Escuela Politécnica Superior de Jaén

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN
Nombre del Centro

Trabajo Fin de Grado

APLICACIÓN DE UN MOTOR PASO A PASO PARA EJERCICIOS DE REHABILITACIÓN

Alumno: Álvaro Sánchez Anquela

Tutor: Ángel Gaspar González Rodríguez
Dpto: Ingeniería de Sistemas y Automática

Julio, 2020

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don ÁNGEL GASPAR GONZALEZ RODRÍGUEZ , tutor del Proyecto Fin de Carrera titulado:

APLICACIÓN DE UN MOTOR PASO A PASO PARA EJERCICIOS DE REHABILITACIÓN, que presenta ÁLVARO SÁNCHEZ ANQUELA, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, JUNIO de 2020

El alumno:

ÁLVARO SANCHEZ ANQUELA

El tutor:

GONZALEZ RODRIGUEZ ANGEL GASPAR - 26014077L

Firmado digitalmente por GONZALEZ RODRIGUEZ ANGEL GASPAR - 26014077L
Fecha: 2020.09.02 16:32:37 +02'00'

ÁNGEL GASPAR GONZALEZ RODRIGUEZ

INDICE

1. INTRODUCCIÓN

Los seres humanos están sujetos a tener cierto grado de incidentes traumáticos a lo largo de la vida, los cuales obstaculizan a las personas en realización de tareas de su vida cotidiana. Cuando un músculo no se utiliza, ya sea por una operación o por un simple traumatismo, tiende a acortarse dando como resultado que las articulaciones se vuelvan rígidas y puedan ocasionar deformidades o contracturas. Realizando estímulos a través del movimiento de las articulaciones con máquinas de fisioterapia o con terapia psicoterapéutica, todos estos problemas se reducen considerablemente.

Casi cualquier rehabilitación post-operación suele ser normalmente bastante larga y dolorosa, dependiendo del tipo de cirugía realizada. La mejor manera de realizar una rehabilitación es ponerse en manos de un especialista, ya que son ellos los que normalmente tienen todas las máquinas y/o herramientas necesarias para una correcta recuperación.

Una máquina de rehabilitación puede ser desde algo tan sencillo como una banda elástica, o algo tan complejo que apenas se pueda intuir como funciona. Existe un gran rango de máquinas y aparatos del que poder valerse para superar cualquier tipo de traumatismo. También resultan de gran ayuda para la realización de estiramientos activos y pasivos, y hacer mejorar en general la calidad de vida.

No todo el mundo dispone de tiempo y/o recursos para poder realizar sesiones de fisioterapia en una clínica, o simplemente algunas veces es más fácil y más cómodo quedarse en casa para la realización de dichos ejercicios. En este proyecto se realizará el estudio, montaje y programación de una pequeña máquina de rehabilitación que se pueda usar en casa y que sea de bajo coste. Esta máquina de fisioterapia está dirigida principalmente a usuarios que se encuentren en las primeras fases de recuperación tras una cirugía traumatológica.

2. OBJETIVO Y ESPECIFICACIONES DE DISEÑO

El objetivo de este proyecto se basa en la aplicación de un motor paso a paso para realización de ejercicios básicos de movilidad en articulaciones y miembros del cuerpo. Está diseñado específicamente para una movilidad pasiva del paciente el cual puede, gracias a un mando a distancia, controlar ciertas variables del motor tales como velocidad, dirección y recorrido. La diferencia entre movilidad activa y pasiva, es que en movilidad activa es el paciente quien realiza la fuerza, mientras que en la pasiva quien realiza la fuerza es la máquina.

Este proyecto está pensado para pacientes con movilidad reducida tras una cirugía o traumatismo grave, o personas mayores que no dispongan de ayuda para realizar ciertos ejercicios de movilidad pasiva.

Se pretende que con un sistema de control barato como es Arduino, y el hardware adicional para controlar el motor paso a paso, cualquier persona o clínica pueda tener acceso a una máquina de fisioterapia a precio reducido y un gran abanico de posibilidades de uso.

El proyecto se diseña desde cero, empezando desde la selección de todo el hardware necesario así como el diseño del software y el encapsulamiento de todo el sistema para su fácil uso y transporte.

2.1 Especificaciones de diseño

El equipo dispone de una fuente de alimentación para transformar la tensión de 220 V de cualquier enchufe a una tensión de 60 V que será la utilizada por el driver de control. Dicho driver, que funciona entre 24 V y 80 V, es el responsable de alimentar y controlar el motor paso a paso NEMA 34 que se ha utilizado.

Para controlar al driver y por extensión al motor, se utiliza un Arduino UNO en el que se programa el software necesario. También se hace uso de un sensor de proximidad y de un sensor de corriente.

Para la interfaz hombre-máquina que se necesita para regular el motor según las necesidades de cada persona, se utiliza un mando a distancia vía infrarrojos.

El equipo dispone de una serie de ejercicios pre programados que el usuario puede seleccionar fácilmente. En esos ejercicios el motor se mueve de acuerdo a un perfil de velocidades previamente definidos y programados, y del que se puede escoger la velocidad, la dirección y la extensión que se quiere avanzar. También se puede desactivar la activación de las bobinas para que no estén constantemente en funcionamiento, y dos botones para activar y desactivar el muestreo de valores de intensidad.

2.2 Modificaciones sobre el objetivo inicial

Dadas las circunstancias acontecidas a lo largo de 2020 por el COVID-19, fecha en la que se realiza este proyecto, y con las instalaciones universitarias cerradas se hace indispensable la modificación de algunos de los objetivos inicialmente requeridos para este proyecto. Dichas modificaciones se realizan por la imposibilidad tener acceso a un laboratorio donde realizar ciertas pruebas necesarias.

La primera modificación es la utilización de Arduino en lugar de Raspberry PI o un PLC Siemens S7. Se utiliza este hardware, ya que es el único que se tenía disponible para realizar el proyecto.

La segunda y última modificación se realiza por la imposibilidad de realizar el estudio de las características del par, y por tanto no se puede dar ni recibir perfiles de par. Para compensar esta modificación se realiza la interfaz hombre-máquina con el mando a distancia y que así el usuario sea capaz de controlar el motor, no pudiendo sin embargo controlar el par. Este también es el motivo de que se enfoque el proyecto sobre todo para movilidad pasiva, ya que el par motor es una característica bastante importante si se quisiese realiza movilidad activa.

Dado que el par es una característica importante, podría realizarse su estudio y adaptación posteriormente con algunas mejoras. También sería interesante la colaboración con alguien del sector de la readaptación deportiva, ya que son ellos los que estudian todo tipo de ejercicios, ángulos de las articulaciones y fuerzas que deben de utilizarse para el correcto uso de esta máquina.

3. MATERIAL UTILIZADO

3.1 Motor paso a paso

El encargado de dar el movimiento y la fuerza al sistema es el motor paso a paso bipolar Nema 34 de doble eje de la Ilustración 3.2, capaz de realizar un movimiento de 1.8º por paso y con una torsión mecánica de 18 kg/cm.

Dicho motor bipolar contiene 2 bobinas, A y B, que se activan y desactivan de forma secuencial y alterna para controlar en todo momento la posición del motor. En la BIBLIOGRAFÍA se adjunta un directorio donde explica en profundidad el funcionamiento de un motor paso a paso.

Dispone de 4 cables los cuales se muestran en Ilustración 3.1 y siguen las instrucciones del fabricante:

Rojo → Bobina A+

Verde → Bobina A-

Amarillo → Bobina B+

Azul → Bobina B-

Conexionado motor:

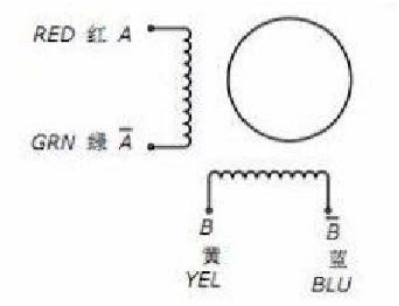

Ilustración 3.1 Conexionado del motor

Ilustración 3.2 Motor paso a paso

3.2 Fuente de alimentación

La fuente de alimentación o transformador de la Ilustración 3.3 es la encargada de transformar la tensión de 220 V de cualquier enchufe doméstico a una tensión de 60 V en corriente continua, que será el voltaje que utilizará el driver para controlar el motor paso a paso.

Ilustración 3.3 Fuente de alimentación

3.3 Driver

En la Ilustración 3.4 se muestra el driver encargado de controlar totalmente el motor, tanto la dirección como la velocidad, y también la activación de desactivación de las bobinas con el “Enable”.

Funciona con un rango de tensiones de entrada de entre 24 V y 80 V en corriente continua, y con un rango de máximo amperaje de salida de entre 2.8 A y 7.8 A programable según un sistema de interruptores o “switches”.

También según el mismo sistema se pueden configurar los pulsos necesarios para obtener una revolución del motor, concretamente se puede configurar entre 400 pulsos/revolución y 51.200 pulsos/revolución. Más adelante en el capítulo CONEXIONADO se expondrá la configuración utilizada y su explicación, así como el esquema de conexiones. En la BIBLIOGRAFÍA se adjunta una pequeña guía para un driver de similares características a este.

Ilustración 3.4 Driver

3.4 Arduino

Todo el entorno de control está desarrollado con hardware de Arduino, tanto la placa con el procesador, como otros elementos que se especifican a continuación.

3.4.1 Arduino Uno

Como microprocesador se usa Arduino, un microprocesador bien conocido por la comunidad electrónica por ser “Open Source” y un precio bastante reducido. Cumple perfectamente la labor que se le pide para este proyecto, pudiendo desarrollarse en él todo el software necesario para controlar el driver y con ello, el motor paso a paso. Se muestra a continuación en la Ilustración 3.5.

Ilustración 3.5 Arduino Uno

3.4.2 Arduino Screw Shield

También se requiere de la tarjeta de desarrollo Arduino Screw Shield de la Ilustración 3.6 para el correcto anclaje de todo el conexionado, ya que no es posible anclar con tornillos directamente en Arduino. Con esta placa se permite la conexión de los cables en terminales con tornillos.

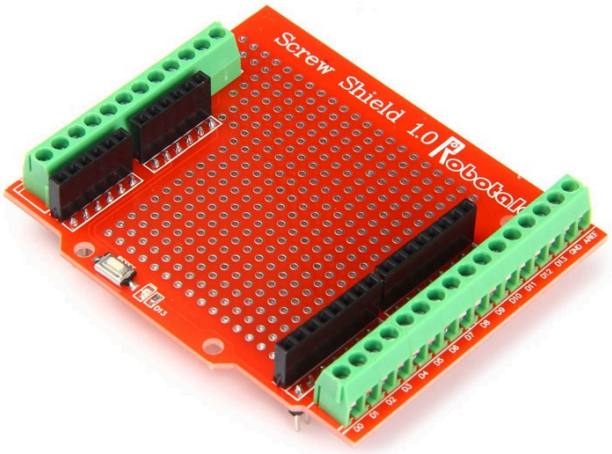

Ilustración 3.6 Arduino Screw Shield

3.4.3 Mini placa board y bloque de terminales

Igualmente se utiliza una pequeña placa board (Ilustración 3.7) que es posible acoplar encima de la Screw Shield. En ella también se puede realizar algún sencillo conexionado así como la colocación de unos pequeños bloques de terminales (Ilustración 3.8) para el correcto anclaje del cableado.

Ilustración 3.7 Mini placa board

Ilustración 3.8 Bloques de terminales

3.4.4 Mando a distancia y receptor infrarrojo

Se usa un pequeño mando a distancia como el de la Ilustración 3.9, el cual sirve de enlace para interactuar entre paciente y máquina. En la BIBLIOGRAFÍA se adjunta una pequeña guía sobre las diferentes funciones de la biblioteca IRremote.

Para que Arduino pueda recibir la información proveniente del mando a distancia también se hace necesario un pequeño receptor infrarrojo (Ilustración 3.10).

Ilustración 3.9 Mando a distancia

Ilustración 3.10 Receptor Infrarrojo

3.4.5 Sensor de corriente efecto Hall

Pequeño sensor de corriente efecto Hall como el de la Ilustración 3.11 que conectado a la bobina A permite medir la intensidad en dicha bobina en cualquier momento. Se conecta tanto a Arduino como al motor paso a paso y permite medir valores de hasta 30 A de intensidad.

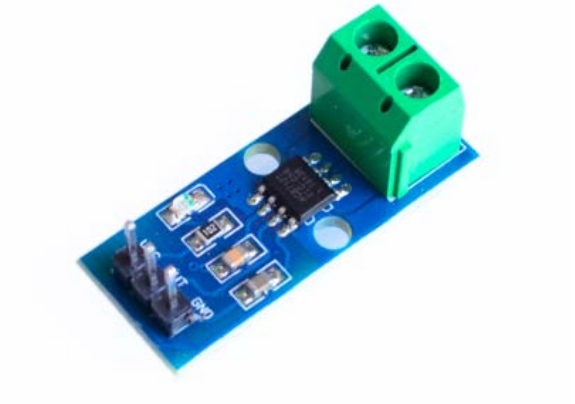

Ilustración 3.11 Sensor efecto Hall

3.4.6 Sensor de proximidad infrarrojo

Se usa también un sensor infrarrojo de proximidad (Ilustración 3.12), cuya función es detectar el mango que el paciente usa para realizar los ejercicios creando así un punto “home” al inicio del programa, y con ello pudiendo saber en todo momento donde está situado el motor ya sea en coordenadas absolutas o relativas y actuar respecto a ello.

Ilustración 3.12 Sensor de proximidad infrarrojo

3.5 Caja de encapsulado

Caja metálica donde va embutido todo el sistema y donde se anclan todos los materiales anteriormente mencionados, ya sea directamente a la caja o entre sí. La caja se muestra la Ilustración 3.13.

Ilustración 3.13 Caja de encapsulado

4. CONEXIONADO

El inicio de todo es la fuente de alimentación que se conecta a cualquier enchufe común de 220 V por un lado y a las conexiones de VCC y GND del driver por el otro.

Cada uno de los cuatro cables del motor se lleva a un pin específico del driver, que está definido por el fabricante según su color. Uno de los cables que alimenta a una bobina, concretamente el rojo conectado en A+, se hace pasar en serie por los terminales atornillados del sensor de efecto Hall (Ilustración 4.1). El circuito de salida del sensor se ha alimentado a 5V y GND, y su salida se conecta a uno de los pines analógicos de Arduino, concretamente al pin A0.

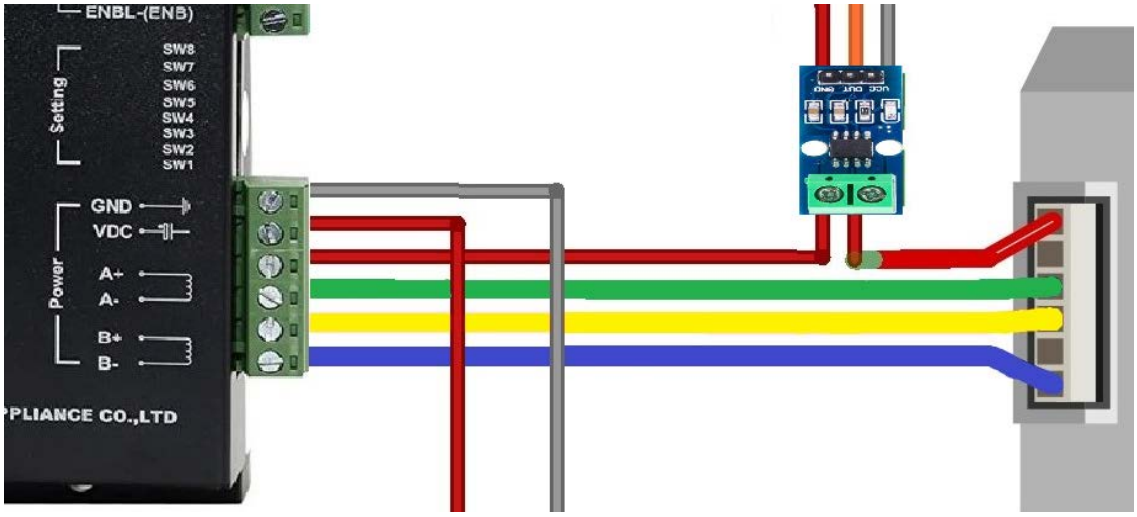

Ilustración 4.1 Conexión sensor efecto Hall

Para la conexión de la parte de control del driver se utiliza una configuración de ánodo común, en la que todos los positivos se conectan entre sí y al pin de 5 V de Arduino, la señal llega a través de cada uno de los negativos correspondientes.

PULL (-) → Pin 7

DIR (-) → Pin 6

ENBL (-) → Pin 5

CONEXIONADO

Los dos elementos que quedan por mencionar son el receptor infrarrojo que va conectado a pin 11 y el sensor de proximidad que se conecta al pin 4. Ambos están conectados también a 5 V y GND según las especificaciones del fabricante.

En la Ilustración 4.2 se esquematiza el conexionado completo utilizado en el proyecto.

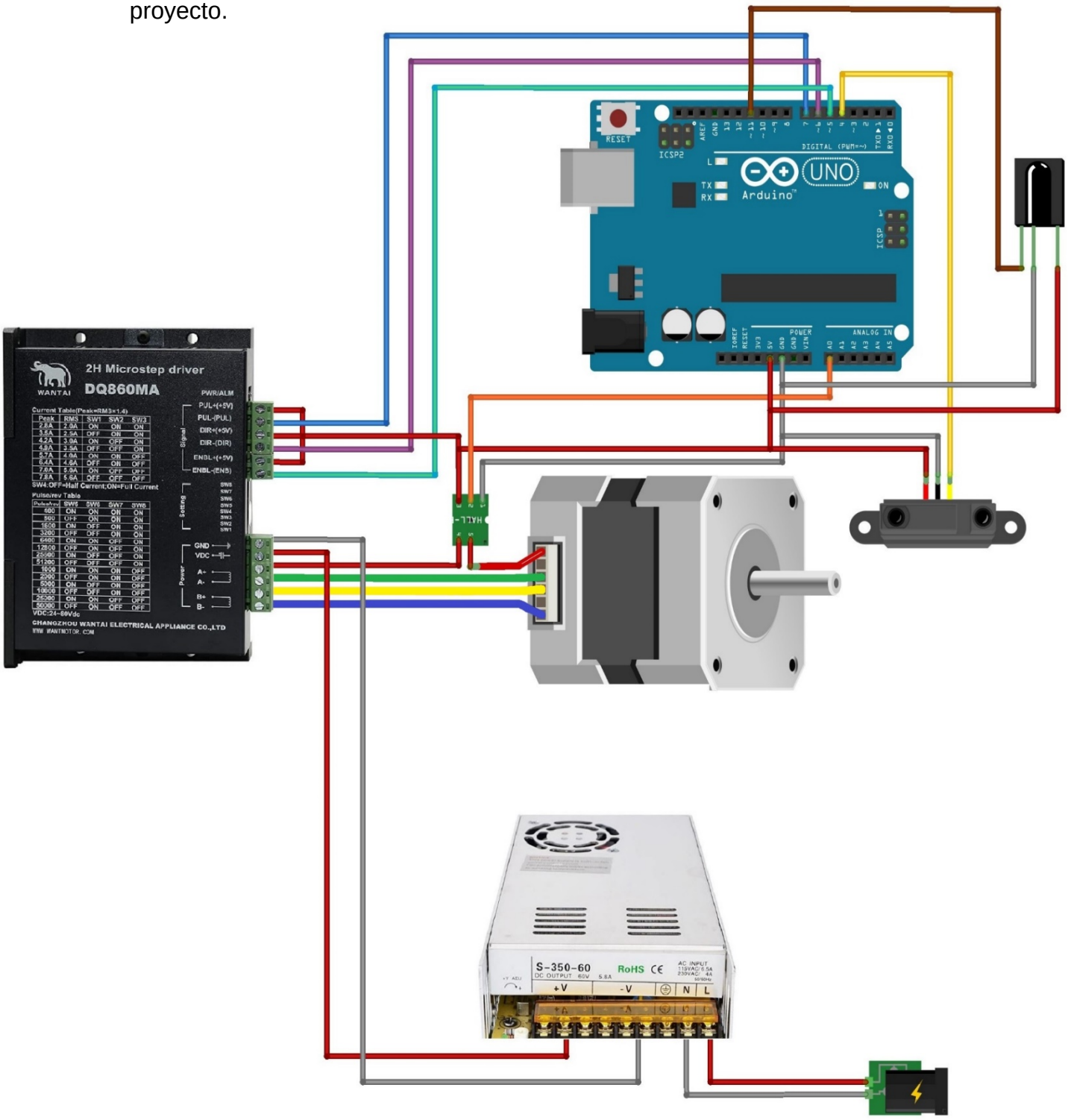

Ilustración 4.2 Esquema de conexiones

5. ENTORNO DE PROGRAMACIÓN

5.1 Microsoft Visual Studio

Como ya se ha mencionado anteriormente el hardware utilizado es Arduino, pero para el desarrollo del software en lugar del Arduino IDE (integrated development environment) que es propio de Arduino, se ha utilizado “Microsoft Visual Studio” (Ilustración 5.1).

Al igual que el software de Arduino, este entorno de programación también es código abierto o “Open Source” y permite el desarrollo en prácticamente todos los lenguajes de programación existentes.

El principal motivo por el que se cambia de entorno de programación es porque el entorno Arduino no dispone de ninguna opción para depuración de código, algo de gran importancia en programas con cierta complejidad.

En BIBLIOGRAFÍA se adjunta un enlace para la descarga y configuración de este entorno de programación.

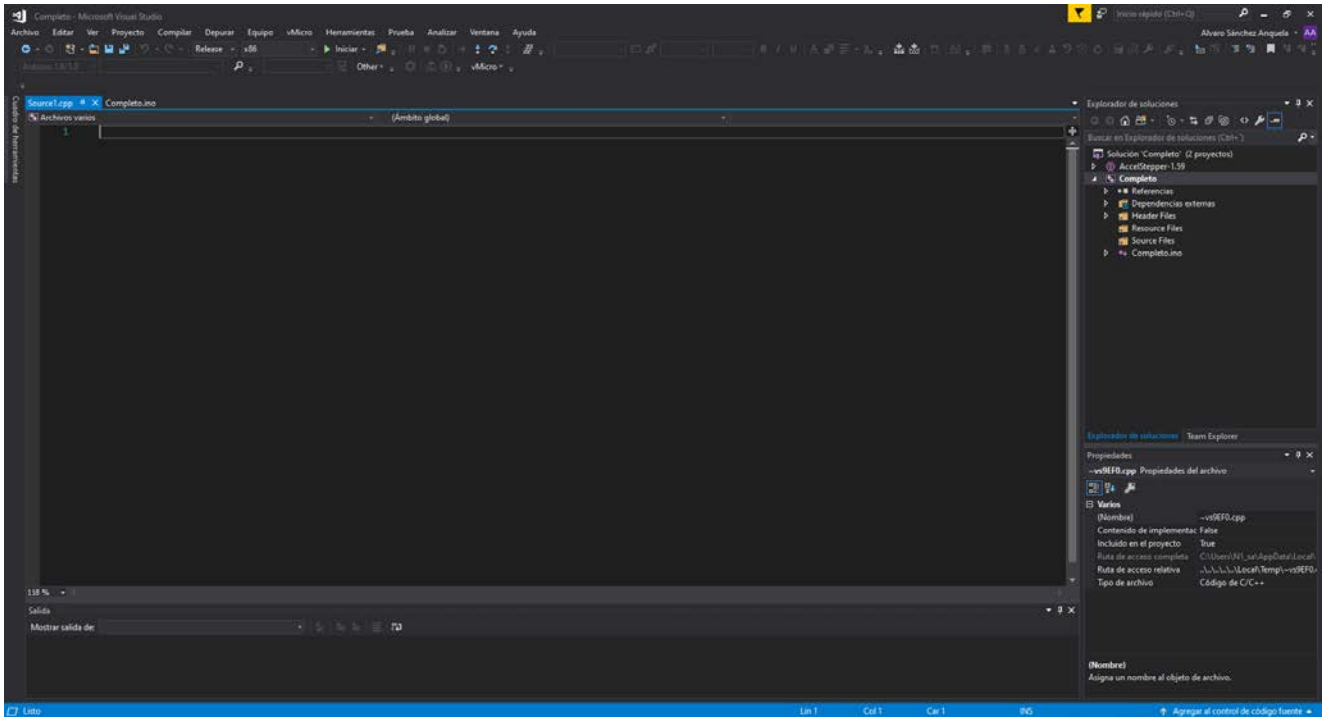

Ilustración 5.1 Entorno de programación Microsoft Visual Studio

6. FLUJOGRAMA

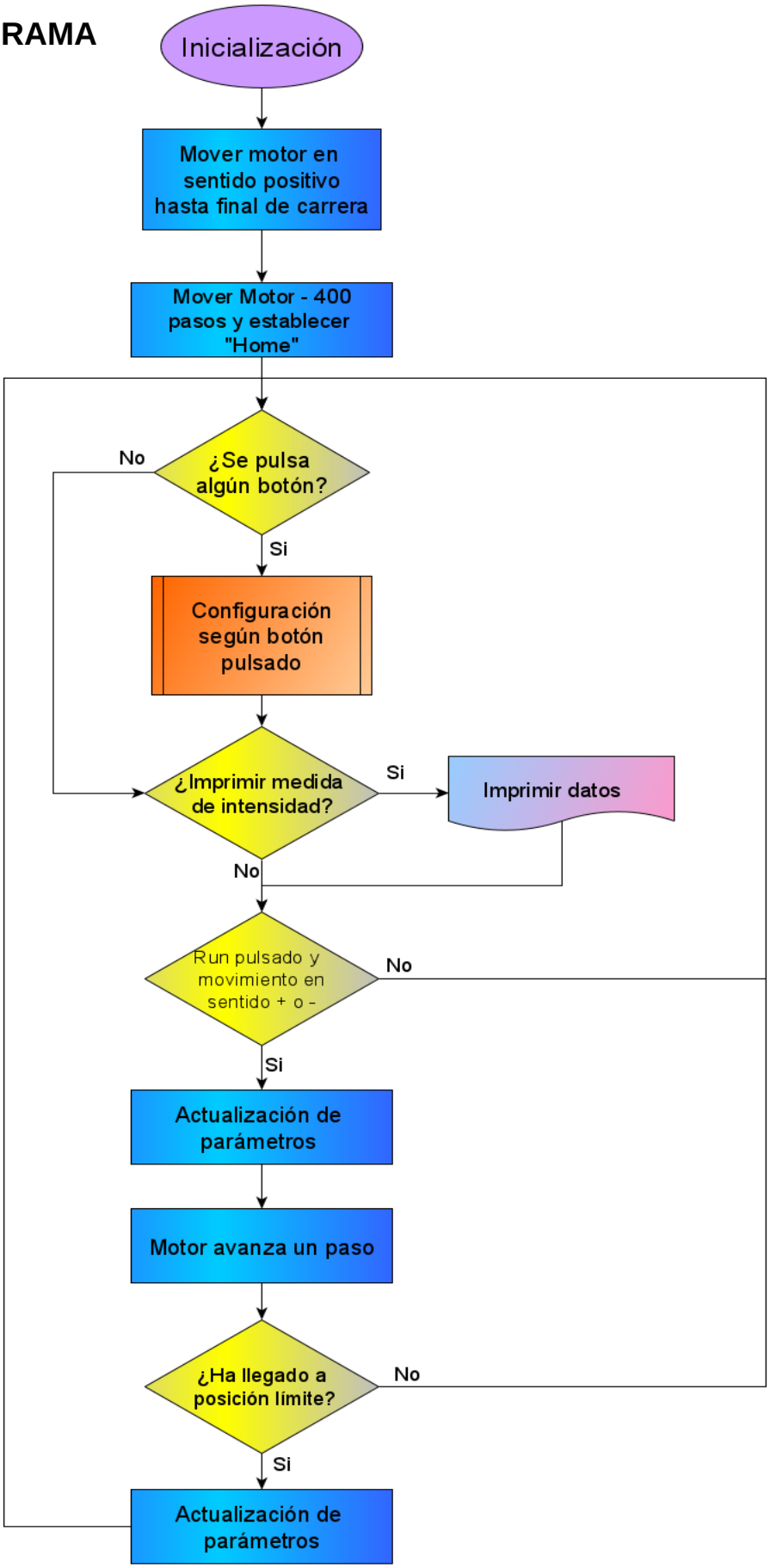

Ilustración 6.1 Diagrama de flujo

FLUJOGRAMA

A continuación se explica el desarrollo del diagrama de flujo de la Ilustración 6.1.

Una vez inicializado el programa y todas sus variables, el motor comienza a girar en sentido horario hasta que la agarradera que se usa para realizar los ejercicios es detectada por el sensor de movimiento infrarrojo. Una vez en ese punto se guarda la posición “home” y el motor gira 400 pasos en sentido antihorario, equivalente a 29.84 mm como posteriormente se verá. Esto se hace para que el máximo de cuerda a recoger sea la posición “home” menos 400 pasos, lo que hará que no se active continuamente el sensor de movimiento. Con ello se consigue que una vez guardada la posición “home”, el detector de movimiento pase a ser una parada de emergencia por si hubiese cualquier fallo.

El siguiente paso es preguntar si ha pulsado algún botón desde la última iteración. Si la respuesta es que sí, se busca qué botón ha sido pulsado y se realiza una asignación de variables según el botón pulsado. Se realiza así para no tener que preguntar directamente si alguno de los veintiún botones ha sido pulsado en todas las iteraciones, sino que solo entra en esa subrutina si alguno de ellos se ha pulsado.

A continuación se observa si está activada o no la variable que indica si se realiza la visualización de los datos de intensidad, ya que esta se puede activar o desactivar con dos botones del mando.

Una vez hecho esto se comprueba que las variables de funcionamiento estén activadas. Si no lo están simplemente se vuelve a iniciar el bucle preguntando si se ha pulsado algún botón.

Hecho esto se actualizan las variables y parámetros y se le pasan al motor a través del driver, y este avanza un paso. En esta actualización de parámetros se configura si se gira en un sentido o en otro y la velocidad del motor.

Justo después de avanzar un paso se comprueba si se ha llegado a la posición límite establecida, dándole así al motor un rango de giro máximo y mínimo.

7. CONFIGURACIÓN DE HARDWARE

Para controlar el motor, el driver permite la configuración física de una serie de “switches” o interruptores en los que se permite variar tres parámetros:

- **Intensidad pico:** Intensidad máximo que puede recorrer cada bobina durante el funcionamiento normal del motor. “Switches” del 1 al 3.

- **Función en semi-flujo: “**Switch” 4. Con esta función activada, la corriente de salida del controlador se reduce en un 40% si no se realiza ningún paso en los siguientes 200 ms tras el último paso.

- **Pulsos/revolución:** Cantidad de pulsos necesarios para que el motor gire una vuelta completa. “Switches” del 5 al 8.

En la Ilustración 3.4 se puede ver la tabla que el driver lleva impresa, con todos los valores ajustables y sus correspondientes “switches”.

Para este proyecto la configuración usada es la que se muestra en la Ilustración 7.1.

Ilustración 7.1 "Switches" de configuración del driver

Para la parte de intensidad de pico, 2.8 A es lo máximo que recibe cada bobina del motor en condiciones normales. Esta intensidad es más que suficiente para realizar la movilidad pasiva de cualquier extremidad del cuerpo. Los siguientes escalones de intensidad vistos en las tablas (hasta un máximo de 7.8 A) se podría utilizar para una mejora del proyecto en la que se realice movilidad activa, y el propio paciente tuviese que vencen la fuerza acometida por el motor.

La función semi-flujo permanece en modo desconectada, ya que no es conveniente para este proyecto que se reduzca la intensidad, y por tanto la fuerza del motor, de forma automática. Para suplir esta opción y evitar que las bobinas del motor estén continuamente excitadas, se configura una opción en el mando para que el propio usuario pueda eliminar la excitación ellas. Esto se realiza con la opción “Enable” que el propio driver lleva incorporada.

Para finalizar, la cantidad de pulsos configurada es de 1600, lo que significa que tenemos que enviar 1600 pulsos al driver para conseguir un giro de 360º. Esta opción se ha calibrado empíricamente, de forma que permita tener estabilidad en el motor tanto en movimientos lentos como en movimientos algo más rápidos.

El motor tiene un ángulo de giro de 1.8º nominales por activación de bobina, o lo que es lo mismo, se necesitarían 200 pulsos para realizar un giro completo. Al configurar el driver en 1600 pulsos/revolución lo que se está haciendo es hacer que el motor paso a paso funcione con micro pasos para reducir ese ángulo que gira el motor, pasando de girar 1.8º por pulso si no se utilizan micro pasos, a 0.225º por pulso en la configuración adoptada. Esto permite una gran exactitud en el giro y la consecución de un mayor patrón de par más uniforme sin ser necesario para este proyecto que una mayor exactitud, como haría falta por ejemplo en la implantación de un motor paso a paso para impresión 3D.

8. FUNCIONAMIENTO

Al arrancar el programa lo primero que se hace es girar el motor en sentido positivo (horario) hasta que el detector localice el agarre de la cuerda y el motor para. En este punto se localiza el punto 0 o punto “Home”, a partir del cual el resto de programa puede tener localizado en todo momento en que punto se encuentra el motor.

Teniendo el punto “Home” localizado, el motor se separa 400 pasos en sentido antihorario para que el detector no se active continuamente cada vez que llegue al punto “Home”. En lugar de ello a partir de aquí el detector simplemente actúa como un sistema de seguridad, en el que si por cualquier motivo la recuerda se intentase recoger demasiado, detectaría el mango y se pararía.

8.1 Funciones de los botones del mando del usuario

Una vez en este punto, si no se pulsa ningún botón del mando (Ilustración 8.1), el motor no hace nada. A continuación se define cual es la función de cada botón al presionarlo.

Ilustración 8.1 Mando a distancia

- Botón Apagar (Ilustración 8.2):

  El motor se detiene por completo. Se puede usar en todo momento excepto cuando está buscando su posición “Home”, ya sea al iniciar el programa o al reiniciar con “Botón EQ”.

Ilustración 8.2 Botón apagar

- Botón Encender (Ilustración 8.3):

  El motor comienza a girar con los parámetros de dirección y velocidad establecidos. Al inicio del programa, se tiene que realizar una selección en sentido antihorario, para que el motor no se active automáticamente si se pulsa este botón por accidente.

Ilustración 8.3 Botón encender

- Botón Mode (Ilustración 8.4):

  Esta es la opción “Enable” y se activa para que las bobinas reciban corriente .Está activado por defecto y para anularlo se pulsa el siguiente botón.

Ilustración 8.4 Botón Mode

- Botón Silenciar (Ilustración 8.5):

La otra opción de “Enable”, con este botón las bobinas del motor dejan de excitarse, por lo cual se puede girar con la mano sin ningún tipo de esfuerzo. Para poder activar esta opción el motor debe estar detenido. Su uso es reducido pero uno de sus usos podría ser por ejemplo que la cuerda quedara enganchada en algún lugar e hiciera falta girar el motor sin que este esté excitado para desengancharla.

Ilustración 8.5 Botón Silenciar

- Botón Decremento (Ilustración 8.6):

Con este botón se hace que el motor gire en sentido antihorario. Para poder activar esta dirección el motor tiene que estar detenido, para así evitar cambios bruscos de dirección y que el paciente pueda sufrir cualquier daño.

Ilustración 8.6 Botón decremento

- Botón Incremento (Ilustración 8.7):

Funciona exactamente igual que el botón anterior pero para configurar el motor en sentido horario.

Ilustración 8.7 Botón Incremento

- Botón Menos (Ilustración 8.8):

  Reduce la velocidad del motor. Se puede usar tanto si el motor está detenido como si está en movimiento. Dispone de 10 saltos de velocidades, con una velocidad mínima de 100 y máxima de 1000. Estos valores se encuentran definidos en el siguiente apartado.

Ilustración 8.8 Botón menos

- Botón Más (Ilustración 8.9):

  Funciona como el botón Más pero aumentando la velocidad.

Ilustración 8.9 Botón más

- Botón Medir (Ilustración 8.10):

  Inicia el muestreo de valores de intensidad en “bobina A” a través del puerto serie. Se puede usar tanto si el motor está detenido como en funcionamiento. Su uso sólo tiene sentido en modo desarrollo o

depuración, no siendo de utilidad para el usuario final. Como es lógico para poder visualizar los datos por puerto serie, ya sea en valores o en una gráfica, será necesario conectar Arduino a un ordenador.

Ilustración 8.10 Botón medir

- Botón U/SD (Ilustración 8.11):

  Cancela el muestreo de datos de corriente.

Ilustración 8.11 Botón U/SD

- Botón EQ (Ilustración 8.12):

  Devuelve el motor a la posición “Home” y reinicia los valores de posición, velocidad y dirección. Deja el sistema tal y como si se acabase de iniciar.

Ilustración 8.12 Botón EQ

- Botón 0 (Ilustración 8.13):

  Mueve el motor 2 pasos en el último sentido seleccionado y a una velocidad de 50. Las definiciones de velocidad y pasos del motor se muestran en capítulo DEFINICIÓN DE PARÁMETROS.

  Ejemplo: si se estaba girando en sentido horario y el motor se detiene, al pulsar este botón se gira en sentido horario 2 pasos a una velocidad de 50.

Ilustración 8.13 Botón 0

- Botón 1 (Ilustración 8.14):

  Mueve el motor 20 pasos en el último sentido seleccionado y a una velocidad de 50.

Ilustración 8.14 Botón 1

- Botón 2 (Ilustración 8.15):

  Mueve el motor 30 pasos en el último sentido seleccionado y a una velocidad de 50.

Ilustración 8.15 Botón 2

- Botón 3 (Ilustración 8.16):

  Mueve el motor 40 pasos en el último sentido seleccionado y a una velocidad de 50.

Ilustración 8.16 Botón 3

- Botón 4 (Ilustración 8.17):

  Mueve el motor 50 pasos en el último sentido seleccionado y a una velocidad de 50.

Ilustración 8.17 Botón 4

- Botón 5 (Ilustración 8.18):

  Mueve el motor 60 pasos en el último sentido seleccionado y a una velocidad de 50.

Ilustración 8.18 Botón 5

- Botón 6 (Ilustración 8.19):

  Mueve el motor 70 pasos en el último sentido seleccionado y a una velocidad de 50.

Ilustración 8.19 Botón 6

- Botón 7 (Ilustración 8.20):

  Mueve el motor 80 pasos en el último sentido seleccionado y a una velocidad de 50.

Ilustración 8.20 Botón 7

- Botón 8 (Ilustración 8.21):

  Mueve el motor 90 pasos en el último sentido seleccionado y a una velocidad de 50.

Ilustración 8.21 Botón 8

- Botón 9 (Ilustración 8.22):

Mueve el motor 100 pasos en el último sentido seleccionado y a una velocidad de 50.

Ilustración 8.22 Botón 9

8.2 Ejemplo de uso

Un ejemplo de uso cotidiano sería el siguiente:

Tras la puesta en marcha y llegada a la posición “Home” el usuario puede mover el motor hasta la posición deseada con los botones de encendido y apagado y controlar el sentido de giro y la velocidad con los botones de incremento o decremento y más o menos respectivamente.

Una vez colocado el usuario en una posición correcta para el estiramiento según su lesión, utilizando los botones desde el 0 al 9 podrá ir estirando paulatinamente según lo crea conveniente, siendo el botón 9 el que más pasos mueve el motor y el 0 el que menos.

En esta fase en la que el usuario usa los botones del 0 al 9 no se puede controlar la velocidad, ya que está limitada para que se realice de forma lenta y controlada.

Terminado el ejercicio realizado el usuario puede optar por soltar la agarradera y recolocar el motor con los botones expuestos en el primer párrafo, o directamente cambiar la dirección de giro y pulsar el botón encender para que se deje de realizar el estiramiento y el usuario pueda terminar.

Una vez hecho esto y con algunos segundos de descanso el usuario puede volver a empezar.

9. DEFINICIÓN DE PARÁMETROS

En este apartado se definen los valores de algunas variables del programa y se relacionan con su magnitud física real.

Velocidad

La variable velocidad se encuentra programada entre un valor de 50 para realizar movimientos lentos con los botones del 0 al 9 hasta un máximo de 1.000 que se alcanza pulsando el “Botón Más”, pero hasta ahora no se ha definido el equivalente de esa velocidad en una magnitud física. Dicha magnitud se medirá en segundos/revolución, ya que el valor medido en r.p.m. como indica el sistema internacional es muy pequeño para este proyecto.

Esta medición se realiza de forma empírica, programando el software para que realice 1600 pasos, que como se vio anteriormente son los pasos necesarios para que el motor realice una revolución.

Valores de velocidad utilizados:

- Velocidad = 50 → 32 segundos/revolución
- Velocidad = 100 → 16 segundos/revolución
- Velocidad = 200 → 14.4 segundos/revolución
- Velocidad = 300 → 12.8 segundos/revolución
- Velocidad = 400 → 11.2 segundos/revolución
- Velocidad = 500 → 9.6 segundos/revolución
- Velocidad = 600 → 8 segundos/revolución
- Velocidad = 700 → 6.4 segundos/revolución
- Velocidad = 800 → 4.8 segundos/revolución
- Velocidad = 900 → 3.2 segundos/revolución
- Velocidad = 1000 → 1.6 segundos/revolución

Este parámetro de velocidad está controlado por la librería AccelStepper de Arduino. En BIBLIOGRAFÍA se adjunta un enlace con funciones de dicha librería.

Pasos

Cada pulso que se le manda al motor este avanza un paso pero, ¿cuánta sección de cuerda se desenrolla por cada paso que el motor avanza?

Esta magnitud resulta más difícil de medir, ya que dependiendo de la cantidad de cuerda que quede enrollada en la polea, la longitud de cuerda que la polea desenrolla es diferente. La medición del radio se establece entre el eje de la polea y la distancia hasta la zona donde se encuentra la cuerda (con un 50% de cuerda desenrollada). El radio obtenido es de 1.9 cm.

Con esto y sabiendo que para realizar un giro completo del motor se necesitan 1600 pasos:

- 1 paso → 0.0746 mm
- 1600 pasos → 119.36 mm

Para una vuelta completa del motor la sección de cuerda entregada al usuario es 119.36 mm que será algo más cuando la cuerda esté totalmente enrollada y algo menos cuando esté completamente desenrollada. Con esto se puede calcular la sección de cuerda que se desenrolla al pulsar cado uno de los botones del 0 al 9.

- Botón 0 → 0.4192 mm
- Botón 1 → 1.492 mm
- Botón 2 → 2.238 mm
- Botón 3 → 2.984 mm
- Botón 4 → 3.730 mm
- Botón 5 → 4.476 mm
- Botón 6 → 5.222 mm
- Botón 7 → 5.968 mm
- Botón 8 → 6.714 mm
- Botón 9 → 7.460 mm

10. MONTAJE FINAL

A continuación en la Ilustración 9.1, Ilustración 10.2, Ilustración 10.3 e Ilustración 10.4 se muestra el proceso de montaje.

Ilustración 10.1 Caja vacía

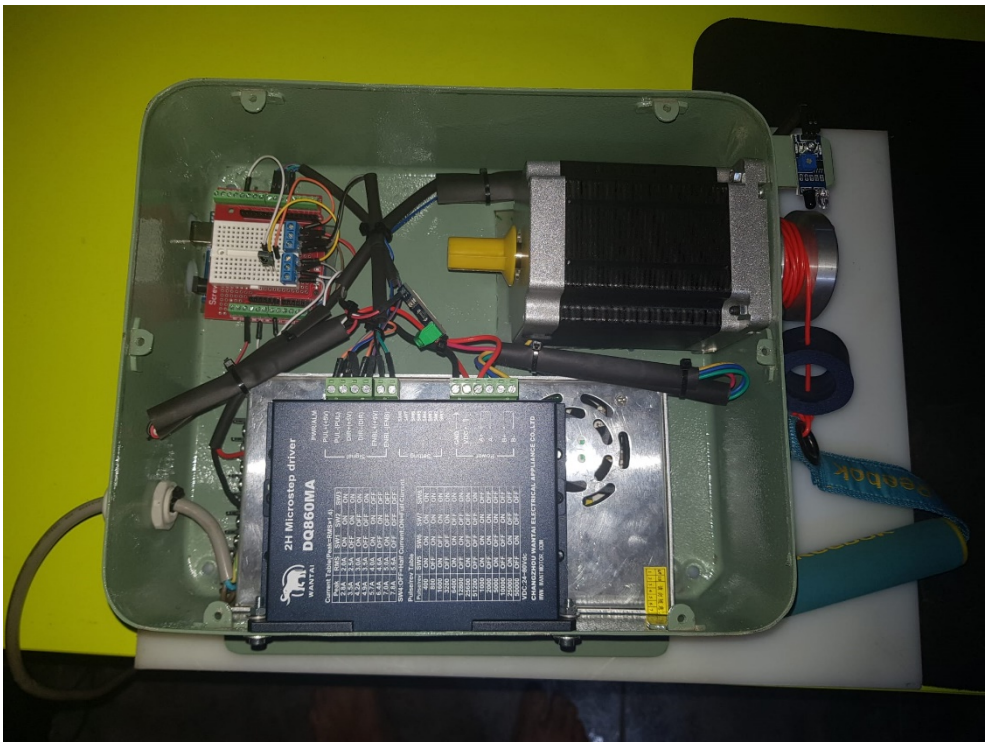

Ilustración 10.2 Caja montada

Ilustración 10.3 Lateral caja montada

Ilustración 10.4 Caja cerrada

11. INTENSIDAD REQUERIDA DURANTE EL FUNCIONAMIENTO

El seguimiento del valor de la intensidad que transcurre en cada momento por la bobina se puede activar y desactivar con el mando como ya se vio en el capítulo de FUNCIONAMIENTO. Aquí se muestran una serie de imágenes donde se puede observar dicha intensidad de forma gráfica en varios puntos de la ejecución como son:

- Retención motor desactivado (Ilustración 11.1):

  Aquí la intensidad que circula por la bobina es prácticamente cero, y es por este motivo por el que cuando la retención está desactivada el motor se puede hacer girar con la mano sin ningún tipo de dificultad.

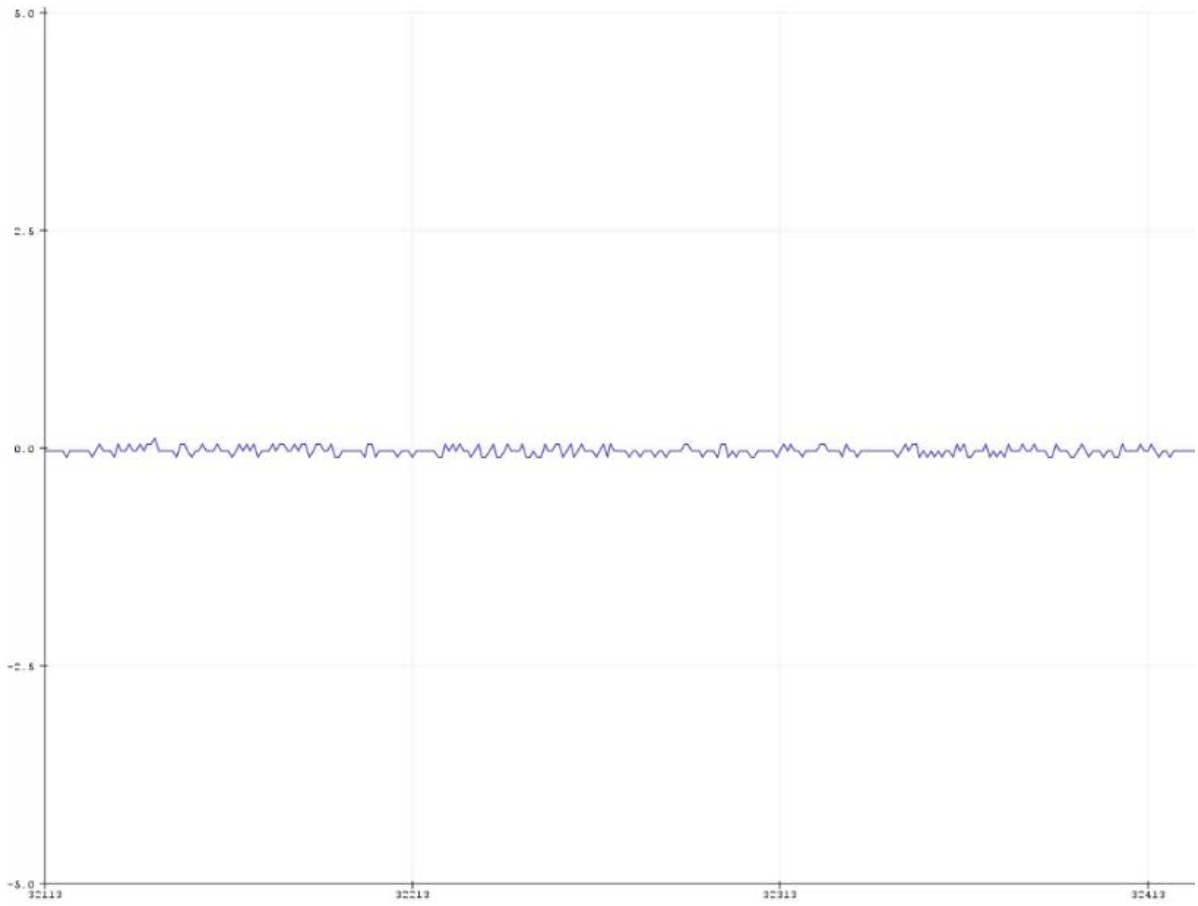

Ilustración 11.1 Intensidad con retención desactivada

- Retención activada (Ilustración 11.2):

  Con dicha intensidad recorriendo la bobina ya no existe la posibilidad de girar manualmente el motor con facilidad, ya que la intensidad podría elevarse hasta un máximo de 2.8 A si se realizase la fuerza necesaria para mover el motor. Esta intensidad varía según la configuración dada al driver a través de los “switches”.

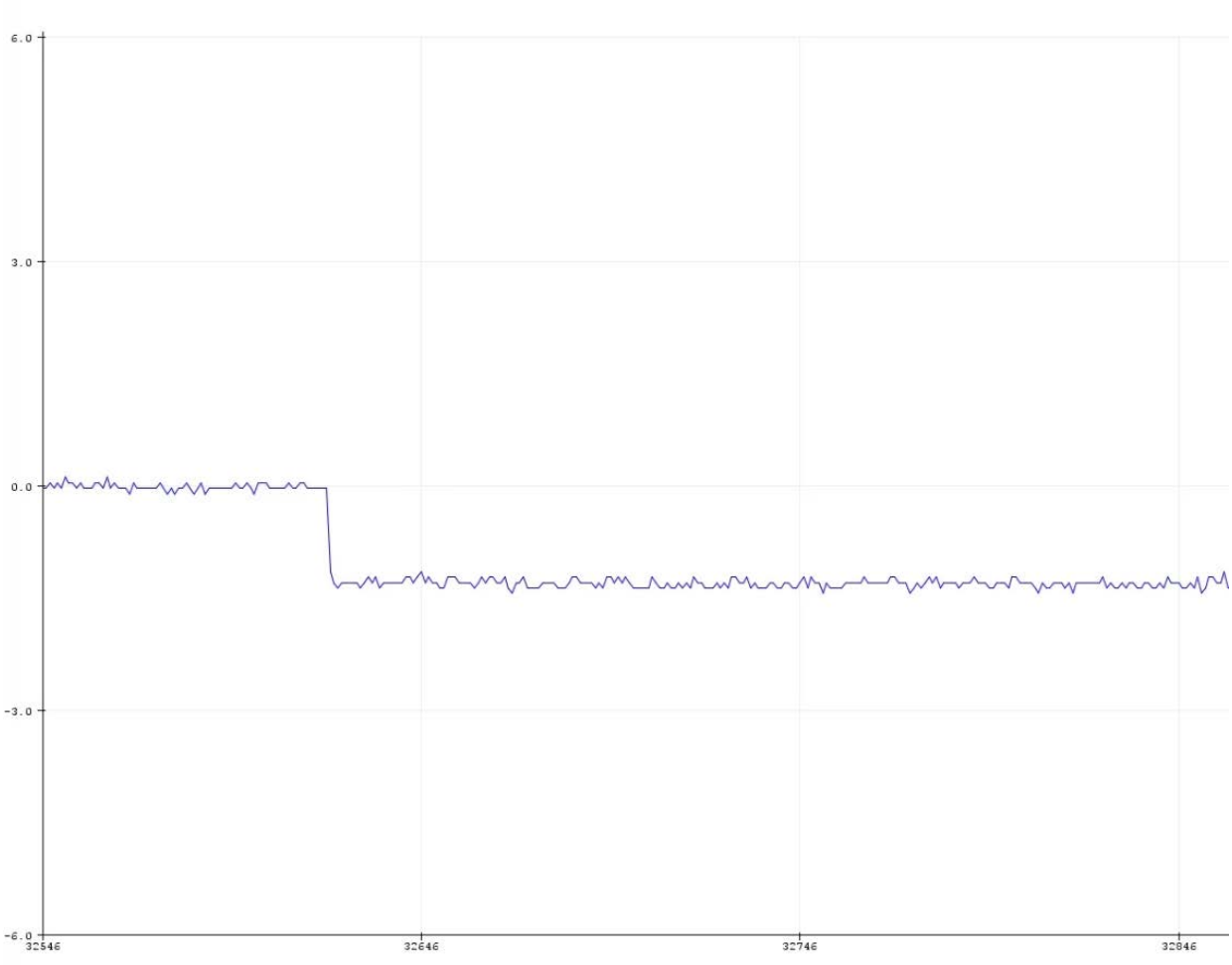

Ilustración 11.2 Intensidad con retención activada

- Motor en marcha:

Cuando el motor se pone en marcha y comienza a girar se observa como la intensidad empieza a realizar un ciclo sinusoidal que va entre 3 y -3 A para realizar los giros del motor. El muestreo de la señal cambia al cambiar la velocidad como se muestra en la Ilustración 11.3, Ilustración 11.4 e Ilustración 11.5, ya que el tiempo de muestreo se reduce al aumentar la velocidad.

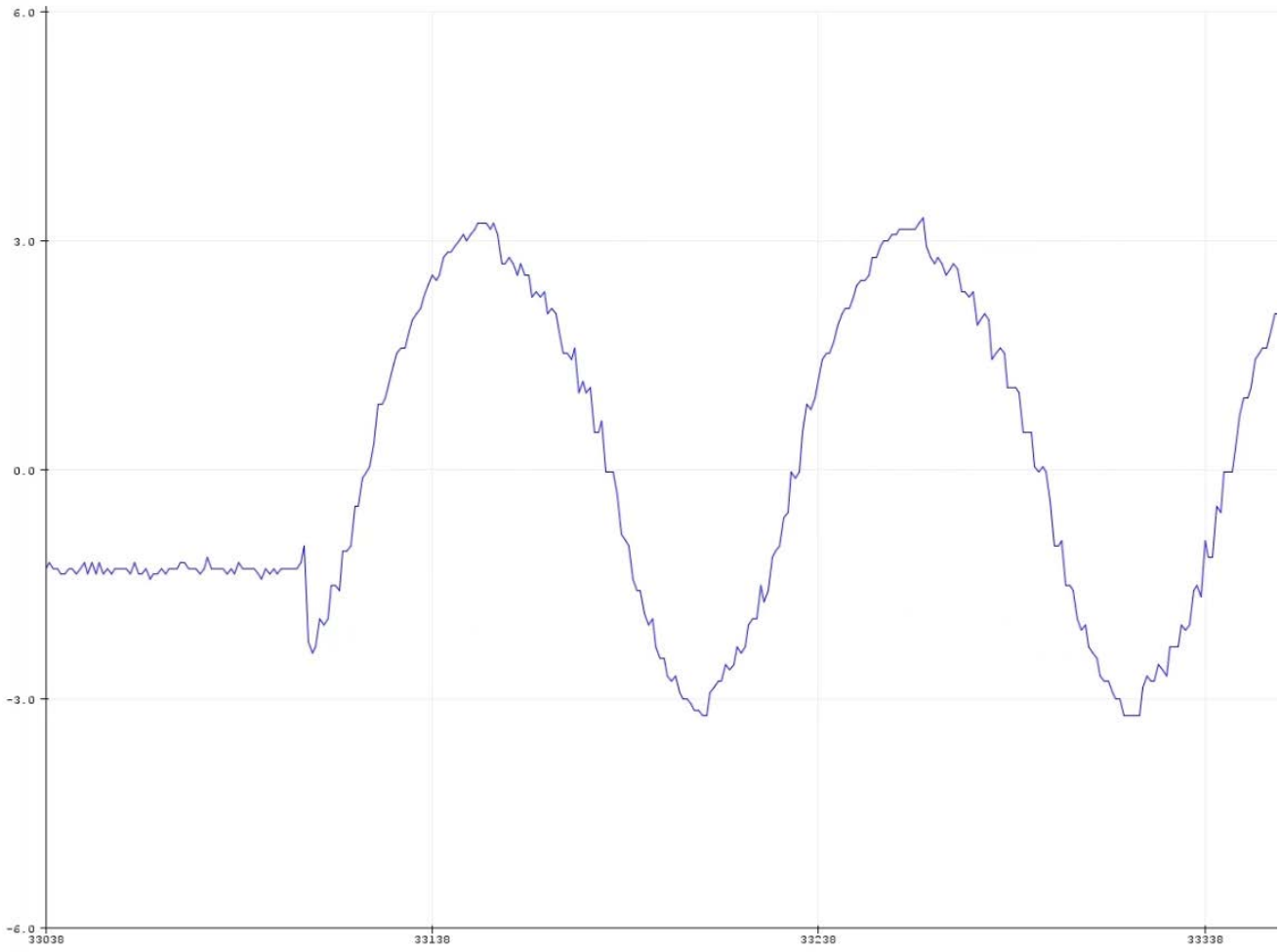

Ilustración 11.3 Intensidad motor girando con velocidad 400

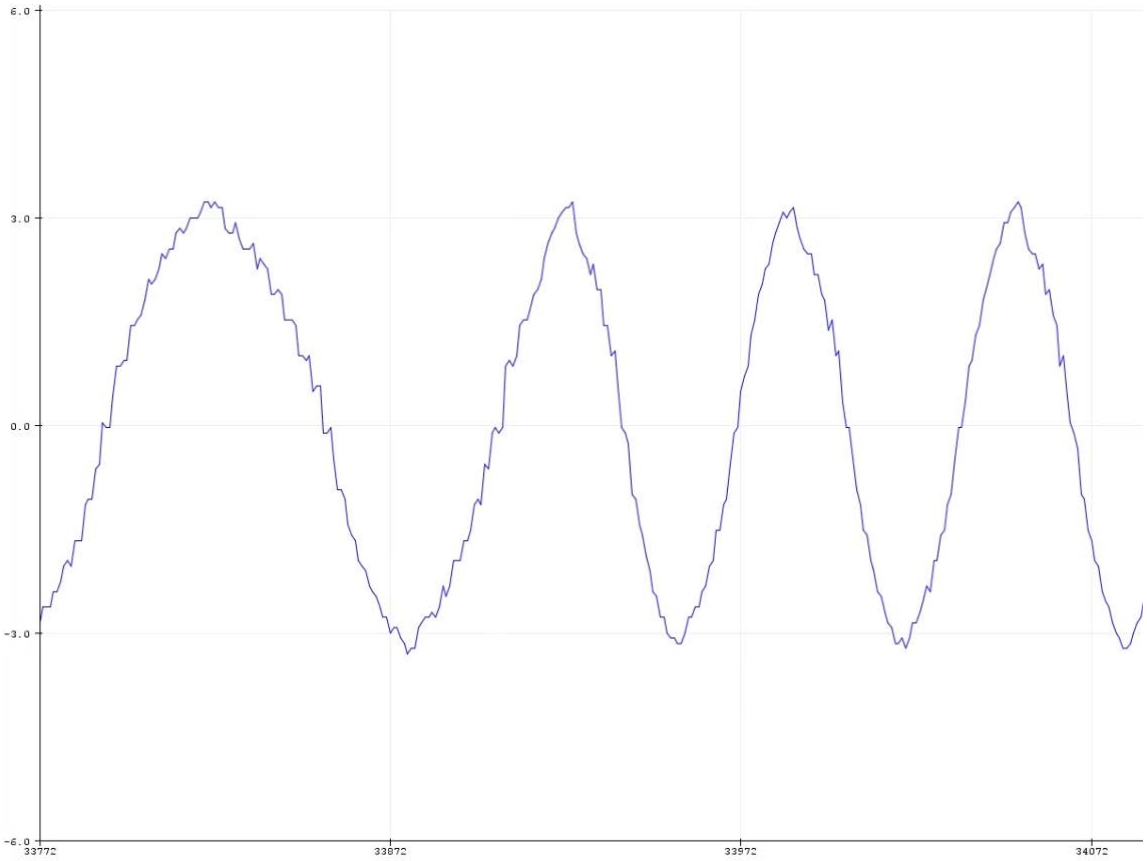

Ilustración 11.4 Intensidad aumentando velocidad a 600

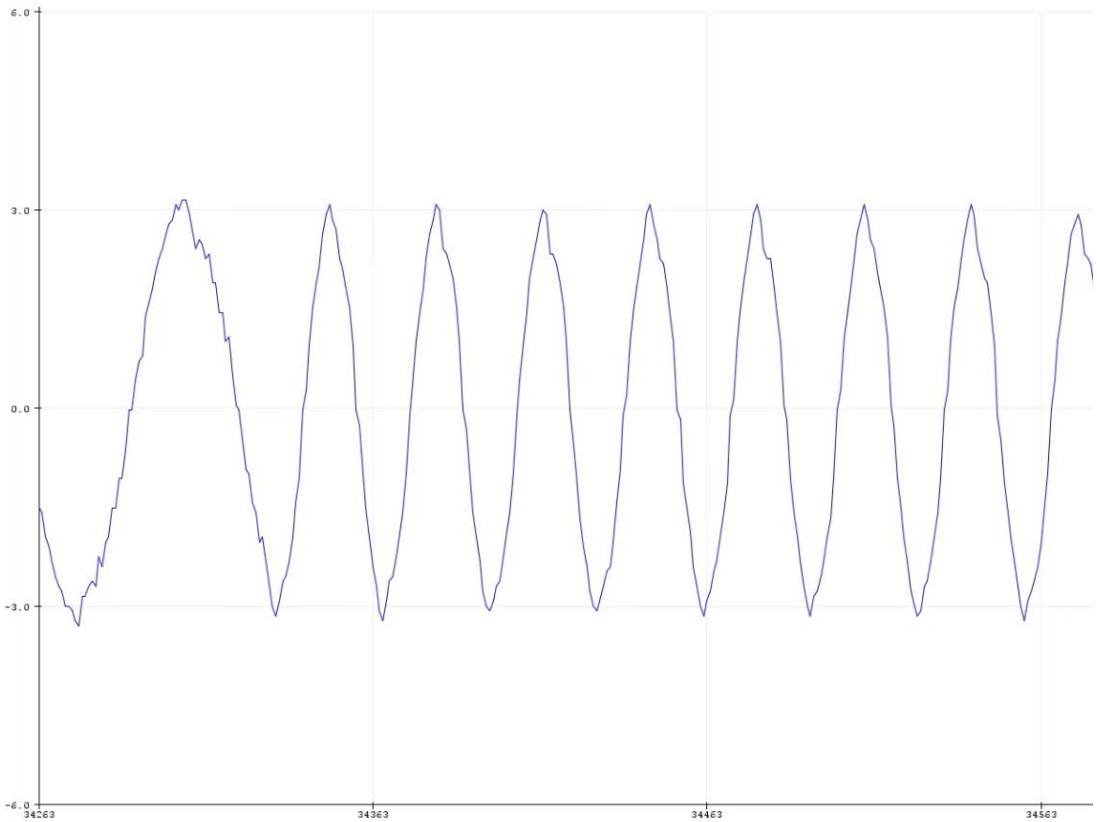

Ilustración 11.5 Intensidad aumento velocidad a 800

12. PRESUPUESTO

En la Tabla 12.1, Tabla 12.2 y Tabla 12.3 se desglosa el presupuesto del proyecto.

12.1 Material

Concepto	Cantidad	Coste total (€)
Motor paso a paso	1	44,73
Fuente alimentación	1	28,29
Driver	1	35,14
Arduino Uno	1	13,95
Arduino Screw Shield	1	7,00
Mando y receptor IR	1	2,99
Sensor de corriente	1	4,49
Sensor de proximidad	1	2,26
Mini placa board	1	1,45
Bloques de terminales	2	0,22
Subtotal		140,52

Tabla 12.1 Tabla de material

12.2 Mano de obra

Tipo de mano de obra	Concepto	Cantidad [horas]	Coste unitario [€/h]	Coste total [€]
Mecánico	Mecanizado caja ensamblaje	2	9,00	18,00
Ingenieril	Programación de código	20	12,00	240,00
Subtotal				258,00

Tabla 12.2 Tabla de mano de obra

12.3 Total

Tipo de coste	Coste total [€]
Material	140,52
Mano de obra	258,00
TOTAL	398,52

Tabla 12.3 Presupuesto total

13. ANEXOS

13.1 Código

```
#include <AccelStepper.h>
#include <IRremote.h>

int receptor = 11; //pin receptor infrarrojos
int led = 13; //pin led del shield
int ENABLE = 5; //pin Enable
int sw = 2; //direccion
int last_sw; //ultima direccion seleccionada
int go = 0; //activar giro del motor
int posRelMov; // posición relativa de movimiento
int final_carrera = 4; //pin del detector de movimiento
int minSpeed = 100; //velocidad minima permitida
int maxSpeed = 1000; //velocidad maxima permitida
int actualSpeed = 400; //velocidad a la que se moverá el motor y que puede variar
int stepSpeed = 100;//saltos de velocidad para botones incremeto y decremento
float sensibilidad = 0.066; //Sensibilidad en Voltios/Ampereo para sensor de 30A
a 66 mV/A
float I; //variable para graficar valor de intensidad
float voltajeSensor; //variable para el muestreo de señal de intensidad
bool medir = false; //medir o dejar de medir
int contEQ = 0; //contador para botón EQ


IRrecv irrecv(receptor); //asignacion de receptor
decode_results codigo; // nombre de la variable que contiene el codigo

AccelStepper stepper(1, 7, 6); //Motor 1, Dir en pin7, Pulse en pin6


void setup() {
	Serial.begin(9600); //puerto serie
	irrecv.enableIRIn(); //Inicia la recepcion
	pinMode(led, OUTPUT); //asignar led como salida
	pinMode(ENABLE, OUTPUT); //variable enable como salida
	pinMode(final_carrera, INPUT); //asignación detector de movimiento como
entrada
	digitalWrite(ENABLE, HIGH); // activación de la retención del motor


	stepper.setMaxSpeed(maxSpeed); //velocidad máxima del motor
	stepper.setSpeed(-400); //Velocidad del motor y sentido de giro


	//Inicio busqueda de "home"

	while (digitalRead(final_carrera) == 1)
		stepper.runSpeed();

	delay(1500);
	stepper.setAcceleration(500);
	stepper.setCurrentPosition(0);
	stepper.moveTo(400);
	stepper.runToPosition();
	stepper.setAcceleration(5000);
```

```
	//Final de movimiento Inicial
}

void loop() {

	if (irrecv.decode(&codigo)) //recpción de código
	{

		switch (codigo.value) {
		case 0xFF22DD: //Botón encender
			go = 1;
			break;

		case 0xFFA25D: //Botón apagar
			go = 0;
			break;

		case 0xFF02FD: // Botón decremento
			go = 0;
			sw = 1;
			last_sw = 2;
			stepper.moveTo(15000);
			break;

		case 0xFFC23D: //Botón incremento
			go = 0;
			sw = 2;
			last_sw = 1;
			stepper.moveTo(400);
			break;

		case 0xFF906F: // Botón +
			if (actualSpeed < maxSpeed)
				actualSpeed += stepSpeed;
			break;

		case 0xFFA857: // Botón -
			if (actualSpeed > minSpeed)
				actualSpeed -= stepSpeed;
			break;

		case 0xFF629D: // Botón mode
			go = 0;
			digitalWrite(ENABLE, HIGH);
			break;

		case 0xFFE21D: // Botón silenciar
			go = 0;
			digitalWrite(ENABLE, LOW);
			break;

		case 0xFFE01F:  // Botón EQ
			contEQ = contEQ + 1;
			if (contEQ == 3)
			{
				Reinicio();
				contEQ = 0;
			}
			break;

		case 0xFF9867: //Botón medir
			medir = true;
```

```
                break;

        case 0xFFB04F: //Botón U/SD
                medir = false;
                break;

        case 0xFF6897: //Botón 0
                posRelMov = 2;
                Slow_Mov();
                break;

        case 0xFF30CF: //Botón 1
                posRelMov = 20;
                Slow_Mov();
                break;

        case 0xFF18E7: //Botón 2
                posRelMov = 30;
                Slow_Mov();
                break;

        case 0xFF7A85: //Botón 3
                posRelMov = 40;
                Slow_Mov();
                break;

        case 0xFF10EF: //Botón 4
                posRelMov = 50;
                Slow_Mov();
                break;

        case 0xFF38C7: //Botón 5
                posRelMov = 60;
                Slow_Mov();
                break;

        case 0xFF5AA5: //Botón 6
                posRelMov = 70;
                Slow_Mov();
                break;

        case 0xFF42BD: //Botón 7
                posRelMov = 80;
                Slow_Mov();
                break;

        case 0xFF4AB5: //Botón 8
                posRelMov = 90;
                Slow_Mov();
                break;

        case 0xFF52AD: //Botón 9
                posRelMov = 100;
                Slow_Mov();
                break;
        }

        irrecv.resume(); //Preparado para recibir siguiente valor
}

MotorOn();
```

```
        if (digitalRead(final_carrera) == 0) {
            go = 0;
               digitalWrite(led, HIGH);
               delay(150);
               digitalWrite(led, LOW);
        }
}

void Slow_Mov() {  //Función de configuración de movimiento lento para las teclas
del 0 al 9
        if (last_sw == 1) {
               posRelMov = posRelMov * (-1);
               stepper.move(posRelMov);
               sw = 1;
        }

        if (last_sw == 2) {
               stepper.move(posRelMov);
               sw = 2;
        }
        actualSpeed = 50;
        go = 1;
}


void MotorOn() { //Función principal para el movimiento del motor
        if (medir == true) {
               float voltajeSensor = analogRead(A0)*(5.0 / 1023.0); //Para la
lectura del sensor
               I = (voltajeSensor - 2.5) / sensibilidad; // Fórmula para obtener
corriente
               Serial.println(I);
        }


        if (sw == 1 && go == 1) { //Giro sentido antihorario
               stepper.setSpeed(actualSpeed);
               stepper.runSpeedToPosition();
               if (stepper.distanceToGo() == 0 || stepper.currentPosition() >
15000) {
                      sw = 0;
               }
        }

        if (sw == 2 && go == 1) { //Giro sentido horario
               stepper.setSpeed(actualSpeed);
               stepper.runSpeedToPosition();
               if (stepper.distanceToGo() == 0 || stepper.currentPosition() < 400)
{
                      sw = 0;
               }
        }
}

void Reinicio() { //Función para botón EQ
        go = 0;
        sw = 2;
        stepper.setSpeed(-400);

        while (digitalRead(final_carrera) == 1)
               stepper.runSpeed();
```

```
        delay(1500);
        stepper.setAcceleration(500);
        stepper.setCurrentPosition(0);
        stepper.moveTo(400);
        stepper.runToPosition();
        stepper.setSpeed(400);
}
```

14. BIBLIOGRAFÍA

Guía Instalación Visual Studio

https://docs.microsoft.com/es-es/visualstudio/install/install-visual-studio?view=vs-2019

Librería AccelStepper

https://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html

Librería IRremote

https://naylampmechatronics.com/blog/36_Tutorial-Arduino-y-control-remoto-Infrarrojo.html

Principios de funcionamiento motor paso a paso

http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/robotica/sistema/motores_p-p.htm

Tutorial Driver

https://descubrearduino.com/tb6600/