UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Scanner 2D para aplicaciones de imágenes sub-THz

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Gómez Pijoan, Miryam

Tutor/a: Ponce Alcántara, Salvador

Cotutor/a: Vidal Rodriguez, Borja

CURSO ACADÉMICO: 2022/2023

RESUMEN:

La espectroscopia de imagen de terahercios es una técnica emergente. La radiación que emplea es capaz de penetrar la mayoría de los materiales no polares o metálicos. Una amplia variedad de materiales dieléctricos que son opacos o altamente dispersores de las frecuencias ópticas son, en cambio, transparentes a las frecuencias de terahercios. Este tipo de medida se puede aplicar para detectar defectos en materiales, cuadros o pinturas.

Actualmente, el escaneo de imágenes de terahercios se realiza con un sistema de posicionamiento manual, lo que dificulta la obtención de movimientos precisos y repetibles. Por ello, existe la necesidad de implementación de una solución para realizar dichas imágenes con un sistema de movimiento automático y de bajo coste.

El presente proyecto consiste en el diseño y programación de un sistema de control para una mesa XY, incluyendo la elección de cada uno de los componentes que forman el sistema: motores, microcontrolador, controlador de motores y el software para la interfaz de usuario. El sistema se ha diseñado y programado específicamente para esta aplicación, integrando todos los componentes en un circuito electrónico sencillo.

El programa diseñado para el microcontrolador se encarga de recibir unos comandos (que se envían a través de la interfaz gráfica), e interpretarlos para realizar el movimiento de los motores requerido por el usuario.

Por otra parte, se ha desarrollado una interfaz gráfica que permite al usuario establecer, desde una aplicación ejecutable en el ordenador, los parámetros de movimiento para que el escáner pueda captar las imágenes correspondientes (distancia a recorrer en los ejes X e Y, tiempo de espera para captar la imagen, y distancia entre medidas). También permite realizar un ajuste manual, para indicar la posición inicial del barrido de dicho escáner.

Se trata pues de un diseño completamente personalizado en cuanto a software y hardware, que proporciona una solución precisa al problema planteado, que es el diseño y la programación del control de movimiento de una mesa XY.

PALABRAS CLAVE:

Microcontrolador, Interfaz de usuario, Controlador, Motor, Circuito, Mesa XY, escáner.

ABSTRACT:

Terahertz imaging spectroscopy is an emerging technique. The radiation it employs is capable of penetrating most nonpolar or metallic materials. A wide variety of dielectric materials that are opaque or highly dispersive at optical frequencies are instead transparent at terahertz frequencies. This type of measurement can be applied to detect defects in materials, pictures or paintings.

Currently, terahertz image scanning is done with a manual positioning system, which makes it difficult to obtain precise and repeatable movements. For this reason, there is a need to implement a solution to scan that images with an automatic and low-cost movement system.

This project consists of the design and programming of a control system for an XY table, it also includes the choice of each of the components that make up the system: motors, microcontroller, motor controller and the software for the user interface. The system has been designed and programmed specifically for this application, integrating all the components in a simple electronic circuit.

The program that has been designed for the microcontroller is in charge of receiving some commands, which are sent through the graphical interface, and interpreting them to carry out the movement of the motors required by the user.

On the other hand, a graphic interface has been developed that allows the user to establish from an executable application on the computer the movement parameters so that the scanner can capture the corresponding images (distance to travel in the X and Y axes, waiting time for capture the image and distance between measurements), as well as perform a manual adjustment to indicate the initial position of the sweep of the scanner.

It is therefore a completely customized design in terms of software and hardware that provides a precise solution to the problem that is the design and programming of the movement control of an XY table.

KEYWORDS:

Microcontroller, User interface, Controller, Motor, Circuit, XY Table, Scanner.

RESUM:

L'espectroscòpia d'imatge de terahercis és una tècnica emergent. La radiació que empra és capaç de penetrar la majoria dels materials no polars o metàl·lics. Una àmplia varietat de materials dielèctrics que són opacs o altament dispersors de les freqüències òptiques són, en canvi, transparents a les freqüències de terahercis. Aquest tipus de mesura es pot aplicar per detectar defectes en materials, quadres o pintures.

Actualment, l'escaneig d'imatges de terahercis es realitza amb un sistema de posicionament manual, el que dificulta l'obtenció de moviments precisos i repetibles. Per això, existeix la necessitat d'implementació d'una solució per a escanejar aquestes imatges amb un sistema de moviment automàtic i de baix cost.

El present projecte consisteix en el disseny y programació d'un sistema de control per a una taula XY, també inclou l'elecció de cadascun dels components que formen el sistema: motors, microcontrolador, controlador de motors y el software per a la interfície d'usuari. El sistema s'ha dissenyat i programat específicament per a aquesta aplicació, integrant tots els components en un circuit electrònic senzill.

El programa dissenyat per al microcontrolador s'encarrega de rebre uns comandos (que s'envien a través de la interfície d'usuari) i interpretats per a realitzar el moviment dels motors requerits per l'usuari.

Per altra banda, s'ha desenvolupat una interfície gràfica que permet a l'usuari establir des d'una aplicació executable en l'ordinador els paràmetres de moviment per a que l'escàner puga captar les imatges corresponents (distància a recórrer en el eixos X i Y, temps d'espera per a captar la imatge i distància entre mesures), així com realitzar un ajustament manual per a indicar la posició inicial del recorregut de aquest escàner.

Es tracta doncs d'un disseny completament personalitzat en quant a software y hardware que proporciona una solució precisa al problema plantejat que es el disseny i programació del control de moviment d'una taula XY.

PARAULES CLAU:

Microcontrolador, Interfície d'usuari, Controlador, Motor, Circuit, Taula XY, escàner.

ÍNDICE GENERAL

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ESCÁNER 2D PARA APLICACIONES DE IMÁGENES SUB-THZ

Documento 01: Memoria

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Miryam Gómez Pijoan

Tutor/a: Salvador Ponce Alcántara

Cotutor/a: Borja Vidal Rodríguez

CURSO ACADÉMICO: 2022/2023

ÍNDICE MEMORIA

Capítulo 1: Objetivo

El dominio de los terahercios es un sector en crecimiento. Los sistemas de terahercios son generalmente costosos, lo que limita su uso a los laboratorios de investigación académica o al sector industrial de alto nivel.

El escáner que incorporará el sistema diseñado tiene como objetivo proporcionar una integración de terahercios de código abierto para reducir el coste en un factor de diez a cincuenta veces en comparación con los sistemas comerciales. El dominio de los terahercios es un sector en crecimiento, pero los sistemas de terahercios son generalmente costosos, lo que limita su uso a los laboratorios de investigación académica o al sector industrial de alto nivel.

En particular, los sistemas de medición de terahercios muestran un creciente interés en varios campos, como la inspección de materiales compuestos para la industria aeroespacial, para el arte y conservación del patrimonio y para el control de alimentos o desarrollos biomédicos [1].

El escáner se centra en la implementación y configuración de unidades de radar integradas, enfocándose en distancias de trabajo cortas (menos de 20 cm) para varios campos de aplicación. Al configurar el radar con una rampa de frecuencia rápida, se desvía de su uso inicial de campo lejano, y se puede implementar como parte de un sistema de medición de corto alcance o una unidad de imágenes de terahercios.

Este escáner está compuesto por dos lentes, que forman el haz para obtener imágenes puntuales a una distancia fija, normalmente de 50 mm [1]. Un ordenador controla el movimiento de escaneo y adquiere la señal procedente del sensor. La muestra se coloca horizontalmente sobre una mesa, usando pies ajustables para evaluar su distancia al cabezal de medición.

La unidad de radar FMCW (del inglés Frequency-Modulated Continuous Wave radar) integrada utilizada en este escáner incluye tanto el transmisor y el detector como el procesamiento en una placa electrónica de 3x3 cm [1]. Este radar opera en la banda de frecuencia de 122-123 GHz, y se puede ampliar de 119 a 126 GHz. El transceptor consta de antenas recepción/transmisión de parche cuádruple ensambladas en una sola antena. Este diseño frontal proporciona la ventaja no despreciable de tener las antenas recepción y transmisión cerca del mismo punto focal, casi equivalente a un radar monoestático en la escala de la configuración óptica, sin dejar de ser biestático.

Los radares monoestáticos son aquellos sistemas donde el transmisor y el receptor se encuentran situados en la misma localización mientras que los radares biestáticos son aquellos donde el transmisor y el receptor se encuentran en localizaciones diferentes [2].

La comunicación de este transceptor de radar se puede realizar directamente a través de un puerto serie, o de forma remota a través de una conexión WI-FI. Para obtener una configuración de alta resolución y velocidad optimizada a distancias de trabajo pequeñas, se sugiere un conjunto de parámetros para cubrir la banda de frecuencia más grande posible, y ofrecer la tasa de muestreo más alta. Sin embargo, para evitar el desbordamiento del procesamiento de banda base, no es factible maximizar todos los parámetros.

La unidad de movimiento realiza un escaneo de trama del objeto, que genera un archivo de bits, que contiene los datos, con valores individuales de la señal de escaneo de profundidad registrada para cada píxel escaneado. El encabezado de este archivo contiene la información de exploración de parámetros (el tamaño de la imagen a lo largo de los ejes X e Y en píxeles, el tamaño de paso de exploración de trama, el número de capas consideradas), necesaria para el procesamiento de los datos registrados [1].

Por lo tanto, el presente documento tiene por objetivo desarrollar una unidad de movimiento económica basada en una mesa XY, la cual estará dirigida por un microcontrolador. El tamaño total que debe alcanzar el área medible tiene que ser de, como mínimo, 300x300 mm.

El sistema de control de movimiento de la mesa debe realizarse mediante la programación de los dos motores. Además, se deberá de hacer uso de un software para crear una interfaz de usuario que sea sencilla de utilizar. De esta forma, debe lograrse una comunicación efectiva entre el hardware y el software para permitir el control y movimiento de la mesa de manera intuitiva y precisa.

Acoplando sobre la mesa cualquier tipo de escáner y mediante la interfaz, el usuario tiene que poder realizar las siguientes acciones:

- Indicar la distancia a desplazarse en los ejes.
- Introducir la separación entre medidas.
- Fijar el tiempo de espera para que el equipo de medida sea capaz de captar la imagen correspondiente.

El sistema propuesto en este Trabajo de Fin de Grado (TFG) requerirá de una fuente de alimentación necesaria para alimentar los motores a través de un controlador. Un microcontrolador se encargará de gestionar los comandos recibidos, y realizar la acción correspondiente.

Así pues, será objeto de la solución lo incumbente al diseño electrónico y a la programación tanto del microcontrolador como de la interfaz de usuario.

Capítulo 2: Planteamiento de soluciones alternativas y justificación de la solución adoptada

Con tal de poder determinar la mejor solución al problema planteado, cumpliendo con las especificaciones requeridas, se ha diseñado el sistema que se muestra en el diagrama de bloques de la ilustración 1.

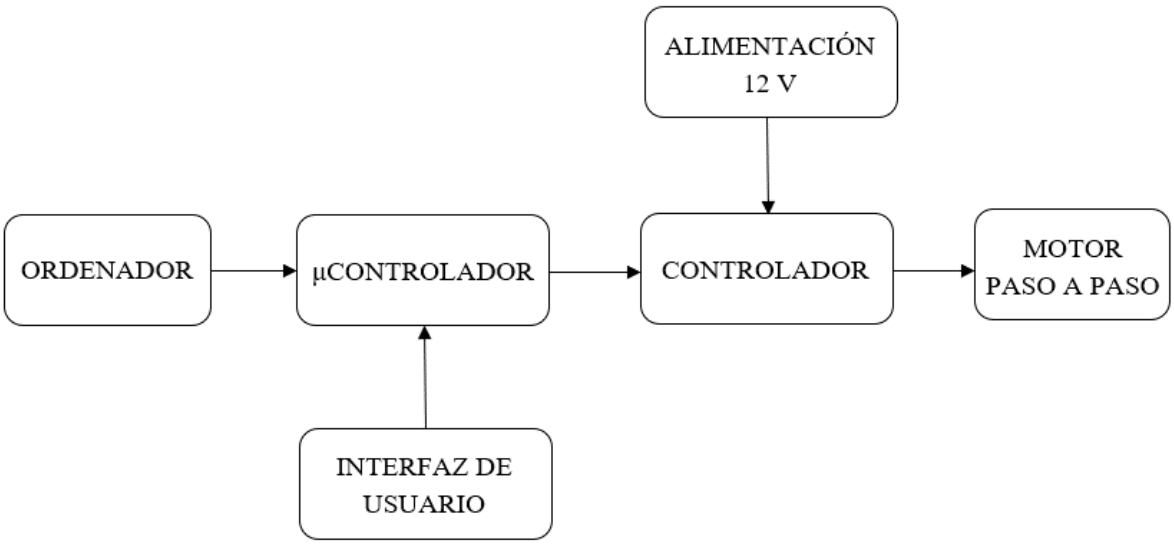


Ilustración 1. Diagrama de bloques del sistema.

Se plantea a continuación una descripción detallada de las diferentes soluciones alternativas de cada bloque exponiendo sus ventajas y desventajas para así poder justificar la solución escogida.

2.1. Mesa XY

Las mesas XY (ilustración 2), son mesas de posicionamiento manuales o motorizadas que permiten movimientos lineales basados en cojinetes impulsados por un mecanismo de accionamiento. El accionamiento suele ser proporcionado por un motor lineal o un volante manual [3].

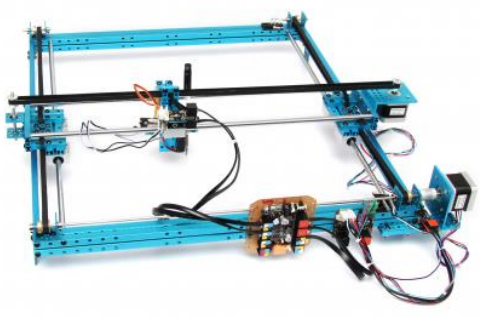


Ilustración 2. Mesa XY [4].

Estas mesas, cuentan con una larga historia puesto que han sido utilizadas en una amplia variedad de aplicaciones a lo largo del tiempo.

Los primeros indicios de mesas XY se remontan al siglo XIX, cuando se crearon mecanismos de desplazamiento bidimensionales para su uso en laboratorios y otros entornos científicos. Estas mesas permitían el posicionamiento preciso de objetos o muestras para realizar mediciones y experimentos [5].

A finales del siglo XIX y principios del XX, se desarrollaron mesas XY para aplicaciones de trazado y dibujo técnico. Estos sistemas permiten crear dibujos precisos y completos para su uso en ingeniería, arquitectura y diseño.

La sofisticación de las mesas XY aumentó como resultado del avance tecnológico, particularmente en electrónica e informática, integrándose así con sistemas de control automatizados. Esto hizo posible su uso en entornos industriales como producción, ensamblaje y procesamiento de materiales.

A medida que la industria de semiconductores se desarrolló, las mesas XY se convirtieron en componentes clave en la fabricación de chips. De esta forma, estas mesas se utilizaban para posicionar con precisión las obleas de silicio y permitir el depósito de capas y la realización de patrones microscópicos en el proceso de fabricación.

Asimismo, en la industria de la impresión, las mesas XY también se han utilizado ampliamente, tanto en impresión gráfica como en impresión en 3D. Estas mesas permiten un posicionamiento preciso del cabezal de impresión, lo que resulta en una mayor calidad y precisión en la reproducción de imágenes y objetos tridimensionales [5].

Las mesas XY tienen un amplio ámbito de aplicación en diversas áreas:

- **Automatización industrial**: El control preciso y programado de una mesa XY puede ser utilizado en la automatización de procesos industriales, como ensamblaje de componentes, manipulación de materiales, pruebas de calidad o producción de dispositivos electrónicos [6].

- **Impresión 3D**: El proyecto podría utilizarse en impresoras 3D, donde la mesa XY se utiliza para posicionar con precisión la plataforma de impresión, lo que le permite la producción de objetos tridimensionales con alta calidad y detalle [7].

- **Grabado y corte láser**: Una mesa XY controlada programada es necesaria en la industria del grabado y corte por láser para posicionar con precisión los elementos o materiales que se van a grabar o cortar. Esto permite realizar tareas de corte y grabado con un alto grado de precisión y repetibilidad [8].
- **Máquinas de enrutador CNC**: En las máquinas de enrutador CNC (control numérico por computadora), las mesas XY programadas se pueden usar para mover con precisión la mesa, lo que permite el mecanizado de materiales como madera, plástico o metal [9].

- **Investigación científica**: Una mesa XY programada se puede utilizar en laboratorios para el posicionamiento preciso de muestras en experimentos, análisis de imágenes, microscopía o estudios de materiales [10].

2.1.1. Solución adoptada

Puesto que no se encontró ninguna otra alternativa que fuese válida para esta aplicación, se escogió el modelo XY-Plotter proporcionado por la casa makeblock [11].

Este paquete contiene todo lo necesario para el montaje, incluyendo los cuatro finales de carrera y dos motores paso a paso modelo 42BYG. Además, contaba con un manual de montaje cosa que la hace sencilla de montar para el usuario. La mesa XY comprada tiene una longitud de trabajo adecuada para el sistema requerido, la cual es de 356 mm en el eje X y de 346 mm en el eje Y.

2.2. Alternativas de motor

En el presente apartado se van a analizar dos alternativas de motor que son: el motor de corriente continua (DC) y el motor paso a paso.

2.2.1. Motor paso a paso

Un motor paso a paso es un dispositivo electromecánico que convierte una serie de pulsos eléctricos en desplazamientos angulares, lo que significa que es capaz de girar una cantidad de grados (paso, medio paso o micropaso), dependiendo de sus entradas de control [12]. Esta característica los hace ideales para mecanismos donde se necesitan movimientos muy precisos.

Consiste en un rotor que gira en pasos discretos, en lugar de girar de manera continua como lo hace un motor de corriente continua. Tal y como aparece en la ilustración 3, este motor cuenta con las siguientes partes: estator, rotor, eje y controlador.

El estator es la parte fija del motor y está compuesto por bobinas de alambre enrolladas alrededor de un núcleo ferromagnético. Estas bobinas se activan secuencialmente para crear un campo magnético rotativo.

El rotor es la parte móvil del motor. Está compuesto por un imán permanente o una serie de imanes permanentes, que están dispuestos en un patrón específico para girar siguiendo el campo magnético creado alrededor del estator.

El eje del motor se conecta al rotor y es la parte que gira. La posición y la velocidad del eje se controlan mediante la activación secuencial de las bobinas del estator.

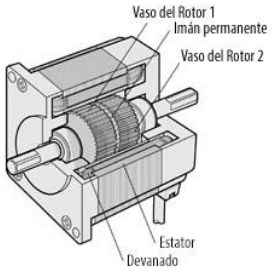


Ilustración 3. Esquema partes motor paso a paso [13].

El controlador del motor paso a paso es un circuito formado por un driver y los componentes necesarios para así suministrar la corriente adecuada al motor, para que el rotor gire en los pasos deseados. El controlador se encarga de cambiar la secuencia de activación de las bobinas para girar el motor en la dirección y el número de pasos necesarios [14].

Los motores paso a paso pueden dividirse en tres tipos: de reluctancia variable, de imán permanente e híbridos.

- **Reluctancia variable**:

  Los motores de reluctancia variable no utilizan un campo magnético permanente, por tanto, pueden moverse sin limitaciones. Este tipo de montaje es el menos común y se usa, generalmente, en aplicaciones que no requieren un alto grado de par de fuerza, como puede ser el posicionamiento de un mando de desplazamiento.

  Se desarrolló con objeto de poder conseguir unos desplazamientos angulares reducidos, sin que por este motivo haya de aumentarse considerablemente el número de bobinados. El estator presentará la forma cilíndrica habitual conteniendo generalmente un total de tres devanados distribuidos de tal forma que existirá un ángulo de 120° aproximadamente entre dos de ellos [12]. En la ilustración 4 se muestra el esquema interno de estos motores.

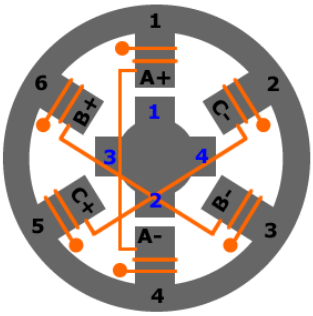


Ilustración 4. Motor paso a paso con reluctancia variable [15].

- **Imán permanente**: Estos motores pueden ser unipolares o bipolares.

Los motores unipolares suelen tener 5 o 6 cables de salida dependiendo de su conexionado interno. Suelen ser 4 cables por los cuales se recibe los pulsos que indican la secuencia y duración de los pasos y los restantes sirven como alimentación del motor. Se caracterizan por ser los más simples de controlar. Para este tipo de motores existen tres secuencias diferentes para manejarlo:

- Secuencia normal: el motor siempre avanza un paso por vez debido a que siempre existen 2 bobinas activadas, con esta secuencia se obtiene un alto torque de paso y retención.

- Secuencia de paso completo: se activa solo una bobina por vez, lo que ocasiona que el eje del motor gire hacia la bobina activa. En algunos motores esto brinda un funcionamiento más suave, pero el torque de paso y retención es menor.

- Secuencia de medio paso: se activan las bobinas de tal manera que se combinan las secuencias anteriores, el resultado que se obtiene es un paso más corto (la mitad del paso).

Los motores bipolares por lo general tienen 4 cables de salida. Para realizar un movimiento, es necesario un puente H por cada bobina del motor por lo que para controlar un motor paso a paso con dos bobinas, será necesario usar dos puentes H. Esto puede hacer que la tarjeta controladora se vuelva más compleja y costosa. Su uso no es tan común.

Estos motores requieren de la inversión de la corriente que circula por sus bobinas en una secuencia determinada, esta secuencia determina el sentido de giro del motor. Cada inversión de polaridad provoca el movimiento del eje en un paso [12].

En la ilustración 5 se puede observar el esquema interno de un motor paso a paso de imanes permanentes.

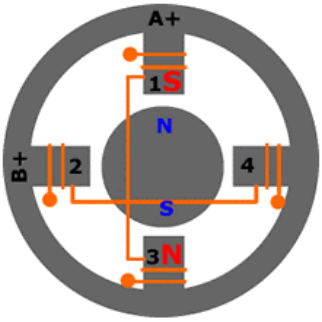


Ilustración 5. Motor paso a paso de imanes permanentes [15].

- **Híbridos**:

  Es un motor eléctrico del tipo paso a paso, cuyo funcionamiento se basa en la combinación de los otros dos tipos de motores [12]. El esquema interno de este tipo de motores puede visualizarse en la ilustración 6.

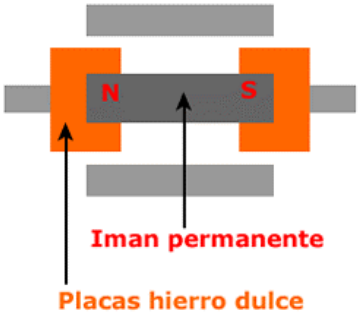


Ilustración 6. Motor paso a paso híbrido [15].

2.2.2. Motor de corriente continua (DC)

Un motor de corriente continua (también llamado motor de corriente directa, motor CC o motor DC) es una máquina que convierte energía eléctrica en energía mecánica, provocando un movimiento giratorio, gracias a la acción de un campo magnético [16].

Tal y como aparece en la ilustración 7, los componentes de un motor de corriente continua son:

- Estator: parte que da soporte mecánico al aparato y contiene los polos de la máquina, que pueden ser devanados de hilo de cobre sobre un núcleo de hierro o imanes permanentes [16].

- Rotor: es un componente generalmente de forma cilíndrica, también devanado y con núcleo, alimentado con corriente continua a través del colector formado por delgas. Las delgas se fabrican generalmente de cobre y están en contacto alternante con las escobillas fijas [16].

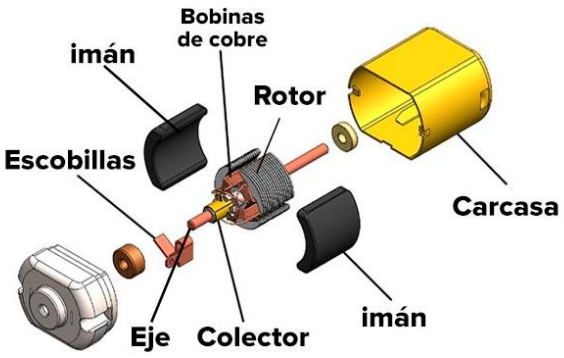


Ilustración 7. Partes de un motor de corriente continua [17].

El sentido de giro de estos motores puede ajustarse fácilmente mediante la polaridad de los dos terminales de alimentación conectados al motor DC. Así, por ejemplo, +12V para la rotación en el sentido de las agujas del reloj, y -12V para la rotación en sentido contrario [18]. Se muestra un esquema del funcionamiento en la ilustración 8.

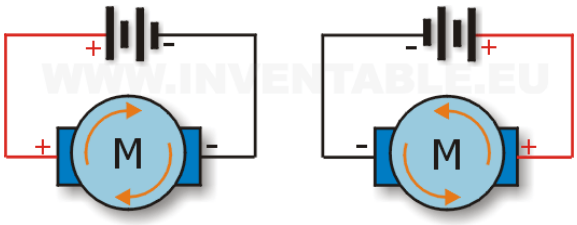


Ilustración 8. Sentido de giro del motor en función de la polaridad [19].

La velocidad a la que gira el motor puede modificarse a través de la tensión del motor de corriente continua. Sin embargo, para controlar la velocidad correctamente se necesita un sensor o codificador. También existen métodos especiales para calcular la velocidad a partir de la corriente del motor. Se puede utilizar un controlador PI (controlador proporcional e integral) como regulador, que ajusta la tensión en función de la desviación entre la consigna y la velocidad real. El nivel de tensión ajustado de este modo modifica la corriente continua que circula por el motor, de esta manera, la velocidad del motor de CC se controla a la velocidad establecida [18].

Hay cuatro tipos de motores de corriente continua según la forma de conexión de las bobinas inductoras e inducidas entre sí.

- **Motor serie**: los devanados del inductor y del inducido se encuentran en serie (ilustración 9).

La conexión forma un circuito en serie en el que la intensidad absorbida por el motor al conectarlo a la red (también llamada corriente de carga) es la misma, tanto para la bobina conductora (del estator) como para la bobina inducida (del rotor) [20].

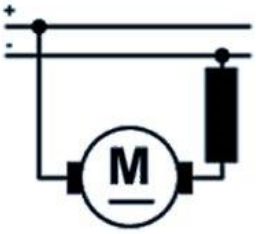


Ilustración 9. Motor de corriente continua en serie [21].

- **Motor en derivación o motor Shunt**: los devanados inductor e inducido se encuentran en paralelo (ilustración 10).

De este modo, de toda la corriente absorbida por el motor, una parte circula por las bobinas inducidas y la otra por la inductoras. El circuito de excitación (inductor) está a la misma tensión que el inductor [20].

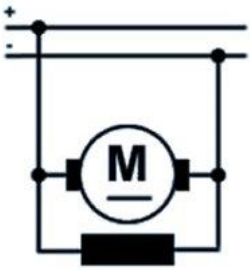


Ilustración 10. Motor de corriente continua tipo Shunt [22].

- **Motor Compound**: consta de dos devanados inductores, uno está en serie con el devanado inducido y el otro en paralelo (ilustración 11), por lo que se puede decir que es una combinación del motor serie y el motor shunt.

Una parte de la intensidad de corriente absorbida circula por las bobinas inducidas y, por ende, por una de las inductoras, mientras que el resto de la corriente recorre la otra bobina inductora.

Se caracteriza por tener un elevado par de arranque, pero no corre el peligro de ser inestable cuando trabaja en vacío, como ocurre con el motor serie, aunque puede llegar a alcanzar un número de revoluciones muy alto [20].

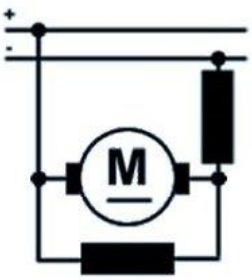


Ilustración 11. Motor de corriente continua tipo Compound [23].

2.2.3. Comparativa y solución adoptada

Con la finalidad de poder seleccionar la mejor opción entre las alternativas planteadas, se va a realizar una comparativa entre ambas exponiendo sus ventajas y desventajas.

- ***Motor paso a paso***
  - Ventajas [14]:
    - Funcionamiento preciso.
    - Fácil control de la posición del rotor y de su velocidad de rotación.
    - El par motor es muy alto a baja velocidad.

- No hay escobillas en la construcción del motor, lo que conlleva en una alta durabilidad mecánica y una mayor fiabilidad.
- Se puede controlar fácilmente el motor debido a que tiene un arranque rápido gracias a un par elevado, parada fácil gracias al par de retención elevado y la capacidad de cambiar rápidamente el sentido de giro.
- Facilidad para configurar las características de inicio y parada.

- Desventajas [14]:
  - Gran requerimiento de energía, dado que el motor necesita energía tanto cuando está en movimiento como cuando está parado.
  - El par motor es mayor a velocidades de giro relativamente bajas, y disminuye a altas velocidades de giro.
  - Es imposible obtener una alta velocidad de rotación mientras se mantiene el par y la capacidad del motor para soportar la carga establecida.

➢ ***Motor de corriente continua***

- Ventajas [24]:
  - El par de giro de arranque es alto y varía de forma lineal con la velocidad, además este par de giro es proporcional a la intensidad porque, a mayor intensidad, mayor flujo magnético.
  - Se pueden alimentar con energía almacenada en baterías o mediante energía fotovoltaica.
  - La regulación de la velocidad es sencilla y económica.
  - Se pueden usar con reductoras para multitud de aplicaciones de tal forma que consiguen tener más fuerza de giro.
  - Sólo llevan dos cables.
  - Pueden trabajar a bajas velocidades.

- Desventajas [24]:
  - Si la energía proviene de la red estándar de corriente alterna, para funcionar necesitará una fuente de alimentación que pase de alterna a continua.
  - El voltaje de trabajo es bajo por lo que para determinada potencia necesitan cableado más grueso.
  - Vida útil limitada debido al desgaste por fricción de las escobillas, lo que también limita la velocidad de los motores [18].

Atendiendo a todo lo anterior, se ha decidido utilizar para esta aplicación un motor paso a paso puesto que se necesita precisión para realizar el movimiento y facilidad de control. Además, estos motores tienen mayor tiempo de uso, lo que conlleva un ahorro de dinero ya que no será necesario reemplazar los motores con frecuencia.

Para el movimiento de la mesa en los ejes se ha elegido el motor paso a paso 42BYG de la casa *makeblock* (ilustración 12) porque además de tener buenas prestaciones, este motor viene incluido en el paquete de la mesa XY comprado.

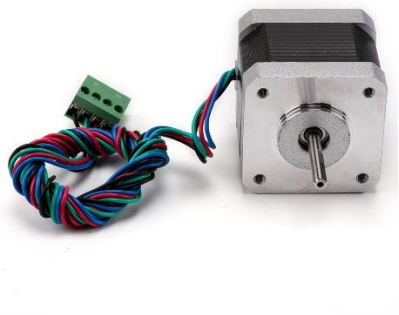


Ilustración 12. Motor paso a paso 42BYG [25].

2.3. Alternativas de controlador

Todo motor eléctrico, independientemente de su tamaño o propósito, requiere algún tipo de mecanismo de control. Un controlador es un amplificador de corriente cuya función es tomar una señal de control de baja corriente y convertirla en una señal de alta corriente que pueda alimentar al motor [26].

El controlador de motor más simple es un interruptor de encendido y apagado ordinario que conecta el motor a la fuente de alimentación. Este interruptor puede ser manual o un relé conectado a un sensor automático con el fin de poder arrancar o detener el motor. En otra escala, puede haber controladores de motor muy complejos, que controlen muchos motores junto con una amplia variedad de sensores, con el fin de proporcionar elementos de lógica para el funcionamiento de estos motores.

Los controladores de motor pueden realizar muchas funciones, tales como arrancar o detener un motor eléctrico de forma automática o manual, establecer su velocidad, dirección o controlar el par si es necesario [27].

Existen diferentes tipos de controladores:

- **Controlador manual** [27]:

Los controladores manuales son generalmente dispositivos muy simples que conectan el motor directamente a la conexión eléctrica. Un controlador manual puede ser simplemente un interruptor conectado en serie con un motor. En los controladores manuales, los operadores deben ir físicamente al controlador para activarlo. La protección contra sobrecargas o la desconexión por bajo voltaje pueden o no estar provistas en un controlador manual.

- **Controlador semiautomático** [27]:

Un controlador semiautomático es un dispositivo mucho más complejo y puede tener varios botones, interruptores y sensores para controlar el funcionamiento de un arrancador de motor. El arrancador está conectado al motor, que está conectado a un panel de control ubicado lejos del arrancador y del motor. Sin embargo, el operador debe iniciar todas las acciones, tales como arrancar y parar el motor, pero se puede hacer de forma remota.

- **Controlador automático** [27]:

Un controlador automático es un dispositivo sofisticado, de modo que una vez que el operador configura los parámetros de funcionamiento, el controlador controla automáticamente el funcionamiento del motor.

A continuación, se van a analizar dos alternativas de controlador que son: el Pololu Driver DRV8825 y el Microstep Driver TB6600.

2.3.1. Pololu Driver DRV8825

Este controlador de motores paso a paso bipolares está basado en el chip DRV8825 de Texas Instruments y permite controlar motores de hasta 1.5 A por canal (2.2 A de máximo con suficiente ventilación) [28].

El Pololu Driver DRV8825 (ilustración 13) tiene limitación de corriente ajustable, protección contra sobre corriente, y cinco resoluciones diferentes de microstepping que son 1/2, 1/4, 1/8, 1/16 y 1/32. Funciona desde 8 V a 45 V, y puede suministrar hasta 2.2 A por bobina utilizando ventilación forzada y/o un disipador [2828].

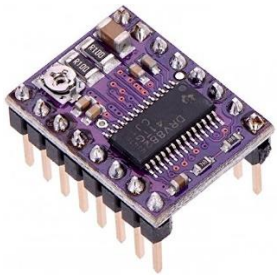


Ilustración 13. Pololu Driver DRV8825 [29].

Al tener una resolución hasta 1/32 de paso, los 3 pines Modo, permiten configurar los micropasos del motor, según la tabla 1.

M0	M1	M2	Resolución - Micropasos
Low	Low	Low	Paso completo
High	Low	Low	Medio Paso
Low	High	Low	1/4 de paso
High	High	Low	1/8 de paso
Low	Low	High	1/16 de paso
High	Low	High	1/32 de paso
Low	High	High	1/32 de paso
High	High	High	1/32 de paso

Tabla 1. Modos de funcionamiento del Pololu Driver DRV 8825.

En la ilustración 14 se muestran los diferentes pines con los que cuenta este driver y cómo se conectarían a un motor y a un microcontrolador.

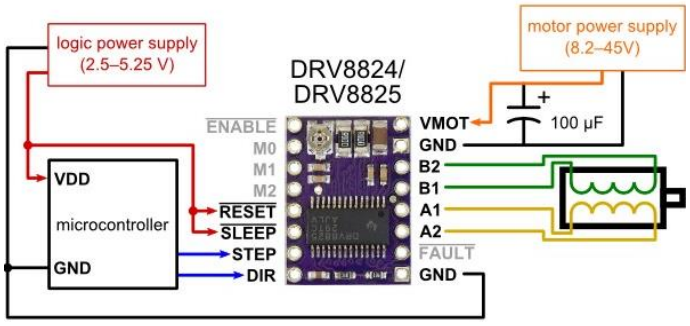


Ilustración 14. Conexionado del driver Pololu DRV8825 [33].

2.3.2. Microstep Driver TB6600

Este controlador de motores paso a paso bipolares está basado en el chip TB6600 y permite controlar motores de hasta 3.5 A por canal (picos de 4 A como máximo). Además, cuenta con limitación de corriente ajustable, protección contra sobre corriente, y 7 resoluciones diferentes de microstepping [30].

El TB6600 (ilustración 15) es un controlador profesional para motores paso a paso bipolares. Es compatible con microcontroladores como Arduino y otros que puedan generar señales de pulsos de 5 V. Por otro lado, soporta una gran variedad de voltajes de entrada, de 9 a 42 V DC. Es capaz de proporcionar hasta 3.5 A de corriente de forma continuada y 4 A de pico por cortos periodos de tiempo. De esta forma puede controlar una gran variedad de motores [30].

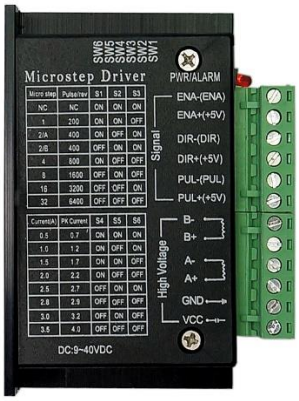


Ilustración 15. Microstep Driver TB6600 [31].

El controlador soporta el control de dirección y de paso como en todos los controladores de este tipo. También puede configurarse para microstepping mediante unos microinterruptores incluidos. Hay 7 valores posibles además de 8 posiciones para el ajuste de corriente. Todas las señales están protegidas internamente mediante optoacopladores de alta velocidad para evitar interferencias y mejorar el aislamiento del circuito de control. Viene ensamblado en una caja de metal para una medir refrigeración [30].

En la ilustración 16 se muestran los diferentes pines con los que cuenta este driver y como se conectarían a un motor y a un microcontrolador.

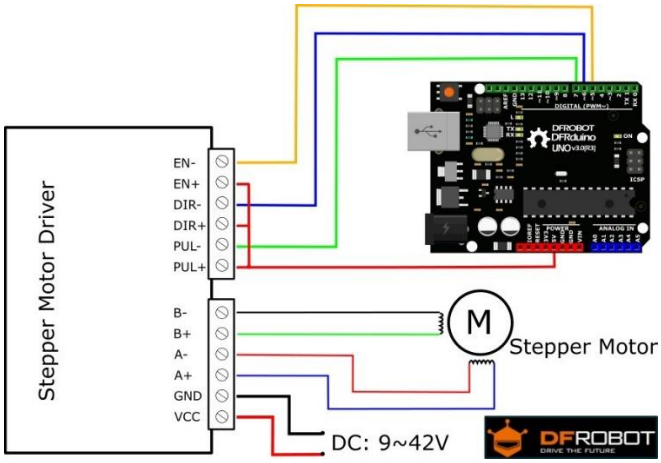


Ilustración 16. Conexionado del Microstep Driver TB6600 [32].

2.3.3. Comparativa y solución adoptada

Con la finalidad de poder seleccionar la mejor opción entre las alternativas planteadas, se va a realizar una comparativa entre ambas exponiendo sus ventajas y desventajas.

- ***Pololu Driver DRV8825***
  - Ventajas [33]:
    - Cuenta con un nivel sonoro bajo.
    - Es eficiente térmicamente.
    - Tiene una alta resolución (micropasos 1/32).
    - La corriente se puede ajustar.
    - Amplia compatibilidad.
    - Es económico [34].

  - Desventajas:
    - No tiene buena estabilidad.
    - No son fiables puesto que fallan bastante.
    - Requiere calibración [33].

- ***Microstep Driver TB6600***
  - Ventajas [35]:
    - Ofrece una mayor estabilidad que el anterior driver.
    - Alta resolución (micropasos 1/32).
    - Permite elegir el modo de micropaso.
    - La corriente se puede ajustar.
    - Protección térmica debido a que tiene un disipador incorporado.
    - Gran compatibilidad.

  - Desventajas [36]:
    - Precio más elevado que el anterior driver.

Atendiendo a todo lo indicado, y aunque ambos son compatibles con un microcontrolador, se ha decidido utilizar el controlador Microstep Driver TB6600 para controlar los motores paso a paso que incorpora la mesa XY. Esto se debe a que a pesar de tener un precio más elevado que el otro controlador, ofrece mejores características para un tiempo prolongado de uso al contar con un disipador de temperatura y también proporciona la corriente deseada al motor al tener más estabilidad que el controlador Pololu DRV8825.

Otra de las ventajas que tiene respecto al DRV8825 es que tiene más modos de micropasos, más libertad para ajustar la corriente y es más cómodo de configurar ya que la configuración se realiza con los interruptores que lleva incorporados.

2.4. Alternativas de microcontrolador

Un microcontrolador (ilustración 17) es un circuito integrado digital que puede ser usado para diversos propósitos debido a que es programable. Está compuesto por una unidad central de proceso (CPU), memorias (ROM y RAM) y líneas de entrada y salida (periféricos). Tiene los mismos bloques de funcionamiento básicos de una computadora lo que nos permite tratarlo como un pequeño dispositivo de cómputo [37].

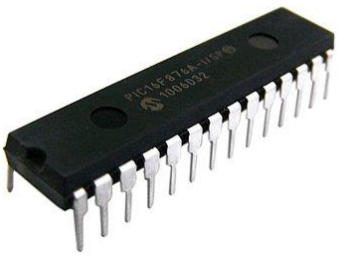


Ilustración 17. Microcontrolador [38].

Como el hardware ya viene integrado en un solo chip. Para usar un microcontrolador se debe especificar su funcionamiento por software a través de programas que indiquen las instrucciones que el microcontrolador debe realizar. En la memoria se guardan los programas y la CPU se encarga de procesar paso por paso las instrucciones del programa (ilustración 18). Los lenguajes de programación típicos que se usan para este fin son ensamblador y C [37].

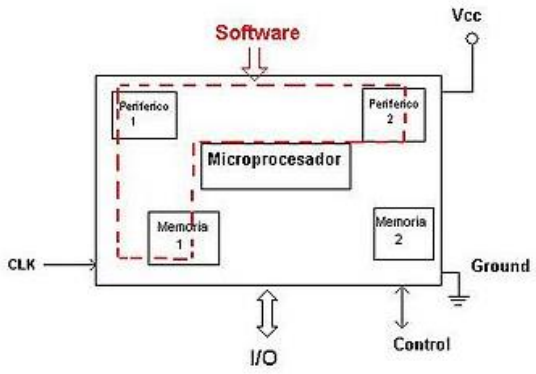


Ilustración 18. Esquema de funcionamiento de un microcontrolador [39].

Los microcontroladores tienen muchas aplicaciones en los sistemas digitales. Por ejemplo, para el diseño de controladores de temperatura automáticos, máquinas dispensadoras, dispositivos biomédicos. En la industria del entretenimiento como juguetes. Incluso en aplicaciones aeroespaciales, sistemas de medición, sistemas de instrumentación [40].

A continuación, se van a analizar dos alternativas de microcontrolador que son: la placa Arduino Nano y la Raspberry Pi.

2.4.1. Arduino Nano

Arduino inició su creación en el año 2005 por un grupo de estudiantes e ingenieros que formaban parte del Interaction Design Institute Ivrea (IDII) de Ivrea (Italia). El objetivo principal era desarrollar una plataforma de hardware y software de código abierto para facilitar la creación de prototipos electrónicos que fuese también de menor coste dado que el coste de los microcontroladores que se utilizaban era bastante alto.

Antes del Arduino Nano (ilustración 19), el cual fue lanzado en 2008, hubo varias versiones y modelos de placas Arduino que fueron lanzados al mercado. Uno de los modelos más conocidos de Arduino es el Arduino Uno que fue lanzado en septiembre de 2010 [41].

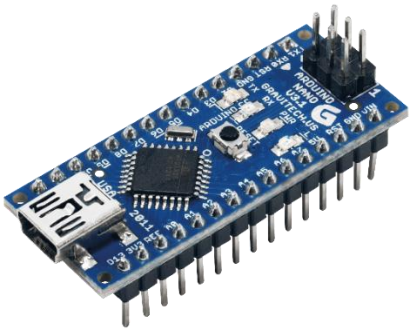


Ilustración 19. Placa Arduino Nano [42].

La placa Arduino es un componente esencial para la recepción de las ordenes provenientes del ordenador. Gracias al microcontrolador incorporado en este tipo de circuitos (ilustración 20), se pueden realizar tareas complejas con alta precisión y bajo consumo de energía.

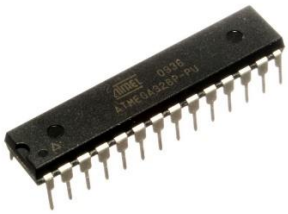


Ilustración 20. Microcontrolador ATMEGA328P [43].

A pesar de tener su propio regulador de voltaje en su esquema, este componente es alimentado por el puerto USB. Además, debido a que se alimenta en un circuito aislado, su uso es más seguro porque los motores suelen generar corrientes inversas al perder su fuente de alimentación, dañando los diferentes componentes conectados al mismo ramal [44].

El circuito integrado utilizado dispone de un puerto USB el cual realiza diferentes tareas a lo largo de las distintas fases del proyecto [44]. El primer paso es programar el dispositivo con el código deseado. Mediante el código enviado, el controlador sabrá qué acciones debe tomar ante las diferentes señales que perciba del ordenador. Una vez en funcionamiento, el microcontrolador escucha repetidas veces el puerto serie, esperando los valores necesarios para que el motor paso a paso se mueva.

Cabe destacar que, para evitar problemas electrónicos, las referencias de los finales de carrera y del Arduino deben estar unidas. Por otra parte, las referencias de ambos motores deben de estar unidas para así tomar el mismo voltaje de referencia de la fuente de alimentación.

Además, el modelo de Arduino seleccionado dispone de diferentes entradas y salidas digitales, destinadas a enviar señales de control a los actuadores y dos restringidas a la comunicación mediante el puerto serie (ilustración 21). Por ello, aunque el puerto serie se conecta mediante el puerto USB, la información se envía al microcontrolador mediante los pines cero y uno, que se utilizarán únicamente para este fin [44].

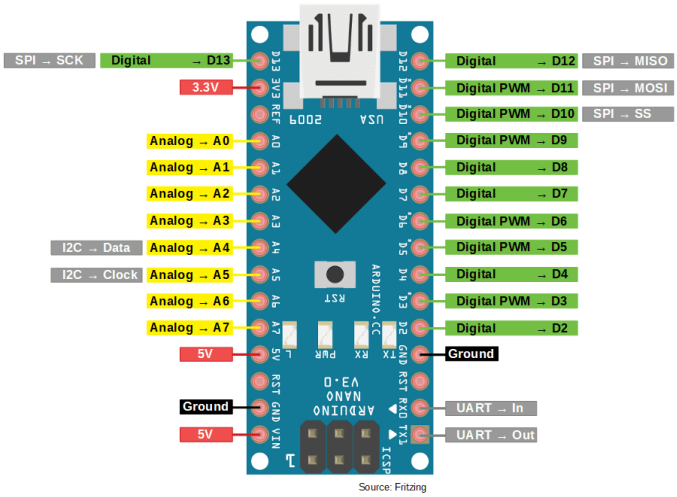


Ilustración 21. Esquema pines placa Arduino Nano [45].

2.4.2. Raspberry Pi

La placa Raspberry Pi (ilustración 22) es un ordenador monoplaca u ordenador de placa simple (SBC por las siglas del anglicismo Single Board Computer). Fue desarrollado por la Fundacion Raspberry Pi que surgió en el año 2009 en Reino Unido, con el objetivo animar a los niños a aprender informática en las escuelas de una manera más sencilla y asequible [46].

Ilustración 22. Placa Raspberry Pi [47].

Esta placa se puede usar en proyectos de electrónica y para tareas básicas que haría cualquier ordenador de sobremesa como navegar por internet, hojas de cálculo, procesador de textos, reproducir vídeo en alta definición e incluso jugar a ciertos juegos.

Está compuesta de diferentes partes que son: CPU (Unidad Central de Procesamiento), memoria RAM, puertos de entrada y salida de audio y video, conectividad de red, ranura SD para almacenamiento, reloj, toma de alimentación y conexiones para periféricos de bajo nivel. Los diferentes elementos que la conforman se muestran en la ilustración 23.

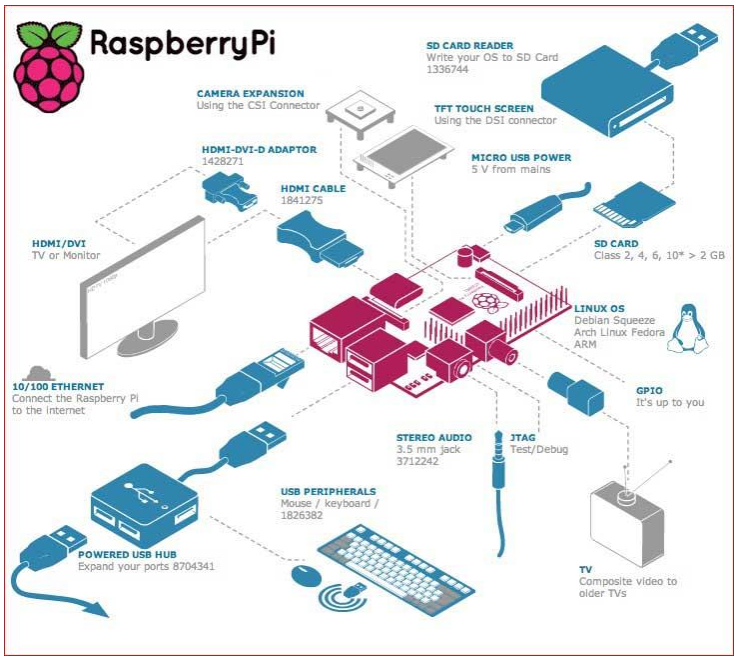


Ilustración 23. Elementos de una placa Raspberry Pi [48].

Para poner la placa en marcha será necesario conectar periféricos de entrada y salida para poder interactuar, como una pantalla, un ratón y un teclado, así como grabar un sistema operativo para Raspberry Pi en la tarjeta SD [46].

2.4.3. Comparativa y solución adoptada

Con la finalidad de poder seleccionar la mejor opción entre las alternativas planteadas, se va a realizar una comparativa entre ambas exponiendo sus ventajas y desventajas.

- ***Arduino Nano***

Es una placa que funciona como un microcontrolador basado en el microcontrolador ATmega328P y desarrollada por Arduino.cc. Esta placa fue lanzada inicialmente en 2008 y ofrece la misma conectividad y especificaciones de la placa Arduino Uno con un tamaño más reducido [49].

- Ventajas [50]:
  - Posee un software sencillo lo que lo convierte en una gran plataforma para principiantes.
  - Consume poca energía eléctrica y es de tamaño reducido.
  - El software de Arduino es gratuito, de código abierto y compatible con diferentes sistemas operativos.
  - Cuenta con una gran cantidad de bibliotecas que pueden ser utilizadas para programar la placa.
  - Tiene un precio económico [51] lo cual hace que sea más accesible a los usuarios.

- Desventajas [50]:
  - Tiene limitaciones en cuanto a memoria y velocidad de procesamiento, lo que puede ser un problema en proyectos complejos. Cuenta con una memoria Flash de 32 kB, una SRAM de 2kB y una EEPROM de 1kB y la velocidad de procesamiento es de 16 MHz [44].

- ***Raspberry Pi***
  - Ventajas [52]:
    - Cuenta con una salida HDMI, lo que le puede permitir conectarse directamente a un televisor de alta definición.
    - Tiene conectividad Wi-Fi integrada.
    - Ofrece potencia de procesamiento.

  - Desventajas [52]:
    - El consumo de energía eléctrica de esta placa es elevado.
    - Es compleja de poner a punto ya que hay que instalar un sistema operativo y configurar los servicios necesarios para realizar la tarea que queremos.

Atendiendo a todo lo anterior, y como uno de los objetivos a alcanzar es que el equipo diseñado sea económico, se optó por el Arduino Nano. Su coste es menor que el de una placa Raspberry Pi y además tiene tamaño más reducido y un bajo consumo de energía. Asimismo, el programa a realizar no requiere de muchos recursos, por lo que la memoria Flash con la que cuenta el Arduino es suficiente para la correcta ejecución del movimiento. Finalmente indicar que el software que dispone el Arduino es sencillo de utilizar.

2.5. Alternativas de interfaz de usuario

Para este proyecto es necesaria una interfaz de usuario para que la persona que vaya a utilizar este sistema pueda controlar el movimiento de la mesa sin necesidad de modificar el programa del microcontrolador. La interfaz de usuario contará con una parte para hacer un ajuste grueso y un ajuste fino inicial, y otra parte en la que el usuario podrá introducir los datos necesarios para que la mesa pueda ir realizando el recorrido, y el escáner pueda captar las imágenes necesarias.

2.5.1. LabVIEW

LabVIEW (acrónimo de Laboratory Virtual Instrument Engineering Workbench) es una plataforma y entorno de desarrollo para diseñar sistemas, utilizando un lenguaje de programación visual gráfico pensado para sistemas hardware y software de pruebas, control y diseño, etc.

Este programa fue creado por National Instruments (1976) para funcionar en máquinas MAC. Salió al mercado por primera vez en 1986 [53], teniendo versiones disponibles para diferentes sistemas operativos como Windows y Linux.

Los programas desarrollados con LabVIEW se llaman Instrumentos Virtuales, o Vis. Su origen provenía del control de instrumentos. En la actualidad, se ha expandido ampliamente, no solo al control de todo tipo de electrónica (Instrumentación electrónica) sino que también se ha expandido en campos como la programación embebida, las comunicaciones, las matemáticas, etc.

Es una herramienta gráfica de programación, esto significa que los programas no se escriben, sino que se dibujan, facilitando su comprensión. Al tener ya pre-diseñados una gran cantidad de bloques, se le facilita al usuario la creación del proyecto. Con ello, en vez de estar una gran cantidad de tiempo en programar un dispositivo/bloque, es posible reducir dicho tiempo y dedicarlo un poco más en la interfaz gráfica y la interacción con el usuario final [53]. Cada VI consta de dos partes diferenciadas:

- **Panel Frontal** (ilustración 24): es la interfaz con el usuario. Se utiliza para interactuar con el usuario cuando el programa se está ejecutando. Los usuarios podrán observar los datos del programa actualizados en tiempo real. En esta interfaz se definen los controles e indicadores [53].

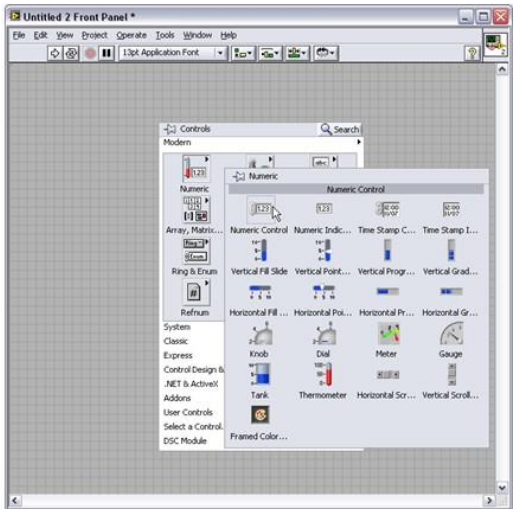


Ilustración 24. Panel frontal LabVIEW [54].

- **Diagrama de bloques** (ilustración 25): es el programa propiamente dicho, donde se define su funcionalidad y se colocan íconos que realizan una determinada función interconectados [53].

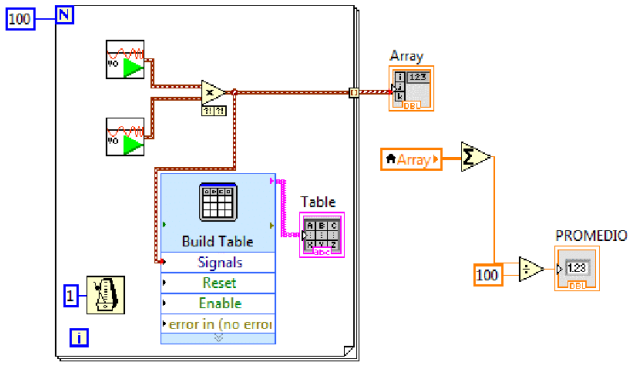


Ilustración 25. Diagrama de bloques LabVIEW [55].

2.5.2. Visual Studio

La plataforma de Visual Studio (ilustración 26) es un entorno de desarrollo integrado (IDE, por sus siglas en inglés), creado por Microsoft para Windows y macOS. Es una plataforma de lanzamiento creativa que puede utilizar para editar, depurar y compilar código. Además del editor y depurador estándar que ofrecen la mayoría de IDE, Visual Studio incluye compiladores, herramientas de completado de código, diseñadores gráficos, y muchas más funciones para mejorar el proceso de desarrollo de software de cualquier aplicación realizada [56].

Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc. [57].

Ilustración 26. Visual Studio [58].

2.5.3. Comparativa y soluciones adoptadas

Con la finalidad de poder seleccionar la mejor opción entre las alternativas planteadas, se va a realizar una comparativa entre ambas exponiendo sus ventajas y desventajas.

- ***LabVIEW***
  - Ventajas [59]:
    - Cuenta con una biblioteca de instrumentos incorporada que permite a los usuarios interaccionar con dispositivos de adquisición de datos y control de instrumentos.
    - Permite crear fácilmente interfaces de usuario gráficas personalizadas.
    - Puede ser más eficiente que otros programas en términos de uso de memoria y velocidad de ejecución.

  - Desventajas [59]:
    - Uso difícil para personas que no han utilizado nunca este tipo de programas.
    - Es costoso en términos de licencias y herramientas adicionales.
    - Tiene limitaciones en cuanto al hardware, por lo que no es muy flexible.
    - Ocupa mucho espacio de almacenamiento en el equipo donde se instale.

- ***Visual Studio***
  - Ventajas [60]:
    - Emplea una programación basada en texto.
    - Es compatible con una amplia gama de lenguajes de programación y plataformas, por lo que es muy flexible.
    - Se integra bien con otras herramientas de desarrollo.
    - Es un software gratuito.
    - Cuando se realiza un proyecto, genera un archivo ejecutable (.exe). Éste se puede ejecutar desde cualquier ordenador sin necesidad de tener el programa instalado.

  - Desventajas:
    - La programación basada en texto puede ser menos intuitiva que la programación visual para personas que no cuenten con experiencia en la programación.

Atendiendo a todo lo anterior, se decidió usar Visual Studio. Aunque ambos programas son compatibles con Arduino, haciendo pruebas con LabVIEW se llegó a la conclusión de que esta última era más compleja de utilizar. Visual Studio cuenta con una interfaz mucho más intuitiva y sencilla de aprender, ya que se programa por texto en lugar de por bloques. Además, se trata de un software gratuito.

2.6. Esquema eléctrico de la solución adoptada

En referencia a los apartados anteriores, se muestra a continuación la solución adoptada en las diferentes partes que la componen. Éstas se muestran en el diagrama de bloques de la ilustración 27, formado por las siguientes etapas: alimentación, microcontrolador, interfaz de usuario, controlador de motores y motor paso a paso.

Este diagrama de bloques sigue la misma estructura que el mostrado en la ilustración 1, pero se han actualizado los bloques indicando los componentes que se han seleccionado en el apartado anterior.

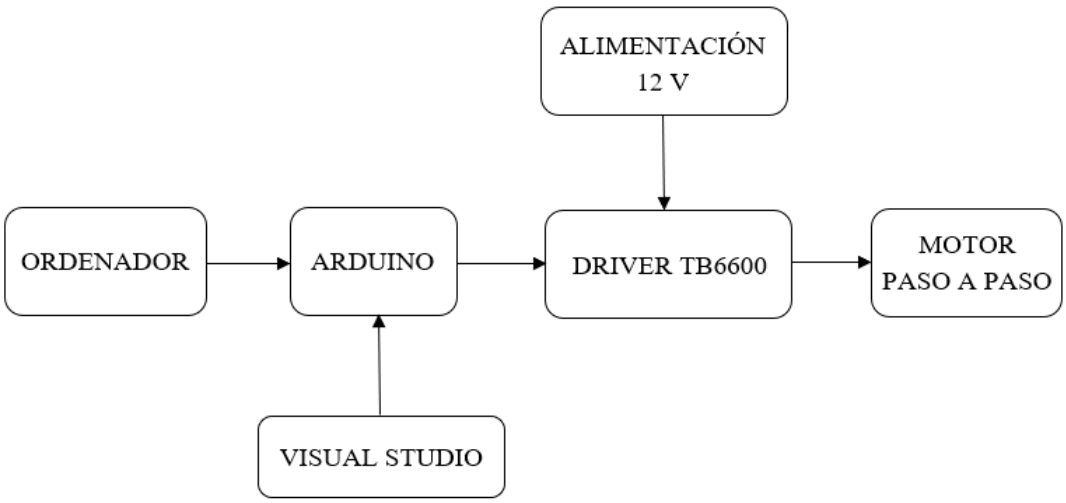


Ilustración 27. Diagrama de bloques de la solución adoptada.

Siguiendo el diagrama de bloques se ha diseñado el circuito con todos los componentes. La interconexión entre ellos se muestra en la ilustración 28.

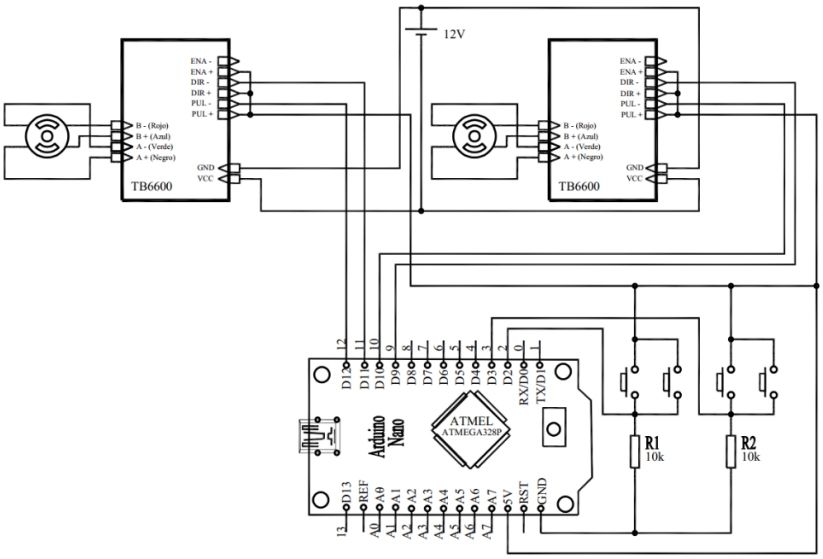


Ilustración 28. Circuito completo.

En este circuito pueden observarse todas y cada una de las conexiones realizadas para el correcto funcionamiento de la mesa XY. Los finales de carrera se han simulado con pulsadores, y puede observarse como se han conectado de dos en dos, en paralelo. La decisión de conectar de esta forma los finales de carrera se debe a que el Arduino Nano solo tiene dos pines donde hay interrupciones, que son el D2 y el D3. Además, el escáner no puede estar a la vez en dos extremos de un mismo eje.

Por otro lado, es necesaria una resistencia de 10 kΩ para cada dos finales de carrera, para así evitar cortocircuitos de la fuente de alimentación de 5 V cuando el final de carrera está activado. Todos estos componentes mencionados estarán alimentados con 5 V, mediante el pin correspondiente de Arduino.

Los actuadores a utilizar van a ser dos motores paso a paso proporcionados por la casa *Makeblock*. En la tabla 2 aparecen los pines de la placa de Arduino que se han utilizado para controlar los diferentes elementos.

Pines digitales	Control
Pin 11	Pin dirección Eje X
Pin 12	Pin movimiento Eje X
Pin 9	Pin dirección Eje Y
Pin 10	Pin movimiento Eje Y
Pin 3	Final de Carrera Eje X
Pin 2	Final de Carrera Eje Y

Tabla 2. Conexión de los pines de Arduino.

Por otra parte, se puede visualizar el conexionado del controlador utilizado, el cual tiene conectado el motor, la alimentación de 12 V, necesaria para el motor y las conexiones con Arduino de los pines necesarios para mover el motor. Durante las pruebas realizadas para el funcionamiento se midió la corriente que consumía el motor, siendo esta de aproximadamente 100 mA en reposo y 140 mA en movimiento.

La superficie de barrido total de la mesa es de es de 356 mm en el eje X y de 346 mm en el eje Y la cual cumple con los mínimos requeridos que era poder barrer una superficie de 300x300 mm.

Para medir la distancia por paso se introdujeron en la interfaz de usuario los parámetros necesarios para que diese 5 vueltas, lo que equivaldría a 8000 paso ya que cada vuelta son 1600 pasos. Con ello, se obtuvo una distancia por paso de 22.79 µm.

Capítulo 3: Desarrollo del software de medida

3.1. Estructura principal

Para poder programar la estructura principal se ha realizado el diagrama de flujo que se muestra en la ilustración 29.

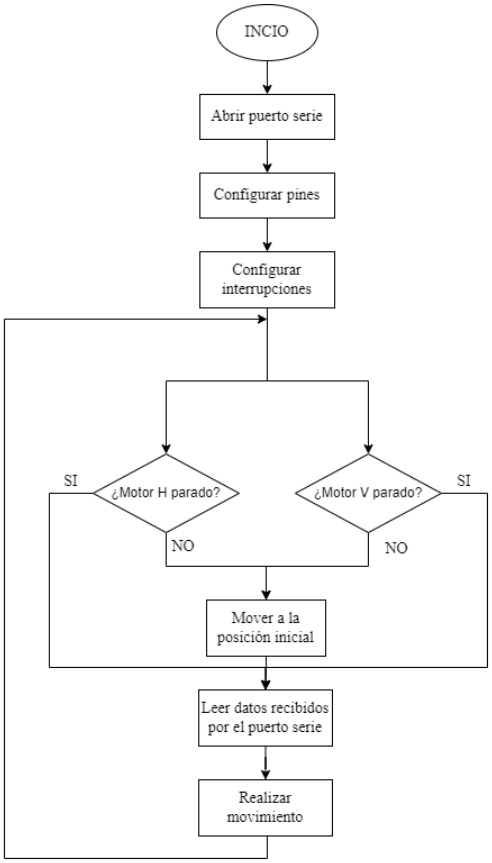


Ilustración 29. Diagrama de flujo del programa principal.

En primer lugar, es necesario incluir en el código las instrucciones necesarias para abrir el puerto serie, configurar las entradas y salidas, y configurar las interrupciones.

Una vez realizado esto, habrá que comprobar si los motores están parados o no. En caso de no estar parados la mesa deberá moverse a la posición inicial y una vez ahí esperará a recibir instrucciones desde la interfaz de usuario. Cuando se reciban datos, estos serán interpretados, y la mesa podrá realizar el movimiento correspondiente.

3.2. Interrupciones

Con tal de poder programar las interrupciones se ha realizado el diagrama de flujo que se muestra en la ilustración 30.

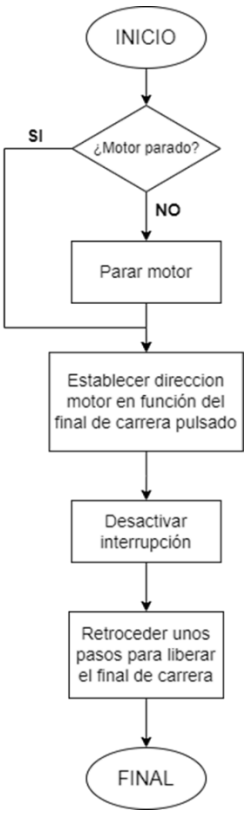


Ilustración 30. Diagrama de flujo asociado a las interrupciones.

Las interrupciones se utilizan para que, cuando realice el movimiento de desplazarse hasta la posición inicial, pare al llegar al final de carrera y no siga avanzando. Una vez alcanza el final de carrera, retrocede unos pocos pasos para que el final de carrera quede liberado y se queda en esa posición hasta recibir nuevas órdenes. Durante el proceso de medida del escáner las interrupciones estarán desactivadas.

Como el escáner no puede estar a la vez en ambos extremos de los ejes, y hay 4 finales de carrera, se va a utilizar una misma interrupción para dos finales de carrera. Esto se puede conseguir leyendo la dirección en la que va el motor y estableciendo la dirección contraria a la que se está moviendo para el retroceso de pasos.

3.3. Función para procesar comandos

Con tal de poder programar la función para procesar los comandos introducidos por el usuario (matriz) se ha realizado el diagrama de flujo que se muestra en la ilustración 31.

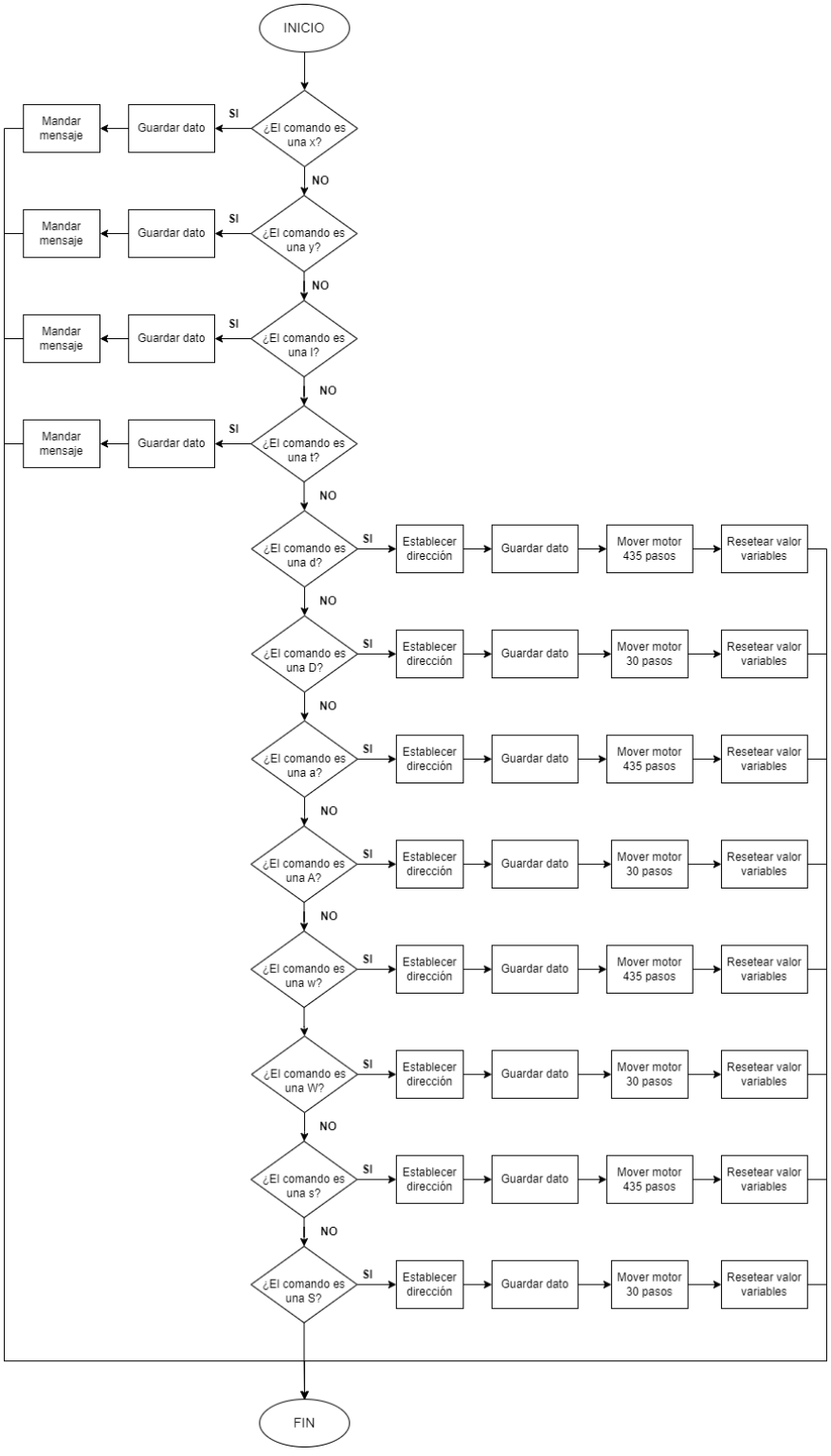


Ilustración 31. Diagrama de flujo para procesar comandos.

En esta función se procesan tanto los comandos enviados por las flechas utilizadas para realizar el posicionamiento inicial del escáner, como los datos numéricos necesarios para recorrer la superficie requerida. A cada botón de la interfaz de usuario habrá que asignarle un comando.

3.4. Función para recorrer la matriz

Con tal de poder programar la función para recorrer la matriz introducida por el usuario se ha realizado el diagrama de flujo que se muestra en la ilustración 32.

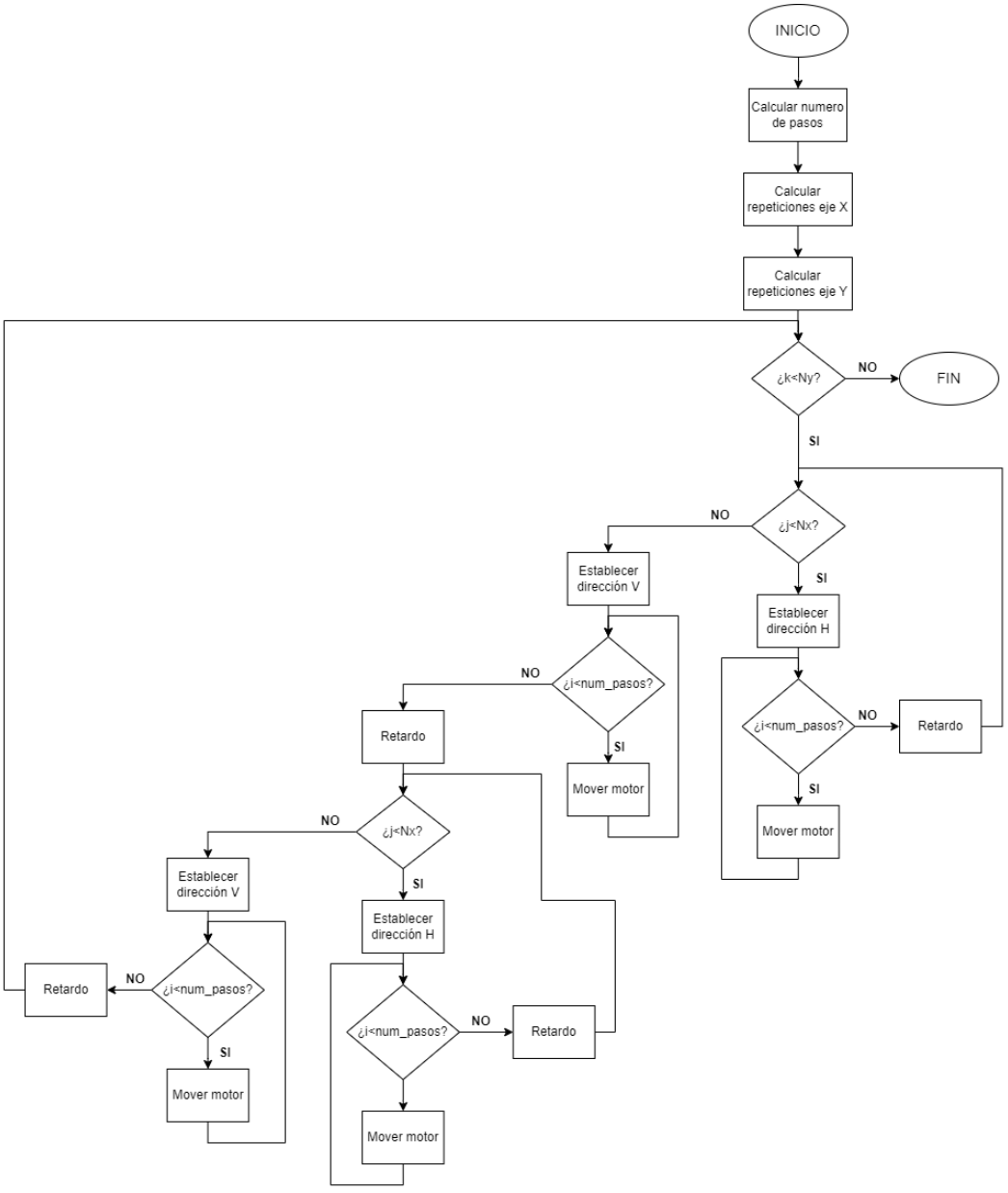


Ilustración 32. Diagrama de flujo de la función movimiento.

Esta función se encargará de recorrer la superficie indicada mediante el usuario, haciendo las medidas a la distancia especificada y esperando el tiempo necesario para efectuar cada medida.

Está programada mediante bucles anidados que le posibilitan realizar los movimientos necesarios mientras las repeticiones sean menores o iguales que las necesarias para alcanzar la distancia total que se quiere medir.

3.5. Desarrollo del código en Arduino y Visual Studio

El movimiento de los motores está programado mediante el lenguaje de Arduino, en el cual se ha realizado el código necesario para llevar a cabo las funciones principales para la ejecución de desplazamiento indicado. Por otro lado, se ha realizado la programación y diseño de la interfaz de usuario en Visual Studio.

Ambos códigos mencionados, debido a su extensión, se desarrollan en el Anexo Nº 3. Programación del sistema.

Capítulo 4: Manual de usuario

La interfaz gráfica de usuario cuenta con todos los elementos necesarios para realizar el posicionamiento manual inicial, y para establecer los parámetros de caracterización. La interfaz de usuario mencionada se muestra en la ilustración 33.

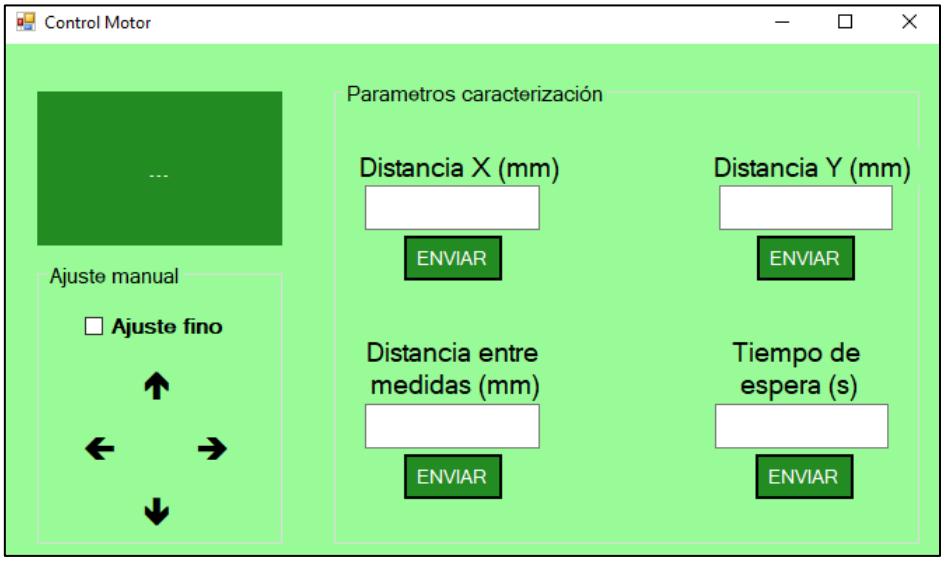


Ilustración 33. Interfaz gráfica de usuario.

Una vez el escáner se haya movido a la posición inicial de la mesa XY, se podrá realizar el ajuste manual para llevar el escáner hasta el punto donde el usuario quiera empezar la medida. Para ello, deberá de hacer uso de las flechas que se encuentran dentro de la parte de ajuste manual (ilustración 34) y una vez estén cerca del punto se podrá activar la casilla de ajuste fino (30 pasos) para un ajuste más preciso del punto. El ajuste más grueso tiene un desplazamiento de 435 pasos cada vez que se pulse la flecha.

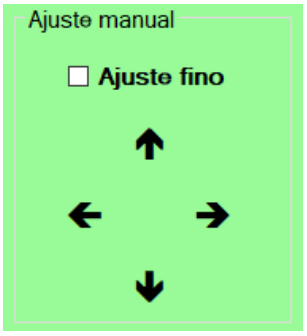


Ilustración 34. Ajuste manual del escáner.

Tras alcanzar la posición inicial del escaneo fijada por el usuario, se procederá a introducir los parámetros correspondientes para realizar la medida en la ventana de trabajo denominada “parámetros de caracterización” (ilustración 35).

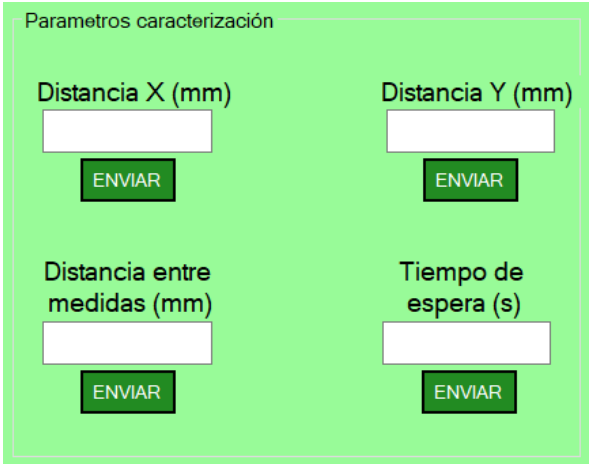


Ilustración 35. Parámetros de caracterización.

Los parámetros se deben de introducir en el orden mostrado a continuación:

1) Distancia que se desea recorrer en el eje X (expresada en milímetros).
2) Distancia que se desea recorrer en el eje Y (expresada en milímetros).
3) Tiempo de espera necesario para que el escáner pueda realizar la medida (expresado en segundos).
4) La distancia de separación entre medida y medida (expresada en milímetros).

Cada vez que se introduzca un parámetro se enviará a Arduino presionando el botón enviar. De esta manera, si el parámetro se ha recibido correctamente aparecerá un mensaje de confirmación en la pantalla incorporada en la interfaz de usuario, tal y como se muestra en la ilustración 36.

Ilustración 36. Ejemplo de mensaje enviado desde Arduino.

Una vez introducidos todos los parámetros, la mesa empezará a realizar el movimiento.

Capítulo 5: Medida experimental

Tratando de verificar el correcto funcionamiento del sistema se han llevado a cabo distintas medidas de prueba. Para comprobar que el escáner recorra la distancia deseada se ha utilizado un papel milimetrado y un puntero láser (ilustración 37), el cual se ha colocado en la parte móvil de la mesa mediante cinta adhesiva. Una vez colocado el puntero se ha recortado un trozo de papel milimetrado de 10 cm x 10 cm y se ha fijado con cinta adhesiva a la mesa, para evitar su movimiento y que esto afecte a la medida.

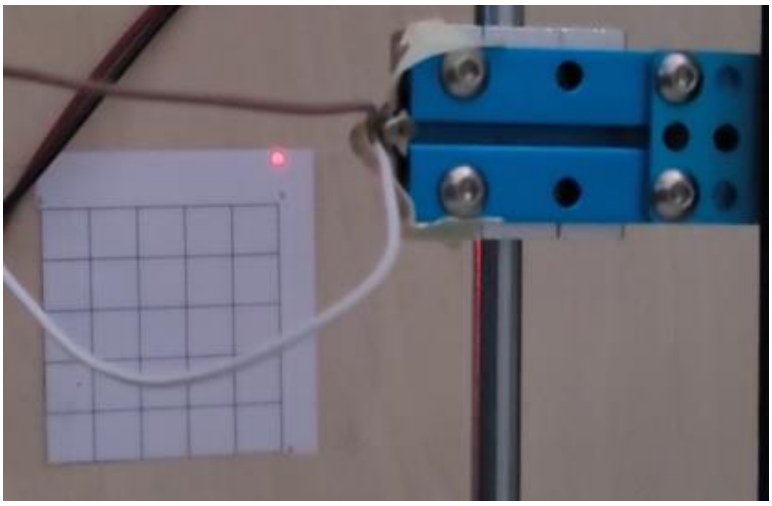


Ilustración 37. Prueba con el papel milimetrado y el puntero láser.

El controlador se ha alimentado mediante una fuente de tensión regulada de 12 V, y se ha conectado el Arduino al ordenador para así poder cargarle el programa. Para realizar la primera prueba se ha desplazado mediante las flechas de la interfaz de usuario el puntero hasta la esquina superior izquierda del cuadrado de papel y se han introducido los parámetros mostrados en la ilustración 38.

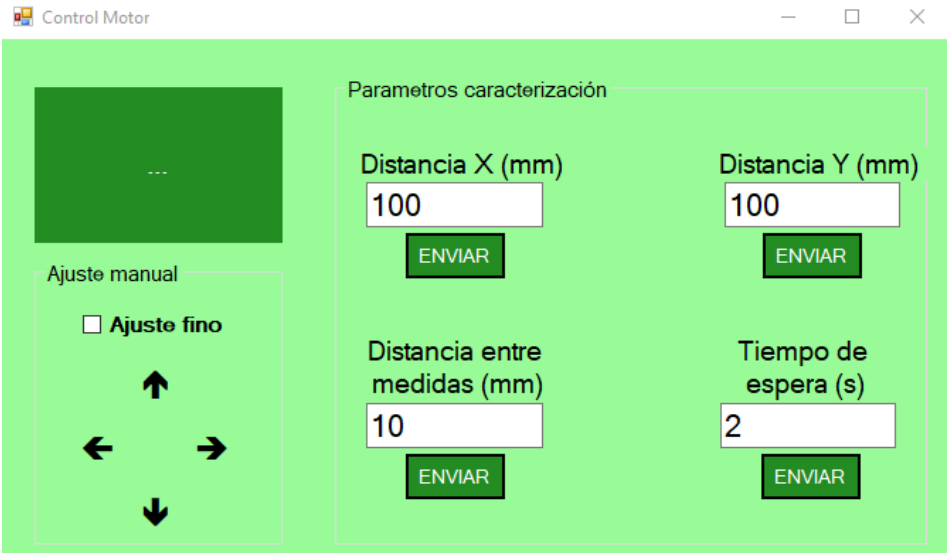


Ilustración 38. Parámetros de la primera prueba.

Tras la primera prueba se comprobó que había un pequeño error (de 1 mm aproximadamente) al recorrer tanta distancia, y aunque errores tan pequeños no afecten, puesto que se van a recorrer superficies grandes, se modificó el valor de la distancia por paso de 0.02279 mm a 0.0235 mm. Así se consiguió reducir el error en la medida.

La segunda prueba realizada pretendía comprobar que se pueden recorrer matrices rectangulares y no únicamente matrices cuadradas. Para ello, y una vez posicionado el puntero al inicio mediante las flechas, se han introducido los parámetros que se muestran en la ilustración 39.

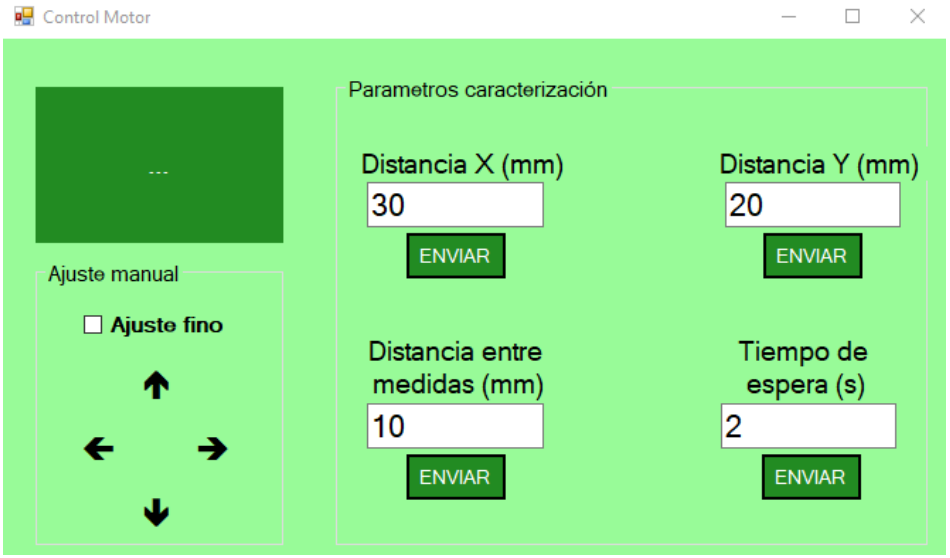


Ilustración 39. Parámetros de la segunda prueba.

Los resultados de ambas pruebas fueron exitosos puesto que las distancias recorridas fueron muy similares a las distancias programadas. En el caso de la segunda prueba, se consiguió demostrar que también se pueden recorrer matrices que sean rectangulares.

Capítulo 6: Conclusiones

En el presente Trabajo de Fin de Grado se ha llevado a cabo el diseño e implementación de un sistema de control mediante la programación de dos motores de una mesa XY, además de la interfaz de usuario. Este sistema de posicionamiento XY está destinado para el uso de un radar de terahercios el cual se utiliza para el análisis de piezas como una encimera de granito o alguna pieza aeronáutica. Además, se puede utilizar para analizar si hay desperfectos en cuadros o pinturas.

El proyecto engloba diferentes tareas tales como: diseño de un circuito electrónico y elección de sus diferentes elementos, manejo de dos motores con sus respectivos controladores, desarrollo de firmware y software para la creación de la interfaz de usuario, y la programación del microcontrolador.

Una vez comprobado el funcionamiento del sistema, se ha podido observar cómo se puede realizar un control del movimiento del escáner preciso, sencillo y de bajo costo con programas gratuitos y de fácil acceso a cualquier persona (como Visual Studio y Arduino).

Una vez terminado el programa se realizaron distintas pruebas las cuales resultaron exitosas. En este sentido se pudo comprobar que podía hacerse un ajuste manual inicial preciso mediante unas flechas incorporadas en la interfaz gráfica, y que el escáner podía recorrer tanto matrices cuadradas como matrices rectangulares según los parámetros establecidos por el usuario.

Para finalizar, cabe indicar distintos puntos de mejora de este Trabajo de Fin de Grado:

- Diseñar una placa de circuito impreso (PCB) con el fin de aportar mayor comodidad a la hora de transportarse y ser montado el circuito. Esta placa incluiría unos conectores para conectar todos los elementos externos como los controladores o los finales de carrera.
- Incluir un botón en la interfaz de usuario de parada de emergencia para que así en caso de cualquier fallo, una vez el usuario lo pulse, todo el sistema se detenga.
- Sincronizar el programa de control de la mesa con el programa del control del escáner para que el tiempo de espera para realizar la medida venga determinado por este último.

Referencias

[1] Adrien Chopard, Frederic Fauquet, Jing Shun Goh, Mingming Pan, Patrick Mounaix and Jean-Paul Guillet, mayo de 2021, https://ieeexplore.ieee.org/document/9455312 Consultado el día 17 de junio de 2023.

[2] José A. López-Salcedo, febrero de 2014, https://openaccess.uoc.edu/bitstream/10609/77345/1/Sistemas%20de%20radionavegaci%C3%B3n_M%C3%B3dulo%202_Sistemas%20radar.pdf Consultado el día 21 de junio de 2023.

[3] Igus motion plastics, 2023, https://www.igus.es/info/mesas-lineales-en-cruz#:~:text=Las%20mesas%20en%20cruz%2C%20tambi%C3%A9n,lineal%20o%20un%20volante%20manual. Consultado el día 17 de junio de 2023.

[4] https://static.rapidonline.com/catalogueimages/product/75/06/s75-0697p01wl.jpg Consultado el día 13 de abril de 2023.

[5] Tecnimetal, 3 de enero de 2020, https://maquinasdemedicionporcoordenadas.com/2020/01/03/historia-de-las-maquinas-de-medicion-por-coordenadas/ Consultado el día 10 de junio de 2023.

[6] Nexus Integra, 2023, https://nexusintegra.io/es/10-beneficios-de-contar-con-un-sistema-de-automatizacion-industrial/ Consultado el día 10 de junio de 2023.

[7] Roberto Adeva, 9 de marzo de 2023, https://www.adslzone.net/reportajes/tecnologia/impresion-3d/ Consultado el día 10 de junio de 2023.

[8] Garvotech, 2023, https://www.gravograph.es/maquinas-de-grabado-y-recorte-laser Consultado el día 10 de junio de 2023.

[9] Sideco, 2023, https://sideco.com.mx/que-es-un-router-cnc/ Consultado del día 10 de junio de 2023.

[10] Equinlab, 2023, https://www.equinlabsac.com/content/mesa-xy Consultado el día 10 de junio de 2023.

[11] https://www.makeblock.es/productos/plotter_v2/ Consultado el día 21 de junio de 2023.

[12] Ingeniería Mecafenix, 20 de abril de 2017, https://www.ingmecafenix.com/electronica/motores-electronicos/motor-paso-a-paso/ Consultado el día 10 de junio de 2023.

[13] https://www.orientalmotor.com.mx/images/technology/step_motor_cutaway.jpg Consultado el día 16 de junio de 2023.

[14] TME, 8 de septiembre de 2020, https://www.tme.eu/es/news/library-articles/page/41861/Motor-paso-a-paso-tipos-y-ejemplos-del-uso-de-motores-paso-a-paso/#:~:text=El%20motor%20paso%20a%20paso%20est%C3%A1%20compuesto%20por%20rotor%20y,giratorio%20creado%20alrededor%20del%20estator Consultado el día 16 de junio de 2023.

[15] Nichese, 2013, https://motores.nichese.com/pasoapaso.html Consultado el día 16 de junio de 2023.

[16] Wikipedia, 28 de mayo de 2023, https://es.wikipedia.org/wiki/Motor_de_corriente_continua Consultado el día 15 de junio de 2023.

[17] https://www.mundodelmotor.net/wp-content/uploads/2022/05/Partes-de-un-Motor-CC-corriente-continua.jpg Consultado el día 15 de junio de 2023.

[18] Oswos, https://oswos.com/es/motor-dc/ Consultado el día 15 de junio de 2023.

[19] https://www.inventable.eu/wp-content/uploads/2017/05/motor_dc_sentido_de_giro.png Consultado el día 15 de junio de 2023.

[20] IES Villalba Hervas, febrero de 2012, https://iesvillalbahervastecnologia.files.wordpress.com/2010/02/motores-electricos-parte-ii1.pdf Consultado el día 15 de junio de 2023.

[21] https://upload.wikimedia.org/wikipedia/commons/7/72/Motor_serie.jpg Consultado el día 15 de junio de 2023.

[22] https://upload.wikimedia.org/wikipedia/commons/b/ba/Motor_paralelo.jpg Consultado el día 15 de junio de 2023.

[23] https://upload.wikimedia.org/wikipedia/commons/3/31/Motor_compuesto.jpg Consultado el día 15 de junio de 2023.

[24] Zuendo, 2021, https://www.zuendo.com/smartblog/26_Motores-cc-ventajas-inconvenientes.html Consultado el día 15 de junio de 2023.

[25] https://www.makeblock.es/public/robot_kit/MT_STEPPER//makeblock_MT_STEPPER.jpg Consultado el día 16 de junio de 2023.

[26] Jose Carlos Villajulca, 8 de noviembre, https://instrumentacionycontrol.net/video-revision-del-driver-de-motores/#:~:text=Un%20driver%20de%20motor%20es,que%20pueda%20alimentar%20el%20motor Consultado el día 16 de junio de 2023.

[27] Wattco, 2023, https://www.wattco.com/es/2020/12/controladores-motor/#:~:text=Los%20controladores%20de%20motor%20pueden,el%20par%20si%20es%20necesario. Consultado el día 15 de junio de 2023.

[28] BricoGeek, https://tienda.bricogeek.com/impresion-3d/853-controlador-de-alta-corriente-drv8825.html?gad=1&gclid=Cj0KCQjw7aqkBhDPARIsAKGa0oK3WHG9V_DVwFlhhonHKm9tUcKD55oyY9KrBUDGj9ic0T2aavp7vm4aAq6qEALw_wcB Consultado el día 15 de junio de 2023.

[29] https://m.media-amazon.com/images/I/51Z5L-nhGkL._AC_UF1000,1000_QL80_.jpg Consultado el día 16 de mayo de 2023.

[30] BrikoGeek, https://tienda.bricogeek.com/controladores-motores/992-controlador-de-motores-paso-a-paso-35a-tb6600.html Consultado el día 15 de junio de 2023.

[31] https://ae01.alicdn.com/kf/Sa0f144d2623c4da4af4b9402c020567fi/Controlador-de-Motor-paso-a-paso-versi-n-mejorada-42-57-86-32-segmentos-TB6600-4.jpg Consultado el día 21 de junio de 2023.

[32] https://tienda.bricogeek.com/img/p/4/1/3/5/4135.jpg Consultado el día 16 de junio de 2023.

[33] Danilo Muñoz Jaramillo, enero de 2023, https://programarfacil.com/blog/arduino-blog/a4988-drv8825-tmc2209/#Driver_para_motores_paso_a_paso_DRV8825 Consultado el día 10 de abril de 2023.

[34] Luis Llamas, 24 de agosto de 2016, https://www.luisllamas.es/motores-paso-paso-arduino-driver-a4988-drv8825/ Consultado el día 22 de marzo de 2023.

[35] Altronics, https://altronics.cl/driver-tb6600-4a Consultado el día 10 de abril de 2023.

[36] Solectro, 2023, https://solectroshop.com/es/drivers-y-kits-de-control/1494-tb6600-cnc-enrutador-1-eje-controlador-motor-paso-a-paso-driver-35a-impresora.html?gclid=CjwKCAjwvpCkBhB4EiwAujULMjleVu8Ze6xcpnnIYDJu9-VQ_-KqncKJ1cBtLdzWb_ZW7I2yP6vzNRoCjcAQAvD_BwE Consultado el día 10 de abril de 2023.

[37] Sherlin Xbot, https://sherlin.xbot.es/microcontroladores/introduccion-a-los-microcontroladores/que-es-un-microcontrolador Consultado el día 15 de junio de 2023.

[38] https://robots-argentina.com.ar/didactica/imagenes/pic16f876.jpg Consultado el día 15 de junio de 2023.

[39] https://upload.wikimedia.org/wikipedia/commons/thumb/c/cb/Microcontrolador.jpg/400px-Microcontrolador.jpg Consultado el día 15 de junio de 2023.

[40] Dr. Rubén E-Marmolejo, 2017, https://hetpro-store.com/TUTORIALES/microcontrolador/ Consultado el día 15 de junio de 2023.

[41] Wikipedia, 26 de mayo de 2023, https://es.wikipedia.org/wiki/Arduino Consultado el día 11 de abril de 2023.

[42] https://kumotica.es/2701-large_default/arduino-nano-v30-atmega328-compatible.jpg Consultado el día 11 de abril de 2023.

[43] https://leantec.es/wp-content/uploads/2018/02/p_1_3_3_8_1338-Microcontrolador-ATMEGA328P-DIP-28-Original-Atmel.jpg Consultado el día 11 de abril de 2023.

[44] José Guerra Carmenate, 2023, https://programarfacil.com/blog/arduino-blog/familia-arduino-nano/ Consultado el día 11 de abril de 2023.

[45] https://elosciloscopio.com/wp-content/uploads/2021/03/Tutorial-de-Arduino-Nano-Pinout.png Consultado el día 11 de abril de 2023.

[46] Eva Rodríguez de Luis, 18 de septiembre de 2018, https://www.xataka.com/makers/cero-maker-todo-necesario-para-empezar-raspberry-pi#:~:text=La%20Raspberry%20Pi%20es%20la,bajo%20nivel%2C%20reloj... Consultado el día 15 de junio de 2023.

[47] https://upload.wikimedia.org/wikipedia/commons/thumb/f/f1/Raspberry_Pi_4_Model_B_-_Side.jpg/280px-Raspberry_Pi_4_Model_B_-_Side.jpg Consultado el día 15 de junio de 2023.

[48] https://electrojoan.com/wp-content/uploads/2017/01/esquema-raspberry-pi.jpg Consultado el día 15 de junio de 2023.

[49] Wikipedia, 13 de junio de 2023, https://en.wikipedia.org/wiki/Arduino_Nano Consultado el día 10 de abril de 2023.

[50] Miguel Dos Santos, 24 de marzo de 2023, https://polaridad.es/ventajas-de-utilizar-arduino-en-proyectos-electronicos/ Consultado el día 10 de abril de 2023.

[51] Kumotica, https://kumotica.es/componentes-robotica/702-arduino-nano-v30-atmega328-compatible.html Consultado el día 10 de abril de 2023.

[52] Edgar Landivar, 4 de junio de 2021, https://neomano.com/arduino-vs-raspberry-pi-cual-es-mejor/ Consultado el día 15 de junio de 2023.

[53] Wikipedia, 4 de abril de 2022, https://es.wikipedia.org/wiki/LabVIEW Consultado el día 10 de junio de 2023.

[54] https://ni.scene7.com/is/image/ni/Clipboard07_20080702082501?scl=1 Consultado el día 10 de junio de 2023.

[55] https://www.researchgate.net/publication/315599964/figure/fig5/AS:475439673155588@1490365059740/Figura-5-Diagrama-de-bloques-del-instrumento-virtual-en-LabVIEW-para-adquirir-senales-de.png Consultado el día 10 de junio de 2023.

[56] Microsoft, 2023, https://visualstudio.microsoft.com/es/#vs-section Consultado el día 10 de junio de 2023.

[57] Wikipedia, 13 de junio de 2023, https://es.wikipedia.org/wiki/Microsoft_Visual_Studio Consultado el día 15 de junio de 2023.

[58] https://venturebeat.com/wp-content/uploads/2019/11/visual-studio-logo.jpeg?w=1200&strip=all Consultado el día 15 de junio de 2023.

[59] Mecalux, 27 de marzo de 2014, https://www.mecalux.es/articulos-de-logistica/trazo-firme-labview Consultado el día 17 de mayo de 2023.

[60] Conecta Software, 2023, https://www.conectasoftware.com/apps/visual-studio/ Consultado el día 17 de mayo de 2023.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ESCÁNER 2D PARA APLICACIONES DE IMÁGENES SUB-THZ

Documento 02: Planos

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

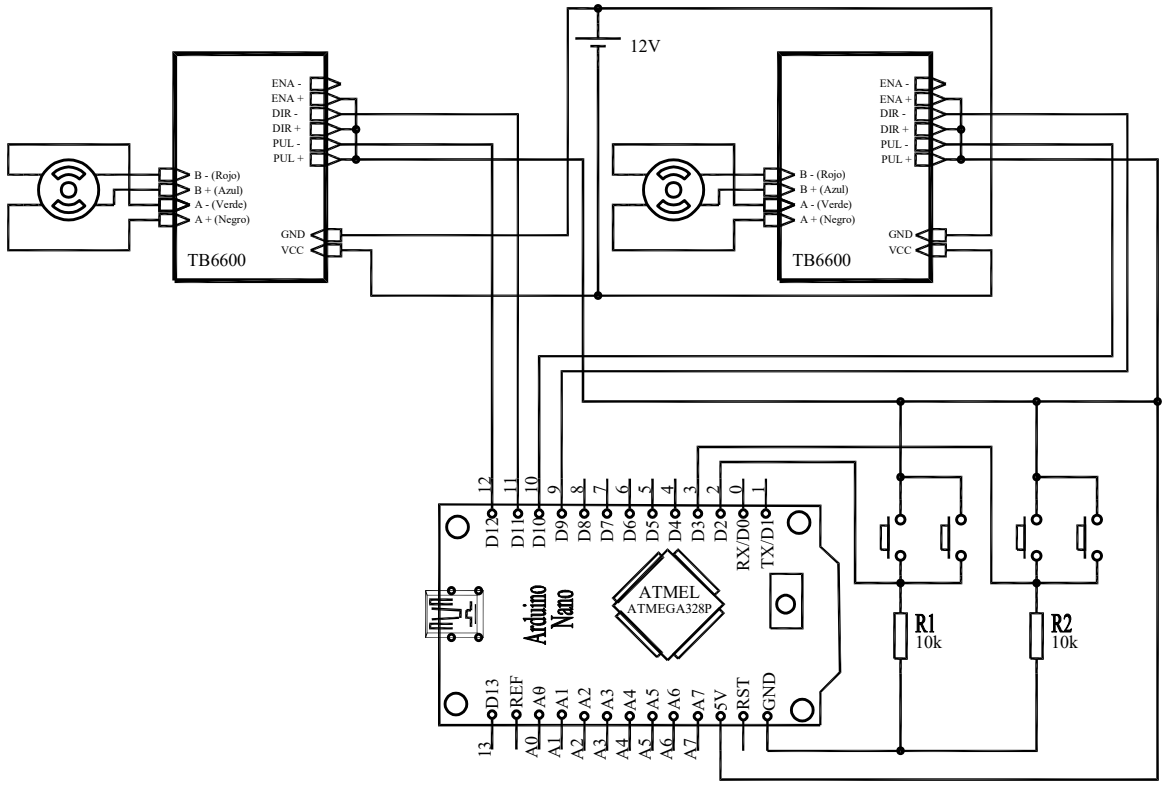
AUTOR/A: Miryam Gómez Pijoan

Tutor/a: Salvador Ponce Alcántara

Cotutor/a: Borja Vidal Rodríguez

CURSO ACADÉMICO: 2022/2023

ÍNDICE PLANOS



Proyecto: SCANNER 2D PARA APLICACIONES DE IMÁGENES SUB-THZ Titular: Miryam Gómez Pijoan Emplazamiento: ETSID, Camino de Vera, Edificio 7B, 46022, Valencia		02/05/2023 Escala 20:1
Autora: Miryam Gómez Pijoan	Plano: Esquema circuito electrónico	Plano Nº 1

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SCANNER 2D PARA APLICACIONES DE IMÁGENES SUB-THZ

Documento 03: Pliego de condiciones

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Miryam Gómez Pijoan

Tutor/a: Salvador Ponce Alcántara

Cotutor/a: Borja Vidal Rodríguez

CURSO ACADÉMICO: 2022/2023

ÍNDICE PLIEGO

1. Definición y alcance del pliego

El objeto del presente documento es establecer las condiciones técnicas para la ejecución de un proyecto que consiste en la programación de una mesa XY mediante un microcontrolador y una interfaz de usuario. Se especifica el montaje, las precauciones a tener en cuenta, la ejecución de la solución propuesta, los materiales, y las características técnicas y de calidad necesarios.

El proyecto describe una solución al problema planteado, así como el diseño de la correspondiente interfaz de usuario, para que el cliente pueda llevar a cabo el control necesario sin tener conocimientos de electrónica. La solución incluye, por tanto, la parte del software, así como la electrónica necesaria para el funcionamiento.

1.1. Pliegos oficiales

El contratista es responsable, en el aspecto laboral, del cumplimiento del Real Decreto 486/2010, de 23 de abril, sobre la protección de la salud y la seguridad de los trabajadores contra los riesgos relacionados con la exposición a radiaciones ópticas artificiales. Respecto a la gestión de residuos generados en la fabricación del dispositivo, es de obligado cumplimiento el Real Decreto 110/2015, de 20 de febrero, sobre residuos de aparatos eléctricos y electrónicos.

1.2. Modificaciones

Si fuesen necesarias, todas las modificaciones durante la ejecución del proyecto deben ser revisadas y aprobadas por el responsable de la dirección del proyecto.

2. Condiciones técnicas

2.1. Motor

2.1.1. Características

El motor ha de cumplir con unas características que le permitan realizar los movimientos correspondientes de la mesa. Además, el motor debe de cumplir con los siguientes requisitos:

- Debe de permitir girar ángulos pequeños (1.8º aproximadamente) para un control más preciso de movimiento.
- La corriente que consume el motor, tanto en movimiento como en reposo debe de ser inferior a 500 mA.
- El motor tiene que ser adecuado para que la temperatura se mantenga por debajo del máximo establecido por el fabricante, para evitar daños.
- El motor debe de poder operar a una frecuencia de 3 Hz aproximadamente para que pueda moverse a una velocidad adecuada.

Para ello, se ha seleccionado el motor paso a paso 42BYG producido por el fabricante *Makeblock* o un modelo equivalente.

Su voltaje de operación es de 12 V y tiene un giro de 1.8 grados por paso. El tamaño del motor es aproximadamente de 42x42x40 mm y su velocidad de giro depende del controlador.

La conexión de los cables es la siguiente:

- **Cable negro** del motor a la **entrada A+** del controlador del motor.
- **Cable verde** del motor a la **entrada A-** del controlador del motor.
- **Cable azul** del motor a la **entrada B+** del controlador del motor.
- **Cable rojo** del motor a la **entrada B-** del controlador del motor.

2.1.2. Calidad de los materiales

Los motores paso a paso llevan un sistema de control para regular la dirección de giro y la velocidad del mismo. Estos motores deben estar fabricados con materiales de buena calidad y duraderos, como imanes de neodimio, acero inoxidable y cobre de alta pureza. En cuanto a la carcasa del motor, debe estar hecha de materiales como aluminio o plástico, que son resistentes y ligeros. También pueden incluir elementos adicionales de protección, como recubrimientos contra la corrosión o sistemas de sellado para evitar la entrada de polvo y humedad.

2.2. Alimentación del sistema

En el presente apartado se exponen las condiciones correspondientes a la alimentación del sistema, el cual será alimentado mediante una fuente de alimentación externa. Es necesario el cumplimiento de estas condiciones para un correcto funcionamiento:

- La fuente de alimentación deberá proporcionar una tensión de 12 V aproximadamente.
- La fuente de alimentación debe de ser capaz de proporcionar la corriente necesaria al motor para realizar su movimiento, la cual debe estar en un rango comprendido entre 110 y 150 mA.
- La fuente de alimentación debe tener protecciones incorporadas para evitar daños en caso de cortocircuitos y sobrecargas.
- La fuente de alimentación debe tener una baja emisión de interferencia electromagnética y debe de ser inmune también a las interferencias externas.

2.3. Controlador

El controlador empleado para los motores debe cumplir con las especificaciones que se detallan a continuación:

- El controlador debe de ser capaz de manejar la corriente (140 mA) y el voltaje (12 V) requeridos por el motor sin sobrecargarse.
- El controlador debe proporcionar las funciones de control de velocidad y de dirección.
- El controlador debe incluir protecciones contra sobrecorrientes y sobretensiones.
- El controlador debe incluir un disipador de temperatura para reducir su calentamiento.
- El controlador debe incluir la posibilidad de utilizar modos de micropasos, para poder dividir el paso del motor entre 1/2, 1/4 y 1/16.

Por ello se recomienda el controlador Microstep Driver TB6600 el cual opera con un voltaje de entrada de entre 12 V y 42 V, y puede proporcionar al motor una corriente de hasta 4 A. Tiene un peso de 200 gramos lo que lo hace ligero para su utilización. Otro aspecto importante es que este controlador es compatible con el motor escogido.

La conexión de los cables es la siguiente:

- Conexión ENA- al aire.
- Conexión ENA+ al pin de 5V del Arduino.
- Conexión DIR- a los pines 11 o 9 (dependiendo del motor) para establecer la dirección.
- Conexión DIR+ al pin de 5V del Arduino.
- Conexión PUL- a los pines 12 o 10 (dependiendo del motor) para el movimiento del motor.
- Conexión PUL+ al pin de 5V del Arduino.
- Conexión B- al cable rojo del motor paso a paso.
- Conexión B+ al cable azul del motor paso a paso.
- Conexión A- al cable verde del motor paso a paso.

- Conexión A+ al cable negro del motor paso a paso.
- Conexión GND a la referencia de la fuente de alimentación.
- Conexión VCC a la tensión de la fuente de alimentación.

2.4. Microcontrolador

El microcontrolador empleado debe cumplir con las especificaciones que se detallan a continuación:

- El microcontrolador tiene que poder gestionar el movimiento de los dos motores de la mesa XY en cada eje.
- El microcontrolador debe tener un puerto USB para poder establecer la comunicación con el ordenador.
- El microcontrolador debe de tener una arquitectura ARM (Advanced RISC Machines) para la aplicación.
- El microcontrolador debe de disponer de una memoria de al menos 15 KB para almacenar el programa.
- El microcontrolador debe de llevar integrados periféricos como puertos de entrada/salida o puertos para la comunicación. Serán necesarios como mínimo 6 pines de entrada/salida y 1 puerto para la comunicación serie.
- El microcontrolador debe de consumir menos de 30 mA en reposo.
- El microcontrolador debe de tener un entorno de desarrollo adecuado.

Por todo ello se aconseja la placa Arduino Nano del fabricante Elegoo o equivalente. Este microcontrolador tiene una memoria Flash de 32 KB, una memoria SRAM de 2 KB, una memoria EEPROM de 1 KB, y un total de 22 puertos de entrada/salida de los cuales 14 son digitales (6 son PWM) y 8 son analógicos.

La placa de Arduino Nano, permite alimentarla conectado a la alimentación eléctrica de 5 V, mediante el puerto USB tipo B 2.0. Las referencias de esta placa, la de los motores y la de los demás componentes electrónicos deben estar unidas.

2.5. Mesa XY

La mesa XY empleada debe cumplir con las especificaciones que se detallan a continuación:

- La mesa XY debe tener una superficie de trabajo de al menos 300 x 300 mm.
- La mesa XY debe de estar construida en un material resistente como podría ser aluminio.
- La mesa XY debe de disponer de piezas que se puedan sujetar a los extremos de los ejes para poder colocar los finales de carrera.
- La mesa XY debe de ser fácil de montar.

Por todo ello, se aconseja el modelo XY-Plotter Robot Kit del fabricante *makeblock*. Este paquete incluye los cuatro finales de carrera necesarios con sus soportes, y los dos motores paso a paso. Contiene también un manual de montaje con ilustraciones muy intuitivo y sencillo de seguir, por lo que resulta fácil de montar.

Esta mesa, tiene una superficie total de 620x620 mm con un área de trabajo de 356x346 mm lo que la hace ideal para la superficie mínima requerida. Además, está fabricada en aluminio anodizado por lo que será una estructura resistente.

2.6. Interfaz de usuario

La interfaz de usuario empleada para el control de los motores debe cumplir con las especificaciones que se detallan a continuación:

- La interfaz de usuario debe incorporar una interfaz gráfica que permita al usuario ajustar manualmente el escáner y definir los parámetros de movimiento de los motores para poder escanear una superficie determinada.
- Se recomienda que el software para realizar la interfaz sea gratuito.
- En la interfaz de usuario se debe mostrar, a través de una etiqueta, que se han recibido correctamente los parámetros para el movimiento.
- La comunicación con el microcontrolador se realizará mediante el envío de cadenas de caracteres, a través del puerto de comunicación.

3. Ejecución

3.1. Generalidades de la ejecución

La finalidad de la documentación técnica, incluida junto con esta especificación técnica, es aportar información detallada para la instalación de la mesa XY y del circuito electrónico correspondiente. En caso de que algún material no aparezca en dicha documentación, pero aparezca en los planos del proyecto, se deberá suministrar e instalar igualmente.

Todo el funcionamiento del sistema irá en concordancia con los planos que se adjuntan, los cuales se han diseñado de acuerdo con la documentación técnica.

3.2. Procesos de ejecución de la instalación

El técnico encargado de instalar la mesa debe contar con los conocimientos suficientes para una correcta instalación de la mesa y un correcto montaje del circuito electrónico. Las ejecuciones descritas a continuación son las realizadas por el ingeniero electrónico encargado.

3.2.1. Placa Arduino Nano y circuito electrónico

En primer lugar, será necesario conectar la placa a la alimentación mediante el puerto USB, se alimenta en un circuito aislado para garantizar la seguridad del dispositivo y de las personas que lo manipulan, y así también evitar dañar los circuitos por corrientes inversas. La referencia del Arduino (GND) se conecta a la referencia de los demás elementos electrónicos del circuito.

Se realiza la conexión de los pines 9 y 10 (motor del eje horizontal), 11 y 12 (motor del eje vertical) para el control del motor, que corresponden al control de la dirección y el paso del motor respectivamente. Posteriormente se conectarán los finales de carrera a los pines 2 y 3 de la placa. Las conexiones son mediante conductores con una sección de 0.5 mm^2.

4. Prueba de servicio

Para garantizar el correcto funcionamiento, se realizará una prueba de servicio una vez completada la instalación del proyecto para así poder corregir los posibles fallos que tuviese. Este control se llevará a cabo por la persona encargada de la instalación de la mesa una vez completada la mencionada instalación.

Primeramente, se comprueba que todas las conexiones del circuito electrónico están como se indican en el proyecto, cumpliendo así todas las especificaciones y se comprueba también que no se produzca ningún fallo. Este control es realizado por el ingeniero electrónico que monta el circuito.

Tras la primera prueba se continua con una segunda en la cual se comprueba que funcione correctamente junto con la parte mecánica. Si esto funciona bien. los motores deberían moverse de la manera esperada.

Tras la instalación se procede a comprobar el correcto funcionamiento de la solución planteada, pudiendo así el usuario establecer que distancia quiere que recorra el scanner en el eje X, en el eje Y, distancia de caracterización, y tiempo de espera entre cada medida. Además, el usuario podrá realizar un ajuste fino para colocar el scanner exactamente donde quiere o necesita. Será necesario verificar que no existe ningún tipo de error en el proceso de escaneado. En caso de irregularidades, se podrán resolver esos fallos especificando donde suceden los mismos.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ESCÁNER 2D PARA APLICACIONES DE IMÁGENES SUB-THZ

Documento 04: Presupuesto

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Miryam Gómez Pijoan

Tutor/a: Salvador Ponce Alcántara

Cotutor/a: Borja Vidal Rodríguez

CURSO ACADÉMICO: 2022/2023

Índice de contenido

1.0. Justificación de precios

Se procede a documentar el presupuesto por precios descompuestos o unidades de obra, que corresponde con la ejecución del presente proyecto de programación de dos motores para el control de movimiento de una mesa XY.

1.1. Precios unitarios

Cuadro de materiales

Materiales					
Ref	Ud	Descripción	Precio	Cantidad	Total
m1	u.	Viga 0824 16 mm	1,25 €	1	1,25 €
m2	u.	Viga 0824 48 mm	2,25 €	4	9,00 €
m3	u.	Viga 0824 80 mm	2,75 €	1	2,75 €
m4	u.	Viga 0824 96 mm	3,25 €	4	13,00 €
m5	u.	Viga 0824 112 mm	3,75 €	2	7,50 €
m6	u.	Viga 0824 496 mm	12,95 €	2	25,90 €
m7	u.	Viga 0808 72 mm	1,31 €	1	1,31 €
m8	u.	Viga 2424 504 mm	12,75 €	2	25,50 €
m9	u.	Placa 3x6	4,95 €	5	24,75 €
m10	u.	Belt Connector	1,65 €	3	4,95 €
m11	u.	Enlace cortable	1,37 €	6	8,22 €
m12	u.	Soporte 3x3	1,71 €	2	3,42 €
m13	u.	Soporte U1	2,35 €	5	11,75 €
m14	u.	Soporte Stepper Motor 42BYG	5,58 €	2	11,16 €
m15	u.	Polea de distribución 18T	4,04 €	6	24,24 €
m16	u.	Correa de distribución 1.3 m	5,26 €	3	15,78 €
m17	u.	Eje Roscado 4x39 mm	0,94 €	1	0,94 €
m18	u.	Eje D 4x56 mm	0,94 €	2	1,88 €
m19	u.	Eje de movimiento lineal D4x80 mm	2,35 €	1	2,35 €
m20	u.	Eje de movimiento lineal D4x512 mm	10,82 €	1	10,82 €
m21	u.	Eje de movimiento lineal D8x496 mm	10,82 €	4	43,28 €
m22	u.	Collar eje 4 mm	0,89 €	10	8,90 €
m23	u.	Aclopamiento flexible 4x4 mm	6,00 €	1	6,00 €
m24	u.	Unidad deslizante de movimiento lineal de 8 mm	4,70 €	6	28,20 €
m25	u.	Rodamiento de brida 4x8x3 mm	1,48 €	10	14,80 €
m26	u.	Tornillo cabeza de botón M4x8 mm	0,001 €	36	0,04 €
m27	u.	Tornillo cabeza de botón M4x14 mm	0,091 €	30	2,73 €
m28	u.	Tornillo cabeza de botón M4x16 mm	0,098 €	28	2,74 €
m29	u.	Tornillo cabeza de botón M4x22 mm	0,10 €	12	1,20 €
m30	u.	Tornillo cabeza de botón M4x30 mm	0,16 €	18	2,88 €
m31	u.	Tornillo avellanado M3x8 mm	0,046 €	10	0,46 €
m32	u.	Tornillo sin cabeza M3x5 mm	0,058 €	26	1,51 €
m33	u.	Tuerca M4	0,06 €	50	3,00 €
m34	u.	Remache de plástico R4060	0,075 €	16	1,20 €
m35	u.	Remache de plástico R4100	0,075 €	6	0,45 €
m36	u.	Anillo de plástico 4x7x2 mm	1,15 €	20	23,00 €
m37	u.	Bridas de nailon	0,023 €	30	0,69 €
m38	u.	Banda elástica	0,024 €	5	0,12 €
m39	u.	Soporte base B	3,75 €	1	3,75 €
m40	u.	Arduino Nano	9,20 €	1	9,20 €
m41	u.	Microstep driver TB6600	10,00 €	2	20,00 €
m42	u.	Micro Switch Me	2,00 €	4	8,00 €
m43	u.	Fuente de alimentación 12 V	16,34 €	1	16,34 €
m44	u.	Cable USB 2.0 A macho a micro B macho	3,75 €	1	3,75 €
m45	u.	Resistencia de 10 k	0,05 €	2	0,10 €
m46	u.	42BYG Stepper Motor	17,95 €	2	35,90 €
				Subtotal	444,71 €

Cuadro de equipos

Maquinaria y equipos						
Ref	Ud	Descripción	Precio	Horas	F. de amortización	Total
s1	h	Fuente de alimentacion regulada	200,00 €	60	0,008242	1,70 €
s2	h	Ordenador portátil	750,00 €	280	0,01832	29,76 €

A continuación, se indica como se ha calculado el factor de amortización:

Fuente de alimentación regulada

Horas de uso: 60 h

Total de horas de trabajo en 1 año: 1820 h

Vida útil: 4 años

Total de horas de uso de la fuente: 7280 h (1820 x 4)

$$Factor\ de\ amortización = \frac{Horas\ de\ uso}{Horas\ de\ trabajo\ de\ la\ fuente} = \frac{60}{7280} = \mathbf{0.008242}$$

Ordenador portátil

Horas de uso: 280 h

Total de horas de trabajo en 1 año: 1820 h

Vida útil: 3 años

Total de horas de uso del ordenador: 5460 h (1820 x 3)

$$Factor\ de\ amortización = \frac{Horas\ de\ uso}{Horas\ de\ trabajo\ en\ 1\ año} = \frac{100}{5460} = \mathbf{0.01832}$$

Una vez obtenido, se calcula el precio multiplicando el factor de amortización por el coste del equipo.

Cuadro de mano de obra

Mano de obra					
Ref	Ud	Descripción	Precio	Horas	Total
h1	h	Técnico de montaje	20,00 €	10	200,00 €
h2	h	Ingeniero electrónico	35,00 €	300	10.500,00 €

1.2. Resumen del presupuesto

	Importe
Capítulo 1 Sistema Mesa XY	**11.176,17 €**
Presupuesto de ejecución material	11.176,17 €
12% de beneficio industrial	1.341,14 €
10% de medios auxiliares sobre costes directos	1.117,62 €
Suma	**13.634,93 €**
21% de IVA	2.863,33 €
Presupuesto de ejecución por contrata	**16.498,26 €**

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SCANNER 2D PARA APLICACIONES DE IMÁGENES SUB-THZ

ANEXO Nº1. FICHAS TÉCNICAS

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTORA: Miryam Gómez Pijoan

Tutor: Salvador Ponce Alcántara

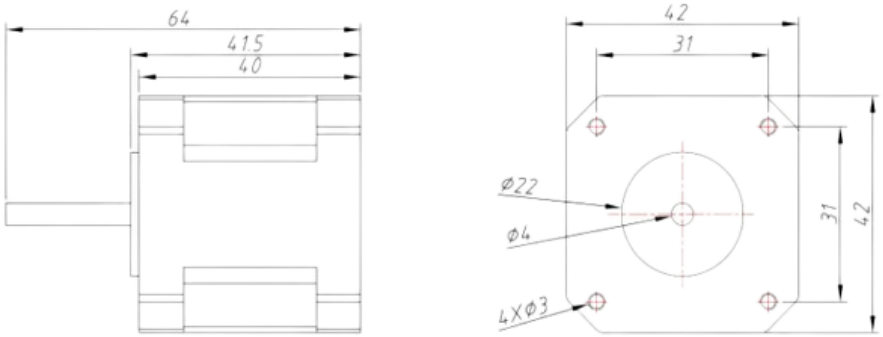
Cotutor: Borja Vidal Rodríguez

CURSO ACADÉMICO: 2022/2023

ÍNDICE ANEXO Nº1

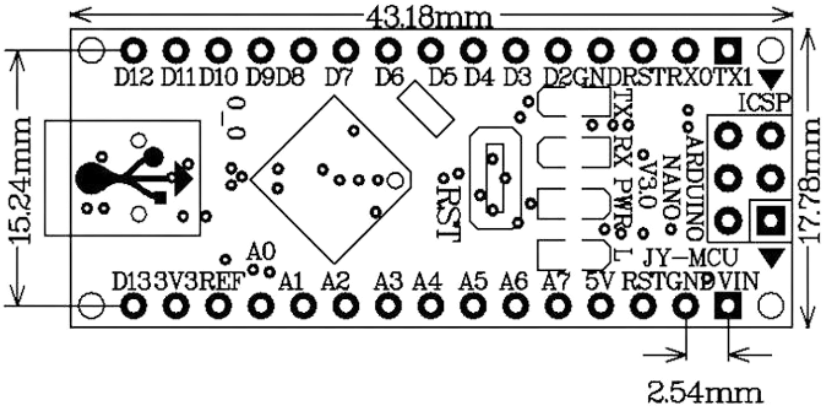
1. Fichas técnicas

1.1. Stepper motor 42BYG



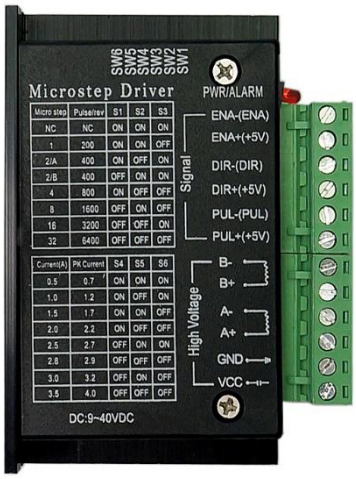
Características	
Número de fases	2
Ángulo de paso	1.8 ± 5% grados
Voltaje de entrada	12 V
Corriente de fase	1.7 A
Resistencia de fase	1.5 ± 10% Ω
Inductancia de fase	2.8 ± 20% mH
Torque de agarre	40 N·cm
Torque detenido	2.2 N·cm
Clase de aislamiento	B
Estilo de plomo	AWG26 UL1007
Torque del rotor	54 G·cm^2

1.2. Placa Arduino Nano



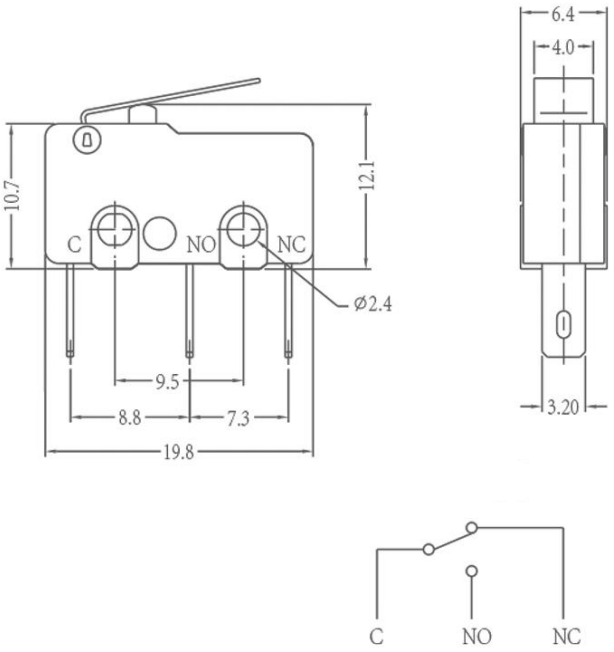
Características	
Microcontrolador	ATmega168/ATmega328P
Voltaje de operación	5 V
Voltaje de alimentación	7-12 V
Velocidad del reloj	16 MHz
Pines de E/S analógicas	8
Pines de E/S digitales	14 (6 son PWM)
E-eprom	1 kB
Memoria Flash	32 kB
SRAM	2 kB
Longitud	43.18 mm
Ancho	17.78 mm
Peso	7 g
Corriente continua para 3.3 V	50 mA
Corriente continua por pin de E/S	20 mA

1.3. Microstep Driver TB6600



Características	
Voltaje de entrada (DC)	8 V – 40 V
Voltaje operativo	12 V – 42 V
Corriente de entrada	1 a 4 A
Corriente de salida (valor nominal máximo absolutos, pico, de 100 ms)	4.0 A
Corriente de salida (rango de operación, valor máximo)	3.5 A
Temperatura de funcionamiento	-10 a 45 ºC
Temperatura de almacenamiento	-40 a 70 ºC
Peso	200 gramos

1.4. Final de carrera HK-04G



Características	
Electrical Rating	250 VAC
Contact Resistance	≤ 50 mΩ
Insulation Resistance	≥ 100 MΩ (Initial Value)
Dielectric Voltage (between non-connected terminals)	500 V / 0.5 mA
Dielectric Voltage (between terminals and the metal frame)	1500 V / 0.5 mA
Electrical Life	≥ 10000 cycles
Mechanical Life	≥ 100000 cycles
Operating Temperature	-25 – 125 ºC
Operating frequency	15 cycles (electrical), 60 cycles (mechanical)
Solder Ability	235 ± 5 ºC
Solder Heat Resistance	260 ± 5 ºC (Dip Soldering), 300 ± 5 ºC (Manual Soldering)
Test conditions	20 ± 5 ºC (ambient temperature) 65 ± 5% RH (Relative Humidity) 86 – 106 kPa (Air Preassure)

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SCANNER 2D PARA APLICACIONES DE IMÁGENES SUB-THZ

ANEXO Nº2. INDICADORES ODS

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTORA: Miryam Gómez Pijoan

Tutor: Salvador Ponce Alcántara

Cotutor: Borja Vidal Rodríguez

CURSO ACADÉMICO: 2022/2023

Índice de contenido

1. Contexto de los Objetivos de Desarrollo Sostenible

La situación empresarial actual está marcada por la Agenda 2030 y los Objetivos de Desarrollo Sostenible (ODS) ya que, la mayoría de las empresas están vinculadas a cumplir con los parámetros que se determinan en esta Agenda.

Los ODS fueron establecidos en 2015 por la Asamblea General de las Naciones Unidas (AG-ONU) y se pretende alcanzarlos para 2030. Están incluidos en una Resolución de la AG-ONU llamada 2030 Agenda o lo que se conoce coloquialmente como Agenda 2030. Los ODS se desarrollaron en la Agenda de Desarrollo después de 2015 como el futuro marco de desarrollo global para suceder a los Objetivos de Desarrollo del Milenio en 2015. Los objetivos declarados a conseguir son 17, y se representan de la siguiente forma:

2. Los ODS en el proyecto

Un proyecto de programación mediante Arduino de los motores paso a paso de una mesa XY podría contribuir a varios Objetivos de Desarrollo Sostenible (ODS), entre ellos:

- **ODS 9: Industria, innovación e infraestructura**. El proyecto involucra la creación de una infraestructura que permite el control preciso de la posición y movimiento, utilizando tecnologías innovadoras como Arduino y motores paso a paso.
- **ODS 11: Ciudades y comunidades sostenibles**. La mesa XY puede ser utilizada en diversas aplicaciones, desde la automatización de procesos de producción hasta el control de dispositivos médicos. Esto puede contribuir a crear comunidades más sostenibles y mejorar la calidad de vida de las personas.

- **ODS 12: Producción y consumo responsables**. El proyecto podría ayudar a reducir el desperdicio y la contaminación al permitir un control más preciso de los procesos de producción, lo que puede reducir el consumo de materiales y energía.
- **ODS 17: Alianzas para lograr los objetivos**. El proyecto podría fomentar la colaboración entre diferentes sectores, como la industria, la academia y la sociedad civil, para abordar los desafíos globales de manera conjunta.

El presente proyecto pretende contribuir con los Objetivos de Desarrollo Sostenible en la mayor medida posible.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

SCANNER 2D PARA APLICACIONES DE IMÁGENES SUB-THZ

ANEXO Nº3. PROGRAMACIÓN DEL SISTEMA

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTORA: Miryam Gómez Pijoan

Tutor: Salvador Ponce Alcántara

Cotutor: Borja Vidal Rodríguez

CURSO ACADÉMICO: 2022/2023

Índice de contenido

1. Programación en Arduino

1.1. Estructura principal

Para poder programar la estructura principal se ha realizado el diagrama de flujo que se muestra en la ilustración 1.

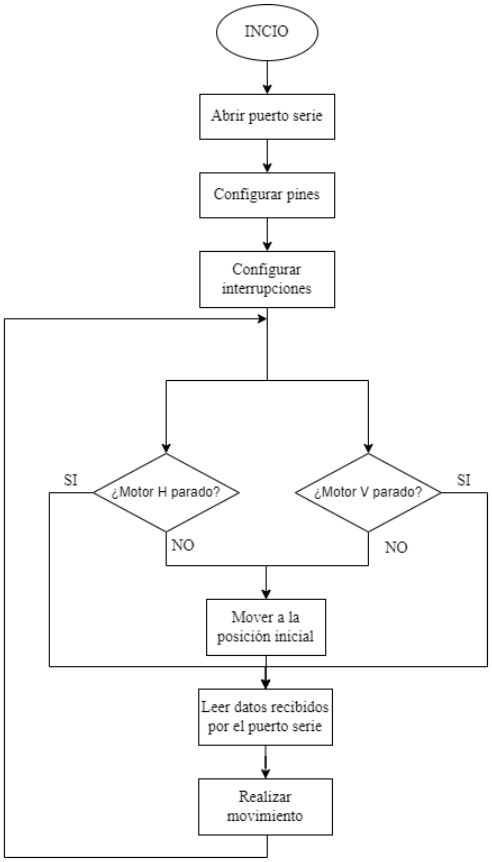


Ilustración 1. Diagrama de flujo programa principal.

En primer lugar, será necesario definir todas las variables que serán necesarias para la ejecución del programa. Las variables mencionadas se muestran a continuación.

```
//Definición de los pines a los que van conectados los motores y finales de carrera
const int dirPinH = 11;
const int stepPinH = 12;
const int dirPinV = 9;
const int stepPinV = 10;
const int fdcV = 2, fdcH = 3;

//Variables númericas necesarias para la ejecución del programa
int stepDelay = 350, giroMotor_int = 0, i = 0;
float dist_paso = 0.02279, num_pasos, Nx, Ny;

//Definición de banderas
volatile bool stopMotorH = false;
volatile bool stopMotorV = false;

//Definición de las cadenas necesarias para almacenar los datos
String inputString = "";
String distX = "";
String distY = "";
String caract = "";
String t_1 = "";
String giroMotor = "";
```

Las variables que cuentan con una H al final de ellas hacen referencia al motor que hace mover el escáner en su eje horizontal y las que tienen una V son las que se refieren al motor que hace que se mueva el escáner verticalmente.

A continuación, es necesaria la función de *void setup*, en la cual figura la configuración de los pines e interrupciones y la instrucción necesaria para abrir el puerto serie de Arduino especificando también su velocidad de transmisión.

```
void setup() {
  //Abrir puerto serie y establecer velocidad de transmisión
  Serial.begin(9600);

  //Configuración pines
  pinMode(dirPinH, OUTPUT);
  pinMode(stepPinH, OUTPUT);
  pinMode(dirPinV, OUTPUT);
  pinMode(stepPinV, OUTPUT);
  pinMode(fdcH, INPUT);
  pinMode(fdcV, INPUT);

  //Configuración interrupciones
  attachInterrupt(digitalPinToInterrupt(fdcV), interruptEndstop2, FALLING);
  attachInterrupt(digitalPinToInterrupt(fdcH), interruptEndstop, FALLING);
}
```

En la función principal de *void loop* se necesitará en primer lugar que el escáner se desplace hasta la posición inicial (0, 0) siendo esta la esquina superior izquierda de la mesa.

Una vez alcanzada, debe esperar a recibir las instrucciones de movimiento que enviará el usuario mediante la interfaz. La función necesaria para leer estos datos se encuentra también en el *void loop*. Una vez leídos, la mesa puede ejecutar el movimiento deseado por el usuario.

```
void loop() {
  //Mover a la posición inicial
  if(!stopMotorH){
    digitalWrite(dirPinH, LOW);
    digitalWrite(stepPinH, HIGH);
    delayMicroseconds(stepDelay);
    digitalWrite(stepPinH, LOW);
    delayMicroseconds(stepDelay);
  }
  if(!stopMotorV){
    digitalWrite(dirPinV, HIGH);
    digitalWrite(stepPinV, HIGH);
    delayMicroseconds(stepDelay);
    digitalWrite(stepPinV, LOW);
    delayMicroseconds(stepDelay);
  }

  //Leer datos recibidos
  if (Serial.available() > 0) {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n') {
      inputString.trim();
      processCommand();
      inputString = "";
    }

    //Realizar movimiento especificado
    Movimiento();
  }
}
```

1.2.Interrupciones

Con tal de poder programar las interrupciones se ha realizado el diagrama de flujo que se muestra en la ilustración 2.

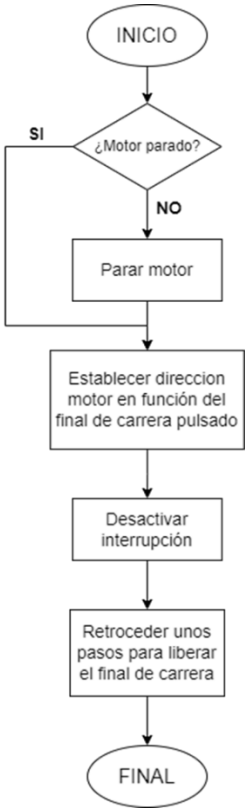


Ilustración 2. Diagrama de flujo asociado a las interrupciones.

Las interrupciones se utilizan para que, cuando realice el movimiento de desplazarse hasta la posición inicial, pare al llegar al final de carrera y no siga avanzando. Una vez alcanza el final de carrera, retrocede unos pocos pasos para que el final de carrera quede liberado y se queda en esa posición hasta recibir nuevas órdenes. Se utilizan las interrupciones de los pines 2 y 3, que son las que incluye la placa de Arduino Nano.

Dado que el Arduino únicamente cuenta con dos interrupciones y hay 4 finales de carrera se va a utilizar una misma interrupción para dos finales de carrera. Además, el escáner no puede estar a la vez en ambos extremos del eje. Esto se puede conseguir leyendo la dirección en la que va el motor y estableciendo la dirección contraria a la que se está moviendo para el retroceso de pasos.

Durante el proceso de medida del escáner las interrupciones estarán desactivadas tal y como se puede ver en el código del programa.

Interrupción para los finales de carrera del eje horizontal:

```
void interruptEndstop() {
  if(!stopMotorH){
    stopMotorH = true;
  //Establece la dirección en funcion del final de carrera alcanzado
  if(digitalRead(dirPinH)== LOW){
    digitalWrite(dirPinH, HIGH);
  }else {digitalWrite(dirPinH, LOW);}
    //Desactiva la interrupción
    detachInterrupt(digitalPinToInterrupt(fdcH));

    //Retrocede unos pocos pasos
    for (int i = 0; i < 90; i++) {
      digitalWrite(stepPinH, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinH, LOW);
      delayMicroseconds(stepDelay);
    }
  }
}
```

Interrupción para los finales de carrera del eje vertical:

```
void interruptEndstop2() {
  if(!stopMotorV){
    stopMotorV = true;
  //Establece la dirección en funcion del final de carrera alcanzado
  if(digitalRead(dirPinV)== HIGH){
    digitalWrite(dirPinV, LOW);
  }else {digitalWrite(dirPinV, HIGH);}
    //Desactiva la interrupción
    detachInterrupt(digitalPinToInterrupt(fdcV));

    //Retrocede unos pocos pasos
    for (int i = 0; i < 90; i++) {
      digitalWrite(stepPinV, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinV, LOW);
      delayMicroseconds(stepDelay);
    }
  }
}
```

1.3. Función para procesar comandos

Con tal de poder programar la función para procesar los comandos introducidos por el usuario (matriz) se ha realizado el diagrama de flujo que se muestra en la ilustración 3.

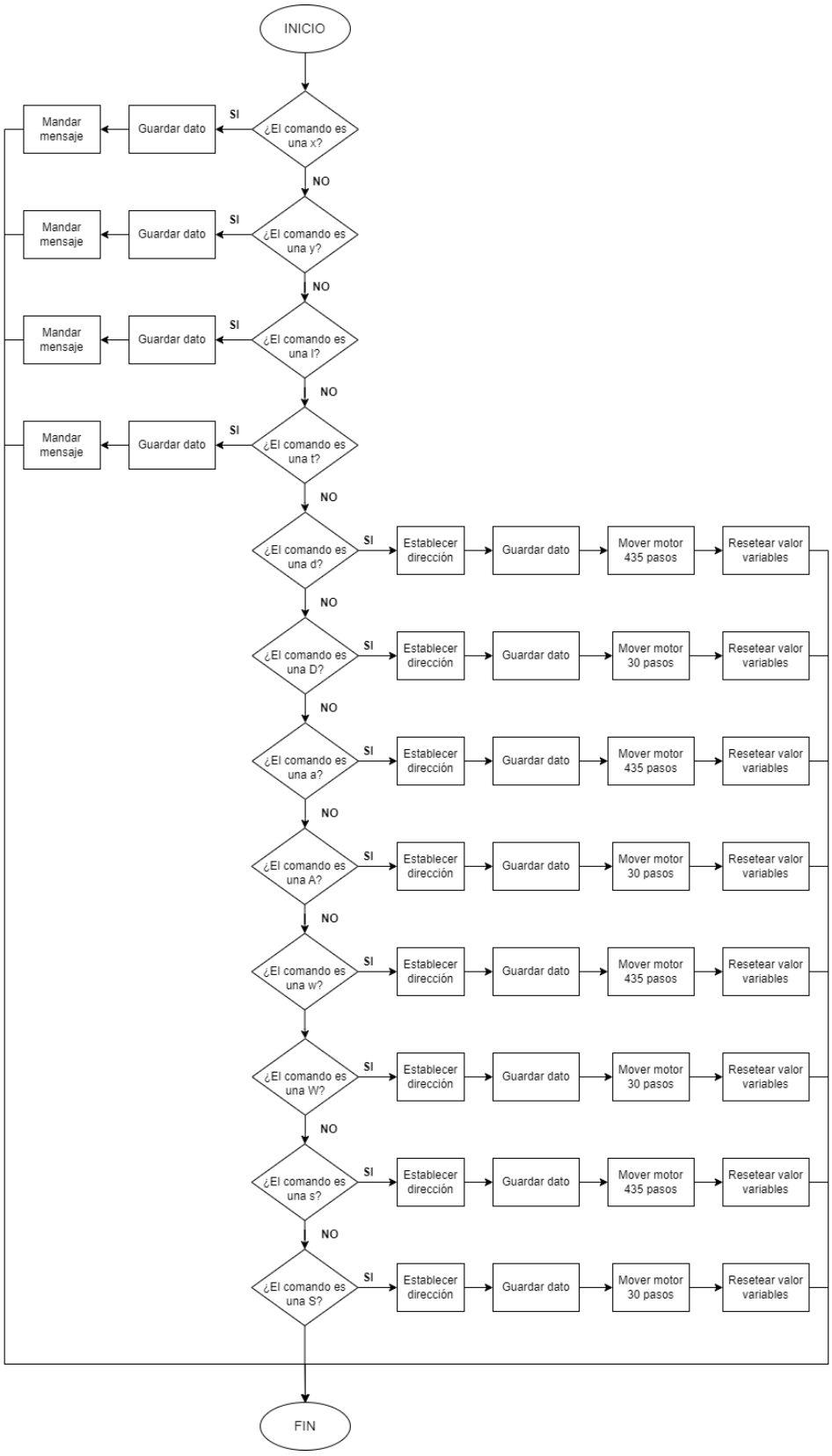


Ilustración 3. Diagrama de flujo procesar comandos.

En esta función se procesan tanto los comandos enviados por las flechas utilizadas para realizar el posicionamiento inicial del escáner, como los datos numéricos necesarios para recorrer la superficie requerida. A continuación, se muestra un listado con las acciones que realiza cada comando y el código implementado para su lectura en Arduino:

- **Comando 'x':** se emplea para identificar que se está recibiendo la distancia que se quiere recorrer en el eje X.
- **Comando 'y'**: se emplea para identificar que se está recibiendo la distancia que se quiere recorrer en el eje Y.
- **Comando 'l'**: se emplea para identificar que se está recibiendo la distancia de separación entre cada medida.
- **Comando 't'**: se emplea para identificar que se está recibiendo el tiempo de espera para la realización de cada medida, será necesario establecer un tiempo suficiente para que el escáner sea capaz de captar las imágenes.

```
void processCommand(){
  //Lectura comandos del movimiento para el escaneo
  if (inputString.startsWith("x")) {
    distX = inputString.substring(1);
    Serial.println("DistX recibida");
  }
  else if (inputString.startsWith("y")) {
    distY = inputString.substring(1);
    Serial.println("DistY recibida");
  }
  else if (inputString.startsWith("l")) {
    caract = inputString.substring(1);
    Serial.println("Caract recibida");
  }
  else if (inputString.startsWith("t")) {
    t_1 = inputString.substring(1);
    Serial.println("t_1 recibida");
  }
```

- **Comando 'd' y 'D':** se utilizan para ajustar inicialmente la mesa con las flechas, la minúscula es para un ajuste grueso y la mayúscula para un ajuste fino. Estos comandos hacen que se desplace hacia la **derecha**.

```
else if (inputString.startsWith("d")) {
  digitalWrite(dirPinH, HIGH); //Establecer dirección derecha
  Serial.println("Derecha");
  giroMotor = inputString.substring(1); //Viene determinado en Visual Studio
  giroMotor_int = giroMotor.toInt(); //Convertir a int

  while(giroMotor_int == 3){
    for(i=0;i<=435;i++){ //435 es el numero de pasos que avanza
      digitalWrite(stepPinH, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinH, LOW);
      delayMicroseconds(stepDelay);
      }
      //Resetear el valor de las variables
      giroMotor = "";
      giroMotor_int = 0;
  }
}
else if (inputString.startsWith("D")) {
  digitalWrite(dirPinH, HIGH); //Establecer dirección derecha
  giroMotor = inputString.substring(1); //Viene determinado en Visual Studio
  giroMotor_int = giroMotor.toInt(); //Convertir a int

  while(giroMotor_int == 1){
    for(i=0;i<=30;i++){ //30 es el numero de pasos que avanza
      digitalWrite(stepPinH, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinH, LOW);
      delayMicroseconds(stepDelay);
      }
      //Resetear el valor de las variables
      giroMotor = "";
      giroMotor_int = 0;
  }
}
```

- **<u>Comando ‘a’ y ‘A’</u>:** se utilizan para ajustar inicialmente la mesa con las flechas, la minúscula es para un ajuste grueso y la mayúscula para un ajuste fino. Estos comandos hacen que se desplace hacia la **izquierda**.

```
else if (inputString.startsWith("a")) {
  digitalWrite(dirPinH, LOW); //Establecer dirección izquierda
  Serial.println("Izquierda");
  giroMotor = inputString.substring(1); //Viene determinado en Visual Studio
  giroMotor_int = giroMotor.toInt(); //Convertir a int

  while(giroMotor_int == 4){
    for(i=0;i<=435;i++){ //435 es el numero de pasos que avanza
      digitalWrite(stepPinH, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinH, LOW);
      delayMicroseconds(stepDelay);
      }
      //Resetear el valor de las variables
      giroMotor = "";
      giroMotor_int = 0;
  }
}
```

```
else if (inputString.startsWith("A")) { //Establecer dirección izquierda
  digitalWrite(dirPinH, LOW);
  giroMotor = inputString.substring(1); //Viene determinado en Visual Studio
  giroMotor_int = giroMotor.toInt(); //Convertir a int

  while(giroMotor_int == 2){
    for(i=0;i<=30;i++){ //30 es el numero de pasos que avanza
      digitalWrite(stepPinH, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinH, LOW);
      delayMicroseconds(stepDelay);
      }
      //Resetear el valor de las variables
      giroMotor = "";
      giroMotor_int = 0;
  }
}
```

- **Comando ‘w’ y ‘W’:** se utilizan para ajustar inicialmente la mesa con las flechas, la minúscula es para un ajuste grueso y la mayúscula para un ajuste fino. Estos comandos hacen que se desplace hacia **delante**.

```
else if (inputString.startsWith("w")) { //Establecer dirección arriba
  digitalWrite(dirPinV, HIGH);
  Serial.println("Arriba");
  giroMotor = inputString.substring(1); //Viene determinado en Visual Studio
  giroMotor_int = giroMotor.toInt(); //Convertir a int

  while(giroMotor_int == 5){
    for(i=0;i<=435;i++){ //435 es el numero de pasos que avanza
      digitalWrite(stepPinV, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinV, LOW);
      delayMicroseconds(stepDelay);
      }
      //Resetear el valor de las variables
      giroMotor = "";
      giroMotor_int = 0;
  }
}
else if (inputString.startsWith("W")) { //Establecer dirección arriba
  digitalWrite(dirPinV, HIGH);
  giroMotor = inputString.substring(1); //Viene determinado en Visual Studio
  giroMotor_int = giroMotor.toInt(); //Convertir a int

  while(giroMotor_int == 6){
    for(i=0;i<=30;i++){ //30 es el numero de pasos que avanza
      digitalWrite(stepPinV, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinV, LOW);
      delayMicroseconds(stepDelay);
      }
      //Resetear el valor de las variables
      giroMotor = "";
      giroMotor_int = 0;
  }
}
```

- **Comando ‘s’ y ‘S’:** se utilizan para ajustar inicialmente la mesa con las flechas, la minúscula es para un ajuste grueso y la mayúscula para un ajuste fino. Estos comandos hacen que se desplace hacia **atrás**.

```
else if (inputString.startsWith("s")) { //Establecer dirección abajo
  digitalWrite(dirPinV, LOW);
  Serial.println("Abajo");
  giroMotor = inputString.substring(1); //Viene determinado en Visual Studio
  giroMotor_int = giroMotor.toInt(); //Convertir a int

  while(giroMotor_int == 7){
    for(i=0;i<=435;i++){ //435 es el numero de pasos que avanza
      digitalWrite(stepPinV, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinV, LOW);
      delayMicroseconds(stepDelay);
      }
      //Resetear el valor de las variables
      giroMotor = "";
      giroMotor_int = 0;
  }
}
 else if (inputString.startsWith("S")) { //Establecer dirección abajo
   digitalWrite(dirPinV, LOW);
   giroMotor = inputString.substring(1); //Viene determinado en Visual Studio
   giroMotor_int = giroMotor.toInt(); //Convertir a int

   while(giroMotor_int == 8){
     for(i=0;i<=30;i++){ //30 es el numero de pasos que avanza
       digitalWrite(stepPinV, HIGH);
       delayMicroseconds(stepDelay);
       digitalWrite(stepPinV, LOW);
       delayMicroseconds(stepDelay);
       }
       //Resetear el valor de las variables
       giroMotor = "";
       giroMotor_int = 0;
   }
 }
 delay(50); //Para que le de tiempo a procesar los datos
}
```

1.4. Función para recorrer la matriz

Con tal de poder programar la función para recorrer la matriz introducida por el usuario se ha realizado el diagrama de flujo que se muestra en la ilustración 4.

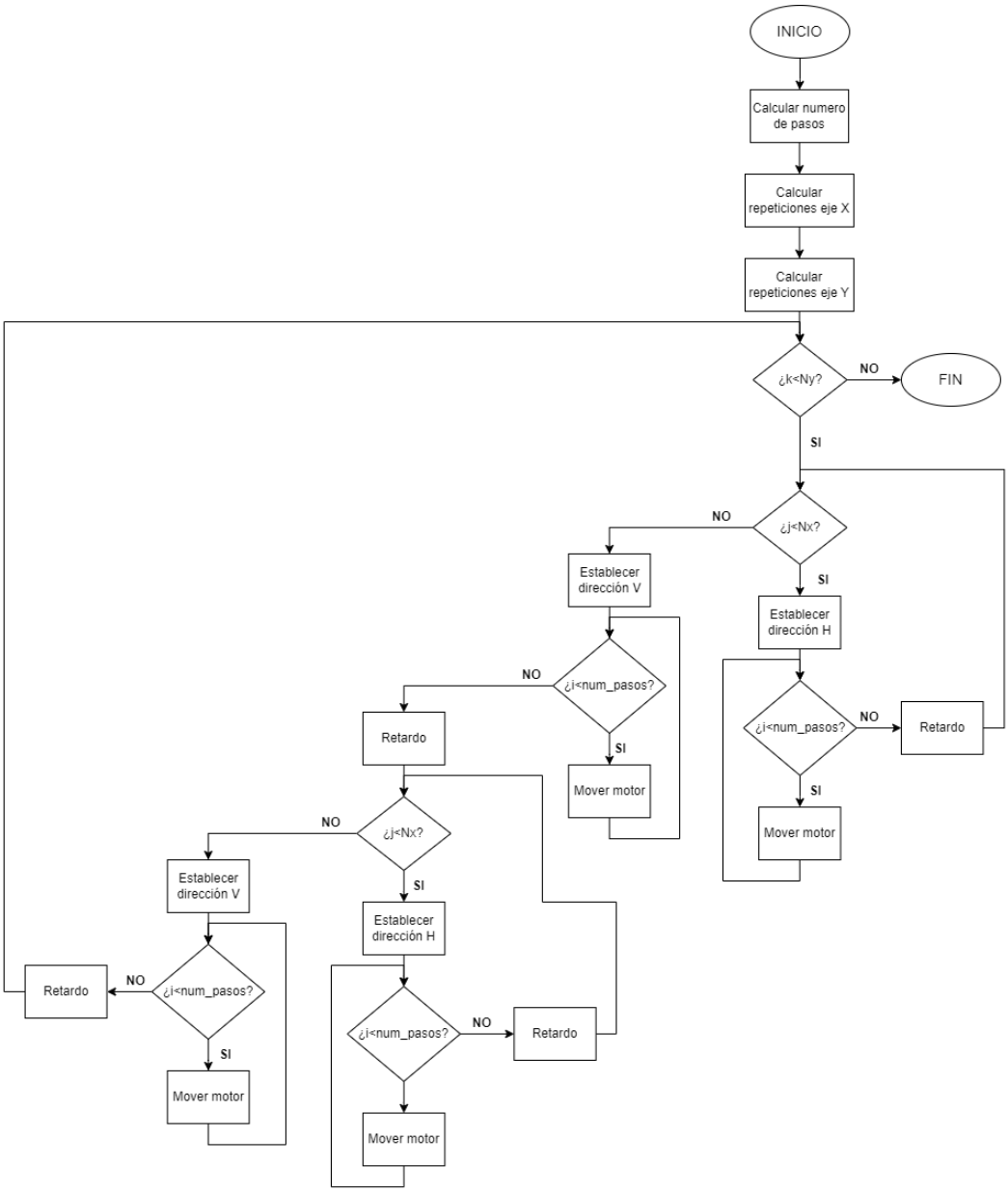


Ilustración 4. Diagrama de flujo función movimiento.

Esta función se encargará de recorrer la superficie indicada mediante el usuario, haciendo las medidas a la distancia especificada y esperando el tiempo necesario para efectuar cada medida.

Está programada mediante bucles *for* anidados que le posibilitan realizar los movimientos necesarios mientras las repeticiones sean menores o iguales que las necesarias para alcanzar la distancia total que se quiere medir.

Para saber el número de pasos que tiene que dar para recorrer la distancia de caracterización, será necesario dividir dicha longitud entre la distancia que se recorre en un paso. Y para el numero de repeticiones habrá que dividir la distancia total que se quiere recorrer en cada eje entre la distancia de caracterización.

La programación de la función mencionada se muestra a continuación.

```
void Movimiento(){
  //Número de pasos que tiene que dar para recorrer la distancia de caracterización
  num_pasos = round(caract.toFloat()/dist_paso);
  //Número de repeticiones necesarias para llegar a la distancia deseada
  Nx = round(distX.toFloat()/caract.toFloat());
  Ny = round(distY.toFloat()/caract.toFloat());

  for(int k=0; k<Ny; k++){
  for(int j=0; j<Nx; j++){
    //Movimiento en el eje X
    digitalWrite(dirPinH, HIGH);
    for(int i=0; i<num_pasos; i++){
      digitalWrite(stepPinH, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinH, LOW);
      delayMicroseconds(stepDelay);
    }
    delay(t_1.toFloat()*1000);
  }
  //Movimiento en el eje Y
  digitalWrite(dirPinV, LOW);
  for(int i=0; i<num_pasos; i++){
    digitalWrite(stepPinV, HIGH);
    delayMicroseconds(stepDelay);
    digitalWrite(stepPinV, LOW);
    delayMicroseconds(stepDelay);
  }
  delay(t_1.toFloat()*1000);


  for(int j=0; j<Nx; j++){
    //Movimiento en el eje X
    digitalWrite(dirPinH, LOW);
    for(int i=0; i<num_pasos; i++){
      digitalWrite(stepPinH, HIGH);
      delayMicroseconds(stepDelay);
      digitalWrite(stepPinH, LOW);
      delayMicroseconds(stepDelay);
    }
    delay(t_1.toFloat()*1000);
  }
  //Movimiento en el eje Y
  digitalWrite(dirPinV, LOW);
  for(int i=0; i<num_pasos; i++){
    digitalWrite(stepPinV, HIGH);
    delayMicroseconds(stepDelay);
    digitalWrite(stepPinV, LOW);
    delayMicroseconds(stepDelay);
  }
  delay(t_1.toFloat()*1000);
  }
}
```

2. Programación en Visual Studio

2.1. Configuración puerto serie

Para poder comunicarse con el Arduino será necesario añadir un elemento llamado *serialPort* y configurarlo correctamente.

Cuando se ejecuta el *form* será necesario abrir el puerto serie e indicarle a que puerto del ordenador está conectado, en este caso será el COM4, si no se puede abrir el puerto se enviará un mensaje al usuario. Una vez se cierre el *form* el puerto serie tendrá que cerrarse.

```
1 referencia
private void Form1_Load(object sender, EventArgs e)
{
    try
    {
        serialPort1.PortName="COM4";
        serialPort1.Open();

        //Para poder enviar mensajes desde el Arduino al Visual Studio
        serialPort1.DataReceived += serialPort1_DataReceived;
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message);
    }
}

1 referencia
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (serialPort1.IsOpen)
    {
        try
        {
            serialPort1.Close();
        }
        catch (Exception error)
        {
            MessageBox.Show(error.Message);
        }
    }
}
```

Por otra parte, se ha programado una función en Visual Studio para que se muestre en la pantalla información procedente de Arduino.

```
1 referencia
private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    string receivedData = serialPort1.ReadLine(); // Lee la linea recibida
                                                  // Actualiza los controles en el hilo principal de la interfaz de usuario
    this.Invoke(new Action(() =>
    {
        // Muestra el valor recibido en un control de texto
        pantalla.Text = receivedData;
    }));
}
```

Para ello se emplea el elemento *label*.

2.2. Ajuste manual

Los elementos de esta parte de la interfaz de usuario están agrupados en una ventana de trabajo (*GroupBox*). Cuenta con unas flechas para hacer un ajuste inicial y mover el escáner a la posición deseada. Además, incluye una casilla que marcándose permite ajustar de una manera más fina la posición. Lo descrito anteriormente se muestra en la ilustración 5.

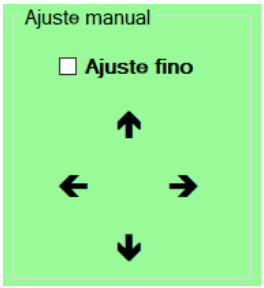


Ilustración 5. Ventana de trabajo asociada al ajuste manual de la posición inicial del escáner.

El código correspondiente para el control de las flechas se muestra a continuación, donde podrá verse que para cada flecha se envía una letra diferente. A su vez, si se ha seleccionado la casilla de ajuste fino, se envía otra distinta. Dichas letras, ya indicadas en el programa de Arduino (punto 1.4), se envían con un número, el cual interpretará el Arduino.

Código de la flecha para delante:

```
1 referencia
private void botArriba_Click(object sender, EventArgs e)
{
    try
    {
        if(checkBox1.Checked)
        {
            serialPort1.WriteLine($"W{6}");
        }
        else
        {
            serialPort1.WriteLine($"w{5}");
        }
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message);
    }
}
```

Código flecha para atrás:

```
1 referencia
private void botAbajo_Click(object sender, EventArgs e)
{
    try
    {
        if (checkBox1.Checked)
        {
            serialPort1.WriteLine($"S{8}");
        }
        else
        {
            serialPort1.WriteLine($"s{7}");
        }
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message);
    }
}
```

Código flecha para la derecha:

```
1 referencia
private void botDer_Click(object sender, EventArgs e)
{
    try
    {
        if (checkBox1.Checked)
        {
            serialPort1.WriteLine($"D{1}");
        }
        else
        {
            serialPort1.WriteLine($"d{3}");
        }
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message);
    }
}
```

Código flecha para la izquierda:

```
1 referencia
private void botIzq_Click(object sender, EventArgs e)
{
    try
    {
        if (checkBox1.Checked)
        {
            serialPort1.WriteLine($"A{2}");
        }
        else
        {
            serialPort1.WriteLine($"a{4}");
        }
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message);
    }
}
```

2.3. Parámetros de caracterización

Los elementos de esta parte de la interfaz de usuario están agrupados en una ventana de trabajo (*GroupBox*) diferente al de las flechas. Este *GroupBox* incluye las diferentes casillas para introducir los datos necesarios para el movimiento y los botones para enviar estos datos. Lo mencionado se muestra en la ilustración 6.

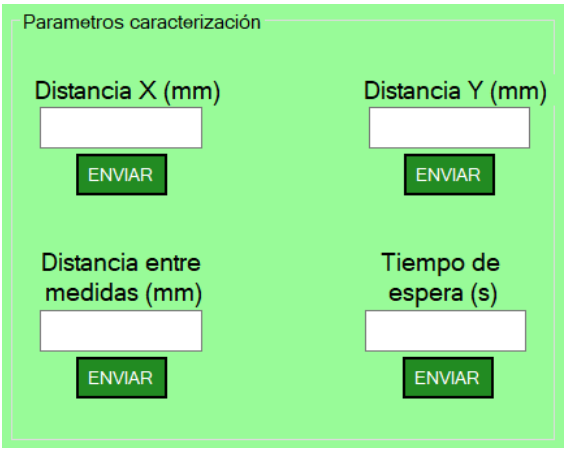


Ilustración 6. Ventana de trabajo relativa a los parámetros de caracterización.

Las diferentes etiquetas muestran en que unidades tienen que introducirse los datos en las casillas para que el Arduino pueda procesarlos correctamente.

El código utilizado en la programación de los botones es el siguiente:

```
1 referencia
private void button1_Click(object sender, EventArgs e)
{
    string x = textBox1.Text;
    serialPort1.WriteLine($"x{x}");
}

1 referencia
private void button2_Click(object sender, EventArgs e)
{
    string y = textBox2.Text;
    serialPort1.WriteLine($"y{y}");
}

1 referencia
private void button3_Click(object sender, EventArgs e)
{
    string l = textBox3.Text;
    serialPort1.WriteLine($"l{l}");
}

1 referencia
private void button4_Click(object sender, EventArgs e)
{
    string t = textBox4.Text;
    serialPort1.WriteLine($"t{t}");
}
```

Para cada botón se guarda el dato introducido por el usuario en una variable y posteriormente se manda un comando identificativo junto con el valor al Arduino. El comando identificativo se pone para que el Arduino pueda identificar que dato se le está enviando y pueda almacenarlo en la variable correcta.

2.4. Diseño gráfico

Tal y como se ha mencionado anteriormente se pueden observar los dos *GroupBox* con todos los elementos junto con la pantalla por donde se avisa al usuario de que los datos se han recibido correctamente.

La interfaz de usuario completa se muestra en la ilustración 7.

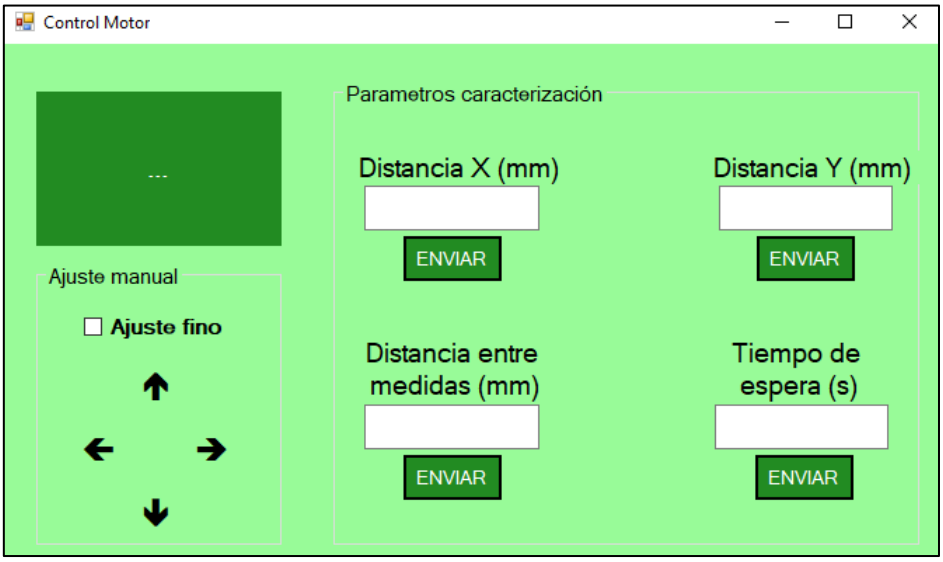


Ilustración 7. Interfaz de usuario completa.

2.4.1. Diseño gráfico del ajuste manual

Dentro de la parte de ajuste manual pueden encontrarse cuatro botones que sirven para ajustar inicialmente el escáner y colocarlo en la posición que el usuario quiera. Para que aparezcan las flechas en los botones será necesario cambiar la propiedad *Text* de la ventana de propiedades y si quiere ajustarse el tamaño o tipo de letra de estas flechas habrá que modificar la propiedad *Font* (ilustración 8). Cambiando la propiedad *Name* se le puede dar un nombre identificativo a cada botón con tal de que sea más fácil identificarlos posteriormente en el código (ilustración 9).

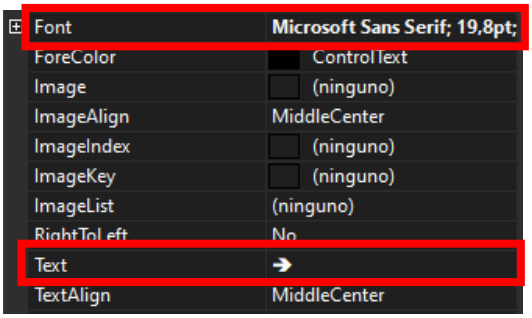


Ilustración 8. Propiedades de las teclas de posicionamiento del escáner (a) *Font* y *Text*.

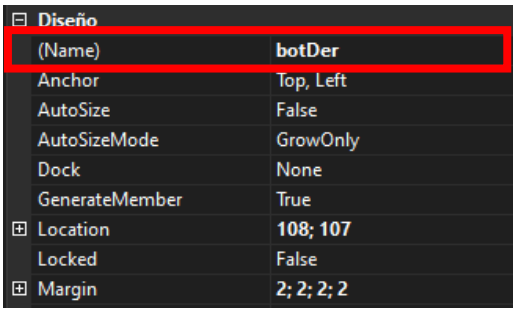


Ilustración 9. (b) *Name*.

Por otro lado, desplegando la opción *FlatAppearance* y asignando un color diferente al del fondo en *MouseOverBackColor* (ilustración 10) se puede hacer que cuando el usuario pase el ratón por encima del botón este cambie de color, así el usuario sabe con certeza si está presionando bien el botón (ilustración 11).

Ilustración 10. Propiedad *MouseOverBackColor*.

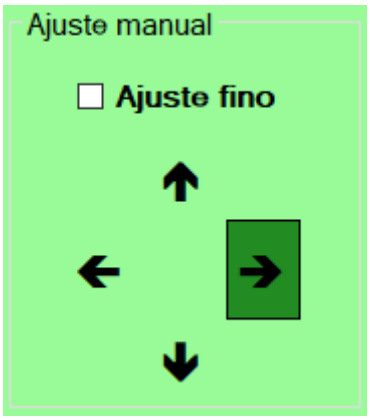


Ilustración 11. Ejemplo cambio de color.

Por último, en este *GroupBox* también está la casilla de verificación (*CheckBox*) que, como ya se ha mencionado anteriormente, se emplea para realizar un ajuste más fino si está marcada.

2.4.2. Diseño gráfico de los parámetros de caracterización

Dentro de la parte de parámetros de caracterización se encuentran las etiquetas (*Label*). Estas indican que dato debe introducirse en cada casilla y con qué unidades. En este caso, solo será necesario cambiar la propiedad *Text* tal y como se ha mencionado anteriormente. Adicionalmente, si el tamaño de letra es muy pequeño puede aumentarse en la propiedad *Font* (ilustración 12).

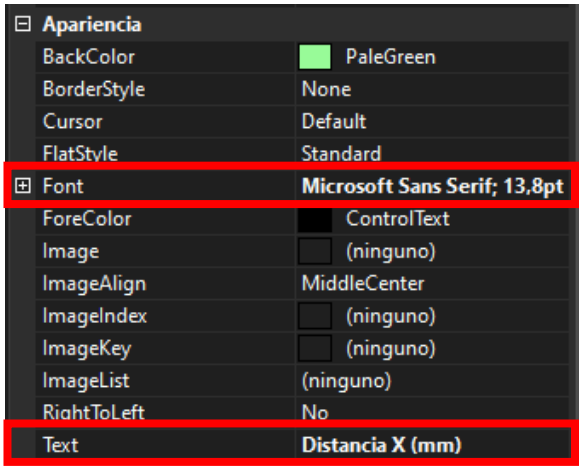


Ilustración 12. Propiedades *Text* y *Font* de *Label*.

En la ventana de trabajo asociada a los parámetros de caracterización también se tienen los campos de texto (*TextBox*). Este elemento permite al usuario introducir un dato con el teclado. Los únicos aspectos que se han modificado son las propiedades *Text* para eliminar el texto que viene por defecto, y *Font* para el tamaño de letra (ilustración 13).

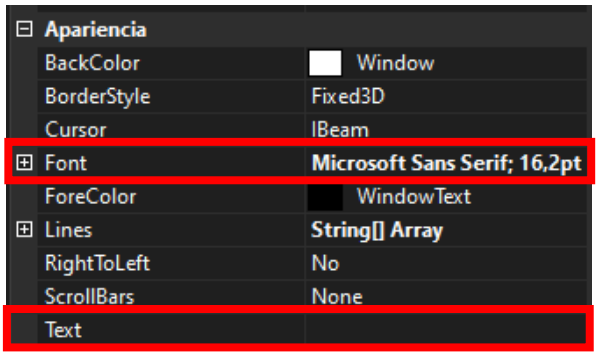


Ilustración 13. Propiedades *Text* y *Font* de *TextBox*.

Finalmente, los botones utilizados para indicar el envío de la información a Arduino funcionan igual que los botones de flechas mencionados anteriormente. La única diferencia es que se le ha incluido un borde al botón. Este borde se puede añadir también desde la pestaña *FlatAppearance* (ilustración 14).

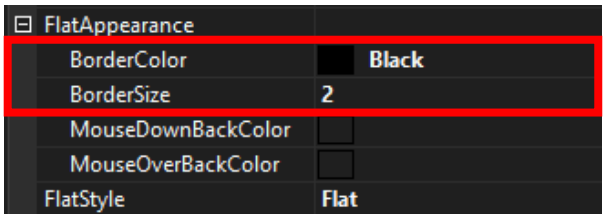


Ilustración 14. Propiedad *FlatAppearance* asociada a los pulsadores presentes en la ventana de trabajo.