UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ETSI Aeroespacial y Diseño Industrial

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño y realización de un sistema de inyección infusión de alta precisión, para aplicaciones de sensado en tiempo real

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Olmos Saldías, Luis Fernando

Tutor/a: Ponce Alcántara, Salvador

CURSO ACADÉMICO: 2023/2024

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ETSI Aeroespacial y Diseño Industrial

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TRABAJO DE FIN DE GRADO

Diseño y realización de un sistema de inyección – infusión de alta precisión, para aplicaciones de sensado en tiempo real

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

AUTOR: LUÍS FERNANDO OLMOS SALDÍAS

TUTOR: SALVADOR PONCE ALCÁNTARA

Valencia, Julio 2024

Resumen

Una bomba de jeringa, también conocido como inyector, es una bomba de tamaño reducido de infusión y extracción de líquidos, que es utilizado en investigaciones químicas y biomédica debido a que puede administrar gradualmente cantidades muy pequeñas de fluido. El diseño consiste en el control de un motor al que se le conecta un sistema mecánico que desplazará el émbolo de una jeringa, variando así el volumen succionado o inyectado. En dicha bomba se pueden diferenciar dos bloques:

- Un bloque mecánico formado por la jeringa y el motor que mueve el émbolo de esta.

- Un bloque electrónico, que está formado un display LCD encargado de mostrar información al usuario, unos pulsadores que nos permitirán acceder a las distintas funciones de la bomba, un encoder rotativo que proporcionará una señal eléctrica que confirma el giro del motor, y un microcontrolador responsable de gestionar los componentes mencionados.Los laboratorios del Instituto Universitario de Tecnología Nanofotónica de Valencia disponen de diferentes modelos de bombas usadas para la realización de pruebas de sensado sobre sensores fotónicos. A pesar de que estas bombas ofrecen garantías de precisión y repetibilidad, sus elevados costes, así como los tiempos de envío o reparación, son sus principales inconvenientes a la hora de realizar su adquisición.

En este proyecto tiene como finalidad la reutilización de una bomba de jeringa que se encontraba estropeada. Para ello se ha empleado software libre y componentes accesibles.

Considerando lo anterior, a continuación se aborda el diseño del sistema de control de la bomba de jeringa, basado en software libre y componentes de fácil adquisición, y se utilizará la propia matriz de una bomba de jeringa dañada, aprovechando la parte mecánica y su estructura.

Durante el desarrollo del proyecto se explicará detalladamente el proceso de selectividad de cada componente, así como la programación del microcontrolador. Se ha destinado también un apartado al diseño del circuito impreso necesario, y a las pruebas realizadas para llevar a cabo la calibración de la bomba.

Por último, se expondrán y valorarán los resultados obtenidos de las pruebas experimentales, y se evaluará el proyecto tanto técnica como económicamente para un posterior uso práctico en cualquier laboratorio.

Palabras clave

Bomba de jeringa, flujo, sensado, motor paso a paso, encoder, display LCD, engranaje, controlador, Arduino.

Summary

A syringe pump, also known as an injector, is a small-sized infusion and extraction pump used in chemical and biomedical research because it can gradually administer very small amounts of fluid. The design involves controlling a motor connected to a mechanical system that displaces the syringe plunger, thereby varying the volume sucked in or injected. This pump can be divided into two blocks:

- A mechanical block consisting of the syringe and the motor that moves its plunger.
- An electronic block, which includes an LCD display that shows information to the user, buttons that allow access to the pump's various functions, a rotary encoder that provides an electrical signal confirming the motor's rotation, and a microcontroller responsible for managing the aforementioned components.

The laboratories of the Nanophotonics Technology University Institute of Valencia have different models of pumps used for sensing tests on photonic sensors. Despite these pumps offering guarantees of precision and repeatability, their high costs, as well as shipping or repair times, are their main disadvantages when it comes to acquisition.

This project aims to reuse a syringe pump that was broken. To achieve this, free software and accessible components have been used. Considering the above, the following addresses the design of the syringe pump control system, based on free software and easily obtainable components, utilizing the mechanical part and structure of a damaged syringe pump.

The project development will detail the process of selecting each component and programming the microcontroller. A section is also dedicated to designing the necessary printed circuit, and the tests carried out to calibrate the pump.

Finally, the results obtained from the experimental tests will be presented and evaluated, and the project will be assessed both technically and economically for subsequent practical use in any laboratory.

Keywords

Syringe pump, flow, sensing, stepper motor, encoder, LCD display, gear, controller, Arduino.

Resum

Una bomba de xeringa, també conegut com a injector, és una bomba de grandària reduïda d'infusió i extracció de líquids, que és utilitzat en investigacions químiques i biomèdica pel fet que pot administrar gradualment quantitats molt xicotetes de fluid. El disseny consistix en el control d'un motor al qual se li connecta un sistema mecànic que desplaçarà l'émbol d'una xeringa, variant així el volum succionat o injectat. En esta bomba es poden diferenciar dos blocs:

- Un bloc mecànic format per la xeringa i el motor que mou l'émbol d'esta.

- Un bloc electrònic, que està format un display LCD encarregat de mostrar informació a l'usuari, uns polsadors que ens permetran accedir a les diferents funcions de la bomba, un encoder rotatiu que proporcionarà un senyal elèctric que confirma el gir del motor, i un microcontrolador responsable de gestionar els components esmentats.Els laboratoris de l'Institut Universitari de Tecnologia Nanofotònica de València disposen de diferents models de bombes usades per a la realització de proves de sensado sobre sensors fotònics. A pesar que estes bombes oferixen garanties de precisió i repetitivitat, els seus elevats costos, així com els temps d'enviament o reparació, són els seus principals inconvenients a l'hora de realitzar la seua adquisició.

En este projecte té com a finalitat la reutilització d'una bomba de xeringa que es trobava desbaratada. Per a això s'ha emprat programari lliure i components accessibles.

Considerant l'anterior, a continuació s'aborda el disseny del sistema de control de la bomba de xeringa, basat en programari lliure i component de fàcil adquisició, i s'utilitzarà la pròpia matriu d'una bomba de xeringa danyada, aprofitant la part mecànica i la seua estructura.

Durant el desenrotllament del projecte s'explicarà detalladament el procés de selectivitat de cada component, així com la programació del microcontrolador. S'ha destinat també un apartat al disseny del circuit imprés necessari, i a les proves realitzades per a dur a terme el calibratge de la bomba.

Finalment, s'exposaran i valoraran els resultats obtinguts de les proves experimentals, i s'avaluarà el projecte tant tècnica com econòmicament per a un posterior ús pràctic en qualsevol laboratori.

Paraules clau

Bomba de xeringa, flux, sensado, motor pas a pas, encoder, display LCD, engranatge, controlador, Arduino.

Me gustaría dar las gracias a mis padres Edil y Yoly, a mi hermana Ana Paola y a mi cuñado Joaquín por todo el apoyo mostrado durante todo el proyecto. A José, Fernando, Nicolás y Bernat por haberme acompañado y alegrado durante este camino en la universidad. Y para finalizar quería mostrar todo mi agradecimiento y respeto a Salva, mi tutor de este proyecto y uno de los mejores profesores que he tenido durante mi etapa en la UPV.

Contenido

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ETSI Aeroespacial y Diseño Industrial

UNIVERSIDAD POLITÉCNICA DE VALENCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño y realización de un sistema de inyección – infusión de alta precisión, para aplicaciones de sensado en tiempo real

DOCUMENTO Nº1: MEMORIA TÉCNICA

Trabajo final de grado

Grado en Ingeniería Electrónica Industrial y Automática

Autor: Olmos Saldías, Luis Fernando

Tutor: Ponce Alcántara, Salvador

Índice

Índice de imágenes

Capítulo 1. Objeto

El presente Trabajo de Fin de Grado tiene como objeto el reacondicionamiento e implementación de una bomba de jeringa, para satisfacer las necesidades de los laboratorios del Instituto de Investigación de Tecnología Nanofotónica de Valencia, en aplicaciones de sensado. Por ello, no deberá disponer de un gran tamaño, debe tener una interfaz sencilla para que cualquier usuario pueda modificar parámetros.

Para alimentar el sistema, se emplea un convertidor AC/DC de 12 V y 2 A, capaz de alimentar toda la electrónica necesaria, incluyendo el motor paso a paso. Además, se añade un regulador de tensión 7805 destinado a proporcionar 5 V a las diferentes entradas o salidas digitales como los pulsadores y el display LCD.

Asimismo, se dispondrá de una interfaz de usuario sencilla y de fácil operación. Para ello se han seleccionado todos los componentes necesarios para que la bomba sea intuitiva, incluyendo además un notable control sobre los flujos programados. Para proporcionar información al usuario, se dispone de una pantalla LCD de dos líneas y dieciséis caracteres, en la que se podrán observar los distintos parámetros asociados a la bomba. En este sentido, se dispone de un menú de ajustes en el que se podrá cambiar parámetros como el flujo deseado, el sentido del flujo (succión o inyección), y el diámetro de la jeringa. La selección de los parámetros se realizará mediante tres pulsadores.

Así mismo, el programa desarrollado permitirá al usuario de modificar el flujo en tiempo real.

Además, cabe recalcar que el proyecto emplea un software libre, él cual es accesible para otros usuarios y adaptable sus necesidades.

Capítulo 2. Situación de partida y estudio de necesidades

2.1 Situación de partida

El empleo de bombas de jeringa es muy habitual en diversos campos de investigación, especialmente aquellos vinculados con aplicaciones biomédicas y químicas. Pese a que estos dispositivos suelen ser de alta calidad, su coste suele oscilar entre 1000 y 3500 € [1].

Además, su plazo de entrega que suele superar los 30 días, dificultando considerablemente tanto su accesibilidad, como su empleo. Además, se pueden tener problemas con el funcionamiento de la bomba, lo que implica costes adicionales en reparaciones y posibles retrasos en proyectos de investigación que se estuviesen llevando a cabo con el uso de la bomba.

El Instituto Universitario de Tecnología Nanofotónica de Valencia está llevando a cabo medidas de sensado empleando sensores fotónicos. Dichos sensores modifican su respuesta óptica al fluir sobre ellos ciertas disoluciones, y en ese paso de fluídica se hace obligatorio el uso de algún tipo de bomba. En este sentido, actualmente los laboratorios emplean dos clases de bombas: las peristálticas y las de jeringa. Ambas bombas las podemos observar en las Imágenes 1 y 2 respectivamente presentadas a continuación.

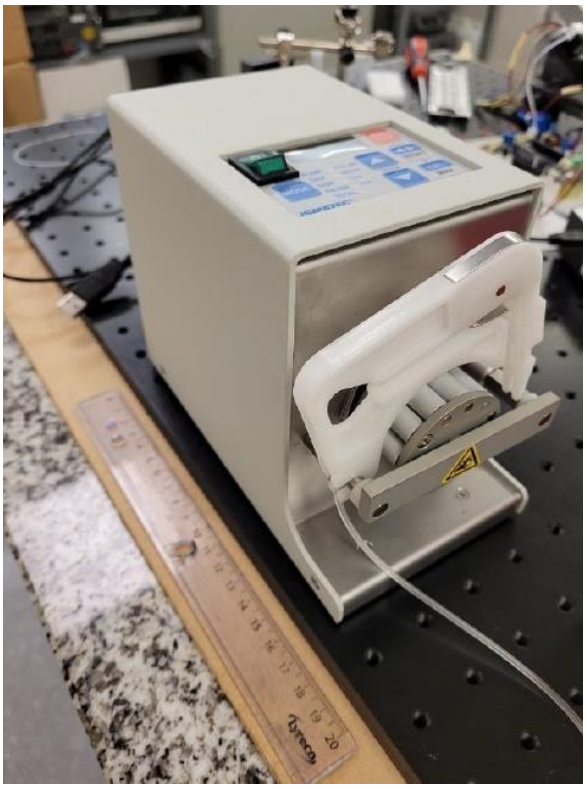


Imagen 1. Bomba peristáltica ISMATEC

Imagen 2. Bomba de jeringa SyringePumpPro

En la Imagen 1 se puede observar una bomba peristáltica. Su funcionamiento se basa en la compresión y descompresión de un flexible mediante una serie de rodillos o zapatas que giran (Imagen 3).

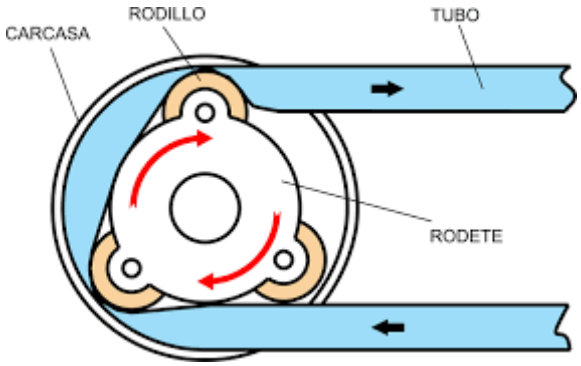


Imagen 3. Principio de funcionamiento de una bomba peristáltica [2]

En la imagen 2 se aprecia una bomba de jeringa comercial. El principio de funcionamiento se basa en el movimiento lineal de una jeringa desechable tal como se observa en la imagen 4.

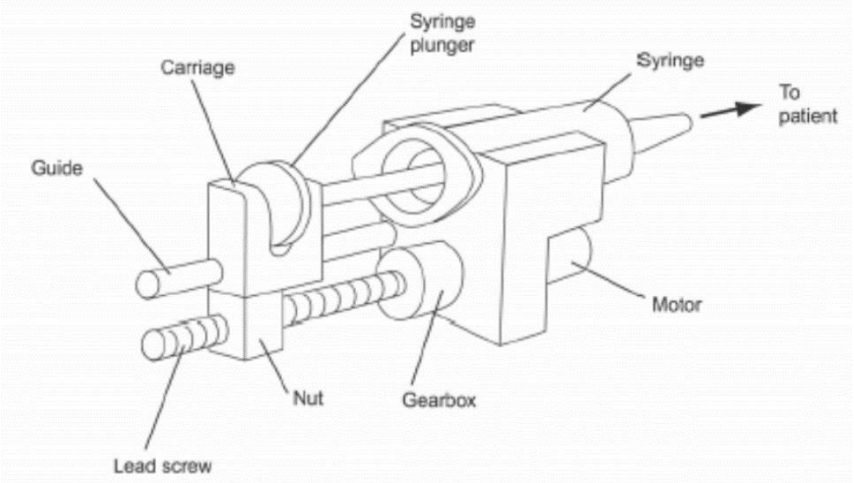


Imagen 4. Esquema de funcionamiento de una bomba de jeringa [3]

Las bombas peristálticas tienen tres inconvenientes notables respecto a las bombas de jeringa:

- La precisión de flujo que suelen ofrecer.
- La compatibilidad con diferentes fluidos.
- Vida útil limitada de la manguera.

Mientras que las bombas de jeringa nos ofrecen:

- Un flujo continuo y estable debido a su principio de funcionamiento (Imagen 6).
- Un alto grado de repetitividad.

Teniéndolas en cuenta, se ha valorado la reutilización de una que se encontraba estropeada.

Por consiguiente, este proyecto tiene como finalidad el reacondicionamiento de una bomba de jeringa. Ha de ser capaz de fluir líquidos de manera continua y con flujos del orden de pocos microlitros por minuto (uL/min), de forma similar a lo presente en las características de la bomba original.

Uno de los motivos que impulsaron la idea del proyecto es el alto precio de las bombas de jeringa comerciales. El objetivo es reacondicionarla, de manera que su coste y su reparación sean muy inferiores a las bombas de jeringa del mercado. Por ello, se ha optado por el empleo de componentes de baja adquisición económica, mínimo consumo eléctrico, y un tamaño reducido. Aprovechando la matriz de la bomba, se diseñará el circuito electrónico considerando que ha de ajustarse a las dimensiones de esta.

2.2 Setup de medida de sensores fotónicos

Los sensores fotónicos son aquellos cuya respuesta óptica se modifica al pasar sobre ellos una solución determinada. De cara a realizar un experimento de sensado, la Imagen 5 presenta un esquema típico de un setup de medida utilizado.

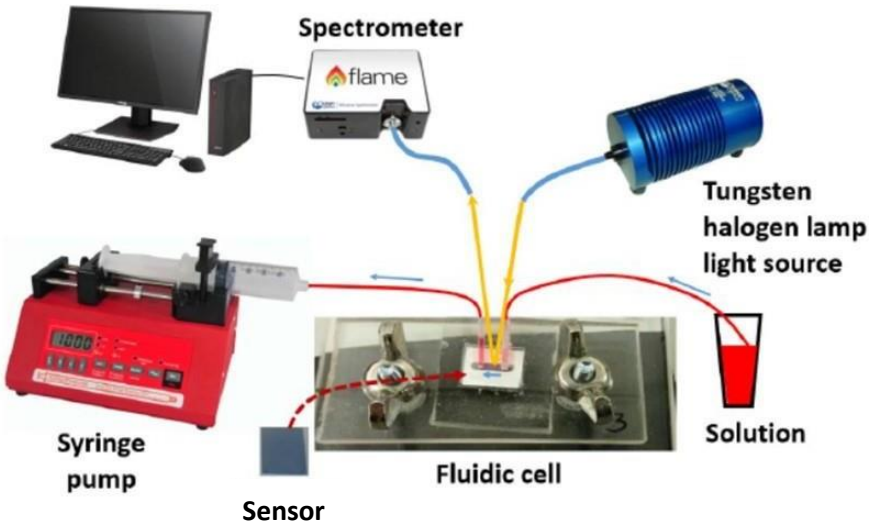


Imagen 5. Esquema de un sistema de medida que emplea sensores fotónicos para realizar pruebas de detección en tiempo real [4]

El sensor se sitúa en una celda de metacrilato. Ésta posee un canal a través del cual se hace fluir una solución determinada. La celda posee dos tubos de silicona conectados al canal de sensado, que permiten uno la entrada y otro la salida de la solución. El tubo de entrada se introduce en un recipiente (eppendorf) que contiene la solución que se desea analizar, y el de salida se conecta a una bomba de jeringa. Dicha bomba ha de ser capaz de succionar la comentada solución, con un flujo determinado. Dicho flujo suele ser de pocos microlitros por minuto (uL/min).

El sensor es iluminado mediante una lámpara halógena, y su reflectividad es recogida por una fibra óptica conectada a un espectrómetro. El espectrómetro se conecta a un ordenador que, mediante el software adecuado, permite detectar variaciones en el espectro del sensor en tiempo real.

De acuerdo con lo anterior, una parte imprescindible del setup de medida es la bomba de jeringa. Y la reutilización de una de estas bombas mediante el rediseño de la parte electrónica será el objetivo de este proyecto.

2.2 Requerimientos

Para desarrollar correctamente el proyecto, es fundamental definir ciertos criterios iniciales que deben ser alcanzados para lograr el objetivo establecido de ofrecer una herramienta funcional y sencilla de utilizar.

Dichos requerimientos son:

- Flujo entre 5 uL/min y 120 uL/min.
- Un tamaño del circuito electrónico que pueda ser colocado en el interior de la bomba.
- Bajo consumo de electricidad.
- Costo reducido.
- Diseño simple.
- Manejo sencillo e intuitivo para cualquier usuario.
- Continuidad de flujo. Este último punto es de vital importancia. Se debe asegurar la continuidad del flujo, principalmente cuando éste es inferior a 10 uL/min.

2.3 Relación con los ODS

Los Objetivos de Desarrollo Sostenible (Imagen 6), son un conjunto de 17 metas globales adoptadas por las Naciones Unidas en 2015 como parte de la agenda 2030 para el desarrollo sostenible. Su finalidad es abordar diferentes desafíos globales [5]. Cada objetivo posee metas específicas para ser alcanzadas en los próximos 15 años.

El presente proyecto está alineado con varios ODS establecidos por la ONU. En primer lugar, el ODS 9, centrado en Industria, Innovación e Infraestructura, se relaciona con el avance tecnológico e innovación que implica la fabricación de dispositivos microfluídicos. Asimismo, el ODS 3, enfocado en Salud y Bienestar, está vinculado a través del monitoreo de parámetros de salud y la administración precisa de medicamentos que este proyecto facilita. Además, este trabajo se alinea con el ODS 4, que promueve la Educación de Calidad, al fomentar la investigación, el desarrollo de tecnologías innovadoras y sus aplicaciones potenciales en la formación de profesionales en los campos de la salud y la ciencia. También con el ODS 13 debido a que se pretende adaptar medidas para combatir el cambio climático y sus efectos, al aprovechar la matriz de una bomba que se iba a desechar. Con ello se ahorra recursos y energía, promoviendo así una gestión sostenible de los recursos y una reducción de los desechos electrónicos.

Imagen 6. Objetivos de desarrollo sostenible [5]

2.4 Diagrama de Gantt

Previamente a comenzar con el proceso de reacondicionamiento de la bomba, se ha realizado un diagrama de Gantt. En él se representa gráficamente el cronograma de trabajo asociado a la estructura organizativa que se debe seguir para la realización de este proyecto. Dicho diagrama muestra el tiempo en semanas asociadas a cada tarea necesaria para la consecución del proyecto. Para ello se ha estimado unas 20 horas de dedicación al proyecto semanales. Hay que tener en cuenta que se trata de una estimación, pues pueden surgir inconvenientes e imprevistos durante la realización del proyecto. En cualquier caso, ayudará y servirá de base para el seguimiento del proyecto. En la siguiente imagen (Imagen 7) se puede observar la programación que se realizó para el diseño e implementación del proyecto. Donde cada línea naranja representa 14 días y como consiguiente cada línea azul son 7 días.

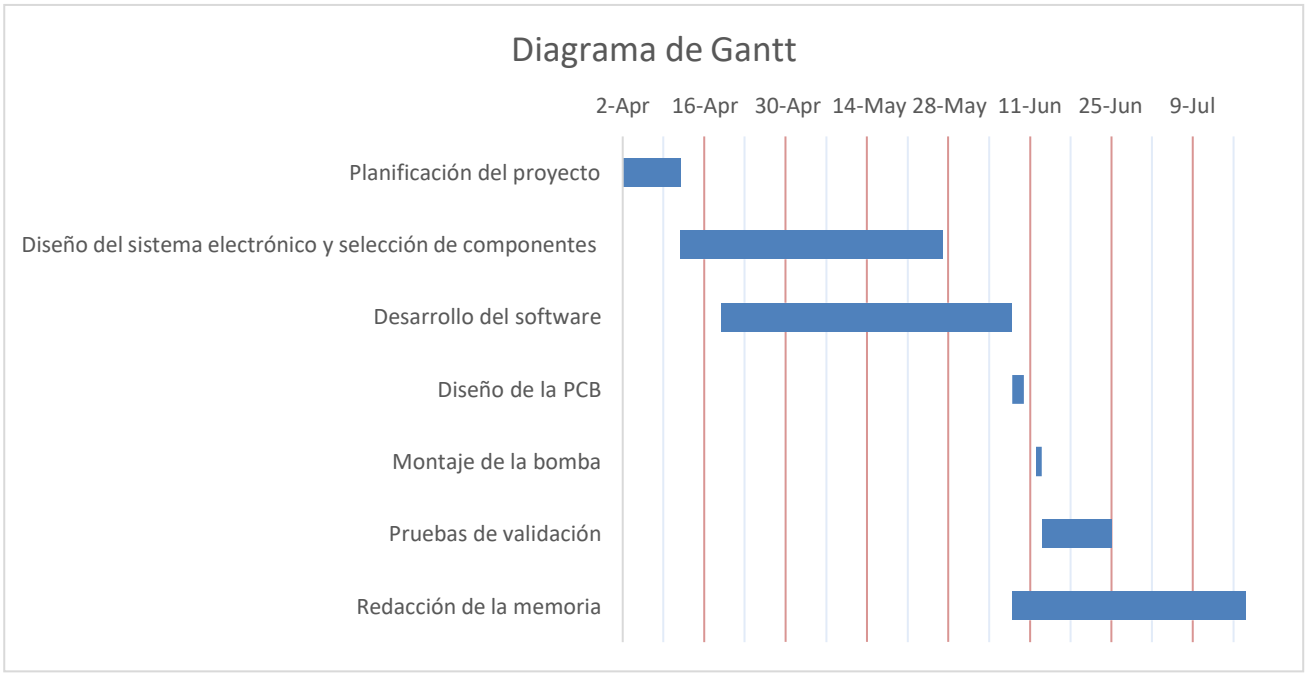


Imagen 7. Diagrama de Gantt del proyecto

Capítulo 3. Descripción detallada de la solución optada

3.1 Vision general del sistema

La bomba diseñada en este proyecto se basa en un motor paso a paso que proporciona un control preciso de la velocidad de rotación requerida. El motor es alimentado y controlado mediante un driver, asegurando un rendimiento óptimo. El sistema está controlado por un microcontrolador Arduino. Arduino es una plataforma de hardware libre y fácilmente programable, lo que permite más flexibilidad y capacidad de adaptarse al sistema.
Para lograr la precisión en el flujo que buscamos, nuestro proyecto empleará un encoder rotatorio que la matriz de la bomba ya poseía y que enviará una señal eléctrica al microcontrolador con la información sobre los pulsos realizados. Con esta información el microcontrolador compara velocidad calculada con la velocidad estipulada en la calibración y la ajusta.
El sistema dispondrá de una pantalla LCD, que ofrece una interfaz gráfica para que el usuario pueda visualizar y navegar por un menú de opciones. La navegación y la interacción del menú se llevará a cabo mediante tres pulsadores, con los cuales facilitan la configuración y ajuste de parámetros del sistema.
El eje del motor está conectado mediante correa de goma con una serie de engranajes de difícil acceso, por lo que la relación entre los giros de cada parte se realizará de forma experimental. Todas las velocidades se calcularán en función del diámetro de la jeringa.

3.2 Dimensionamiento

Como se observa en la Imagen 4, este movimiento es realizado por un motor con un cabezal unido al eje cuyo diámetro y número de dientes es desconocido. Dicho cabezal está conectado a una serie de engranajes permitiendo una reducción de la velocidad de giro del motor, así como obtener más fuerza de giro, y con ello, más capacidad de succión. Internamente la segunda rueda estará conectada por un tornillo sin fin que girará a una velocidad reducida, previamente mencionada. En este tornillo sin fin podemos observar una base fija y un apoyo móvil donde colocaremos la jeringa cuyo émbolo se desplazará en función del giro del motor.
Como se ha adelantado, al no conocer las características de los engranajes se estimará relación de reducción que obtenemos realizando el cálculo de los flujos con diferentes retardos.

Teniendo en cuenta lo comentado, las Imágenes 8 y 9 presentan la bomba que se va a reacondicionar a lo largo de este proyecto, así como el motor y parte de los engranajes disponibles.

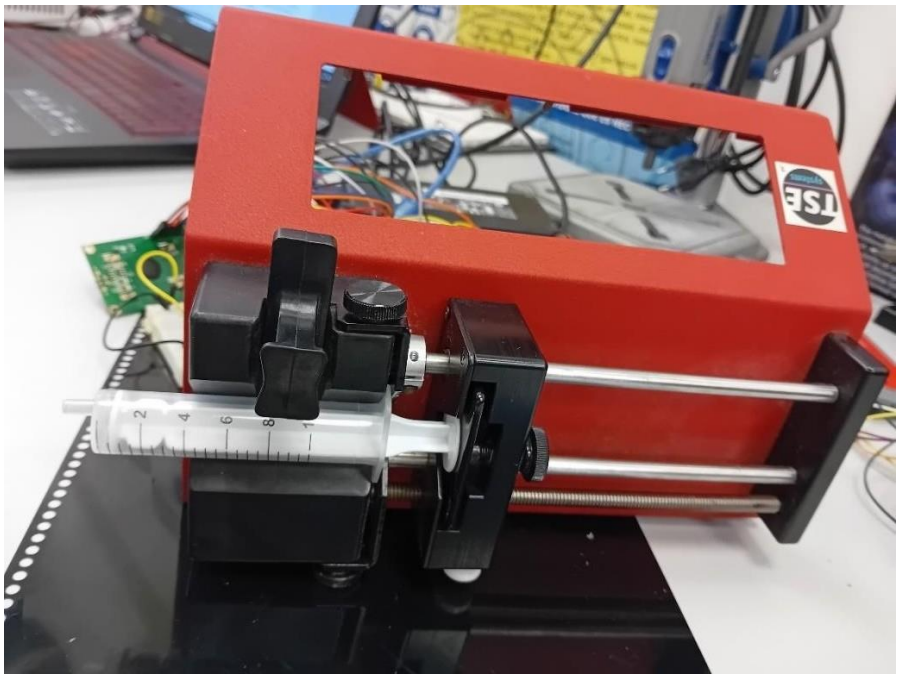


Imagen 8. Matriz de sujeción de la jeringa.

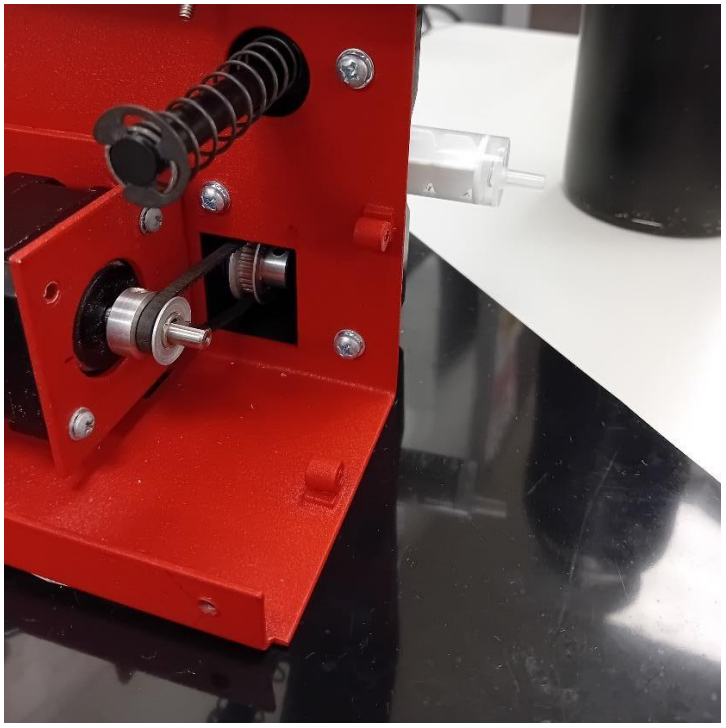


Imagen 9. Sistema mecánico presente entre la matriz y el motor paso a paso.

3.3 Control del motor

3.3.1 Esquema del sistema de control del motor

Los componentes necesarios para el control del motor deben seguir el esquema de la Imagen 10, en el que se pueden observar líneas rojas para alimentación y líneas negras que proporcionarán información.

Una fuente de alimentación debe estar conectada a un regulador de tensión y al driver del motor. El regulador, a su vez, debe proporcionar el suministro de energía al encoder, al microcontrolador y al driver. El driver recibe alimentación directamente de la fuente de alimentación para enviar energía a las bobinas del motor paso a paso. Sin embargo, el regulador de tensión le proporciona la alimentación necesaria para la lógica de control y las señales de entrada del driver.

Una vez alimentados todos los componentes, el microcontrolador envía información al driver, lo cual es crucial para regular la alimentación del motor según los requerimientos. Con esta alimentación, el encoder, instalado en serie con el motor, detecta el movimiento y la posición del motor, transformando estos datos en pulsos eléctricos que son enviados de vuelta al microcontrolador. Finalmente, el microcontrolador evalúa estos pulsos y envía nuevamente la información necesaria al driver para alimentar el motor, repitiéndose el proceso.

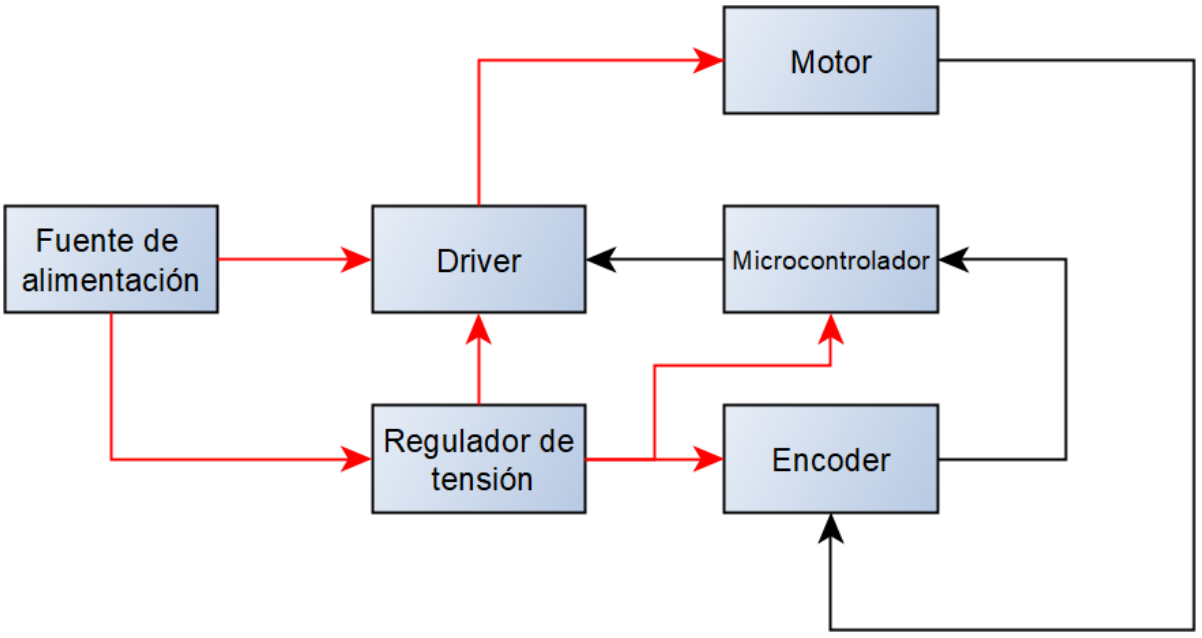


Imagen 10. Esquema del sistema de control del motor.

3.3.2 Motor

Como ya se ha comentado antes, el otro componente fijo que se encontraba incorporado en la matriz es el motor. Puede apreciarse junto con el encoder en la Imagen 11. Se ha empleado de la función monitor serie para conocer en tiempo real las variables donde se ha contado con una función el número de pasos que debe realizar para dar una vuelta.

Se trata de un motor paso a paso de 200 pasos por vuelta, es decir que por cada paso girará 1.8 grados. Su alimentación es de 12 voltios requiriendo una corriente de alimentación de 290 mA. El consumo de corriente más alto en estado parado se debe principalmente a la necesidad de mantener la posición precisa del rotor mediante la generación de un torque de retención constante.

Estos motores son los más indicados para el cumplimiento de los requisitos debido a su control precioso de velocidad y posición. A pesar de que los motores de corriente continua son los más utilizados, los motores paso a paso ofrecen la precisión necesaria para esta función.

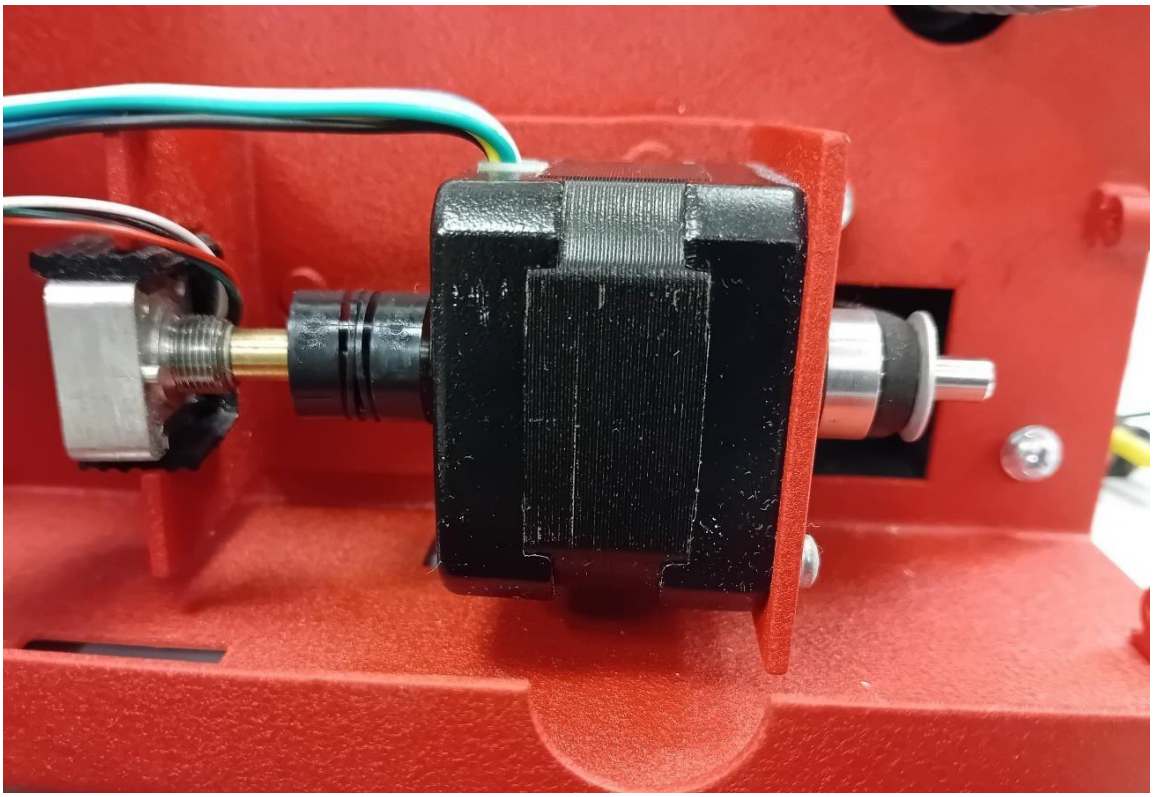


Imagen 11. Motor incorporado en la matriz.

3.3.3 Driver

Los drivers son fundamentales a la hora de un correcto control de un motor paso a paso. Con ellos se activan las bobinas del motor, se regula la intensidad para un correcto funcionamiento de este, aplicando también protecciones contra sobreintensidades, sobretensiones y sobrecalentamientos. Además, el driver es un elemento esencial para controlar de forma sencilla un motor de alta potencia mediante señales de bajo voltaje.

Se han valorado el uso de diferentes tipos de drivers para la implementación del proyecto. Al tener espacio suficiente para una selectividad más amplia, se han analizado tres tipos de drivers o controladores. Dichos drivers son: el A4988, el DRV8825, y el TB6600.

.

A4988

Se trata de un controlador muy popular en proyectos de electrónica que utilizan motores paso a paso. Sus principales características son:

- Corriente máxima por bobina que puede proporcionar al motor: 2 A.
- Permite alimentar al motor con tensiones de hasta 37 V.
- Posee unas dimensiones reducidas: 15 x 20 mm.
- Soporta una reducción en los pasos del motor de 1/16.

Imagen 12. Driver A4988 [6]

DRV8825

Igual que el A4988, el DRV8825 es un controlador muy utilizado en proyectos de electrónica con motores paso a paso. Sus características son:

- Corriente máxima por bobina que proporciona es de 2.5 A.
- Puede proporcionar una alimentación al motor de 45 V.
- Posee unas dimensiones de 15.5 x 20.5 mm.
- Soporta una reducción al motor de 1/32.

Imagen 13. Driver DRV8825 [7]

TB6600

El modelo TB6600 a pesar de tener unas dimensiones superiores a los mencionados anteriormente, sus características y su eficacia de funcionamiento nos permiten valorar su implementación en el circuito. Posee características como:

- Corriente máxima por bobina que proporciona es de 4 A
- Puede proporcionar una alimentación al motor de 40 V.
- Posee unas dimensiones de 96 x 56 x 33 mm.
- Soporta una reducción al motor de 1/32.

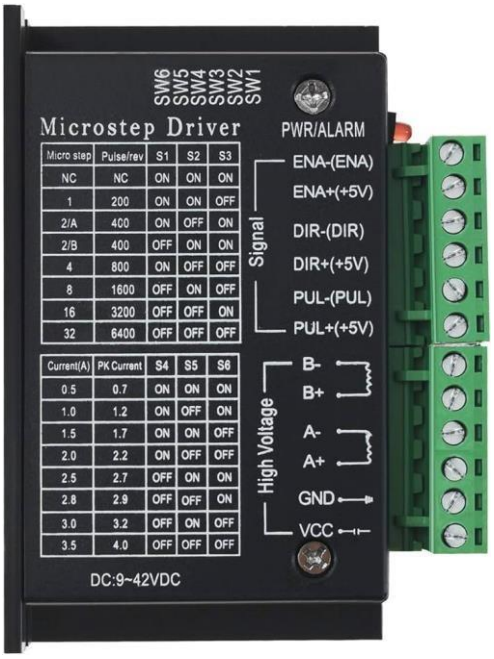


Imagen 14. Driver TB6600 [8]

Aunque los tres controladores pueden gestionar el motor paso a paso cumpliendo con los requisitos del proyecto, hemos optado por el modelo TB6600. Esta decisión se basa en la experiencia previa, donde los otros modelos presentaban fallos frecuentes durante las largas pruebas de sensado, las cuales pueden durar incluso más de 8 horas. Además, resulta económico, situándose su precio en torno a 5 €. A pesar de su mayor tamaño en comparación con las otras dos opciones, se dispone del espacio suficiente para su integración en la bomba.

Se emplea la reducción de 1/8, resultando 1600 pasos por vuelta. Se ha seleccionado este valor después de realizar pruebas experimentales a la mínima velocidad de flujo requerida (5 uL/min), donde presentó un flujo continuo durante todo el proceso.

3.3.4 Encoder

La función de un encoder conectado al motor es supervisar el giro del motor y corregir un posible error en la velocidad, asegurando un funcionamiento óptimo. No es necesario un proceso de selectividad del encoder debido a que la matriz de la bomba de jeringa llevaba uno incorporado. Como bien se ha comentado durante el trabajo, las condiciones esenciales de la bomba de jeringa son la precisión y la continuidad del flujo. Por tanto, con una retroalimentación con el encoder nos aseguramos más fiabilidad en el diseño y funcionamiento.

Así, la función del encoder es medir la velocidad de giro del motor y compararla con la velocidad que debería llevar, obtenida mediante una calibración de la bomba. En caso de ser diferentes, se ajustará esta velocidad mediante una modificación en el programa, y se volverá a comparar con la velocidad que debería llevar hasta

que se haya ajustado correctamente. Se comentará este procedimiento de forma detallada en el punto 3.5, dedicado al programa realizado para el microcontrolador que gobernará el funcionamiento de la bomba.

El encoder proporcionado en la bomba es rotativo de 20 pulsos por revolución (ppr), Posee una relación de 1 cada 10 pasos del motor. Se ha obtenido este valor al programar una función en la que el motor realice una vuelta completa y sume todos los pulsos eléctricos enviados por el encoder. Se ha utilizado la función monitor serie para comprobar el número de pulsos por vuelta en tiempo real.

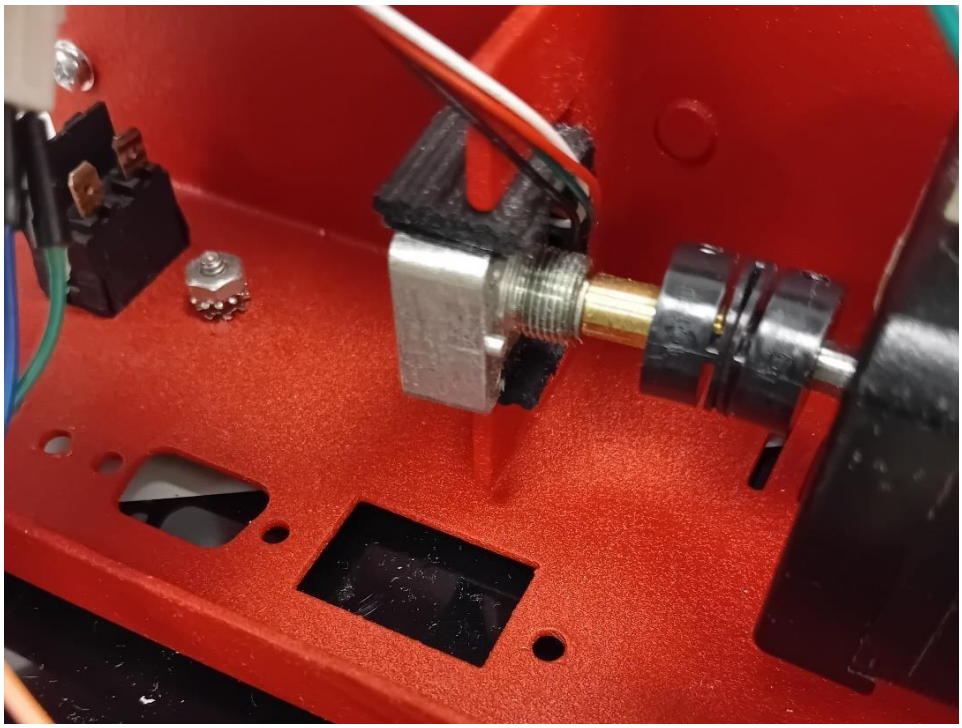


Imagen 15. Fotografía del encoder rotatorio de la bomba de jeringa

3.3.5 Microcontrolador

El microcontrolador que se ha seleccionado para este proyecto es un Microcontrolador Arduino Nano (Imagen 16). Siguiendo uno de los requisitos un software de diseño simple, para poder adaptarlo a nuevas necesidades si son requeridas, y libre, para que sea accesible para otros usuarios. El microcontrolador seleccionado ha sido el de Arduino. Tiene un lenguaje de programación basado en C++, con algunas adaptaciones que facilitan su uso. Los modelos de Arduino están constituidos por el controlador ATmega328P. Entre los diferentes modelos de Arduino en el mercado se ha procedido a elegir para el proyecto el modelo Arduino Nano.

Consta de unas reducidas dimensiones de 43 x 15 mm. Necesita una fuente de alimentación de 5 V, que se aplicará directamente al pin 5 V, bypasando el regulador de voltaje que posee dicha placa. Proporcionando directamente la alimentación se consigue una mayor eficiencia energética, y una reducción en la temperatura

del controlador. Además de diferentes pines de alimentación regulada, el modelo Arduino nano dispone de 14 entradas/salidas digitales y 8 entradas/salidas analógicas.

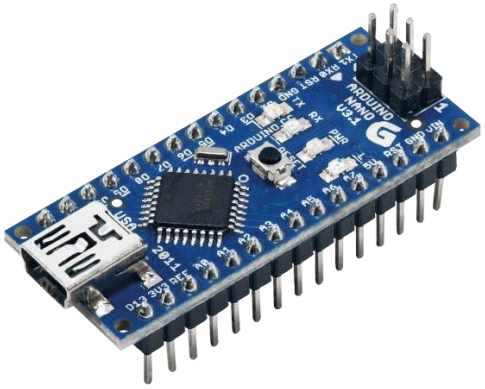


Imagen 16: Microcontrolador Arduino NANO [9]

3.3.6 Fuente de alimentación de 12 V

En este punto se ha optado por utilizar un convertidor AC/DC de 230 VAC AC a 12 VDC, y una corriente máxima igual o mayor a 1 A [Imagen 13]. Lo que hace de él una opción simple, económica y que nos proporciona adaptabilidad al entorno de la investigación pues únicamente es necesaria una toma de corriente, y nos ocuparía muy poco volumen en comparación de una fuente de tensión. Entre las diferentes opciones del mercado, que proporcionan diferentes valores de corriente, es necesario que disponga de 1 A para asegurar una correcta alimentación de los componentes. Debido a que debe suministrar corriente tanto al motor como al microcontrolador, al display LDC, al driver, al encoder y a los pulsadores que serán entradas digitales. Tanto porque la bomba se utilizará durante muchas horas seguidas como por su disponibilidad y precio, la corriente seleccionada para la fuente de alimentación ha sido de 1 A.

Imagen 17. Adaptador de pared de 230 V a 12 V [10]

3.3.7 Regulador de voltaje

Para alimentar la parte de control de la bomba se requiere de una tensión de 5 V. Para obtenerla se ha optado por el uso de un regulador de tensión 7805 situado a la salida de la fuente de alimentación de 12 V. En el mercado, las dos opciones más comunes son las siguientes:

7805

- Tensión de salida 5V.
- Tensión de entrada entre 7.5 y 30 V.
- Corriente de salida máxima 1.5 A.

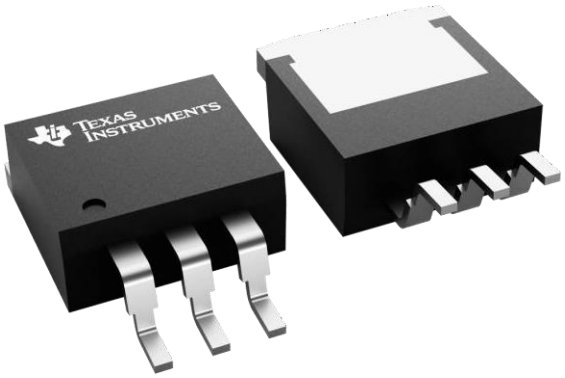


Imagen 18. Regulador de tensión 7805[11]

LM317

- Tensión de salida variable entre 1.25 y 37 V
- Tensión de entrada entre 3 y 40 V
- Corriente máxima de salida 1.5 A.

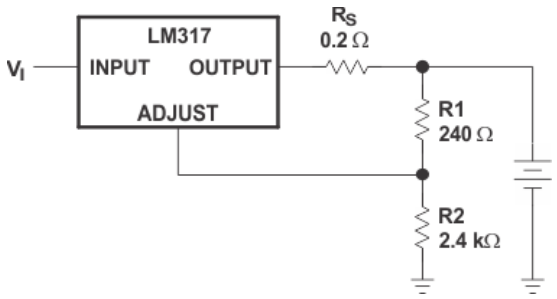


Imagen 19. Regulador de tensión variable LM317[12]

A pesar de que ambos componentes pueden cumplir la función correctamente se ha optado por escoger el 7805 debido a su simplicidad, fiabilidad, disponibilidad y su bajo coste. Además, puede proporcionar una corriente más que suficiente. A la salida de dicho componente se situará un filtro formado por un condensador de 100 uF.

3.4 Interfaz de usuario

El requisito a la hora de la implementación de la interfaz de usuario es que su control sea sencillo e intuitivo, para que cualquier persona pueda ser capaz de programar la bomba según su conveniencia sin necesidad de tener experiencia previa, ni haberse leído un manual de instrucciones.

La interfaz de usuario está formada por una pantalla LCD 16x2 caracteres, donde se podrán visualizar las diferentes variables establecidas como también navegar por el menú de ajustes. El control se realizará mediante 3 pulsadores, dos de los cuales serán para variar las opciones y el botón central será para aceptar. Se tienen tres diferentes parámetros a modificar: el sentido de giro de la bomba, el diámetro de la jeringa que se utilizará, y el flujo deseado. Como última opción se podrá volver ATRÁS, actualizándose con ello los parámetros de la bomba.

En general, tanto el sentido de giro como el diámetro de la jeringa suelen ser predeterminados por el sistema, pues son los parámetros utilizados habitualmente por el Instituto Universitario de Tecnología Nanofotónica de Valencia. En cualquier caso, y como se ha comentado con anterioridad, se pueden modificar en el menú de ajustes. Una vez iniciado el funcionamiento de la bomba, es posible modificar el flujo en tiempo real mediante los botones BUTTON1 y BUTTON2 para aumentar las decenas o las unidades respectivamente de la velocidad de este. El pulsador START/STOP, como su nombre indica, sirve para inicializar la bomba y pararla cuando se desee. Además, habrá un último botón que, al pulsarlo y según el sentido prefijado en la bomba, realizará la succión o la inyección de fluidos a máxima velocidad. De este modo se podrán rellenar o vaciar rápidamente los canales empleados durante la prueba de sensado, así como retirar posibles burbujas presentes en dicho canal.

Como se ha expuesto en el apartado anterior, el control de la interfaz de usuario estará gobernado por un microcontrolador Arduino Nano, donde hemos programado todas las funciones necesarias para el correcto funcionamiento, cumpliendo los requisitos expuestos al principio de este proyecto.

Imagen 20. Display LCD

3.5 Programa

El programa está diseñado tanto para el control del motor, como la modificación de sus parámetros mediante el menú de ajustes. Se dispondrá de cinco botones diferentes para realizar las tareas que busquemos, y un encoder rotatorio con el que se comprobará el giro del motor. Tres pulsadores serán para el control del menú o ajuste de parámetros. También tendremos un pulsador de START/STOP que encenderá y apagará el motor siguiendo los parámetros definidos, con la excepción de poder aumentar o reducir la velocidad al pulsar alguno de los botones BUTTON1 y BUTTON2 respectivamente. Y como último pulsador tenemos el de vaciado, cuya función es succionar o eyectar el fluido con la dirección estipulada por para la prueba de sensado, pero con un flujo máxima.

La Imagen 21 muestra el diagrama de flujo utilizado para realizar el programa del microcontrolador.

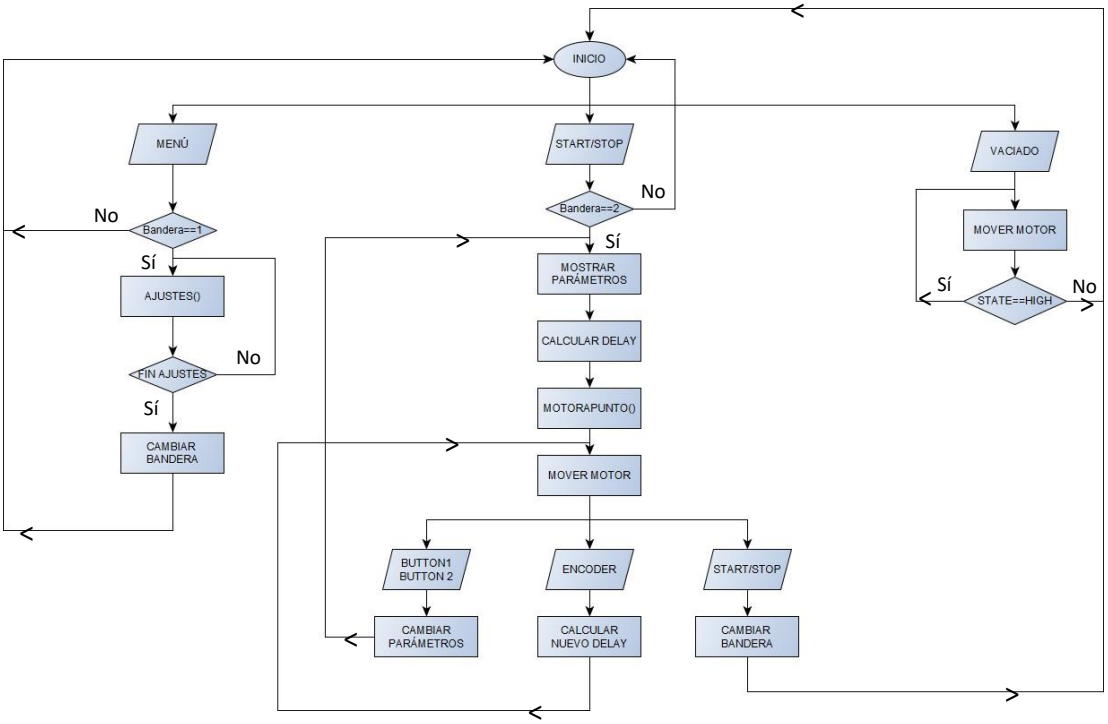


Imagen 21. Diagrama de flujo utilizado para realizar el programa del microcontrolador

Siguiendo ese diagrama de flujo se ha realizado el programa utilizando el entorno de desarrollo integrado (IDE) de Arduino. A continuación, se presentan distintas imágenes capturadas del programa realizado. Primero se ha comenzado declarando la librería utilizada para la pantalla LCD y sus variables.

```
#include <LiquidCrystal.h>
```

Imagen 22. Librería del display LCD

Después se definen los pines de conexión utilizados para el display LCD, los botones, el encoder y el driver del motor formando una estructura que nos facilite el diseño del posterior circuito impreso.

```
//Pines del display LCD
LiquidCrystal lcd(6,7,8,9,10,11);

// Pines de conexión del driver
const int dirPin = 4;
const int stepPin = 5;

// Pin de interrupción del botón de vaciado
const int interruptPin = 3;

// Pin de interrupción del botón del motor
const int interruptMotor=2;

// Pines de conexión del encoder rotatorio
const int CLK=13;
const int DT=12;

// Pines de conexión de BUTTON1, BUTTON2 y MENU USANDO LAS ENTRADAS ANALOGICAS COMO DIGITALES
const int analogPin1 = A1;  // Define el pin analógico A1
const int Button2 = 15; // Número del pin digital correspondiente a A1 en Arduino Uno
const int analogPin2 = A2;  // Define el pin analógico A2
const int MENU = 16; // Número del pin digital correspondiente a A2 en Arduino Uno
const int analogPin3 = A3;  // Define el pin analógico A3
const int Button1 = 17; // Número del pin digital correspondiente a A3 en Arduino Uno
```

Imagen 23. Definición de pines del circuito

Y añadiremos las variables del encoder, de la función menuajustes(), las del cálculo del delay que las usaremos también en la realimentación del motor mediante el encoder en la función motoronoff().

```
// Variables encoder
int enco=0;
int op=1;
int StateCLK;
int lastStateCLK;

// Variables de la función menuajustes()
float diam=14.5;
char sentg[3];
int counter1=1;
int velod=0;
int velou=5;
int sentgir;
int vel=0;

// Variables para el cálculo del delay y la realimentación del motor
unsigned long startMilis; // Variable para almacenar el tiempo de inicio
unsigned long endMilis;   // Variable para almacenar el tiempo de fin
unsigned long elapsedMilis; // Variable para almacenar el tiempo transcurrido
float tteorico;
float treal;
float stepDelay;
float teoricostepdelay;
float velocidad=5;
float diametro=1;
```

Imagen 24. Variables del encoder menuajustes() y las del motoronoff()

Seguidas de la variable de la interrupción de vaciado, las variables de las banderas y el último estado de los pulsadores BUTTON1, BUTTON2, MENU Y MOTOR respectivamente. Las banderas son empleadas para cambiar la función. El programa tiene tres posibles valores de bandera, que son: 0, 1, y 2. Estos valores dan paso a las principales funciones del programa que son Loop(), menuajustes() y motoronoff().

```
// Variable para el estado
volatile bool state = LOW;

// Variables de las banderas
int bandera=0;
int bandera1=1;

// Variables para los botones BUTTON1, MENU, BUTTON2 y Motor
unsigned long lastButtonPress1 = 0;
unsigned long lastButtonPress2 = 0;
unsigned long lastButtonPress3 = 0;
unsigned long lastButtonPress4 = 0;
```

Imagen 25. Variables de VACIADO, de las banderas y de los pulsadores utilizados.

A continuación, definiremos los pines en la función setup() del programa donde definiremos los pines como entradas y salidas, inicializaremos variables como el vector sentg y calcularemos la velocidad inicial predeterminada mostrada en las anteriores imágenes.

```
void setup() {

   lcd.begin(16,2);
   Serial.begin(9600); // Comunicación monitor serie
   sentg[0] = '<';
   sentg[1] = '-';
   sentg[2] = '\0';
   pinMode(dirPin, OUTPUT);
   pinMode(stepPin, OUTPUT);
   pinMode(Button1, INPUT);
   pinMode(Button2, INPUT);
   pinMode(MENU, INPUT);
   pinMode(CLK,INPUT);
   pinMode(DT,INPUT);
   digitalWrite(dirPin, LOW);
   pinMode(interruptPin, INPUT);
   attachInterrupt(digitalPinToInterrupt(interruptPin), interruptHandler, RISING);
   attachInterrupt(digitalPinToInterrupt(interruptMotor), interruptHandlermotor, RISING);
   velocidad=velod+velou;
}
```

Imagen 26. Función setup()

La siguiente función de nuestro programa es la función loop(), donde mostraremos los parámetros en nuestro display LCD. También si la bandera es igual a 1 se llama a la función menuajustes(), en cambio, si la bandera es

igual a 2 se llama a la función motoronoff(). El primer caso ocurre cuando detecta el pulsador MENU en HIGH y después de un delay para evitar el rebote cambia el valor de la bandera de 0 a 1 como se muestra en la siguiente imagen.

```
void loop() {
    // Igualamos nuestra variable int vel a la float velocidad
    // para mostrarla correctamente en la pantalla lcd
    vel=velocidad;
    //Mostramos los parámetros en el display LCD
    lcd.setCursor(0,0);
    lcd.print("                ");
    lcd.setCursor(0,0);
    lcd.print("DIAM: ");
    lcd.setCursor(5,0);
    lcd.print(diam);
    lcd.setCursor(10,0);
    lcd.print("mm OFF");
    lcd.setCursor(0,1);
    lcd.print("                ");
    lcd.setCursor(0,1);
    lcd.print(sentg);
    lcd.setCursor(2,1);
    lcd.print("FLOW:");
    lcd.setCursor(7,1);
    lcd.print(vel);
    lcd.setCursor(10,1);
    lcd.print("uL/min");

    if(digitalRead(MENU)==HIGH){
     delay(500);
     bandera=1;
    }
    if(bandera==1){
    delay(500);
    menuajustes();
    }
    if(bandera==2){
     motoronoff();
    }

}
```

Imagen 27. Función loop()

Para cambiar el estado de la bandera de 0 a 2 debemos llamar a la interrupción del pulsador interruptMotor que irá sumando cada pulso a una variable bandera1, y termina con una condición donde establece que si la variable bandera1 es par cambia el valor de la bandera de 0 a 2, en cambio si la variable bandera1 es impar el valor de la bandera vuelve a tener el valor 0 inicial. El motivo del uso de una interrupción para realizar un contador es

debido a la necesidad de poder parar el motor al instante si se encuentra en marcha y necesitamos pararlo de urgencia, sin embargo, si el programa se encuentra ajustando los parámetros en la función menuajustes() no debe empezar la función del motoronoff() debido a que debemos terminar ajustarlos correctamente y salir de la función, donde si se ha pulsado por error el pulsador interruptMotor no comenzará la función motoronoff() hasta que haya completado y/o salido de la función anterior.

```
void interruptHandlermotor() {

 int btnState4 = digitalRead(interruptMotor);
  if (btnState4 == HIGH) {
    if (millis() - lastButtonPress4 >= 200) {
      banderal++;
      Serial.println("BANDERA1=");
      Serial.print(banderal);
    }
    lastButtonPress4 = millis();
  }
  delay(10);
  if(banderal % 2 == 0)
    bandera=2;
  else bandera=0;
}
```

Imagen 28. Interrupción del pulsador interruptMotor.

La función menuajustes() sigue el diagrama de flujo de la Imagen 29.

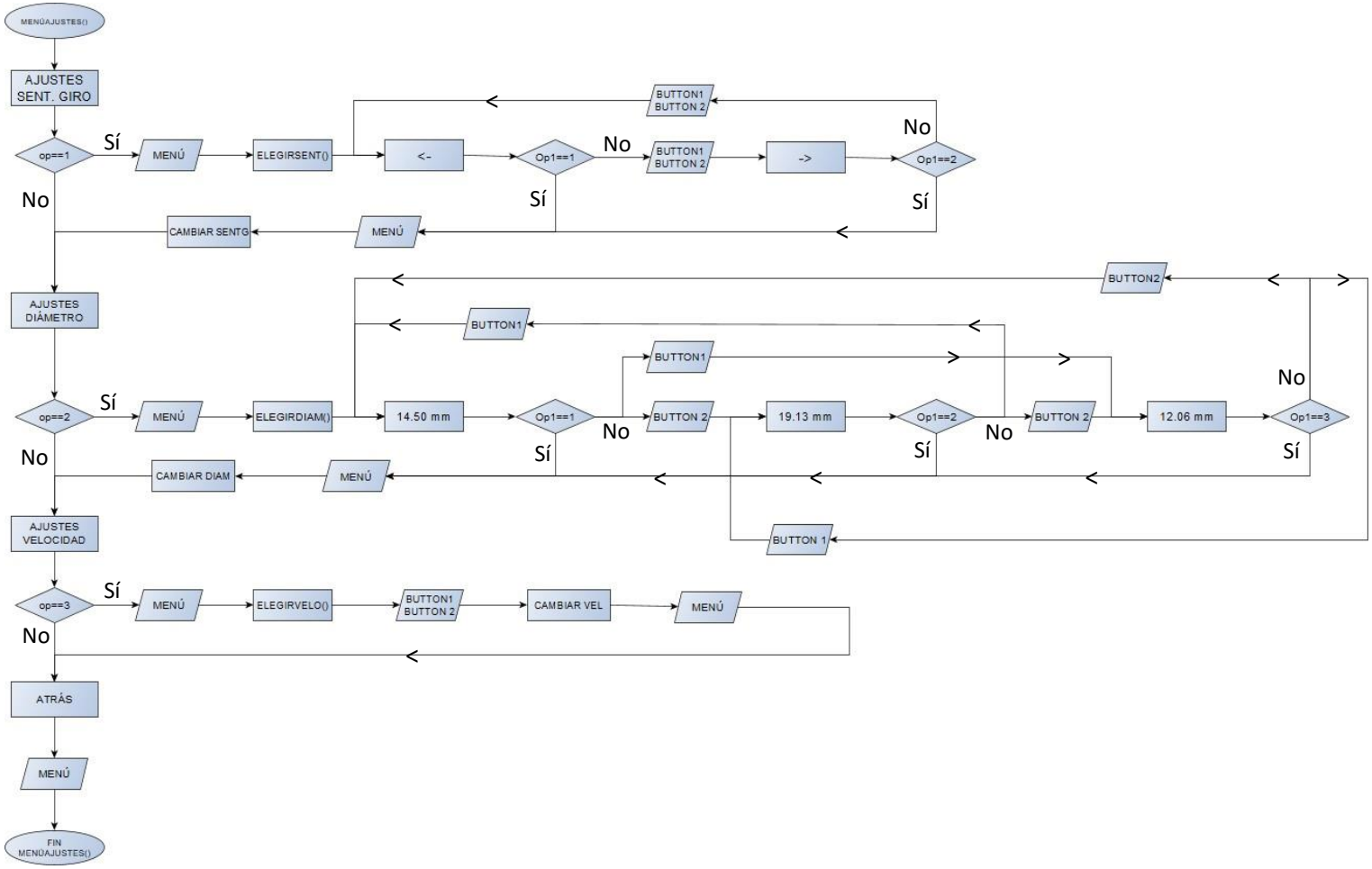


Imagen 29: Diagrama de flujo de la función menuajustes().

Donde comenzamos con un bucle do while, que perdurará mientras el valor de la variable bandera sea igual a 1. Dentro de este bucle llamamos a las funciones contador1 y contador2 para cambiar la opción de ajuste.

```
int contador1(int counter){
  int btnState = digitalRead(Button1);
  if (btnState == HIGH) {
    if(millis() - lastButtonPress1 >= 80) {
      counter--;
      }
    lastButtonPress1 = millis();
  }
  delay(1);
  return counter;
}
```

Imagen 30. Función contador1().

```
int contador2(int counter){
  int btnState = digitalRead(Button2);
  if (btnState == HIGH) {
   if (millis() - lastButtonPress2 >= 80) {
     counter++;
    }
  lastButtonPress2 = millis();
  }
  delay(1);
  return counter;
}
```

Imagen 31. Función contador2().

```
void menuajustes(){
  do{
  if(digitalRead(Button1)==HIGH){
    op=contador1(op);
    if(op<1)
    op=4;
    Serial.print("Opción:");
    Serial.println(op);
  }
   if(digitalRead(Button2)==HIGH){
    op=contador2(op);
    if(op>4)
    op=1;
    Serial.print("Opción:");
    Serial.println(op);
  }
```

Imagen 32. Función menuajustes() I

Después empleamos un switch case para elegir que parámetro queremos cambiar en su defecto si queremos salir, modificando la variable op mediante los pulsadores BUTTON1 y BUTTON2 que llaman a las funciones contador1 y contador2 de la imagen 25 y 26 respectivamente.

```
switch(op){
  case 1:
    lcd.setCursor(0,0);
    lcd.print("                ");
    lcd.setCursor(4,0);
    lcd.print("AJUSTES");
    lcd.setCursor(0,1);
    lcd.print("                ");
    lcd.setCursor(2,1);
    lcd.print("SENT. GIRO");
    if(digitalRead(MENU)==HIGH){
      delay(500);
      elegirsent();
    }
  break;

  case 2:
    lcd.setCursor(0,0);
    lcd.print("                ");
    lcd.setCursor(4,0);
    lcd.print("AJUSTES");
    lcd.setCursor(0,1);
    lcd.print("                ");
    lcd.setCursor(3,1);
    lcd.print("DIAMETRO");
    if(digitalRead(MENU)==HIGH){
      delay(500);
      elegirdiam();
    }
  break;
```

```
    case 3:
      lcd.setCursor(0,0);
      lcd.print("                ");
      lcd.setCursor(4,0);
      lcd.print("AJUSTES");
      lcd.setCursor(0,1);
      lcd.print("                ");
      lcd.setCursor(4,1);
      lcd.print("FLUJO");
      if(digitalRead(MENU)==HIGH){
        delay(500);
        elegirvelo();
      }
    break;

    case 4:
      lcd.setCursor(0,0);
      lcd.print("                ");
      lcd.setCursor(4,0);
      lcd.print("AJUSTES");
      lcd.setCursor(0,1);
      lcd.print("                ");
      lcd.setCursor(3,1);
      lcd.print("ATRAS");
      if(digitalRead(MENU)==HIGH){
        bandera=0;
        delay(500);
        op=1;
      }
    break;
   }
  } while(bandera==1);

}
```

Imagen 33. Función menuajustes() II

En los tres primeros valores de la variable op se llaman a funciones externas encargadas de realizar la elección del parámetro seleccionado si el pulsador MENU se detecta en un estado alto. En cambio, si se presiona el pulsador MENU en la cuarta opción, el valor de la bandera cambia al 0 y volvemos a la función loop() con los valores modificados.

En el primer caso se llamaría a la función elegirsent(), donde podremos elegir si queremos que la bomba succione el líquido o en su defecto que lo eyecte.

```
void elegirsent(){
 int opl=1;
  do{
  if(digitalRead(Button1)==HIGH){
    opl=contador1(opl);
    if(opl<1)
    opl=2;
    Serial.print("Opción:");
    Serial.println(opl);
  }
   if(digitalRead(Button2)==HIGH){
    opl=contador2(opl);
    if(opl>2)
    opl=1;
    Serial.print("Opción:");
    Serial.println(opl);
  }
```

Imagen 34. Función elegirsent() I

```
switch(opl){
  case 1:
       lcd.setCursor(0,0);
       lcd.print("                ");
       lcd.setCursor(2,0);
       lcd.print("SENT. GIRO");
       lcd.setCursor(0,1);
       lcd.print("                ");
       sentg[0] = '<';
       sentg[1] = '-';
       sentg[2] = '\0';
       lcd.setCursor(6,1);
       lcd.print(sentg);
       if(digitalRead(MENU)==HIGH){
            sentg[0] = '<';
            sentg[1] = '-';
            sentg[2] = '\0';
            digitalWrite(dirPin, LOW);
            op=2;
           }
   break;
   case 2:
       lcd.setCursor(0,0);
       lcd.print("                ");
       lcd.setCursor(2,0);
       lcd.print("SENT. GIRO");
       lcd.setCursor(0,1);
       lcd.print("                ");
       sentg[0] = '-';
       sentg[1] = '>';
       sentg[2] = '\0';
       lcd.setCursor(6,1);
       lcd.print(sentg);
       if(digitalRead(MENU)==HIGH){
            sentg[0] = '-';
            sentg[1] = '>';
            sentg[2] = '\0';
            digitalWrite(dirPin, HIGH);
            op=2;
           }
  break;
 }
 } while(digitalRead(MENU)==LOW);
 opl=1;
 delay(500);
}
```

Imagen 35. Función elegirsent() II

En el segundo caso, se llamaría a la función elegirdiam() donde se elegirá entre los tres diámetros que nos ofrecen las jeringas de 10 mL, 20 mL, y 5 mL respectivamente, comenzando por la jeringa de 10 mL porque es más frecuente su uso. También modifica la variable diámetro que nos ayudará en la función motoronoff() para el correcto cálculo del delay que varía de una forma directamente proporcional en función de su sección.

```
void elegirdiam(){
  int op1=1;
  do{
    if(digitalRead(Button1)==HIGH){
    op1=contador1(op1);
    if(op1<1)
    op1=3;
    Serial.print("Opción:");
    Serial.println(op1);
  }
   if(digitalRead(Button2)==HIGH){
    op1=contador2(op1);
    if(op1>3)
    op1=1;
    Serial.print("Opción:");
    Serial.println(op1);
  }
   switch(op1){
    case 1:
        lcd.setCursor(0,0);
        lcd.print("                ");
        lcd.setCursor(3,0);
        lcd.print("Diametro");
        lcd.setCursor(0,1);
        lcd.print("                ");
        lcd.setCursor(4,1);
        lcd.print("14.50");
        if(digitalRead(MENU)==HIGH){
              diametro=1; //variable para hacer el cáculo de velocidades
              diam=14.50;
              Serial.println("Diametro=14.50");
              Serial.print(diametro);
              op=3;
            }
    break;
```

Imagen 36. Función elegirdiam()

```
    case 2:
        lcd.setCursor(0,0);
        lcd.print("                ");
        lcd.setCursor(3,0);
        lcd.print("Diametro");
        lcd.setCursor(0,1);
        lcd.print("                ");
        lcd.setCursor(4,1);
        lcd.print("19.13");
        if(digitalRead(MENU)==HIGH){
              diametro=pow(19.13/2,2)/pow(14.5/2,2); //variable para hacer el cáculo de velocidades
              diam=19.13;
              op=3;
              Serial.println("Diametro=12.06");
              Serial.print(diametro);
            }
    break;
    case 3:
        lcd.setCursor(0,0);
        lcd.print("                ");
        lcd.setCursor(3,0);
        lcd.print("Diametro");
        lcd.setCursor(0,1);
        lcd.print("                ");
        lcd.setCursor(4,1);
        lcd.print("12.06");
        if(digitalRead(MENU)==HIGH){
              diametro=pow(12.06/2,2)/pow(14.5/2,2); //variable para hacer el cáculo de velocidades
              diam=12.06;
              op=3;
              Serial.println("Diametro=12.06");
              Serial.print(diametro);
            }
    break;
   }
  } while(digitalRead(MENU)==LOW);
  op1=0;
  delay(500);
}
```

Imagen 37. Función elegirdiam()

Al presionar MENU en la tercera opción llamaría a la función elegirvelo(), donde al detectar el alto el BUTTON1 o el BUTTON2 llama a las funciones contador1d() o contador2() previamente mostrada en la Imagen 31. Añadimos una nueva función de contador en la función contador1d(), porque a diferencia de la función contador1() cada vez que detecte un pulso HIGH del BUTTON1 se sumarán valores de la variable velod, que representa el número de decenas, obteniendo así, un pulsador para ir sumando decenas y otro para sumar unidades, teniendo un límite inferior de 5 uL/min y un límite superior de 120 uL/min.

```
int contador1d(int counter){
   int botonState = digitalRead(Button1);
  if (botonState == HIGH) {
    if(millis() - lastButtonPress1 >= 80) {
      counter++;
      }
    lastButtonPress1 = millis();
  }
delay(1);
return counter;
}
```

Imagen 38. Función contador1d()

```
void elegirvelo(){
  do{

   if(digitalRead(Button1)==HIGH){
      velod=contador1d(velod);
      if(velod>12)
        velod=0;
   }
   if(digitalRead(Button2)==HIGH){
      velou=contador2u(velou);
      if(velou>9)
        velou=0;
   }
   velocidad=velod*10+velou;
      if(velocidad<5){
        velocidad=5;
      velou=5;
   }
   if(velocidad>120){
      velocidad=120;
      velou=0;
      velod=12;
   }
   vel=velocidad;

   lcd.setCursor(0,0);
   lcd.print("                ");
   lcd.setCursor(4,0);
   lcd.print("FLUJO");
   lcd.setCursor(0,1);
   lcd.print("                ");
   lcd.setCursor(4,1);
   lcd.print(vel);

  }while(digitalRead(MENU)==LOW);
   delay(500);
   op=4;
}
```

Imagen 39. Función elegirvelo()

Cuando el valor de la variable bandera es 2 llama a la función motoronoff(). Ésta comienza mostrando los parámetros en el display LCD, y calculando el retardo en función de la velocidad de flujo determinada por el usuario. Y antes de pasar al bucle “do while”, llama a una función denominada motorapunto() donde finalmente después de completarla tomaríamos un tiempo determinado por la variable startMilis que se empleará para el cálculo del nuevo retardo.

```
void motoronoff(){

    lcd.setCursor(0,0);
    lcd.print("                ");
    lcd.setCursor(0,0);
    lcd.print("DIAM: ");
    lcd.setCursor(5,0);
    lcd.print(diam);
    lcd.setCursor(10,0);
    lcd.print("mm ON");
    lcd.setCursor(0,1);
    lcd.print("                ");
    lcd.setCursor(0,1);
    lcd.print(sentg);
    lcd.setCursor(2,1);
    lcd.print("FLOW:");
    lcd.setCursor(7,1);
    lcd.print(vel);
    lcd.setCursor(10,1);
    lcd.print("uL/min");
    stepDelay =(1/velocidad)*20.98/(1/63.9)*diametro;
    teoricostepdelay=stepDelay;
    tteorico=stepDelay*80*2*2;
    motorapunto(enco);
    startMilis=millis();
```

Imagen 40. Función motoronoff() I

Esta función fue añadida al programa debido a la diferencia de pulsos por vuelta del encoder y del motor, que desembocaba en un error añadido. Y es que 80 pasos del motor ese corresponden con un paso del encoder. Si la bomba se detiene en una posición intermedia del encoder, su primer paso requerirá de menos pasos del motor y por consiguiente de un tiempo menor al estimado. Por ese motivo, fue necesaria la implementación de una función de puesta a punto que ordena realizar un bucle “do while” hasta que el valor de la variable enco pasa de 0 a 1. Después cambia el valor de la variable a 0 como se puede observar en el diagrama de flujo de la siguiente imagen. Como resultado, elimina el error del primer paso del encoder.

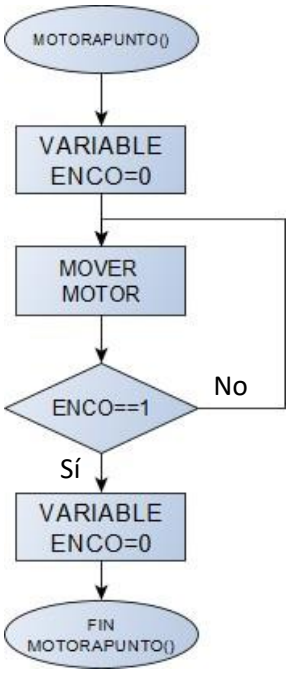


Imagen 41. Diagrama de flujo de la función motor a punto.

Lo primero que se realiza es incluir dos funciones if para variar la velocidad, cambiar el retardo entre los pulsos del motor, y los parámetros mostrados en el display si se detecta un valor HIGH en el BUTTON1 o BUTTON2.

```
int motorapunto(int encoder){
  encoder=0;

  do{
     if(digitalRead(Button1)==HIGH){
        velod=contadorId(velod);
        if(velod>12)
        velod=0;
        velocidad=velod*10+velou;
        if(velocidad<5)
        velocidad=5;
        if(velocidad>120){
        velocidad=120;
        velou=0;
        }
        vel=velocidad;
        lcd.setCursor(0,0);
        lcd.print("                ");
        lcd.setCursor(0,0);
        lcd.print("DIAM: ");
        lcd.setCursor(5,0);
        lcd.print(diam);
        lcd.setCursor(10,0);
        lcd.print("mm ON");
        lcd.setCursor(0,1);
        lcd.print("                ");
        lcd.setCursor(0,1);
        lcd.print(sentg);
        lcd.setCursor(2,1);
        lcd.print("FLOW:");
        lcd.setCursor(7,1);
        lcd.print(vel);
        lcd.setCursor(10,1);
        lcd.print("uL/min");
        stepDelay =(1/velocidad)*20.98/(1/63.9)*diametro;
        teoricostepdelay=stepDelay;
        tteorico=stepDelay*80*2*2;

     }
```

Imagen 42. Función motorapunto() I

```
if(digitalRead(Button2)==HIGH){
   velou=contador2u(velou);
   if(velou>9)
   velou=0;
   velocidad=velod*10+velou;
   if(velocidad<5)
   velocidad=5;
   if(velocidad>120){
   velocidad=120;
   velod=12;
   velou=0;
   }
   vel=velocidad;
   lcd.setCursor(0,0);
   lcd.print("                ");
   lcd.setCursor(0,0);
   lcd.print("DIAM: ");
   lcd.setCursor(5,0);
   lcd.print(diam);
   lcd.setCursor(10,0);
   lcd.print("mm ON");
   lcd.setCursor(0,1);
   lcd.print("                ");
   lcd.setCursor(0,1);
   lcd.print(sentg);
   lcd.setCursor(2,1);
   lcd.print("FLOW:");
   lcd.setCursor(7,1);
   lcd.print(vel);
   lcd.setCursor(10,1);
   lcd.print("uL/min");
   stepDelay =(1/velocidad)*20.98/(1/63.9)*diametro;
   teoricostepdelay=stepDelay;
   tteorico=stepDelay*80*2*2;

}
```

Imagen 43. Función motorapunto() II

Estas condiciones if van seguidas de las instrucciones para mover el motor de manera que van a ejecutarse hasta que el encoder cambie su valor de 0 a 1. El bucle se realizará siempre que el encoder sea igual a 0 y la bandera1 sea un valor par. Como ya se ha comentado previamente esta última condición irá en función de la interrupción del pin interruptMotor que podrá parar el funcionamiento y cambiar el valor de la bandera de 2 a 0. En el caso de que la bandera1 no haya cambiado y la variable encoder haya variado de 0 a 1, el bucle terminará, la variable encoder se modificará al valor 0 y la función motorapunto() terminará y volverá a la función motoronoff().

```
        lastStateCLK = digitalRead(CLK);
        digitalWrite(stepPin, HIGH);
        delay(stepDelay);
        digitalWrite(stepPin, LOW);
        delay(stepDelay);
        StateCLK = digitalRead(CLK);

if (StateCLK != lastStateCLK  && StateCLK == 1){

   if (digitalRead(DT) != StateCLK) {
      enco --;

    } else {
      enco ++;
    }
}
  lastStateCLK = StateCLK;
       }
       while(enco==0 && bandera1%2==0);

       enco=0;
       return encoder;
}
```

Imagen 44. Función motorapunto() III

Al retornar a la función motoronoff() se utiliza dos if para poder variar la velocidad flujo. Como en la función elegirvelo() se utiliza las funciones contador1d() y contador2() para variar las decenas y las unidades respectivamente de la velocidad de flujo, como también calculando y mostrando los nuevos parámetros. Después tiene las instrucciones de mover el motor seguidas de una función para contar los pulsos del encoder y en el momento que haya dado 2 pulsos, se le daría el valor de tiempo de la función millis() a la variable endMilis donde mediante una resta se obtendrá el tiempo transcurrido durante los 2 primeros pulsos y se podrá realizar un cálculo del nuevo retardo que debemos programar para su correcto funcionamiento, y cambiando el valor de la variable enco a 0 para poder repetir el proceso anterior cambiando nuevamente el delay en función de los tiempos obtenidos. Este bucle terminará cuando se llame a la interrupción del pin interruptMotor y cambie el valor de la variable bandera1 a un valor impar y el valor de la variable bandera a 0.

```
do{


 if(digitalRead(Button1)==HIGH){
        velod=contadorId(velod);
        if(velod>12)
        velod=0;
        velocidad=velod*10+velou;
        if(velocidad<5)
        velocidad=5;
        if(velocidad>120){
        velocidad=120;
        velod=12;
        velou=0;
        }
        vel=velocidad;
        lcd.setCursor(0,0);
        lcd.print("                ");
        lcd.setCursor(0,0);
        lcd.print("DIAM: ");
        lcd.setCursor(5,0);
        lcd.print(diam);
        lcd.setCursor(10,0);
        lcd.print("mm ON");
        lcd.setCursor(0,1);
        lcd.print("                ");
        lcd.setCursor(0,1);
        lcd.print(sentg);
        lcd.setCursor(2,1);
        lcd.print("FLOW:");
        lcd.setCursor(7,1);
        lcd.print(vel);
        lcd.setCursor(10,1);
        lcd.print("uL/min");
        stepDelay =(1/velocidad)*20.98/(1/63.9)*diametro;
        teoricostepdelay=stepDelay;
        tteorico=stepDelay*80*2*2;
        motorapunto(enco);
        startMilis=millis();
    }
```

Imagen 45. Función motoronoff() II

```
if(digitalRead(Button2)==HIGH){
   velou=contador2u(velou);
   if(velou>9)
   velou=0;
   velocidad=velod*10+velou;
   if(velocidad<5)
   velocidad=5;
   if(velocidad>120){
   velocidad=120;
   velod=12;
   velou=0;
   }
   vel=velocidad;
   lcd.setCursor(0,0);
   lcd.print("                ");
   lcd.setCursor(0,0);
   lcd.print("DIAM: ");
   lcd.setCursor(5,0);
   lcd.print(diam);
   lcd.setCursor(10,0);
   lcd.print("mm ON");
   lcd.setCursor(0,1);
   lcd.print("                ");
   lcd.setCursor(0,1);
   lcd.print(sentg);
   lcd.setCursor(2,1);
   lcd.print("FLOW:");
   lcd.setCursor(7,1);
   lcd.print(vel);
   lcd.setCursor(10,1);
   lcd.print("uL/min");
   stepDelay =(1/velocidad)*20.98/(1/63.9)*diametro;
   teoricostepdelay=stepDelay;
   tteorico=stepDelay*80*2*2;
   motorapunto(enco);
   startMilis=millis();
}
```

Imagen 46. Función motoronoff() III

```
    digitalWrite(stepPin, HIGH);
    delay(stepDelay);
    digitalWrite(stepPin, LOW);
    delay(stepDelay);
    StateCLK = digitalRead(CLK);
if (StateCLK != lastStateCLK  && StateCLK == 1){
    if (digitalRead(DT) != StateCLK) {
      enco --;
    } else {
      enco ++;
      if(enco==2){
      endMilis=millis();
      elapsedMilis= endMilis-startMilis;
      stepDelay=(tteorico/elapsedMilis)*teoricostepdelay*diametro;
      enco=0;
      startMilis=millis();
      }
    }
  }
  lastStateCLK = StateCLK;
}
while(banderal % 2 == 0);
}
```

Imagen 47. Función motoronoff() IV

Finalmente, el código posee la interrupción asociada al pin interruptPin, donde al detectar un valor HIGH en el pin, moverá el motor a una alta velocidad de flujo mientras esté en valor alto. En el momento que este valor pase a ser LOW se mostrarán los parámetros antiguos en el display LCD.

```
void interruptHandler() {

    lcd.setCursor(0,0);
    lcd.print("                ");
    lcd.setCursor(4,0);
    lcd.print("VACIADO");
    lcd.setCursor(0,1);
    lcd.print("                ");
   int stepDelay2 = 250;

  do {
     state = digitalRead(interruptPin);
     digitalWrite(stepPin, HIGH);
     delayMicroseconds(stepDelay2);
     digitalWrite(stepPin, LOW);
     delayMicroseconds(stepDelay2);
  }
  while(state==HIGH);
        lcd.setCursor(0,0);
        lcd.print("                ");
        lcd.setCursor(0,0);
        lcd.print("DIAM: ");
        lcd.setCursor(5,0);
        lcd.print(diam);
        lcd.setCursor(10,0);
        lcd.print("mm ON");
        lcd.setCursor(0,1);
        lcd.print("                ");
        lcd.setCursor(0,1);
        lcd.print(sentg);
        lcd.setCursor(3,1);
        lcd.print("FLOW:");
        lcd.setCursor(7,1);
        lcd.print(vel);
        lcd.setCursor(10,1);
        lcd.print("uL/min");
        motorapunto(enco);

}
```

Imagen 48. Función asociada a la interrupción relativa al vaciado

Capítulo 4. Calibración de la bomba de jeringa

Tras la realización de un prototipo del circuito y la programación del microcontrolador, se ha procedido a calibrar la bomba. Para ello se ha medido el tiempo que tardaba en succionar distintos volúmenes de agua desionizada, empleando también varios retardos para los pulsos del motor. La Imagen 49 muestra una fotografía realizada durante el paso de calibración.

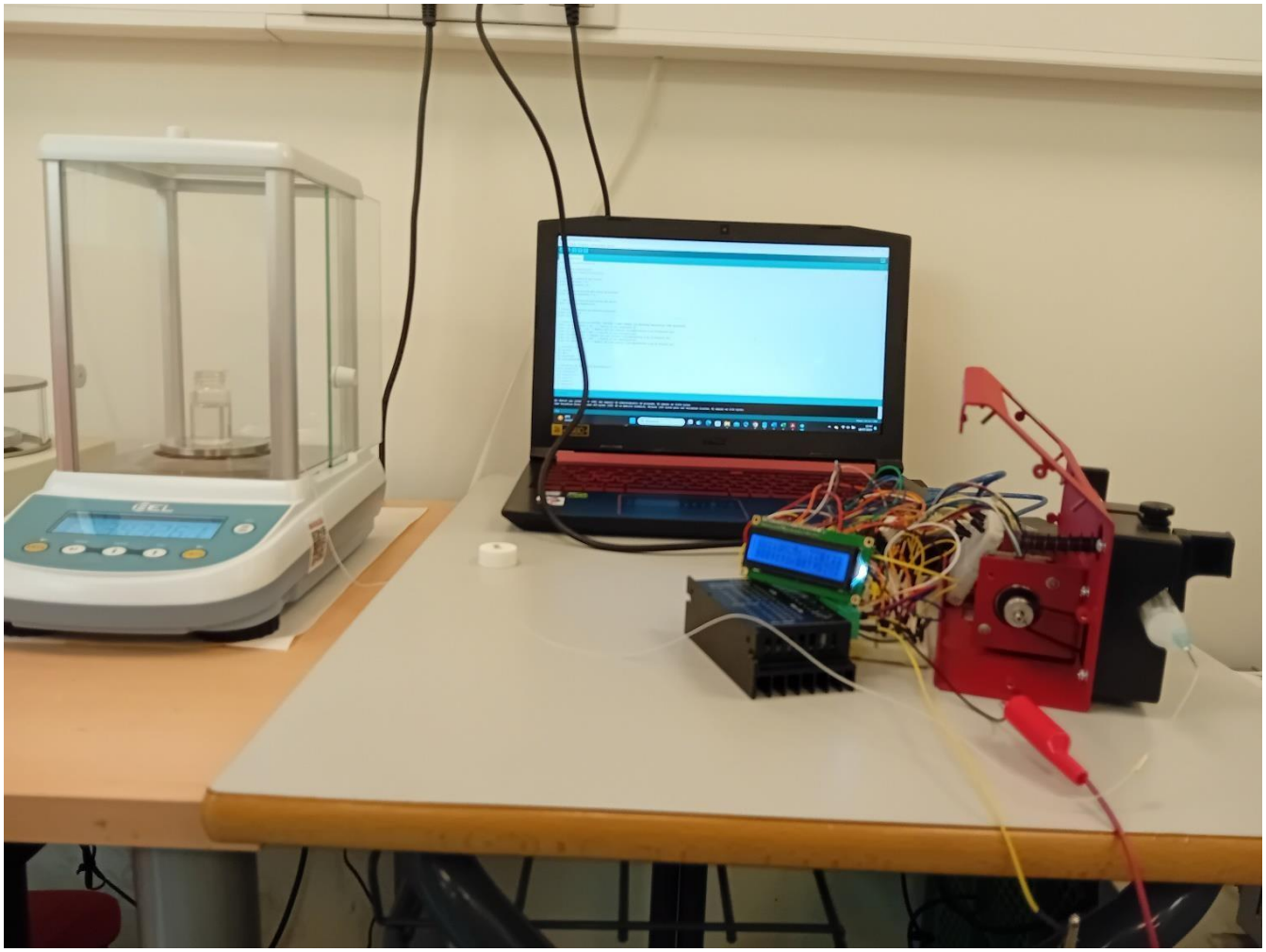


Imagen 49. Proceso de calibración.

Primero, realizamos la preparación inicial basada en los siguientes pasos:

- Llenado de un recipiente con volumen de agua desionizada suficiente para realizar las mediciones. Dicho recipiente fue situado en una báscula de precisión.
- Calibración inicial y tarado de la báscula de precisión, tras situar el recipiente con agua desionizada.
- Después se coloca la jeringa bien fijada en la bomba, y se comprueban las conexiones presentes entre el circuito electrónico, el motor y el encoder de la bomba. La jeringa posee un canal de silicona, que es introducido en el recipiente de agua desionizada.

- A continuación, se utilizó la función de succión a máxima velocidad previamente comentada, hasta que todo el tubo estuvo completamente lleno de agua. Para ello nos aseguramos de que dicha agua había llegado a la jeringa. Entonces se anotó la masa inicial del recipiente con agua comentado con anterioridad.
- Finalmente, se succionó un volumen de líquido determinado, anotando el tiempo necesario para ello según el retardo utilizado en los pasos del motor de la bomba. Dichos volúmenes fueron de 2, 4 y 6 mL.

Dicho proceso fue repetido 4 veces para cada retardo. En nuestro caso los retardos escogidos fueron 6.25, 12.5 y 25 ms por paso. En el proceso de calibración se ha utilizado una jeringa de 10 mL y 14.5 mm de diámetro, debido a que suele ser la más utilizada en los pasos de sensado.

Después de tomar todas las medidas nos ayudaremos de la herramienta Excel para obtener una relación entre los retardos aplicados y los tiempos que ha tardado la bomba en llegar a los volúmenes deseados. La gráfica presente en la Imagen 50 muestra que hay una relación lineal entre el volumen succionado y el tiempo necesario para ello. Las pendientes obtenidas son distintas, según los retardos utilizados para los pasos del motor de la bomba.

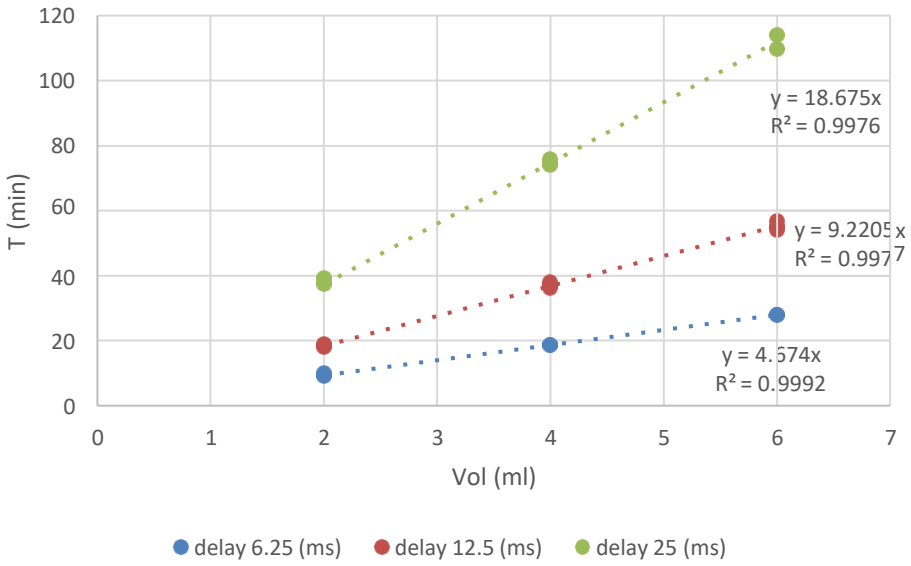


Imagen 50. Gráfica de los tiempos de cada retardo para llegar a cada volumen.

A partir de los datos anteriores, y de acuerdo con la gráfica presente en la imagen 51, podemos obtener la relación entre el flujo y el retardo a programar. Ésta sigue una tendencia logarítmica multiplicado por un factor negativo. Con ello, si el flujo tiende a cero, el retardo a programar tenderá a infinito.

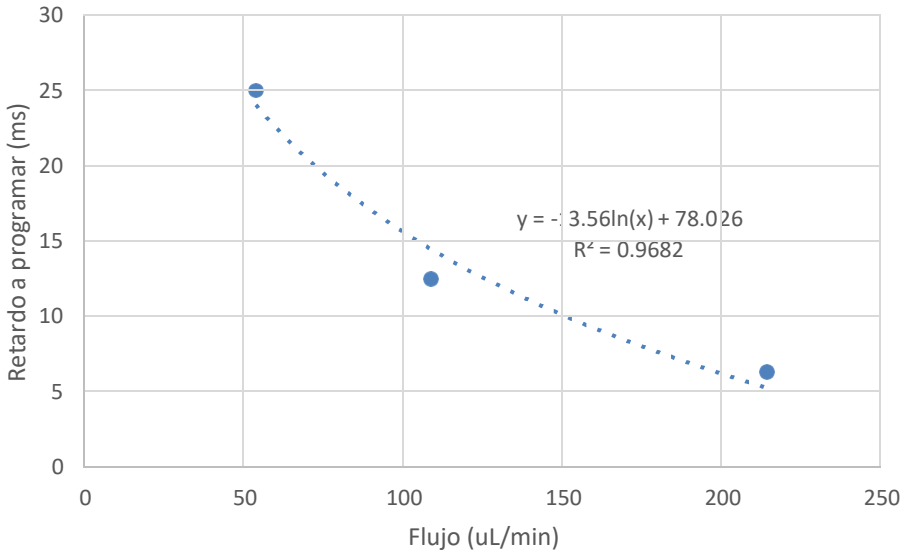


Imagen 51. Gráfica del flujo en función del retardo.

La expresión obtenida a partir de la imagen 51son las pruebas de calibración obtenidas, y es la que se ha programado en la bomba para fijar el retardo necesario para conseguir el flujo requerido.

Capítulo 5. Conclusiones

En este trabajo de fin de grado, se ha explicado el proceso elaborado para la reutilización de una bomba de jeringa destinada a pruebas de sensado. Para ello se ha diseñado el circuito electrónico y el software de control de una bomba, aprovechando una matriz de una bomba desechada. De este modo se ha logrado reducir los costes de reparación y los tiempos de espera. Además, se ha utilizado un software de libre uso, para poder realizar modificaciones sobre el programa si fuese necesario. La bomba ha sido rediseñada atendiendo a los requerimientos presentes en los laboratorios del Instituto Universitario de Tecnología Nanofotónica de Valencia, en aplicaciones de sensado en tiempo real. Dichos requerimientos pasan por la elección del diámetro de la jeringa, el flujo entre 5 y 120 uL/min, y el sentido (succión o inyección). Además, la bomba ha de permitir la modificación del flujo en tiempo real, así como realizar una succión o inyección a máxima velocidad en un momento en el que pueda ser necesario.

Los componentes elegidos han sido seleccionados y justificados cuidadosamente para satisfacer los requerimientos iniciales, y ajustarse correctamente a las dimensiones de la matriz de la bomba.

Para conseguir el control adecuado de la velocidad de flujo, se dispone de un encoder rotatorio en la matriz, compartiendo un eje con el motor. Esta información es recibida por el microcontrolador, quien modifica el retardo de los pasos del motor para lograr el flujo de acuerdo con los parámetros seleccionados.

La interfaz de usuario se ha implementado mediante un display LCD, y 5 pulsadores emulando a la bomba original que contiene los mismos componentes. Permite ajustar los parámetros de una manera sencilla e intuitiva, además de poder visualizar todos los parámetros o funciones que se están realizando en el display LCD y en tiempo real.

Una vez diseñado el circuito electrónico, se ha procedido a realizar un circuito impreso (presente en los planos). Éste proporciona una mayor fiabilidad e imagen que el circuito electrónico montado en una placa de prototipos.

Se realizaron diferentes pruebas durante el diseño de la bomba, mediante el uso del monitor serie de Arduino. Éste permitía comunicarse en tiempo real con el microcontrolador, el cual daba información sobre los cambios de las variables. De este modo nos asegurábamos del estado del motor y del resto de componentes.

Una vez terminado el diseño y realización del circuito electrónico, se procedió a calibrar la bomba, obteniéndose una relación entre el diámetro de la jeringa, el flujo deseado y el retardo necesario para dar cada paso del motor.

A partir de la calibración, se calcula el tiempo necesario para que el encoder realice 2 pulsos, y se compara con el tiempo experimental. El retardo asociado a los pasos del motor se modificará de acuerdo con la comentada comparación. Se ha escogido una realimentación cada 2 pulsos del encoder, debido a que es suficientemente largo para que el error introducido por el tiempo de ejecución del programa sea mínimo, y suficientemente corto para el error producido en la velocidad de flujo sea bajo. A modo de referencia el retardo se actualizará cada aproximadamente 30 segundos para un flujo de 20 uL/min.

Sin embargo, y si bien ha sido diseñado, de cara a la elaboración del producto final ha faltado la realización del circuito impreso. Sí que ha sido posible su uso debido a que los componentes y el microcontrolador fueron montados en una placa de prototipos. Las primeras pruebas realizadas han sido satisfactorias.

Como aspectos a mejorar cabe recalcar la necesidad de implementar el circuito impreso diseñado en la matriz de la bomba de jeringa. También es necesario mencionar la sustitución del encoder rotativo por otro de mayor ppr.

Finalmente, indicar que el proyecto está en línea con los Objetivos de Desarrollo Sostenible 3, 4, 9 y 13. Destacar el ODS 13, debido a que la reutilización de la bomba de jeringa evita la fabricación de una nueva, relacionándose de forma directamente con la lucha contra el cambio climático. Y sobre todo, destacar el ODS 4, relacionado con una educación de calidad. Y es que han sido muchos conceptos nuevos adquiridos durante la realización de este proyecto.

Bibliografía y referencias

Finalmente, se terminará esta memoria técnica enumerando la biografía empleada para su desarrollo:

Jorge Sesma Jarauta. Diseño y realización de una bomba peristáltica para aplicaciones de sensado en tiempo real. Trabajo de Fin de Grado del año 2023.

José Manuel Mena Rodríguez. Fabricación de placas de circuito impreso con Proteus. Trabajo de Fin de Grado del año 2015.

https://www.graco.com/es/es/in-plant-manufacturing/solutions/articles/benefits-of-electric-diaphragm-pumps-vs-peristaltic-roller-pumps.html Consultado el 10 Jul. 2024

[1] https://www.micruxfluidic.com/es/instrumentacion/sistemas-de-bombeo/serie-ne/ Consultado el día 6 Abr. 2024
[2] https://tecno-products.com/blog/novedades/conduccion/como-funciona-una-bomba-peristaltica/ Consultado el día 6 Abr. 2024
[3] https://areasaludplasencia.es/wasp/pdfs/7/717004.pdf Consultado el día 19 Jun. 2024
[4] S. Ponce-Alcántara *et al*., "Dual Refractive Index and Viscosity Sensing Using Polymeric Nanofibers Optical Structures," in *IEEE Sensors Journal*, vol. 19, no. 24, pp. 11850-11857, 15 Dec.15, 2019
[5] https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ Consultado el día 28 Jun. 2024
[6] https://fervi3d.com/drivers/110-pololu-driver-a4988.html Consultado el día 30 Jun. 2024
[7] https://www.az-delivery.de/es/products/drv8825-schrittmotor-treiber-modul-mit-kuhlkorper Consultado el día 30 Jun. 2024
[8] https://www.amazon.es/UsongShine-TB6600-Controlador-h%C3%ADbrido-impresora/dp/B07HHS14VQ Consultado el día 30 Jun. 2024
[9] https://www.empowerlaptop.com/p/atmega328p/?wmc-currency=EUR&gad_source=1&gclid=CjwKCAjwqf20BhBwEiwAt7dtdbQDcTkSDAQaHCYo1BI2ZCUfGD7PMQ4UXWZ5Wj1AH550rEm8SljjuBoCkRoQAvD_BwE Consultado el día 6 Jul. 2024
[10] https://www.amazon.es/dp/B07GXZGT5Q/ref=twister_B0CX28HN25?_encoding=UTF8&psc=1 Consultado el día 6 Jul. 2024
[11]https://www.sparkfun.com/datasheets/Components/LM7805.pdf Consultado el día 14 Jul. 2024
[12]https://www.ti.com/lit/ds/slvs044y/slvs044y.pdf?ts=1721841938691&ref_url=https%253A%252F%252Fwww.google.com%252F Consultado el día 16 Jul. 2024

UNIVERSIDAD POLITÉCNICA DE VALENCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño y realización de un sistema de inyección – infusión de alta precisión, para aplicaciones de sensado en tiempo real

DOCUMENTO Nº2: PLANOS Y CIRCUITO ELECTRÓNICO

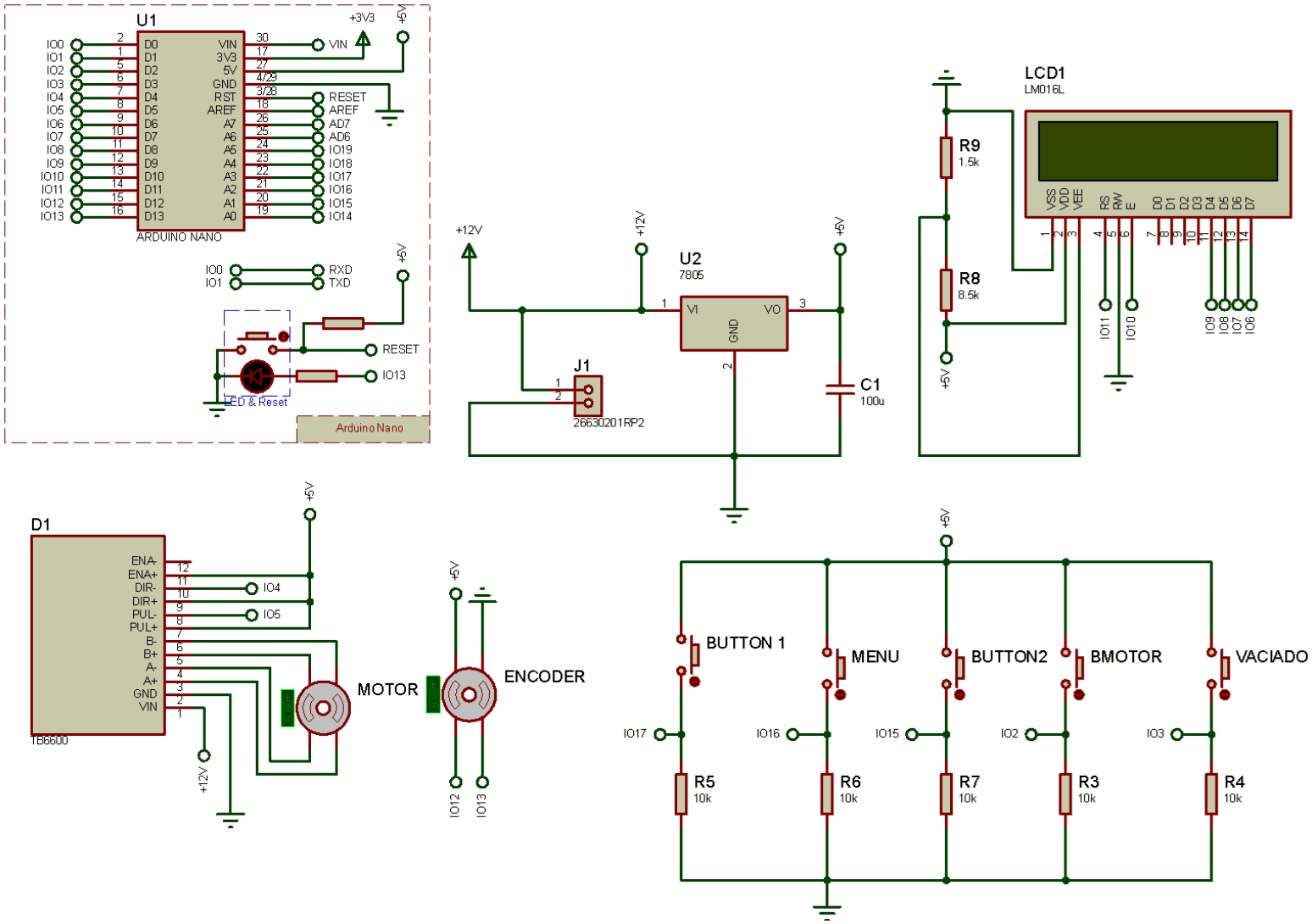
Trabajo final de grado

Grado en Ingeniería Electrónica Industrial y Automática

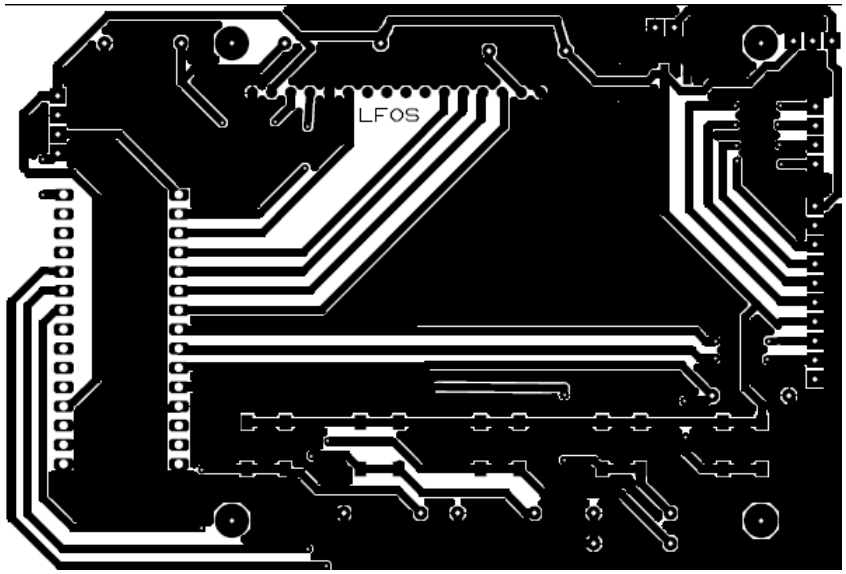
Autor: Olmos Saldías, Luis Fernando

Tutor: Ponce Alcántara, Salvador

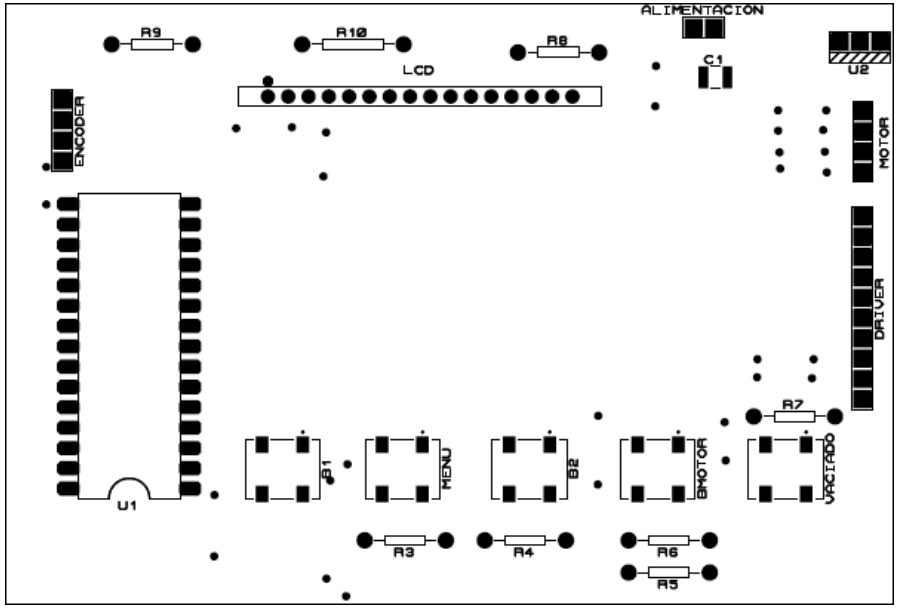
Índice de los planos



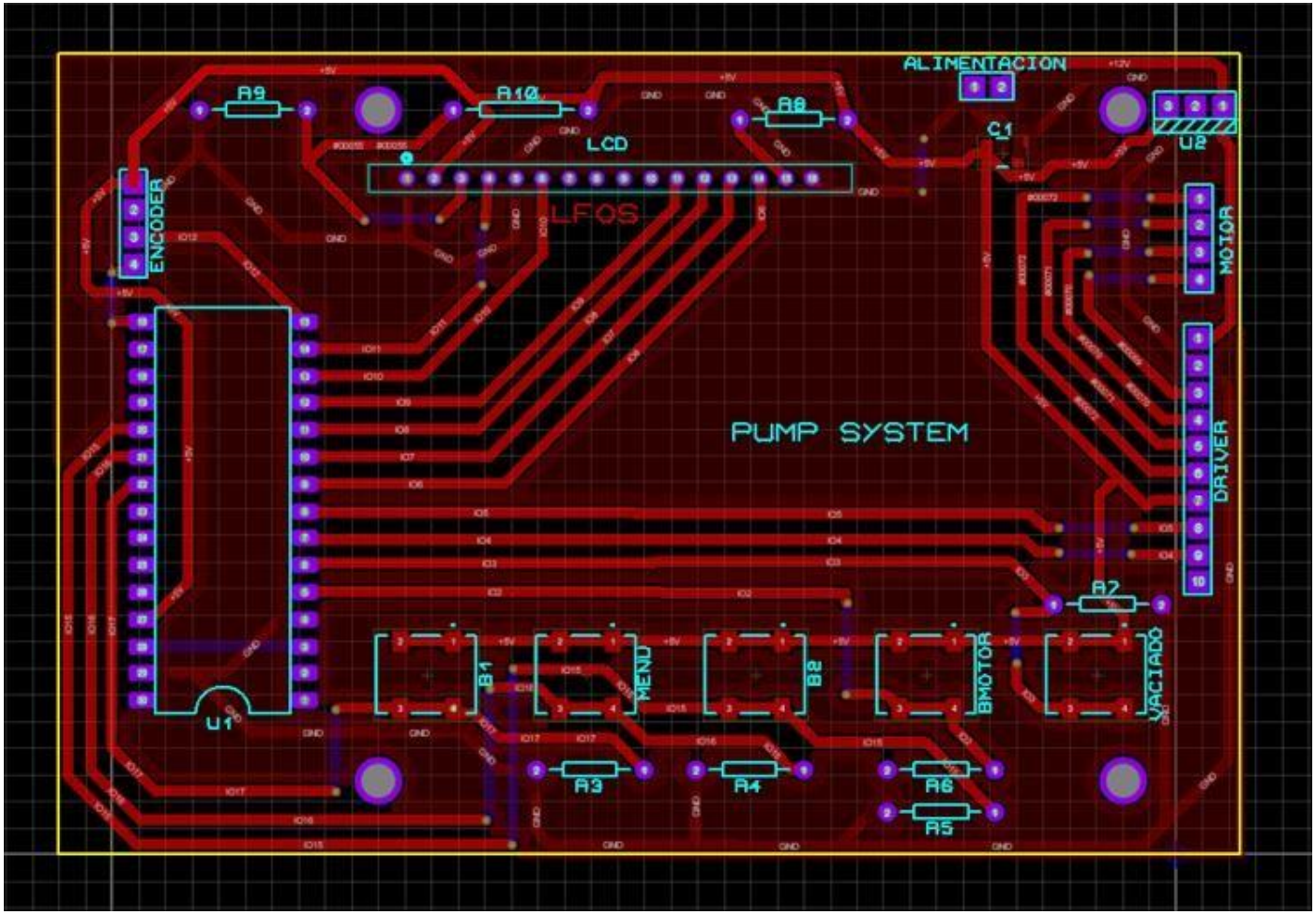
PROYECTO: Diseño y realización de un sistema de inyección – infusión de alta precisión, para aplicaciones de sensado en tiempo real		**ESCALA:**
		FECHA: 13/07/2024
AUTOR: Luis Fernando Olmos Saldías	**PLANO:** Circuito eléctrico	**PLANO:** 01



PROYECTO: Diseño y realización de un sistema de inyección – infusión de alta precisión, para aplicaciones de sensado en tiempo real		**ESCALA: 1:1**
		FECHA: 15/07/2024
AUTOR: Luis Fernando Olmos Saldías	**PLANO:** Circuito impreso	**PLANO:** 02

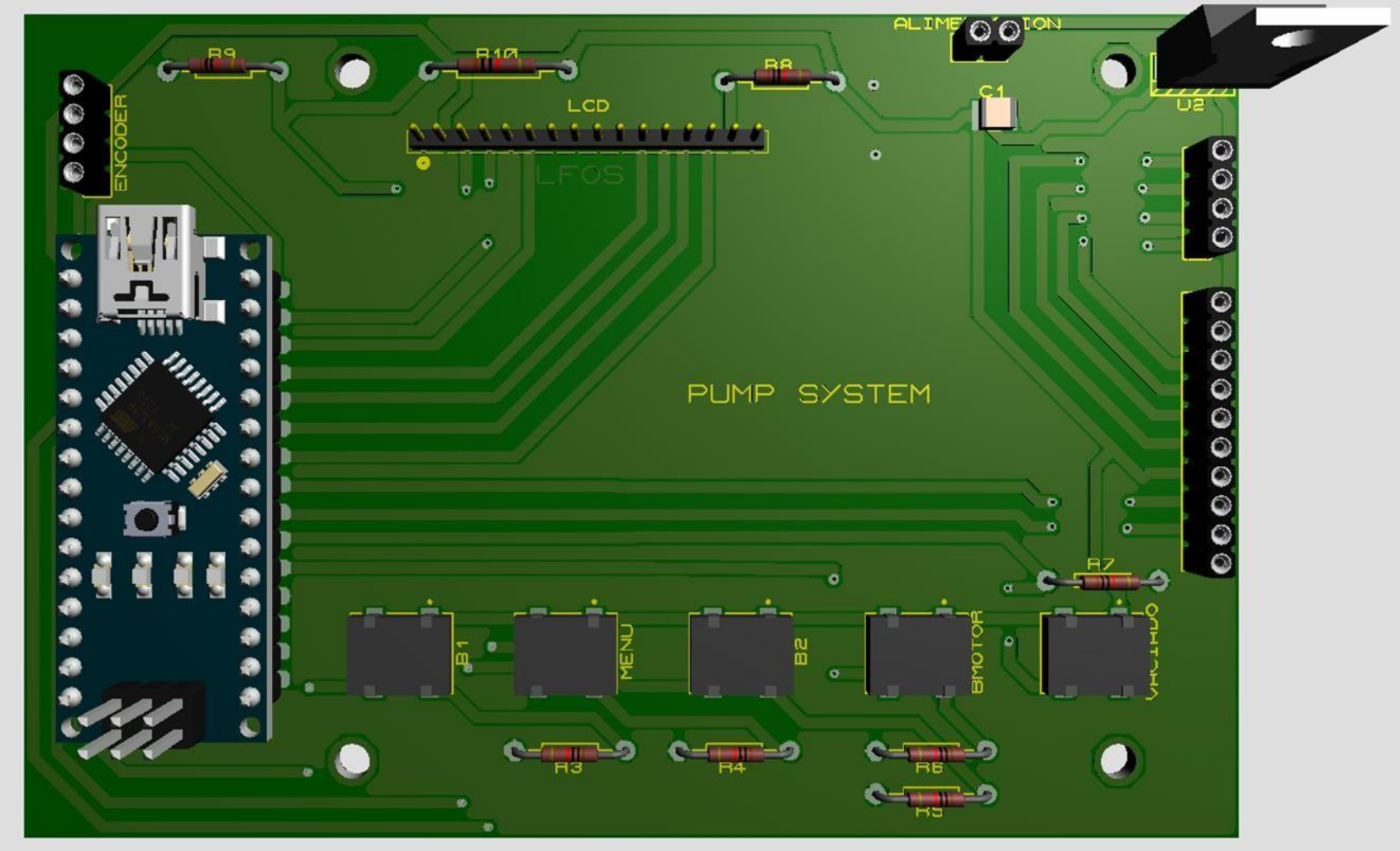


PROYECTO: Diseño y realización de un sistema de inyección – infusión de alta precisión, para aplicaciones de sensado en tiempo real		**ESCALA: 1:1**
		FECHA: 15/07/2024
Autor: Luis Fernando Olmos Saldías	**Plano:** Cara de los componentes del circuito	**PLANO:** 03



PROYECTO: Diseño y realización de un sistema de inyección – infusión de alta precisión, para aplicaciones de sensado en tiempo real		**ESCALA:** Cada cuadrado corresponde a 2.5 mm
		FECHA: 15/07/2024
AUTOR: Luis Fernando Olmos Saldías	**PLANO:** Circuito impreso con los componentes integrados	**PLANO:** 04

.



PROYECTO: Diseño y realización de un sistema de inyección – infusión de alta precisión, para aplicaciones de sensado en tiempo real		**ESCALA:**
		FECHA: 15/07/2024
Autor: Luis Fernando Olmos Saldías	**Plano:** Vista 3D del circuito impreso con los componentes integrados	**PLANO:** 05

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ETSI Aeroespacial y Diseño Industrial

UNIVERSIDAD POLITÉCNICA DE VALENCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño y realización de un sistema de inyección – infusión de alta precisión, para aplicaciones de sensado en tiempo real

DOCUMENTO Nº3: PLIEGO DE CONDICIONES

Trabajo final de grado

Grado en Ingeniería Electrónica Industrial y Automática

Autor: Olmos Saldías, Luis Fernando

Tutor: Ponce Alcántara, Salvador

Índice del pliego de condiciones

1.CONDICIONES GENERALES

1.1 Vigencia

El presente pliego de condiciones técnicas adquiere carácter de obligatorio cumplimiento una vez sellado y legalizado. Estará en vigor durante todo el proceso de reacondicionamiento de la bomba de jeringa, incluyendo el diseño del sistema electrónico y la programación del software y su posterior fabricación. En el caso de discrepancias entre los distintos documentos que conforman el proyecto, el orden de prioridad será el siguiente:

1- Planos del circuito electrónico y circuito impreso
2- Pliego de condiciones
3- Memoria
4- Presupuesto

1.2 Descripción

El presente proyecto de fin de grado se basa en un reacondicionamiento de una bomba de jeringa comercial, con el fin de ser utilizada en aplicaciones de sensado en tiempo real. El sistema está basado en un motor paso a paso, un sistema de engranajes reductores junto con la matriz de una bomba de jeringa desechada, una jeringa, un microcontrolador y una interfaz de control. Sus principales características son el bajo coste, su sencilla interfaz de control, y el uso de un software libre y accesible para los usuarios.

1.3 Pliegos oficiales

El contratista es responsable de cumplir con la Directiva de Baja Tensión (2014/35/UE) del Parlamento Europeo y del Consejo. En cuanto a la gestión de los residuos generados durante la fabricación del dispositivo, es obligatorio cumplir con el Real Decreto 110/2015, de 20 de febrero, sobre residuos de aparatos eléctricos y electrónicos.

1.4 Modificaciones

Todas las modificaciones realizadas durante la ejecución del proyecto deben ser revisadas y aprobadas por el responsable de la dirección del proyecto.

1.5 Dirección e inspección

El responsable de la dirección del proyecto tiene la obligación de asegurar la correcta fabricación del dispositivo, pudiendo delegar esta tarea en el personal encargado de la ejecución del proyecto.

2. CONDICIONES FACULTATIVAS

El director del proyecto es el responsable de revisar el trabajo realizado y los materiales empleados para su montaje. En el caso de no disponer los materiales indicados para la realización de este, los nuevos materiales que sustituyan a los indicados deben ser aprobados por el director del proyecto. Todas las modificaciones o aclaraciones del proyecto deben constar en un documento redactado por el contratista. La fabricación y el montaje del sistema electrónico debe ser realizada estrictamente como se ha estipulado en el proyecto que ha servido la base a la contratación y a posibles modificaciones aprobadas por el director. En caso haber dudas, se estipulará un comité entre el director de obra y el contratista para aclararlas.

3. CONDICIONES DE LOS COMPONENTES

3.1 Descripción

3.1.1 Alimentación del sistema

La fuente de alimentación para el motor será de 12 V y 1 A. Junto con ella, se incluirá un regulador de tensión de 5 V, que será utilizado para alimentar al controlador y los demás componentes del proyecto.

3.1.2 Motor

El motor presente en la matriz de la bomba de jeringa es un motor paso a paso de 200 pasos por vuelta, con una alimentación de 12 V y 300 mA.

3.1.3 Driver del motor

El controlador del motor debe tener la capacidad de poder alimentar con 12 V al motor y al menos 150 mA por bobina en cada canal. También deberá ser capaz de gestionar el giro del motor mediante salidas digitales y tener una reducción de pasos de al menos 1600 pasos por vuelta.

3.1.4 Microcontrolador

Debe tener al menos 16 salidas digitales. Algunas de ellas pueden ser analógicas empleadas como digitales que se utilizarán para el completo control del driver y la interfaz del motor. La alimentación del microcontrolador debe ser de 5 V la placa debe tener un software libre. El tamaño debe ser valorado poder implementarlo en una PCB con los demás componentes, debido a que el circuito impreso está pensado para un controlador de 45 x 18 mm.

3.1.5 Interfaz

El control del sistema debe ser sencillo e intuitivo. El usuario debe ser capaz de modificar todos los parámetros de la bomba mostrarlos mediante un display LCD en tiempo real.

3.2 Controles de calidad

3.2.1 Alimentación del sistema

Para comprobar el correcto funcionamiento de la alimentación, se ha de conectar un voltímetro en paralelo para comprobar que la tensión.

3.2.2 Motor

Para comprobar correctamente la intensidad demandada por el motor debemos ponerlo en funcionamiento y, mediante un amperímetro en serie con la bobina del motor, comprobar que la corriente demandada no supera la corriente nominal.

3.2.3 Driver del motor

Se debe alimentar a la tensión requerida por el motor, en este proyecto 12 V. Una vez alimentado se ha de comprobar que la comunicación con el microcontrolador es correcta, y que el driver no se sobrecalienta tras al menos 4 horas de uso.

3.2.4 Microcontrolador

Se debe alimentar el microcontrolador por su pin de +5 V. Una vez alimentado se debe encender el led de la placa.

3.2.5 Interfaz

Una vez se haya montado el circuito electrónico como se muestra en los planos, se debe comprobar que el display LCD tiene una correcta resolución, y que se muestran todos los parámetros de forma legible y clara para el usuario.

4. Condiciones de ejecución

4.1 Descripción

Para el montaje del circuito se deben conectar y soldar todos los componentes a la placa PCB. Y conectar la PCB en la matriz de la bomba. En lugar del montaje debe estar libre de polvo y suciedad. Y para el montaje del circuito se deben emplear guantes de latex. En el supuesto que algún componente se haya ensuciado, ha de ser limpiado con cuidado con alcohol.

4.2 Controles de calidad

Mediante un multímetro comprobaremos que las conexiones hacen contacto correctamente, y que no existe ningún cortocircuito que pueda dañar algún componente o perjudicar el funcionamiento de la bomba. Antes de realizar este paso debemos llevar a cabo las comprobaciones del correcto funcionamiento de todos los componentes como se ha indicado en el punto anterior.

5. PRUEBAS Y AJUSTES FINALES

Se debe comprobar el funcionamiento final de la bomba de jeringa, comprobando que los valores del flujo obtenidos coinciden con los valores expuestos en el proyecto. Se debe verificar también que el dispositivo cumpla con lo establecido en el Real Decreto 186/2016, de 6 de mayo, que regula la compatibilidad electromagnética de los equipos eléctricos y electrónicos.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ETSI Aeroespacial y Diseño Industrial

UNIVERSIDAD POLITÉCNICA DE VALENCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

Diseño y realización de un sistema de inyección – infusión de alta precisión, para aplicaciones de sensado en tiempo real

DOCUMENTO Nº4: PRESUPUESTO

Trabajo final de grado

Grado en Ingeniería Electrónica Industrial y Automática

Autor: Olmos Saldías, Luis Fernando

Tutor: Ponce Alcántara, Salvador

Índice de tablas

Tabla 1. Presupuestó de planificación, diseño y ejecución del proyecto

Descripción	Precio (€/h)	Horas	Total (€)
Planificación del proyecto	35	35	1225
Diseño del sistema electrónico y selección de componentes	35	70	2450
Desarrollo del software	35	75	2625
Diseño de la PCB	35	15	525
Montaje del circuito	25	4	100
Validación	35	55	1925
Redacción de la memoria	35	50	1750
	Subtotal	**304**	**10600**
21% IVA			2226
		Total	**12826**

Tabla 2. Presupuestó de los equipos utilizados

Descripción	Vida útil (años)	Horas de uso	Factor de amortización	Precio (€)	Total (€)
Fuente de alimentación	6	90	0.008	91.28	0.73
Multímetro	6	50	0.001	49.59	0.05
Ordenador portátil	5	160	0.017	850.41	14.46
Software	1	160	0.083	467.77	38.82
				Subtotal	54.06
21% IVA					11.35
				Total	65.41

Tabla 3. Presupuesto de los componentes utilizados.

Descripción	Precio (€/ud.)	Unidades	Total (€)
Arduino NANO	5.77	1	5.77
Driver TB6600	4.96	1	4.96
Botones, display, resistencias, condensador	2.57	1	2.57
Fuente de alimentación de 12 V y 2 A	7.85	1	7.85
Regulador de tensión 7805	1.46	1	1.46
Circuito impreso	35.00	1	35.00
		Subtotal	51.67
21% IVA			10.85
		Total	62.52

Tabla 4. Resumen del presupuesto

Descripción	Precio (€)
Diseño y ejecución del proyecto	10600.00
Equipos	54.06
Componentes	51.67
Subtotal	10705.73
10% de costes indirectos	1070.57
10% de beneficio industrial	1070.57
Suma	12846.87
21% IVA	2697.84
Total	15544.71

El presupuesto total para realizar este proyecto es de quince mil quinientos cuarenta y cuatro euros con setenta y un céntimos.

Anexo I

```
#include <LiquidCrystal.h>

//Pines del display LCD
LiquidCrystal lcd(6,7,8,9,10,11);

// Pines de conexión del driver
const int dirPin = 4;
const int stepPin = 5;

// Pin de interrupción del botón de vaciado
const int interruptPin = 3;

// Pin de interrupción del botón del motor
const int interruptMotor=2;

// Pines de conexión del encoder rotatorio
const int CLK=13;
const int DT=12;

// Pines de conexión de BUTTON1, BUTTON2 y MENU USANDO LAS ENTRADAS ANALOGICAS
COMO DIGITALES
const int analogPin1 = A1;  // Define el pin analógico A1
const int Button2 = 15; // Número del pin digital correspondiente a A1 en Arduino Uno
const int analogPin2 = A2;  // Define el pin analógico A2
const int MENU = 16; // Número del pin digital correspondiente a A2 en Arduino Uno
const int analogPin3 = A3;  // Define el pin analógico A3
const int Button1 = 17; // Número del pin digital correspondiente a A3 en Arduino Uno

// Variables encoder
int enco=0;
int op=1;
int StateCLK;
int lastStateCLK;

// Variables de la función menuajustes()
float diam=14.5;
char sentg[3];
int counter1=1;
int velod=0;
int velou=5;
int sentgir;
int vel=0;

// Variables para el cálculo del delay y la realimentación del motor
unsigned long startMilis; // Variable para almacenar el tiempo de inicio
```

```
unsigned long endMilis;   // Variable para almacenar el tiempo de fin
unsigned long elapsedMilis; // Variable para almacenar el tiempo transcurrido
float tteorico;
float treal;
float stepDelay;
float teoricostepdelay;
float velocidad=5;
float diametro=1;

// Variable para el estado
volatile bool state = LOW;

// Variables de las banderas
int bandera=0;
int bandera1=1;

// Variables para los botones BUTTON1, MENU, BUTTON2 y Motor
unsigned long lastButtonPress1 = 0;
unsigned long lastButtonPress2 = 0;
unsigned long lastButtonPress3 = 0;
unsigned long lastButtonPress4 = 0;

void setup() {

  lcd.begin(16,2);
  Serial.begin(9600); // Comunicación monitor serie
  sentg[0] = '<';
  sentg[1] = '-';
  sentg[2] = '\0';
  pinMode(dirPin, OUTPUT);
  pinMode(stepPin, OUTPUT);
  pinMode(Button1, INPUT);
  pinMode(Button2, INPUT);
  pinMode(MENU, INPUT);
  pinMode(CLK,INPUT);
  pinMode(DT,INPUT);
  digitalWrite(dirPin, LOW);
  pinMode(interruptPin, INPUT);
  attachInterrupt(digitalPinToInterrupt(interruptPin), interruptHandler, RISING);
  attachInterrupt(digitalPinToInterrupt(interruptMotor), interruptHandlermotor, RISING);
  velocidad=velod+velou;
}

void loop() {
   // Igualamos nuestra variable int vel a la float velocidad
   // para mostrarla correctamente en la pantalla lcd
   vel=velocidad;
   //Mostramos los parámetros en el display LCD
   lcd.setCursor(0,0);
```

```
    lcd.print("              ");
    lcd.setCursor(0,0);
    lcd.print("DIAM: ");
    lcd.setCursor(5,0);
    lcd.print(diam);
    lcd.setCursor(10,0);
    lcd.print("mm OFF");
    lcd.setCursor(0,1);
    lcd.print("              ");
    lcd.setCursor(0,1);
    lcd.print(sentg);
    lcd.setCursor(2,1);
    lcd.print("FLOW:");
    lcd.setCursor(7,1);
    lcd.print(vel);
    lcd.setCursor(10,1);
    lcd.print("uL/min");

    if(digitalRead(MENU)==HIGH){
     delay(500);
     bandera=1;
    }
    if(bandera==1){
    delay(500);
    menuajustes();
    }
    if(bandera==2){
     motoronoff();
    }

}



void motoronoff(){

    lcd.setCursor(0,0);
    lcd.print("              ");
    lcd.setCursor(0,0);
    lcd.print("DIAM: ");
    lcd.setCursor(5,0);
    lcd.print(diam);
    lcd.setCursor(10,0);
    lcd.print("mm ON");
    lcd.setCursor(0,1);
    lcd.print("              ");
    lcd.setCursor(0,1);
    lcd.print(sentg);
    lcd.setCursor(2,1);
```

```
  lcd.print("FLOW:");
  lcd.setCursor(7,1);
  lcd.print(vel);
  lcd.setCursor(10,1);
  lcd.print("uL/min");
  stepDelay =(1/velocidad)*20.98/(1/63.9)*diametro;
  teoricostepdelay=stepDelay;
  tteorico=stepDelay*80*2*2;
  motorapunto(enco);
  startMilis=millis();



do{


 if(digitalRead(Button1)==HIGH){
      velod=contador1d(velod);
      if(velod>12)
      velod=0;
      velocidad=velod*10+velou;
      if(velocidad<5)
      velocidad=5;
      if(velocidad>120){
      velocidad=120;
      velod=12;
      velou=0;
      }
      vel=velocidad;
      lcd.setCursor(0,0);
      lcd.print("              ");
      lcd.setCursor(0,0);
      lcd.print("DIAM: ");
      lcd.setCursor(5,0);
      lcd.print(diam);
      lcd.setCursor(10,0);
      lcd.print("mm ON");
      lcd.setCursor(0,1);
      lcd.print("              ");
      lcd.setCursor(0,1);
      lcd.print(sentg);
      lcd.setCursor(2,1);
      lcd.print("FLOW:");
      lcd.setCursor(7,1);
      lcd.print(vel);
      lcd.setCursor(10,1);
      lcd.print("uL/min");
      stepDelay =(1/velocidad)*20.98/(1/63.9)*diametro;
      teoricostepdelay=stepDelay;
```

```
  tteorico=stepDelay*80*2*2;
  motorapunto(enco);
  startMilis=millis();
}

if(digitalRead(Button2)==HIGH){
  velou=contador2u(velou);
  if(velou>9)
  velou=0;
  velocidad=velod*10+velou;
  if(velocidad<5)
  velocidad=5;
  if(velocidad>120){
  velocidad=120;
  velod=12;
  velou=0;
  }
  vel=velocidad;
  lcd.setCursor(0,0);
  lcd.print("              ");
  lcd.setCursor(0,0);
  lcd.print("DIAM: ");
  lcd.setCursor(5,0);
  lcd.print(diam);
  lcd.setCursor(10,0);
  lcd.print("mm ON");
  lcd.setCursor(0,1);
  lcd.print("              ");
  lcd.setCursor(0,1);
  lcd.print(sentg);
  lcd.setCursor(2,1);
  lcd.print("FLOW:");
  lcd.setCursor(7,1);
  lcd.print(vel);
  lcd.setCursor(10,1);
  lcd.print("uL/min");
  stepDelay =(1/velocidad)*20.98/(1/63.9)*diametro;
  teoricostepdelay=stepDelay;
  tteorico=stepDelay*80*2*2;
  motorapunto(enco);
  startMilis=millis();
}



digitalWrite(stepPin, HIGH);
delay(stepDelay);
digitalWrite(stepPin, LOW);
```

```
   delay(stepDelay);
   StateCLK = digitalRead(CLK);
if (StateCLK != lastStateCLK  && StateCLK == 1){
   if (digitalRead(DT) != StateCLK) {
     enco --;
   } else {
     enco ++;
     if(enco==2){
     endMilis=millis();
     elapsedMilis= endMilis-startMilis;
     stepDelay=(tteorico/elapsedMilis)*teoricostepdelay*diametro;
     enco=0;
     startMilis=millis();
     }
   }
  }
  lastStateCLK = StateCLK;
}
while(bandera1 % 2 == 0);
}


int contador1(int counter){
  int btnState = digitalRead(Button1);
  if (btnState == HIGH) {
    if(millis() - lastButtonPress1 >= 80) {
      counter--;
     }
    lastButtonPress1 = millis();
  }
  delay(1);
  return counter;
}

int contador2(int counter){
   int btnState = digitalRead(Button2);
   if (btnState == HIGH) {
    if (millis() - lastButtonPress2 >= 80) {
      counter++;
     }
   lastButtonPress2 = millis();
   }
   delay(1);
   return counter;
}

int motorapunto(int encoder){
  encoder=0;
```

```
do{
  if(digitalRead(Button1)==HIGH){
    velod=contador1d(velod);
    if(velod>12)
    velod=0;
    velocidad=velod*10+velou;
    if(velocidad<5)
    velocidad=5;
    if(velocidad>120){
    velocidad=120;
    velou=0;
    }
    vel=velocidad;
    lcd.setCursor(0,0);
    lcd.print("              ");
    lcd.setCursor(0,0);
    lcd.print("DIAM: ");
    lcd.setCursor(5,0);
    lcd.print(diam);
    lcd.setCursor(10,0);
    lcd.print("mm ON");
    lcd.setCursor(0,1);
    lcd.print("              ");
    lcd.setCursor(0,1);
    lcd.print(sentg);
    lcd.setCursor(2,1);
    lcd.print("FLOW:");
    lcd.setCursor(7,1);
    lcd.print(vel);
    lcd.setCursor(10,1);
    lcd.print("uL/min");
    stepDelay =(1/velocidad)*20.98/(1/63.9)*diametro;
    teoricostepdelay=stepDelay;
    tteorico=stepDelay*80*2*2;

  }

  if(digitalRead(Button2)==HIGH){
    velou=contador2u(velou);
    if(velou>9)
    velou=0;
    velocidad=velod*10+velou;
    if(velocidad<5)
    velocidad=5;
    if(velocidad>120){
    velocidad=120;
    velod=12;
    velou=0;
    }
```

```
      vel=velocidad;
      lcd.setCursor(0,0);
      lcd.print("              ");
      lcd.setCursor(0,0);
      lcd.print("DIAM: ");
      lcd.setCursor(5,0);
      lcd.print(diam);
      lcd.setCursor(10,0);
      lcd.print("mm ON");
      lcd.setCursor(0,1);
      lcd.print("              ");
      lcd.setCursor(0,1);
      lcd.print(sentg);
      lcd.setCursor(2,1);
      lcd.print("FLOW:");
      lcd.setCursor(7,1);
      lcd.print(vel);
      lcd.setCursor(10,1);
      lcd.print("uL/min");
      stepDelay =(1/velocidad)*20.98/(1/63.9)*diametro;
      teoricostepdelay=stepDelay;
      tteorico=stepDelay*80*2*2;

    }

    lastStateCLK = digitalRead(CLK);
    digitalWrite(stepPin, HIGH);
    delay(stepDelay);
    digitalWrite(stepPin, LOW);
    delay(stepDelay);
    StateCLK = digitalRead(CLK);

if (StateCLK != lastStateCLK  && StateCLK == 1){

  if (digitalRead(DT) != StateCLK) {
     enco --;

   } else {
     enco ++;
   }
}
 lastStateCLK = StateCLK;
      }
      while(enco==0 && bandera1%2==0);

      enco=0;
      return encoder;
}
```

```
void menuajustes(){
 do{
 if(digitalRead(Button1)==HIGH){
  op=contador1(op);
  if(op<1)
  op=4;
  Serial.print("Opción:");
  Serial.println(op);
 }
  if(digitalRead(Button2)==HIGH){
  op=contador2(op);
  if(op>4)
  op=1;
  Serial.print("Opción:");
  Serial.println(op);
 }
 switch(op){
  case 1:
   lcd.setCursor(0,0);
   lcd.print("              ");
   lcd.setCursor(4,0);
   lcd.print("AJUSTES");
   lcd.setCursor(0,1);
   lcd.print("              ");
   lcd.setCursor(2,1);
   lcd.print("SENT. GIRO");
   if(digitalRead(MENU)==HIGH){
    delay(500);
    elegirsent();
   }
  break;

  case 2:
   lcd.setCursor(0,0);
   lcd.print("              ");
   lcd.setCursor(4,0);
   lcd.print("AJUSTES");
   lcd.setCursor(0,1);
   lcd.print("              ");
   lcd.setCursor(3,1);
   lcd.print("DIAMETRO");
   if(digitalRead(MENU)==HIGH){
    delay(500);
    elegirdiam();
   }
  break;

  case 3:
   lcd.setCursor(0,0);
```

```
    lcd.print("              ");
    lcd.setCursor(4,0);
    lcd.print("AJUSTES");
    lcd.setCursor(0,1);
    lcd.print("              ");
    lcd.setCursor(4,1);
    lcd.print("FLUJO");
    if(digitalRead(MENU)==HIGH){
     delay(500);
     elegirvelo();
    }
   break;

   case 4:
    lcd.setCursor(0,0);
    lcd.print("              ");
    lcd.setCursor(4,0);
    lcd.print("AJUSTES");
    lcd.setCursor(0,1);
    lcd.print("              ");
    lcd.setCursor(3,1);
    lcd.print("ATRAS");
    if(digitalRead(MENU)==HIGH){
     bandera=0;
     delay(500);
     op=1;
    }
   break;
  }
 } while(bandera==1);

}

void elegirsent(){
 int op1=1;
  do{
   Serial.print("estamos dentro");
  if(digitalRead(Button1)==HIGH){
   op1=contador1(op1);
   if(op1<1)
   op1=2;
   Serial.print("Opción:");
   Serial.println(op1);
  }
   if(digitalRead(Button2)==HIGH){
   op1=contador2(op1);
   if(op1>2)
   op1=1;
   Serial.print("Opción:");
```

```
  Serial.println(op1);
 }
 switch(op1){
  case 1:
      lcd.setCursor(0,0);
      lcd.print("              ");
      lcd.setCursor(2,0);
      lcd.print("SENT. GIRO");
      lcd.setCursor(0,1);
      lcd.print("              ");
      sentg[0] = '<';
      sentg[1] = '-';
      sentg[2] = '\0';
      lcd.setCursor(6,1);
      lcd.print(sentg);
      if(digitalRead(MENU)==HIGH){
          sentg[0] = '<';
          sentg[1] = '-';
          sentg[2] = '\0';
          digitalWrite(dirPin, LOW);
          op=2;
         }
   break;
   case 2:
      lcd.setCursor(0,0);
      lcd.print("              ");
      lcd.setCursor(2,0);
      lcd.print("SENT. GIRO");
      lcd.setCursor(0,1);
      lcd.print("              ");
      sentg[0] = '-';
      sentg[1] = '>';
      sentg[2] = '\0';
      lcd.setCursor(6,1);
      lcd.print(sentg);
      if(digitalRead(MENU)==HIGH){
          sentg[0] = '-';
          sentg[1] = '>';
          sentg[2] = '\0';
          digitalWrite(dirPin, HIGH);
          op=2;
         }
  break;
 }
 } while(digitalRead(MENU)==LOW);
 op1=1;
 delay(500);
}
```

```
void elegirdiam(){
 int op1=1;
 do{
  if(digitalRead(Button1)==HIGH){
  op1=contador1(op1);
  if(op1<1)
  op1=3;
  Serial.print("Opción:");
  Serial.println(op1);
 }
  if(digitalRead(Button2)==HIGH){
  op1=contador2(op1);
  if(op1>3)
  op1=1;
  Serial.print("Opción:");
  Serial.println(op1);
 }
  switch(op1){
  case 1:
     lcd.setCursor(0,0);
     lcd.print("              ");
     lcd.setCursor(3,0);
     lcd.print("Diametro");
     lcd.setCursor(0,1);
     lcd.print("              ");
     lcd.setCursor(4,1);
     lcd.print("14.50");
     if(digitalRead(MENU)==HIGH){
          diametro=1; //variable para hacer el cáculo de velocidades
          diam=14.50;
          Serial.println("Diametro=14.50");
          Serial.print(diametro);
          op=3;
         }
  break;

  case 2:
     lcd.setCursor(0,0);
     lcd.print("              ");
     lcd.setCursor(3,0);
     lcd.print("Diametro");
     lcd.setCursor(0,1);
     lcd.print("              ");
     lcd.setCursor(4,1);
     lcd.print("19.13");
     if(digitalRead(MENU)==HIGH){
          diametro=pow(19.13/2,2)/pow(14.5/2,2); //variable para hacer el cáculo de velocidades
          diam=19.13;
          op=3;
```

```
          Serial.println("Diametro=12.06");
          Serial.print(diametro);
         }
   break;
   case 3:
      lcd.setCursor(0,0);
      lcd.print("             ");
      lcd.setCursor(3,0);
      lcd.print("Diametro");
      lcd.setCursor(0,1);
      lcd.print("             ");
      lcd.setCursor(4,1);
      lcd.print("12.06");
      if(digitalRead(MENU)==HIGH){
          diametro=pow(12.06/2,2)/pow(14.5/2,2); //variable para hacer el cáculo de velocidades
          diam=12.06;
          op=3;
          Serial.println("Diametro=12.06");
          Serial.print(diametro);
         }
   break;
  }
 } while(digitalRead(MENU)==LOW);
 op1=0;
 delay(500);
}

void elegirvelo(){
 do{

  if(digitalRead(Button1)==HIGH){
    velod=contador1d(velod);
    if(velod>12)
     velod=0;
  }
  if(digitalRead(Button2)==HIGH){
    velou=contador2u(velou);
    if(velou>9)
     velou=0;
  }
  velocidad=velod*10+velou;
    if(velocidad<5){
     velocidad=5;
    velou=5;
  }
  if(velocidad>120){
    velocidad=120;
    velou=0;
    velod=12;
```

```
  }
  vel=velocidad;

  lcd.setCursor(0,0);
  lcd.print("              ");
  lcd.setCursor(4,0);
  lcd.print("FLUJO");
  lcd.setCursor(0,1);
  lcd.print("              ");
  lcd.setCursor(4,1);
  lcd.print(vel);

 }while(digitalRead(MENU)==LOW);
  delay(500);
  op=4;
}

int contador1d(int counter){
  int botonState = digitalRead(Button1);
 if (botonState == HIGH) {
   if(millis() - lastButtonPress1 >= 80) {
     counter++;
     }
   lastButtonPress1 = millis();
 }
delay(1);
return counter;
}
int contador2u(int counter){
  int botnState = digitalRead(Button2);
 if (botnState == HIGH) {
   if(millis() - lastButtonPress1 >= 80) {
     counter++;
     }
   lastButtonPress1 = millis();
 }

delay(1);
return counter;
}

void interruptHandler() {

   lcd.setCursor(0,0);
   lcd.print("              ");
   lcd.setCursor(4,0);
   lcd.print("VACIADO");
   lcd.setCursor(0,1);
   lcd.print("              ");
```

```
 int stepDelay2 = 250;

 do {
   state = digitalRead(interruptPin);
   digitalWrite(stepPin, HIGH);
   delayMicroseconds(stepDelay2);
   digitalWrite(stepPin, LOW);
   delayMicroseconds(stepDelay2);
 }
 while(state==HIGH);
      lcd.setCursor(0,0);
      lcd.print("              ");
      lcd.setCursor(0,0);
      lcd.print("DIAM: ");
      lcd.setCursor(5,0);
      lcd.print(diam);
      lcd.setCursor(10,0);
      lcd.print("mm ON");
      lcd.setCursor(0,1);
      lcd.print("              ");
      lcd.setCursor(0,1);
      lcd.print(sentg);
      lcd.setCursor(3,1);
      lcd.print("FLOW:");
      lcd.setCursor(7,1);
      lcd.print(vel);
      lcd.setCursor(10,1);
      lcd.print("uL/min");
      motorapunto(enco);

}

void interruptHandlermotor() {

 int btnState4 = digitalRead(interruptMotor);
 if (btnState4 == HIGH) {
   if (millis() - lastButtonPress4 >= 200) {
     bandera1++;
   }
   lastButtonPress4 = millis();
 }
 delay(10);
 if(bandera1 % 2 == 0)
   bandera=2;
 else bandera=0;
}
```