también conocido como 2 alambres) con el Arduino. I2C es un puerto y protocolo de comunicación por serie. Los datos son transferidos entre dos cables, SDA (datos) y SCL (reloj). El protocolo de comunicación I2C es muy usado para la transmisión de datos de sensores digitales, permitiendo una velocidad de transmisión 1000kbits/s.

La programación del sensor de temperatura es simple. Se debe de usar la librería del propio sensor, facilitando la tarea de comunicación entre el sensor y el Arduino. La librería que se va a usar es "*Adafruit_MLX90614.h*", la librería permite instanciar el objeto directamente y leer los datos de temperatura con una función propia de la librería. También se hace uso de la librería "*Wire.h*", esta permite la comunicación por el bus I2C.

```
#include <Wire.h>
#include <Adafruit_MLX90614.h>

// Instanciar objeto
Adafruit_MLX90614 termometroIR = Adafruit_MLX90614();

void setup() {
  // Iniciar comunicación serie
  Serial.begin(9600);

  // Iniciar termómetro infrarrojo con Arduino
  termometroIR.begin();

}

void loop() {
  // Obtener temperaturas grados Celsius
  float temperaturaAmbiente = termometroIR.readAmbientTempC();
  float temperaturaObjeto = termometroIR.readObjectTempC();
  // Mostrar información
  Serial.print("Temp. ambiente => ");
  Serial.print(temperaturaAmbiente);
  Serial.println("°C");
  Serial.print("Temp. objeto => ");
  Serial.print(temperaturaObjeto);
  Serial.println("°C");

  delay(2000);
}
```

Código 3.2. Código toma de datos sensor temperatura

En la parte superior del código se observa la declaración de las dos librerías mencionadas anteriormente. Posteriormente se instancia el sensor infrarrojo de temperatura. En la función setup de inicia el sensor de temperatura.

Para terminar, en la función loop se llama a la función propia de la librería "*readAmbientTempC*" y a la función "readObjectTempC". La primera se encarga de leer la temperatura ambiental y la segunda de leer la temperatura del objeto al que apunta el sensor. Posteriormente este dato será la entrada del controlador PID que actuará sobre los ventiladores.

3.3.2. Programación etapa de control

La segunda etapa es algo más compleja de programar debido al ajuste y calibración del controlador que se encarga de ajustar el espesor del filamento.

La programación del servomotor es bastante sencilla, gracias a la librería de Arduino que permite la programación del servo de una manera muy sencilla. Lo único que hay que hacer es incluir la librería "*Servo.h*", y declarar el pin (tiene que ser un pin con salida PWM). Y por último se escribe el comando con el ángulo deseado. Véase en el código a continuación.

```
#include <Servo.h>

Servo myservo;

void setup()
{
  myservo.attach(11);
  myservo.writeMicroseconds(1500);  // set servo to mid-point
}

void loop() {
  // put your main code here, to run repeatedly:
  myservo.write(135);
  delay(1000);
  myservo.write(50);
  delay(1000);
}
```

Código 3.3. Código ejemplo actuación servo

Se observa que la estructura es la mencionada en el párrafo anterior. Se incluye la librería de "*Servo.h*". Se declara a continuación el servo. El siguiente paso es declarar el pin (PWM), en este caso es el pin 11. Por último, se mandan las instrucciones de mover el servo de 135 grados a 50 grados.

Esto es solo un código de ejemplo para comprender el funcionamiento del servo. En el código final el ángulo que se le manda al servo es la salida del PID que se verá en el apartado 3.3.4.

A continuación, se programa el sensor de espesor. El sensor de espesor es un sensor de efecto hall lineal. El sensor de efecto hall usado tiene 3 pines. Vcc, GND y data. Siendo Vcc la entrada de voltaje (5V), GND es la tierra, y data es el pin que da la salida del espesor. Este es un valor analógico que puede leerse con un pin analógico del Arduino. Véase el código de ejemplo a continuación.

```
/**
 * PINES
 */
const int sensor_diameter_pin = A4;    // seleccionar la entrada para el sensor
/**
 * SENSOR ESPESOR
 */
int sensor_diameter_value;       // variable que almacena el valor raw (0 a 1023)
bool debug = true;

void setup()
{
   Serial.begin(9600);
}

void loop()
{
   sensor_diameter_value = analogRead(sensor_diameter_pin);   // realizar la lectura
   if(debug)
   {
    Serial.print("Sensor : ");
    Serial.println(sensor_diameter_value);
    //Serial.println("\n");
   }
}
```

Código 3.4. Código toma de datos sensor espesor

El primer paso es declarar el pin necesario, en este caso se declara el pin analógico 4 (A4). A continuación, se declara una variable para almacenar el valor analógico de salida. Por último, en la función loop se realiza la lectura del sensor.

Este sensor tiene un defecto. El defecto es el ruido que puede tener el propio sensor, pero para corregir este ruido se pueden aplicar filtros. En este caso se ha usado un filtro de media. Véase en más detalle en el apartado 3.3.2.1.

Para continuar, se podría tomar varias muestras del sensor con diversos espesores controlados, para realizar una interpolación y conseguir la recta y así obtener el valor real del espesor. Es decir, crear una función que muestre el valor del espesor a partir del valor analógico. Pero no es necesario, ya que el PID puede actuar directamente sobe este valor analógico y de esta forma se ahorra tiempo. Ya que la idea es que tenga el menor retardo posible, porque el sistema ya tiene su propio retardo.

Por último, se programa el motor paso a paso encargado de mover el filamento de plástico. La programación de este es simple, tiene que dar un paso por cada ciclo de la función loop.

```
void avance_fil(bool  motor)
{
  if (motor){
    digitalWrite(Z_DIR_PIN , LOW);
    digitalWrite(Z_STEP_PIN , HIGH);
    delay(1);
    digitalWrite(Z_STEP_PIN , LOW);
  }
}
```

Código 3.5. Código de avance de un paso en motor paso a paso filamento

3.3.2.1. Filtro de media para el sensor de espesor

En este punto se detalla el funcionamiento de un filtro aplicado al sensor de espesor. El sensor de espesor tiene un ruido que se puede corregir con un filtro. Este filtro debe tener un coste computacional bajo. Para no producir mucho retardo en la señal.

Se ha decidido realizar un filtro de media, este filtro toma 15 medidas del sensor de espesor y calcula el valor medio de ese conjunto. Consiguiendo reducir bastante algunos picos que mostraba el sensor de espesor.

```
//SENSOR ESPESOR
bool debug_espesor = false; //Para mostrar las trazas
int num_muestra = 15;       //Numero de muestras para calcular la media del valor (filtro)
float espesor;              //Variable de salida del filtro
float media=0;              //Variable auxiliar, donde se almacena la media
int iterador = 0;           //Numero de muestra
float media_arr[15];        //Array donde se guardan los datos
float sensor_espesor()
{
  int sensor_diameter_value = analogRead(sensor_diameter_pin);   // realiza la lectura del sensor analógico

    if(debug_espesor) //Traza para ver el valor del sensor
    {
      Serial.print("Sensor : ");
      Serial.println(sensor_diameter_value);
    }

     if(iterador >= num_muestra)      //Si el el numero de muestra es mayor o igual al deseado (15)
     {
      for(int i=0;i<num_muestra;i++)//Suma todos los valores
      {
        media = media + media_arr[i];
      }
      media = media/num_muestra; //calculo la media
      espesor = media; // guarda la media en la variable de salida
      media = 0; //reset media
      iterador=0; //reset del iterador

      if(debug_espesor) //Traza para ver el valor medio del sensor
      {
        Serial.print("MEDIA Sensor MEDIA : ");
        Serial.println(espesor);
      }

    }

        else{ //Si el numero de muestra es menor que el deseado (15)
        media_arr[iterador]=sensor_diameter_value; //guardo en un array el valor
        iterador++;
        }
        return espesor; //devuelvo el valor medio
     }
```

Código 3.6. Código filtro de media

En la parte superior del código se declaran las variables necesarias. Como se aprecia, este programa recopila información del sensor y calcula la media aritmética para reducir el ruido de lectura. La media la calcula cuando ha tomado 15 datos. Una vez esta función haya sido llamada 15 veces se calcula el valor medio de esos 15 datos y devuelve como salida el valor calculado.

A partir de ese valor el controlador PID podrá actuar sobre el servomotor moldeando el filamento.

3.3.3. Programación etapa de bobinado

Para continuar, se desarrolla la programación de la tercera etapa, se recuerda que esta etapa es la encargada de bobinar el filamento en una bobina vacía. La tarea difícil de este

apartado no es la programación. Ya que es un motor paso a paso y la programación es sencilla, lo único que hay que hacer es que el motor de un paso por cada ciclo de la función loop. Por tanto, se crea una función que de un paso y el loop será el encargado de llamar a la siguiente función.

```
void avance_bob(bool motor)
{
  if (motor){ //Variable para activar o desactivar el motor
    digitalWrite(Y_DIR_PIN , LOW); //Se selecciona la direccion
    digitalWrite(Y_STEP_PIN , HIGH); //Se realiza el paso, primero con High
    delay(1);
    digitalWrite(Y_STEP_PIN , LOW); //Luego con Low
  }
}
```

Código 3.7. Código avance de un paso motor paso a paso bobina

El problema de esta tarea es la sincronización con el motor paso a paso de la segunda etapa. Puesto que, la velocidad lineal del filamento producida por el motor paso a paso de la segunda etapa es diferente a la velocidad lineal del motor paso a paso de la tercera etapa. Debido a que la velocidad lineal viene dada por la velocidad angular multiplicada por el radio de la bobina.

Por lo tanto, existen dos problemas, el primero que la velocidad con la que gira el motor de la segunda etapa no puede ser el mismo que la velocidad angular del motor de la tercera etapa. Y el otro problema que la velocidad lineal variará según pase el tiempo porque el radio variará en función lo que lleve bobinado.

El primer problema se puede resolver de manera práctica. Se hace funcionar ambos motores y se ajusta la velocidad angular de los motores. La velocidad angular se puede variar modificando la frecuencia con la que el motor da un paso.

La solución para el segundo problema es algo más complicado. Se podría medir la variación del radio a medida que va bobinando y se va ajustando la velocidad en función del uptime. Esta solución es válida pero algo compleja, porque hay que tener en cuenta varios factores como sería el uptime, resetear la velocidad angular cuando se coloque una nueva bobina, etc. Otra solución que se ha pensado es usar la misma velocidad angular en todo momento. Con un sistema en el engranaje de liberación de torque. Cuando el filamento se esté bobinando la velocidad lineal de la tercera etapa irá aumentando, pero si se implementa un muelle que evita dar un paso cuando la tensión del filamento sea muy grande el problema se soluciona de manera sencilla. Es cierto que no es la mejor solución, pero sí que es la solución más simple, efectiva y funcional. Por eso se ha escogido esa solución.

Figura 3.6. Mecanismo de liberación de tensión

3.3.4. Programación PID

El objetivo de este apartado es informar cómo se ha programado los dos controladores PID que existen en la máquina. Antes de comenzar hay que entender que es un controlador PID.

Un controlador PID es un mecanismo de control con una realimentación, gracias a la realimentación será capaz de calcular el error entre la salida y el valor deseado. Según el error el controlador maneja la variable de salida para llegar a un punto deseado.

El PID consta de tres partes una parte proporcional, una integral y otra derivativa. La parte proporcional depende del error actual, la integrativa depende del error pasado y la derivativa esta un error futuro. El sumatorio de estos tres errores de usa para controlar la variable de salida.

El controlador PID ha sido muy usado históricamente debido a su correcto funcionamiento y se ajusta bastante bien a cualquier situación. En este caso, en la bobinadora se han usado 2 controladores PID. Uno para el control de espesor y otro para el control de enfriamiento del filamento.

Antes de comenzar hay que conocer cómo actúan las constante proporcional, integrativa y derivativa en la salida del sistema.

La constante proporcional aumenta progresivamente la acción de control disminuyendo el error. Si el sistema se vuelve inestable antes de conseguir la salida deseada, se debe aumentar la constante derivativa.

La constante derivativa actúa de forma que si el sistema es inestable, se puede estabilizar el sistema aumentando el valor de la constante.

La constante integral reduce el error, siempre que este sea mayor al deseado. Si el error es mayor al requerido y la acción integral actuará progresivamente hasta minimizar el error.

Una vez conocidas las acciones de control se muestra la ecuación resultante que se va a implementar. Posteriormente se programa y se afina el controlador PID.

$$PID = K_p * er(t) + K_i \int_0^t er(t)dt + K_d * \frac{d(er(t))}{dt}$$

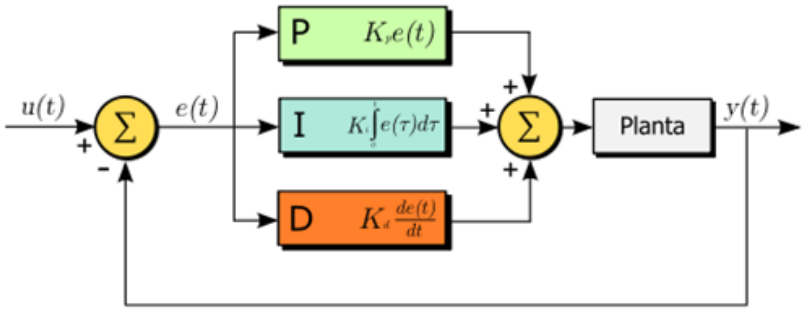


Figura 3.7. Representación PID.

Se comienza programando el PID que controla la temperatura, para ello se declaran todas las variables necesarias para el control.

Se necesitan:

- Una variable para calcular el periodo, para calcular posteriormente la acción derivativa.
- Una variable para almacenar el valor del sensor en este caso es *temperatura*.
- Variables para almacenar el error.
- Variables para aplicar las constantes (proporcional: Kp, integral: Ki, derivativa: Kd)
- Variable que guarde la salida deseada (*temperatura_setpoint*)
- Variables para guardar las acciones por separado (*PID_Temp_p, PID_Temp_i, PID_Temp_d*)
- Variable para calcular el sumatorio de las tres acciones
- Una variable para calcular la actuación del controlador (*PID_map)*

El siguiente paso es declarar en el código todas las variables necesarias.

```
//PID TEMPERATURA
unsigned long previousMillis_temp = 0; //Variable para el calculo del periodo

float temperatura = 0.0;
float temperatura_previous_error, temperatura_error;

float kp_temp=15;
float ki_temp=0.05;
float kd_temp=1;
float temperatura_setpoint = 25;  //Should be the temperatura

float PID_Temp_p, PID_Temp_i, PID_Temp_d;
float PID_Temp_total;
float PID_Temp_map;
```

Código 3.8. Variables necesarias para el control PID temperatura

Una vez declaradas las variables, se otorga un valor a las variables Kp, Kd y Ki. Posteriormente ese valor se calibrará y se cambiará para mejorar el funcionamiento del PID.

A continuación, se declara una función que devuelva el número de ventiladores que deben de estar activos para el correcto control de temperatura. En este caso debe de devolver 0 para activar cero ventiladores, 1 para un ventilador, y así hasta el máximo de ventiladores que es tres.

Los pasos que hay que seguir para programar el controlador son los siguientes:

1. Lectura del sensor de temperatura.
2. Cálculo del error.
3. Cálculo de la actuación proporcional. Se calcula como el producto de la constante Kp. por el error actual.
4. Cálculo del periodo, para posteriormente calcular la actuación derivativa.
5. Se halla la actuación derivativa con el producto de Kd por una estimación del error futuro ((error actual – error pasado) partido el periodo).
6. Se computa la actuación integral siempre que el error esté en un margen que no pueden corregir las actuaciones derivativa ni proporcional. Se calcula sumando a la actuación anterior el producto del error por la constante Ki.
7. A continuación, se suman todas las actuaciones.
8. Por último, se convierte esa actuación al número de ventiladores necesarios con la función "*map*" propia de Arduino. También se han limitado los valores, por si las actuaciones son muy grandes y se sale del rango establecido con el map.

```
temperatura = termometroIR.readObjectTempC();

temperatura_error = (temperatura_setpoint - temperatura);
PID_Temp_p = kp_temp * temperatura_error;
float period = currentMillis - previousMillis_temp;
PID_Temp_d = kd_temp*((temperatura_error - temperatura_previo

if(PID_Temp_map>=0 && PID_Temp_map<=3)
{
  if(-3 < temperatura_error && temperatura_error < 3)
  {
    PID_Temp_i = PID_Temp_i + (ki_temp * temperatura_error);
  }
  else
  {
    PID_Temp_i = 0;
  }
}

PID_Temp_total = PID_Temp_p + PID_Temp_i + PID_Temp_d;
PID_Temp_map = map(PID_Temp_total, -150, 3, 3, 0);

if(PID_Temp_map < 0){PID_Temp_map = 0;}
if(PID_Temp_map > 3) {PID_Temp_map = 3; }
```

Código 3.9. Código PID temperatura

A continuación, se actualizan las variables del error pasado y del tiempo. Para terminar la función devuelve el número de ventiladores necesarios.

```
temperatura_previous_error = temperatura_error;
previousMillis_temp = currentMillis;
return PID_Temp_map;
```

Código 3.10. Actualización de valores PID temperatura

Ahora es el turno de programar el PID del servomotor. Para realizar esta tarea se siguen los pasos anteriormente descritos.

Primero se declaran las variables necesarias para realizar el control.

```
//VARIABLES PARA EL PID SERVO
int period_servo; //variable para controlar el periodo de actualización del PID servo
float medida_espesor = 0.0; //variable donde se guarda el valor del espesor actual
float medida_espesor_previous_error, medida_espesor_error; //Variables para guardar los errores, anteripor y presente

float kp_servo=2; //Kp del servo
float ki_servo=0.05; //Ki del servo
float kd_servo=3; //Kd del servo
float medida_espesor_setpoint = 176;  //El valor deseado, que equivale a 1,75 mm de espesor

float PID_servo_p, PID_servo_i, PID_servo_d; //Almacena los valores P, I y D del controlador
float PID_servo_total; //Suma de los valores
float PID_servo_map; //Se conversion de la suma para obtener una salida
```

Código 3.11. Variables PID servo

El siguiente paso es replicar los pasos para crear la función del controlador PID.

```
////////////////////////////////////////////////////////////
//////////////////////PID SERVO/////////////////////////////
////////////////////////////////////////////////////////////

 void PID_Servo()
 {
    medida_espesor_error = (medida_espesor_setpoint - medida_espesor);
    PID_servo_p = kp_servo * medida_espesor_error;
    PID_servo_d = kd_servo*((medida_espesor_error - medida_espesor_previous_error)/period_servo);

    if(PID_servo_map>=50 && PID_servo_map<=135)
    {
      if(-3 < medida_espesor_error && medida_espesor_error < 3)
      {
        PID_servo_i = PID_servo_i + (ki_servo * medida_espesor_error);
      }
      else
      {
        PID_servo_i = 0;
      }
    }

    PID_servo_total = PID_servo_p + PID_servo_i + PID_servo_d;
    PID_servo_map = map(PID_servo_total, -50, 4, 50, 135);

    if(PID_servo_map < 50){PID_servo_map = 50;}
    if(PID_servo_map > 135) {PID_servo_map = 135; }
```

Código 3.12. Código PID servo

En esta función se ve la diferencia de que el mapa va de 50 a 135 que coinciden con los ángulos del servomotor. Cundo se le manda la señal de 135º al servo este se encuentra totalmente en horizontal sin ejercer ninguna tensión al filamento, si se manda la señal de 50º el eje pivotante se encontrará a 90º con respecto a la base, por tanto, estará ejerciendo la máxima tensión.

Por último, quedaría la calibración de ambos PID. Esta tarea se debe de realizar de forma experimental. Primero se ajustan todas las constantes a 0 (Kp, Ki y Kd). Posteriormente se incrementa la constante proporcional (Kp) hasta ver que está actuando de forma correcta ante la entrada. Posteriormente, se incrementa un poco la constante derivativa, no mucho porque puede hacer el sistema oscilante. Si el sistema se ajusta bien y no es oscilante es la hora de ajustar la constante integral. La constante integral se debe de aumentar muy poco, ya que tiene

bastante efecto. Va acumulando según los ciclos que haga el controlador, por tanto, no puede ser muy grande.

Se ha mencionado que la calibración se ha hecho experimentalmente. El sensor de temperatura se ha calibrado introduciendo un pequeño filamento a muy alta temperatura. El sensor de espesor ha sido el más difícil de ajustar, puesto que la máquina extrusora no está terminada. Lo suyo es calibrarla con la extrusora lista, para ver cómo actúa experimentalmente. Se ha ajustado de otra forma la máquina, se ha cogido una bobina de filamento de mayor espesor, 2.85mm y se calentado con un mechero para ver la actuación del servomotor. De esta manera se han calibrado los parámetros del PID. Es cierto que para mejorar la calidad del bobinado se deberá de ajustar mejor los parámetros cuando la máquina extrusora esta lista y funcionando.

3.3.5. Programación conjunta y esquemático

En este apartado se unirán todos los códigos de las tres estaciones y se realizará un calibrado para garantizar su correcto funcionamiento.

```
/*
 * LIBRERIAS
 */
#include <Wire.h>
#include <Servo.h>
#include <Adafruit_MLX90614.h>

/**
 * Constantes
 */
Servo myservo;  // create servo object to control a servo
Adafruit_MLX90614 termometroIR = Adafruit_MLX90614(); //Se instancia el objeto

/*
 * VARIABLES GLOBALES
 */

//VARIABLES PARA EL PID SERVO
int period_servo; //variable para controlar el periodo de actualización del PID servo
float medida_espesor = 0.0; //variable donde se guarda el valor del espesor actual
float medida_espesor_previous_error, medida_espesor_error; //Variables para guardar los errores, anteripor y presente

float kp_servo=2; //Kp del servo
float ki_servo=0.05; //Ki del servo
float kd_servo=3; //Kd del servo
float medida_espesor_setpoint = 176;  //El valor deseado, que equivale a 1,75 mm de espesor

float PID_servo_p, PID_servo_i, PID_servo_d; //Almacena los valores P, I y D del controlador
float PID_servo_total; //Suma de los valores
float PID_servo_map; //Se conversion de la suma para obtener una salida


//VELOCIDAD DE LOS MOTOERES PASO A PASO
unsigned long previousMillis_fil = 0; //Variable para el calculo de tiempo
unsigned long previousMillis_bob = 0; //Variable para el calculo de tiempo

int velocidad_fil = 30; //Velocidad del filamento
int velocidad_bob = 290; //Velocidad de la bobina. Se recomiend que velocidad_bob = 10 * velocidad_fil

//SENSOR ESPESOR
int num_muestra = 15;       //Numero de muestras para calcular la media del valor (filtro)
float espesor;              //Variable de salida del filtro
float media=0;              //Variable auxiliar, donde se almacena la media
int iterador = 0;           //Numero de muestra
float media_arr[15];        //Array donde se guardan los datos


//PID TEMPERATURA
unsigned long previousMillis_temp = 0; //Variable para el calculo del periodo

float temperatura = 0.0;
float temperatura_previous_error, temperatura_error;
```

```
float kp_temp=15;
float ki_temp=0.05;
float kd_temp=1;
float temperatura_setpoint = 25;  //Should be the temperatura

float PID_Temp_p, PID_Temp_i, PID_Temp_d;
float PID_Temp_total;
float PID_Temp_map;

/**
 * TRAZAS
 */
bool debug_espesor = false;  //SENSOR ESPESOR
bool debug_PID_servo = false; //PID SERVO
bool debug_PID_temp = true;  //PID TEMPERATURA

/**
 * Definicion De Pines
 */
#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

#define Z_STEP_PIN 46
#define Z_DIR_PIN 48
#define Z_ENABLE_PIN 62
#define Z_MIN_PIN 18
#define Z_MAX_PIN 19

#define sensor_diameter_pin A4

const int Fans[3] = {16,17,23};// the number of the FAN pin

/**
 * FUNCION SETUP
 */
void setup()
{
 pinMode(Z_STEP_PIN , OUTPUT);
 pinMode(Z_DIR_PIN , OUTPUT);
 pinMode(Z_ENABLE_PIN , OUTPUT);
 pinMode(Y_STEP_PIN , OUTPUT);
 pinMode(Y_DIR_PIN , OUTPUT);
 pinMode(Y_ENABLE_PIN , OUTPUT);

 digitalWrite(Z_ENABLE_PIN , LOW);
 digitalWrite(Y_ENABLE_PIN , LOW);

 pinMode(Fans[0], OUTPUT);
 pinMode(Fans[1], OUTPUT);
 pinMode(Fans[2], OUTPUT);

 termometroIR.begin(); // Iniciar termómetro infrarrojo con Arduino


 myservo.attach(11);  // attaches the servo on pin 9 to the servo object
 myservo.write(60);
 Serial.begin(115200);
 if(debug_PID_servo)
 Serial.println("Referencia Medida_espesor Error PID_servo_p PID_servo_i PID_servo_d Actuacion");
 if(debug_PID_temp)
 Serial.println("Referencia Temperatura Error PID_temp_p PID_temp_i PID_temp_d Actuacion");
}

/**
 * FUNCIONES AUXILIARES
 */

float sensor_espesor(); //Toma la medida del espesor
void avance_fil(bool motor); //Avanza un paso el motor del filamento
void avance_bob(bool motor); //Avanza un paso el motor de la bobina
void PID_Servo(); //PID del servomotor
int PID_Temp(unsigned long currentMillis); //PID de los ventiladores, devuelve el num de ventiladores activos
void ventiladores(int mapa); //Enciende los ventiladores segun el numero de ventiladores necesarios

/*
 * FUNCION PRINCIPAL LOOP
 */
void loop ()
{
 unsigned long currentMillis = millis();
```

```
  if (currentMillis - previousMillis_fil >= velocidad_fil) {
    period_servo = velocidad_fil;
    previousMillis_fil = currentMillis;
    avance_fil(true);
    medida_espesor = sensor_espesor();
    PID_Servo();
  }

  if (currentMillis - previousMillis_bob >= velocidad_bob) {
    previousMillis_bob = currentMillis;
    avance_bob(true);
  }

  int mapa = PID_Temp(currentMillis);
  ventiladores(mapa);

}


/*
 * FUNCIONES AUXILIARES
 */

////////////////////////////////////////////////////////////
//////////////////////VENTILADORES//////////////////////////
////////////////////////////////////////////////////////////
void ventiladores(int mapa)
{
  switch (mapa) {
   case 0:
     digitalWrite(Fans[0], LOW);
     digitalWrite(Fans[1], LOW);
     digitalWrite(Fans[2], LOW);
     break;

   case 1:
     digitalWrite(Fans[0], LOW);
     digitalWrite(Fans[1], HIGH);
     digitalWrite(Fans[2], LOW);
     break;

   case 2:
     digitalWrite(Fans[0], HIGH);
     digitalWrite(Fans[1], LOW);
     digitalWrite(Fans[2], HIGH);
     break;

   case 3:
     digitalWrite(Fans[0], HIGH);
     digitalWrite(Fans[1], HIGH);
     digitalWrite(Fans[2], HIGH);
     break;
  }
}

////////////////////////////////////////////////////////////
//////////////////////PID VENTILADORES//////////////////////
////////////////////////////////////////////////////////////
int PID_Temp(unsigned long currentMillis)
{

  temperatura = termometroIR.readObjectTempC();

  temperatura_error = (temperatura_setpoint - temperatura);
  PID_Temp_p = kp_temp * temperatura_error;
  float period = currentMillis - previousMillis_temp;
  PID_Temp_d = kd_temp*((temperatura_error - temperatura_previous_error)/period);

  if(PID_Temp_map>=0 && PID_Temp_map<=3)
  {
    if(-3 < temperatura_error && temperatura_error < 3)
    {
      PID_Temp_i = PID_Temp_i + (ki_temp * temperatura_error);
    }
    else
    {
      PID_Temp_i = 0;
    }
  }

  PID_Temp_total = PID_Temp_p + PID_Temp_i + PID_Temp_d;
  PID_Temp_map = map(PID_Temp_total, -150, 3, 3, 0);
```

```
  if(PID_Temp_map < 0){PID_Temp_map = 0;}
  if(PID_Temp_map > 3) {PID_Temp_map = 3; }

  if(debug_PID_temp)
  {
    Serial.print(temperatura_setpoint);
    Serial.print(" ");
    Serial.print(temperatura);
    Serial.print(" ");
    Serial.print(PID_Temp_p);
    Serial.print(" ");
    Serial.print(PID_Temp_i);
    Serial.print(" ");
    Serial.print(PID_Temp_d);
    Serial.print(" ");
    Serial.println(PID_Temp_map);
  }

  temperatura_previous_error = temperatura_error;
  previousMillis_temp = currentMillis;
  return PID_Temp_map;
}

//////////////////////////////////////////////////////////
//////////////////////PID SERVO//////////////////////////
//////////////////////////////////////////////////////////

 void PID_Servo()
 {
    medida_espesor_error = (medida_espesor_setpoint - medida_espesor);
    PID_servo_p = kp_servo * medida_espesor_error;
    PID_servo_d = kd_servo*((medida_espesor_error - medida_espesor_previous_error)/period_servo);

    if(PID_servo_map>=50 && PID_servo_map<=135)
    {
      if(-3 < medida_espesor_error && medida_espesor_error < 3)
      {
        PID_servo_i = PID_servo_i + (ki_servo * medida_espesor_error);
      }
      else
      {
        PID_servo_i = 0;
      }
    }

    PID_servo_total = PID_servo_p + PID_servo_i + PID_servo_d;
    PID_servo_map = map(PID_servo_total, -50, 4, 50, 135);

    if(PID_servo_map < 50){PID_servo_map = 50;}
    if(PID_servo_map > 135) {PID_servo_map = 135; }

    if(debug_PID_servo)
    {
      Serial.print(medida_espesor_setpoint);
      Serial.print(" ");
      Serial.print(medida_espesor);
      Serial.print(" ");
      Serial.print(medida_espesor-medida_espesor_setpoint);
      Serial.print(" ");
      Serial.print(PID_servo_p);
      Serial.print(" ");
      Serial.print(PID_servo_i);
      Serial.print(" ");
      Serial.print(PID_servo_d);
      Serial.print(" ");
      Serial.println(PID_servo_map);
    }

    myservo.write(PID_servo_map);
    medida_espesor_previous_error = medida_espesor_error;
 }

//////////////////////////////////////////////////////////
//////////////LECTURA SENSOR ESPESOR//////////////////////
//////////////////////////////////////////////////////////
```

```
float sensor_espesor()
{
  int sensor_diameter_value = analogRead(sensor_diameter_pin);   // realiza la lectura del sensor analógico

    if(debug_espesor) //Traza para ver el valor del sensor
    {
      Serial.print("Sensor : ");
      Serial.println(sensor_diameter_value);
    }

     if(iterador >= num_muestra)      //Si el el numero de muestra es mayor o igual al deseado (15)
     {
      for(int i=0;i<num_muestra;i++)//Suma todos los valores
      {
        media = media + media_arr[i];
      }
      media = media/num_muestra; //calculo la media
      espesor = media; // guarda la media en la variable de salida
      media = 0; //reset media
      iterador=0; //reset del iterador

      if(debug_espesor) //Traza para ver el valor medio del sensor
      {
        Serial.print("MEDIA Sensor MEDIA : ");
        Serial.println(espesor);
      }


    }
    else{ //Si el numero de muestra es menor que el deseado (15)
    media_arr[iterador]=sensor_diameter_value; //guardo en un array el valor
    iterador++;
    }
    return espesor; //devuelvo el valor medio
}

//////////////////////////////////////////////////////////
//////////////////////AVANZA UN PASO//////////////////////
//////////////////////////////////////////////////////////
void avance_fil(bool  motor)
{
  if (motor){
    digitalWrite(Z_DIR_PIN , LOW);
    digitalWrite(Z_STEP_PIN , HIGH);
    delay(1);
    digitalWrite(Z_STEP_PIN , LOW);
  }
}

void avance_bob(bool motor)
{
  if (motor){
    digitalWrite(Y_DIR_PIN , LOW);
    digitalWrite(Y_STEP_PIN , HIGH);
    delay(1);
    digitalWrite(Y_STEP_PIN , LOW);
  }
}
```

Código 3.13. Código completo bobinadora

Este es el código completo de la máquina que se encarga de bobinar el filamento. Se ha estructurado en varias partes para que sea más simple entenderlo. En la parte superior se observa la declaración de las variables, donde se pueden alterar los parámetros del PID, la velocidad de los motores, etc. Posteriormente se encuentra la declaración de los pines físicos. A conituación se encuentra la función setup, que es la unión de las funciones setup de los códigos de ejemplo. Se inicializa la Ramps para el control de los motores paso a paso, se inicializa el sensor de temperatura y se inicializa el servomotor. Después, se encuentra la declaración de funciones auxiliares que se han usado para realizar determinadas tareas. Como por ejemplo, control PID del servo o de los ventiladores, etc. y así se consigue dejar la función loop más limpia y fácil de entender.

Por último se encuentre la función loop, esta es la función principal, se encarga de llamar a las funciones auxiliares en el momento oportuno. Debajo de la función loop se encuentran definidas las funciones auxiliares. Estas funciones coinciden con las funciones vistas en los ejemplos o funcionan de una forma similar.

Una vez visto el código, ahora hay que conocer las conexiones físicas que se han implementado. Para ello se ha diseñado un esquemático con la herramienta Fritzing, que simula las conexiones entre los componentes y el controlador. Vease las conexiones en la figura 3.8.

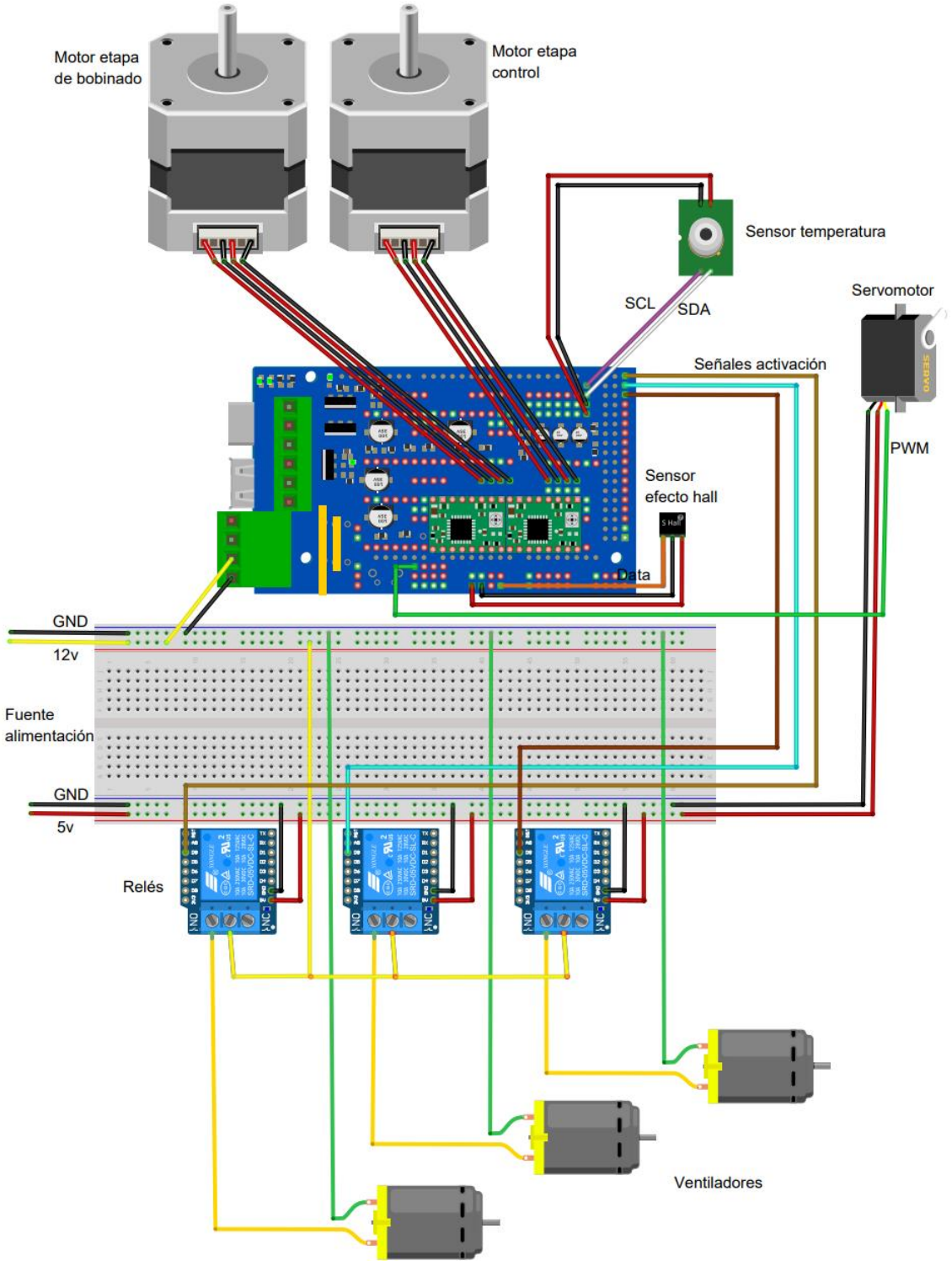


Figura 3.8. Esquema de conexiones físicas

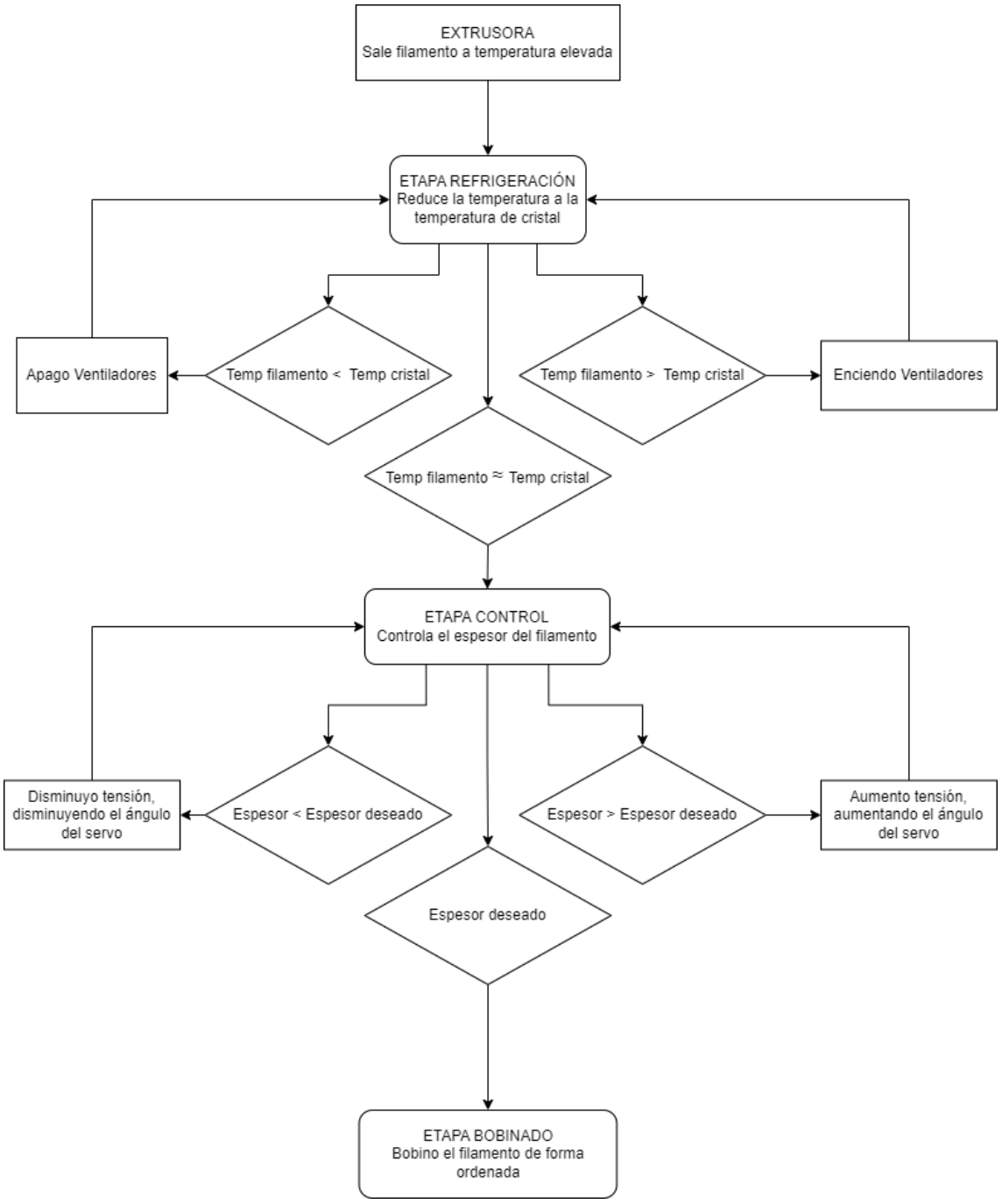


Figura 3.9. Diagrama de flujo.

CAPÍTULO 4. PLANOS

4. Planos

En este capítulo se adjuntan los planos de los diseños realizados para la parte mecánica. Los planos se han realizado mediante la herramienta solidworks.

Las piezas que se muestran a continuación han sido impresas en una impresora 3D con el material PETG.

El material PETG es un material plástico resistente y barato, por eso se ha seleccionado este material. Además, es más resistente a temperaturas que otros plásticos como por ejemplo el PLA y es bastante fácil de usarlo en impresoras 3D. No se necesitan ningún complemento ni es tóxico a la hora de imprimir.

CAPÍTULO 5. CONCLUSIONES Y EXPERIMENTOS

5.1. Experimentos realizados

Una vez montada la máquina es turno de realizar experimentos y ponerla a prueba. El problema principal es que no se ha podido probar con la extrusora de forma directa. No obstante, se ha realizado un experimento que simula la extrusora.

El experimento consiste es convertir un filamento de mayor grosor (2,85mm) que se usaba en impresoras 3D antiguas y transformar el grosor del filamento en un grosor normalizado (1,75mm). Para ello se ha calentado el filamento con un soplete a la potencia mínima. Posteriormente, se introduce el filamento en la bobinadora empezando por la etapa de refrigeración y sigue el proceso por las dos etapas restantes.

De esta manera se han realizado las calibraciones para los controladores PID, aunque se deberá de terminar de ajustar cuando la extrusora esté en funcionamiento. Mientras tanto se ha realizado el experimento mencionado anteriormente.

Se ha obtenido un buen resultado, se ha conseguido transformar el filamento de 2,85mm en filamento de 1,75mm con una buena precisión. Se ha medido el resultado y se ha obtenido un grosor máximo de 1,82 y un grosor mínimo de 1,73mm, se tiene un error de ± 0,09.

Las bobinas comerciales tienen un error de ± 0,05. Por tanto, se puede obtener un buen resultado con una buena calibración.

5.2. Conclusiones sobre la evaluación económica

Después de investigar e indagar en el tema de reciclado de plásticos, se puede sacar de conclusión que las máquinas para un uso más domestico son casi inexistentes. Siendo de precio muy elevado y de grandes dimensiones.

Analizando los resultados y la actuación de la bobinadora, se puede concluir que funciona bastante bien para su precio. Puesto que gran parte de los materiales usados son reciclados y esto hace que el coste material sea bastante bajo. Si se comparasen el funcionamiento de las máquinas ya diseñadas y de precio elevado, se vería poca diferencia. Es cierto, que la calidad de la bobina no será la más optima, pero si será aceptable y suficiente para la impresión 3D.

Analizando las etapas por separado. La primera etapa la constituyen 3 ventiladores con 3 relés más el sensor de temperatura, siendo los ventiladores reciclados. Se obtiene un coste de 58,33€.

La segunda etapa, es algo más costosa, debido a que tiene un servomotor, aunque este ha sido reciclado ya que lo ha proporcionado mí tutor. Una varilla roscada, proporcionada por la universidad, con un costo inferior a los 5€. Un sensor de efecto hall, que ha sido proporcionado por mí tutor y un imán, estos componentes no superan el precio de 1-2€. Por último, un motor paso a paso con un soporte para mover filamento 3D, este ha sido comprado por la universidad, costando 8€ aproximadamente. En total es coste aproximado de la segunda etapa es de: 12€

La última etapa la constituyen otro motor paso a paso con un coste de 8€ y 4 varillas roscadas recicladas, pero el precio de las varillas no superara los 15€.

Hay que sumar también las piezas impresas. Se puede calcular el precio por el peso de las piezas, siendo este inferior a 1Kg. Aproximadamente se ha usado 800 g de plástico (PETG). Teniendo un costo de 18€/kg. Por tanto, el precio de todas las piezas aumentaría a un total de 14,4€.

Además, hay que tener en cuenta el coste del controlador, de la fuente de alimentación y de interruptores para controlar el encendido y apagado de la propia máquina. El controlador es un Arduino con una ramps. El pack no supera el precio de 35€. La fuente de alimentación es una vieja reciclada de un ordenador. Pero estimando el valor, en el mercado de segunda mano debe rondar 5-10€. Los interruptores, se pueden reciclar perfectamente de juguetes u otros dispositivos obsoletos, se puede despreciar el coste de los interruptores.

Todo esto hace un total de 135€, comparando este precio con las máquinas más desarrolladas y que se comercializan. Rondando los 3000-5000€, se puede ver que es un precio bastante competente. Aunque el resultado sea ligeramente de peor calidad, el ahorro es muy alto.

5.3. Conclusiones sobre la programación

La programación de la bobinadora no ha sido tarea fácil. Debido a diversos problemas que han ido surgiendo a la hora del montaje y de la calibración. Sobre todo, a la hora de sincronizar el código de las tres etapas.

Pero después de un par de ajustes y cambios, el código ha quedado bastante limpio y comprensible. Evitando el uso de bucles for, delays, que son malas prácticas ya que ralentiza la velocidad de los ciclos.

Además, ha sido bastante entretenido la programación, ya que he aprendido bastante y he usado diferentes tecnologías, por ejemplo, bus I2C, diferentes sensores, uso de PID con retardo, etc. He adquirido nuevos conocimientos y aprendido durante la elaboración del proyecto.

5.4. Conclusiones generales

Como conclusiones generales, el proyecto ha sido bastante interesante además he adquirido bastantes conocimientos sobre las diferentes tecnologías y controles aplicados. Lo más interesante es ver como el prototipo es capaz de actuar de manera correcta.

Es cierto que la calidad de la bobina no es como una comprada. Pero si que es funcional. Personalmente pensaba que el reciclado de plástico no estaría al alcance de cualquier persona. Pero indagando e investigando he cambiado totalmente de opinión. Reciclar los residuos es posible y a bajo costo, no hace falta máquinas que valen miles de euros.

El desarrollo de la máquina ha sido algo complejo y costoso, pero una vez visto los resultados merece la pena. Es una inversión, en casa tengo una impresora 3D y se pierde bastante plástico a la hora de imprimir los soportes, piezas fallidas, etc. Esto tiene un coste, pero con esta máquina junto a la de los compañeros es posible convertir los restos en una bobina de nuevo, por lo tanto, no existen perdidas.

El desarrollo ha sido complejo debido a la gran cantidad de problemas que han ido surgiendo durante el montaje. Se ha tenido que cambiar el diseño del modelo varias veces, teniendo que volver a imprimir piezas en 3D, por tanto, pérdida de tiempo. Algunas piezas han tardado más de 11 horas en imprimirse, por su relleno y su tamaño.

Pero una vez solventado los problemas surgidos, el proceso de calibrado ha sido simple, a base de pruebas prácticas. Ha sido curioso ver cómo actúa el control PID del sensor de

espesor. Porque, para su calibración se han ido variando los parámetros, viendo cómo actúa cada parámetro en la respuesta. Esto ha enriquecido bastante mi conocimiento sobre los controles realimentados.

BIBLIOGRAFÍA

[1] *Blog – Mareaplastic*. (s. f.). Recuperado 14 de junio de 2022, de https://www.mareaplastic.uma.es/blog/

[2] BASURAS MARINAS. (s. f.). *AMBIENTE EUROPEO*. Recuperado 14 de junio de 2022, de https://ambienteeuropeo.org/basuras-marinas/

[3] Rigatos, G., Zervos, N., Siano, P., Abbaszadeh, M., & Wira, P. (2019). Non-linear optimal control for the hot-steel rolling mill system. *IET Collaborative Intelligent Manufacturing*, *1*(3), 97-107. https://doi.org/10.1049/iet-cim.2019.0010

[4] Thingiverse.com. (s. f.). *Dial indicator bracket, Filament Diameter measurement by newcrz*. Recuperado 1 de noviembre de 2022, de https://www.thingiverse.com/thing:4835232

[5] *Welcome to InFiDEL's documentation! —InFiDEL <> documentation*. (s. f.). Recuperado 8 de octubre de 2022, de https://infidel-sensor.readthedocs.io/en/latest/

[6] Arduino MEGA Shield—RAMPS. (s. f.). *DomoticX Knowledge Center*. Recuperado 18 de octubre de 2022, de http://domoticx.com/arduino-mega-shield-ramps/

[7] *DESCRIPCIÓN DEL DRIVER A4988 – Electrónica Práctica Aplicada*. (s. f.). Recuperado 1 de noviembre de 2022, de https://www.diarioelectronicohoy.com/blog/descripcion-del-driver-a4988

[8] Espeso, A. (2015, agosto 18). *Control PID de Barra y Bola con Arduino*. Estudio Roble. https://roble.uno/control-pid-barra-y-bola-arduino/

[9] Filament diameter sensor. (s. f.). YouMagine. Recuperado 25 de octubre de 2022, de https://www.youmagine.com/designs/filament-diameter-sensor

[10] GP. (s. f.). Greenpeace España. Recuperado 14 de junio de 2022, de http://archivo-es.greenpeace.org/espana/es/

[11] Meet the Composer and Precision—Desktop Filament Makers | 3devo. (s. f.). Recuperado 14 de junio de 2022, de https://www.3devo.com/filament-makers

[12] Noztek Filament Winder 2.0—Noztek Extrusion System. (s. f.). Recuperado 14 de junio de 2022, de https://noztek.com/product/filament-winder-2/

[13] Precious Plastic Basic Machines. (s. f.). Recuperado 14 de junio de 2022, de https://preciousplastic.com/solutions/machines/basic.html

[14] Tolerance Puller. (s. f.). *Noztek Extrusion System*. Recuperado 14 de junio de 2022, de https://noztek.com/product/tolerance-puller/

[15] YouMagine – InFiDEL - Inline Filament Diameter Estimator (lowcost) by Thomas Sanladerer – YouMagine 🔧. (s. f.). Recuperado 14 de junio de 2022, de https://www.youmagine.com/designs/infidel-inline-filament-diameter-estimator-lowcost-10-24

[16] Arduino PID ping pong balance servo IR distance sensor. (s. f.). Recuperado 25 de octubre de 2022, de http://www.electronoobs.com/eng_arduino_tut100.php

[17] Controlador PID - Control Automático—Picuino. (s. f.). Recuperado 18 de octubre de 2022, de https://www.picuino.com/es/control-pid.html