UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Automatización de soldadora industrial para blindaje de lámina utilizando el método de soldadura de arco metálico con gas inerte

Trabajo de graduación en modalidad de Tesis presentado por
Erick Alejandro Saucedo Morales
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Automatización de soldadora industrial para blindaje de lámina utilizando el método de soldadura de arco metálico con gas inerte

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Automatización de soldadora industrial para blindaje de lámina utilizando el método de soldadura de arco metálico con gas inerte

Trabajo de graduación en modalidad de Tesis presentado por
Erick Alejandro Saucedo Morales
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,
2018

Vo.Bo.:

(f) _____

MAEB. Otto Girón

Tribunal Examinador:

(f) _____

MAEB. Otto Girón

(f) _____

Msc. Carlos Esquit

(f) _____

Ing. Kurt Kellner

Fecha de aprobación: Guatemala, 4 de Diciembre del 2018.

Agradecimientos

Primero que nada doy gracias a mi padre y madre, por brindarme siempre apoyo a lo largo de mi carrera universitaria y de no ser por la confianza que me tienen para poder culminar mis estudios, también agradezco a mis hermanos por servirme de guía para orientarme a un ámbito más profesional, agradezco a mi hermana por estar siempre ahí por mí y por ser una segunda madre para mí.

Agradezco a Antonio MacKenney por poderme dar la oportunidad de elaborar mis prácticas profesionales en su empresa y por la oportunidad de presentar un proyecto propio de su empresa para ser presentado en éste trabajo de graduación. También agradezco por todo el apoyo y orientación que me dio a lo largo del año 2018 con respecto al proyecto y respecto a realidad como profesional en el ámbito laboral.

Agradezco a mi compañero de carrera Jose Sagastume, ya que él me ayudó a aplicar a la realización de mis prácticas en su lugar de trabajo, lugar en el cuál realicé mi proyecto de trabajo de graduación.

Agradezco al Ing. Estuardo Mancio por todos los consejos y guías, tanto de vida como de estudio, a lo largo de mi carrera universitaria para poder seguir adelante y llegar hasta este punto de mi vida.

Agradezco a mis compañeros, por toda la ayuda y apoyo que me brindaron en los cursos que se me dificultaban y por todo el tiempo de estudio invertido para poder llegar lejos.

Le doy gracias a uno de mis mejores amigos, Estib Soto por siempre apoyarme en todo lo que hay que hacer, con respecto a la Universidad como en la vida. Le doy gracias por esa paciencia que posee y por ser una gran persona y futuro profesional. Como también le agradezco a su familia por ser de una manera humilde y servicial en todo momento.

En fin gracias a todas las personas que me apoyaron, y agradezco a las personas que creyeron en mí y en mi capacidad de llegar hasta tan lejos, sin dejarme caer.

Índice

Lista de figuras

Lista de cuadros

Resumen

El proyecto consiste en el diseño e implementación de la automatización de una máquina soldadora industrial utilizada para el blindaje de láminas por medio del método de soldadura a gas y arco metálico produciendo un resultado final de bajo costo.

Siendo esta dividida por módulos, que llevan un orden en los cuales se requiere tener en cuenta en que por más que sea trabajoso un módulo, al final la integración de todo es lo que en realidad hace a la máquina como una herramienta potente para el desarrollo y crecimiento laboral para las personas.

El siguiente proyecto consiste en el diseño y programación del movimiento por medio de ingreso de parámetros definidos y totalmente modificables, para así realizar una automatización del proceso del blindaje de lámina.

También como parte importante de este trabajo es que se requiere del control de potencia de de unas antorchas de soldadura para poder lograr una soldadura de blindaje uniforme durante todo el recorrido, por medio del uso de sensores de corriente y la implementación de microcontroladores ATmega2560 y ATmega328P como los principales componentes que dan movimiento y control a toda la maquinaria de tipo industrial.

En este trabajo de graduación se tratan temas de diversas disciplinas de la rama de la ingeniería, ya que se debe de tener en cuenta que un buen diseño de un panel de operador en conjunto un buen diagrama de conexiones y un cableado estructurado se puede llegar a obtener resultados satisfactorios.

Como en este caso fue, el cumplimiento de todos los objetivos definidos y obteniendo resultados satisfactorios con respecto al diseño de elementos, como el panel y una caja reductora con freno mecánico incorporado.

Los resultados finales se consideraron satisfactorios, ya que la variación de los datos obtenidos de la experimentación, no fue de gran magnitud y se recomienda tomar en cuenta que la integración completa de los principales funcionamientos no fue propuesta en este trabajo, dejando las puertas abiertas a la persona que desee retomarlo y realizar dicha implementación realizando pruebas de operación en tiempo real ya en producción de las láminas blindadas de bajo costo.

CAPÍTULO 1

Introducción

La soldadura se remonta ya varios años atrás, haciendo su aparición desde la edad de bronce y la edad de hierro por la región de Europa y Oriente Medio. Se sabe que la soldadura fue utilizada en el año 310, en la contracción del Pilar de hierro de Delhi en la India.

La soldadura es actualmente utilizada de muchas maneras y conforme la tecnología va avanzando van apareciendo nuevos métodos para realizar esta acción para unir piezas de metal.

De la mano con la automatización de procesos, y la soldadura automatizada o robotizada es un campo que los innovadores han utilizado para muchas aplicaciones, utilizando como método la soldadura por arco.

En el siguiente trabajo se tratará el tema de una automatización de soldadora industrial para blindaje de lámina utilizando el método de soldadura de arco metálico con gas inerte. En el cual se emprende el proyecto de tomar el proceso de el blindaje de lámina por método de soldadura MIG, elaborado de forma manual, por medio de un solo eje de movimiento lineal, un motor de corriente alterna y una palanca de mando, para realizar una serie cordones de soldadura paralelas para así proporcionarle un revestimiento de soldadura a una lámina metálica y así poder incrementar su resistencia ante esfuerzos externos, a un precio moderado a comparación de una lamina blindada de fábrica.

En este documento se podrá encontrar todo lo necesario para poder comprender la necesidad de la automatización al realizar el proyecto de implementar dos ejes de movimiento en las direcciones necesarias, el uso de motores paso a paso como nueva fuente de transmisión de movimiento y eficientizar el proceso por medio de un control automático de movimiento y depósito del material como soldadura sobre la lámina a blindar por medio de el ingreso de parámetros modificables.

El proyecto se dividió en dos módulos principales, como lo es únicamente la implementación del movimiento de los motores según los parámetros ingresados y el control de la potencia, a través de la corriente, generada por la máquina de soldar para mantener una

soldadura uniforme a lo largo de un trayecto. Trabajando los módulos por aparte dejando la oportunidad abierta para que otro estudiante que desee optar por un trabajo de graduación retome el proyecto para realizar la integración de éstos dos módulos mencionados.

Para lograr cumplir estos dos módulos se llevó a cabo una serie de actividades para lograr lo deseado.

CAPÍTULO 2

Objetivos

2.1. Objetivo general

Diseñar e implementar la automatización de una soldadora industrial especializada para el blindaje de lámina utilizando el método de soldadura a gas y arco metálico. La cual consiste en el movimiento automático de las antorchas por medio de los diferentes ejes a implementar a través de la introducción de diferentes parámetros por parte del usuario.

2.2. Objetivos específicos

1. Diseñar y programar microcontroladores para ser utilizados como controladores para motores paso a paso y codificadores con funciones predefinidas con parámetros modificables. Los parámetros deben de tomar en cuenta, diámetro del hilo metálico, velocidad y distancia en los ejes X y Y y potencia de las antorchas para lograr los diversos resultados deseados por el operador.
2. Diseñar y programar microcontroladores para detectar los cambios de potencia de la antorcha debido a los cambios en la deflexión de la lámina causados por la temperatura y mantener una potencia constante a lo largo de todo el blindaje.
3. Diseñar y elaborar paneles de operador de fácil uso y amigable al usuario.
4. Diseñar y realizar cableado estructurado del gabinete de control y gabinete de potencia.
5. Elaboración de planos eléctricos de la implementación de la red de microcontroladores maestro-esclavos y las conexiones del panel de operador hacia el tablero principal.
6. Diseño y elaboración de elementos de una caja reductora con freno mecánico para todos los ejes a implementar en la máquina.

CAPÍTULO 3

Justificación

La motivación principal para la realización de este trabajo es el lograr eficientizar un proceso industrial que sea de bajo costo, debido a que las láminas blindadas en Guatemala presentan un costo elevado, al realizar este proyecto se obtendrá una lámina que presenta características de resistencia similares pero el costo de producción es menor y así mismo permitir que el producto sea más accesible a las personas que requieran del mismo.[19]

CAPÍTULO 4

Marco teórico

4.1. Automatización industrial

La automatización consta en la utilización de nuevas tecnologías de manera específica, las cuales tienden a monitorear y visualizar en tiempo real y de forma detallada lo que está aconteciendo durante el proceso de un producto.

Generalmente la automatización está dedicada el control y monitoreo de procesos industriales, máquinas o dispositivos que suelen realizar funciones repetitivas haciendo que éstas trabajen de manera automática reduciendo así en mayor cantidad la intervención del ser humano.

Los seres humanos desde la antigüedad han requerido de la creación de mecanismos o máquinas para sustituir el esfuerzo propio, haciendo que por medio de éstas realizar de una mejor manera los trabajos deseados. En 1801 durante la revolución de la industria textil fue donde se generó este avance de la ingeniería, con el objetivo de poder aumentar la producción, mejorar la calidad y evitar los riesgos hacia las personas.[1]

Los sistemas de automatización pueden ser parciales o completos, siempre teniendo una adaptación a una aplicación en específico. Cabe destacar que la automatización se puede dar desde una simple máquina hasta un sistema encargado de una planta o industria completa.

La automatización se encarga de llevar a cabo un control mejorado del proceso de productos, siempre teniendo en cuenta que los parámetros y/o lineamientos del proceso de fabricación y el equipo en sí se deben de mantener una alineación, para que el trabajo a realizar sea el más óptimo.[6]

Según Clextral (2018), la automatización es una herramienta indispensable para los operación, debido a que les permite monitorear de una manera más efectiva el proceso de producción.

Un sistema automatizado se caracteriza por dos partes principales: Parte operativa y

Parte de mando, como se explica en la Tabla4.1.

Sistema automatizado	
Clasificación	**Descripción**
Parte operativa	Parte que actúa directamente sobre la máquina. Son los elementos que hacen que la máquina se mueva y realice la operación deseada. Los elementos que forman la parte operativa son los accionadores de las máquinas como motores, cilindros, compresores y los captadores como foto-diodos, finales de carrera
Parte de mando	Suele ser un autómata programable (tecnología programada), aunque hasta hace poco se utilizaban relevadores electromagnéticos, tarjetas electrónicas o módulos lógicos neumáticos (tecnología cableada) . En un sistema de fabricación automatizado el autómata programable esta en el centro del sistema. Este debe ser capaz de comunicarse con todos los constituyentes de sistema automatizado.

Tabla 4.1: Partes principales de un sistema automatizado.[12]

4.2. Soldadura

Soldar consiste en reunir las partes integrantes de una construcción asegurando la continuidad del material entre ellas. Para realizar este proceso generalmente se requiere intercalar un material fundido el cual tiene la función de rellenar el espacio que existe entre las piezas a unir, como se puede ver en la Figura 4.1.

Los partícipes en la soldadura son el material base, conformado por las piezas a unir, y el material de aportación, el material fundido que se aplica en el espacio existente entre las piezas. Para realizar una soldadura no es necesario la aportación adicional de material, como se ve en la Figura4.2, en donde el mismo material base es fundido por medio de una llama para rellenar el espacio deseado para que exista continuidad del material.[5]

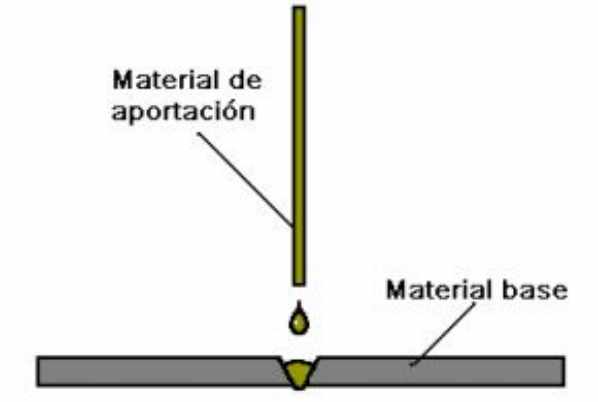


Figura 4.1: Soldadura con material de aportación.[24]

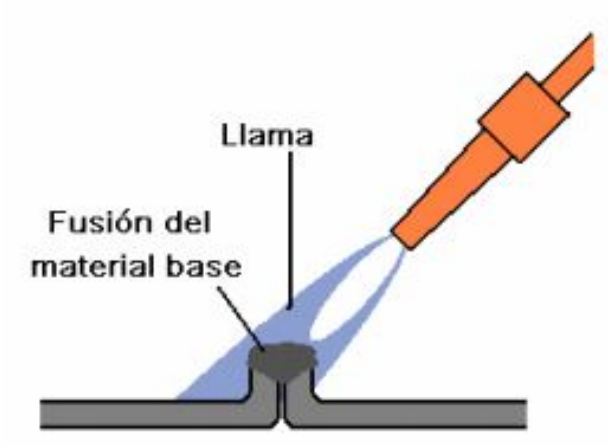


Figura 4.2: Soldadura sin material de aportación.[24]

La soldadura se puede clasificar en dos grupos: heterogéneas y homogéneas:

4.2.1. Soldadura heterogénea

En este tipo de soldadura el material base y el material de aportación son de diferente naturaleza, tomando en cuenta que las dos piezas a unir, también pueden ser del mismo material, como se puede observar en la Figura 4.3. Esta soldadura consiste en fundir el material de aportación y depositarlo entre las dos piezas a unir, mientras que estas permanecen en estado sólido en todo el proceso.[4]

Figura 4.3: Piezas unidas por medio de soldadura heterogénea.[24]

Para este tipo de soldadura se pueden clasificar dos procesos de soldadura en los cuales se toma en cuenta la combinación de materiales de aportación.

Soldadura blanda

Consiste en uniones metálicas en donde el material de aportación posee un menor punto de fusión que el del material base, el material de aportación se funde a una temperatura menor a 450^{o}C.[14]

Soldadura fuerte

También denominada soldadura amarilla, este tipo de soldadura consiste en la unión de dos piezas mediante la fundición del material de aportación, el cual posee un punto de fusión elevado. Cabe destacar que la temperatura para llegar al punto de fusión del material de aporte debe de ser menor al del material base a unir. Esta temperatura debe de ser mayor a los 450ºC para la clasificación de este tipo de soldadura.[9]

4.2.2. Soldadura homogénea

En la soldadura homogénea el material base y el material de aportación, si se implementa, son de la misma naturaleza, como se puede observar en la Figura 4.4 En este proceso se realiza la fundición de los bordes de las piezas a unir mientras se añade el material de aportación, fundido de la misma manera. Durante la fusión se realiza una mezcla entre el material base y el material de aportación, debido a que son de la misma naturaleza. Cuando esta mezcla se encuentra en estado líquido, se le denomina baño de fusión. Luego de solidificado el baño de fusión, se formará el cordón de soldadura como se puede ver en la Figura 4.5.[4]

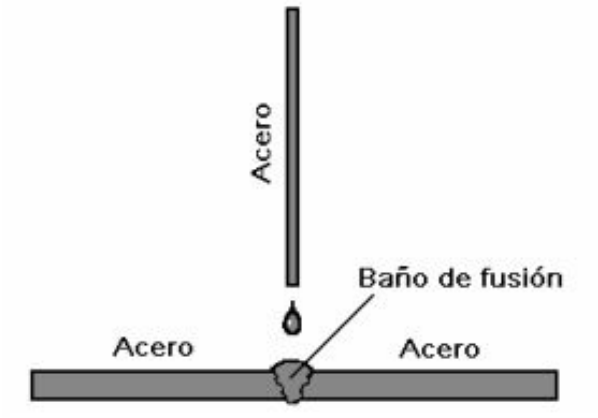


Figura 4.4: Ejemplo de soldadura homogénea.[24]

Figura 4.5: Piezas unidas por medio de soldadura homogénea. [24]

Para la soldadura homogénea existen varios procesos característicos, los cuales son:

Soldadura con llama

También denominada soldadura oxiacetilénica, esta soldadura consiste en soldar por fusión en donde la temperatura necesaria es proporcionada por la combustión de un gas combustible y un gas comburente, los gases generalmente utilizados son el oxígeno y el acetileno como se puede ver en la Figura 4.6.[4]

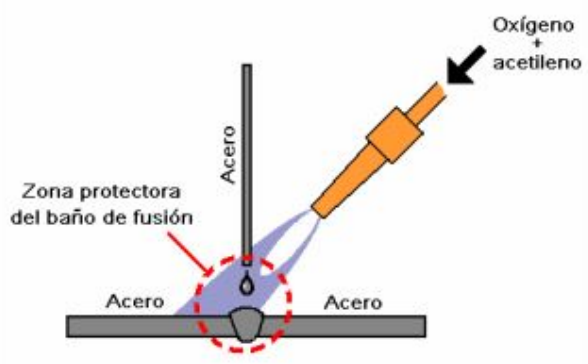


Figura 4.6: Soldadura oxiacetilénica. [24]

Soldadura por arco

En el método de soldadura por arco la energía se obtiene al generarse un arco eléctrico entre el electrodo y la pieza como se puede ver en la Figura 4.7. Estos pueden ser con o sin aplicación de presión y con o sin metal de aporte. La energía eléctrica que entra al sistema es transformada en energía térmica permitiendo llegar hasta una temperatura de aproximadamente 4000^{o}C. El arco eléctrico es un flujo continuo de electrones a través de un medio gaseoso, en el cual se genera calor y luz.

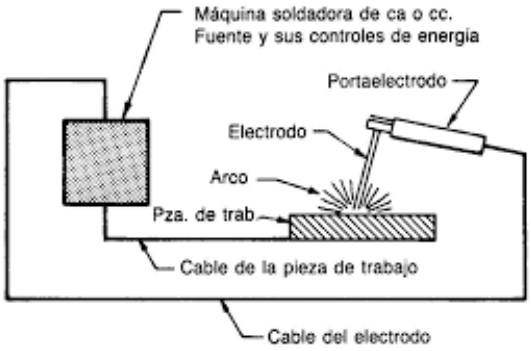


Figura 4.7: Soldadura por arco.[23]

El arco eléctrico es una corriente eléctrica que salta, a través del aire o de un gas, entre dos cuerpos conductores llamados electrodos. Se establece al calentarse las moléculas de gas que rodean el electrodo negativo, haciendo que se liberen electrones cargados de electricidad negativa, que serán atraídos por el otro electrodo cargado positivamente.

Aplicando una tensión en determinadas condiciones, se puede originar una corriente electrónica que, debido especialmente a la ionización por choque, cumple las condiciones necesarias para la ionización de la columna de gas existente entre los electrodos o entre el electrodo y la pieza de trabajo, ya que, según la teoría de los iones, las moléculas neutras de gas están sometidas a la descomposición de iones de gas. De aquí que este gas ionizado constituya el verdadero camino por el que se efectúa la marcha o migración de la electricidad. En el interior de la columna gaseosa, los electrones (negativos) avanzan con enorme velocidad hacia el polo positivo. Este extraordinario y rápido movimiento de los electrones o de los iones se debe a su elevada energía cinética. Estas partículas aceleradas, al chocar con las moléculas neutras que contiene la corriente de gas, producen inmediatamente, como consecuencia, su descomposición en iones electropositivos y electronegativos, los cuales, por su parte, quedan igualmente a la disposición del transporte o a la migración de la electricidad. La columna de gas adquiere en este momento una media luminosidad, y entra una intensa radiación que produce arco eléctrico o voltaico. Los átomos cargados positivamente (cationes) son atraídos por el polo negativo (cátodo), que por el choque de los iones se calienta considerablemente. Este proceso de descomposición de los átomos en iones y electrones se denomina ionización.

El choque de los electrones con el polo positivo (ánodo) que ha tenido lugar en la distancia aérea con una velocidad muy elevada, se produce con extraordinaria violencia y la energía cinética se transforma en calor en el lugar del choque.

Los elementos de un arco eléctrico para soldadura con electrodo desnudo son: el núcleo del arco, la columna de vapor, la llama y el cráter o parte de la pieza fundida por el arco como se muestra en la Figura 4.8.[25] [23]

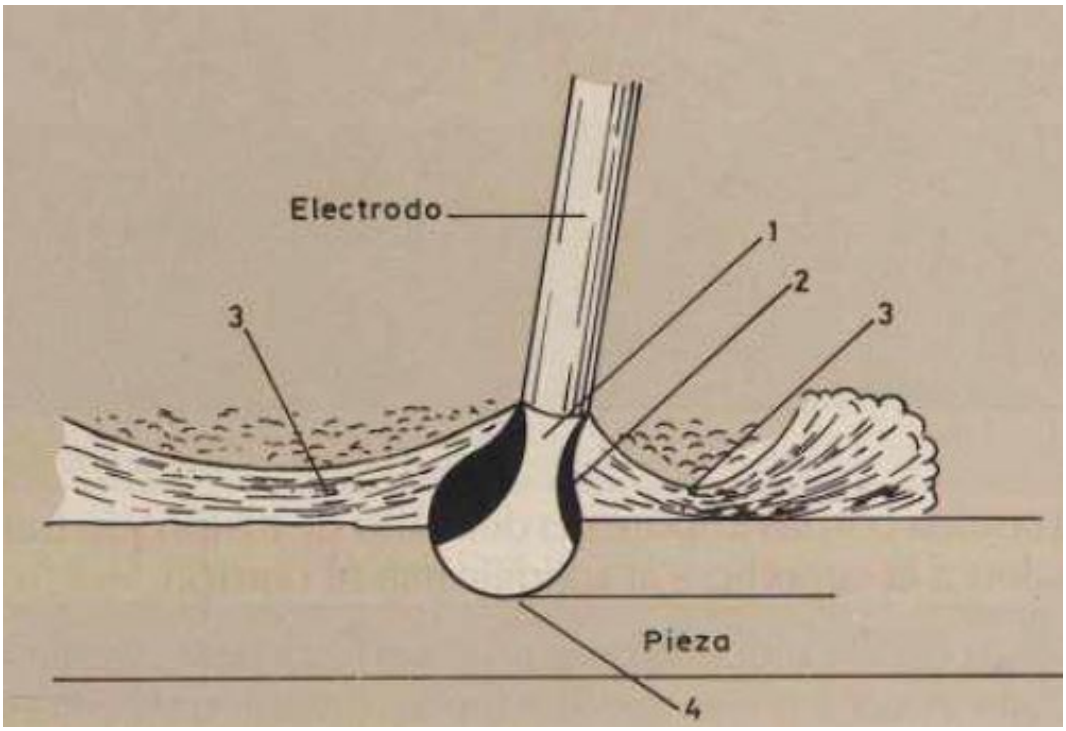


Figura 4.8: Arco eléctrico. 1)núcleo; 2)columna; 3)llama y 4)cráter. [25]

El proceso de soladura con arco eléctrico con gas inerte protector (MIG, por sus siglas en inglés) es un proceso en el cual la fusión se genera por el calentamiento con el arco eléctrico formado entre un electrodo de metal de aporte continuo y la pieza, en el cual la protección del arco se recibe por medio el gas suministrado de manera externa, generalmente se utiliza argón, el cual protege el metal líquido de la contaminación del exterior y permite una mejor estabilización del arco eléctrico, como se puede ver en la Figura 4.9.

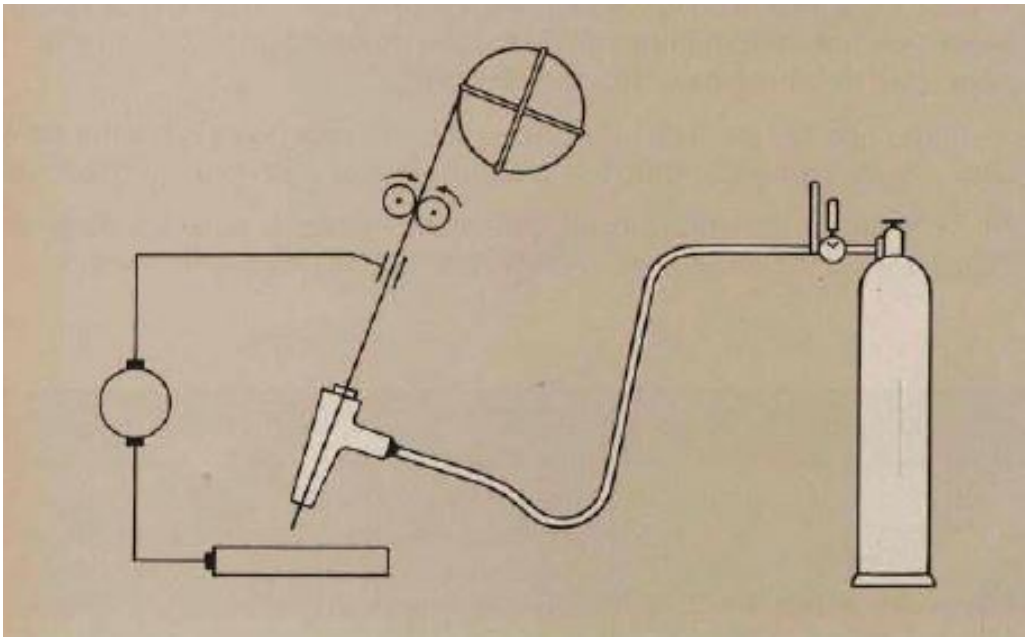


Figura 4.9: Configuración soldadura (MIG). [25]

El proceso (MIG) posee una amplia variedad de aplicaciones, ya que puede ser utilizada para la soldadura de aceros, cobre, aluminio, aceros inoxidables, entre otros. [26]

Para el proceso de soldadura (MIG) existe una serie de parámetros generales a tomar en cuenta dependiendo del tipo de material base, como lo es el tipo de gas recomendado a utilizar, al igual que el diámetro del hilo metálico y el amperaje recomendado en la configuración de corriente directa y polaridad inversa, como se puede observar en la Tabla 4.2.

Metal base	Gas inerte	Diámetro alambre	Amperaje corriente directa-polaridad inversa
Aluminio y aleaciones	Argón Helio-Argón	0.030” 0.035” 0.045” 1/16”	50-150 55-200 90-250 160-350
Acero dulce	CO2 Argón-CO2	0.030” 0.035” 0.045” 1/16”	50-150 60-180 90-200 300-450
Acero de baja aleación	Argón-Oxígeno Argón CO2	0.030” 0.035” 0.045” 1/16’	50-150 75-230 100-350 300-450
Acero inoxidable	Argón-Oxígeno Helio-Argón-CO2	0.030” 0.035” 0.045” 1/16’	75-150 100-160 140-310 280-350
Níquel aleaciones	Argón Helio-Argón	0.035” 0.045” 1/16’	100-150 150-260 200-400
Bronces	Argón Helio-Argón	1/16” 5/64”	225-300 275-350
Cobre	Argón Helio-Argón	1/16”	300-470
Cobre-Níquel	Argón	1/16”	260-300
Magnesio aleaciones	Argón Helio-Argón	0.045” 1/16”	220-280 240-390

Tabla 4.2: Recomendaciones generales soldadura (MIG).[26]

4.2.3. Deformación por soldadura

Los fenómenos físicos que se producen debido a la temperatura son los causantes de deformaciones en las piezas a las que se les aplica la soldadura, al realizar una soldadura en una chapa, para realizar un proceso de revestimiento duro, se genera una contracción longitudinal. Al no realizar una sujeción adecuada del material base, éste sufrirá una deflexión en dirección al cordón de soldadura mediante este se enfría, como se puede ver en la Figura 4.10.[13]

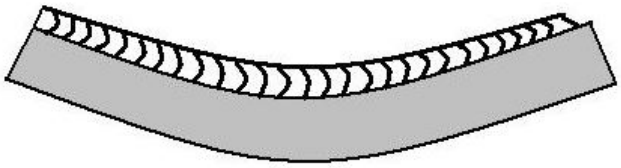


Figura 4.10: Deformación por contracción transversal debido a la temperatura.[13]

4.3. Motores paso a paso

Un motor paso a paso es un dispositivo electromecánico que por medio de una serie de pulsos eléctricos se genera un desplazamiento angular, permitiéndole así rotar una cantidad de grados dependiendo de las entradas de control. Los motores paso a paso son ideales para la elaboración de mecanismos en donde se requieran movimientos precisos. Lo que caracteriza a estos motores es la controlabilidad de los mismos, ya que permiten un movimiento de un paso a la vez por cada pulso que se le aplique. El rango en el que estos pasos se pueden aplicar van desde 1.8^{o} hasta 90^{o}. Debido a ello este tipo de motores son muy utilizados ya que se pueden acoplar a la necesidad del usuario por medio de un microcontrolador.

El principio de funcionamiento de estos motores se basa en un estator construido por una serie de bobinados de material ferromagnético y un rotor en el cual el estator posee un giro libre. Al ingresar la corriente por estos bobinados se genera un campo electromagnético haciendo así que el estator sea atraído generando un desplazamiento angular al que se le denomina paso angular.[18]

Existen dos variantes de motores paso a paso, los motores de magneto permanente y de reluctancia variable, independientemente del etiquetado en cada motor, la diferencia que se puede notar entre los mismos, es el movimiento del rotor principal cuando no se encuentran conectado a una fuente de corriente.

Los motores de magneto permanente generan un sonido peculiar cuando se realiza este movimiento en el rotor, mientras que los motores de reluctancia variable giran libremente sin emitir sonido. también se puede diferenciar estos motores por medio de un óhmetro.[17]

4.3.1. Motores de reluctancia variable

Los motores de reluctancia variable poseen dientes uniformemente distribuidos, hechos de hierro al igual que el rotor de hierro liso y opera basado en el principio de que la reluctancia mínima ocurre en un espaciamiento mínimo, por lo tanto, los puntos del rotor son atraídos hacia los polos del imán del estator, como se puede ver en la Figura 4.11, usualmente poseen tres bobinados con un retorno común, como se muestra en la Figura 4.12, este tipo de configuración permite lograr pequeños o medios ángulos de paso y operar a altas frecuencias de control, sin embargo los motores de este tipo no pueden mantener su posición.[17]

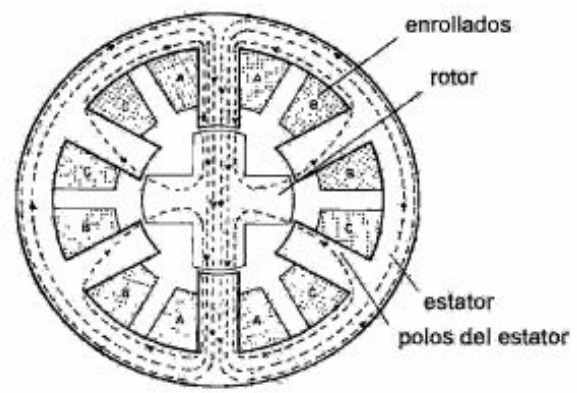


Figura 4.11: Sección de un motor de paso de reluctancia variable.[11]

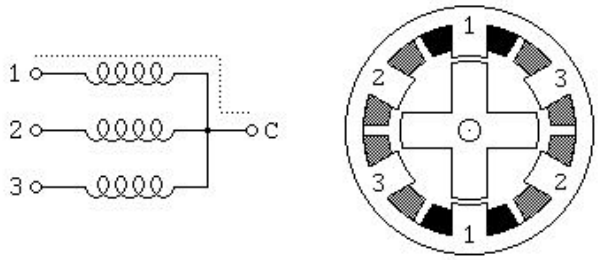


Figura 4.12: Motor paso a paso de reluctancia variable[17]

4.3.2. Motores de magneto permanente

A diferencia de los motores de reluctancia variable, estos motores poseen dientes hechos de material de magneto permanente con polos ubicados de forma radial, con una construcción del estator similar. Cuando los bobinados del estator son energizados, los campos electromagnéticos interactúan con el flujo del magneto permanente generando un torque para mover el rotor como se puede ver en la Figura 4.13.

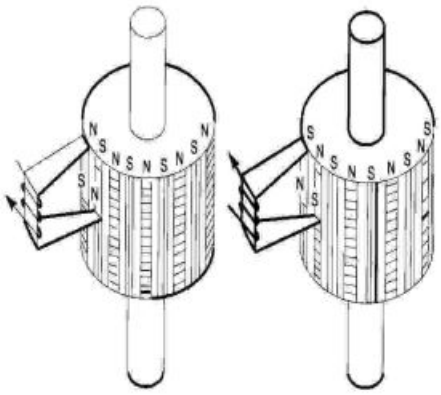


Figura 4.13: Motor paso a paso de magneto permanente [11]

Los motores de magneto permanente usualmente poseen dos bobinados independientes, con o sin retornos comunes cada bobinado, como se puede observar en la Figura 4.14 y 4.15, los bobinados con retorno común son utilizados en motores de magneto permanente unipolares y los que no tienen retorno común son utilizados en motores de magneto permanente bipolares.

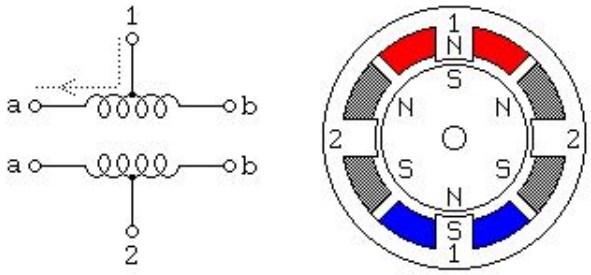


Figura 4.14: Motor paso a paso de magneto permanente unipolar[17]

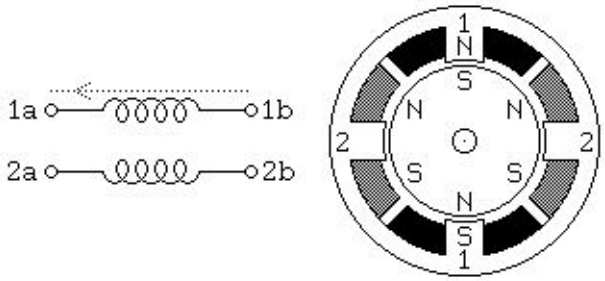


Figura 4.15: Motor paso a paso de magneto permanente bipolar[17]

Este tipo de motores puede proveer altos torques y poseen la propiedad de mantenerlo cuando los bobinados no están energizados.[17]

4.4. Controlador de motor paso a paso

Los controladores de motores paso a paso están diseñados únicamente para realizar el manejo de dichos motores, los cuales proporcionan un control preciso de posición y son capaces de brindar una rotación continua.

4.4.1. Controlador ST-M5045

Es un controlador de micro-pasos bipolar de dos etapas rentable de alto rendimiento, que aplica una técnica de control de corriente sinusoidal pura. Es el más adecuado para las aplicaciones que desean un nivel extremadamente bajo de ruido y calor. Funciona bien en un entorno donde el suministro de electricidad experimenta inestabilidad y fluctuaciones.

Figura 4.16: Controlador de micro-pasos ST-M5045.

4.5. Codificadores

Los codificadores son dispositivos denominados de detección, ya que estos brindan una respuesta. Los codificadores transforman el movimiento en una señal eléctrica que puede ser interpretada por cualquier dispositivo de control en un sistema de control de movimiento, como en un microcontrolador o un controlador lógico programable (PLC, por sus siglas en inglés). La señal de respuesta del codificador puede ser empleado en la determinación de posiciones, velocidades, dirección o simplemente realizar un conteo. Los codificadores pueden

ser utilizados como una forma de retroalimentación para poder conocer el estado actual de la maquinaria en la que se esté utilizando, tanto para el conocimiento del operador, como para la máquina en sí.

En cualquier aplicación a la que el codificador esté sometido, el proceso es el de generar un recuento para así ser enviado al controlador, el cual interpreta dicha información y éste envía las instrucciones a la maquinaria para realizar una función en específico.

Existen diversos tipos de tecnologías utilizadas por los codificadores para generar una señal, estas pueden ser, mecánicas, magnéticas, ópticas y de resistencia, siendo la óptica la más común. En la detección óptica, el codificador se encarga de proporcionar información basada en la interrupción de la luz.

Los codificadores rotativos se clasifican en dos tipos: absolutos e incrementales (relativos). Un codificador absoluto envía un código digital de N cantidad de bits que representa el valor de la posición de su eje, mientras que un codificador incremental, como se puede ver en la Figura 4.17, envía una señal de pulsos a medida que el eje avanza o retrocede, lo que permite conocer la variación de posición.[7]

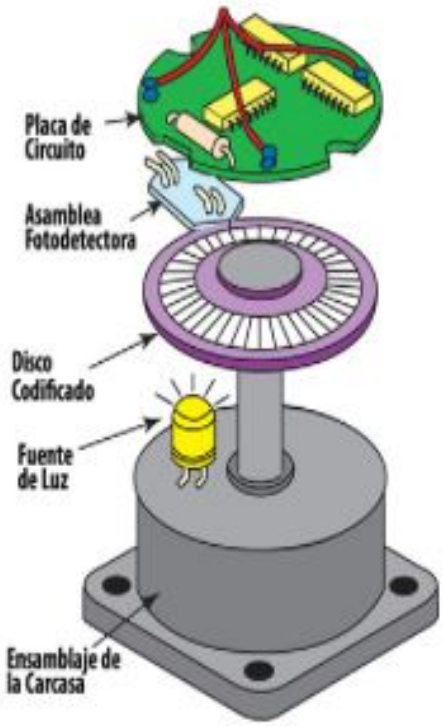


Figura 4.17: Codificador rotativo incremental con tecnología óptica[7]

Un haz de luz emitida por un diodo emisor de luz (LED, por sus siglas en inglés) pasa a través del disco de código, que está modelada con líneas opacas. A medida que el eje del codificador gira, la viga de luz del LED es interrumpida por las líneas opacas en el disco de código antes de ser recogido por la Asamblea Foto-detectora. Esto produce una señal de pulso. La señal se envía al contador o controlador, que a su vez enviará la señal para

producir la función deseada.[7]

4.6. Caja reductora

Una caja reductora es un mecanismo que consigue reducir la velocidad de rotación de un eje secundario o de salida con respecto a la velocidad de un eje principal o de entrada, por medio de un conjunto de engranajes, poleas o ruedas dentadas, en una o varias etapas de reducción. Debido a la reducción de la velocidad, se consigue un aumento del par torsional en el eje reducido, proporcional a la reducción.

4.6.1. Engrane de tornillo sin fin

Se utiliza en la transmisión del movimiento entre dos árboles que se cruzan sin cortarse, normalmente formando un ángulo de 90^{o}. Se compone de un tornillo cilíndrico (impulsor) que engrana en una rueda dentada cilíndrica con dentado helicoidal (impulsado), como se puede ver en la Figura 4.18.

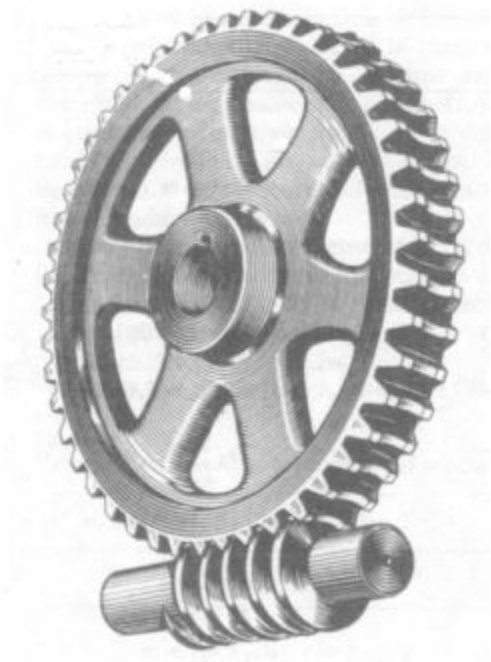


Figura 4.18: Engrane de tornillo sin fin.[2]

Este tipo de engranaje permite obtener una gran reducción de velocidad, presentando un bajo rozamiento y una marcha silenciosa. Sin embargo, como en todos los engranajes helicoidales, presenta un empuje axial elevado, por lo que exige la utilización de cojinetes adecuados para poder soportar dichos esfuerzos.[2]

4.6.2. Engrane de tornillo sin fin como freno mecánico

Debido al contacto lineal existente entre el filete del tornillo y los dientes de la rueda, al girar el tornillo sin desplazarse axialmente, transmite un movimiento de giro a la rueda; de tal forma que, en una rotación completa del tornillo, la rueda gira un arco igual al paso de la rosca del tornillo. La transmisión del movimiento se realiza siempre del tornillo sin fin a la rueda helicoidal y no al revés, esto permite eliminar cualquier posibilidad de movimiento que provenga de la rueda sea transmitido al tornillo, tomando en cuenta que este efecto de frenado se ve afectado por la cantidad de hilos que posea el tornillo, mientras más hilos, mayor es la posibilidad de que exista una transmisión de la rueda hacia el tornillo. Configuraciones del sin fin en las que el equipo no puede transmitir movimientos al tornillo se dice que son autoblocantes, circunstancia que depende del ángulo de ataque entre engranajes y del coeficiente de fricción entre ambos.[3]

4.7. Microcontroladores

Los microcontroladores son circuitos integrados que contienen todos los componentes que posee una unidad central de procesamiento, (CPU pos sus siglas en inglés). Estos son usados para controlar el funcionamiento de una tarea en específico, y debido a su tamaño reducido generalmente se encuentra incorporado dentro del dispositivo que controla aún sin un sistema de retroalimentación.

Se puede decir que los microcontroladores son computadores dedicados, ya que en su memoria se encuentra únicamente el programa el cual ejecuta la aplicación a la que está determinado. Los periféricos, los cuales son las entradas y salidas de señales, son capaces de soportar tanto sensores como actuadores del dispositivo que se desea controlar, una vez programado el microcontrolador es útil únicamente para desarrollar la tarea designada, tomando en cuenta que este permite ser reprogramado para ser utilizado en otra tarea específica.[10]

El propósito fundamental de los microcontroladores es el de leer y ejecutar los programas que el usuario genera, por lo tanto la programación es indispensable al diseñar circuitos y sistemas que incluyan un microcontrolador. El hecho que sean programables influye de gran manera en el diseño de circuitos, ya que simplifican dichos diseños. Los microcontroladores permiten flexibilidad y modularidad.

Las aplicaciones en las que se puede utilizar un microcontrolador son vastas, debido a que se pueden encontrar en campos como la industria del entretenimiento, la robótica, en la industria automotriz, en el hogar, entre otros.[21]

4.7.1. Microcontrolador ATMEL

La familia de microcontroladores ATMEL esta basada en una arquitectura denominada RISC (Computadora de conjunto reducido de instrucciones, por sus siglas en inglés) que implementa memoria FLASH para el programa y memoria EEPROM (Memoria de solo

lectura programable y borrable eléctricamente, por sus siglas en inglés) para los datos, con arquitectura diseñada para permitir un trabajo de programación de alto nivel.[22]

4.7.2. Microcontrolador ATmega328p

A diferencia de los demás microcontroladores de la familia ATMEL el microcontrolador ATmega328p posee las características que se presentan en la Tabla 4.3.

Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Pines de entrada/salida digitales	14 (6 proporcionan salida PWM)
Pines de entrada analógica	6
Corriente máxima por pin	40 mA
Corriente máxima en pin 3.3V	50 mA
Memoria flash	32 Kbytes
SRAM	2 Kbytes
EEPROM	1 KB
Velocidad de reloj	16 MHz

Tabla 4.3: Características del microcontrolador ATmega328p.[10]

4.7.3. Microcontrolador ATmega2560

El microcontrolador ATmega2560 posee las características que se presentan en la Tabla 4.4.

Voltaje de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Pines de entrada/salida digitales	54 (15 proporcionan salida PWM)
Pines de entrada analógica	16
Corriente máxima por pin	40 mA
Corriente máxima en pin 3.3V	50 mA
Memoria flash	256 Kbytes
SRAM	8 Kbytes
EEPROM	4 KB
Velocidad de reloj	16 MHz

Tabla 4.4: Características del microcontrolador ATmega2560.[15]

4.8. Comunicación serial

La comunicación serial es un protocolo muy común y es utilizado para la comunicación entre dispositivos. El funcionamiento de la comunicación serial es sencillo, el puerto serial envía y recibe bytes de información un bit a la vez, tiene como ventaja que puede alcanzar distancias relativamente mayores que la comunicación en paralelo. La comunicación serial es utilizada generalmente para transmitir datos en formato ASCII y utiliza tres líneas de transmisión, tierra o referencia, una de transmisión y otra de recepción. Para asegurar la comunicación entre dispositivos por medio de este método, se requiere cumplir con las siguientes características: velocidad de transmisión, la cual consiste en indicar el numero de bits por segundo que se transfieren, bits de datos, los cuales indican la cantidad de bits que se están transmitiendo, bits de parada, el cual es utilizado para indicar el final de la transmisión de un solo paquete y la paridad, la cual es una forma de verificar si hay existencia de errores en la transmisión serial.[16]

4.8.1. Universal asynchronous receiver/transmitter (UART)

UART (Transmisor/Receptor Universal Asíncrono, por sus siglas en inglés) es un componente electrónico que se encarga de realizar la comunicación serial entre computadoras, dispositivos y microcontroladores. Se encarga de tomar los bytes de información y realiza la transmisión bit por bit de forma secuencial y en el dispositivo de destino, éste mismo se encarga de traducir la información de bits a bytes nuevamente.

La peculiaridad de la comunicación asíncrona es que éste permite que los datos sean transmitidos sin tener que enviar al receptor una señal de reloj. En su lugar el emisor y el receptor deben acordar los parámetros de temporización previamente y se deben añadir bits especiales a cada palabra que se utilizan para sincronizar el emisor y el receptor. Cuando una palabra es enviada al UART, el bit de inicio es añadido al inicio de la misma, el cual le indica al receptor que está por recibir una palabra, posteriormente son enviados los bits iniciando con el bit menos significativo, cada bit es transmitido exactamente la misma cantidad de tiempo. Posteriormente son enviados los bits de paridad y el bit de parada.[8]

4.9. Sensores de corriente

Los sensores de corriente son utilizados para monitorear corriente continua o alterna. se incluyen sensores de corriente lineales ajustables, de balance nulo, digitales y lineales. Los sensores de corriente digitales pueden ser utilizado como un activador de una alarma, abrir una válvula, accionar un motor, entre otros. Mientras que los sensores de corriente lineales duplican la forma de la onda de la corriente captada, y puede ser utilizada como un elemento de respuesta para controlar un motor o regular la cantidad de trabajo que realiza una máquina.

Existen cuatro tipos diferentes de sensores de corriente, los cuales son: Shunt de corriente

de baja resistencia, Sensor de efecto Hall, Transformador de corriente y Bobina Rogowski.[27]

4.9.1. Sensor de efecto hall

Se utiliza el sensor de efecto Hall para la medición de corrientes o campos magnéticos, o para determinar la posición. Su función sigue unos pasos en donde si fluye corriente por un sensor Hall y se aproxima a un campo magnético que fluye en dirección vertical al sensor, entonces el sensor crea un voltaje saliente proporcional al producto de la fuerza del campo magnético y de la corriente. Si se conoce el valor de la corriente, entonces se puede calcular la fuerza del campo magnético, si se crea el campo magnético por medio de corriente que circula por una bobina o un conductor, entonces se puede medir el valor de la corriente en el conductor o bobina.

Hay dos tipos de sensores de efecto Hall, anillo cerrado y anillo abierto. El sensor de anillo cerrado brinda una mejor precisión y proporciona rangos dinámicos más amplios, pero esta funcionalidad posee un costo más elevado. El sensor de efecto Hall se caracteriza por tener una respuesta eficiente ante la frecuencia y es capaz de realizar una medición de corrientes elevadas.

En la Tabla 4.5. Se puede observar un resumen de características y efectos para cada tipo de sensor de corriente.

Sensor de corriente	Características	Efectos
Shunt	De 100 a 500 micro-ohms. Bajo costo. Inmune al problema de la saturación de corriente directa. No está aislado. Buena linealidad. No útiles para grandes corrientes.	Alta disipación de potencia en forma de calor. Necesidad de aislamiento galvánico.
Transformador de corriente	Capaz de medir altas corrientes. Se necesita emparejar la fase. Baja disipación comparada con la del Shunt. Proporciona aislamiento. Tiene problemas de saturación de corriente directa.	Un desfase de 0,1º produce un error en la facturación.
Bobina de Rogowski	Tiene todas las ventajas del Transformador de Corriente, pero es más barato. Necesita un integrador, Inmune a la saturación de corriente directa. Buena linealidad. Bajo consumo.	Es difícil tener un integrador que sea muy estable con el tiempo.
Efecto hall	Son más caros. Baja linealidad. Bueno para altas corrientes.	Consumo medio. Variación alta con la temperatura

Tabla 4.5: Resumen de características y efectos para cada sensor de corriente.[27]

4.10. Tableros eléctricos

Los tableros eléctricos son gabinetes que albergan a los dispositivos de comando, protección, conexión, señalización, alarma, con sus respectivos soportes, para así realizar la tarea específica dentro del sistema eléctrico. Para la elaboración y armado de un tablero eléctrico se deben de cumplir con los criterios de diseño y las normativas respectivas que permitan su funcionamiento de manera correcta, una vez sea energizado, asegurando la seguridad de los operarios y de las instalaciones en donde se ubican.

Tanto los instrumentos de medición como los equipos que se utilizarán, como lo son el equipo de control y protección, generalmente son instalados en tableros eléctricos teniendo como referencia un diagrama de conexión.

Una importante medida de seguridad para los tableros eléctricos es la instalación de interruptores de seguridad, dichos interruptores de seguridad suelen ser de dos tipos: termo magnético, que se encarga de proteger tanto el tablero eléctrico como la instalación de variaciones en la corriente, y diferencial, que está dirigido a la protección de los usuarios. Los tableros contienen en su interior equipos eléctricos que a su vez contienen: Barras de Distribución. Elementos de Protección, Elementos de Señalización, Elementos de Comando y eventualmente, instrumentos de medida.

Existen diversidad de tableros eléctricos, su clasificación depende según su función y ubicación y según el uso de energía eléctrica.[28] Algunos de ellos son:

4.10.1. Tableros generales

Son los tableros principales de las instalaciones. En ellos estarán montados los dispositivos de protección y maniobra que protegen los alimentadores y que permiten operar sobre toda la instalación interior en forma conjunta o fraccionada, como se puede observar en la Figura 4.19.

Figura 4.19: Tablero general[20]

4.10.2. Tableros de distribución

Son tableros que contienen dispositivos de protección y maniobra que permiten proteger y operar directamente los circuitos en que está dividida la instalación o una parte de ella. Pueden ser alimentados desde un tablero general, desde un tablero general auxiliar o directamente desde el empalme, como se puede ver en la Figura 4.20.

Figura 4.20: Tablero de distribución[20]

4.10.3. Tableros de fuerza

Podemos indicar que un centro de carga, es un tablero metálico que contiene una cantidad determinada de interruptores termo magnéticos, generalmente empleados para la protección y desconexión de pequeñas cargas eléctricas y alumbrado. En el caso de que en un tablero eléctrico se concentre exclusivamente interruptores para alumbrado, se conoce como tablero de alumbrado; si concentra otros tipos de cargas, se conoce como tablero de fuerza, como se puede ver en la Figura 4.21.

Figura 4.21: Tablero de fuerza[20]

4.10.4. Tablero de control

Es una herramienta, el diagnóstico y monitoreo permanente de determinados indicadores e información ha sido y es la base para mantener un buen control de situación. El tablero permite a través del color de las luces y alarmas ser el disparador para la toma de decisiones,es fundamental definir los indicadores a monitorear, como se puede ver en la Figura 4.22.

Figura 4.22: Tablero de control[20]

4.10.5. Tableros de señalización

Es el encargado de interactuar con el operario, generalmente estos son conformados por luces piloto, pantallas, pulsadores, selectores, entre otros, en donde el operario puede visualizar alertas u operar las funcionalidades de la máquina en sí, como se puede ver en la Figura 4.23.

Figura 4.23: Tablero de señalización[20]

CAPÍTULO 5

Marco metodológico

5.1. Observación

Para la realización de este proyecto será mediante la investigación sobre el funcionamiento de los diferentes componentes electrónicos a utilizar y cómo estos pueden ser implementados correctamente entre sí para lograr un funcionamiento idóneo de los módulos a realizar, también se investigará en diversos libros, artículos, y trabajos de graduación para el grado de licenciatura y maestría obtenidas por el motor de búsqueda de Google Académico.

5.2. Descripción

Principalmente se utilizará el microcontrolador ATmega2560 el cual será utilizado como el controlador maestro, encargado de la recepción y almacenamiento en la memoria de almacenamiento aleatorio de los parámetros, los cuales pueden ser modificados, indicados previamente para así enviar la serie de instrucciones a la red de microcontroladores esclavo por medio del protocolo de comunicación serial, en los cuales se utilizarán ATmega328P, encargados del control tanto de los motores paso a paso que darán el movimiento a los ejes de movimiento implementados como al control de la potencia de las antorchas que realizarán el depósito del material para el blindaje de las láminas.

Para el controlador de potencia para la soldadura se utilizará como base, la máquina ESAB PEJ la cual es un sistema destinado al control preciso de los parámetros de soldadura en arco sumergido que se implementará en este proyecto utilizando la fuente que esta ya trae incorporado, añadiéndole el uso de un microcontrolador ATmega328P y sensores de corriente de efecto Hall para la recolección e interpretación de los datos brindados, para así poder ser modificados y poder tener un control directo de dicha máquina por medio del microcontrolador antes mencionado.

Para el panel de operador se utilizarán componentes como pulsadores, selectores, indica-

dores luminosos (diodos emisores de luz), palancas de mando, pantallas de cristal líquido y potenciómetros. La base en cual serán ubicados será de alumicon y se elaborará con máquinas de corte como lo es una sierra da vaivén y de perforación como lo es barreno de mano y taladro de banco.

Para la caja reductora y freno mecánico se utilizarán placas de aluminio de $\frac{1}{4}$ de pulgada las cuales se elaborarán con las máquinas utilizadas para el panel de operador y los componentes internos se utilizarán, engranes, tornillos sin fin, ejes y chumaceras las cuales serán adquiridos por un proveedor.

5.3. Examen cxrítico

Se realizarán pruebas previas durante el proceso de elaboración para así verificar que el funcionamiento de cada avance en el proyecto es el deseado y de no ser así realizar las correcciones necesarias tanto en programación como en estructura para obtener los resultados que se proponen al finalizar el mismo.

5.4. Recomendaciones

Se recomienda verificar el consumo al cual estará sometido cada microcontrolador y los valores ideales de trabajo para mantener una estabilidad que se mantenga en el rango óptimo de cada componente para lograr un trabajo en conjunto eficiente sin pérdida de información.

CAPÍTULO 6

Resultados

6.1. Diseño y elaboración de paneles de operador

En las siguientes figuras se pueden observar el diseño de los paneles de operador realizado en el software Visio seguido por el resultado final de la elaboración en el material Alumicon y la unión de los mismo al marco que los contiene con respectivo etiquetado de cada componente y su función.

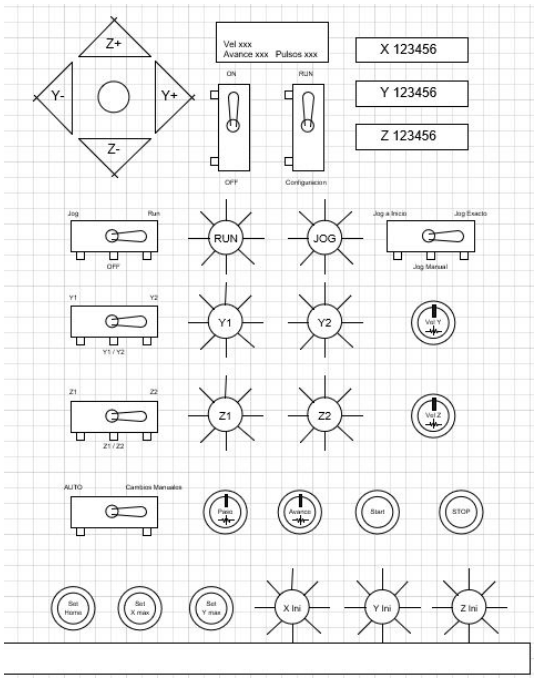


Figura 6.1: Diseño panel de operador de control de movimiento ejes Y y Z y modos de operación.

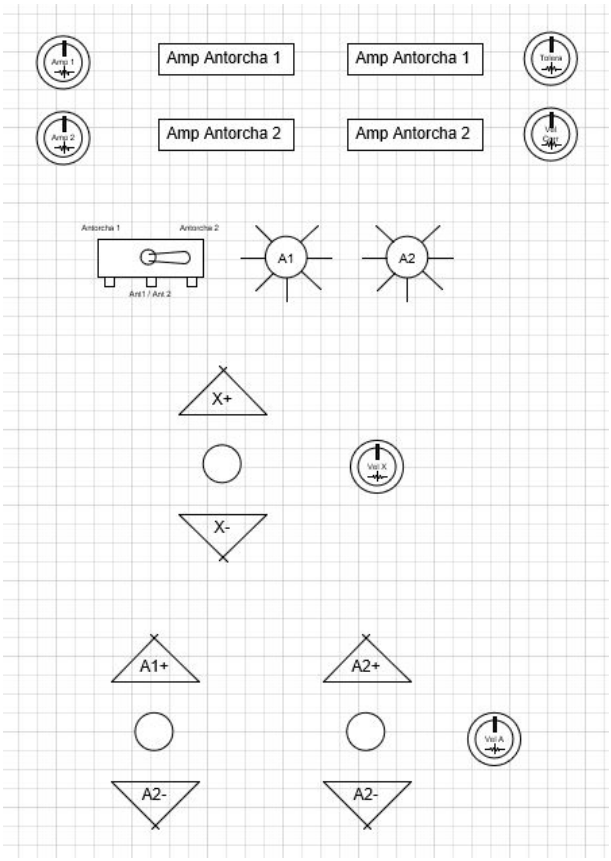


Figura 6.2: Diseño panel de operador de control de antorchas y movimiento eje X.

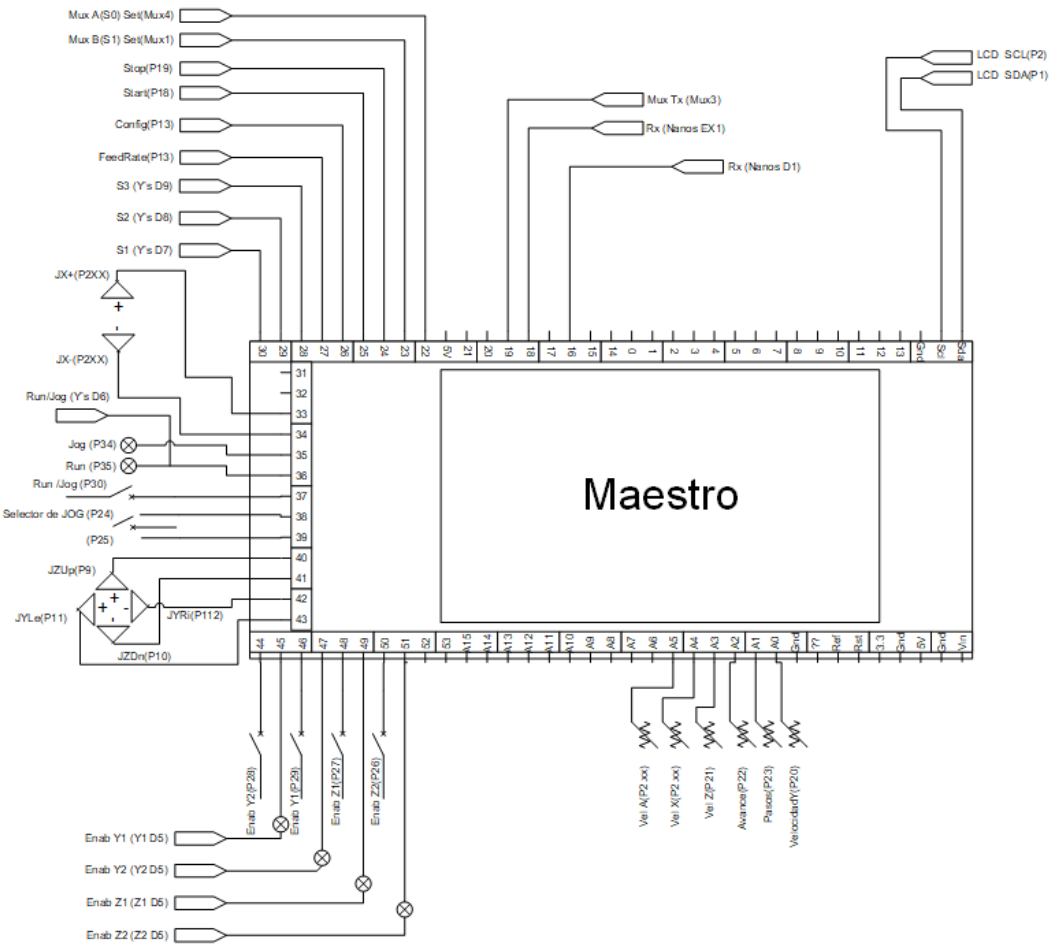
Figura 6.3: Panel de operador final.

Figura 6.4: Prueba de funcionamiento de paneles de operador.

Al realizar las pruebas de funcionamiento del panel de operador se obtuvo que la distribución de los componentes era la adecuada tanto para la visualización de los componentes de señalización como los de operación. Este resultado fue obtenido al realizar una simulación de operación en la que fueron partícipes operarios de maquinaria industrial.

6.2. Planos eléctricos y diseño de cableado

A continuación se encuentran las figuras que representan los planos eléctricos diseñados para la conexión de la red de microcontroladores maestro-esclavos como las conexiones del panel de operador de control de movimiento de los ejes y los modos de operación.




```
//INPUTS
const int JUp = 40;     //Jog up
const int JDn = 41;     //Jog Down
const int JRi = 42;     //Jog right
const int JLe = 43;     //Jog Left
const int RJ = 37;      //Selector Run/Jog
const int JS1 = 38;      //Selector1 de tipo de Jog JS1 HIGH Mov hasta Limit Switch
const int JS2 = 39;      //Selector2 de tipo de Jog JS2 HIGH Mov Manual  JS1 y JS2 HIGH Movimiento exacto
const int Y1 = 44;      //Enable de Y1
const int Y2 = 46;      //Enable de Y2
const int Z1 = 48;      //Enable de Z1
const int Z2 = 50;      //Enable de Z2

//Analogos
const int Vel = A0;     //Potenciometro de velocidad de movimiento (frecuencia)
const int Pul = A1;     //Potenciometro de Cantidad de pulsos a dar
const int Ava = A2;      //Potenciometro de candidad de pulsos a avanzar

//OUTPUTS
const int pY1 = 45;     //Luz piloto de Y1
const int pY2 = 47;     //Luz piloto de Y2
const int pZ1 = 49;     //Luz piloto de Z1
const int pZ2 = 51;     //Luz piloto de Z2
const int pR = 36;      //Luz de Run
const int pJ = 35;      //Luz de JOG
const int S1 = 30;      //S1 Indicador de limite de movimiento en X+ (arriba)
const int S2 = 29;      //S1 Indicador de limite de movimiento en X- (abajo)
const int S3 = 28;      //S1 Indicador de limite de movimiento en Y+ (derecha)
```

Figura 6.5: Plano eléctrico de conexiones microcontrolador maestro.

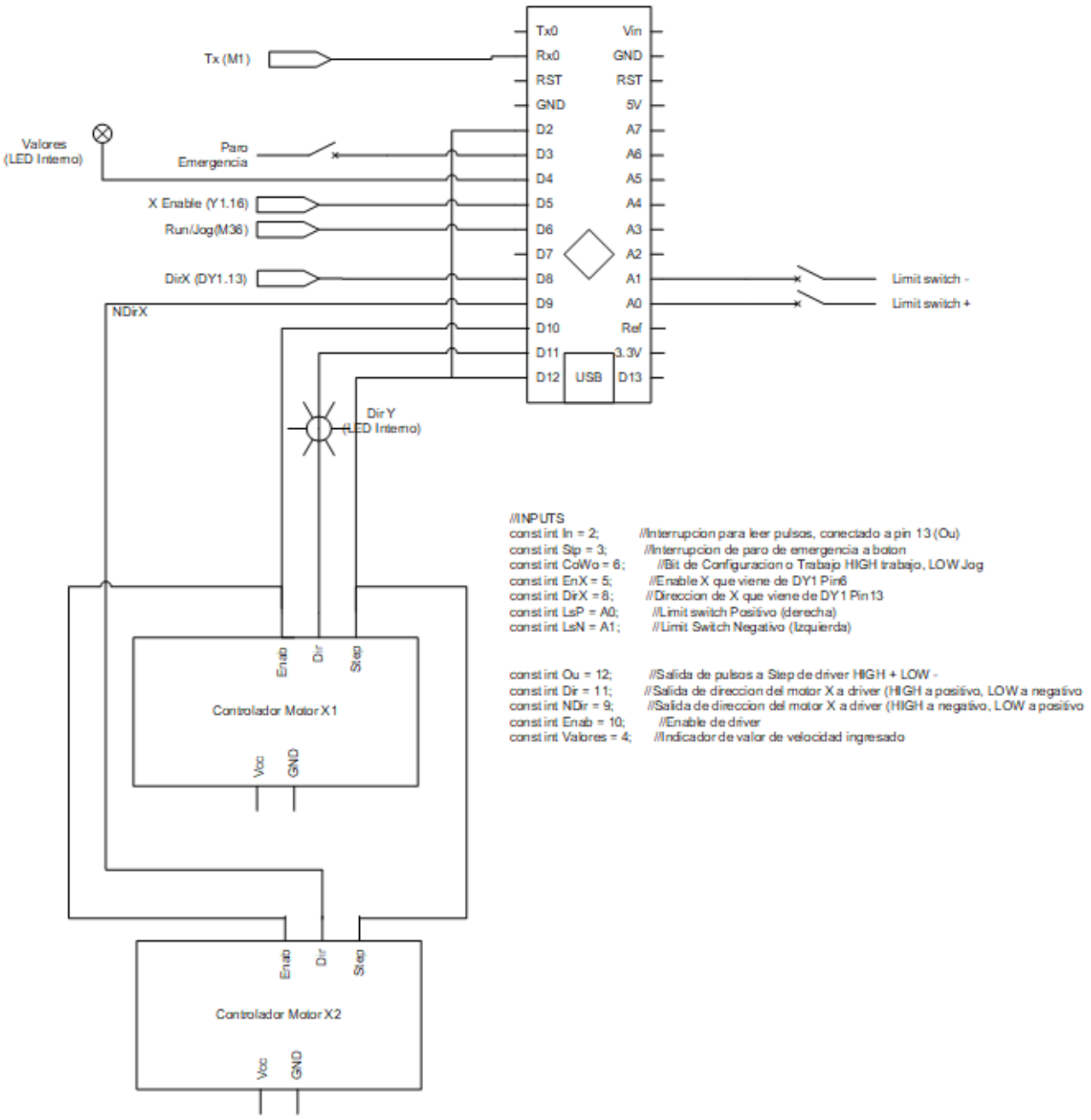


Figura 6.6: Plano eléctrico de conexiones microcontrolador controlador x.

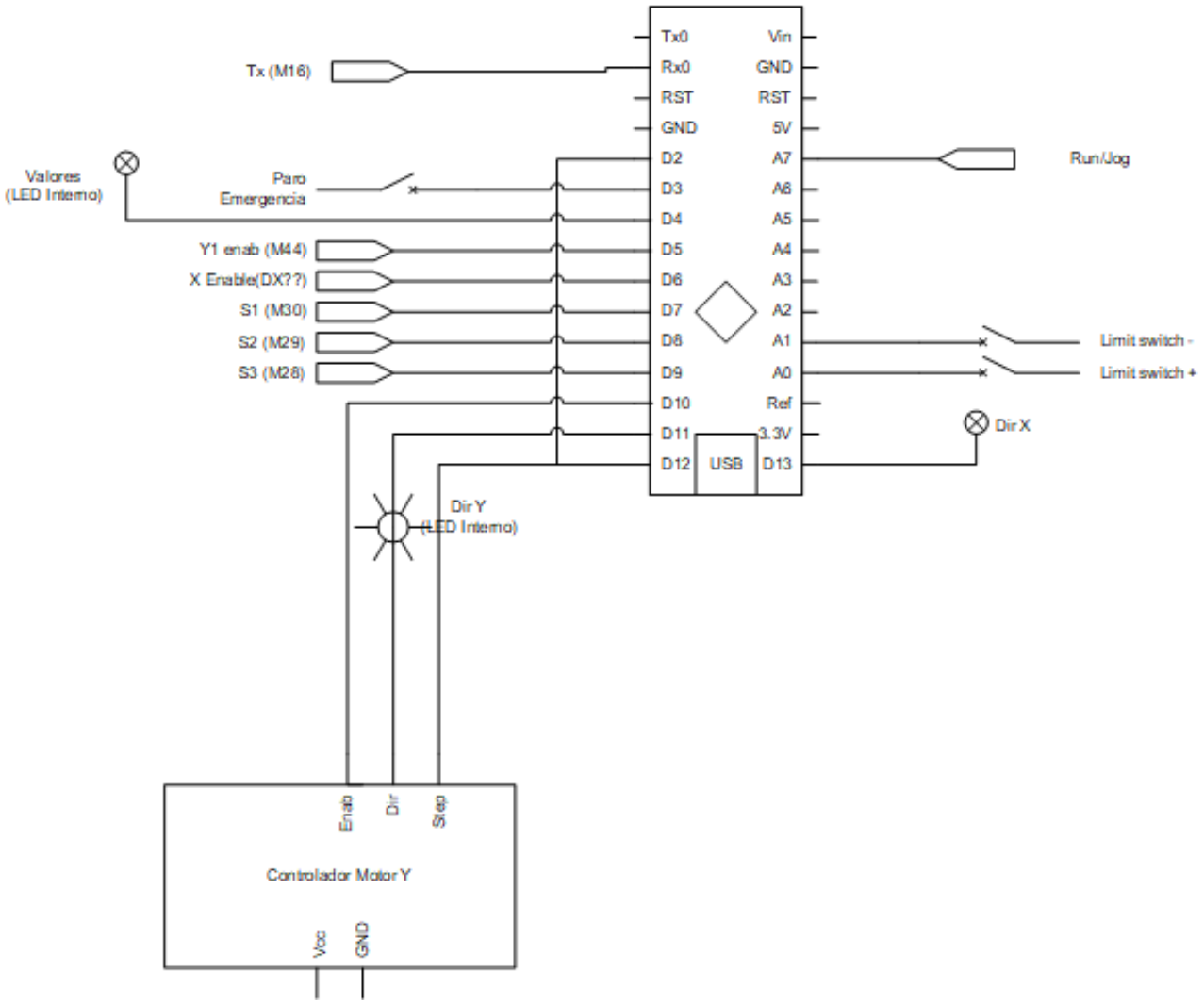


Figura 6.7: Plano eléctrico de conexiones microcontrolador controlador y.

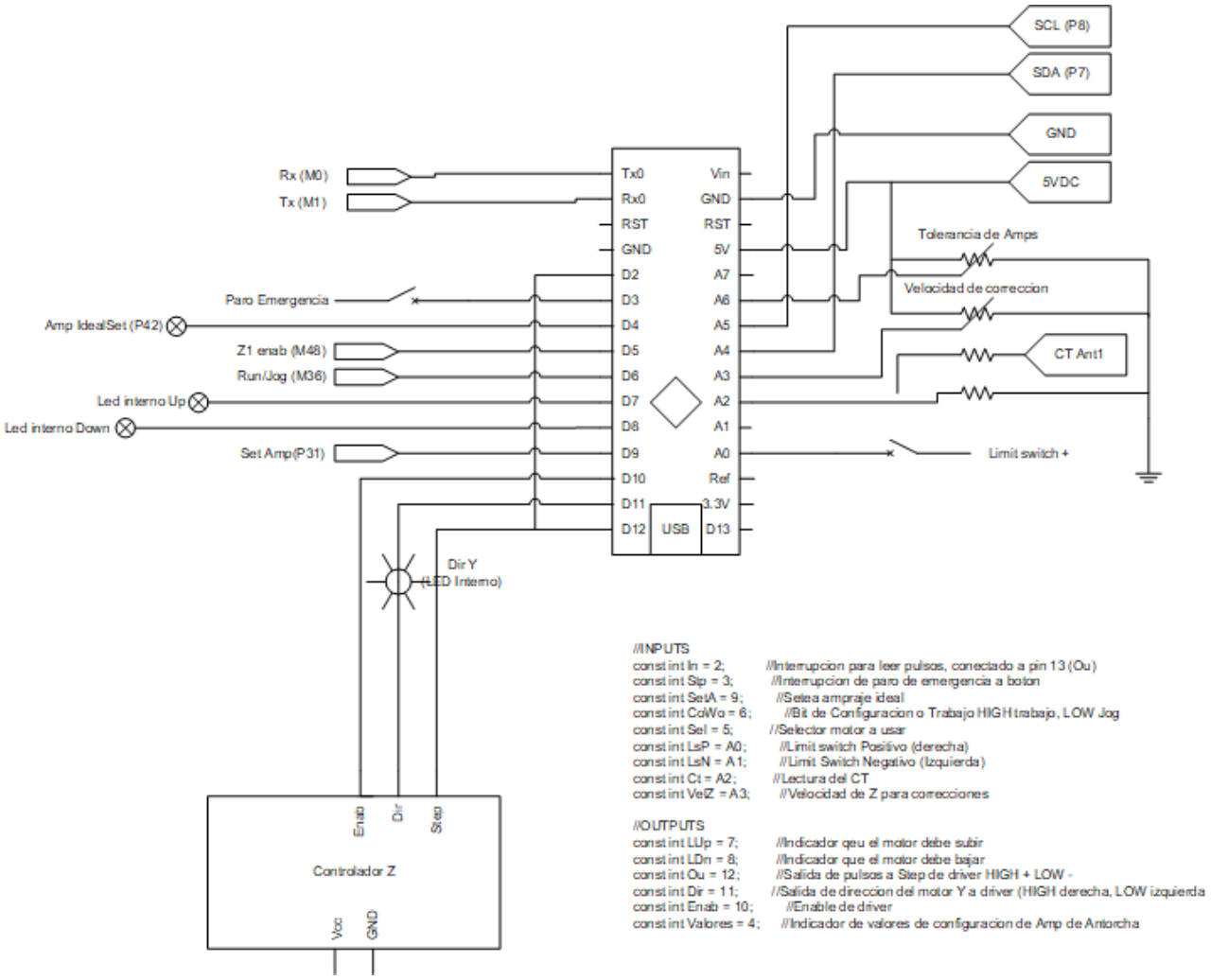


Figura 6.8: Plano eléctrico de conexiones microcontrolador controlador z.

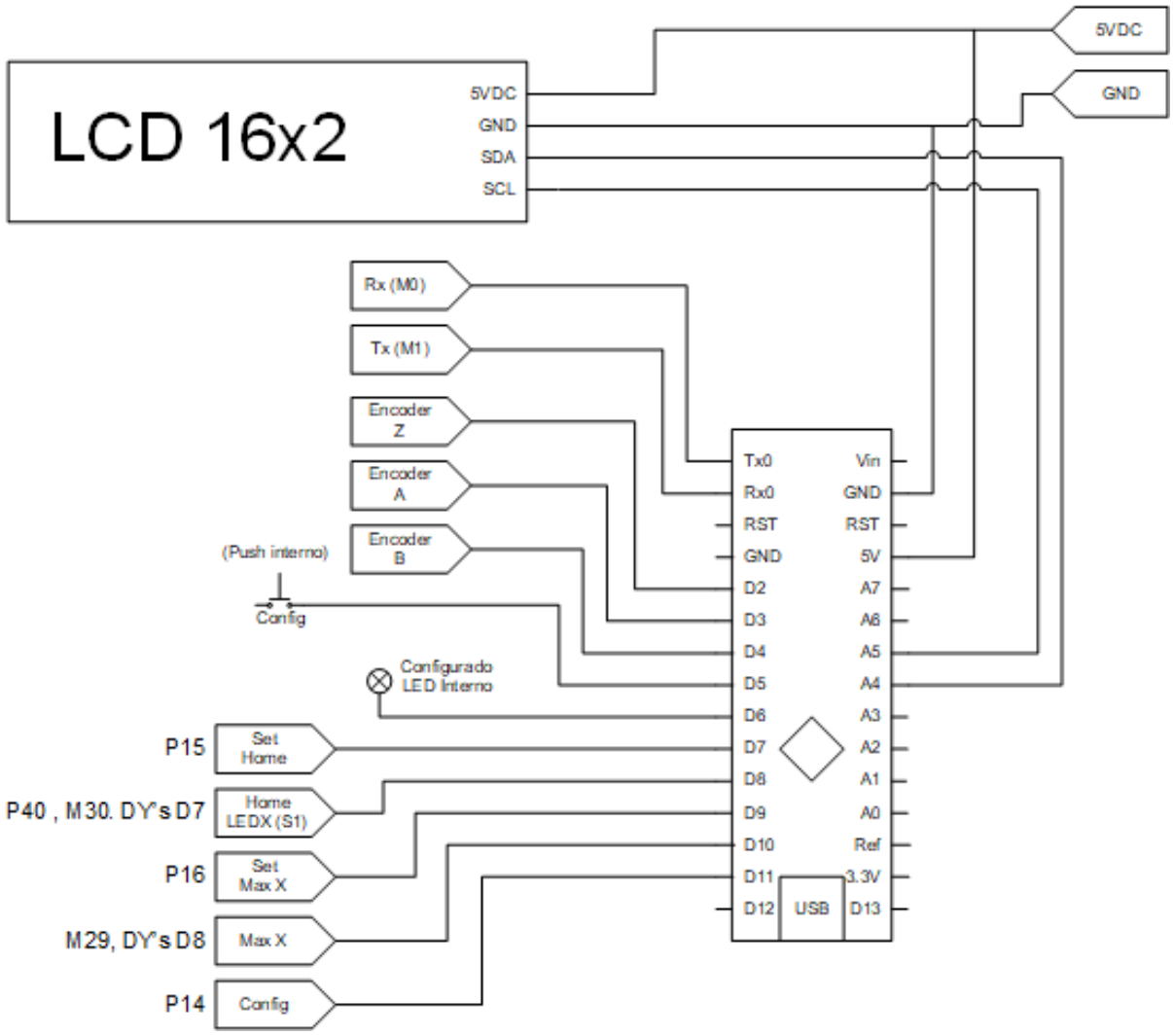


Figura 6.9: Plano eléctrico de conexiones microcontrolador codificador x.

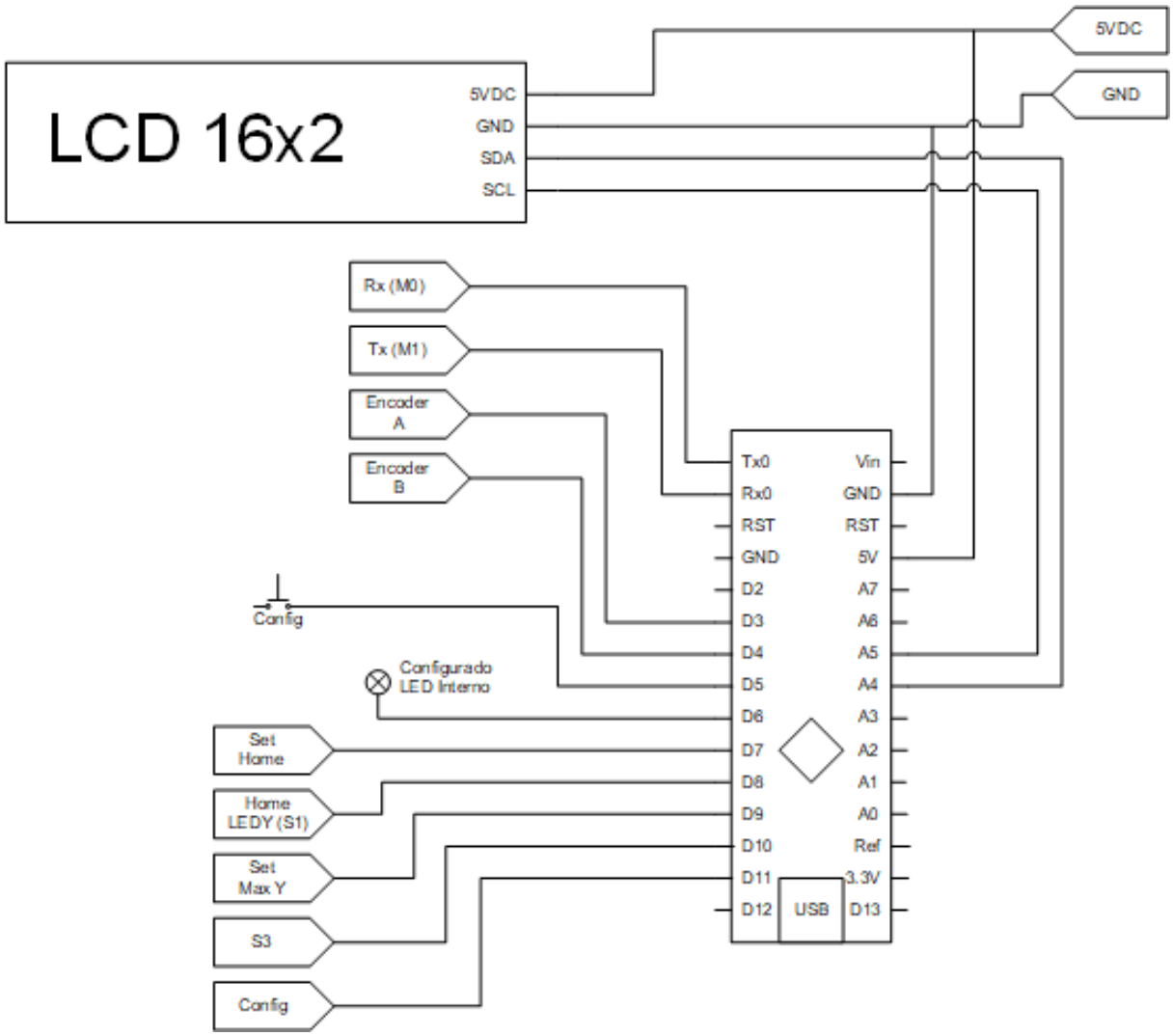


Figura 6.10: Plano eléctrico de conexiones microcontrolador codificador y.

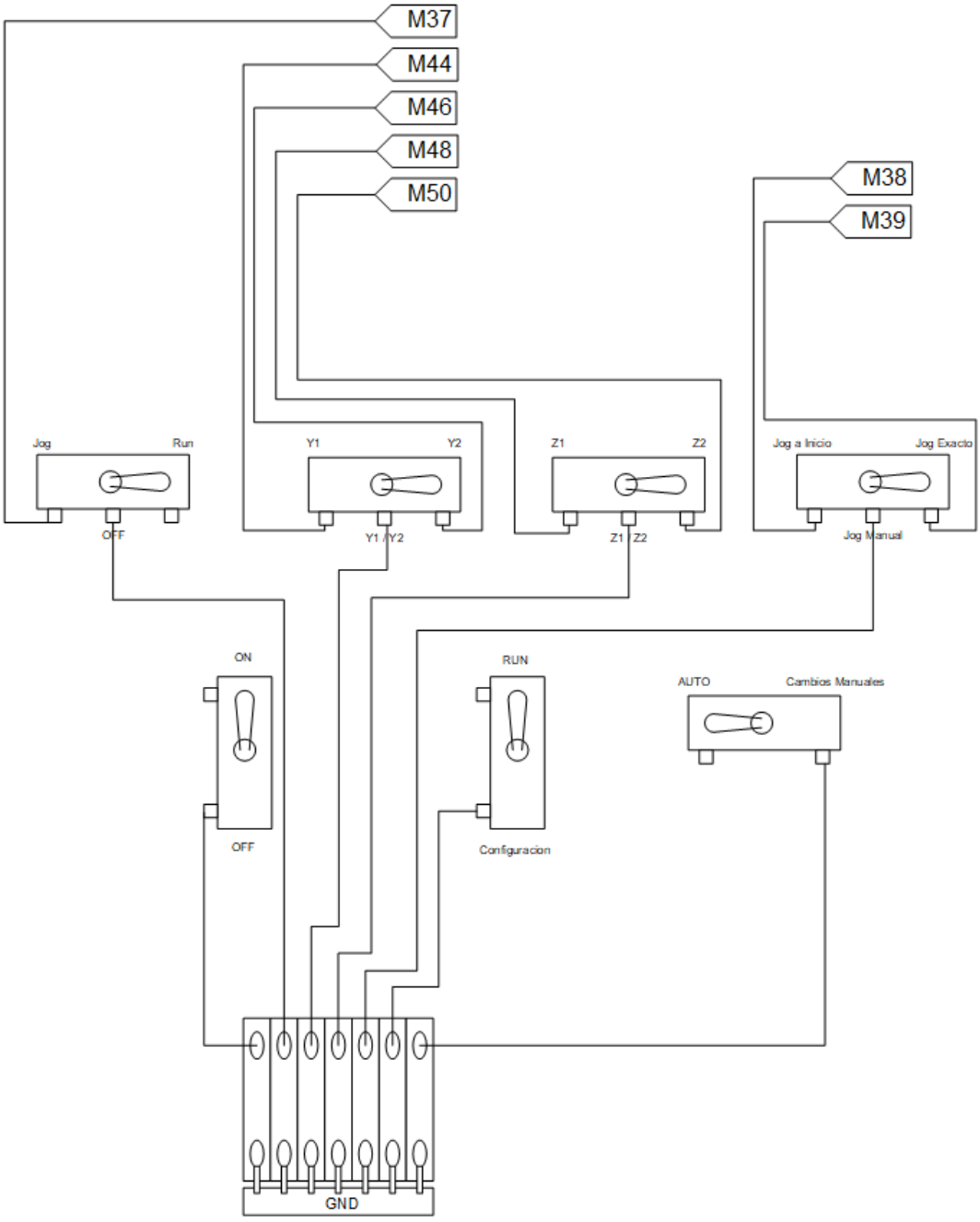


Figura 6.11: Plano eléctrico de conexiones de selectores del panel de control.

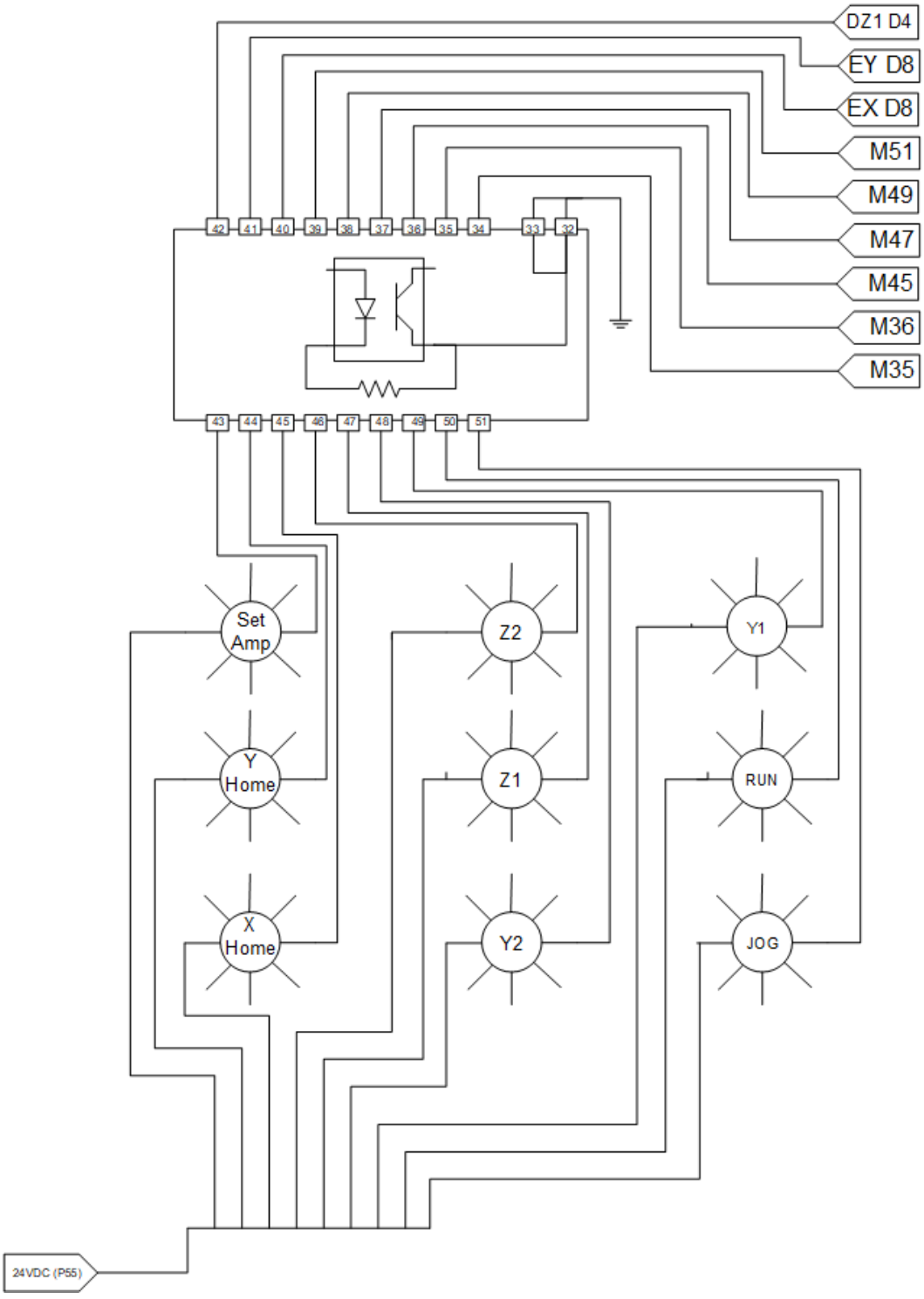


Figura 6.12: Plano eléctrico de conexiones de luces piloto del panel de control.

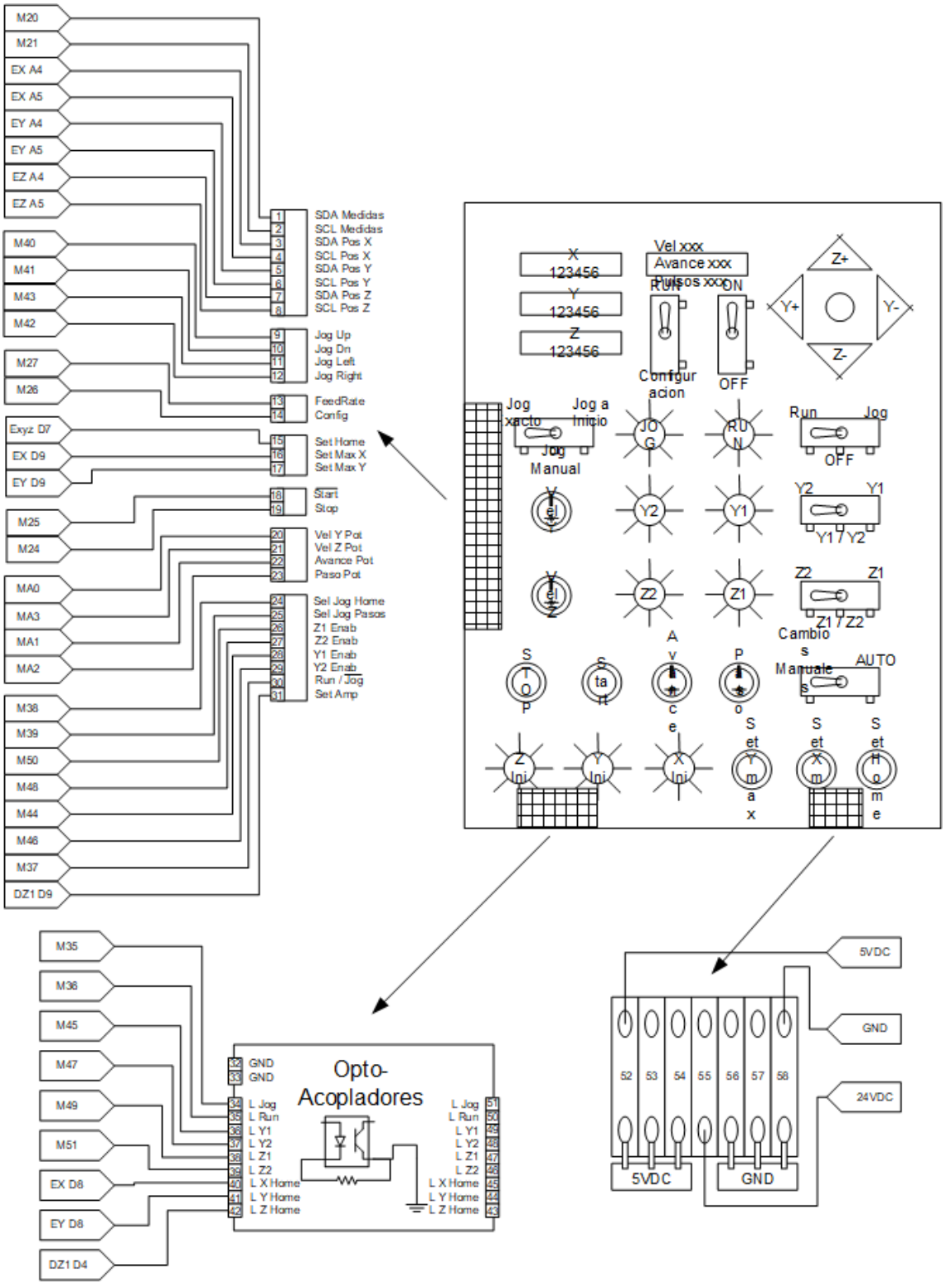


Figura 6.13: Plano eléctrico de conexiones del panel de control.

6.3. Cableado estructurado de gabinetes de control y potencia

En las figuras a continuación se puede observar como fue realizado el cableado estructurado dentro de los gabinetes de control y potencia, tomando como base la conexión de la red de maestro-esclavos y la conexión de los paneles de operador hacia el gabinete respectivo.

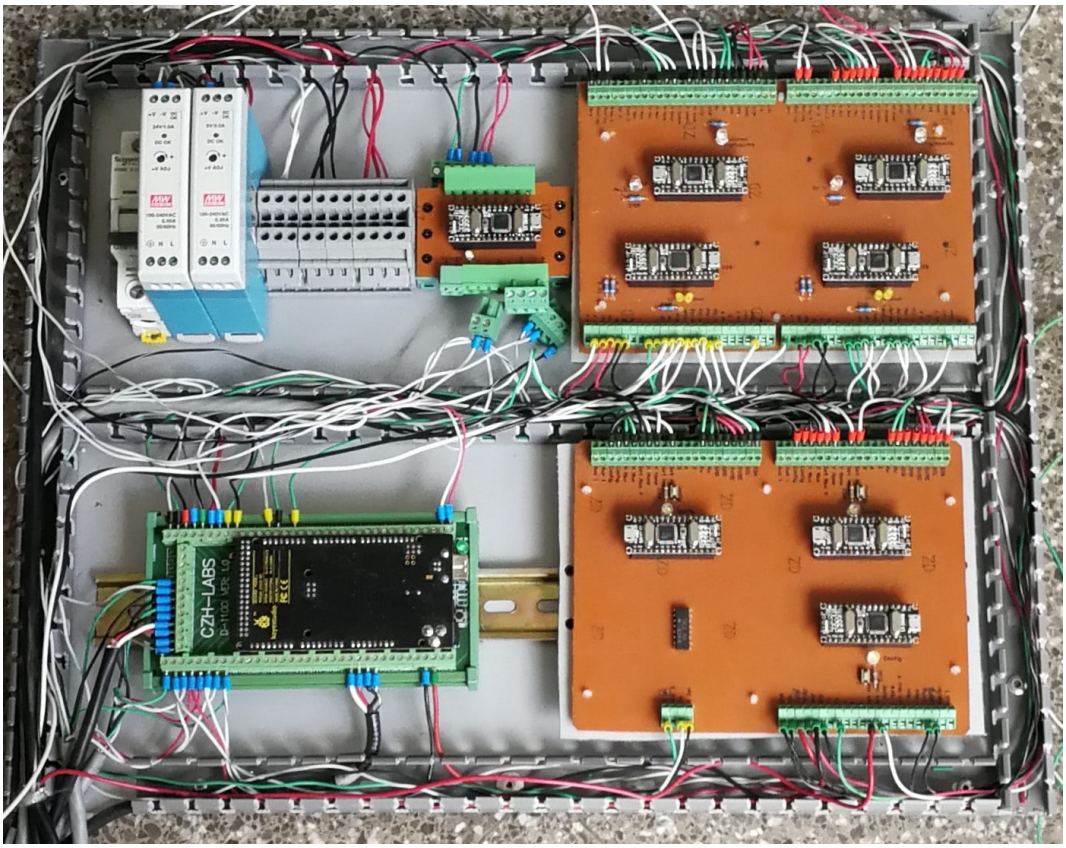


Figura 6.14: Cableado de conexiones de la integración de la red maestro-esclavos.

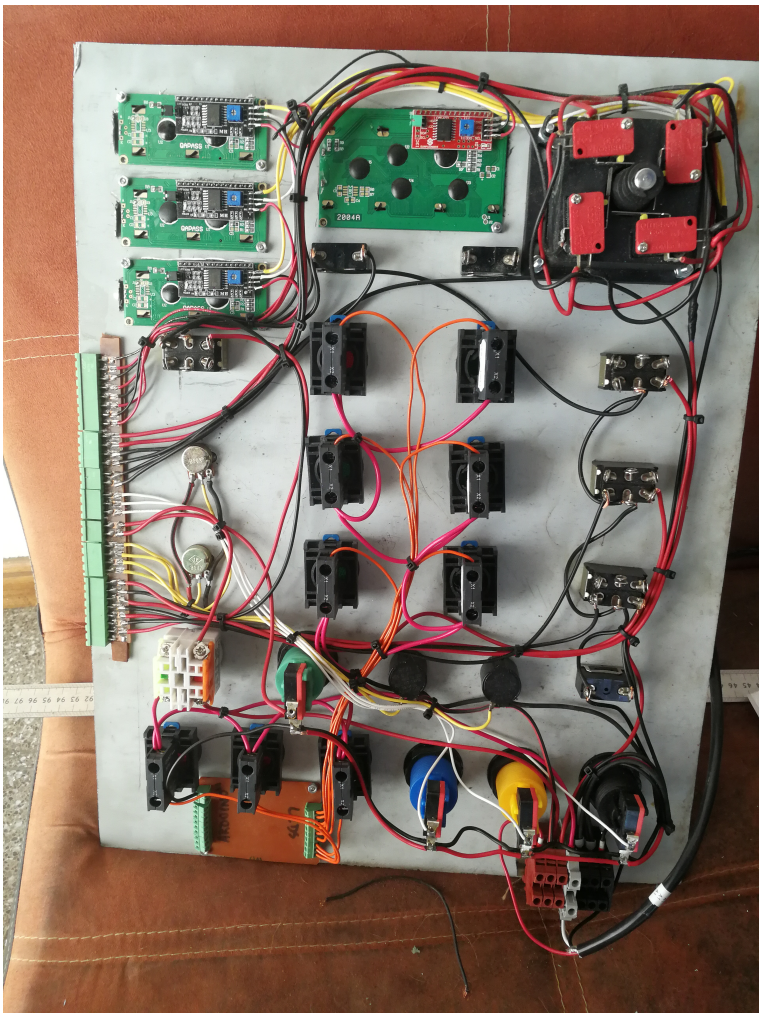


Figura 6.15: Cableado de conexiones del panel de operador a terminales de tornillo de salida.

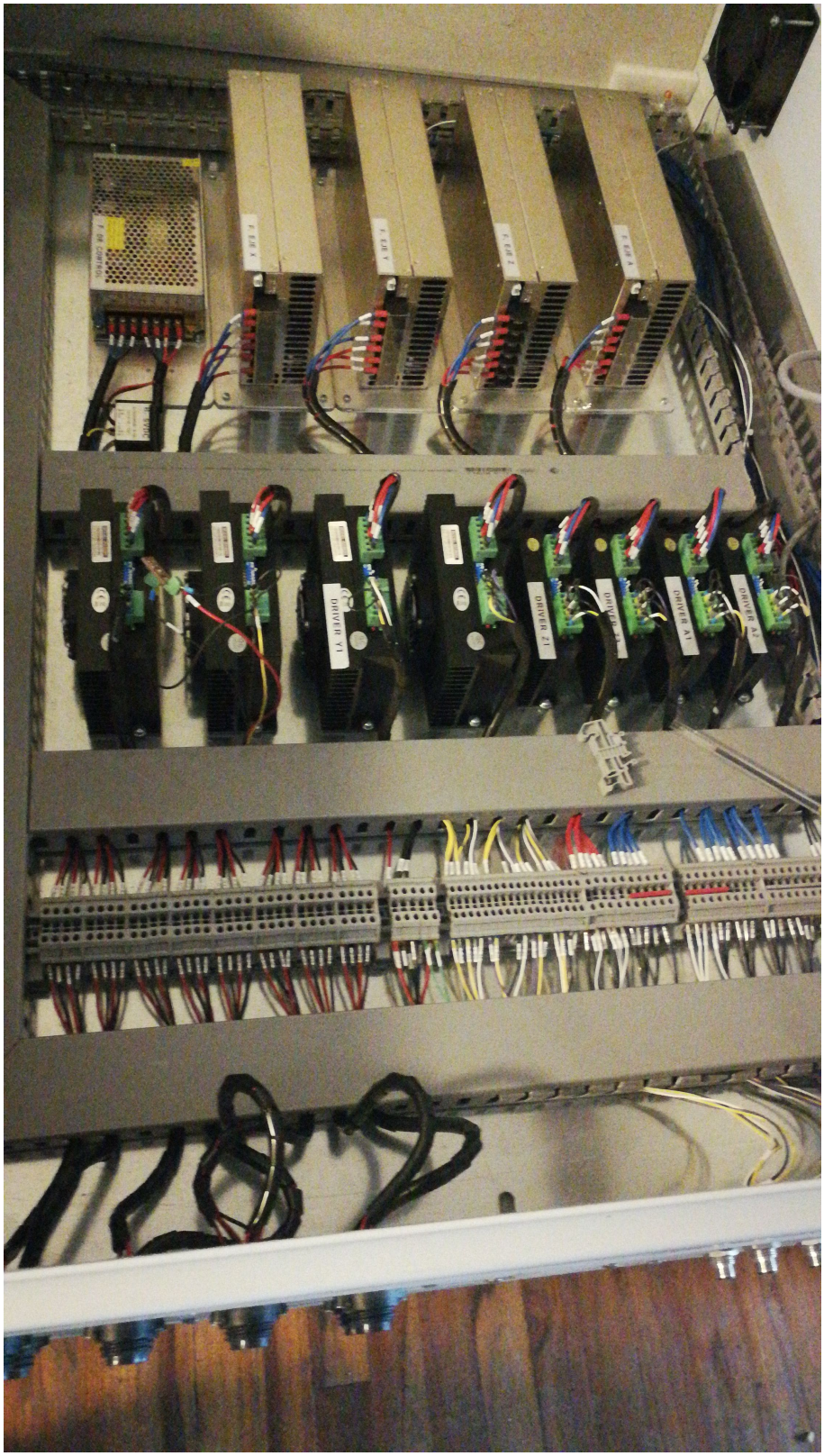


Figura 6.16: Cableado de conexiones del gabinete de potencia.

6.4. Caja reductora

A continuación se presentan las figuras en las cuales se pueden observar los resultados del diseño en software Inventor y elaboración de los elementos y ensamble de la caja reductora que se utilizó en los ejes de movimiento respectivos.

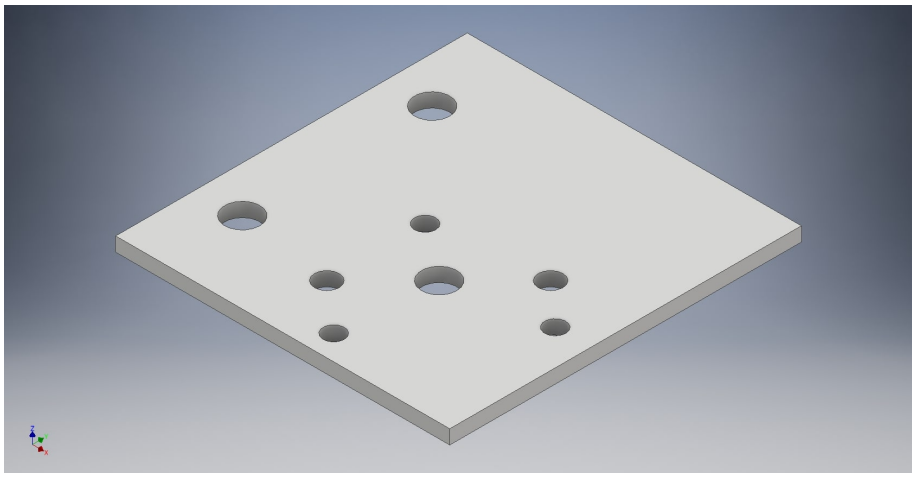


Figura 6.17: Diseño de platina con agujeros para montar al eje.

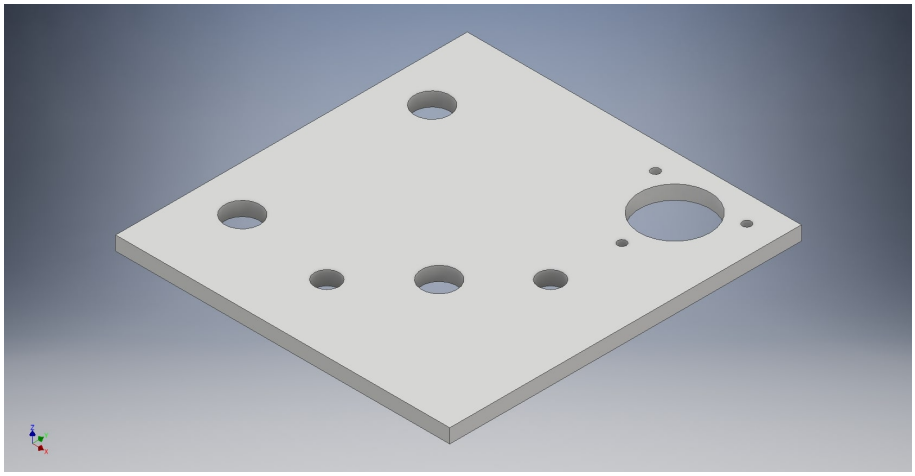


Figura 6.18: Diseño de platina con agujeros para montar el codificador.

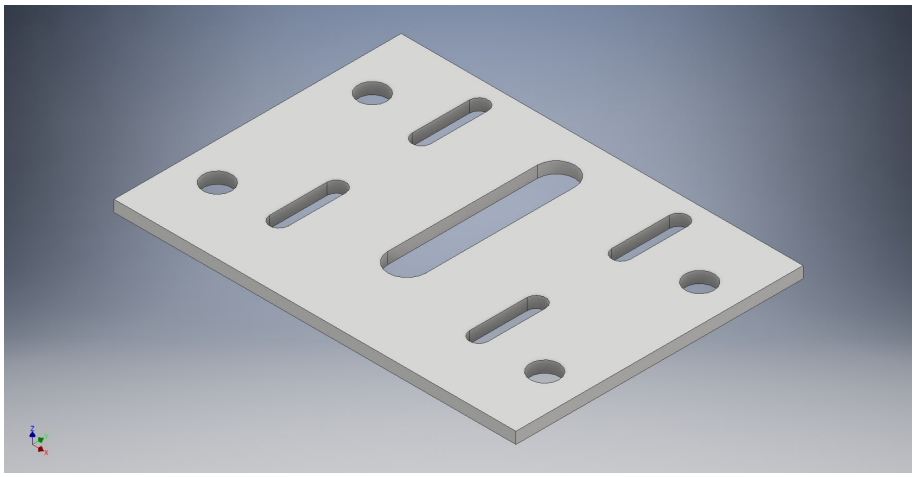


Figura 6.19: Diseño de platina con ranuras para montar el motor paso a paso.

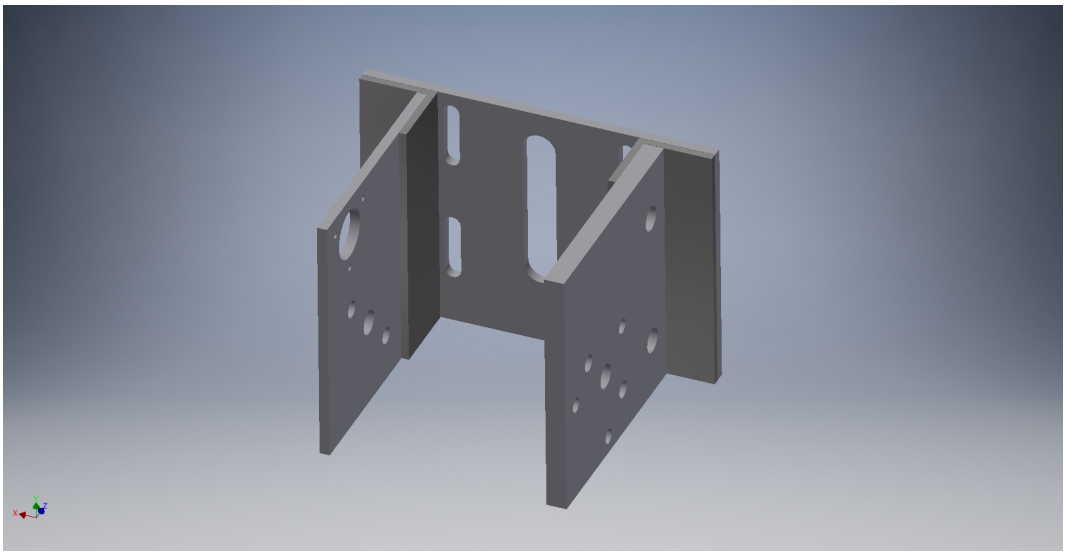


Figura 6.20: Diseño del ensamble de la caja reductora con angulares de fábrica

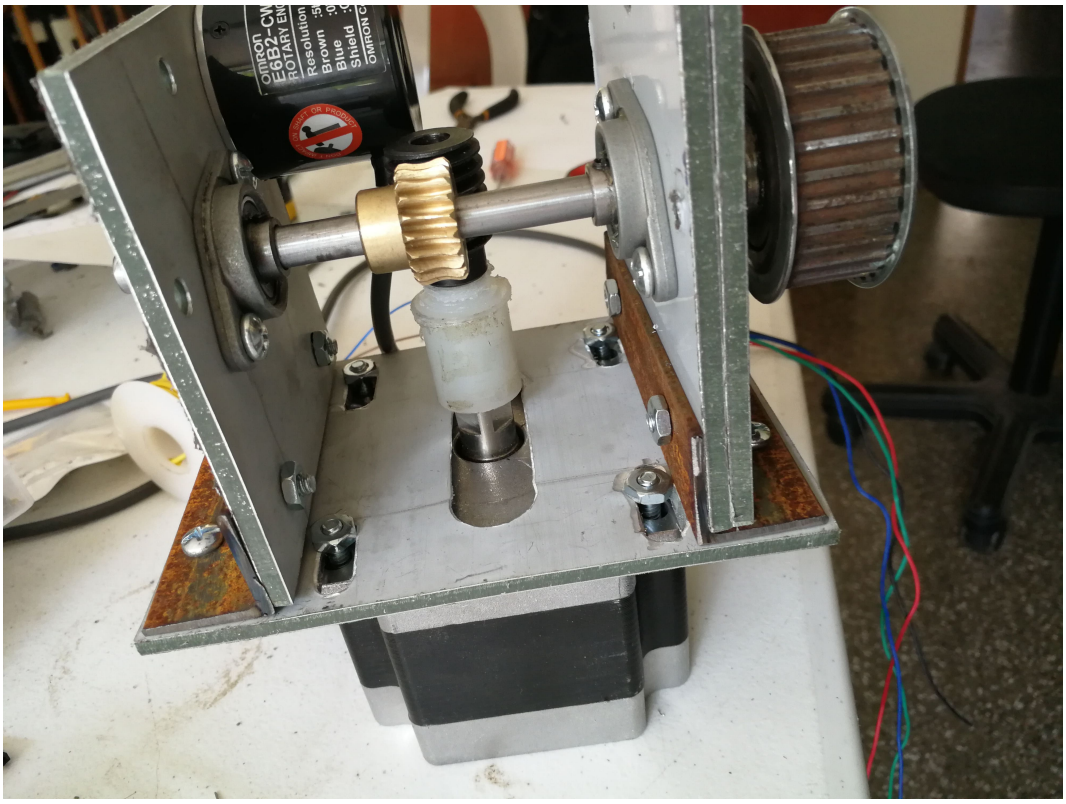


Figura 6.21: Ensamble de la caja reductora elaborada en Alumicon.

El diseño presentado fue el utilizado para la realización de todas las cajas reductoras para cada uno de los ejes.

6.5. Movimiento de ejes con parámetros modificables

Posteriormente se podrán observar figuras y tabla en donde se ven reflejados los resultados obtenidos por medio de la realización de pruebas con un par de motores paso a paso identificados como los "motores Y", para realizar a configuración inicial en la cual se obtiene la cantidad de pulsos requeridos para que el motor realice el desplazamiento de 10 mm para así almacenarlo en la memoria interna del microcontrolador maestro y no volver a realizar ésta configuración inicial, en base a eso realizar una conversión en software para desplazarse una cantidad definida por los parámetros, tomando en cuenta que pueden ser modificados.

Figura 6.22: Conexión de señales de activación, pulsos y dirección de controlador programado hacía controlador de motor paso a paso.

Figura 6.23: Controlador de motor paso a paso utilizado para recibir señales y realizar movimiento de los motores con modo de operación de micro-pasos.

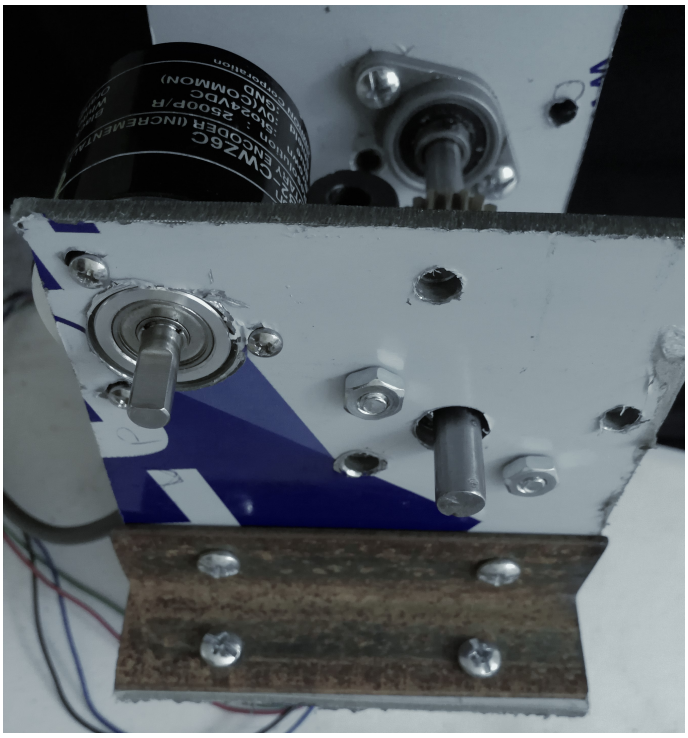


Figura 6.24: Ensamble de codificador rotativo incremental en caja reductora para retroalimentación de cantidad de pasos.

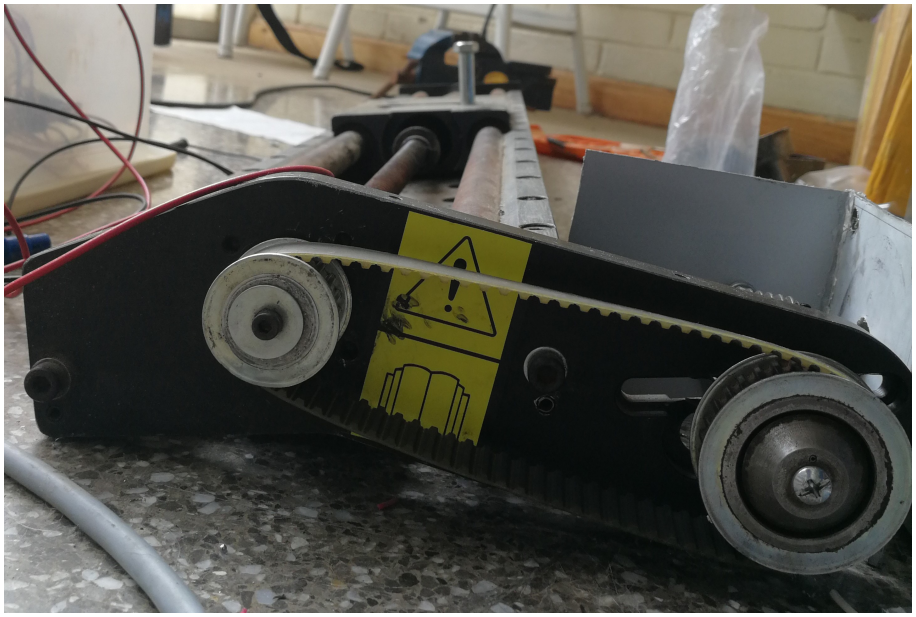


Figura 6.25: Juego de poleas para transmitir el movimiento rotativo al eje principal de movimiento lineal.

Prueba de desplazamiento de motores							
	Valor de desplazamiento deseado			Valor de desplazamiento obtenido por codificador			Porcentaje de error
No. de Prueba	cm	mm	pulsos	cm	mm	pulsos	(%)
1 X	10	100	4,200	9.83	98.3	4,129	1.70 %
1 Y1	10	100	4,200	10.12	101.2	4,251	1.20 %
1 Y2	10	100	4,200	9.93	99.3	4,171	0.7 %
2 X	45	450	18,900	44.6	446	18,732	0.88 %
2 Y1	45	450	18,900	44.8	448	18,816	0.44 %
2 Y2	45	450	18,900	45.3	453	19,026	0.67 %

Tabla 6.1: Resultados obtenidos por el codificador programado, utilizando codificador E6B2-CWZ6C, resolución:2,500 pulsos/revolución.

6.6. Control de potencia de antorchas

En las figuras y tabla que a continuación se presentan se puede visualizar los resultados obtenidos de la medición del sensor de efecto Hall, por medio de una configuración inicial de alturas en el eje z a un valor de voltaje constante de 27V, proporcionado por la fuente de poder ESAB PEJ.

Figura 6.26: Sensor utilizado para lectura de corriente en el proceso de soldadura MIG (rango de operación 50A - 500A).

Figura 6.27: Fuente utilizada en pruebas a 27V constantes.

Configuración inicial voltaje constante 27 V	
Longitud del arco eléctrico (mm)	**Amperaje medido por el sensor (A)**
6	267
8	246
10	230
12	217
14	205

Tabla 6.2: Calibración inicial de antorcha a 27 voltios con hilo metálico de 1.2mm de diámetro par encontrar la longitud de arco ideal

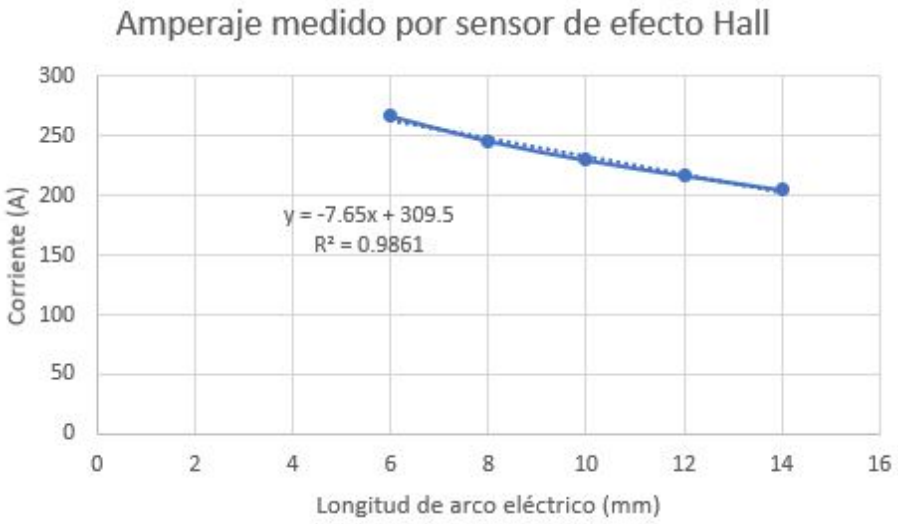


Figura 6.28: Gráfica generada a través de los datos recopilados de la calibración para obtener el comportamiento ideal del motor paso a paso y el sensor de efecto Hall.

Prueba con voltaje constante 27 V Corriente nominal 260 A - Longitud de arco 6.47mm					
Amperaje medido por el sensor	**Longitud del arco eléctrico necesario para alcanzar la corriente nominal**		**Longitud del arco eléctrico proporcionado por el codificador para alcanzar el valor nominal**		**Porcentaje de error**
309.5 (A)	+6.47 (mm)	+272 Pulsos	+6.58 (mm)	+277 Pulsos	1.83 %
260	0	0	+0.09	+4	-
280	+2.62	+110	+2.68	+113	2.72 %
246	-1.83	-77	-1.9	-80	3.89 %
251	-1.18	-50	-1.23	-52	4.00 %
275	+1.97	+83	+2	+84	1.20 %

Tabla 6.3: Longitud de arco y pulsos necesario para llegar al valor nominal de 260 Amperios

CAPÍTULO 7

Discusión

7.1. Paneles de operador

Los paneles de control elaborados obtuvieron excelentes resultados al ser utilizados por los operadores en la simulación de operación, ya que cada uno de los componentes con los que cuenta el panel se encuentran debidamente rotulados de manera que el operario no tenga ningún problema para determinar la función que realiza cada uno de estos componentes. También se contó con pulsadores de distintos colores para indicar si el componente debe utilizarse con mayor cuidado ya que puede causar un daño en el proceso o si es de importancia media ya que no provocaría un daño, como se puede observar en la Figura 6.4 los pulsadores de color rojo se utilizaron como indicadores con los que se debe de tener mucha precaución, el de color verde indica que son de mucha importancia para el proceso, como lo es el pulsador de start, ya que si no es presionado el proceso nunca dará inicio y los demás colores indican una importancia media, ya que al ser presionados se indica un valor que se tendrá de referencia de posicionamiento previo a dar inicio al modo automático.

Con respecto a los indicadores luminosos, se conserva el significado del color rojo, ya que al estar en esa configuración, como lo es el indicador de JOG, se debe de tener precaución al operar la maquinaria, porque en esa modalidad, el operario puede hacer uso de la maquinaria de forma manual, en la cual se puede, por más mínimo que sea, obtener algún daño en el proceso. También cabe destacar que se utilizaron selectores de grado industrial, los cuales al ser ubicados en una posición, estos evitan que de manera fácil, ya sea por accidente, sean accionados o desplazados a otra posición durante el proceso automático esté activo.

Se utilizaron potenciómetros de alta precisión, ya que por medio de estos componentes se realizará el ingreso de los parámetros requeridos por el proceso, permitiendo que éstos valores sean modificables previamente a dar inicio al proceso.

Las pantallas que se colocaron fueron configurados con un contraste ideal para la cantidad de luz con la que se trabaja en el ambiente en la que se encuentra ubicado el panel, de esta manera se hace mucho más sencillo poder configurar el panel de una manera eficiente,

permitiendo también la modificación del mismo, facilitando la opción de que si se desea trasladar el panel o realizar cambios en el área, respecto a la iluminación, en la que se encuentra el panel, dicho contraste no perjudique al momento de ingresar las configuraciones. Lo cual fue muy bien entendido y asimilado por el usuario obteniendo un cumplimento del tercer objetivo planteado este proyecto.

Por lo tanto se puede afirmar que se cumplió con el objetivo planteado de diseñar y elaborar paneles de operador de fácil uso y amigable al usuario.

7.2. Planos eléctricos y cableado

Los resultados obtenidos en este segmento del proyecto, se consideraron satisfactorios, ya que debido a la elaboración de los planos eléctricos de la interconexión de los componentes del sistema, tanto de la red de microcontroladores maestro-esclavos como de los paneles de operador hacia el tablero principal, se permitió realizar un mejor y más rápido cableado de los mismos, ya que al estar bien identificadas las rutas de la conexión desde los pines del microcontrolador maestro hacía los pines de los microcontroladores esclavo, se pudo ubicar de mejor mejor manera la cantidad de cable a utilizar por medio de las canaletas instaladas, como se pudo observar en la Figura 6.14. Cumpliendo así con el quinto objetivo planteado, el cual indica la elaboración de los planos eléctricos antes mencionados.

Se puede afirmar el cumplimiento del cuarto objetivo, debido a que se considera que el cableado respectivo de cada componente interno del gabinete de control y potencia es estructurado, como se puede observar en las figuras 6.14, 6.15 y 6.16, respetando un código de color respectivamente en cada uno de los módulos de la interconexión, tanto de los cables como de las capuchas utilizadas a los extremos de los mismos y el etiquetado respectivo en dichos extremos de cable indicando tanto qué señal será la que se manejará como hacia donde va conectado, también se considera que cumple esta cualidad debido a que hace uso de canaletas para poder situar los cables, estableciendo los recorridos que tomarán dichos cables, para así trazar la mejor ruta de cableado para realizar las conexiones minimizando la cantidad de cable a utilizar y permitiendo mantener un estado de orden dentro de los gabinetes.

7.3. Caja reductora

Los resultados de la elaboración y ensamble de los elementos de la caja reductora se consideran correctos, ya que al ensamblarlos de tal manera permite que, al agregarle el motor paso a paso, se pueda ajustar en dado caso se desee cambiar de tamaño de motor debido a las ranuras que se pueden visualizar en la Figura 6.19. También se concluye con esta afirmación debido al diseño de la platina que se puede observaren la Figura 6.17, ya que posee un mayor espesor y agujeros en los cuales se ensambla al eje de manera fija, permitiendo que el área en contacto con los tornillos de fijación sea mayor, permitiendo una mayor sujeción permitiendo una mayor resistencia debido al peso de los demás elementos que caracterizan a la caja reductora. Otra razón por la cual se escogió un sistema de engranaje de tornillo sin fin y corona, es porque al tener una mayor capacidad de reducción de velocidad, éste

proporciona una mayor precisión al realizar los movimientos respectivos, otro motivo de la elección es por el uso del mecanismo de engranaje en función de freno mecánico, ya que éste no permite que la corona transmita el movimiento al tornillo sin fin evitando algún futuro problema en la máquina. Se optó por este mecanismo por la facilidad de ubicación del motor paso a paso y para el fácil desensamble, al momento de realizar algún cambio, debido a que deben de estar a 90^{o} uno del otro.[2] Por otra parte este diseño de caja reductora permite la unión de otro mecanismo para transmitir el movimiento rotatorio desde el eje hacia un codificador incremental, el cual proporciona la retroalimentación de la posición luego de realizarse la configuración inicial de la máquina. Por lo tanto se puede afirmar que el sexto objetivo de cumple.

7.4. Movimiento de ejes con parámetros modificables

Los resultados obtenidos para el movimiento de los ejes X y Y se puede considerar satisfactorio, ya que luego de ingresados los valores de la calibración de la cantidad de pulsos necesarios para cumplir con 1cm de movimiento lineal (420 pulsos), se realizó el procedimiento de ingresar el parámetro de la ubicación de posicionamiento deseado de los respectos ejes, a la misma velocidad de movimiento y realizar dicha prueba tres veces para corroborar los resultados, se obtuvo que en las primeras tres corridas con los mismos parámetros de posicionamiento, los ejes se posicionaron en un valor muy cercano, con un valor de 99.3mm equivalentes a 4,251 pulsos siendo el eje Y2 con un porcentaje de error de 0.7 % en la segunda prueba que se realizó con otro valor de posicionamiento, se obtuvo un valor cercano, correspondiente al eje Y1 con valores de 448mm y 18,816 pulsos con un porcentaje de error de 0.44 % siendo los valores más bajos obtenidos. Estos resultados se puede considerar aceptables, debido a que la diferencia fue únicamente de 0.7mm en el primer caso y de 2 mm en el segundo caso, a pesar de cuadruplicar la distancia a recorrer. Por lo tanto se puede afirmar que se logra cumplir con el primer objetivo planteado.

La falta de precisión se pudo deberse tanto al juego de poleas y faja, ya que a pesar de ser dentada esta puede aun generar un deslizamiento, por más mínimo que sea. [2]

7.5. Control de potencia

Los resultados obtenidos en este módulo de control de potencia de las antorchas se consideran aceptables, debido que al realizar la configuración inicial, se observó de mejor manera el efecto que llega a provocar el arco eléctrico evidenciando un claro cambio en la corriente, lo cual permitió de mejor manera realizar un comportamiento casi lineal en las cinco mediciones realizadas, eso quiere decir, que el sensor de efecto Hall utilizado, posee una gran resolución y un amplio rango capaz de censar la maquinaria, utilizada como lo es la ESAB PEJ observada en la Figura 6.27. Para el ingreso de los parámetros en este módulo, únicamente se ingresó el valor nominal de forma manual , el diámetro del hilo metálico a utilizar, el valor que más se acercó al resultado deseado fue cuando se simuló una deformación de la lámina, tal que era el valor más cercano al valor nominal, el desplazamiento que tenía que realizar era relativamente poco, para ser exactos un valor de únicamente un pulso o de 0.03mm, obteniendo un porcentaje de error de 1.2 %, el cual pudo no ser detectado por el

codificador. La ventaja del control de los motores paso a paso que generan el movimiento en sentido vertical es que no requieren un desplazamiento de gran tamaño como los ejes X y Y cabe destacar que realizar dicho desplazamiento, conlleva a poder interpretar de mejor manera el funcionamiento del sensor de efecto Hall.

Cabe destacar que no fueron necesarios los valores de movimiento de los ejes X y Y debido a que al ser este un proyecto por módulos definidos, no se llegó a implementar el funcionamiento total de la maquinaria. pero cabe destacar que fue un reto lograr estos resultado, afirmando así que con la obtención de estos resultados se considera cumplido el segundo objetivo planteado en este trabajo de graduación.

CAPÍTULO 8

Conclusiones

1. Se implementaron los ejes adicionales a la máquina original, evidenciando el considerado buen movimiento por parte de los ejes y de los motores paso a paso utilizados, obteniendo 0.7mm en el primer caso y de 2 mm en el segundo caso, a pesar de cuadruplicar la distancia a recorrer se obtuvo un porcentaje de error del 0.7 % y 0.44 % respectivamente.

2. Se realizó el control de la potencia de las antorchas y la buena interpretación de los controladores y codificadores respectivos de una manera eficiente, obteniendo un resultado de medición y ajuste de 0.03mm o un pulso con un porcentaje de error del 1.20 %.

3. Por medio de las pruebas realizadas a operarios se obtuvo que los paneles de operador son de fácil uso y se acopla a las normativas acostumbradas.

4. La implementación de una caja reductora con freno mecánico fue de gran importancia para la transmisión de potencia de los motores y la reducción máxima de velocidad para lograr precisión en el movimiento de los ejes y control de potencia.

5. El diseño, elaboración y tanto de los planos eléctricos de interconexiones entre módulos, como la realización de el cableado estructurado fue de suma importancia, para que la conexión fuera eficiente, generando una reducción en los costos de cableado y permitiendo que el mantenimiento se pueda realizar de manera mas sencilla .

6. Se logró cumplir con el margen de error deseado por el cliente el cual no debía de sobrepasar un 5 % en ambos módulos.

CAPÍTULO 9

Recomendaciones

1. Se recomienda para la continuación de este proyecto evaluar la posibilidad de implementar juegos de engranajes para así optar por que la máquina posea un respaldo más matemático y de diseño con sus respectivos análisis.

2. Se recomienda tomar en cuenta el consumo energético de cada uno de los componentes, para así poder realizar un diseño en el que se requiera de menos mandos físicos que puedan afectar al funcionamiento de la maquinaria.

3. Se recomienda para futuros estudiantes, que deseen aplicar por la continuación de éste proyecto, que se implemente un panel táctil junto con una unidad central de procesamiento que permita mayor alcance en la implementación final.

4. Se recomienda tomar en cuenta que se pueden implementar diferentes protocolos de comunicación entre microcontroladores, ya que el utilizado, Serial, no es muy adecuado en procesos de tamaño industrial, o al menos en el lenguaje IDE de aurduino, porque se deben de interpretar muchos datos y saber realizar el manejo de qué dato le corresponde a qué microcontrolador y eso toma tiempo importante en la implementación de la maquinaria en sí.

5. Se recomienda en futuras versiones realizar un cambio en la estructura de los gabinetes, haciendo énfasis en el tema de traslado del mismo, ya que éstos no poseen puedas para su fácil manejo, por ende siempre se trabajó con el panel de manera horizontal.

6. En la elaboración de futuros PCB´s para una mejor y más fija implementación, el utilizar componentes de superficie, ya que éstos permitieron más área de trabajo disponible dentro de los paneles,como de los gabinetes respectivos.

7. Se recomienda participar de mejor manera con los clientes que solicitaron la maquinaria para así tener una idea directa que que es lo que desean y como era el funcionamiento real de la máquina en su inicio.

CAPÍTULO 10

Bibliografía

[1] Grupo Aldakin. *Automatización Industrial y robótica. Qué es y sus claves de éxito.* 2017. URL: http://www.aldakin.com/automatizacion-industrial-robotica-claves-exito/ (visitado 04-09-2018).

[2] R.G. Budynas y J.K. Nisbett. *Diseño en ingeniería mecánica de Shigley.* Vol. 9. Editorial Mc Graw Hill, 2012. ISBN: 9781456245238.

[3] Richard Gordon Budynas, J Keith Nisbett y col. *Shigley's mechanical engineering design.* Vol. 8. McGraw-Hill New York, 2008. ISBN: 97880073121932.

[4] Gabriel José Castro Pinto y Humberto Javier García Hurtado. "Influencia de los parámetros del proceso de soldadura GTAW en la aparicióan de porosidades en juntas de tuberıas de acero SA 106 B". B.S. thesis. 2015.

[5] Alomoto Chicaiza, Kleber David y Carrera Gualoto. "Diseño y construcción de un equipo para automatizar el proceso de soldadura TIG perimetral para los casquetes de radiadores refrigerantes de los transformadores de la fábrica RVR." B.S. thesis. 2015.

[6] Clextral. *Automatización.* 2018. URL: http://www.clextral.com/es/tecnologias-lineas/tecnologias-y-procesos/automatizacion/ (visitado 04-09-2018).

[7] Miguel Ángel Contreras Ribera. "Control de velocidad y dirección de un vehıculo terrestre autónomo". Tesis de mtría. Universidad de Alicante, 2018.

[8] Frank Durda. *Serial and UART tutorial.* 2014. URL: https://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/ (visitado 08-09-2018).

[9] Federación de Enseñanza. "Proceso de Soldadura". En: *Temas para la Educación: revista digital para profecionales de la enseñanza* 6 (2010). Ed. por CC.OO. de Andalucía. ISSN: 1989-4023.

[10] Felipe Santiago Espinosa y Zenón Belarmino Martınez Cruz. "CARGADOR DE APLICACIONES PARA EL MICROCONTROLADOR ATMEGA328P BUSCANDO APROVECHAR LOS RECURSOS DE UNA TARJETA ARDUINO". En: *Pistas Educativas* 38.120 (2018).

[11] A Codina Garcıa. "Posicionamiento y proyección actual del motor de paso en aplicaciones industriales.//Position and proyection in actual industrial applications of stepping motors." En: *Ingenierıa Mecánica* 5.2 (2002), págs. 65-73.

[12] Luis Gonzales y Karen Macias. *Tablero de Control Eléctrico*. Inf. téc. Instituto Politécnico Nacional, 2013.

[13] Francisco Gonzalez. "Guía Práctica Para Prevenir Deformaciones Por Soldadura". Tesis de mtría. Universidad Austral de Chile, 2006.

[14] Mikell P Groover y col. *Introducción a los procesos de manufactura*. McGraw Hill Education, 2014.

[15] Priyansha Gupta. "Implementing Security in a Personal Security Device". En: *University of California* (2013).

[16] National Instruments. *Comunicación Serial: Conceptos Generales*. 2006. URL: `http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1` (visitado 09-09-2018).

[17] Douglas W. Jones. *Control of Stepping Motors*. Ed. por The University of Iowa Department of Computer Science. 2014. URL: `http://homepage.divms.uiowa.edu/~jones/step/index.html#abstract` (visitado 05-09-2018).

[18] Frank Mecafenix. *Motor paso a paso ¿que es y como funciona?* 2017. URL: `http://www.ingmecafenix.com/electricidad-industrial/motor-paso-a-paso/` (visitado 05-09-2018).

[19] Bob Miller. "Preguntas frecuentes acerca de Soldaduras de Blindaje – Hardfacing". En: *Hardface Tecnologies* (2016).

[20] Norberto Morel. "Técnico electricista - Protecciones eléctricas y tableros: Curso visual y práctico". En: *RedUsers* 14 (2016).

[21] Pedro Pérez Roig. "Diseño de un metro contador de energıa monofásico con microcontrolador PIC." En: (2011).

[22] Ayovı Plata, Luis Armando y John Roberto Mero Sosa. "Construcción de un sistema de entrenamiento para microcontroladores ATMEL para la EIE-CRI." B.S. thesis. Escuela Superior Politécnica de Chimborazo, 2011.

[23] Carlos Christian Retamozo Meza. "Estudio del comportamiento mecánico de los materiales durante el proceso de soldadura manual por arco eléctrico con electrodo revestido". En: (2015).

[24] Pedro Rodriguez. *Manual de soldadura*. Francisco Etchelecu, 2013.

[25] M. Solá. *Soldadura industrial: clases y aplicaciones*. Colección productiva. Marcombo, 1992. ISBN: 9788426708755.

[26] S Torres y col. "Evolución microestructural de la aleación de aluminio 6061 durante el proceso de soldadura MIG". En: *Ingenierıa y Desarrollo* 12 (2002).

[27] Ignacio Vargas y col. *AUTOMATIZACION DE PROCESOS INDUSTRIALES SENSORES DE CORRIENTE*. Inf. téc. Universidad de Talca, Campus Los Niches, 2011.

[28] Jose Villajulca. "Comenzando desde la nada: El tablero electrico". En: *CURSO AVANZADO PLC'S* (2012).

CAPÍTULO 11

Anexos

11.1. Códigos de programación

11.1.1. Microcontrolador maestro

A continuación se encuentra el código realizado y utilizado para el microcontrolador ATmega2560 denominado como Maestro, encargado de el manejo de parámetros y comunicación con controladores y codificadores.

```
#include  <Wire.h>   //I2C
#include  <LiquidCrystal_I2C.h>   //Liquid cristal
#include <EEPROM.h>

//Crear el objeto lcd  dirección  0x27 y 16 columnas x 2 filas
LiquidCrystal_I2C lcd(0x27,20,4);  //


//INPUTS
const int JUp = 40;       //Jog up
const int JDn = 41;       //Jog Down
const int JRi = 42;       //Jog right
const int JLe = 43;       //Jog Left
const int JXU = 33;       //Jog X Up
const int JXD = 34;       //Jog X Down
const int RJ = 37;        //Selector Run/Jog
const int JS1 = 38;       //Selector1 de tipo de Jog JS1 HIGH Mov hasta Limit Switch
const int JS2 = 39;       //Selector2 de tipo de Jog JS2 HIGH Mov Manual  JS1 y JS2 HIGH Movimiento exacto
const int Y1 = 44;        //Enable de Y1
const int Y2 = 46;        //Enable de Y2
const int Z1 = 48;        //Enable de Z1
const int Z2 = 50;        //Enable de Z2
const int FR = 27;        //FeedRate enable
const int Conf = 26;      //Master configuracion
const int Start = 25;     //Boton de Start
const int Stop = 24;      //Boton de Stop
const int SensorBarra= 52; //Sensor de Barra

//Analogos
const int Vel = A0;       //Potenciometro de velocidad de movimiento (frecuencia) motor Y
const int Pul = A2;       //Potenciometro de Cantidad de pulsos a dar
```

```
const int Ava = A1;       //Potenciometro de candidad de pulsos a avanzar
const int VelZ = A3;       //Potenciometro de velocidad motor Z
const int VelX = A4;       //Potenciometro de velocidad motor Z

//OUTPUTS
const int pY1 = 45;       //Luz piloto de Y1
const int pY2 = 47;       //Luz piloto de Y2
const int pZ1 = 49;       //Luz piloto de Z1
const int pZ2 = 51;       //Luz piloto de Z2
const int pR = 36;       //Luz de Run
const int pJ = 35;       //Luz de JOG
const int S1 = 30;       //S1 Indicador de limite de movimiento en X+ (arriba)
const int S2 = 29;       //S1 Indicador de limite de movimiento en X- (abajo)
const int S3 = 28;       //S1 Indicador de limite de movimiento en Y+ (derecha)
const int MuxA = 22;      //Serial Mux A de encoder
const int MuxB = 23;      //Serial Mux B de encoder


//VARIABLES
int JOpc = 0;           //Seleccion de tipo de JOG
String sJOpc = "";      //String de OPCION DE JOG a mandar
int JOpcZ = 0;           //Seleccion de tipo de JOG en Z
String sJOpcZ = "";      //String de OPCION DE JOG a mandar en Z
String stJOpc ="";      //Temporal de opcion de Y
int JDY = 0;            //Direccion de movimiento en Y
int JDZ = 0;            //Direccion de movimiento en Z
String sJDY = "";       //String de direccion de Y
String sJDZ = "";       //String de direccion de Z
String sJDX = "";       //String de direccion de X

int iVel = 0;           //Lectura de potenciometro de velocidad
int aVel = 0;           //Mapeo de potenciometro de velocidad
String sVel ="";        //Temporal de velocidad Convertido a string
String saVel = "";      //Sting de VELOCIDAD o FRECUENCIA a mandar

int aPul = 0;           //Lectura de potenciometro de Pulsos
String saPul = "";      //Sting de Pulsos exactos a mandar
String sPul ="";        //Temporal de pulso
int iPul = 0;

int iAva = 0;
int aAva = 0;           //Lectura de potenciometro de avance
String saAva = "";      //Sting de Avance a mandar
String sAva ="";        //Temporal de avance

int aVelZ = 0;           //Lectura de potenciometro de velocidad de Z
String saVelZ = "";      //Sting de VELOCIDAD o FRECUENCIA a mandar de Z
int iVelZ = 0;           //Variable a mandar de velocidad de Z
String sVelZ ="";        //Temporal de velocidad de Z

int aVelX = 0;           //Lectura de potenciometro de velocidad de X
String saVelX = "";      //Sting de VELOCIDAD o FRECUENCIA a mandar de X
int iVelX = 0;           //Variable a mandar de velocidad de X
String sVelX ="";        //Temporal de velocidad de X

int PulsoXcmY = 0;       //cantidad de pulsos por centimetro (pedido de encoder Y)

String inputString = "";

void setup() {
  pinMode(JUp,INPUT);
  pinMode(JDn,INPUT);
  pinMode(JRi,INPUT);
  pinMode(JLe,INPUT);
  pinMode(JXU,INPUT_PULLUP);
  pinMode(JXD,INPUT_PULLUP);
  pinMode(RJ,INPUT_PULLUP);
  pinMode(JS1,INPUT_PULLUP);
```

```
  pinMode(JS2,INPUT_PULLUP);
  pinMode(Y1,INPUT_PULLUP);
  pinMode(Y2,INPUT_PULLUP);
  pinMode(Z1,INPUT_PULLUP);
  pinMode(Z2,INPUT_PULLUP);
  pinMode(FR,INPUT_PULLUP);
  pinMode(Conf,INPUT_PULLUP);
  pinMode(Start,INPUT_PULLUP);
  pinMode(Stop,INPUT_PULLUP);
  pinMode(SensorBarra,INPUT_PULLUP);

  pinMode(pY1,OUTPUT);
  digitalWrite(pY1,LOW);
  pinMode(pY2,OUTPUT);
  digitalWrite(pY2,LOW);
  pinMode(pZ1,OUTPUT);
  digitalWrite(pZ1,LOW);
  pinMode(pZ2,OUTPUT);
  digitalWrite(pZ2,LOW);
  pinMode(pR,OUTPUT);
  digitalWrite(pR,LOW);
  pinMode(pJ,OUTPUT);
  digitalWrite(pJ,LOW);

  pinMode(MuxA,OUTPUT);
  digitalWrite(MuxA,LOW);
  pinMode(MuxB,OUTPUT);
  digitalWrite(MuxB,LOW);


  lcd.init();
  lcd.backlight();

  Serial3.begin(9600);      //Serial compu
  Serial2.begin(9600);      //Serial a Drivers
  Serial1.begin(9600);      //Serial a encoders
  Serial.begin(9600);       //Serial a HMI



}

void loop() {
  if ((digitalRead(RJ)==HIGH) && (digitalRead(Conf)==LOW) && (digitalRead(JUp)==HIGH)) {
        lcd.clear();
        lcd.setCursor(8,0);
        lcd.print("APSA");
        lcd.setCursor(3,1);
        lcd.print("Configurando" );
        lcd.setCursor(3,2);
        lcd.print("pulsos por cm");
        inputString = "";
        delay(1000);
        PideCm(2);
  }
  Enables();
  Display();

  if (digitalRead(RJ) == LOW) {             //Verifica que esta en JOG
    digitalWrite(pJ,HIGH);                  //Prende luz de JOG
    digitalWrite(pR,LOW);                   //Apaga luz de RUN
    if ((digitalRead(JS1) == HIGH) && (digitalRead(JS2) == LOW)) JOpc = 1; //Jog para llegar a Limit Switch
    if ((digitalRead(JS1) == HIGH) && (digitalRead(JS2) == HIGH)) JOpc = 3;//Jog Manual
    if ((digitalRead(JS1) == LOW) && (digitalRead(JS2) == HIGH)) JOpc = 2; //Jog Exacto
    stJOpc = String(JOpc);
    sJOpc = "JO" + stJOpc;                    //String de opcion de jog

    //Lectura y mapeo de velocidades para Jog
```

```
    iVel = analogRead(Vel);                //Lectura de Pot de Velocidad en Y
    aVel = map(iVel, 0, 1023, 200, 2500);   //Mapeo de velocidad en Y
    sVel = String(aVel);
    saVel = "JV" + sVel;                   //String de velocidad en Y

    iVelZ = analogRead(VelZ);              //lee Pot Vel Z
    aVelZ = map(iVelZ, 0, 1023, 200, 2500);   //Mapeo de velocidad En Z
    sVelZ = String(aVelZ);
    saVelZ = "ZJV" + sVelZ;                    //String de velocidad de Z

    iVelX = analogRead(VelX);                 //lee Pot Vel X
    aVelX = map(iVelX, 0, 1023, 200, 2500);   //Mapeo de velocidad en X
    sVelX = String(aVelX);
    saVelX = "XJV" + sVelX;                    //String de velocidad de X

    iPul = analogRead(Pul);                //Lectura de Pot de Pulsos
    aPul = map(iPul, 0, 1023, 4000, 0);     //Mapeo de Pulsos (ver cuantas vueltas se necesitan para llegar
    al final)
    sPul = String(aPul);
    saPul = "JP" + sPul;                   //String de Pulsos

    //Movimientos en Y

    if ((digitalRead(JRi) == HIGH) || (digitalRead(JLe) == HIGH)) {
      MovYJog();
    }

     //Movimientos en Z
    if ((digitalRead(JUp) == HIGH) || (digitalRead(JDn) == HIGH)) {
      MovZJog();
    }

  }else {
    digitalWrite(pJ,LOW);                  //Apaga luz de JOG
    digitalWrite(pR,HIGH);                 //Prende luz de RUN
    //Programacion de RUN

    //Lectura y mapeo de velocidades para RUN
    iVel = analogRead(Vel);                //Lectura de Pot de Velocidad
    aVel = map(iVel, 0, 1023, 200, 2500);   //Mapeo de velocidad
    sVel = String(aVel);
    saVel = "f" + sVel;                   //String de velocidad

    iVelZ = analogRead(VelZ);              //lee Pot Vel Z
    aVelZ = map(iVelZ, 0, 1023, 200, 2500);   //Mapeo de velocidad En Z
    sVelZ = String(aVelZ);
    saVelZ = "Zf" + sVelZ;                    //String de velocidad de Z

    iVelX = analogRead(VelX);                 //lee Pot Vel X
    aVelX = map(iVelX, 0, 1023, 200, 2500);   //Mapeo de velocidad en X
    sVelX = String(aVelX);
    saVelX = "Xf" + sVelX;                    //String de velocidad de X

    //Lectura y mapeo de pasos(Pul) y avance (Ava)
    iPul = analogRead(Pul);                //Lectura de Pot de Pulsos
    aPul = map(iPul, 0, 1023, 5000, 0);     //Mapeo de Pulsos (2 vueltas = 5000 pulsos)
    sPul = String(aPul);
    saPul = "p" + sPul;                   //String de Pulsos

    iAva = analogRead(Ava);                //Lectura de Pot de Pulsos
    aAva = map(iAva, 0, 1023, 5000, 0);     //Mapeo de avance (2 vueltas = 5000 pulsos)
    sAva = String(aAva);
    saPul = "a" + sAva;                   //String de Avance
  }

}

void Enables() {
```

```
  if ((digitalRead(SensorBarra)==HIGH)&&(digitalRead(Y1)==HIGH)&&(digitalRead(Y2)==HIGH)){
    digitalWrite(pY1,LOW);
    digitalWrite(pY2,LOW);
    digitalWrite(pJ,LOW);
    delay(200);
    digitalWrite(pJ,HIGH);
    delay(200);
  }
  if((digitalRead(SensorBarra)==HIGH)&&(digitalRead(Y1)==HIGH)&&(digitalRead(Y2)==LOW)){
    digitalWrite(pY1,HIGH);
    digitalWrite(pY2,LOW);
    Serial2.println("MY4");
    Serial.println("MY4");
    }
  if((digitalRead(SensorBarra)==HIGH)&&(digitalRead(Y1)==LOW)&&(digitalRead(Y2)==HIGH)){
    digitalWrite(pY1,LOW);
    digitalWrite(pY2,HIGH);
    Serial2.println("MY5");
    Serial.println("MY5");
    }
  if(digitalRead(SensorBarra)==LOW){

    if (digitalRead(Y1) == HIGH) {digitalWrite(pY1,HIGH);} else {digitalWrite(pY1,LOW);}
    //Indicador de Y1 Enable
    if (digitalRead(Y2) == HIGH) {digitalWrite(pY2,HIGH);} else {digitalWrite(pY2,LOW);}
    //Indicador de Y2 Enable
    if (digitalRead(Z1) == HIGH) {digitalWrite(pZ1,HIGH);} else {digitalWrite(pZ1,LOW);}
    //Indicador de Z1 Enable
    if (digitalRead(Z2) == HIGH) {digitalWrite(pZ2,HIGH);} else {digitalWrite(pZ2,LOW);}
    //Indicador de Z2 Enable
    if((digitalRead(Y1)==HIGH)&&(digitalRead(Y2)==LOW)) Serial.println("MY1");
    if((digitalRead(Y2)==HIGH)&&(digitalRead(Y1)==LOW)) Serial.println("MY2");
    if((digitalRead(Y1)==HIGH)&&(digitalRead(Y2)==HIGH)) Serial.println("MY3");
    }
}

void MovXJog() {
      if (digitalRead(JXU) == LOW) sJDZ = "JD1";         //Direccion arriba
      if (digitalRead(JXD) == LOW) sJDZ = "JD0";         //Direccion abajo
      switch (JOpc) {                        //Opciones de JOG
        /*Opcion 1: Se envia hasta que tope el limit switch.  Se envia direccion,
        velocidad y opcion cuando joystick envie una señal, deja de mandar
        cuando regresa a posicion neutra*/
        case 1:
          if (sJDX == "JD0") {            //Solo se puede hacer jog para arriba, no para
          abajo
            break;
          }
          Serial2.println(sJDX);
          delay(100);
          Serial2.println(saVelX);
          delay(100);
          Serial2.println(sJOpc);
          while ((digitalRead(JXU) == LOW) || (digitalRead(JXD) == LOW)) {
            //Espera joystick regrese a posicion neutra
          }
          break;
        /*Opcion 2: Exactitud, se mueve una cantidad de pulsos fija dada por
        el MAP del potenciometro.  Se envia direccion, velocidad, pulsos y
        opcion cuando joystick envie una señal, deja de mandar cuando
        regresa a posicion neutra*/
        case 2:
          Serial2.println(sJDX);
          delay(10);
          Serial2.println(saVelX);
          delay(10);
          Serial2.println(saPul);
          delay(10);
```

```
          Serial2.println(sJOpc);
          while ((digitalRead(JXU) == LOW) || (digitalRead(JXD) == LOW)) {
            //Espera joystick regrese a posicion neutra
          }
          break;
        /*Opcion 3: Manual.  Se envia velocidad, opcion y la codificacion de
        direccion e inicio: 0)mueva a la derecha, 2)mueva a la izquierda.  Esa se
        envia cuando se manda la señal del joystick y al regresar a su posicion
        neutra envia 1)detengase*/
        case 3:
          Serial2.println(saVelX);
          delay(10);
          Serial2.println(sJOpc);
          delay(10);
          if (digitalRead(JXU) == LOW) Serial.println("0");
          if (digitalRead(JXD) == LOW) Serial.println("2");
          while ((digitalRead(JXU) == LOW) || (digitalRead(JXD) == LOW)) {
            //Espera joystick regrese a posicion neutra
          }
          if ((digitalRead(JXU) == HIGH) && (digitalRead(JXD) == HIGH)) {
           // for (int i=0; i <= 2; i++){
            Serial2.println(saVelX);
            delay(10);
            Serial2.println(sJOpc);
            delay(10);
           Serial2.println("1");
           //   delay(50);
            //}
          break;
          }
      }
}

void MovYJog() {

    if (digitalRead(JRi) == HIGH) sJDY = "JD1";         //Direccion a la derecha
    if (digitalRead(JLe) == HIGH) sJDY = "JD0";         //Direccion a la izquierda


      switch (JOpc) {                       //Opciones de JOG
        /*Opcion 1: Se envia hasta que tope el limit switch.  Se envia direccion,
        velocidad y opcion cuando joystick envie una señal, deja de mandar
        cuando regresa a posicion neutra*/
        case 1:
          Serial2.println(sJDY);
          delay(100);
          Serial2.println(saVel);
          delay(100);
          Serial2.println(sJOpc);

          while ((digitalRead(JRi) == HIGH) || (digitalRead(JLe) == HIGH)) {
            //Espera joystick regrese a posicion neutra
          }
          break;
        /*Opcion 2: Exactitud, se mueve una cantidad de pulsos fija dada por
        el MAP del potenciometro.  Se envia direccion, velocidad, pulsos y
        opcion cuando joystick envie una señal, deja de mandar cuando
        regresa a posicion neutra*/
        case 2:
          Serial2.println(sJDY);
          delay(10);
          Serial2.println(saVel);
          delay(10);
          Serial2.println(saPul);
          delay(10);
          Serial2.println(sJOpc);
          while ((digitalRead(JRi) == HIGH) || (digitalRead(JLe) == HIGH)) {
            //Espera joystick regrese a posicion neutra
```

```
          }
          break;
        /*Opción 3: Manual.  Se envia velocidad, opción y la codificacion de
        dirección e inicio: 0)mueva a la derecha, 2)mueva a la izquierda.  Esa se
        envía cuando se manda la señal del joystick y al regresar a su posición
        neutra envia 1)detengase*/
        case 3:
          Serial2.println(saVel);
          delay(10);
          Serial2.println(sJOpc);
          delay(10);

          if (digitalRead(JRi) == HIGH) Serial2.println("0");
          if (digitalRead(JLe) == HIGH) Serial2.println("2");



Serial.println(saVel);
delay(10);
Serial.println(sJOpc);
delay(10);

 if (digitalRead(JRi) == HIGH) Serial.println("0");
 if (digitalRead(JLe) == HIGH) Serial.println("2");

          while ((digitalRead(JRi) == HIGH) || (digitalRead(JLe) == HIGH)) {
            //Espera joystick regrese a posicion neutra
            Serial.println("Mantiene");
          }
          if ((digitalRead(JRi) == LOW) && (digitalRead(JLe) == LOW)) {
           // for (int i=0; i <= 2; i++){
            //Serial2.println(saVel);
            //delay(10);
            Serial2.println(sJOpc);
            delay(10);
           Serial2.println("1");
            delay(10);



            Serial.println(sJOpc);
            delay(10);
           Serial.println("1");
            //}
          break;
          }
      }
}

void MovZJog() {
      if (digitalRead(JUp) == HIGH) sJDZ = "JD1";         //Direccion arriba
      if (digitalRead(JDn) == HIGH) sJDZ = "JD0";         //Direccion abajo
      switch (JOpc) {                       //Opciones de JOG
        /*Opcion 1: Se envia hasta que tope el limit switch.  Se envia direccion,
        velocidad y opcion cuando joystick envie una señal, deja de mandar
        cuando regresa a posicion neutra*/
        case 1:
          if (sJDZ == "JD0") {  //Solo se puede hacer jog para arriba, no para abajo
            break;
          }
          Serial2.println(sJDZ);
          delay(100);
          Serial2.println(saVelZ);
          delay(100);
          Serial2.println(sJOpc);
          while ((digitalRead(JUp) == HIGH) || (digitalRead(JDn) == HIGH)) {
            //Espera joystick regrese a posicion neutra
          }
```

```
          break;
        /*Opcion 2: Exactitud, se mueve una cantidad de pulsos fija dada por
        el MAP del potenciometro.  Se envia direccion, velocidad, pulsos y
        opcion cuando joystick envie una señal, deja de mandar cuando
        regresa a posicion neutra*/
        case 2:
          Serial2.println(sJDZ);
          delay(10);
          Serial2.println(saVelZ);
          delay(10);
          Serial2.println(saPul);
          delay(10);
          Serial2.println(sJOpc);
          while ((digitalRead(JUp) == HIGH) || (digitalRead(JDn) == HIGH)) {
            //Espera joystick regrese a posicion neutra
          }
          break;
        /*Opcion 3: Manual.  Se envia velocidad, opcion y la codificacion de
        direccion e inicio: 0)mueva a la derecha, 2)mueva a la izquierda.  Esa se
        envia cuando se manda la señal del joystick y al regresar a su posicion
        neutra envia 1)detengase*/
        case 3:
          Serial2.println(saVelZ);
          delay(10);
          Serial2.println(sJOpc);
          delay(10);
          if (digitalRead(JUp) == HIGH) Serial.println("0");
          if (digitalRead(JDn) == HIGH) Serial.println("2");
          while ((digitalRead(JUp) == HIGH) || (digitalRead(JDn) == HIGH)) {
            //Espera joystick regrese a posicion neutra
          }
          if ((digitalRead(JUp) == LOW) && (digitalRead(JDn) == LOW)) {
           // for (int i=0; i <= 2; i++){
            Serial2.println(saVelZ);
            delay(10);
            Serial2.println(sJOpc);
            delay(10);
           Serial2.println("1");
           //   delay(50);
            //}
          break;
          }
      }
}

void Display() {
  lcd.setCursor(8,0);
  lcd.print("APSA");

  //Columna 1
  lcd.setCursor(0,1);
  lcd.print("Vel X:");          //Imprime Vel X en %
  //int ipVelX= map(iVelX, 0, 1023, 0, 100);
  //lcd.print(ipVelX);
  lcd.print("%");
  lcd.setCursor(0,2);           //Imprime velocidad de Y en %
  lcd.print("Vel Y:");
  iVel = analogRead(Vel);                 //Lectura de Pot de Velocidad
  int ipVelY= map(iVel, 0, 1023, 0, 100);
  lcd.print(ipVelY);
  lcd.print("%");
  lcd.setCursor(0,3);           //Imprime velocidad de Z en %
  lcd.print("Vel Z:");
  iVelZ = analogRead(VelZ);                 //Lectura de Pot de Velocidad
  int ipVelZ= map(iVelZ, 0, 1023, 0, 100);
  lcd.print(ipVelZ);
  lcd.print("%");
```

```
  //Columna 2
  lcd.setCursor(10,1);
  lcd.print("Vel A:");          //Imprime Vel X en %
  //int ipVelA= map(iVelXA, 0, 1023, 0, 100);
  //lcd.print(ipVelA);
  lcd.print("%");
  lcd.setCursor(10,2);           //Imprime Pasos
  lcd.print("Paso:");
  int cmPul = 0;
  EEPROM.get(5,cmPul);
  iPul = analogRead(Pul);                 //Lectura de Pot de Pulsos
  aPul = map(iPul, 0, 1023, 5000, 0);     //Mapeo de Pulsos (2 vueltas = 5000
  pulsos)
  float ipPul= ((float)aPul / (float)cmPul) * 10;
  lcd.print(ipPul,1);
  lcd.print("mm");
  lcd.setCursor(10,3);           //Imprime Avance
  lcd.print("Avan:");
  int cmPula = 0;
  EEPROM.get(5,cmPula);
  iAva = analogRead(Ava);                 //Lectura de Pot de Pulsos
  aAva = map(iAva, 0, 1023, 5000, 0);     //Mapeo de avance (2 vueltas = 5000
  pulsos)
  float ipAva = ((float)aAva / (float)cmPula) * 10;
  lcd.print(ipAva);
  lcd.print("mm");
}


//Pide cuantos pulsos es 1cm a los encoders (debe mandar 1 para X, 2 para
Y, 3 para Z)
void PideCm(int XYZ) {
  switch (XYZ) {
    case 1:
      digitalWrite(MuxA,HIGH);
      digitalWrite(MuxB,HIGH);         //Selecciona Encoder X
      break;
    case 2:
      digitalWrite(MuxA,LOW);
      digitalWrite(MuxB,HIGH);         //Selecciona Encoder Y
      break;
    case 3:
      digitalWrite(MuxA,HIGH);
      digitalWrite(MuxB,LOW);         //Selecciona Encoder Z
      break;
    case 4:
      digitalWrite(MuxA,LOW);
      digitalWrite(MuxB,LOW);         //Selecciona Encoder Aux
      break;
  }
  delay(1000);
  Serial1.println("cm");
}

void serialEvent1(){
  while (Serial1.available()) {
    char inChar = (char)Serial1.read();
    inputString += inChar;
    if (inChar == '\n') {
      inputString.trim();
      if (inputString.startsWith("c")) {           //Posicion actual
        inputString = inputString.substring(1);
        if ((digitalRead(MuxA)==HIGH) && (digitalRead(MuxB)==HIGH)) {
          EEPROM.put(1,inputString.toInt());            //Graba la posicion en
          EEPROM 1 de pulsos X cm Encoder X
        }
        if ((digitalRead(MuxA)==HIGH) && (digitalRead(MuxB)==LOW)) {
          EEPROM.put(10,inputString.toInt());           //Graba la posicion en
```

```
          EEPROM 10 de pulsos X cm Encoder Z
        }
        if ((digitalRead(MuxA)==LOW) && (digitalRead(MuxB)==HIGH)) {
          EEPROM.put(5,inputString.toInt());            8//Graba la posicion en
          EEPROM 5 de pulsos X cm Encoder Y
        }
        int MuxAB = 0;
        if (digitalRead(MuxA) == HIGH) { MuxAB = MuxAB + 1;}
        if (digitalRead(MuxB) == HIGH) { MuxAB = MuxAB + 2;}

        lcd.clear();
        lcd.setCursor(8,0);
        lcd.print("APSA");
        lcd.setCursor(0,1);
        int TempPos = 0;
        switch (MuxAB) {
          case 1:
            lcd.print("Z");
            TempPos = 10;
            break;
          case 2:
            lcd.print("Y");
            TempPos = 5;
            break;
          case 3:
            lcd.print("X");
            TempPos = 1;
            break;
        }
        long TempEEProm=0;
        EEPROM.get(TempPos,TempEEProm);
        lcd.print(TempEEProm);
        lcd.print(" pul/cm en ");

        inputString = "";
        delay(3000);
      }
    }
  }
}
```

11.1.2. Microcontrolador controlador motor eje X

```
/*
 *  En auto, Se mantiene en movimiento dependiendo de DY1 XEn (Pin6) y la
 *  direccion depende de DY1 DirX(Pin13)
 *  La velocidad depende de la info enviada por el master por puerto serial
 */


//INPUTS
const int In = 2;           //Interrupcion para leer pulsos, conectado a pin 13 (Ou)
const int Stp = 3;          //Interrupcion de paro de emergencia a boton
const int CoWo = 6;         //Bit de Configuracion o Trabajo HIGH trabajo, LOW Jog
const int EnX = 5;          //Enable X que viene de DY1 Pin6
const int DirX = 8;         //Direccion de X que viene de DY1 Pin13
const int LsP = A0;         //Limit switch Positivo (derecha)
const int LsN = A1;         //Limit Switch Negativo (Izquierda)


//OUTPUTS
const int Ou = 12;          //Salida de pulsos a Step de driver HIGH + LOW -
const int Dir = 11;         //Salida de direccion del motor X a driver (HIGH a positivo, LOW a negativo)
const int NDir = 9;         //Salida de direccion del motor X a driver (HIGH a negativo, LOW a positivo)
const int Enab = 10;         //Enable de driver
```

```
const int Valores = 4;      //Indicador de valor de velocidad ingresado


//Variables
String Tex = "";            //Texto recibido en puerto serial
unsigned long C = 0;         //Contador de pulsos
unsigned int Frec = 200;    //Frecuencia
unsigned int Pulsos = 0;    //Pulsos a moverse en soldadura

String Temp = "";          //Temporal para aislar numero de entrada de puerto serial
boolean Start = false;      //Variable para inicio de movimiento
boolean FDir = false;       //Flag de direccion de movimiento de X
boolean Vals = false;       //Flag para ver si estan los 3 valores iniciales
int vSel = 0;               //Var de selector de motor
int vLsP = 0;               //Var de Limit Switch + (derecho)
int vLsN = 0;               //Var de Limit Switch - (Izquierdo)
int vDir = 0;               //Var de direccion de Y
int JOpc = 0;    //Opciones de Jog 1) Mov hasta Limit Switch, 2) Mov exacto de cant
de pulsos, 3) Mov dependiente de joystick
int JDir = 0;              //Direccion de Jog HIGH a + (derecha) LOW a - (izquierda)
unsigned int JVel = 0;      //Velocidad o frecuencia de jog
unsigned long JPul = 0;      //Candidad de pulsos a mover en jog
int JS = 0;                 //Iniciar movimiento de JOG
int iSt = 0;                //Instruccion para opcion 3 de serial (Entero)
String sSt = "";            //Instruccion para opcion 3 de serial (String)



void setup() {
  Serial.begin(9600);
  pinMode(In,INPUT);
  pinMode(Stp,INPUT);
  pinMode(EnX,INPUT_PULLUP);
  pinMode(CoWo,INPUT_PULLUP);
  pinMode(DirX,INPUT);
  pinMode(Enab,OUTPUT);
  pinMode(Ou,OUTPUT);
  pinMode(Dir,OUTPUT);
  pinMode(NDir,OUTPUT);
  pinMode(Valores,OUTPUT);


  attachInterrupt(digitalPinToInterrupt(In), CuentaP, RISING);
  attachInterrupt(digitalPinToInterrupt(Stp), Emergencia, RISING);


}

void loop() {
  if (digitalRead(CoWo) == HIGH) {            //Pin de Jog/Run en 0 para auto
    while (Start == true) {
      if (Frec != 0) {
        digitalWrite(Valores,LOW);       //EL valor de velocidad ya a sido ingresado
        while (EnX == HIGH) {
          digitalWrite(Enab,digitalRead(EnX));
          digitalWrite(Dir,digitalRead(DirX));
          if (digitalRead(DirX) == HIGH) {      //Niega la direccion para X'
            digitalWrite(NDir,LOW);
          }else {
            digitalWrite(NDir,HIGH);
          }
          if (serialEventRun) serialEventRun(); //Revisa puerto serial por cambios en frec
          LimSwi();                               //Revisa limit_Switch
          tone(Ou,Frec);
        }
        noTone(Ou);
      }
    }
```

```
noTone(Ou);
}else {
  //Programacion de JOG  (NOTA: Se deben recibir todos los datos de Direccion, velocidad, pulsos antes
  de la opcion
  //Exepto en opcion 3 que es informacion cte, se debe enviar la velocidad antes de la opcion
  Start = true;

  switch (JOpc) {
    case 1:                            //Opcion de ir a Limit Switch
      //Serial.println("Op 1 hasta LS");
      if (JDir == 1) {               //Pone la direccion
        digitalWrite(Dir,HIGH);
        LimSwi();
        while (vLsP == LOW) {      //Inicia movimiento hasta llegar a limit switch
          LimSwi();
          tone(Ou,JVel);
          if (Start == false) break;      //Paro de emergencia
        }
      } else {
        digitalWrite(Dir,LOW);
        LimSwi();
        while (vLsN == LOW) {      //Inicia movimiento hasta llegar a limit switch
          LimSwi();
          tone(Ou,JVel);
          if (Start == false) break;      //Paro de emergencia
        }
      }
      noTone(Ou);                     //Detiene pulsos
      JOpc = 0;
      break;
    case 2:                           //Opcion de Movimiento exacto
      //Serial.println("Op 2 Exacto");
      if (JDir == 1) {               //Pone la direccion
        digitalWrite(Dir,HIGH);
      } else {
        digitalWrite(Dir,LOW);
      }
      LimSwi();
      while (C < JPul) {            //inicia movimiento con contador de pulsos
        LimSwi();
        //Serial.print("Cont: ");
        //Serial.println(C);
        tone(Ou,JVel);
        if (Start == false) break;      //Paro de emergencia
      }
      noTone(Ou);
      JOpc = 0;
      C = 0;
      //Serial.print("JOpc: ");
      //Serial.println(JOpc);
      break;
    case 3:                     //Opcion manual, enviar velocidad  antes de opcion
      do {
        LimSwi();
        if (Serial.available()>0) {

          char St = (char)Serial.read();
          sSt += St;
          if (St == '\n') {
            sSt.trim();
            iSt = sSt.toInt();
            switch (iSt) {
              case 0:                        //Derecha
                digitalWrite(Dir,HIGH);
                LimSwi();
                tone(Ou,JVel);
                sSt = "";
                JOpc = 0;
```

```
                break;
              case 1:                       //Para
                noTone(Ou);
                sSt = "";
                JOpc = 0;
                break;
              case 2:                       //Izquierda
                digitalWrite(Dir,LOW);
                LimSwi();
                tone(Ou,JVel);
                sSt = "";
                JOpc = 0;
                break;
          }
        }
      }

  } while (JOpc == 3);
 }
}
}

void serialEvent() {
  if (Serial.available()>0) {
    char inChar = (char)Serial.read();
    Tex += inChar;
    if (inChar == '\n') {
      Tex.trim();
      //Serial.println(Tex);
      if (Tex == "s") {          //Start
        Start = true;
      }
      if (Tex == "r") {          //Reset contador
        C = 0;
        Pulsos = 0;
        Frec = 200;
      }
      if (Tex.startsWith("Xf")) {       //Velocidad de X
        Temp = Tex.substring(2);
        Frec = Temp.toInt();
        //Serial.println(Frec);
      }
      if (Tex.startsWith("Xp")) {       //Cantidad de pulsos a moverse en JOG
        Temp = Tex.substring(2);
        Pulsos = Temp.toInt();
      }
      if (Tex.startsWith("XJO")) {       //Opcion de Jog
        Temp = Tex.substring(3);
        JOpc = Temp.toInt();
        //Serial.print("JO: ");
        //Serial.println(JOpc);
      }
      if (Tex.startsWith("XJD")) {       //Direccion de Jog
        Temp = Tex.substring(3);
        JDir = Temp.toInt();
        //Serial.print("JD: ");
        //Serial.println(JDir);
      }
      if (Tex.startsWith("XJV")) {       //Velocidad de Jog
        Temp = Tex.substring(3);
        //Serial.println("JV: " + Temp);
        JVel = Temp.toInt();
      }
      if (Tex.startsWith("XJP")) {       //Pulsos de Jog
        Temp = Tex.substring(3);
        //Serial.println("JP: " + Temp);
        //JPul = Temp.toInt();
        Temp.trim();
```

```
        JPul = Temp.toInt();
      }
      if (Tex.startsWith("XJS")) {       //Iniciar movimiento en jog manual
        Temp = Tex.substring(3);
        //Serial.println("JS: " + Temp);
        JS = Temp.toInt();
        Start = true;
      }
      Tex="";
    }
  }
}

void CuentaP() {
    C++;
}

void Emergencia() {
  Start = false;
  noTone(Ou);
  JOpc = 0;
}

void LimSwi () {
  vDir = digitalRead(Dir);                                    //Lee direccion de motor
  if (analogRead(LsP) > 500) {vLsP = HIGH;} else {vLsP = LOW;} //Lee sensor de tope+ y lo niega
  if (analogRead(LsN) > 500) {vLsN = HIGH;} else {vLsN = LOW;} //Lee sensor de tope- y lo niega
  //Procedimiento para enable
  if(((digitalRead(DirX)==HIGH)&&(vLsP==LOW))||((digitalRead(DirX)==LOW)&&(vLsN==LOW))){
    digitalWrite(Enab,LOW);
  } else {
    digitalWrite(Enab,HIGH);
  }
}
```

11.1.3. Microcontrolador controlador motor ejes Y

```
//INPUTS
const int In = 2;           //Interrupcion para leer pulsos, conectado a pin 13 (Ou)
const int Stp = 3;          //Interrupcion de paro de emergencia a boton
const int S1 = 7;          //Entrada de sensor de posicion del eje X+
const int S2 = 8;          //Entrada de sensor de posicion del eje X-
const int S3 = 9;           //Entrada de sensor de posicion del eje Y+
const int XEn = 6;         //Bit de Configuracion o Trabajo HIGH trabajo, LOW Jog
//const int CoWo = 6;         //Bit de Configuracion o Trabajo HIGH trabajo, LOW Jog
const int Sel = 5;          //Selector motor a usar
const int LsP = 14;         //Limit switch Positivo (derecha)
const int LsN = 15;         //Limit Switch Negativo (Izquierda)
const int RJ = A7;          //Entrada de Run/Jog

//OUTPUTS
const int Ou = 12;          //Salida de pulsos a Step de driver HIGH + LOW -
const int Dir = 11;         //Salida de direccion del motor Y a driver (HIGH derecha, LOW izquierda
const int Enab = 10;         //Enable de driver
const int Valores = 4;      //Indicador de valores de configuracion
const int DirX = 13;         //Direccion del motor X 0 en dir a S2, 1 en dir a S1

String Tex = "";            //Texto recibido en puerto serial
boolean bRJ = false;         //Booleana de JOG false o RUN true
unsigned int C = 0;         //Contador de pulsos
unsigned int Frec = 200;    //Frecuencia
unsigned int Pulsos = 0;    //Pulsos a moverse en soldadura
unsigned int Avance = 0;    //Pulsos a moverse en cambio de linea
String Temp = "";           //Temporal para aislar numero de entrada de puerto serial
```

```
boolean Start = false;      //Variable para inicio de movimiento
boolean FDir = false;       //Flag de direccion de movimiento de X
boolean Vals = false;       //Flag para ver si estan los 3 valores iniciales
int vSel = 0;               //Var de selector de motor
int vLsP = 0;               //Var de Limit Switch + (derecho)
int vLsN = 0;               //Var de Limit Switch - (Izquierdo)
int vDir = 0;               //Var de direccion de Y
int JOpc = 0;               //Opciones de Jog 1) Mov hasta Limit Switch, 2) Mov exacto de cant de pulsos,
3) Mov dependiente de joystick
int JDir = 0;               //Direccion de Jog HIGH a + (derecha) LOW a - (izquierda)
unsigned int JVel = 0;      //Velocidad o frecuencia de jog
unsigned int JPul = 0;      //Candidad de pulsos a mover en jog
int JS = 0;                  //Iniciar movimiento de JOG
int iSt = 0;                //Instruccion para opcion 3 de serial (Entero)
String sSt = "";            //Instruccion para opcion 3 de serial (String)
unsigned int MY=0;          //Selector de sensores a utilizar
void setup() {
  Serial.begin(9600);
  pinMode(In,INPUT);
  pinMode(S1,INPUT);
  pinMode(S2,INPUT);
  pinMode(S3,INPUT);
  pinMode(Sel,INPUT);
  pinMode(Sel,INPUT);
  pinMode(LsP,INPUT_PULLUP);
  pinMode(LsN,INPUT_PULLUP);
  pinMode(Valores,OUTPUT);
  pinMode(DirX,OUTPUT);
  pinMode(Enab,OUTPUT);
  pinMode(Ou,OUTPUT);
  pinMode(Dir,OUTPUT);
  pinMode(XEn,OUTPUT);
  digitalWrite(XEn,LOW);


  attachInterrupt(digitalPinToInterrupt(In), CuentaP, RISING);
  attachInterrupt(digitalPinToInterrupt(Stp), Emergencia, RISING);

  //digitalWrite(Valores,HIGH);
}

void loop() {
  if (analogRead(RJ) > 512){              //Revisa entrada de RUN/JOG
      bRJ = true;
  }else{
    bRJ = false;
  }
  if (bRJ == HIGH) {     //Revisa si JOG (false) o RUN (true)
    JOpc = 0;                          //Anula movimientos de JOG
    while ((Pulsos==0)||(Frec==0)||(Avance==0)){//Verifica si ya estan metidos los datos de config
      digitalWrite(Valores,HIGH);
      if (serialEventRun) serialEventRun();           //Revisa puerto serial
    }
    digitalWrite(Valores,LOW);          //Ya fueron ingresados los valores
    if ((digitalRead(S1)==HIGH) && (digitalRead(S2)==LOW) && (digitalRead(S3)==LOW)){
    //Revisa que este en posicion de inicio
      //Serial.println(FDir);
      if (Start == true) {                //Espera comando de inicio de movimiento
        //digitalWrite(Enab,HIGH);        //activa el uso de driver
        if (FDir == false) {
          while(digitalRead(S3)==LOW) {//Movimiento mientras sensor Y+(limite al que debe llegar) no activo
            if (Start == false) break;      //Paro de emergencia
            digitalWrite(XEn,HIGH);       //Activa que se mueva motor de X
            digitalWrite(DirX,LOW);      //Se debe ir en direccion a S2 el motor X
            while ((digitalRead(S2) == LOW) && (digitalRead(S3) == LOW)) {
            //Movimiento mientras llega a sensor X+
              if (Start == false) break;      //Paro de emergencia
              digitalWrite(Dir,HIGH);
```

```
    LimSwi();
    while (C < Pulsos) { //inicia movimiento de pulsacion hacia derecha
      LimSwi();
      tone(Ou,Frec);
      if (Start == false) break;      //Paro de emergencia
      if (serialEventRun) serialEventRun(); //Revisa puerto serial
    }
    noTone(Ou);
    C = 0;
    digitalWrite(Dir,LOW);
    LimSwi();
    while (C < Pulsos) { //Inicia movimiento de pulsacion hacia izquierda
      LimSwi();
      tone(Ou,Frec);
      if (Start == false) break;      //Paro de emergencia
      if (serialEventRun) serialEventRun();     //Revisa puerto serial
    }
    noTone(Ou);
    C = 0;
  }
  digitalWrite(XEn,LOW);//Detiene motor X para mover Y recto a la derecha
  digitalWrite(Dir,HIGH);
  LimSwi();
  while (C < Avance) {//Movimiento de avance si llega al limite del X+ o X-
    LimSwi();
    tone(Ou,Frec);
    if (Start == false) break;      //Paro de emergencia
    if (serialEventRun) serialEventRun();     //Revisa puerto serial
  }          //Termino el avance,
  C = 0;
  noTone(Ou);
  digitalWrite(DirX,HIGH);      //Se debe ir en direccion a S1 el motor X
  while ((digitalRead(S1) == LOW) && (digitalRead(S3) == LOW)) {                          //Movimiento mientr
    if (Start == false) break;      //Paro de emergencia
    digitalWrite(XEn, HIGH);        //inicia movimiento de motor X a S1
    digitalWrite(Dir,HIGH);
    LimSwi();
    while (C < Pulsos) {
      LimSwi();
      tone(Ou,Frec);
      if (Start == false) break;      //Paro de emergencia
      if (serialEventRun) serialEventRun();  //Revisa puerto serial
    }
    noTone(Ou);
    C = 0;
    digitalWrite(Dir,LOW);
    LimSwi();
    while (C < Pulsos) {
      LimSwi();
      tone(Ou,Frec);
      if (Start == false) break;      //Paro de emergencia
      if (serialEventRun) serialEventRun();   //Revisa puerto serial
    }
    noTone(Ou);
    C = 0;
  }  //Llego al sensor X-
  while (C < Avance) {                  //Movimiento de avance
    LimSwi();
    digitalWrite(XEn,LOW);  //Detiene motor X para mover Y a la derecha
    tone(Ou,Frec);
    if (Start == false) break;      //Paro de emergencia
    if (serialEventRun) serialEventRun();      //Revisa puerto serial
  }          //Termino el avance,
  noTone(Ou);
  C = 0;
  digitalWrite(XEn,HIGH);       //Inicia motor X para llegar a S2
  digitalWrite(DirX,LOW);      //Se debe ir en direccion a S2 el motor X
}
```

```
          Start = false;
          noTone(Ou);
          digitalWrite(Enab,LOW);       //Desactiva uso de driver
        }
      }
    }
  }else {
//Programacion de JOG  (NOTA: Se deben recivir todos los datos de Direccion, velocidad, pulsos antes de la
opcion
    //Exepto en opcion 3 que es informacion cte, se debe enviar la velocidad antes de la opcion
    Start = true;

    switch (JOpc) {
      case 1:                           //Opcion de ir a Limit Switch
        //Serial.println("Op 1 hasta LS");
        if (JDir == 1) {               //Pone la direccion
          digitalWrite(Dir,HIGH);
          LimSwi();
          while (vLsP == LOW) {    //Inicia movimiento hasta llegar a limit switch
            LimSwi();
            tone(Ou,JVel);
            if (Start == false) break;      //Paro de emergencia
          }
        } else {
          digitalWrite(Dir,LOW);
          LimSwi();
          while (vLsN == LOW) {     //Inicia movimiento hasta llegar a limit switch
            LimSwi();
            tone(Ou,JVel);
            if (Start == false) break;      //Paro de emergencia
          }
        }
        noTone(Ou);                     //Detiene pulsos
        JOpc = 0;
        break;
      case 2:                               //Opcion de Movimiento exacto
        //Serial.println("Op 2 Exacto");
        if (JDir == 1) {                    //Pone la direccion
          digitalWrite(Dir,HIGH);
        } else {
          digitalWrite(Dir,LOW);
        }
        LimSwi();
        while (C < JPul) {            //inicia movimiento con contador de pulsos
          LimSwi();
          //Serial.print("Cont: ");
          //Serial.println(C);
          tone(Ou,JVel);
          if (Start == false) break;      //Paro de emergencia
        }
        noTone(Ou);
        JOpc = 0;
        C = 0;
        //Serial.print("JOpc: ");
        //Serial.println(JOpc);
        break;
      case 3:               //Opcion manual, enviar velocidad  antes de opcion
        do {

          LimSwi();
          if (Serial.available()>0) {

            char St = (char)Serial.read();
            sSt += St;
            if (St == '\n') {
              sSt.trim();
              iSt = sSt.toInt();
              switch (iSt) {
```

```
                case 0:                         //Derecha

                  digitalWrite(Dir,HIGH);
                  LimSwi();
                  tone(Ou,JVel);
                  sSt = "";
                  JOpc = 0;
                  break;
                case 1:                         //Para

                  noTone(Ou);
                  sSt = "";
                  JOpc = 0;
                  break;
                case 2:                         //Izquierda

                  digitalWrite(Dir,LOW);
                  LimSwi();
                  tone(Ou,JVel);
                  sSt = "";
                  JOpc = 0;
                  break;
            }
          }
        }

    } while (JOpc == 3);
  }
}
}

void serialEvent() {
  if (Serial.available()>0) {
    char inChar = (char)Serial.read();
    Tex += inChar;
    if (inChar == '\n') {
      Tex.trim();
      //Serial.println(Tex);
      if (Tex.startsWith("a")) {
        Temp = Tex.substring(1);
        Avance = Temp.toInt();
      }
      if (Tex == "s") {
        Start = true;
      }
      if (Tex == "r") {
        C = 0;
        Avance = 0;
        Pulsos = 0;
        Frec = 200;
      }
      if (Tex.startsWith("f")) {
        Temp = Tex.substring(1);
        Frec = Temp.toInt();
        //Serial.println(Frec);
      }
      if (Tex.startsWith("p")) {
        Temp = Tex.substring(1);
        Pulsos = Temp.toInt();
      }
      if (Tex.startsWith("JO")) {       //Opcion de Jog
        Temp = Tex.substring(2);
        JOpc = Temp.toInt();

        //delay(1000);
        //Serial.print("JO: ");
        //Serial.println(JOpc);
      }
```

```
      if (Tex.startsWith("JD")) {       //Direccion de Jog
        Temp = Tex.substring(2);
        JDir = Temp.toInt();
        //Serial.print("JD: ");
        //Serial.println(JDir);
      }
      if (Tex.startsWith("JV")) {       //Velocidad de Jog
        Temp = Tex.substring(2);
        //Serial.println("JV: " + Temp);
        JVel = Temp.toInt();
      }
      if (Tex.startsWith("JP")) {       //Pulsos de Jog
        Temp = Tex.substring(2);
        //Serial.println("JP: " + Temp);
        JPul = Temp.toInt();
        C=0;
      }
      if (Tex.startsWith("JS")) {       //Iniciar movimiento en jog manual
        Temp = Tex.substring(2);
        //Serial.println("JS: " + Temp);
        JS = Temp.toInt();
        Start = true;
      }
     /* if(Tex.startsWith("MY")){
        Temp=Tex.substring(2);
        MY = Temp.toInt();
      }*/
      Tex="";
    }
  }
}

void CuentaP() {
    C++;
}

void Emergencia() {
  Start = false;
  noTone(Ou);
  JOpc = 0;
}

void LimSwi () {
  /*vSel = digitalRead(Sel);                        //Lee si motor seleccionado
  vDir = digitalRead(Dir);                           //Lee direccion de motor
  if (analogRead(LsP) > 500) {vLsP = HIGH;} else {vLsP = LOW;} //Lee sensor de tope+ y lo niega
  if (analogRead(LsN) > 500) {vLsN = HIGH;} else {vLsN = LOW;}      //Lee sensor de tope- y lo niega
  //Procedimiento para enable
  if(((vSel==HIGH)&&(vDir==HIGH)&&(vLsP==LOW))||((vSel==HIGH)&&(vDir==LOW)&&(vLsN==LOW))){
    digitalWrite(Enab,LOW);
  } else {
    digitalWrite(Enab,HIGH);
  }*/
  if (digitalRead(Sel)== HIGH){digitalWrite(Enab,LOW);} else {digitalWrite(Enab,HIGH);}

  if((digitalRead(Sel)==HIGH)&&(digitalRead(LsN)==HIGH)&&(digitalRead(Dir)==HIGH)){
    digitalWrite(Enab,LOW);
    }
  if((digitalRead(Sel)==HIGH)&&(digitalRead(LsP)==HIGH)&&(digitalRead(Dir)==LOW)){
    digitalWrite(Enab,LOW);
    }
  if((digitalRead(Sel)==HIGH)&&(digitalRead(LsN)==LOW)&&(digitalRead(LsP)==LOW)){
    digitalWrite(Enab,LOW);
    }
}
```

11.1.4. Microcontrolador controlador eje Z

```
/*
 *  En auto, mantiene el amperage seteado por boton set
 *  Poner salidas de pantallas LCD con el amperage detectado por CT
 */

/*#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
//#define I2C_ADDR    0x27

//LiquidCrystal_I2C             lcd(I2C_ADDR,HIGH, 1, 0, 4, 5, 6, 7);
*/

#include <stdio.h>
#include <stdlib.h>  // Para convertir string a long
#include  <Wire.h>   //I2C
#include  <LiquidCrystal_I2C.h>   //Liquid cristal

//Crear el objeto lcd  dirección  0x27 y 16 columnas x 2 filas
LiquidCrystal_I2C lcd(0x27,16,2);  //

//INPUTS
const int In = 2;           //Interrupcion para leer pulsos, conectado a pin 13 (Ou)
const int Stp = 3;          //Interrupcion de paro de emergencia a boton
const int SetA = 9;         //Setea ampraje ideal
const int CoWo = 6;         //Bit de Configuracion o Trabajo HIGH trabajo, LOW Jog
const int Sel = 5;          //Selector motor a usar
const int LsP = A0;         //Limit switch Positivo (derecha)
const int LsN = A1;         //Limit Switch Negativo (Izquierda)
const int Ct = A2;          //Lectura del CT
const int VelZ = A3;        //Velocidad de Z para correcciones
const int Tol = A6;         //Tolerancia de Amps para correccion

//OUTPUTS
const int LUp = 7;          //Indicador qeu el motor debe subir
const int LDn = 8;          //Indicador que el motor debe bajar
const int Ou = 12;          //Salida de pulsos a Step de driver HIGH + LOW -
const int Dir = 11; //Salida de direccion del motor Y a driver (HIGH derecha, LOW izquierda
const int Enab = 10;         //Enable de driver
const int Valores = 4;    //Indicador de valores de configuracion de Amp de antorcha

//Variables
String Tex = "";            //Texto recibido en puerto serial
unsigned long C = 0;         //Contador de pulsos
unsigned int Frec = 200;    //Frecuencia
unsigned int Pulsos = 0;    //Pulsos a moverse en soldadura
unsigned int Avance = 0;    //Pulsos a moverse en cambio de linea
String Temp = "";          //Temporal para aislar numero de entrada de puerto serial
boolean Start = false;      //Variable para inicio de movimiento
boolean FDir = false;       //Flag de direccion de movimiento de X
boolean Vals = false;       //Flag para ver si estan los 3 valores iniciales
int vSel = 0;               //Var de selector de motor
int vLsP = 0;               //Var de Limit Switch + (derecho)
int vLsN = 0;               //Var de Limit Switch - (Izquierdo)
int vDir = 0;               //Var de direccion de Y
int JOpc = 0; //Opciones de Jog 1)Mov hasta Limit Switch,2)Mov exacto de cant de pulsos, 3) Mov dependiente
de joystick
int JDir = 0;              //Direccion de Jog HIGH a + (derecha) LOW a - (izquierda)
unsigned int JVel = 0;      //Velocidad o frecuencia de jog
unsigned long JPul = 0;      //Candidad de pulsos a mover en jog
int JS = 0;                 //Iniciar movimiento de JOG
int iSt = 0;                //Instruccion para opcion 3 de serial (Entero)
String sSt = "";            //Instruccion para opcion 3 de serial (String)
int DUp = 0.00;             //Parametro para iniciar subir motor
```

```
int DDn = 0.00;             //Parametro para iniciar bajar motor
int Ideal = 0;              //Ideal a mantener en Volt
int Vel = 0;                //Variable de velocidad
int ValCt = 0;              //Lectura de volt de antorcha
int aTol = 0;               //Tolerancia en Amps
int mTol = 0;               //Mapeo de Tolerancia
int aLec = 0;               //Lectura de Ant1 en Amps

//Caracter extra
byte MaMe[8] = {
B01000,
B11100,
B01001,
B00010,
B00100,
B01000,
B10111,
B00000,
};

void setup() {
  Serial.begin(9600);
  pinMode(In,INPUT);
  pinMode(Stp,INPUT);
  pinMode(SetA,INPUT_PULLUP);
  pinMode(CoWo,INPUT_PULLUP);
  pinMode(Sel,INPUT_PULLUP);
  pinMode(LUp,OUTPUT);
  pinMode(LDn,OUTPUT);
  pinMode(Valores,OUTPUT);
  pinMode(Enab,OUTPUT);
  pinMode(Ou,OUTPUT);
  pinMode(Dir,OUTPUT);


  attachInterrupt(digitalPinToInterrupt(In), CuentaP, RISING);
  attachInterrupt(digitalPinToInterrupt(Stp), Emergencia, RISING);

  //lcd.begin (16,2);    // Inicializar el display con 16 caraceres 2 lineas
  //lcd.setBacklightPin(3,POSITIVE);
  lcd.init();
  lcd.backlight();
  lcd.createChar (0,MaMe);
  lcd.setCursor(0, 0);
  lcd.print("Meta:");
  lcd.setCursor(9,0);
  lcd.print("Tol");
  lcd.write (byte (0));
  lcd.print(":");
  lcd.setCursor(0,1);
  lcd.print("Lectura:");
}

void loop() {
  if (digitalRead(CoWo) == HIGH) {            //Pin de Jog/Run en 0 para auto
    ValCt = analogRead(Ct);                   //Lee sensor de amperaje
    aLec = map(ValCt, 0, 1023, 0, 600);       //Mapeo de lectura de amps
    Vel = analogRead(VelZ);                   //Lee valor de pot de velocidad Z
    Vel = map(Vel, 0, 1023, 200, 2500);       //Mapeo de velocidad
    mTol = analogRead(Tol);                   //Lectura de pot de tolerancia
    aTol = map(mTol, 0, 1023, 5, 41);        //mapeo de lectura de tolerancia(25amp)
    DUp = Ideal-aTol; //Parametro cuando la antorcha se acerca y hay que subir motor
    DDn = Ideal+aTol; //Parametro cuando la antorcha se aleja y hay que bajar motor
    Setea();                                  //Llama procedimiento de parametros


    if (Ideal < 100) {
      lcd.setCursor(5, 0);
```

```
    lcd.print("   ");
  }
  lcd.setCursor(5, 0);
  lcd.print(Ideal);
  if (aTol < 10) {
    lcd.setCursor(14, 0);
    lcd.print("  ");
  }
  lcd.setCursor(14, 0);
  lcd.print(aTol);
  if (aLec < 100) {
    lcd.setCursor(8, 1);
    lcd.print("   ");
  }
  lcd.setCursor(8, 1);
  lcd.print(aLec);
  delay(100);

  if (Ideal == 0) {
    digitalWrite(Valores,HIGH);       //Valor no seteado
  }else {
    digitalWrite(Valores,LOW);       //Valor ya seteado
  }
 if (Start == true) {
  if (Ideal != 0) {                //Revisa si se marco el parametro de Volt ideal
    if (aLec > DUp) {                  //Compara si antorcha1 se acerca
      digitalWrite(Dir,HIGH);
      LimSwi();
      tone(Ou,Vel);
      digitalWrite(LUp,LOW);            //Led indicador que la antorcha debe subir
    }else {
      noTone(Ou);                         //Detiene el movimiento
      digitalWrite(LUp,HIGH);  //Indicador que antorcha esta entre los
      parametros aceptables
    }
    if (aLec < DDn) {                   //Compara si antorcha1 de aleja
      digitalWrite(Dir,LOW);
      LimSwi();
      tone(Ou,Vel);
      digitalWrite(LDn,LOW);            //Led indicador que la antorcha debe bajar
    }else {
      noTone(Ou);                         //Detiene el movimiento
      digitalWrite(LDn,HIGH);             //Indicador que antorcha esta entre los
      parametros aceptables
    }
  }
 }
}else {
  //Programacion de JOG  (NOTA: Se deben recibir todos los datos de Direccion,
  velocidad, pulsos antes de la opcion
  //Exepto en opcion 3 que es informacion cte, se debe enviar la velocidad antes
  de la opcion
  Start = true;

  switch (JOpc) {
    case 1:                           //Opcion de ir a Limit Switch
      //Serial.println("Op 1 hasta LS");
      if (JDir == 1) {              //Pone la direccion
        digitalWrite(Dir,HIGH);
        LimSwi();
        while (vLsP == LOW) {      //Inicia movimiento hasta llegar a limit switch
          LimSwi();
          tone(Ou,JVel);
          if (Start == false) break;    //Paro de emergencia
        }
      } else {
        digitalWrite(Dir,LOW);
        LimSwi();
```

```
          while (vLsN == LOW) {      //Inicia movimiento hasta llegar a limit switch
            LimSwi();
            tone(Ou,JVel);
            if (Start == false) break;      //Paro de emergencia
          }
        }
        noTone(Ou);                     //Detiene pulsos
        JOpc = 0;
        break;
      case 2:                          //Opcion de Movimiento exacto
        //Serial.println("Op 2 Exacto");
        if (JDir == 1) {                //Pone la direccion
          digitalWrite(Dir,HIGH);
        } else {
          digitalWrite(Dir,LOW);
        }
        LimSwi();
        while (C < JPul) {             //inicia movimiento con contador de pulsos
          LimSwi();
          //Serial.print("Cont: ");
          //Serial.println(C);
          tone(Ou,JVel);
          if (Start == false) break;      //Paro de emergencia
        }
        noTone(Ou);
        JOpc = 0;
        C = 0;
        //Serial.print("JOpc: ");
        //Serial.println(JOpc);
        break;
      case 3:                     //Opcion manual, enviar velocidad  antes de opcion
        do {
          LimSwi();
          if (Serial.available()>0) {

            char St = (char)Serial.read();
            sSt += St;
            if (St == '\n') {
              sSt.trim();
              iSt = sSt.toInt();
              switch (iSt) {
                case 0:                       //Derecha
                  digitalWrite(Dir,HIGH);
                  LimSwi();
                  tone(Ou,JVel);
                  sSt = "";
                  JOpc = 0;
                  break;
                case 1:                       //Para
                  noTone(Ou);
                  sSt = "";
                  JOpc = 0;
                  break;
                case 2:                       //Izquierda
                  digitalWrite(Dir,LOW);
                  LimSwi();
                  tone(Ou,JVel);
                  sSt = "";
                  JOpc = 0;
                  break;
              }
            }
          }

    } while (JOpc == 3);
  }
}
}
```

```
void serialEvent() {
  if (Serial.available()>0) {
    char inChar = (char)Serial.read();
    Tex += inChar;
    if (inChar == '\n') {
      Tex.trim();
      //Serial.println(Tex);
      if (Tex.startsWith("a")) {
        Temp = Tex.substring(1);
        Avance = Temp.toInt();
      }
      if (Tex == "Zs") {
        Start = true;
      }
      if (Tex == "Zr") {
        C = 0;
        Avance = 0;
        Pulsos = 0;
        Frec = 200;
      }
      if (Tex.startsWith("Zf")) {
        Temp = Tex.substring(2);
        Frec = Temp.toInt();
        //Serial.println(Frec);
      }
      if (Tex.startsWith("Zp")) {
        Temp = Tex.substring(2);
        Pulsos = Temp.toInt();
      }
      if (Tex.startsWith("ZJO")) {       //Opcion de Jog
        Temp = Tex.substring(3);
        JOpc = Temp.toInt();
        //Serial.print("JO: ");
        //Serial.println(JOpc);
      }
      if (Tex.startsWith("ZJD")) {       //Direccion de Jog
        Temp = Tex.substring(3);
        JDir = Temp.toInt();
        //Serial.print("JD: ");
        //Serial.println(JDir);
      }
      if (Tex.startsWith("ZJV")) {       //Velocidad de Jog
        Temp = Tex.substring(3);
        //Serial.println("JV: " + Temp);
        JVel = Temp.toInt();
      }
      if (Tex.startsWith("ZJP")) {       //Pulsos de Jog
        Temp = Tex.substring(3);
        //Serial.println("JP: " + Temp);
        //JPul = Temp.toInt();
        Temp.trim();
        JPul = atol(Temp);
      }
      if (Tex.startsWith("ZJS")) {       //Iniciar movimiento en jog manual
        Temp = Tex.substring(3);
        //Serial.println("JS: " + Temp);
        JS = Temp.toInt();
        Start = true;
      }
      Tex="";
    }
  }
}

void CuentaP() {
    C++;
}
```

```
void Emergencia() {
  Start = false;
  noTone(Ou);
  JOpc = 0;
}

void LimSwi () {
  vSel = digitalRead(Sel);                          //Lee si motor seleccionado
  vDir = digitalRead(Dir);                          //Lee direccion de motor
  if (analogRead(LsP) > 500) {vLsP = HIGH;} else {vLsP = LOW;}  //Lee sensor de   tope+ y lo niega
  if (analogRead(LsN) > 500) {vLsN = HIGH;} else {vLsN = LOW;}  //Lee sensor de   tope- y lo niega
  //Procedimiento para enable
  if (((vSel==HIGH)&&(vDir==HIGH)&&(vLsP==LOW))||((vSel==HIGH)&&(vDir==LOW)&&(vLsN==LOW))){
    digitalWrite(Enab,LOW);
  } else {
    digitalWrite(Enab,HIGH);
  }
}

void Setea() {                        //Procedimiento para setear parametros
  if (digitalRead(SetA) == LOW) {     //Si presiono boton de SET
    Ideal = aLec;    //Graba en Ideal el volt ideal a mantener
  }
}
```

11.1.5. Microcontrolador controlador de antorchas

```
/*
 *  En auto, Se mantiene en movimiento dependiendo de LS+ que controla la
 *  existencia de alambre XEn (Pin6) y la
 *  La velocidad depende de la info enviada por el master por puerto serial
 */


//INPUTS
const int In = 2;           //Interrupcion para leer pulsos, conectado a pin 13 (Ou)
const int Stp = 3;          //Interrupcion de paro de emergencia a boton
const int CoWo = 6;         //Bit de Configuracion o Trabajo HIGH trabajo, LOW Jog
const int LsP = A0;         //Limit switch Positivo (derecha)

//OUTPUTS
const int Ou = 12;          //Salida de pulsos a Step de driver HIGH + LOW -
const int Dir = 11;         //Salida de direccion del motor X a driver (HIGH a
positivo, LOW a negativo
const int Enab = 10;         //Enable de driver
const int Valores = 4;      //Indicador de valor de velocidad ingresado


//Variables
String Tex = "";            //Texto recibido en puerto serial
unsigned long C = 0;         //Contador de pulsos
unsigned int Frec = 200;    //Frecuencia
unsigned int Pulsos = 0;    //Pulsos a moverse en soldadura

String Temp = "";          //Temporal para aislar numero de entrada de puerto serial
boolean Start = false;      //Variable para inicio de movimiento
boolean FDir = false;       //Flag de direccion de movimiento de X
boolean Vals = false;       //Flag para ver si estan los 3 valores iniciales
int vSel = 0;               //Var de selector de motor
int vLsP = 0;               //Var de Limit Switch + (derecho)
int vLsN = 0;               //Var de Limit Switch - (Izquierdo)
int vDir = 0;               //Var de direccion de Y
int JOpc = 0;               //Opciones de Jog 1) Mov hasta Limit Switch, 2) Mov
   exacto de cant de pulsos, 3) Mov dependiente de joystick
int JDir = 0;              //Direccion de Jog HIGH a + (derecha) LOW a - (izquierda)
```

```
unsigned int JVel = 0;      //Velocidad o frecuencia de jog
unsigned long JPul = 0;      //Candidad de pulsos a mover en jog
int JS = 0;                 //Iniciar movimiento de JOG
int iSt = 0;                //Instruccion para opcion 3 de serial (Entero)
String sSt = "";            //Instruccion para opcion 3 de serial (String)



void setup() {
  Serial.begin(9600);
  pinMode(In,INPUT);
  pinMode(Stp,INPUT);
  //pinMode(EnX,INPUT);
  pinMode(CoWo,INPUT_PULLUP);
  //pinMode(DirX,INPUT);
  pinMode(Enab,OUTPUT);
  pinMode(Ou,OUTPUT);
  pinMode(Dir,OUTPUT);
  //pinMode(NDir,OUTPUT);
  pinMode(Valores,OUTPUT);


  attachInterrupt(digitalPinToInterrupt(In), CuentaP, RISING);
  attachInterrupt(digitalPinToInterrupt(Stp), Emergencia, RISING);


}

void loop() {
  if (digitalRead(CoWo) == HIGH) {            //Pin de Jog/Run en 0 para auto
    while (Start == true) {
      if (Frec != 0) {
        digitalWrite(Valores,LOW);       //EL valor de velocidad ya a sido ingresado
        LimSwi();
        tone(Ou,Frec);
        if (serialEventRun) serialEventRun();//Revisa puerto serial por cambios en frec
        /*while (EnX == HIGH) {
          digitalWrite(Enab,digitalRead(EnX));
          digitalWrite(Dir,digitalRead(DirX));
          if (digitalRead(DirX) == HIGH) {     //Niega la direccion para X'
            digitalWrite(NDir,LOW);
          }else {
            digitalWrite(NDir,HIGH);
          }
          if (serialEventRun) serialEventRun();//Revisa puerto serial por cambios en frec
          LimSwi();                                 //Revisa limit_Switch
          tone(Ou,Frec);
        }*/

      }
    }
    noTone(Ou);

  }else {
    //Programacion de JOG  (NOTA: Se deben recibir todos los datos de Direccion,
    velocidad, pulsos antes de la opcion
    //Exepto en opcion 3 que es informacion cte, se debe enviar la velocidad antes
    de la opcion
    Start = true;

    switch (JOpc) {
      case 1:                          //Opcion de ir a Limit Switch
        //Serial.println("Op 1 hasta LS");
        if (JDir == 1) {              //Pone la direccion
          digitalWrite(Dir,HIGH);
          LimSwi();
          while (vLsP == LOW) {      //Inicia movimiento hasta llegar a limit switch
```

```
      LimSwi();
      tone(Ou,JVel);
      if (Start == false) break;      //Paro de emergencia
    }
  } else {
    digitalWrite(Dir,LOW);
    LimSwi();
    while (vLsN == LOW) {      //Inicia movimiento hasta llegar a limit switch
      LimSwi();
      tone(Ou,JVel);
      if (Start == false) break;      //Paro de emergencia
    }
  }
  noTone(Ou);                     //Detiene pulsos
  JOpc = 0;
  break;
case 2:                          //Opcion de Movimiento exacto
  //Serial.println("Op 2 Exacto");
  if (JDir == 1) {               //Pone la direccion
    digitalWrite(Dir,HIGH);
  } else {
    digitalWrite(Dir,LOW);
  }
  LimSwi();
  while (C < JPul) {             //inicia movimiento con contador de pulsos
    LimSwi();
    //Serial.print("Cont: ");
    //Serial.println(C);
    tone(Ou,JVel);
    if (Start == false) break;      //Paro de emergencia
  }
  noTone(Ou);
  JOpc = 0;
  C = 0;
  //Serial.print("JOpc: ");
  //Serial.println(JOpc);
  break;
case 3:                     //Opcion manual, enviar velocidad  antes de opcion
  do {
    LimSwi();
    if (Serial.available()>0) {

      char St = (char)Serial.read();
      sSt += St;
      if (St == '\n') {
        sSt.trim();
        iSt = sSt.toInt();
        switch (iSt) {
          case 0:                       //Derecha
            digitalWrite(Dir,HIGH);
            LimSwi();
            tone(Ou,JVel);
            sSt = "";
            JOpc = 0;
            break;
          case 1:                       //Para
            noTone(Ou);
            sSt = "";
            JOpc = 0;
            break;
          case 2:                       //Izquierda
            digitalWrite(Dir,LOW);
            LimSwi();
            tone(Ou,JVel);
            sSt = "";
            JOpc = 0;
            break;
      }
```

```
          }
        }

    } while (JOpc == 3);
  }
}
}

void serialEvent() {
  if (Serial.available()>0) {
    char inChar = (char)Serial.read();
    Tex += inChar;
    if (inChar == '\n') {
      Tex.trim();
      //Serial.println(Tex);
      if (Tex == "s") {          //Start
        Start = true;
      }
      if (Tex == "r") {          //Reset contador
        C = 0;
        Pulsos = 0;
        Frec = 200;
      }
      if (Tex.startsWith("A1f")) {       //Velocidad de X
        Temp = Tex.substring(2);
        Frec = Temp.toInt();
        //Serial.println(Frec);
      }
      if (Tex.startsWith("A1p")) {       //Cantidad de pulsos a moverse en JOG
        Temp = Tex.substring(2);
        Pulsos = Temp.toInt();
      }
      if (Tex.startsWith("A1JO")) {       //Opcion de Jog
        Temp = Tex.substring(3);
        JOpc = Temp.toInt();
        //Serial.print("JO: ");
        //Serial.println(JOpc);
      }
      if (Tex.startsWith("A1JD")) {       //Direccion de Jog
        Temp = Tex.substring(3);
        JDir = Temp.toInt();
        //Serial.print("JD: ");
        //Serial.println(JDir);
      }
      if (Tex.startsWith("A1JV")) {       //Velocidad de Jog
        Temp = Tex.substring(3);
        //Serial.println("JV: " + Temp);
        JVel = Temp.toInt();
      }
      if (Tex.startsWith("A1JP")) {       //Pulsos de Jog
        Temp = Tex.substring(3);
        //Serial.println("JP: " + Temp);
        //JPul = Temp.toInt();
        Temp.trim();
        JPul = Temp.toInt();
      }
      if (Tex.startsWith("A1JS")) {       //Iniciar movimiento en jog manual
        Temp = Tex.substring(3);
        //Serial.println("JS: " + Temp);
        JS = Temp.toInt();
        Start = true;
      }
      Tex="";
    }
  }
}

void CuentaP() {
```

```
    C++;
}

void Emergencia() {
  Start = false;
  noTone(Ou);
  JOpc = 0;
}

void LimSwi () {
  vDir = digitalRead(Dir);                                    //Lee direccion de motor
  if (analogRead(LsP) > 500) {vLsP = HIGH;} else {vLsP = LOW;} //Lee sensor de tope+
  y lo niega
  //if (analogRead(LsN) > 500) {vLsN = HIGH;} else {vLsN = LOW;} //Lee sensor de
  tope- y lo niega
  //Procedimiento para enable
  if ((vLsP == LOW)) {
    digitalWrite(Enab,LOW);
  } else {
    digitalWrite(Enab,HIGH);
  }
}
```

11.1.6. Microcontrolador codificador de ejes X, Y y Z

```
//load libraries
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>

LiquidCrystal_I2C lcd(0x3F,16,2);
//INPUT
int const SMax=9;          //Boton de Set Max
int const conf=5;          //Push btn para configuracion
const int A=3;             // Encoder A
const int B=4;             //Encoder B
const int set0=7;          //Set Home
const int Conf0 = 11;      //Configuracion del panel


//OUTPUTS
int const c1=6;            //Led de ya configurado
int const LHome=8;        //Bandera de origen (S1)
const int LMax = 10;       //Bandera de Max (s2 o S3)

//variales internas
unsigned int pulsos=0;
unsigned int puls_1cm=0;    //Pulsos por cm a grabar en eeprom
unsigned long cont=0;
float puls_CM=0;
unsigned long Max = 0;       //Maximo a llegar (S2 o S3)
String inputString = "";
long SerTemp = 0;           //Temporal para mandar por serial



void setup() {
    Serial.begin(9600);                                  //Inicializa puerto serial

    lcd.init();
    lcd.backlight();
    //EEPROM.put(0,datoInicial);            //Inicializa eeprom con valor 0 o vacio
    pinMode(c1,OUTPUT);
    pinMode(SMax,INPUT);
    pinMode(LHome,OUTPUT);
    pinMode(LMax,INPUT_PULLUP);
```

```
    pinMode(A,INPUT_PULLUP);
    pinMode(B,INPUT_PULLUP);
    pinMode(set0,INPUT);
    pinMode(Conf0,INPUT_PULLUP);
    pinMode(conf,INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(A), Cuenta, RISING);

}

void loop() {
  if (digitalRead(Conf0)==HIGH && digitalRead(conf)==LOW && digitalRead(set0)==HIGH) {
  //Combinacion de No config, SetHome y config interno pone en 0 el valor
    EEPROM.put(10, 0);
    while (digitalRead(conf) == LOW){

    }
  }
  Verifica();         //Verifica si tiene valor ingresado

  if((digitalRead(conf)==HIGH) && (digitalRead(Conf0)==HIGH)){    //Ambos botones en
  high trabajo normal
     EEPROM.get(10,pulsos);                                 //Lee EEPROM pulsos por cm
     if (pulsos == 0) {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Debe configurar");
      delay(5000);
     }
      puls_CM = float(float(cont)/float(pulsos));     //Calcula cant de pulsos dados
      lcd.setCursor(0,0);
      lcd.print("Pulsos:");
      lcd.setCursor(7,0);
      lcd.print(puls_CM);
      EEPROM.get(20,Max);
      puls_CM = float(float(Max)/float(pulsos));
      lcd.setCursor(0,1);
      lcd.print("Max:");
      lcd.print(puls_CM);
      //delay(250);                                    //Despliega centimetros movidos

      if(cont < 100) {                     //Revisa si posicion cerca o en de origen
        digitalWrite(LHome,HIGH);
        lcd.setCursor(14,1);
        lcd.print("H");
        //Serial.println("H");
        delay(300);
      }else{
        digitalWrite(LHome,LOW);
        lcd.setCursor(15,1);
        lcd.print("");
      }
      EEPROM.get(20,Max);
      if (long(cont) >= Max) {                      //Revisa si llego a punto maximo
        digitalWrite(LMax,HIGH);
        lcd.setCursor(15,1);
        lcd.print("M");
        //delay(300);
        //Serial.println("M");
      }else {
        digitalWrite(LMax,LOW);
        lcd.setCursor(15,1);
        lcd.print(" ");
      }
      if (digitalRead(set0)==HIGH) {                   //Revisa si presiono set HOME
        cont = 0;                                           //Pone en 0 el contador
      }
      if (digitalRead(SMax)== HIGH) {             //Procedimiento para grabar maximo
        EEPROM.put(20,cont);
```

```
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Grabado Set Max");
        delay(500);
      }
      delay(500);
      lcd.clear();
   }else {                                    //Si esta en forma de configuracion
    if(digitalRead(conf)==LOW && digitalRead(Conf0)==LOW){     //Si 2 botones de
    configuracion en low entra a config
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Entrando en modo");
      lcd.setCursor(0,1);
      lcd.print("de configuracion");
      digitalWrite(c1,HIGH);                          //Aviso de en configuracion
      delay(200);
      digitalWrite(c1,LOW);
      delay(200);
      digitalWrite(c1,HIGH);
      delay(200);
      digitalWrite(c1,LOW);
      delay(200);
      digitalWrite(c1,HIGH);
      delay(200);
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Press SET HOME");
      lcd.setCursor(0,1);
      lcd.print("Para iniciar");
      while (digitalRead(set0)==LOW){                      //Espera set Home
      }
      while (digitalRead(set0)==HIGH){                //Espera que suelten set home
        cont=0;                         //Mover eje para poner adelantar 1 cm exacto
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("Jog 1cm exacto");
        lcd.setCursor(0,1);
        lcd.print("Press SET HOME");
      }
      while (digitalRead(set0)==LOW){                         //Espera set home
      }
      while (digitalRead(set0)==HIGH){    //Espera que suelen set home para grabar
        EEPROM.put(10,cont);
        EEPROM.get(10,puls_1cm);            //Guarda en EEPROM cant de pulsos por cm
        Verifica();                                          //Revisa que se guardo
        lcd.setCursor(0,0);
        lcd.print("pulsos:");
        lcd.setCursor(7,0);
        lcd.print(puls_1cm);
        lcd.setCursor(0,1);
        lcd.print("Por centimetro");//Despliega en pantalla cant de pulsos para 1 cm
        delay(1000);
        lcd.clear();
      }
      digitalWrite(c1,LOW);
   }
   }
}
void Cuenta() {
  if (digitalRead(B)==HIGH){
    cont++;
  }else if(digitalRead(B)==LOW){
    cont--;
  }
}

void Verifica() {                    //Verifica si ya hay informacion en la EEPROM
```

```
  EEPROM.get(10,puls_1cm);
  if (puls_1cm == 0) {
    digitalWrite(c1,HIGH);            //Si no hay info, Prende led
  }else {
    digitalWrite(c1,LOW);             //Si ya se grabo, se apaga el led
  }
}


void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n') {
      inputString.trim();
      if (inputString == "Pos") {           //Posicion actual
        Serial.print("p"); Serial.println(cont);
        inputString = "";
      }
      if (inputString == "cm") {            //Pulsos por centimetro
        EEPROM.get(10,SerTemp);
        Serial.print("c");Serial.println(SerTemp);
        inputString = "";
      }
      if (inputString == "End") {   //Distancia de home a final de corrida en pulsos
        EEPROM.get(20,SerTemp);
        Serial.print("e"); Serial.println(SerTemp);
        inputString = "";
      }
      if (inputString.length() > 4) {       //Limpia inputString para nuevo comando
        inputString = "";
      }
      if (inputString.endsWith(":")) {       //Limpia inputString para nuevo comando
        inputString = "";
      }
      if (inputString == "Cnf") {       //Pregunta para ver si ya esta configurado
        if (digitalRead(c1) == LOW) {
          Serial.println("Ya");
        }else{
          Serial.println("No");
        }
        inputString = "";
      }
    }
  }
}
```

11.1.7. Microcontrolador selector de sensores de límite

```
//INPUTS

const int LsN=8;
const int LsM=9;
const int LsP=10;
//OUTPUTS
const int Y1LsN=3;
const int Y1LsP=4;
const int Y2LsN=5;
const int Y2LsP=6;

String Tex = "";
String Temp = "";
unsigned int MY=0;
void setup() {
  // put your setup code here, to run once:
```

```
pinMode(LsN,INPUT);
pinMode(LsM,INPUT);
pinMode(LsP,INPUT);
pinMode(Y1LsN,OUTPUT);
pinMode(Y1LsP,OUTPUT);
pinMode(Y2LsN,OUTPUT);
pinMode(Y2LsP,OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
switch (MY){

  case 1:

    digitalWrite(Y1LsN,digitalRead(LsN));
    digitalWrite(Y1LsP,digitalRead(LsM));
    break;

  case 4:

    digitalWrite(Y1LsN,digitalRead(LsN));
    digitalWrite(Y1LsP,digitalRead(LsP));
    break;

  case 2:

    digitalWrite(Y2LsN,digitalRead(LsM));
    digitalWrite(Y2LsP,digitalRead(LsP));
    break;

  case 5:

    digitalWrite(Y2LsN,digitalRead(LsN));
    digitalWrite(Y2LsP,digitalRead(LsP));
    break;

  case 3:

    digitalWrite(Y1LsN,digitalRead(LsN));
    digitalWrite(Y1LsP,digitalRead(LsM));
    digitalWrite(Y2LsN,digitalRead(LsM));
    digitalWrite(Y2LsP,digitalRead(LsP));
    break;
}
}
void serialEvent() {
  if (Serial.available()>0) {
    char inChar = (char)Serial.read();
    Tex += inChar;
    if (inChar == '\n') {
      Tex.trim();
      //Serial.println(Tex);
      if (Tex.startsWith("MY")) {
        Temp = Tex.substring(2);
        MY = Temp.toInt();
      }
      Tex="";
    }
  }
}
```

11.1.8. Microcontrolador encargado del sensor de antorcha

```
//Inputs
```

```
const int Set = 13;        //Set Amperaje ideal
const int Auto = 12;       //Automatico o manual

const int Z1 = A0;        //Antorcha 1
//const int Z2 = A1;        //Antorcha 2
const int VelZ = A3;        //Velocidad en Z

//Outputs
const int Sube = 11;       //Indicador de subir antorcha
const int Baja = 10;       //Indicador de Bajar antorcha


//Variables
float Ant1 = 0.00;        //Var Lectura de antorcha 1 en Volt
float Ant2 = 0.00;        //Lectura antorcha 2 en V
float Param = 0.10;       //Parametro limite de variacion (.08V=10A)
float DUp = 0.00;         //Parametro para iniciar subir motor
float DDn = 0.00;         //Parametro para iniciar bajar motor
float Ideal = 0.00;       //Ideal a mantener en Volt
String saVelZ = "";      //Variable a enviar
int iVelZ = 0;           //Lectura de pot de VelZ
String sVelZ = "";        //String de velocidad de Z
int aVelZ = 0;           //Mapeo de lectura de Pot de VelZ
String sJOpc = "ZJO3";        //String de opcion de jog manual


void setup() {
  Serial.begin(9600);
  pinMode(Set,INPUT_PULLUP);
  pinMode(Sube,OUTPUT);
  digitalWrite(Sube,LOW);
  pinMode(Baja,OUTPUT);
  digitalWrite(Baja,LOW);
}

void loop() {
  int Z1 = analogRead(A1);                //Lee Antorcha1
  Ant1 = (Z1 * (4.88 / 1023.000));       //Antorcha 1 a Voltage
  //int Z2 = analogRead(A2);                //Lee Antorcha2
  //Ant2 = (Z2 * (4.88 / 1023.000));       //Antorcha 2 a Voltage
  iVelZ = analogRead(VelZ);                //Lectura de Pot de Velocidad
  aVelZ = map(iVelZ, 0, 1023, 200, 2500);   //Mapeo de velocidad
  sVelZ = String(aVelZ);                    //String de velocidad
  saVelZ = "ZJV" + sVelZ;                    //String a enviar

  Setea();                                 //Llama procedimiento de parametros
  if (Ant1 != 0) {                  //Revisa si se marco el parametro de Volt ideal
    if (Ant1 > DUp) {                   //Compara si antorcha1 se acerca
      Serial.println(saVelZ);              //Manda Velocidad
      delay(10);
      Serial.println(sJOpc);              //Manda opcion 3 de jog
      delay(10);
      Serial.println("0");                //Manda comando para iniciar movimiento
      delay(10);
      digitalWrite(Sube,HIGH);            //Led indicador que la antorcha debe subir
    }else {
      Serial.println("1");                //Detiene el movimiento
      digitalWrite(Sube,LOW);//Indicador que antorcha esta entre los parametros aceptables
    }
    if (Ant1 < DDn) {                    //Compara si antorcha1 de aleja
      Serial.println(saVelZ);              //Manda Velocidad
      delay(10);
      Serial.println(sJOpc);              //Manda opcion 3 de jog
      delay(10);
      Serial.println("2");                //Manda comando para iniciar movimiento
      delay(10);
      digitalWrite(Baja,HIGH);            //Led indicador que la antorcha debe bajar
    }else {
```

```
      Serial.println("1");               //Detiene el movimiento
      digitalWrite(Baja,LOW); //Indicador que antorcha esta entre los parametros aceptables
    }
  }
}

void Setea() {                        //Procedimiento para setear parametros
  if (digitalRead(Set) == LOW) {      //Si presiono boton de SET
    if (Ant1 != 0){ Ideal = Ant1;}    //Graba en Ideal el volt ideal a mantener
    DUp = Ideal - Param;               //Parametro cuando la antorcha se acerca y hay que subir motor
    DDn = Ideal + Param;               //Parametro cuando la antorcha se aleja y hay que bajar motor
    Serial.println(Ideal);             //Mandarlo a pantalla extra
  }
}
```

Glosario

Antorcha: Herramienta utilizada para soldar con arco protegido. 3, 26

Chumacera: Pieza de metal o madera, con una muesca en que descansa y gira cualquier eje de maquinaria. 27

Comburente: Que provoca o favorece la combustión de otras sustancias. 9

Comunicación serial: Proceso de envío de datos de un bit a la vez, de forma secuencial. 21, 26

Estator: Parte fija de una máquina rotativa y uno de los dos elementos fundamentales para la transmisión de potencia. 13, 14

Memoria FLASH: Un tipo de memoria informática de alta velocidad. 19

Microcontrolador: Circuito integrado utilizado para controlar el funcionamiento de una tarea en específico. 3, 13, 16, 19–21, 26, 27, 30–36, 52

Potenciómetro: Resistencia regulable en un circuito eléctrico. 27, 51

Relevador: Dispositivo electromagnético que es usado un interruptor controlado por una señal digital. 6

Reluctancia: : Resistencia que un circuito ofrece al paso del flujo magnético. 13, 14

Selector: Dispositivo que en ciertos aparatos o máquinas sirve para elegir la función deseada. 25, 26, 37, 51