



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Controlador de una máquina de ensayos biaxial

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Pérez González, Ángel

Tutor/a: Sánchez López, Miguel

CURSO ACADÉMICO: 2021/2022

Resumen

El proyecto consiste en el desarrollo, por medio del microcontrolador basado en Arduino, de un sistema de control de una máquina de ensayos biaxial para tejidos de lienzos. Para ello se desarrolla la propia máquina con actuadores y sensores, el controlador para el sistema y la interfaz de usuario para la comunicación con el microcontrolador.

Palabras clave: Arduino, biaxial, tejidos, controlador, sistema, pruebas.

Resum

El projecte consisteix en el desenvolupament, mitjançant el microcontrolador Arduino, d'un sistema de control de una màquina de assatjos biaxial per teixits de quadres. Per aixó es desenvolupa la mateixa màquina amb actuadors i sensors, el controlador per el sistema i l'interfície d'usuari per a la comunicació amb el microcontrolador.

Paraules clau : Arduino, biaxial, teixits, controlador, sistema, proves.

Abstract

The project consists of the development, by means of the microcontroller based on Arduino, of a control system for a biaxial testing machine for canvas fabrics. For this purpose, the machine itself is developed with actuators and sensors, the controller for the system and the user interface for communication with the microcontroller.

Keywords : Arduino, biaxial, fabrics, controller, system, testing.

Tabla de contenidos

1.Introducción	8
1.1.Motivación	8
1.2.Objetivos	8
1.3.Estructura	9
2.Estado del arte	11
2.1.Máquinas de ensayos biaxiales	11
2.2. Máquinas de ensayos biaxiales comerciales	12
2.3.Máquinas de ensayos biaxiales no comerciales	15
3.Análisis del problema	18
4.Solución propuesta	21
4.1. Principales componentes del sistema	21
4.2.Plan de trabajo	22
5.Diseño de la solución	25
5.1.Microcontrolador	25
5.2.Programación del microcontrolador	26
5.3.Actuadores del sistema	28
5.3.1.Motores Nema 17	28
5.3.2.Controladores de los actuadores	30
5.4.Sensores del sistema	31
5.4.1.Células de carga	31
5.4.2.Módulo analógico-digital	33
6.Implementación de la solución	36
6.1.Cableado de los actuadores	36
6.2.Cableado de los sensores	39



6.3. Prototipo de máquina de pruebas	41
6.4. Firmware del Arduino	43
6.4.1. Inicialización de las variables y librerías	43
6.4.2. Setup del programa	44
6.4.3. Loop principal del programa	45
6.5. Software del programa	46
6.5.1. Comunicación serial en Java	50
6.5.2. Funcionamiento de los archivos mcode	50
6.6. Visión general del sistema	51
7. Pruebas realizadas	54
8. Conclusiones finales	57
9. Referencias	59

1. Introducción

A lo largo de este documento se van a describir las características del proyecto basado en Arduino para el control de una máquina de ensayos biaxial de telas de lienzos. Además, se va a hacer un estudio de la situación actual y el por qué de realizar el desarrollo de este trabajo.

1.1.Motivación

Hoy en día la profesionalización de muchos sectores ha llevado a que cada vez más se utilicen máquinas de pruebas y tecnologías. Estos sectores se ven beneficiados por las ventajas que les ofrecen este tipo de máquinas que cada vez son más asequibles de conseguir.

Máquinas que antes eran utilizadas por grandes empresas debido a su alto coste, ahora llegan para quedarse en sectores donde antes ni se planteaba o incluso en las casas de la gente. Este es el caso de las máquinas de mecanizado CNC o impresoras 3D, que abren las puertas a los artesanos a un prototipado, restauración o fabricación más sencillo.

Es por esto que se va a desarrollar un sistema de pruebas basado en las tecnologías de los elementos mencionados que permita a artesanos obtener información de los materiales que van a usar en sus proyectos, ayudando de esta forma a una fabricación o restauración más eficiente y profesional.

1.2.Objetivos

Los principales objetivos que se pretenden conseguir en el proyecto son los siguientes:

- Desarrollo de un código abierto para controlar los actuadores y sensores del sistema basado en Arduino que permite la tracción de materiales y su posterior obtención de datos.
- Creación de una interfaz fácil de usar para gente no familiarizada con el campo de la informática que cree la gráfica en tiempo real y obtenga los valores requeridos de forma sencilla de visualizar.

- Obtener un sistema que se pueda conseguir con pocos medios con la mayor parte de los componentes accesibles económicamente o reutilizados de máquinas que no se usen.
- Hacer uso de los conocimientos obtenidos en la rama de Ingeniería de Computadores como la programación de sistemas empujados y de tiempo real y en la carrera como la creación de interfaces.

1.3. Estructura

La estructura del proyecto se va a dividir en varios pasos hasta la fase final en la que se obtienen las conclusiones.

Las fases en las que se va a dividir van a ser las siguientes ocho fases:

Primero se analizará el estado del arte, es decir, el contexto en el que se encuentra nuestro proyecto y cuáles son las otras opciones que se pueden encontrar en el mercado.

Acto seguido se hará un análisis del problema para determinar cómo se puede obtener una solución mejor a las que hay.

Después se hará una propuesta de solución y acto seguido se desarrollará el diseño de la solución en el que se detalla todo el proyecto desde un punto de vista de arquitectura y componentes.

La parte final del proyecto consta de las fases de implementación de la solución, las pruebas realizadas después de la implementación y por último las conclusiones del proyecto como cierre del documento.



2. Estado del arte

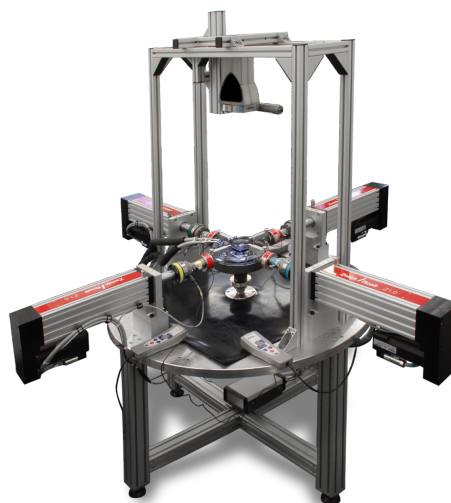
A lo largo de este capítulo se van a analizar las propuestas ya presentes en el mercado actual que obtienen unos resultados similares a los esperados para este proyecto y se analizarán las ventajas de nuestro proyecto sobre estos.

2.1. Máquinas de ensayos biaxiales

A lo largo de los años y con la mejora de las tecnologías se han ido desarrollando máquinas capaces de realizar pruebas sobre materiales para comprobar sus propiedades mecánicas. Estas máquinas han sido empleadas por compañías grandes con necesidades de comprobar la durabilidad de sus productos, siendo una parte importante en el desarrollo de prototipos.

Este tipo de máquinas son hoy en día muy usadas en conjunto con los simuladores de materiales para determinar las propiedades de los materiales, aunque existen máquinas similares para un eje y hasta tres ejes.

El funcionamiento de estos aparatos, en concreto el biaxial, es el de traccionar materiales planos y comprobar mediante sus sensores de fuerza, también llamadas galgas, la fuerza que soportan los materiales en los dos ejes de tracción.



1

¹ Figura 1. Máquina de ensayos biaxial cruciforme.

Cabe destacar que este tipo de pruebas se están llevando a cabo últimamente también en el ámbito de los tejidos biológicos. Estas pruebas proporcionan un entorno de pruebas sencillo en el que los investigadores pueden poner a prueba varios materiales de forma sencilla.

En nuestro caso, la necesidad de este tipo de pruebas viene determinado por la necesidad de dotar a los profesionales de la restauración o artesanos la capacidad de realizar pruebas sobre lienzos para determinar como van a alterar los esfuerzos al lienzo que, en determinadas ocasiones, puede romperse por estrés. Al ser el lienzo una tela plana, una máquina de ensayos biaxial es perfecta para este cometido.

2.2. Máquinas de ensayos biaxiales comerciales

En cuanto a las variables comerciales que podemos encontrar en el mercado, se pueden encontrar varias empresas dedicadas a la fabricación de estas. Este tipo de máquinas están diseñadas para aplicar unas fuerzas sobre el material capaces de romper materiales muy resistentes.

Para ello suelen utilizar cilindros neumáticos o hidráulicos muy costosos de mantener y operar con los que no se puede conseguir tanta precisión en el desplazamiento.

Es por esto que para un uso en materiales que requieren menos precisión pero fuerzas mayores para el test, es la opción más viable.

También hay opciones para materiales orgánicos, ya que sus propiedades son muy distintas a las de los materiales convencionales y necesitan un entorno más controlado. Estas máquinas suelen ser más reducidas y con más modos que las mencionadas anteriormente.

Algunas de las opciones comerciales de las máquinas de ensayos biaxiales que podemos encontrar en el mercado con características interesantes a tener en cuenta son las siguientes:

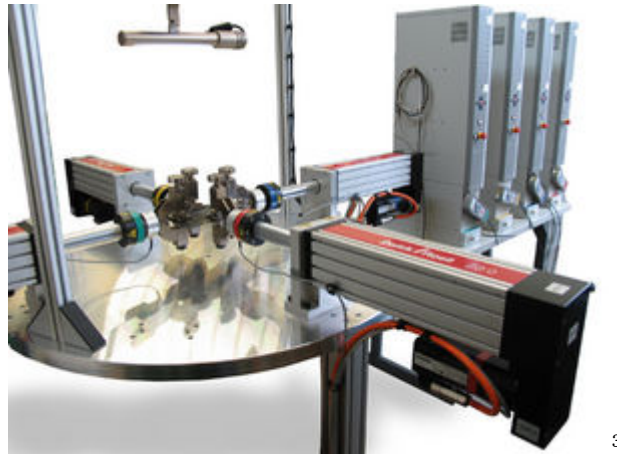
- **BioTester-CellScale.** Esta máquina de ensayos está diseñada para su uso en materiales orgánicos de forma sencilla. Proporciona un entorno controlado para este tipo de pruebas y el software está realizado para que sea sencillo su uso. El software proporciona varios modos de uso como el modo simple, cíclico o multimodal, todos ellos con procesamiento de datos en tiempo real.



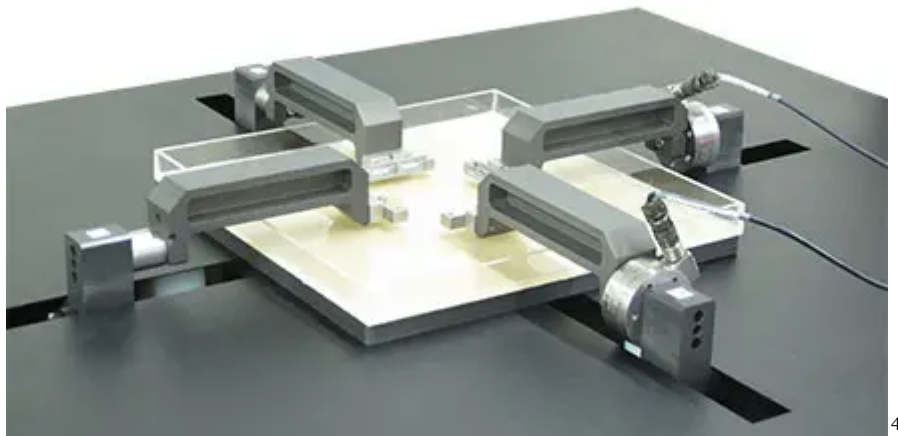
2

- **Biaxial Tester - ZwickRoell.** Esta empresa se dedica íntegramente al desarrollo de estas máquinas a medida del cliente. Ofrecen unas fuerzas máximas de tracción de 2 kN hasta 250kN. En estas máquinas se emplean actuadores electromecánicos que son sencillos de controlar por medio de controladores industriales. En cuanto al software, proporciona una interfaz con un editor secuencial gráfico para el diseño de las pruebas.

² Figura 2. Máquina BioTester de la empresa Cell Scale.



- **eXpert 8000-Admet.** El entorno de pruebas que ofrece la empresa Admet se caracteriza por su fiabilidad para pruebas de materiales menos resistentes. Proporciona una fuerza de tracción de hasta 5 kN. El software que implementa es capaz de tener en cuenta las distintas variables de la prueba como la fuerza, el desplazamiento y la tensión. También es capaz de seguir patrones cíclicos con una forma de onda o mover cada brazo independientemente.



³ Figura 3. Máquina de ensayos de la empresa Zwick Roell.

⁴ Figura 4. Máquina eXpert 8000 de la empresa Admet.

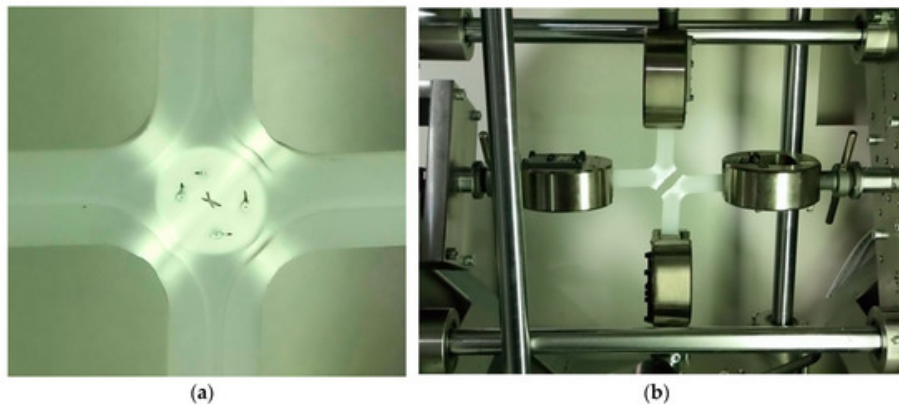
2.3. Máquinas de ensayos biaxiales no comerciales

En este apartado, se van a exponer las opciones existentes fuera del ámbito comercial, es decir, máquinas que se hayan realizado por estudios o por ser un proyecto privado que no está a la venta.

Cabe destacar que estas opciones son más difíciles de encontrar, ya que al tratarse de elementos de un ámbito profesional que todavía no se ha extrapolado a un ámbito más informal, no hay tanto interés en el desarrollo de estas tecnologías por parte de artesanos o aficionados.

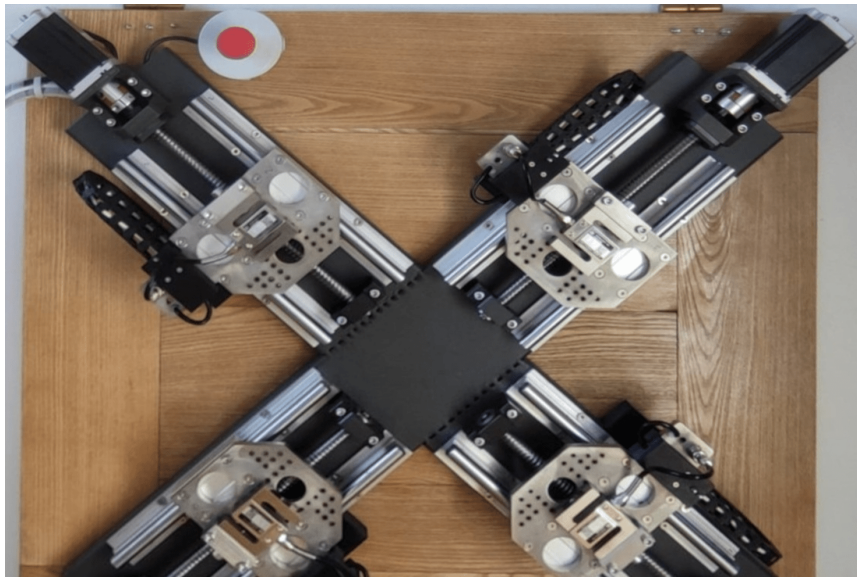
A continuación se pueden ver algunas opciones que se pueden encontrar por internet y son interesantes de comentar por su implementación:

- **Máquina de ensayos biaxial-Universidad de Aveiro.** Esta máquina está realizada por el departamento de ingeniería mecánica de la universidad de Aveiro. En el artículo publicado en la página MDPI se detalla el proceso de creación de la máquina junto con el sistema que implementa. En el artículo se detalla que es importante el sistema de control de la máquina y de obtención de los datos y se da información del uso de husillos para una efectividad mayor.



- **Máquina de ensayos biaxial-Miguel Sánchez.** El proyecto realizado por el tutor de este trabajo que sirve como precedente de éste. Es una máquina basada en un controlador para Arduino y una interfaz simple realizada en Python que permite hacer tests con los cuatro motores a la vez. La comunicación entre la interfaz y el microcontrolador se realiza por el puerto serie. Esto permite la comunicación sea sencilla.

⁵ Figura 5. Máquina de ensayos de la Universidad de Aveiro.



Como se puede observar, no hay mucha información respecto a proyectos similares de particulares o de investigación, ya que es un tópico casi inexplorado en la que la mayoría de estos proyectos son para un uso en particular y no son genéricos en cuanto a sus capacidades para hacer los experimentos.

Las principales carencias que se han mostrado en este apartado han sido la falta de control de las máquinas y el no tener un consenso para el desarrollo de éstas. La falta de homogeneidad en el desarrollo de estas soluciones hace que no se pueda hacer un ejercicio de aprendizaje general, lo que hace que sean muy poco intuitivas.

El desarrollo de un lenguaje común entre estos dispositivos y los ordenadores podría facilitar el uso de estos bancos de pruebas al igual que lo comentado en la motivación de este proyecto con máquinas de control numérico computerizado o el GCODE de las impresoras 3D.

En conclusión, esta tecnología puede expandirse a ámbitos como la conservación de arte compartiendo el desarrollo de los controladores principales en un concepto de código abierto para todo el mundo. En los siguientes apartados se desarrollará el proyecto como tal, mostrando las mejoras o puntos a tener en cuenta que se han podido encontrar respecto a todos los ejemplos mostrados anteriormente.

⁶ Figura 6. Máquina de ensayos realizada por Miguel Sánchez, tutor del TFG.



3. Análisis del problema

Una vez visto los ejemplos en el apartado anterior y visto lo que ofrecen las demás ofertas en el mercado, es el momento de identificar las características que podemos extraer de estos ejemplos y también identificar las mejoras que se van a proponer en nuestra implementación.

El punto de partida del proyecto se va a basar en el control total de los distintos actuadores de forma que el usuario tenga pleno control sobre su dispositivo. Muchas de las empresas que realizan el software no tienen en cuenta todo lo que pueden ofrecer estas máquinas y ofrecen un software con una curva de aprendizaje casi nula al venir los programas por defecto ya incluidos. Esto hace que estas tecnologías no sean capaces de operar entre distintos campos de pruebas.

La mejora que se propone en este caso es el desarrollo de un software capaz de ofrecer una interfaz que funcione tanto para usuarios no expertos como para los que sí que lo son. Por medio de un lenguaje basado en comandos se puede lograr una implementación genérica para un uso del controlador que proporciona más control sobre la máquina.

Esta interfaz, por lo tanto, ofrece una visión gráfica del estado del dispositivo a la vez que ofrece la oportunidad de ejecutar comandos o un programa entero leyendo de un archivo, como hacen las impresoras 3D hoy en día.

Otra de las deficiencias que he podido encontrar es que la mayoría de controladores son de código propietario, por lo que no se pueden hacer mejoras sobre el código ya existente. Esto es un gran fallo porque limita la capacidad de la máquina a un funcionamiento ya preestablecido.

La solución a este problema es la implementación de la interfaz ya mencionada por separado que ejecute los comandos en backend y sea el microcontrolador el que únicamente se encargue de interpretar los comandos sencillos y hacer los movimientos o mediciones.

Con lo mencionado anteriormente conseguimos de una forma sencilla que el cómputo general se realice en dispositivos con mayor capacidad de cómputo mientras que la parte sencilla se ejecuta en un microcontrolador que no tiene por qué ser muy potente computacionalmente. Gracias a esto también conseguimos que el código del controlador no sea muy extenso, lo que ayuda a que se puedan incluir nuevos comandos de forma muy sencilla.

Finalmente se van a mencionar las cosas que se pueden extraer de los distintos ejemplos y que se van a tener en cuenta para realizar nuestro proyecto. El uso de

una interfaz gráfica o la capacidad de movimiento de cada brazo por separado son características que ya he mencionado, así que me voy a centrar en apartados como el funcionamiento cíclico o la comunicación por puerto serial entre el microcontrolador y el ordenador donde se ejecuta la interfaz.

La capacidad de realizar el mismo test una y otra vez en algunos casos es importante, ya que hay materiales que puede que no haga falta romperlos si no comprobar como se comportan con movimientos de compresión y extensión. En este aspecto, la ventaja que proporciona el poder mover los brazos en ambas extensiones es de vital importancia.

En cuanto a la comunicación, el sistema por puerto serie es una solución sencilla de implementar y eficaz, ya que debido al ámbito de este tipo de máquinas, invertir en un sistema de comunicación como podría ser el WiFi sería una pérdida de dinero. Además, esta solución por cable evita la posible pérdida de información siempre que se respeten los tiempos de comunicación.



4. Solución propuesta

Una vez identificados los puntos a tener en cuenta en el anterior punto, vamos a proceder a especificar la solución al problema propuesta. En esta solución vamos a hacer un resumen de los componentes en los que se va a basar el proyecto teniendo en cuenta tanto la funcionalidad como el trabajo que va a involucrar.

4.1. Principales componentes del sistema

El diseño planteado para este proyecto está basado en tres componentes principales, que son los siguientes:

- **Aplicación de escritorio.** Este bloque comprende la creación de una aplicación de escritorio con interfaz gráfica para el manejo de la máquina y el procesamiento interno de los comandos y datos tanto para recibir como mandarlos por el puerto serial al que está conectado el Arduino. La interfaz cuenta con elementos interactivos a simple vista como botones y cuadros de texto y una parte en la que se puede manejar la máquina por medio de comandos escritos como si fuera una terminal. Además, incluye la parte más importante de esta solución, la capacidad de ejecutar un archivo con comandos para realizar la pruebas.
- **Firmware del Arduino.** En este bloque de trabajo se va a realizar el programa principal del Arduino para el procesamiento de los comandos, movimiento de actuadores, captación de datos de los sensores y envío de datos a por puerto serial. Este programa está basado en un bucle constante que una vez conectado a la aplicación de escritorio por el puerto serial lee e interpreta lo que llega. Es un programa lo más sencillo posible que posibilita el tener un microcontrolador con menos recursos que consuma menos o sea más barato.
- **Máquina de pruebas.** Junto con el programa de Arduino, se va a construir una máquina prototipo para probar que todo lo programado se puede llevar a cabo en un medio físico con las complicaciones que puede



tener este tipo de implementaciones. Con esta máquina lo que se pretende es probar si la comunicación Aplicación - Firmware se realiza

de tal modo que los actuadores y sensores se comporten como se espera de ellos. Cabe destacar que este trabajo al tratarse de una solución informática puede no tener en cuenta problemas mecánicos, ya que se escapa de las competencias de este trabajo.

4.2. Plan de trabajo

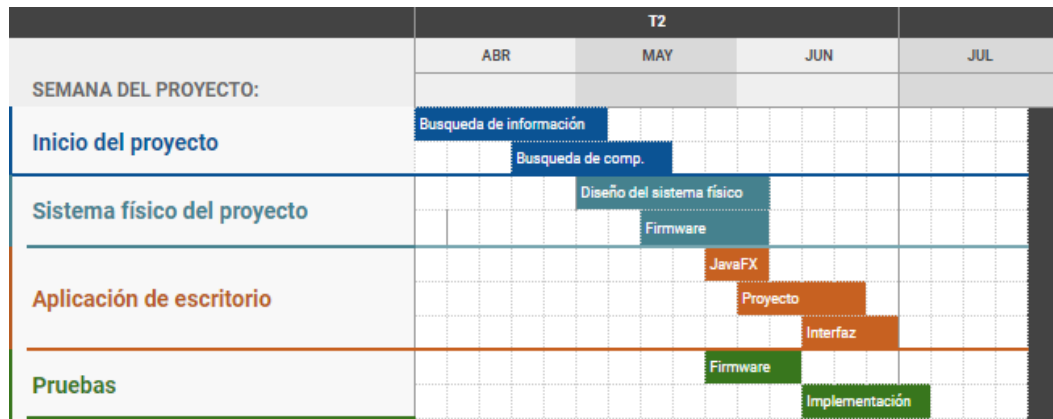
Para la realización del trabajo se ha contado con tres meses de desarrollo, por lo que se han tenido que priorizar tareas. Lo primero que se ha realizado es un estudio de este tipo de máquinas y cuales eran sus principales características. Esto se ve reflejado en los apartados anteriores en los que se ha logrado conseguir la información necesaria.

Una vez conseguida toda la información, se ha realizado una labor de investigación de cuales eran los componentes físicos necesarios para llevar a cabo la implementación y trabajar sobre esa arquitectura. En este caso se han buscado componentes compatibles con las librerías de Arduino para hacer de forma sencilla el firmware y componentes ya existentes en otras tecnologías como las impresoras 3D.

Con los componentes ya seleccionados se empieza a programar el firmware para el Arduino, ya que permite un ahorro temporal al conocer este tipo de tecnología y mayor facilidad para comprobar que el programa se ejecuta como es debido gracias a pruebas sencillas sobre los actuadores y sensores.

Con el programa de Arduino ya casi terminado, se realiza la aplicación de escritorio en Java por facilidad de trabajo debido al conocimiento de este lenguaje en concreto. Se realiza una labor de investigación también en el ámbito de la interfaz gráfica, ya que aunque se ha trabajado ya sobre ello, hay nuevas necesidades y requerimientos para el correcto funcionamiento del sistema. En este apartado se pueden hacer pequeñas modificaciones al firmware para lograr mayor compatibilidad con la interfaz.

Finalmente, todo el sistema es implantado y probado con la ejecución de ambos programas y se resuelven pequeños problemas que hayan surgido al final de estas pruebas.



7

⁷ Figura 7. Diagrama de Gantt de la organización de las tareas del proyecto.



5. Diseño de la solución

A continuación, se van a detallar los componentes elegidos para el proyecto y como estos han sido desarrollados con todas sus características más importantes. Se van a comentar las particularidades de cada componente y como interactúan estos entre sí.

5.1. Microcontrolador

El microcontrolador elegido para el desarrollo del proyecto es una plataforma Arduino, como ya se ha comentado anteriormente. Este tipo de controladores es muy usados en proyectos de electrónica por su polivalencia a la hora de de conectar dispositivos como sensores o actuadores de forma sencilla e intuitiva.

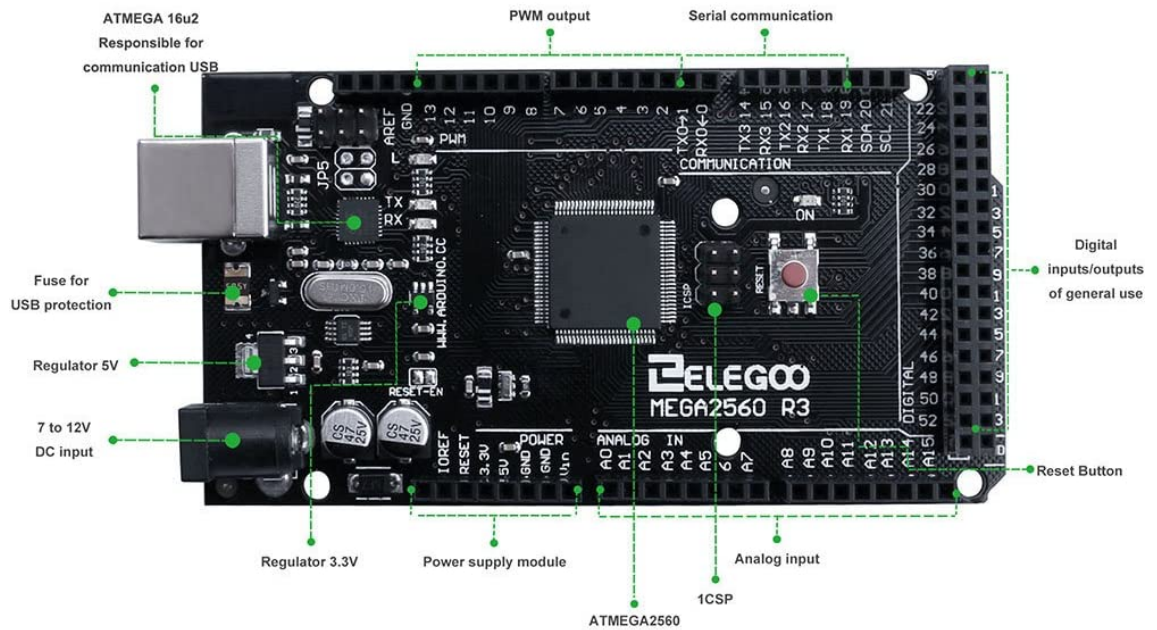
En concreto, el microcontrolador elegido ha sido el Arduino Mega 2560 que cuenta con 54 salidas/entradas digitales, 16 salidas/entradas analógicas y 4 puertos seriales. También cuenta entre las salidas digitales con quince salidas PWM (Modulación por ancho de pulsos). Esto es posible gracias a su procesador, un ATmega2560 de 8 bits capaz de trabajar con muy bajo consumo.

La arquitectura del procesador está basada en RISC, es decir, trabaja con instrucciones pequeñas y sencillas de ejecutar por parte del procesador, al igual que procesadores estudiados en la carrera como el MIPS R2000. El ATmega2560 tiene 256 KB de memoria flash ISP, 8 KB SRAM, 4 KB EEPROM, 86 entradas/salidas de propósito general y 32 registros de propósito general.

El procesador obtiene una potencia de salida de 16 MIPS (Millones de instrucciones por segundo) a una frecuencia de 16 MHz.

A continuación se puede observar una imagen de la plataforma elegida en la que se pueden observar las partes principales como las entradas analógicas señaladas con una "A" al principio y las demás líneas digitales.





8

5.2. Programación del microcontrolador

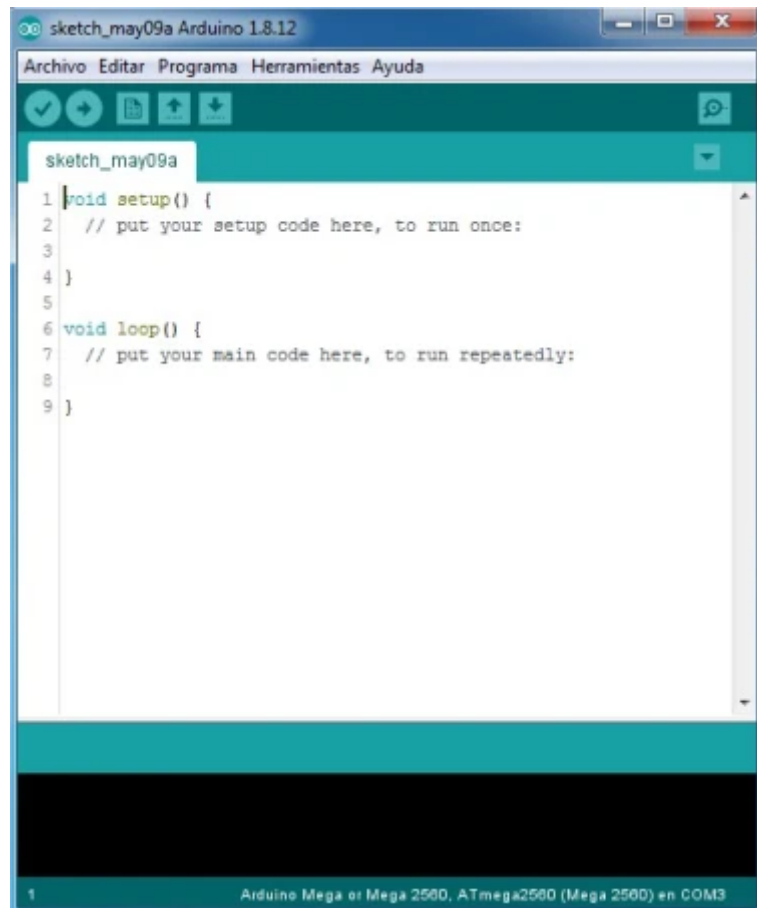
El usar una plataforma como Arduino proporciona ventajas a la hora de hacer los proyectos más sencillos, ya que es una plataforma open source en la que la gente aporta gran cantidad de información para mejorar o solucionar problemas que se puedan tener.

En este aspecto, la propia empresa fabricante de la placa, proporciona su software IDE para la programación de los microcontroladores por medio de un lenguaje basado en C++ que facilita la interpretación de las entradas/salidas en la placa. Se puede decir que abstrae de cierta manera la programación a bajo nivel para democratizar el uso de la plataforma por gente con menos familiaridad con este tipo de programación.

Cabe destacar que, aunque el “sketch” o programa de Arduino tenga una estructura básica, puedes programarlo de una forma similar a C++ ahorrando peso del programa y siendo más eficiente.

La estructura básica del programa se divide en dos partes, el “setup” en el que se ponen los valores que se quiere por defecto en el programa y el “loop”, donde se ejecuta el bucle principal del programa que es el cuerpo principal del programa. La inicialización de variables estáticas, globales o el esquema de entradas/salidas se ponen al principio del programa, como se haría en C++.

⁸ Figura 8. Arduino Mega 2560.



9

Como se puede observar, el entorno de desarrollo es muy sencillo de utilizar y consta de las opciones básicas para el desarrollo y posterior compilación y programación en la placa. También contiene programas de prueba para familiarizarse con las opciones que ofrecen los distintos sensores y actuadores que se pueden conectar a la placa.

También tenemos la opción de desarrollo en entornos de programación como Visual Studio Code instalando la extensión para Arduino.

En cuanto al proyecto, se ha determinado que seguir el esquema principal del “sketch” era una buena opción de cara a no complicar el desarrollo del firmware como se verá en el apartado dedicado a éste. El hecho de que sea menos eficiente no implica ningún problema, ya que en este caso la potencia de cómputo o la memoria necesaria para el programa no suponen ninguna limitación.

⁹ Figura 9. Ejemplo del IDE Arduino con un “sketch” vacío.

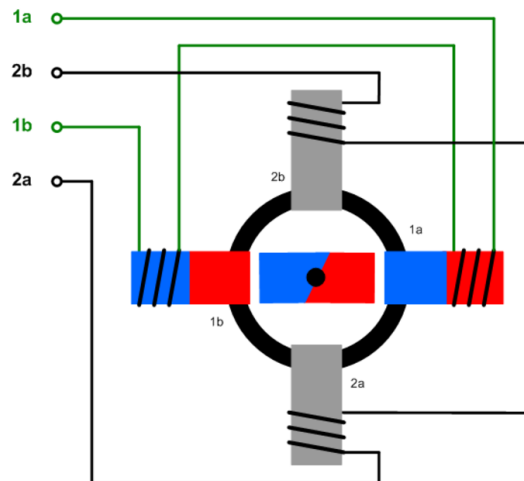
5.3. Actuadores del sistema

Para obtener los movimientos en la propia máquina se necesitan actuadores que tienen que ir conectados a unos controladores que hacen de interfaz entre el Arduino y los propios actuadores. Es por esto que hay que tener muy claro que tipo de actuadores se van a emplear.

En nuestro caso, se ha decidido utilizar los motores paso a paso Nema 17 que se pueden encontrar en las impresoras 3D. Estos motores no ofrecen una gran fuerza por sí mismos y pueden perder pasos cuando la fuerza que tienen que ejercer es muy grande, pero al ser una máquina prototipo con el único cometido de comprobar que el desarrollo software funciona, son más que válidos.

5.3.1. Motores Nema 17

Los motores Nema 17 son unos motores paso a paso bipolares que se basan en las diferentes configuraciones de corriente por sus bobinas principales para moverse en unos determinados ángulos. En la siguiente imagen se puede ver un esquema sencillo de un motor bipolar.



10

En el caso concreto de estos motores, cada paso en “full step” equivale a un ángulo de $1,8^\circ$ por lo que necesita 200 pasos en este modo para completar una vuelta. Cabe destacar que al ser de circuito abierto, pueden dar las vueltas que sean sin importar la posición.

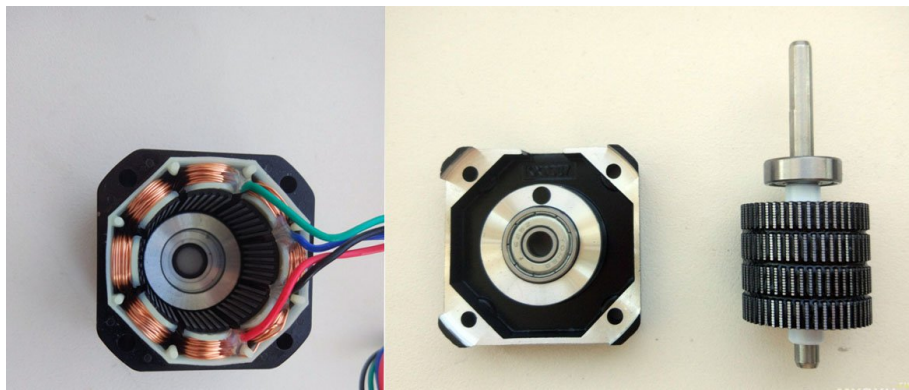
¹⁰ Figura 10. Esquema motor bipolar.

Otras configuraciones de “stepping” también son posibles, aunque no son pasos reales sino micropasos conseguidos por el controlador por medio de la modulación de las ondas que actúan sobre las bobinas principales del motor. Como se explicará en el siguiente apartado del controlador, éste nos permite hasta un “microstepping” de hasta 1/32, lo que equivale a que para lo que antes era un paso, ahora son 32 y para dar una vuelta completa se necesitan 6400 micro pasos.

Continuando con las especificaciones de los motores tenemos que ofrecen 40 Newton por Centímetro de fuerza de torsión para una corriente máxima de 0,9 amperios y un voltaje de casi 4 voltios.

Otros motores de la misma categoría incorporan una reductora que permite ofrecer una fuerza mayor a cambio de perder velocidad de actuación. Estas reductoras se pueden incorporar a los motores que se utilizan en este proyecto, por lo que es válido también.

Otros motores mayores a los Nema 17 ofrecen mayores prestaciones a cambio de un amperaje mayor y por lo general suelen ser utilizados en entornos industriales, por lo que no se han tenido en cuenta en el proyecto.



11

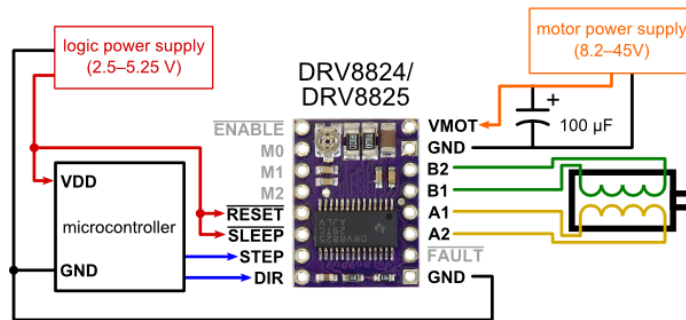
En la imagen se puede observar el motor desmontado con las bobinas por un lado y el eje del motor por otro. Tanto en el eje como en el estator se observan unas “aletas” que sirven para posicionar el motor en las diferentes orientaciones necesarias para los 200 pasos que ofrece el motor de forma normal.

¹¹ Figura 11. Disección de un motor Nema 17.

5.3.2. Controladores de los actuadores

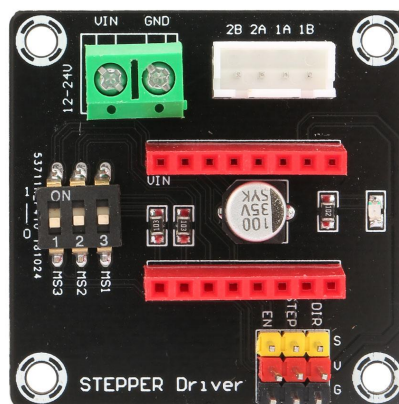
Los controladores elegidos para los motores han sido los DRV8825 que aguantan hasta 1,5 amperios y un microstepping de hasta 1/32 como se ha mencionado antes. Estos controladores son unos chips muy fáciles de manejar y que hacen todo el trabajo de stepping mandando únicamente tres señales desde el Arduino.

Estas señales son la de activación del motor, que hace que el motor se mantenga en la posición indicada; la dirección del motor y la información para los pasos que tiene que dar el propio motor. En la imagen siguiente se puede observar como quedaría interconectado el microcontrolador con las señales STEP y DIR para los pasos y la dirección. En nuestro caso, ENABLE también está controlado por el microcontrolador.



12

Para la interfaz entre el Arduino y el controlador se ha optado por una placa compatible con arduino y microstepping para que proporcione la parte del condensador que maneja el voltaje de los motores desde la fuente de alimentación externa.



13

¹² Figura 12. Controlador de los motores DRV8825.

¹³ Figura 13. Interfaz entre Arduino y el controlador de los motores.

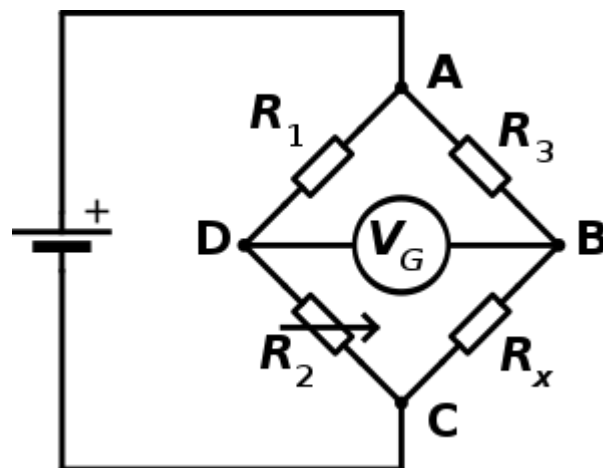
5.4.Sensores del sistema

En una máquina de estas características son tan importantes los sensores como los actuadores, ya que son los que proporcionan los resultados que queremos obtener al final. En este caso, no se necesitan sensores para el propio funcionamiento de la máquina de forma que pueda obtener una acción de movimiento si no que los sensores actúan junto a los actuadores a la hora de obtener la gráfica que se necesita finalmente.

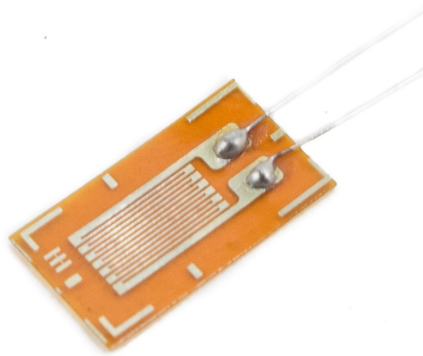
En nuestra máquina se ha optado por incluir células de carga similares a las que se pueden encontrar en las básculas de pesar comunes que podemos encontrar en nuestras casas. Estas proporcionan los datos de fuerza ejercida sobre el objeto y le mandan un valor eléctrico al Arduino. A continuación se va a detallar el tipo de sensor escogido y el funcionamiento interno de éste.

5.4.1.Células de carga

El funcionamiento de una célula de carga de puente completo es sencillo al tratarse de una lámina resistiva que varía su resistencia dependiendo de la deformación de esta. Esto hace que los valores de señal eléctrica medidos en sus diferentes terminales varíen. El término de puente completo se refiere a que este sensor de deformidad está formado por un circuito o puente de Wheatstone como se puede ver en la siguiente imagen junto con el sensor de deformidad o “Strain Gauge”.



¹⁴ Figura 14. Puente de Wheatstone.



15

Como se ha comentado antes, por sí solo, el sensor de deformidad puede medir la deformidad de un objeto, pero para nuestra aplicación necesitamos saber la fuerza ejercida sobre el objeto, de tal forma que necesitamos una estructura de metal a la que va acoplada y de la que se sabe la deformación a cuantos Newtons equivale.

Es por esto que las células de carga usualmente se presentan como un prisma rectangular con un orificio por el que se reduce la sección del prisma para poder medir en ese punto la deformidad y en conclusión los Newtons haciendo la transducción por medio de las señales eléctricas.

En el caso de las células de carga para estas aplicaciones pueden tener una forma de zig zag de tal modo que la medición se haga en el mismo eje del sensor, pero debido a que estos suelen tener un mayor valor económico, se ha decidido utilizar los primeros en una distribución en la máquina similar, por tanto tenemos los 4 sensores colocados de igual forma en cada brazo de la máquina.



16

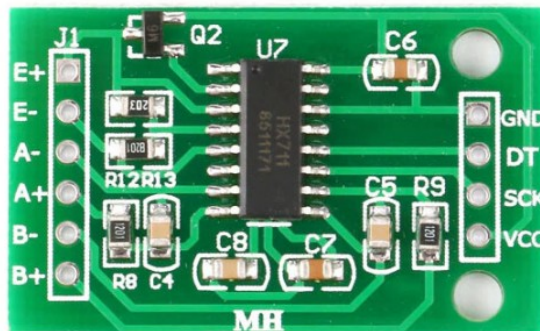
¹⁵ Figura 15. Sensor de deformación.

¹⁶ Figura 16. Célula de carga elegida para el proyecto.

5.4.2. Módulo analógico-digital

Como se ha comentado anteriormente, los sensores de deformación envían señales analógicas por sus terminales, ya que no son más que unos dispositivos que miden la diferencia de voltaje generado por el cambio en la resistencia. Estas señales no tienen valores muy altos, por lo que el Arduino podría no obtener los valores de forma correcta, es por esto que necesitamos otro elemento a modo de “interfaz” para conectar los sensores con el microcontrolador.

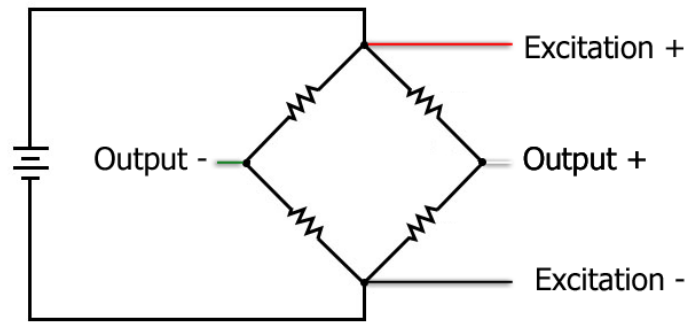
Es por esto que se introduce un módulo que transforma esas señales analógicas con un valor pequeño a señales digitales capaces de ser interpretadas por el microcontrolador. En este caso los módulos AD elegidos han sido los HX711, ya que son pequeños y eficientes a la vez de que proporcionan facilidad en el uso de estos al conectarlos al Arduino.



17

En la imagen superior se puede ver el módulo utilizado. En la parte derecha se sitúan las entradas de señal desde el sensor de tal forma que E indica la señal de excitación del sistema y A las lecturas de señal. Las entradas B no son utilizadas en nuestro caso

¹⁷ Figura 17. Módulo AD HX711.



18

En el lado derecho del módulo encontramos las conexiones con el Arduino, la entrada de voltaje (VCC y GND para el común) y las entradas SCK para el reloj y DT para la salida de datos hacia el microcontrolador.

Las conexiones se verán en detalle en el capítulo siguiente dedicado a la implementación del sistema.

¹⁸ Figura 18. Conexión de la galga extensiométrica con el HX711.



6. Implementación de la solución

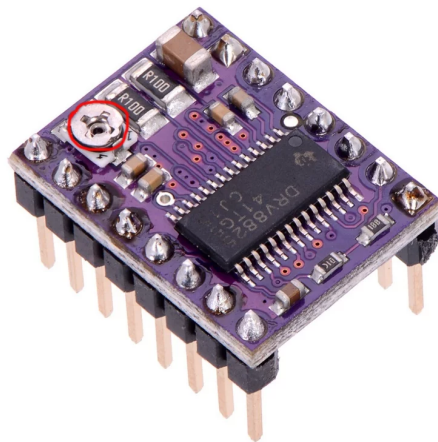
Una vez definidos todos los componentes que se han elegido para el proyecto pasamos a la fase de implementación de la solución, es decir, hacer que todos los elementos funcionen como esperamos de ellos. Este capítulo va a envolver todo lo relacionado con las conexiones de los componentes y su posterior programación teniendo en cuenta las particularidades del sistema.

Vamos a dividir este capítulo en dos partes, una en la que se detallará el sistema físico como base y la segunda parte en la que se entrará al detalle de la programación del sistema.

6.1. Cableado de los actuadores

Los actuadores en nuestro sistema son 4 motores paso a paso Nema 17 que están conectados cada uno a su respectivo “driver” como se ha explicado en el capítulo anterior. Cada driver tiene 3 tres pines que van conectados al Arduino y la entrada de voltaje principal por parte de la fuente de alimentación de 12 voltios. Por tanto, se necesitan un total de 12 entradas/salidas digitales solo para los motores del sistema.

Lo primero que necesitamos hacer es modificar el amperaje máximo con el que puede funcionar el motor desde el controlador DRV8825 gracias a un potenciómetro que lleva incorporado en el cuerpo del PCB.



19

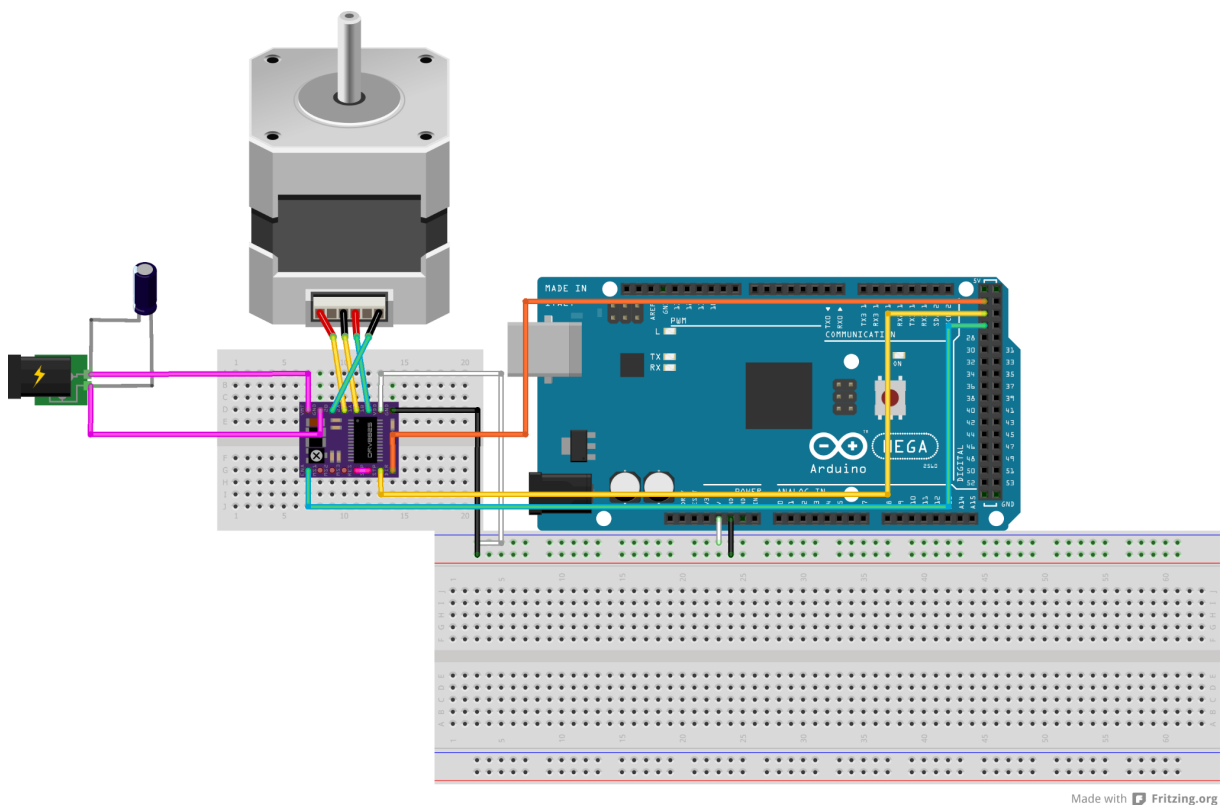
¹⁹ Figura 19. Detalle del potenciómetro incorporado en el controlador.

En nuestro caso, el amperaje máximo es de 0,9A, por lo que debemos girar el potenciómetro hasta que obtengamos un voltaje con la fuente de alimentación y el Arduino conectado que siga la siguiente fórmula.

$$\text{Current Limit} = VREF \times 2$$

De esta forma obtenemos que para nuestros motores es necesario un voltaje de 0,9/2 o lo que es lo mismo, 0,45V.

Una vez configurados todos los controladores para los cuatro motores, procedemos al cableado con los pines asignados para cada elemento de cada motor. Cada motor necesita tres señales, la dirección, la habilitación y los pasos que quieres dar. La solución propuesta es conectar las doce entradas/salidas necesarias en serie alternando las 3 señales para cada motor de tal forma que las tres primeras entradas/salidas sean DIR-STEP-EN sucesivamente. Los pines del Arduino elegidos van desde el 22 hasta el 44 saltando los pines impares por comodidad de conexión.



20

En la imagen se puede observar el diagrama de conexión de uno de los motores bipolares con el controlador DRV8825 al Arduino. La placa de expansión en la que va introducido el controlador está detallada por su diagrama de conexión

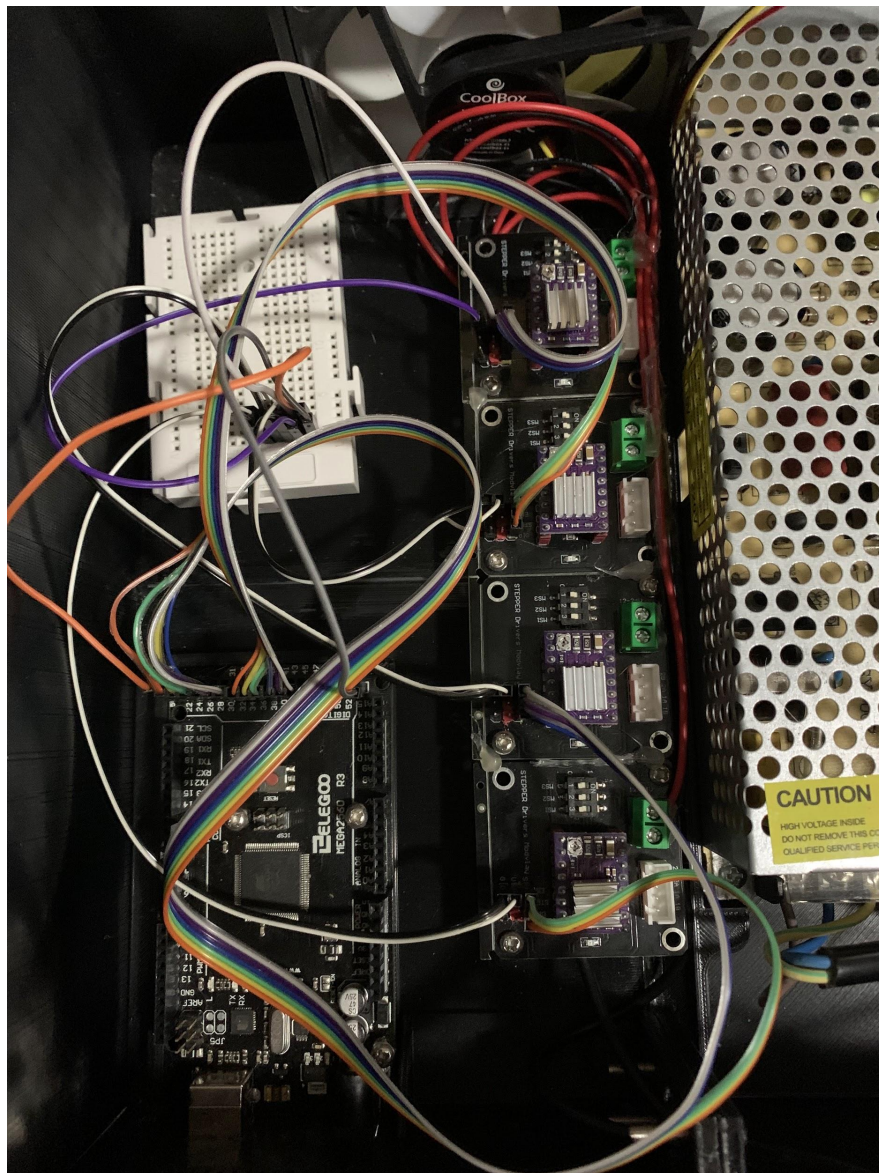
²⁰ Figura 20. Conexión de motor con su driver al Arduino.



interna, en la que cada pin va conectado a su respectiva entrada/salida como se puede ver en la figura 12 que ha sido mostrada anteriormente.

Los pines del controlador MSG1-MSG2-MSG3 no han sido conectados, pues en la placa de expansión se ha dejado por defecto la configuración de pasos a 1/1, es decir, 200 pasos para una vuelta completa.

Se ha decidido presentar la conexión de un motor para facilitar la visión del conexionado, ya que los otros motores tienen la misma configuración solo que en pines distintos del microcontrolador. A continuación se puede observar en detalle la conexión real realizada con los cuatro motores en los que se diferencian los motores del eje X en los seis primeros pines y los motores del eje Y en los siguientes seis.

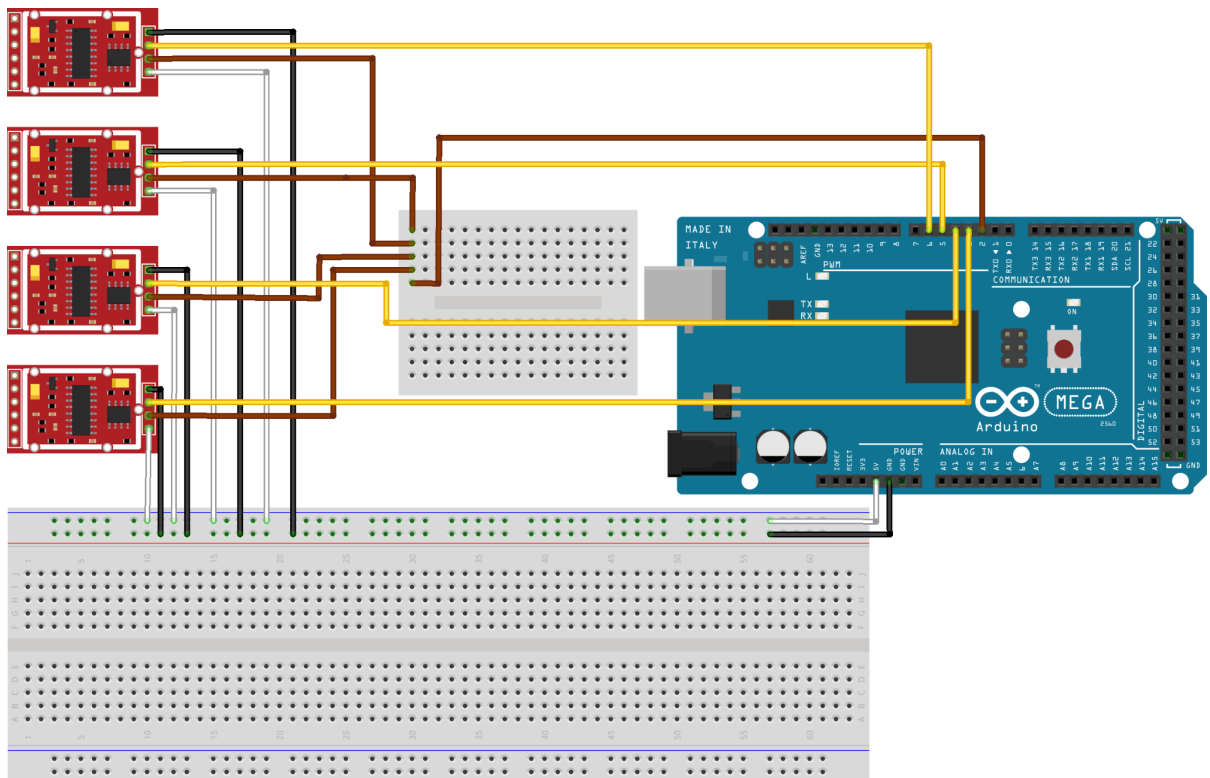


21

²¹ Figura 21. Conexión de los motores en el hardware.

6.2. Cableado de los sensores

Para el cableado de los sensores se ha procedido a hacer lo mismo que con los actuadores, primero un diagrama de conexión y después se ha llevado al hardware real. Cada módulo HX711 ha sido probado antes de hacer las conexiones para asegurarse de que iban bien y además coger los valores de calibración de las células de carga.



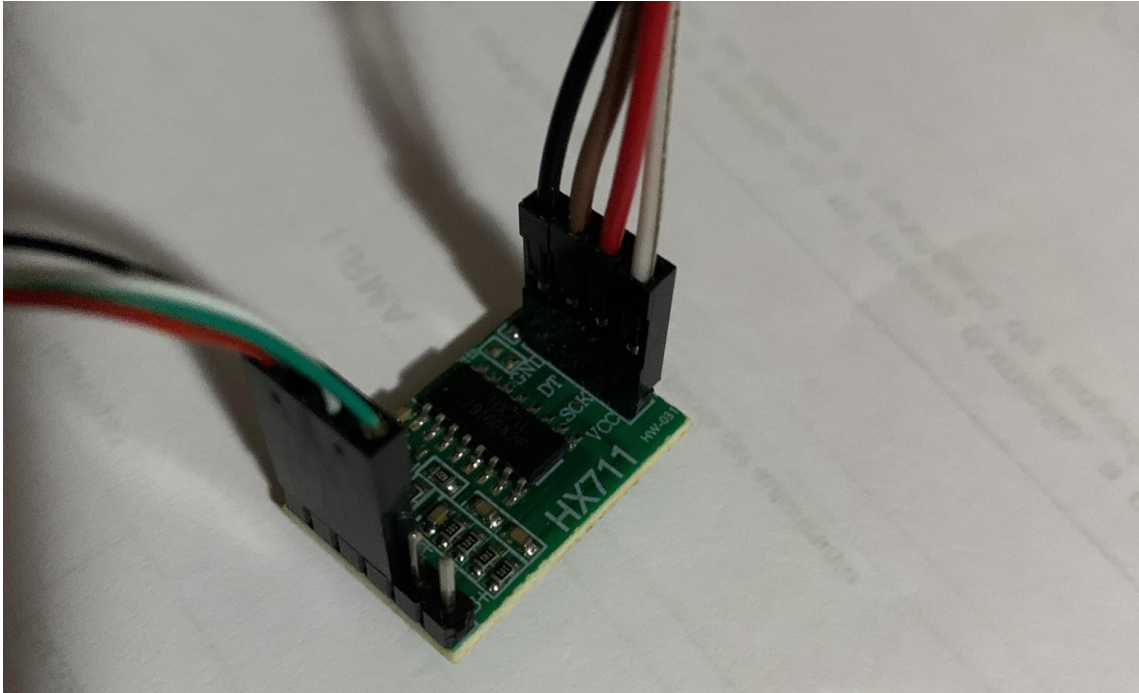
Made with Fritzing.org

22

En el diagrama se pueden observar los cuatro módulos HX711 conectados al Arduino con un código de colores para cada pin del módulo. Los colores blanco y negro indican la entrada de voltaje, los cables amarillos las salidas de información que van hasta los pines 3-4-5-6 y en marrón tenemos la entrada del reloj para los módulos que comparten uno común en el pin 2 del microcontrolador.

La conexión de los módulos hacia el propio sensor no se muestra en el diagrama, pero se puede observar en la siguiente imagen en detalle en la implementación real.

²² Figura 22. Diagrama de conexión de los sensores.

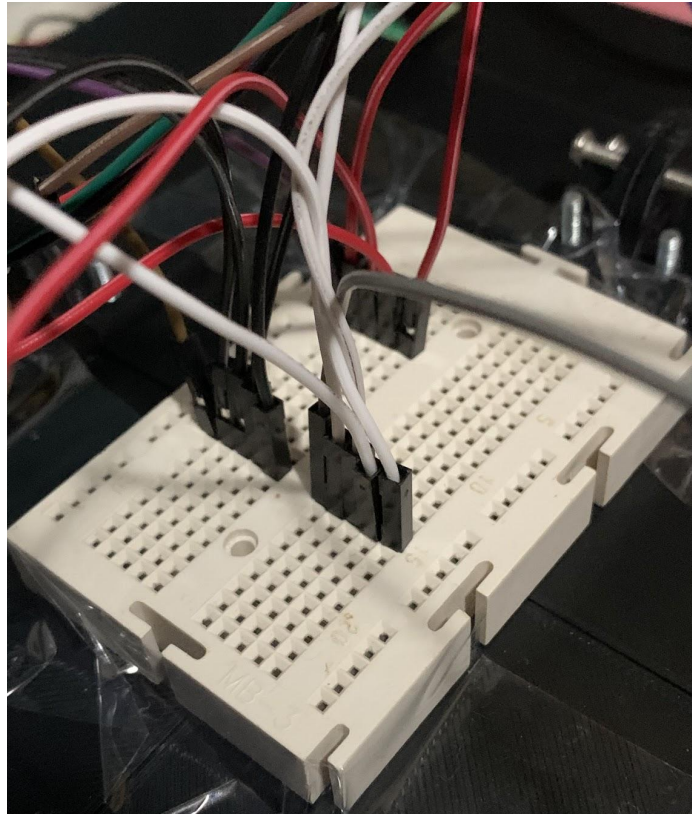


23

Como se puede ver, la parte analógica del módulo sigue un patrón de colores determinado que ha sido comprobado midiendo con el multímetro para saber cuáles eran los terminales de excitación y cuáles los de información. En este caso, los cables rojo y negro corresponden con los de excitación y los verdes y blancos corresponden con los terminales de información especificados con la letra A.

A continuación se va a mostrar el cableado final una vez implementado ya en la máquina. Como se puede observar hay una protoboard pequeña en la que se han realizado las conexiones de VCC, GND y el conjunto de pines de reloj para después conectarlo con el arduino de forma más limpia evitando posibles problemas de cara a cerrar la caja en la que va contenida toda la electrónica y con el fin de evitar sobrecalentamientos que puedan hacer que los drivers no entreguen toda la potencia necesaria a los motores.

²³ Figura 23. Módulo HX711 con conexiones.



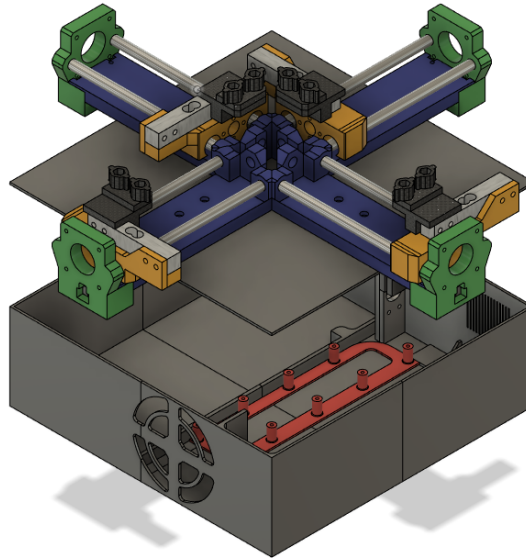
24

6.3. Prototipo de máquina de pruebas

Finalmente, con toda la electrónica cerrada dentro de la propia máquina, hemos conseguido crear un prototipo hecho a base de impresión 3D y piezas de una antigua impresora 3D con el fin de ahorrar en materiales. Esta máquina no tiene que soportar grandes esfuerzos y se ha considerado que realizar la estructura por medio de impresión 3D era una buena técnica para hacer una máquina demostrativa del funcionamiento.

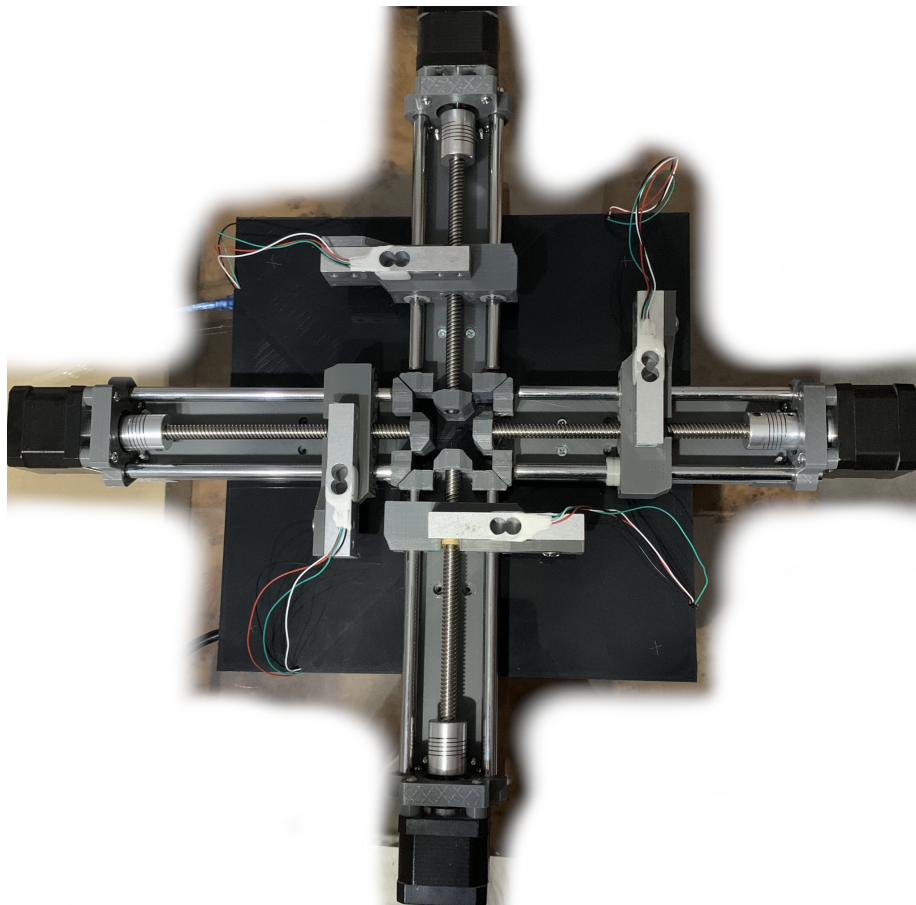
Esta máquina proporciona los elementos necesarios como para comprobar que el software, que es la parte importante de este trabajo, funciona correctamente y comprobar parámetros como la distancia que recorre el carro por cada paso que da el motor o si la dirección de cada motor es la correcta. También nos permite ver si los sensores están funcionando correctamente y en caso contrario poder calibrarlos de forma sencilla.

²⁴ Figura 24. Cableado de los sensores en la protoboard.



25

A continuación se muestra una imagen de la máquina ya terminada y lista para recibir tanto el código firmware del arduino como las órdenes por parte de la interfaz y comprobar si la comunicación entre los dos elementos es funcional.



26

²⁵ Figura 25. Diseño 3D de las piezas de la máquina para imprimir en 3D.

²⁶ Figura 26. Máquina prototipo finalizada.

6.4. Firmware del Arduino

Para crear el firmware del Arduino, lo que se ha buscado es la simplicidad y en la medida de lo posible el tener un código simple y pequeño para en caso de querer añadir más elementos a la máquina que fuera de forma sencilla. El cómputo general de la máquina es hecho por la propia interfaz, de forma que el microcontrolador tuviera la menor carga, ya que elementos como la librería de los módulos HX711 es bastante pesada para este tipo de procesador y tarda en ejecutar hasta los comandos más simples.

En cuanto a la estructura, como se mencionó antes en el documento, se ha seguido la estructura básica de la programación en Arduino con el setup y el loop principal del programa. Sobre esa estructura se han ido codificando funciones como la de mover los motores, cambiar de dirección los motores o deshabilitarlos.

A continuación se van a detallar las partes fundamentales del programa para dar una visión global de como funciona el código y su estructura.

6.4.1. Inicialización de las variables y librerías

Para poder utilizar los módulos HX711 hay que importar una librería que no viene por defecto en el IDE de Arduino, por lo que se ha tenido que buscar en GitHub un repositorio donde hayan creado esa librería. En concreto, la librería usada no es la que se utilizaría para un solo módulo, ya que al contar con cuatro de ellos se haría demasiado complicado el iniciar tantos elementos de ese tipo. Por lo tanto, la librería usada ha sido “HX711-multi.h” que proporciona la capacidad de tener varios módulos e inicializar una sola instancia de esta librería.

Para poder utilizar esta librería se han definido los pines tanto del reloj común en el pin 2 como los pines de información que van del tres al seis. También se han creado tres arrays que contienen el mapeo de los pines de habilitación, dirección y pasos de los motores para simplificar la codificación y que se vea de forma clara el código.

```
#include "HX711-multi.h"

// Pins to the Load cell amp
#define CLK 2 // clock pin to the load cell amp
#define DOUT1 3 // data pin to the first lca
#define DOUT2 4 // data pin to the second lca
#define DOUT3 5 // data pin to the third lca
#define DOUT4 6 // data pin to the forth lca
```



```
const int dirPin[] = {22, 28, 34, 40};
const int stepPin[] = {24, 30, 36, 42};
const int en[] = {26, 32, 38, 44};
```

6.4.2. Setup del programa

El setup del programa contiene la inicialización principal del programa, ya que determina los baudios a los que va a funcionar el puerto serial, inicializa el sistema con una primera medida de los sensores y pone los pines de los motores dependiendo de su nivel, activo o inactivo y la función, dependiendo de si van a ser salidas o entradas.

```
void setup() {
  Serial.begin(115200);
  Serial.setTimeout(1000);
  tare();
  scales.read(results);
  for (int i=0; i<scales.get_count(); ++i) {
    cal_values[i] = results[i];
  }
  for (int i = 0; i < 4; i++) {
    pinMode(dirPin[i], OUTPUT);
    pinMode(stepPin[i], OUTPUT);
    pinMode(en[i], OUTPUT);
  }

  for (int i = 0; i < 4; i++) {
    digitalWrite(en[i], LOW);
    digitalWrite(dirPin[i], HIGH);
  }
}
```

En este caso se ha determinado que un valor de 115200 baudios era perfectamente funcional para los resultados que esperamos obtener. El término baudio no es más que una manera de llamar a cuántas unidades de señal por segundo pueden transmitirse en una comunicación serial. Para obtener la tasa de bits por segundo se puede obtener por la siguiente fórmula:

$$Rb = \frac{2^n}{T} [bps]$$

También se puede ver el error que puede llegar a cometer la comunicación a los baudios que hemos considerado. Para el ATmega 2560 que es el procesador que lleva nuestro Arduino a una frecuencia de 16MHz se comete el siguiente error:

Baud Rate [bps]	$f_{osc} = 16.0000\text{MHz}$			
	U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4K	68	0.6%	138	-0.1%
19.2K	51	0.2%	103	0.2%
28.8K	34	-0.8%	68	0.6%
38.4K	25	0.2%	51	0.2%
57.6K	16	2.1%	34	-0.8%
76.8K	12	0.2%	25	0.2%
115.2K	8	-3.5%	16	2.1%
230.4K	3	8.5%	8	-3.5%
250K	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max. ⁽¹⁾	1Mbps		2Mbps	

27

Se puede observar entonces que el error para la velocidad doble desactivada es del -3,5% y para la velocidad de transferencia doble activada es del 2,1%. Se puede observar también como para valores de baudios menores como 9600 que sería un valor muy utilizado para Arduino el error disminuye, pero se ha necesitado una tasa de transferencia mayor para poder transmitir todos los datos de forma más rápida.

6.4.3. Loop principal del programa

El loop principal del programa es el encargado de gestionar los comandos que llegan desde el puerto serial e interpretar las diferentes partes del comando. Todos los comandos empiezan por una letra para diferenciar la función y los números siguientes para diferenciar los motores a los que van dirigidos o la cantidad de movimiento que se necesita.

El bucle es realmente un switch que comprueba la letra que llega del comando para ejecutar solo la función necesaria. A continuación se muestra el caso del comando de dirección, en el que comprueba la letra “d” y llama a la función dirMotor() con el parámetro que decide a cuál o cuáles de los motores va dirigida la acción.

```

case 'd': //DireccionMotores
    parameter = command.charAt(1);
    data = atoi(&parameter);

```

²⁷ Figura 27. Tasa de error para el procesador ATmega2560 para cada valor de baudios .



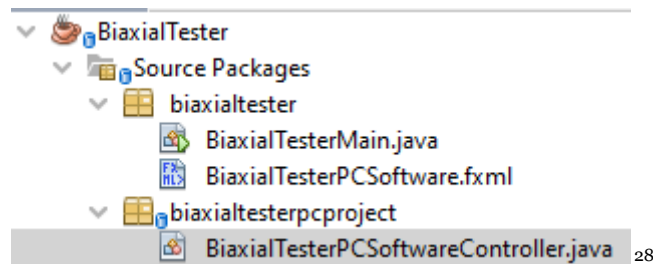
```
dirMotor(data);
break;
```

En el caso de las funciones a las que se llaman a lo largo del bucle, lo que hacen es escribir en el pin dedicado a la función dependiendo de si es de movimiento, de cambio de dirección o de habilitar motores. También están los comandos para conocer el estado de las direcciones de los motores y la función básica para medir con los sensores, que al ser más complicada de ejecutar por el procesador se hace un uso mínimo siempre que se puede, ya que bloquea por unos 2 segundos el procesador.

6.5. Software del programa

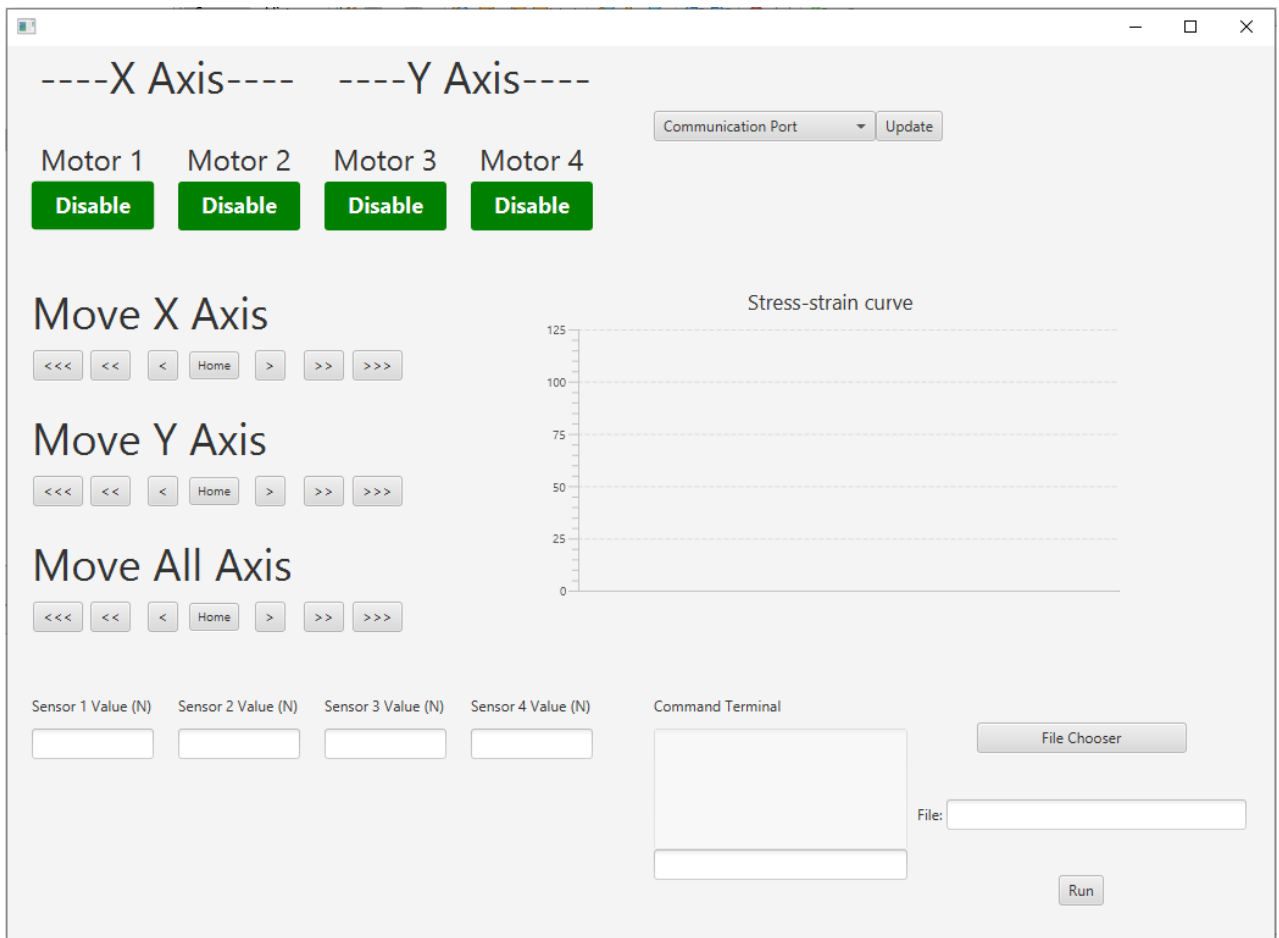
El software del programa está compuesto por una interfaz gráfica hecha con JavaFX por simplicidad y un programa que la gestiona hecho en Java 1.13. Por simplicidad se ha realizado la interfaz en SceneBuilder, que es una herramienta sencilla de utilizar en la que se pueden posicionar los elementos de la interfaz por medio de una GUI en vez de codificar por código.

El desarrollo del proyecto se ha hecho en el IDE NetBeans ya que proporciona todas las herramientas para el desarrollo en JavaFX de forma cómoda con un controlador un programa principal. La fácil integración de la interfaz en código ha hecho que podamos ahorrar tiempo en el desarrollo. La estructura del proyecto queda de la siguiente forma:



El controlador de la interfaz es, por tanto, el programa principal que contiene todas las funciones que son llamadas por los distintos eventos generados por los elementos de la interfaz.

²⁸ Figura 28. Estructura del proyecto de JavaFX.

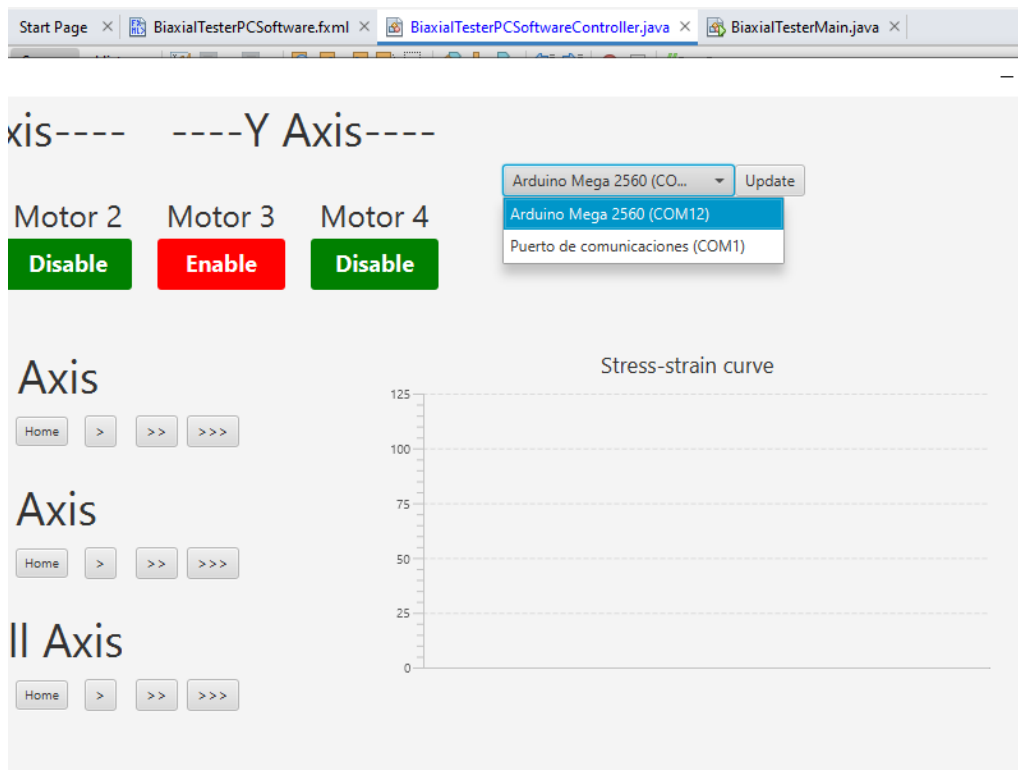


29

Como se puede ver en la imagen superior, la interfaz consta de distintos botones y elementos de texto para tener toda la información posible de la máquina y poder manejarla de forma sencilla.

En la parte superior tenemos los botones para habilitar o deshabilitar los distintos motores, divididos en dos grupos, el eje X y el eje Y. Estos botones indican por medio de los colores verde o rojo si están habilitados o no en ese orden. Al lado podemos encontrar un menú desplegable donde encontramos todos los puertos de comunicación que se encuentran presentes para poder conectarse y el botón Update para confirmar la selección de dicho puerto. En el caso de nuestro Arduino, el sistema lo detecta por su nombre, de forma que es muy intuitivo el seleccionar el puerto correcto.

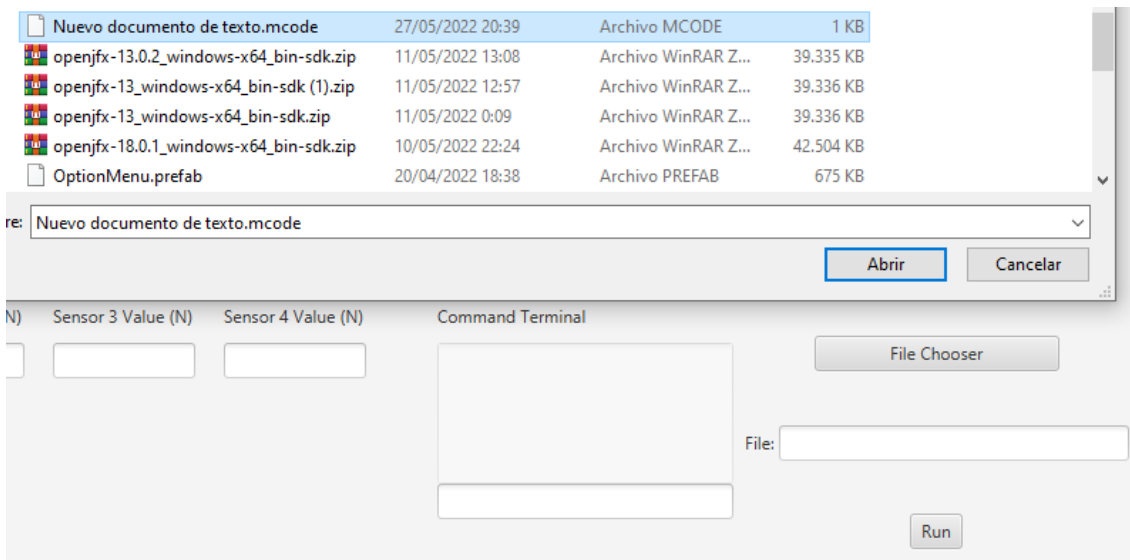
²⁹ Figura 29. Interfaz gráfica de la aplicación de escritorio.



30

Continuando por la parte izquierda de la interfaz, tenemos los botones de movimiento de los carros hacia ambas direcciones y justo debajo el valor obtenido por los sensores en esa iteración o llamada con el comando de estado. En el lado derecho encontramos la gráfica que genera la ejecución del archivo de texto que se puede escribir y debajo la terminal para comandos sueltos y la selección del archivo para ejecución.

³⁰ Figura 30. Detalle de la funcionalidad de la interfaz de usuario.



31

Como se puede observar en la imagen superior, al darle al botón “File Chooser” se ejecuta una ventana de selección de archivos en el que se puede elegir el archivo en formato mcode y una vez seleccionado, dando al botón “Run”, el propio programa ejecuta en otro hilo todas las líneas del archivo compuesto por los comandos diseñados para el Arduino al estilo de GCODE.

En cuanto a la temporización para el envío de datos y posterior recepción, el programa ejecuta pausas de 2s para que le dé tiempo a ejecutar al Arduino todos los datos enviados, lo que evita que puedas mandar dos comandos muy seguidos y que la comunicación serial falle.

En cuanto al uso de un hilo de ejecución adicional para la lectura del documento, se ha realizado para evitar problemas de bloqueos en cuanto a mandar y recibir información se refiere, ya que así el hilo principal del programa se encarga únicamente de recibir la información y el hilo secundario de mandar los comandos al microcontrolador.

```
//Hilo que gestiona la lectura de los datos
class MyThread implements Runnable
{
    public File doc = null;
    Scanner scan = null;
    byte[] b;
    public synchronized void run()
    {
        try {
            scan = new Scanner(doc);
        } catch (FileNotFoundException ex) {
```

³¹ Figura 31. Detalle del sistema de carga de archivos en formato mcode.

```

Logger.getLogger(BiaxialTesterPCSoftwareController.class.getName
()).log(Level.SEVERE, null, ex);
    }

    while (scan.hasNextLine()){
        if(commPort != null)
            commPort.writeBytes(b,
scan.nextLine().getBytes().length);
        try {
            Thread.sleep(2000);
            commPort.writeBytes(b,
scan.nextLine().getBytes().length);
            Thread.sleep(2000);
        } catch (InterruptedException ex) {

Logger.getLogger(BiaxialTesterPCSoftwareController.class.getName
()).log(Level.SEVERE, null, ex);
    }
}
programProcess = false;
}
}
}

```

6.5.1. Comunicación serial en Java

Para poder tener una comunicación serial en java se ha tenido que importar la biblioteca de jSerialComm, ya que esta no va incluida por defecto en las librerías de Java. Cabe destacar que existen otras librerías con la misma funcionalidad, pero se ha elegido esta por su simplicidad a la hora de utilizar lo que ofrece. En este aspecto, se asemeja a la facilidad que se encuentra en la comunicación serial de Python y ha permitido centrarnos en otros aspectos de la programación del programa.

6.5.2. Funcionamiento de los archivos mcode

A la hora de ejecutar un archivo mcode en la aplicación de programa hay que tener varias cosas en cuenta, como la estructura que mantiene. Al igual que los archivos gcode en los que primero se prepara la máquina por comandos y después viene los comandos de movimiento, en nuestro caso el funcionamiento

es similar. Los comandos que existen en el proyecto en este momento son los cinco siguientes:

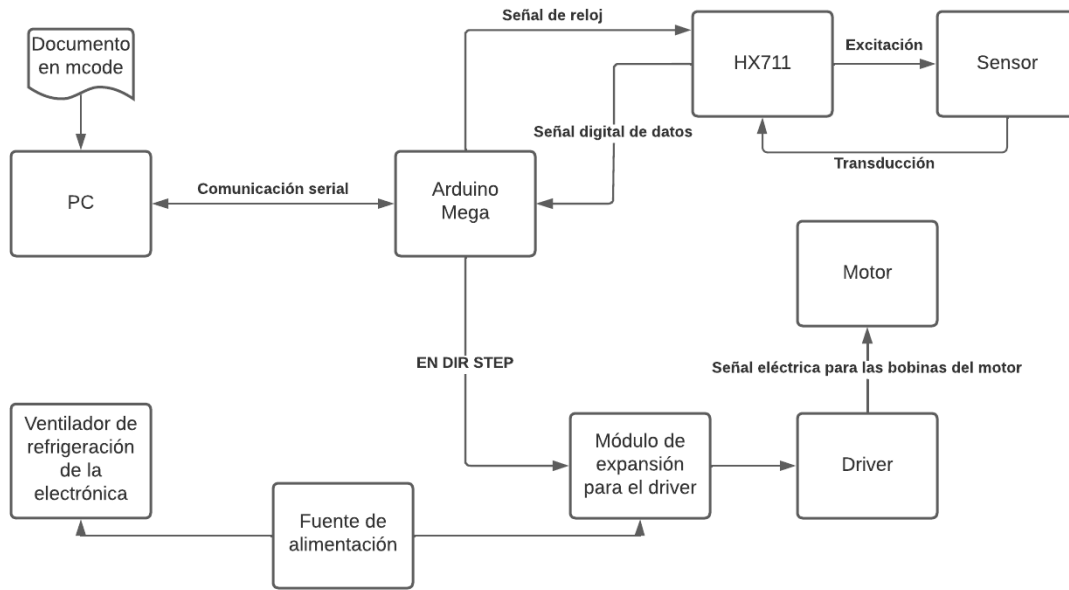
- M - Los comandos que empiezan por M indican que el motor tiene que realizar un movimiento. Este comando va acompañado de un primer valor que indica el motor o motores a mover y los dígitos siguientes indican la cantidad de milímetros a mover. Por ejemplo el comando M11 indican que hay que mover el motor 1 un milímetro.
- D - El comienzo por la letra D indica un cambio de dirección del motor. Este comando va acompañado de un dígito que indica el motor o motores a mover.
- E - Este comando es para habilitar o deshabilitar el motor o motores que indican el dígito que acompaña a esta letra.
- S - Indica el estatus de la dirección de los motores.
- N - El comando realiza el paso de datos de la lectura de los sensores.

El tener la capacidad de poder gestionar la máquina por medio de comandos hace que se puedan realizar todos los test que se requieran, siendo ilimitado el número de pruebas que se pueden realizar.

6.6. Visión general del sistema

Una vez ya implementados todos los componentes del sistema, se va a hacer una visión general del funcionamiento al completo de éste. Mediante un diagrama de bloques se puede observar cómo se relacionan los distintos componentes del sistema.





32

Todo el software del sistema se puede encontrar en un [repositorio GitHub](#), de tal forma que todo el mundo pueda colaborar en hacer de este sistema un gran sistema. Muchas de las ideas se pueden implementar fácilmente en el firmware del Arduino a modo de funciones para cada comando que se quiera añadir, ya que este era uno de los objetivos del proyecto, el ser un trabajo abierto al público.

³² Figura 32. Diagrama de bloques del sistema.



7. Pruebas realizadas

Las pruebas que se han realizado sobre el conjunto del sistema han sido tanto por la terminal del puerto serial para comprobar que el firmware funciona como por parte de la interfaz para comprobar que la comunicación entre los dos programas iba como se esperaba. El tener la máquina en físico ha ayudado a verificar que los datos que se mandaban eran los correctos de forma visual y se han podido corregir errores que de otra forma no hubieran sido capaces de detectar.

En cuanto a debugging, no se han incluido comentarios de ejecución para que se muestren por la terminal en segundo plano de la interfaz. Únicamente se muestran las excepciones que se han ido solventando según se iban haciendo las pruebas, ya que de no encontrar el puerto serial, el programa a veces se bloqueaba.

La temporización también ha sido muy importante, el determinar los 2s de pausa entre mandar el comando y esperar la respuesta ha sido realmente a base de prueba y error, ya que había veces que los sensores mandaban los datos de forma correcta y otras veces se truncaba la información por la cantidad y los valores no llegaban bien.

En la imagen siguiente se puede observar la terminal en segundo plano del programa en la que se muestran los puertos de comunicaciones debido a que es una parte fundamental del programa junto a una excepción que salta al no elegir ningún archivo de ejecución.

```
run:
jun. 28, 2022 12:10:32 A. M. javafx.fxml.FXMLLoader$ValueElement processValue
WARNING: Loading FXML document with JavaFX API of version 18 by JavaFX runtime of version 13.0.2
Lista de puertos: Arduino Mega 2560 (COM12)
Lista de puertos: Puerto de comunicaciones (COM1)
] Exception in thread "JavaFX Application Thread" java.lang.RuntimeException: java.lang.reflect.InvocationTargetException
|   at javafx.fxml/javafx.fxml.FXMLLoader$MethodHandler.invoke(FXMLLoader.java:1787)
|   at javafx.fxml/javafx.fxml.FXMLLoader$ControllerMethodEventHandler.handle(FXMLLoader.java:1670)
```




33

Como se ha comentado anteriormente, las excepciones han sido tratadas según iban apareciendo en las pruebas y ya no son bloqueantes como pasaba anteriormente.

Como se puede observar a continuación, la aplicación al tratarse de JavaFX consume más que otros lenguajes como podrían haber sido C++ o C#, sin embargo considero que no es un consumo que pueda comprometer ningún

³³ Figura 33. Terminal de la aplicación de escritorio.

ordenador, ya sea más nuevo o más antiguo, ya que lo que más consume es memoria RAM y 111MB tampoco es una gran cantidad.

▼  OpenJDK Platform binary (2)	0%	111,4 MB	0 MB/s	0 Mbps	0%	GPU 0 - 3D
 Host de ventana de consola	0%	105,7 MB	0 MB/s	0 Mbps	0%	
 Host de ventana de consola	0%	5,7 MB	0 MB/s	0 Mbps	0%	

34

³⁴ Figura 34. Consumo de la aplicación de escritorio.

8. Conclusiones finales

En conclusión a este trabajo, puedo decir que ha sido un desafío personal en cuanto a buscar las mejores soluciones en ámbitos en los que a lo mejor no me sentía cómodo por no ser objeto de estudio en la rama que he cursado. Sin embargo ha sido una motivación el poder trabajar en un proyecto que aúna a mi modo de ver casi todos los ámbitos de la informática como son el hardware, las comunicaciones y el desarrollo de aplicaciones software con interfaces gráficas.

En este trabajo he trabajado con tecnologías que ya conocía gracias a la carrera como JavaFX o la programación a bajo nivel, sin embargo también me he encontrado con dificultades a la hora de trabajar con Arduino al no haber trabajado sobre este tipo de microcontroladores. He tenido que buscar en foros y en la propia documentación de Arduino para poder realizar la comunicación serie, un elemento con el que no había trabajado antes.

También ha sido un desafío el entender las necesidades que pueden llegar a tener este tipo de máquinas y según iba trabajando me iban viniendo ideas que por tiempo no he podido llegar a implementar.

En definitiva y aunque he encontrado dificultades por el camino por tecnologías que no manejaba antes del trabajo, considero que se han cumplido todos los objetivos propuestos al principio del trabajo e incluso se han podido realizar de una forma mejor de la que en un inicio se había planteado.

Quizás el hecho de intentar abarcar un sistema muy complejo ha hecho que retrase ciertas partes del proyecto y que en mi opinión han sido una pérdida de tiempo que podría no haber tenido.

En cuanto al uso de motores paso a paso, el hecho de haber cursado la asignatura de impresión 3D, de la que el tutor de este TFG es profesor, me ayudó mucho en cuanto a comprender cómo funcionaban los motores paso a paso y decidir utilizar esta tecnología en mi proyecto.

Para acabar, debo decir que ha sido de gran ayuda todo lo aprendido en Java en la carrera, ya que en un principio planteé una interfaz en Python por los conocimientos que había obtenido en los últimos años de carrera pero al replantear toda la interfaz de cara a Java se solucionaron muchos problemas de los que pensaba que no podría salir. En este aspecto considero que la adquisición de las competencias de “Aplicación y pensamiento práctico” junto con la de “Diseño y proyecto” han marcado la diferencia a la hora de parar a pensar cuál era la mejor decisión.



9. Referencias

- I. Pereira, A. B., Fernandes, F. A., de Moraes, A. B., & Maio, J. (2020). Biaxial Testing Machine: Development and Evaluation. *Machines*, 8(3), 40. <https://doi.org/10.3390/machines8030040>
- II. *Event based reading sometimes gets incomplete data from MCU · Issue #197 · Fazecast/jSerialComm*. (2019, 13 abril). [Discusión]. GitHub. <https://github.com/Fazecast/jSerialComm/issues/197>
- III. *DRV8825 Stepper Motor Controller IC*. (s. f.). <https://www.ti.com/lit/ds/symlink/drv8825.pdf>
- IV. *Biaxial tensile tester (1.0)*. (2021). [Software]. Miguel Sanchez. <https://wikifactory.com/@misan/biaxial-tensile-tester>
- V. C. (2017, 20 enero). *Manejar un motor stepper con un driver DRV8825 y Arduino | Carlini's Blog*. Carlini. <http://carlini.es/manejar-un-motor-stepper-con-un-driver-drv8825-y-arduino/>
- VI. *Signal Input/Output - Arduino Reference*. (2022). Arduino. <https://www.arduino.cc/reference/en/libraries/category/signal-input/output/>
- VII. Beccarelli, P. (2015). *Biaxial Testing for Fabrics and Foils: Optimizing Devices and Procedures (SpringerBriefs in Applied Sciences and Technology) (English Edition)* (2015.^a ed.). Springer.



ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Proced
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.				X
ODS 4. Educación de calidad.	X			
ODS 5. Igualdad de género.				X
ODS 6. Agua limpia y saneamiento.			X	
ODS 7. Energía asequible y no contaminante.			X	
ODS 8. Trabajo decente y crecimiento económico.		X		
ODS 9. Industria, innovación e infraestructuras.	X			
ODS 10. Reducción de las desigualdades.				X
ODS 11. Ciudades y comunidades sostenibles.				X
ODS 12. Producción y consumo responsables.	X			
ODS 13. Acción por el clima.			X	
ODS 14. Vida submarina.		X		
ODS 15. Vida de ecosistemas terrestres.				X
ODS 16. Paz, justicia e instituciones sólidas.				X
ODS 17. Alianzas para lograr objetivos.		X		

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

El proyecto que se ha realizado tiene una gran importancia para tener una educación de calidad, ya que son proyectos de libre acceso en los que toda la comunidad puede participar en el desarrollo de este tipo de tecnologías. Esto hace que los estudiantes puedan realizar proyectos que de otra forma no hubieran podido hacer.

También es un proyecto en el que tecnologías dedicadas a la industria se realizan teniendo en cuenta siempre una producción responsable, en este caso se ha reutilizado



tecnología que ya no tenía utilidad transformándola en elementos de nuestro sistema. Tanto los motores como guías y elementos transmisores del movimiento han sido reciclados de una antigua impresora 3D que, de otra forma, hubiera sido para tirar.

En cuanto al crecimiento económico se puede decir que si esta tecnología se democratiza a personas no profesionales puede hacerles ganar valor en futuros trabajos. El tener experiencia en determinados aspectos como puede ser el saber realizar pruebas de test puede garantizar de cara a un futuro el trabajo, siendo un aliciente para contribuir al desarrollo del proyecto.

En cuanto a la acción por el clima, tiene un impacto bajo, ya que no se trata de un gran sistema que consuma una cantidad muy grande de energía, sin embargo, el tener una máquina eficiente en cuanto al consumo eléctrico ayuda si lo llevamos a gran escala, con electrodomésticos y máquinas que aunque consumen poco, si se tiene en cuenta todos a la vez pueden marcar la diferencia. Esto se relaciona también con el objetivo de tener energía asequible.

El objetivo de vida submarina se puede relacionar con el uso de plástico biodegradable como es el PLA, que aunque sigue siendo plástico, es menos perjudicial para el medio ambiente. Hay que recordar que la contaminación por microplásticos en nuestros océanos cada vez está teniendo más importancia en nuestro día a día, por lo que el uso de elementos no contaminantes es bastante importante.

En cuanto a los demás objetivos, están enfocados más en otro tipo de proyectos fuera del ámbito industrial, un marco en el que nuestro proyecto está encasillado. Este proyecto no está enfocado a mejorar el ámbito social, ya que al tratarse de una máquina de test, no mejora la calidad de vida que pueda tener una persona. De igual manera, no tiene una reducción de las desigualdades, ya que no es un proyecto que marque realmente la diferencia entre tenerlo y no tenerlo.

Finalmente, considero que el proyecto relaciona bastantes objetivos de desarrollo sostenible, ya que desde el principio se planteó un proyecto teniendo en mente el hecho de ser económico, funcional y sobre todo abierto a todo el mundo que quiera colaborar en el proceso.