Trabajo Fin de Máster
Máster Universitario en Ingeniería Electrónica, Robótica y Automática

Control de equipo de posicionado de piezas semiautomático en zona de trabajo de robot

Autor: Jorge Andrés Tapia Herrera

Tutor: Luis Fernando Castaño Castaño

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo Fin de Máster
Máster Universitario en Ingeniería Electrónica, Robótica y Automática

Control de equipo de posicionado de piezas semiautomático en zona de trabajo de robot

Autor:
Jorge Andrés Tapia Herrera

Tutor:
Luis Fernando Castaño Castaño

Dpto. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Trabajo Fin de Máster: Control de equipo de posicionado de piezas semiautomático en zona de trabajo de robot

Autor: Jorge Andrés Tapia Herrera

Tutor: Luis Fernando Castaño Castaño

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

A mis padres Jorge y Carmita.

A mis abuelos Andrés, Zoila, José y Laura+.

Agradecimientos

Quisiera agradecer primeramente a Dios, por la oportunidad que tuve de venir a realizar mis estudios en una nación tan grande como lo es España. Quiero hacer llegar de igual modo mis agradecimientos a mi tutor Luis Fernando Castaño Castaño por permitirme desarrollar un proyecto junto a él, demostrando siempre un gran apoyo, consejo y guía en todo momento de la realización y culminación de este trabajo.

Como poder no mencionar de igual manera a mis padres Jorge y Carmita que fueron los pilares fundamentales de mi vida, me han demostrado por sus enseñanzas que nada en la vida resulta fácil desde un inicio, todo en lo que se emprenda conlleva un sacrificio enorme día con día y que de ellos he aprendido muy bien porque me lo demuestran con sus acciones, sus ganas de salir adelante y su profesionalismo al desenvolverse en cada uno de los ámbitos laborales que ellos ocupan. Aunque vosotros padres míos no puedan estar presentes el día que yo defienda este trabajo ya que una larga distancia nos separa por el momento, estoy seguro que de manera espiritual siempre me están acompañando y nunca me dejarán solo, sé de igual manera que con sus buenos deseos y augurios sabré salir victorioso de este objetivo que me he puesto en el camino y al final de todo podré reencontrarme con vosotros para mirarles a los ojos de frente y con orgullo poder decirles que mis esfuerzos y mis ganas de surgir para ser alguien mejor valieron la pena, que cualquier momento de miedo o temor que haya pasado en el transcurso de este tiempo lejos de casa los he sabido sobrellevar y reponerme hasta culminar lo que empecé. Quiero agradecer de igual manera a mis hermanos Diego, Melany y demás miembros de mi familia que me han hecho una inmensa falta en todo esto tiempo que no he podido verlos.

Jorge Andrés Tapia Herrera

Sevilla, 2018

Resumen

El proyecto que se ha desarrollado tiene como objetivo principal el diseño de un posicionador de piezas para situarlas en una zona de trabajo de un robot manipulador. La característica fundamental del posicionado de la pieza es la precisión con la que debe garantizarse las coordenadas X, Y, Z de la pieza respecto de un sistema de referencia propio del posicionador, que a su vez sea conocido respecto al sistema de referencia del robot.

Este sistema se va a implementar en los equipos docentes de robótica del departamento de Ingeniería de Sistemas y Automática de la Escuela Técnica Superior de Ingeniería de Sevilla.

Hasta ahora, las manipulaciones de piezas por parte del alumnado consistían una fase de posicionado manual, invadiendo la zona de trabajo del robot (con el peligro que esta acción puede suponer), seguido de la determinación del punto de cogida de la pieza operando con el robot manualmente con una consola o mediante el PC, una vez conocida la posición de cogida se pasaba a programar al robot con las operaciones y movimientos a realizar. En sucesivas depuraciones del programa, el alumno debía posicionar nuevamente la pieza, de forma manual en el mismo lugar, lo que obligaba a marcarlo en la mesa. Estas acciones suponían, además, un peligro añadido por la invasión del espacio de trabajo del robot.

Con el sistema de posicionado semiautomático diseñado, se consigue que el alumno fije de antemano la posición de la pieza (coordenadas X e Y), y desde una zona segura, introduzca la pieza y genere la orden para que el posicionador lleve la misma a la zona de trabajo del robot, sin tener que acceder esta zona.

Por tanto, se consiguen tres objetivos:

1. Se incrementa la seguridad para las personas en las prácticas con los robots.
2. Se agilizan las operaciones de posicionado.
3. Se mejora la precisión de posicionado y repetitividad en los experimentos.

El posicionador está formado por una parte electromecánica que consistente en una cinta transportadora movida por un motor de corriente continua, que dispone en sus rodillos de un encoder incremental (eje Y de la pieza) y calibrador lineal (eje X de la pieza). Y una parte electrónica basada en un microcontrolador Arduino, que es el encargado de hacer fundamentalmente el interfaz hombre-máquina, así como la supervisión y control de todo el sistema.

Cabe destacar el esfuerzo que se ha hecho para poder integrar este equipo con el controlador del robot (o con un PLC) a través de entradas/salidas digitales, lo que permitiría múltiples operaciones de control remoto desde estas plataformas. Se han desarrollado específicamente modos de trabajo Manual/Automático y Local/Remoto para dar una respuesta flexible a las necesidades que puedan surgir en el posicionado de piezas.

Abstract

The project that has been developed has as main objective the design of a positioner of pieces to place them in a working area of a manipulator robot. The fundamental characteristic of the positioning of the piece is the precision with which the X, Y, Z coordinates of the piece must be guaranteed with respect to a reference system specific to the positioner, which in turn is known with respect to the reference system of the robot.

This system will be implemented in the robotics teaching teams of the Systems and Automation Engineering Department of the School of Engineering of Seville.

Up to now, the manipulations of pieces by the students consisted of a phase of manual positioning, invading the working area of the robot (with the danger that this action may involve), followed by the determination of the point of catching the piece operating with the robot manually with a console or through the PC, once the catch position was known, the robot was programmed with the operations and movements to be performed. In successive refinements of the program, the student had to position the piece again, manually in the same place, which forced him to mark it on the table. These actions also posed an added danger due to the invasion of the robot's workspace.

With the semiautomatic positioning system designed, the student is fixed in advance the position of the piece (coordinates X and Y), and from a safe area, enter the piece and generate the order for the positioner to take it to the working area of the robot, without having to access this area.

Therefore, three objectives are achieved:

1. Security is increased for people in robot practices.
2. Positioning operations are streamlined.
3. The precision of positioning and repetitiveness in the experiments is improved.

The positioner is formed by an electromechanical part consisting of a conveyor belt driven by a DC motor, which has an incremental encoder (Y axis of the part) and linear calibrator (X axis of the part) on its rollers. And an electronic part based on an Arduino microcontroller, which is responsible for fundamentally human-machine interface, as well as the supervision and control of the entire system.

It should be noted the effort that has been made to integrate this equipment with the robot controller (or with a PLC) through digital inputs / outputs, which would allow multiple remote control operations from these platforms. Work modes Manual / Automatic and Local / Remote have been specifically developed to give a flexible response to the needs that may arise in the positioning of parts.

Índice

Agradecimientos **ix**

Resumen **xi**

Abstract **xiii**

Índice **xv**

Índice de Tablas **xvii**

Índice de Figuras **xvii**

Notación **xix**

1 Introducción **1**
1.1 Posicionado de piezas *1*
1.2 Aplicación *1*
1.3 Plataforma Arduino. Hardware Libre *2*

2 Descripción del Hardware **3**
2.1 Arduino Mega 2560 *3*
2.1.1 Alimentación 4
2.1.2 Memoria 5
2.1.3 Entradas y Salidas 5
2.1.4 Comunicación 5
2.2 Motor de Corriente Continua y Driver L298N para motores DC *6*
2.2.1 Conexionado para alimentación del módulo L298N 6
2.3 Encoder Rotativo Incremental *7*
2.3.1 Especificaciones técnicas encoder incremental 7
2.3.2 Cables de Conexiones de encoder incremental 8
2.4 Calibrador Digital *8*
2.5 LCD *9*
2.6 Pulsadores y LEDS *10*
2.7 Sensor fotoeléctrico OMRON *10*
2.7.1 Cables de conexión de sensor fotoeléctrico 11
2.8 Banda transportadora con elementos integrados *11*

3 Interfaz de Usuario **12**
3.1 Croquis de la Botonera *12*
3.2 Modos de configuración e interacción con los botones del sistema *13*
3.3 Modo de operación del usuario con el sistema *14*
3.3.1 Respuesta del sistema estando en modo de Avance Absoluto 15
3.3.2 Respuesta del sistema estando en modo de Avance Discreto 16
3.4 Modo de funcionamiento autónomo operación del robot ABB con el sistema *17*
3.4.1 Señales de salida del microcontrolador hacia el robot ABB 17
3.4.2 Señales de entrada del microcontrolador desde el robot ABB 18
3.5 Accionamiento de sensor fotoeléctrico en el sistema *19*

4 DESARROLLO **20**
4.1 Uso de pantalla LCD *20*

4.2 Pulsadores de manipulación simple, pulsadores de manipulación con anti rebote. 21
4.3 Control de motor de corriente continua mediante Arduino Mega 2560 23
4.4 Lectura de las señales de Encoder Rotativo Incremental 25
4.5 Lectura de señales obtenidas del calibrador digital 27
4.6 Lecturas de señal de sensor fotoeléctrico 29
4.7 Manejo del Driver L298N 29
4.8 Sistema completo para posicionamiento de piezas de manera manual y autónoma 32
5 Conclusiones y desarrollos futuros 64
Referencias 65
Bibliografía y Sitios web de consulta 65

ÍNDICE DE TABLAS

Tabla 1: Especificaciones Técnicas de Arduino Mega 2560 3

ÍNDICE DE FIGURAS

Figura 1: Croquis de los posicionadores y zonas de trabajo 2
Figura 2: Arduino Mega 2560. [1] 3
Figura 3: Pines de conexión del módulo L298N 6
Figura 4: Alimentación de módulo L298N 6
Figura 5: Salidas A y B del encoder incremental 7
Figura 6: Calibrador digital y pines de señales a utilizar 8
Figura 7: Pines que conforman el LCD 16x2. [2] 9
Figura 8: Pulsador a utilizar con Arduino 10
Figura 9: Diodos LED 10
Figura 10: Sensor fotoeléctrico E3JK-R4M de marca OMRON 10
Figura 11: Estructura de la banda transportadora 11
Figura 12: Botonera a disposición del usuario del sistema 13
Figura 13: Modo de funcionamiento según el tipo de avance 13
Figura 14: Posición origen del calibrador digital, eje X cero y eje Y cero 14
Figura 15: Pieza ubicada en el calibrador, eje X 90mm y eje Y 0mm 14
Figura 16: Pieza ubicada en la banda transportadora, eje X 90mm y eje Y 200mm 15
Figura 17: Avance absoluto, posición inicial en Y de 0mm, posición final en Y de 300mm 15
Figura 18: Avance absoluto, posición inicial en Y de 300mm, posición final en Y de 200mm 15
Figura 19: Avance absoluto, posición inicial en Y de 200mm, posición final en Y de 500mm 16
Figura 20: Avance discreto, posición inicial en Y de 0mm, posición final en Y de 15mm 16
Figura 21: Avance discreto, posición inicial en Y de 15mm, posición final en Y de 30mm 16
Figura 22: Avance discreto, posición inicial en Y de 30mm, posición final en Y de 45mm 16
Figura 23: Avance discreto, posición inicial en Y de 45mm, posición final en Y de 30mm 17
Figura 24: Avance discreto, cambio entre modos de configuración y ejecución 17
Figura 25: Envío de posición en X hacia el robot ABB mediante señal PWM 18
Figura 26: Envío de posición en Y hacia el robot ABB mediante señal PWM 18
Figura 27: Conexionado de Arduino Mega 2560 con LCD 16x2. [2] 20
Figura 28: Conexión de un pulsador (input Arduino) y diodo led (output Arduino). [3] 21

Figura 29: Conexionado de motor DC mediante uso de transistor TIP120. [3] 24

Figura 30: Conexión de motor de corriente continua con Encoder Rotativo Incremental. [3] 27

Figura 31: Circuito de conexión para obtener los datos provenientes del calibrador digital. [3] 27

Figura 32: Circuito de conexión de conexión de driver L298N, Arduino Mega2560 y motor DC. [3] 30

Figura 33: Circuito de elementos electrónicos del sistema de posicionamiento de piezas. a) Conexiones en el microcontrolador del sistema, b) Conexión de señales de microcontrolador a robot ABB, c) Diagrama completo de todos los elementos en el proyecto. [3] 33

Figura 34: LCD 16x2 con comunicación I2C 60

Figura 35: Pruebas iniciales del sistema 60

Figura 36: Pruebas de funcionamiento inicial con la banda transportadora 61

Figura 37: Pruebas para comunicación de microcontrolador con robot ABB 61

Figura 38: Pruebas de funcionamiento del sistema con la banda transportadora. 62

Figura 39: Pieza ubicada al inicio de la banda transportadora con medida en X de 95.24mm 62

Figura 40: Pieza ubicada en eje X 95mm y en eje Y 100mm 63

Figura 41: LCD del sistema que indica la ubicación de la pieza en eje X 95mm y en eje Y 100mm 63

Notación

ABB	Acrónimo de Ase Brown Boveri, es una empresa multinacional.
CA	Corriente Alterna
CC	Corriente Continua
CLK	Señal de reloj
cm	Centímetros
DATA	Señal de datos
g	Gramos
GND	Tierra o masa
KB	Kilo Byte
Km	Kilómetros
LCD	Pantalla de tecnología LCD (Liquid Cristal Display)
LSB	Bit menos significativo (Least Significant Bit)
MHz	Mega Hertzios
mm	Milímetros
PWM	Modulación por ancho de pulsos (Pulse width modulation)
USB	Bus universal en Serie (Universal Serial Bus)
V	Voltios

1 INTRODUCCIÓN

1.1 Posicionado de piezas

En las prácticas que se venían haciendo por parte de los alumnos en los equipos docentes de robótica del departamento de Ingeniería de Sistemas y Automática de la Escuela Técnica Superior de Ingeniería de Sevilla limitaba al alumno a posicionar las piezas dentro de la zona de trabajo de un robot de manera única y exclusivamente manual invadiendo la zona en donde el robot desempeña sus tareas, lo que desencadenaba en acciones que resultaban en imprecisión de la ubicación de la pieza dentro de la zona de trabajo del robot a ocupar, mencionar también el peligro presente al que el alumno estaba sometido por cualquier eventualidad que representa trabajar de esta manera con estos tipos de equipos, luego de esto el alumno una vez conocido el punto de cogida de la pieza continuaba con la programación de lo que vendría a realizar el robot y los respectivos movimientos que debería hacer el mismo para que la pieza fuera recolectada del punto donde estaba ubicada.

Tras varios intentos de recolección de la pieza hasta encontrar la ubicación adecuada para que el robot funcione de la mejor manera, el alumno procedía a dejar marcas en donde se fijaba la pieza, invadiendo así el lugar de trabajo del robot.

Ahora bien, con el posicionador semiautomático que se realizará en este proyecto ayudará en las tareas de ubicación de las piezas, evitando la invasión de la zona de trabajo por parte del alumno, también se disminuirá el riesgo que supone interactuar con estos equipos, consiguiendo ciertas ventajas de uso que se mencionan a continuación:

- Posicionamiento de piezas en medidas de ejes X e Y que el usuario requiera para interactuar con el robot.
- Respuesta por parte del sistema en su interfaz, donde indicará la ubicación en el espacio en ejes X e Y dentro de la banda transportadora.
- Una mejora en la rapidez para trabajos de posicionamiento realizados con este tipo de robots.
- Manejo del sistema que no implica riesgo hacia el alumno.
- Capacidad de uso del sistema de manera remota para efectuar trabajos desde el robot.

1.2 Aplicación

Este trabajo se enfoca en realizar un posicionador de piezas, que incorporará para el desarrollo del mismo de dos partes fundamentales, por un lado, la cinta transportadora que dispone de un motor de corriente continua, un encoder incremental, un calibrador digital ubicado a lo ancho de su estructura, así también un sensor de tipo fotoeléctrico. Por otra parte, tenemos la electrónica de este sistema, que establecerá las órdenes dictadas a la cinta transportadora usando un microcontrolador Arduino, este a su vez facilitará las tareas para realizar la interfaz hombre-máquina, la lectura de respuesta de los elementos presentes en el proyecto, y también servirá para la comunicación del sistema con el robot que se esté trabajando. El control del posicionador de piezas en la zona de trabajo de dos robots se muestra en la Figura 1.

El posicionador se podrá emplear mediante la interacción de manera manual del usuario con la interfaz del sistema, así como también de manera remota mediante las acciones dictadas por el robot (o con un PLC) que serán recibidas y procesadas por el Arduino.

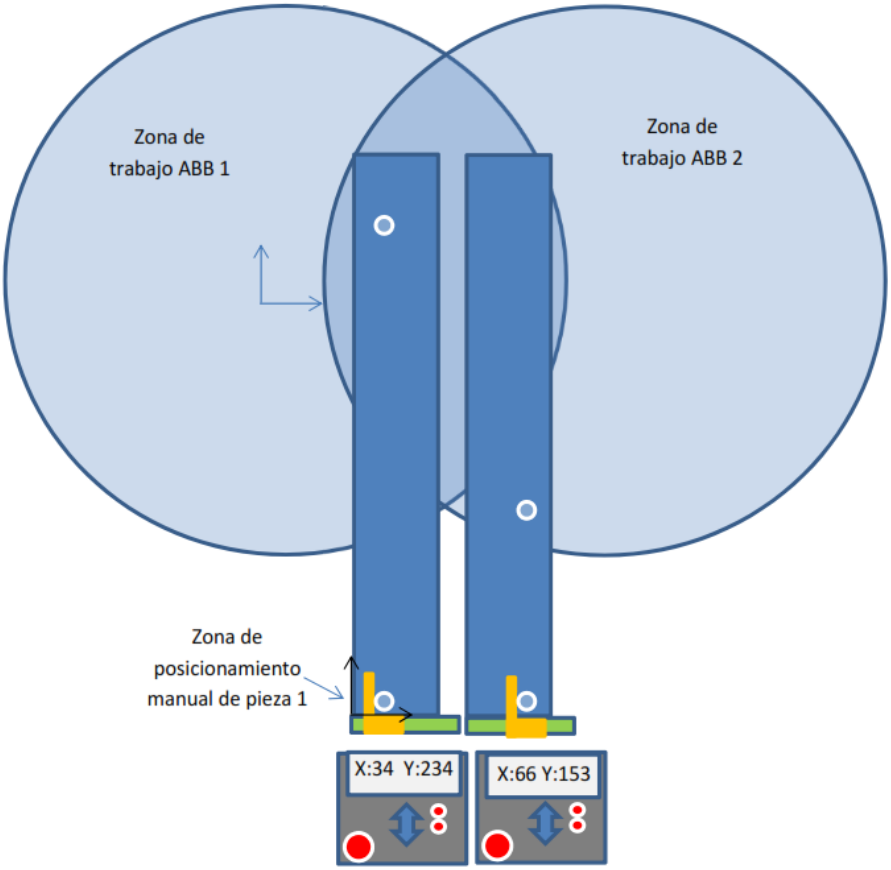


Figura 1: Croquis de los posicionadores y zonas de trabajo

Adicional a esto el sistema a desarrollar facilita la opción de funcionar cuando el microcontrolador Arduino haya caído, quemado o retirado del sistema, teniendo en cuenta que las acciones que se podrán hacer al no estar presente el Arduino son de tipo manual, provocando que la cinta transportadora avance, se detenga o retroceda.

1.3 Plataforma Arduino. Hardware Libre

Se ha elegido para este proyecto el uso de la tarjeta Arduino, la cual trabajará en conjunto con otros elementos que servirán para realizar el control requerido en la banda transportadora.

Arduino es una plataforma de electrónica de código abierto basada en hardware y software fácil de utilizar, las entradas que tiene arduino puede detectar señales de sensores, botones, etc., y producir una respuesta a sus salidas que servirán por ejemplo para dar movimiento a un motor, la activación de un led o publicación de un mensaje. Para realizar todo esto se utiliza el lenguaje de programación de Arduino y el software Arduino (IDE) basado en el procesamiento.

Arduino es relativamente fácil de utilizar, y al ser una comunidad integrada por estudiantes, programadores, profesionales, o cualquier persona que le interese la programación, presenta variedad de opciones al momento de realizar un proyecto ya que se tiene un soporte en cuanto al conocimiento del mismo y múltiples aplicaciones en donde se hace uso de estas tarjetas, toda la información se la puede encontrar de manera sencilla en la red ya que es de dominio público.

Para el proyecto se ha hecho uso de la tarjeta Arduino Mega2560, ya que presenta las mejores opciones en cuanto al hardware para la utilización de todos los elementos electrónicos que se requieren para dar movimiento y control a la cinta transportadora, que conforma parte del posicionador de piezas.

2 DESCRIPCIÓN DEL HARDWARE

En la elaboración de este proyecto hay dos importantes partes que corresponden al hardware y software, los cuales deben trabajar en conjunto para así poder llevar a cabo el correcto funcionamiento de la banda transportadora con los modos de uso requeridos.

Los principales elementos en cuanto al hardware que se ocupa para este trabajo son los que se indican a continuación:

2.1 Arduino Mega 2560

Este dispositivo microcontrolador está basado en el ATmega2560 (Figura 2), el mismo posee 54 pines digitales de entrada / salida (de los cuales 15 se pueden usar como salidas PWM), 16 entradas analógicas, 4 UART (puertos serie de hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, un encabezado ICSP, y un botón de reinicio. Para poder utilizar el mismo es necesario conectarlo mediante cable USB a una computadora, se puede alimentar de voltaje al mismo mediante un adaptador de CA a CC o con una batería.

Figura 2: Arduino Mega 2560. [1]

Las especificaciones técnicas del fabricante para el Arduino Mega 2560 se puede observar de una manera más detallada en la Tabla 1.

Tabla 1: Especificaciones Técnicas de Arduino Mega 2560

Microcontrolador	ATmega2560
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Pines de E / S digitales	54 (de los cuales 15 proporcionan salida de PWM)
Clavijas de entrada analógica	dieciséis

Corriente DC por Pin E / S	20 mA
Corriente DC para 3.3V Pin	50 mA
Memoria flash	256 KB de los cuales 8 KB utilizados por el gestor de arranque
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz
LED_BUILTIN	13
Longitud	101.52 mm
Anchura	53.3 mm
Peso	37 g

2.1.1 Alimentación

Arduino Mega 2560 tiene la posibilidad de ser alimentado mediante la conexión USB que puede ser proporcionada por una computadora por poner un ejemplo o una fuente de alimentación externa. La fuente de poder se selecciona automáticamente.

La conexión que se realiza de manera externa para la alimentación puede ser provista mediante un adaptador CA a CC (wall-wart) o una batería, la conexión para uso con adaptador se la realiza mediante un conector macho de dimensiones de 2.1mm en el conector hembra que posee placa y para la conexión usando la batería, se haría uso de cables de la misma los cuales harían conexión con los pines Vin y GND según corresponda a la parte del Arduino especificado como POWER.

Como se había visto anteriormente en las especificaciones técnicas de la Tabla 1 la placa de Arduino Mega 2560 puede trabajar con una alimentación externa de una tensión entre 6 a 20 voltios, hay que mencionar que si el voltaje de alimentación suministrado es inferior a los 7 voltios el pin de la tensión de 5V puede proporcionar tensiones inferiores a 5V y la placa podría tornarse inestable, por otro lado si se utilizan más de 12V los reguladores de voltaje con los que cuenta la placa pueden provocar un aumento en su temperatura y tendrían como consecuencia daños en la placa por sobrecalentamiento, el rango recomendable de la tensión de entrada está entre los 7 a 12 voltios, con ello asegurando así que la tarjeta funcione de una manera adecuada y sin ningún inconveniente referido a este tipo de situaciones que podrían suceder.

A continuación, se detallan los pines de alimentación de la tarjeta:

- **VIN:** Entrada de voltaje de la placa cuando se realiza una alimentación externa.
- **5V:** Este pin produce una tensión de 5 V regulados.
- **3V3:** Fuente de voltaje con una tensión de 3.3V la misma que es generada por el regulador con el que cuenta la placa, el consumo máximo de corriente es de 50 mA.
- **GND:** Pines que posee la placa para tener conexión a tierra.
- **IOREF:** Pin de la placa encargado de proporcionar la referencia de tensión con la que trabaja el microcontrolador.

2.1.2 Memoria

El ATmega2560 tiene 256KB de memoria flash para almacenar código, tiene 8 KB de memoria SRAM y 4KB de EEPROM. Para el gestor de arranque son utilizados 8KB de memoria.

2.1.3 Entradas y Salidas

Cada uno de los 54 pines digitales del Mega se puede utilizar como entrada o salida, usando las funciones pinMode ( ), digitalWrite ( ) y digitalRead ( ). Operan a una tensión equivalente a los cinco voltios. Cada pin puede proporcionar o recibir 20 mA teniendo en cuenta las condiciones de funcionamiento recomendadas y tiene una resistencia interna de pull-up (desconectada por defecto) de 20-50kohmios. Para evitar daños permanentes en el microcontrolador no se debe exceder de un máximo en corriente igual a 40mA.

Además, algunos pines tienen funciones especializadas:

- **Serie: 0 (RX) y 1 (TX); Serial 1: 19 (RX) y 18 (TX); Serial 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX).** Se usa para recibir (RX) y transmitir (TX) datos en serie TTL. Los pines 0 y 1 también tienen conexión a los pines correspondientes del chip ATmega16U2 USB a TTL.
- **Interrupciones externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3) y 21 (interrupción 2).** Estos pines se pueden configurar para activar una interrupción en un nivel bajo LOW (0V), flancos de subida o bajada, o un cambio en el nivel.
- **PWM:** 2 a 13 y 44 a 46. Proporcionan salida PWM (Pulse Wave Modulation) de 8 bits con la función analogWrite ( ).
- **SPI:** 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Estos pines permiten comunicación SPI utilizando la biblioteca SPI . Los pines SPI también se fraccionan en el encabezado del ICSP, que es físicamente compatible con el Arduino / Genuino Uno y los antiguos Duemilanove y Diecimila Arduino.
- **LED:** 13. Hay un LED integrado conectado al pin digital 13. Cuando el pin tiene un valor ALTO o lo que es lo mismo un valor HIGH (5V), el LED se encenderá, cuando el pin está BAJO es decir con un valor de LOW (0V), el LED se aparará.
- **TWI:** 20 (SDA) y 21 (SCL). Soporte de comunicación TWI utilizando la biblioteca Wire. Se debe tener en cuenta que estos pines no están en la misma ubicación que los pines TWI en las antiguas placas Duemilanove o Diecimila Arduino.

El Mega 2560 tiene 16 entradas analógicas, cada una de las cuales proporciona 10 bits de resolución (1024 valores diferentes). Por defecto, miden desde tierra a 5 voltios, aunque es posible cambiar el extremo superior de su rango empleando el pin AREF y la función analogReference ( ).

Hay un par de otros pines en la placa de Arduino Mega 2560

- **AREF.** Voltaje de referencia para las entradas analógicas. Usado con analogReference ( ).
- **Reset.** Suministra un valor LOW (0V) para reiniciar el microcontrolador. Habitualmente se usa para adicionar un botón de reinicio a los shields que no permiten acceso a este botón en la placa.

2.1.4 Comunicación

La placa de Arduino Mega 2560 tiene diversas facilidades para comunicarse de ser el caso con computadora, otra placa de Arduino u otros microcontroladores.

El ATmega2560 proporciona cuatro UART de hardware para comunicación TTL (5V). Un ATmega16U2 en la placa canaliza uno de estos a través de USB y suministra un puerto virtual para el software en la computadora (las máquinas Windows requerirán un archivo .inf, pero las máquinas OSX y Linux reconocen la placa de forma automática). El software Arduino (IDE) incluye un monitor serie que posibilita el envío de datos de texto simples desde y hacia la placa de Arduino. Los LED RX y TX de la placa parpadearán cuando los datos se transmitan a través del ATmega8U2 / ATmega16U2 chip y la conexión USB al ordenador (pero no para comunicación serial en los pines 0 y 1).

2.2 Motor de Corriente Continua y Driver L298N para motores DC

En este proyecto se hará uso de un motor de corriente continua el mismo que tiene una alimentación igual a 24V, estos motores son muy utilizados en proyectos de tecnología como por ejemplo para la creación de coches teledirigidos, mini robots, robots móviles, etc. Son fáciles de utilizar y se los pueden adquirir a precios muy convenientes.

El motor que se va a usar permitirá el avance o parada de la cinta transportadora a controlar en este proyecto, para realizar el control del motor de corriente continua se ha visto coherente la utilización de Driver L298N. El driver a ocupar presenta entre sus características la posibilidad de poder utilizarlo para tener control de dos motores de corriente continua o un motor paso a paso bipolar de hasta 2 amperios.

El módulo está basado en el chip L298N, dicho módulo cuenta con los componentes necesarios para poder trabajar sin requerir alimentación de tensión adicionales, de los elementos con los que cuenta se puede mencionar al regulador LM7805 que suministra una tensión de 5V a la parte lógica del integrado que conforma el mismo, al igual que diodos que sirven como protecciones. El módulo a emplear tiene ciertos jumpers de selección para habilitar las salidas (A y B) con las que cuenta la tarjeta.

- **Salida A:** Compuesta por un OUT1 y OUT2, con pin de habilitación correspondiente a ENA.
- **Salida B:** Compuesta por OUT3 y OUT4, con pin de habilitación correspondiente a ENB.

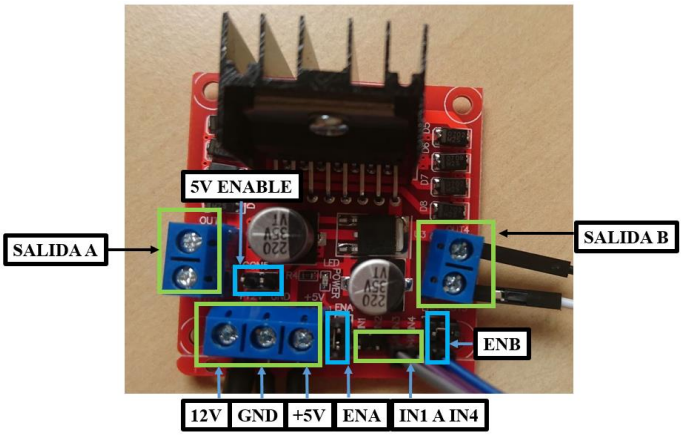


Figura 3: Pines de conexión del módulo L298N

Como se puede observar en la parte inferior la Figura 3 se tiene a disposición en el módulo L298N pines marcados como IN1, IN2, IN3 e IN4 los cuales servirán para realizar el control de dicho módulo.

2.2.1 Conexionado para alimentación del módulo L298N

Se tiene la opción de poder conectar a la tarjeta del módulo mediante dos maneras (Figura 4), ya que el integrado LM7805 posibilita esta opción.

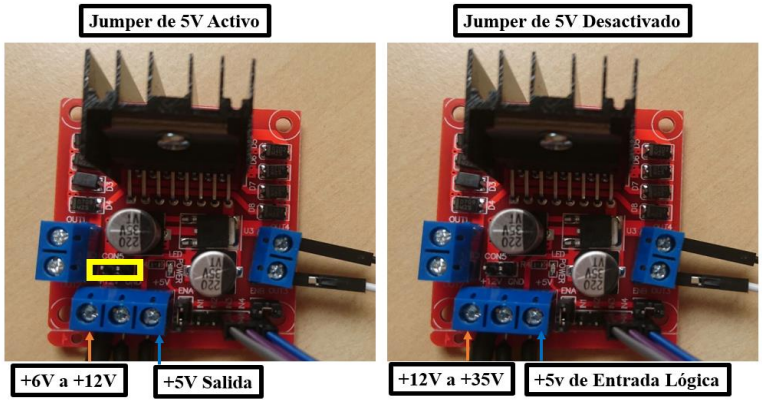


Figura 4: Alimentación de módulo L298N

Mientras el jumper de selección de 5V este activo, el mismo admite tensiones de alimentación entre 6V a 12V de corriente continua. En este caso el regulador está activo y el pin indicado como +5V en el módulo contará con un voltaje de 5V de DC. El consumo recomendado no debe superar los 500mA.

Si el jumper de selección de 5V se encuentra desactivado o no se hace uso del mismo, el módulo en este caso posibilita la alimentación para tensiones entre 12V a 35V de corriente continua. El regulador con este tipo de conexión del jumper no se encuentra trabajando, por lo que es necesario que el pin indicado como +5V en el módulo deberá ser alimentado con una tensión de 5V lo cuales servirán para que funcione la parte lógica del chip L298N.

Teniendo en cuenta estas dos posibles acciones para la manipulación del módulo se debe tener mucha precaución de no conectar una tensión de 5VDC al pin +5V cuando el jumper de selección se encuentra activado, ya que podría provocar un corto circuito y tendría como consecuencia un daño permanente en el módulo.

2.3 Encoder Rotativo Incremental

Para realizar las mediciones de rotación del motor de corriente continua que conforma la banda transportadora se hará uso del encoder incremental de serie LPD3806-600BM el cual es un codificador giratorio con una alta precisión, el encoder aportará de gran manera para proporcionar información física directa de donde se encuentra posicionado el motor.

El codificador giratorio tiene dos salidas de onda cuadrada (A y B) con desfase de 90 grados una de la otra. Siempre que el impulso de la señal A esté en el flanco descendente el impulso de la señal B será leída. Como se puede observar en la Figura 5 cuando el encoder gira en sentido horario, el pulso de B es siempre positivo. El pulso de B es negativo cuando el encoder gira en sentido anti horario. Teniendo conectadas estas dos salidas a un microcontrolador es posible determinar la dirección de giro. Las dos salidas (A y B) representan el movimiento del disco del encoder como un tren de impulsos modulado en cuadratura.

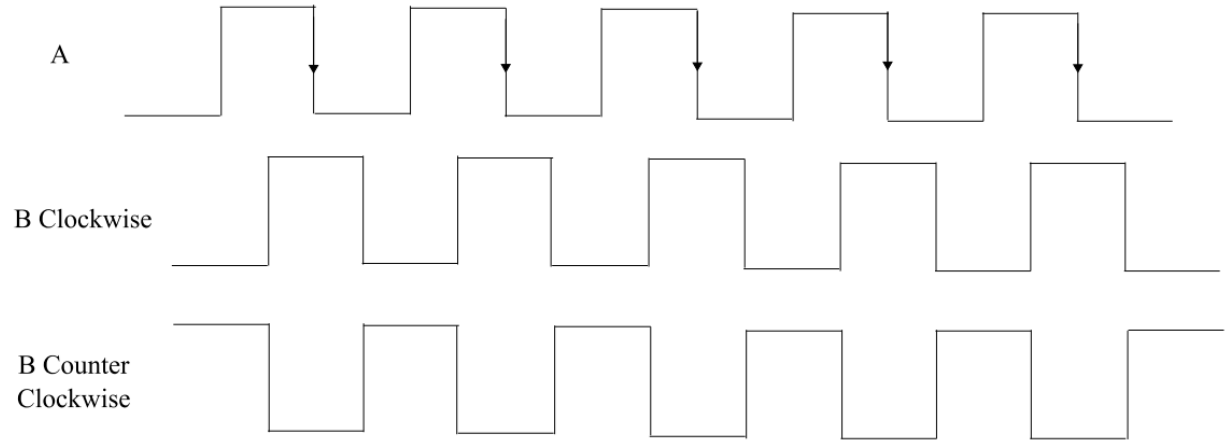


Figura 5: Salidas A y B del encoder incremental

2.3.1 Especificaciones técnicas encoder incremental

El encoder rotativo de serie LPD3806 600BM G5 24C tiene las siguientes especificaciones que se indican a continuación:

- 600 pulsos/ revolución para una sola fase. Por lo tanto, la salida de dos fases conduce a 2400 pulsos/revolución.
- Velocidad mecánica máxima: 5000 revoluciones/minuto.
- Respuesta de frecuencia: 0-30KHz.

2.3.2 Cables de Conexiones de encoder incremental

El encoder cuenta con cuatro cables los cuales están diferenciados por el color que tienen cada uno de ellos.

- Rojo: Alimentación de 5-24 VDC
- Negro: GND o tierra.
- Verde: Fase A.
- Blanco: Fase B.

2.4 Calibrador Digital

El calibrador servirá para tomar las medidas de las piezas que se coloquen a lo ancho de la cinta transportadora, este instrumento tiene un indicador LCD que va mostrando el cambio de la medición cuando se mueve a través de la regleta metálica que conforma su estructura. La herramienta de medición posee una cubierta de plástica que es donde se encuentra el LCD, removiendo la parte superior de esta cubierta se accede a cuatro pines los mismos que indican la alimentación de 1.5VDC, señal de reloj (CLK), señal de datos y por último un pin el cual servirá para realizar la conexión a tierra, todo esto como se indica en la Figura 6.

Figura 6: Calibrador digital y pines de señales a utilizar

Se describe a continuación los puntos más relevantes a tener en cuenta para la transmisión de datos del calibrador digital que se quieren captar mediante la utilización de Arduino. Mientras el calibrador se encuentre encendido los datos se irán transmitiendo de manera continua.

- Cada grupo de datos que se van transmitiendo consiste en una trama de 24 bits.
- El calibrador utiliza un nivel lógico de tensión de 1.5VDC, esta señal debe ser amplificada para que pueda ser detectada por la lógica de Arduino llevándola hasta una tensión de 5VDC.
- Los datos son transmitidos a través de dos cables, uno hace referencia a la señal de reloj (CLK) y el otro indica la línea de datos (DATA).
- El nivel lógico de la línea de datos debe leerse en la transición de la línea de reloj de ALTO a BAJO.
- En cada cadena de 24 bits hay un período más largo durante el cual la señal de reloj (CLK) permanece en alto.
- El bit de datos 21 es el bit que indica el signo, si este bit se encuentra en alto el valor será negativo.

- El primer bit siempre es alto, esta condición viene definida de fábrica por el calibrador digital.
- Los siguientes bits codifican el valor de la lectura del calibrador, iniciando por el bit menos significativo (LSB).
- En modo de medida de mm, el valor resultante corresponde a la medida de la pinza multiplicada por 100.

2.5 LCD

El LCD que se ocupará para la realización del proyecto es el comúnmente utilizado LCD 16x2. Para poder tener un mejor entendimiento del mismo se presenta a continuación en la Figura 7 los pines que forman parte del LCD y de los cuales se hacen uso para poder utilizarlo.

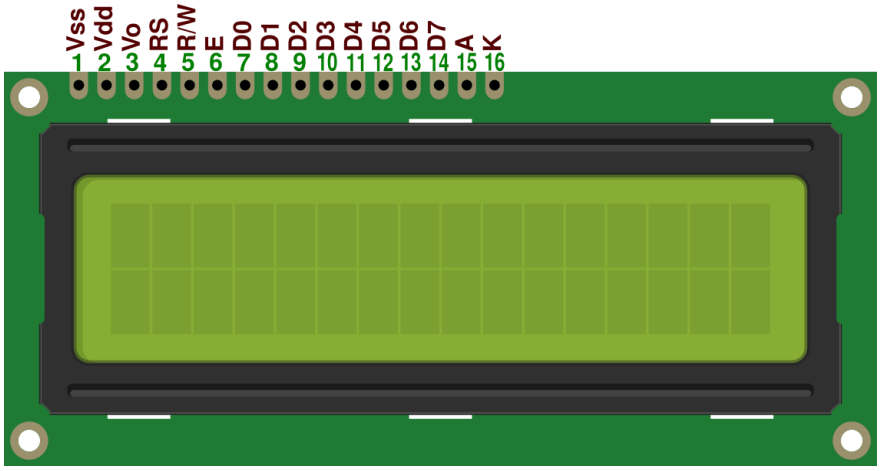


Figura 7: Pines que conforman el LCD 16x2. [2]

A continuación, se detalla la función de cada uno de los pines del LCD 16X2:

- **VSS:** Pin utilizado para conexión a masa, tierra o GND.
- **VDD:** Es por donde se realiza la alimentación equivalente a 5VDC de la pantalla y el chip con el que cuenta el mismo, es recomendable realizar esta conexión con una resistencia en serie para precautelar la integridad del LCD, evitando daños.
- **VO:** Este pin de conexión servirá para la función de contraste, el pin debe ser conectado a un potenciómetro de 10 kohmios que permite la regulación de lo que se presentará en la pantalla, si no se utiliza el potenciómetro no se tendrá visualización alguna de lo que se requiera indicar en el LCD.
- **RS:** Selector de registro (el microcontrolador le informa al LCD si lo que se necesita es mostrar caracteres o la opción de enviar comandos de control, como por ejemplo borrar la pantalla o el cambio de posición del cursor).
- **RW:** Este pin es el que comanda la lectura/escritura. Este se conectará a GND con el objetivo de poder escribir en todo momento.
- **E:** Pin de habilitación de la pantalla (Enable) para poder recibir información.
- **D0 a D7:** Bus de datos de 8 bits, usualmente se utilizan los pines que van del D4 a D7, útiles para establecer comunicación por donde se transportarán los datos.
- **A y K:** Pines que corresponden a la luz de fondo de la pantalla LCD, el pin con la letra A deberá conectarse a una tensión de 5VDC mediante una resistencia en serie (220 ohmios) y el pin con la letra K deberá ser conectado a masa o GND.

2.6 Pulsadores y LEDS

Para la activación de las diferentes acciones que serán contempladas por la banda transportadora se hará uso de pulsadores normales como los que se indica en la Figura 8.

Figura 8: Pulsador a utilizar con Arduino

Mediante la ejecución de una acción respectiva tendrá como respuesta una indicación luminosa proporcionada por un diodo LED como el que se muestra en la Figura 9.

Figura 9: Diodos LED

2.7 Sensor fotoeléctrico OMRON

El sensor fotoeléctrico a utilizar para este trabajo es el E3JK-R4M de marca OMRON el mismo se lo puede observar en la Figura 10, este sensor se encuentra incorporado con la cinta transportadora para poder realizar funciones de detección de objetos y/o piezas que se encuentren sobre la misma.

Sus dimensiones son reducidas (50x50x17,4mm) con el fin de aprovechar mejor el espacio donde el mismo va a ser ocupado, tiene salidas relé con una alta capacidad de conmutación (3 Amperios, 250 VAC). Este sensor es de tipo retroreflectivo polarizado que ayuda a la detección de objetos que pueden ser de cuerpo brillante. Posee un led de color rojo que se enciende cada vez que un objeto es detectado por el mismo.

Figura 10: Sensor fotoeléctrico E3JK-R4M de marca OMRON

2.7.1 Cables de conexión de sensor fotoeléctrico

Cuenta con cables que presentan diferentes colores, los mismo se especifican como sigue a continuación:

- **Marrón (BN) y Azul (BU):** Estos dos cables son utilizados para realizar la conexión de alimentación del sensor entre 12 a 24 VDC.
- **Blanco (WH):** Terminal de salida común para la activación de relé.
- **Negro (BK):** Terminal de salida Relé NA (Normalmente Abierto).
- **Gris (GY):** Terminal de salida Relé NC (Normalmente Cerrado).

2.8 Banda transportadora con elementos integrados

La banda transportadora de la cual se hace uso para la elaboración de este proyecto es igual a la que se muestra en la Figura 11, la misma posee acoplada a su estructura el motor de 24VDC, el calibrador digital, el encoder incremental y el sensor fotoeléctrico.

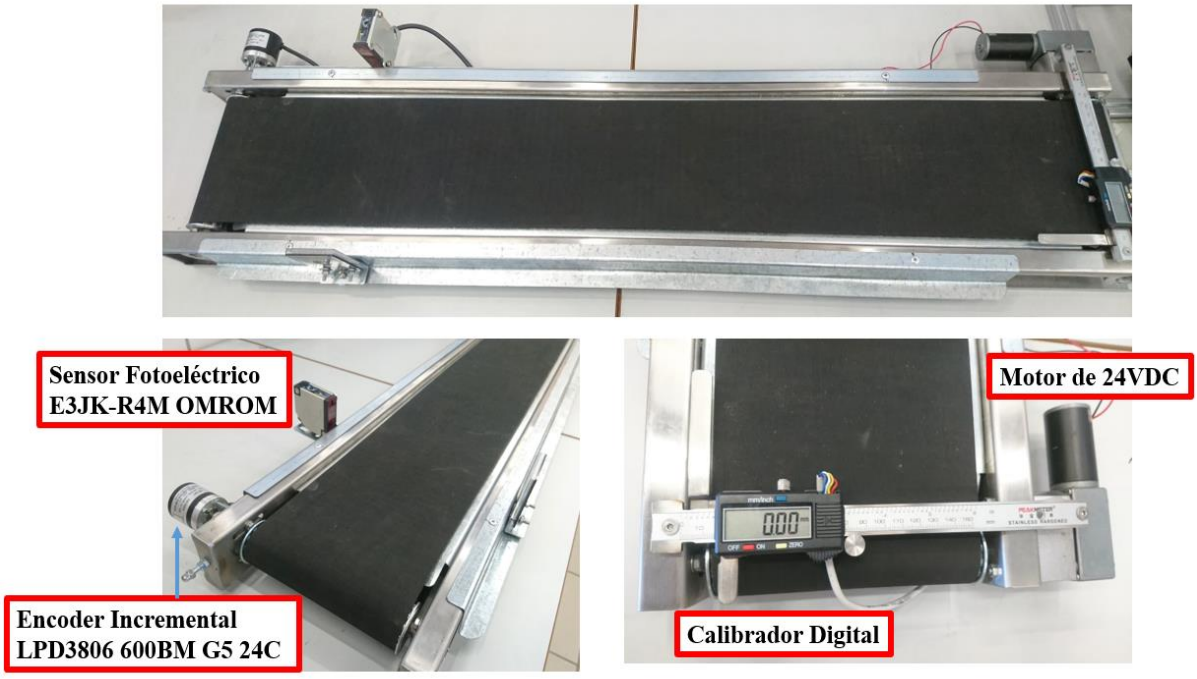


Figura 11: Estructura de la banda transportadora

3 INTERFAZ DE USUARIO

En este capítulo se trata de describir como debe interactuar el usuario con el sistema, considerando las funciones con las que cuenta el mismo, se pone a disposición una botonera con la que podrá manipular la cinta transportadora a su conveniencia realizando movimientos de avance o retroceso dependiendo del pulsador que haya sido activado, el operador debe conocer bien la función que desempeña cada botón y con ello realizar las acciones respectivas de manera eficiente. Adicionalmente a esto se contempla un modo de uso de manera remota que haría accionar a la cinta dependiendo de las acciones programadas por el robot ABB y que serán enviadas al microcontrolador del sistema para posicionar a las piezas donde se considere necesario.

3.1 Croquis de la Botonera

El operador tiene botones (pulsadores) e interruptores (selectores) para operación del sistema distribuidos tal y como se puede observar en la Figura 12, estos deben ser accionados para producir movimientos en la cinta transportadora. La posición considerada como X es la que se realiza mediante el calibrador digital dispuesto a lo ancho de la cinta transportadora y la posición considerada como Y representa el largo de la cinta transportadora por la cual la pieza se movilizará. A continuación, se detalla de manera más clara cada una de las acciones que se puede efectuar con los pulsadores e interruptores en este proyecto.

- **Interruptor Auto/Manual:** Este interruptor permite la transición entre el modo de control automático y control de manera manual. El control automático es realizado por el microcontrolador, el mismo que al trabajar en conjunto con botones, interruptores y demás componentes electrónicos, posibilita las acciones pertinentes que realizará la cinta transportadora para el posicionamiento de piezas. El modo de control manual no cuenta con la posibilidad de uso del microcontrolador, ya sea porque la misma haya sido retirada, se haya quemado o deje de funcionar, la cinta transportadora para este caso se manipulará con los botones de avance y retroceso presentes en la botonera.
- **Interruptor Local/Remoto:** Manipulando este interruptor se podrá trabajar ya sea de manera local o de manera remota. El modo de control local es el que se realizará mediante la utilización de la botonera del sistema. El modo de control de manera remota no hace uso de los botones del sistema, actúa conforme a las acciones dictadas por la lógica que contemple el robot para de esta manera interactuar con la cinta transportadora.
- **Enter:** Este botón tiene la función de permitir el ingreso en los menús del sistema y también sirve para realizar la confirmación de avances o retrocesos de la cinta transportadora.
- **Escape:** Si se acciona este botón tendrá como resultado la salida de los menús que hayan sido desplegados, hasta llegar al menú principal.
- **Botón Avanzar (▲):** Permite desplazarse por el menú del sistema y también ingresar valores de avance de la cinta transportadora.
- **Botón Retroceder (▼):** Permite desplazarse por el menú del sistema y también ingresar valores de retroceso de la cinta transportadora.
- **Emer:** Este botón representa a la seta de emergencia del sistema, se accionará cuando ocurra anomalías que atenten a la seguridad del usuario, el botón de emergencia produce el corte de la alimentación que acciona al motor de la cinta transportadora, al ser accionada la seta de emergencia producirá por tal motivo la detención por completo de la cinta, ayudando con esto a prever cualquier evento inesperado que suceda durante las acciones con el sistema en funcionamiento

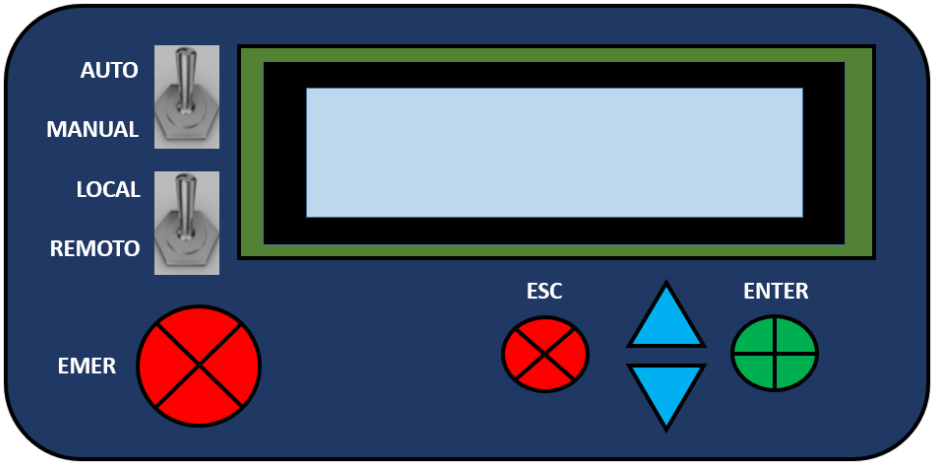


Figura 12: Botonera a disposición del usuario del sistema

3.2 Modos de configuración e interacción con los botones del sistema

El operador del sistema tiene a su disposición un indicador LCD en donde se irán presentando los diferentes menús dependiendo de la configuración que se quiera hacer. Cuando el sistema esté disponible para ser utilizado se visualizará inicialmente un menú principal en donde se podrá escoger mediante los botones avanzar **(▲)** o retroceder **(▼)** el tipo de avance que se quiere hacer.

El sistema cuenta con dos tipos de avance, los mismos que se describirán a continuación:

1. **Avance a una Posición Absoluta:** Este tipo de avance sitúa a la pieza dependiendo de la medida que el usuario especifique en el eje Y, teniendo en cuenta la posición donde se encuentra la pieza actualmente.
2. **Avance Discreto:** El modo discreto está pensando para realizar el posicionamiento de la pieza tomando en cuenta el valor de avance que el usuario quiere desplazar continuamente a la pieza dentro del eje Y, este tipo de avance es utilizado en la industria para facilitar el trabajo en cadenas de procesos que requieren acciones cada cierto intervalo de medida de la banda transportadora, posibilitando así una secuencia de pasos dentro de la elaboración de un producto final.

Al seleccionar mediante el botón ENTER ya sea Avance Absoluto o Avance Discreto, el LCD presenta un submenú en donde se visualizará la posición de la pieza. Se puede regresar al menú principal presionando el botón ESC, con ello se podrá escoger entre los dos tipos de avances presionando nuevamente el botón ENTER. Cabe aclarar que los botones tienen funciones diferentes si se encuentran en un menú o un submenú. La Figura 13 presenta como interactuar con los botones del sistema para entrar en un menú o salir del mismo, además la acción respectiva dependiendo el modo en que se encuentre.

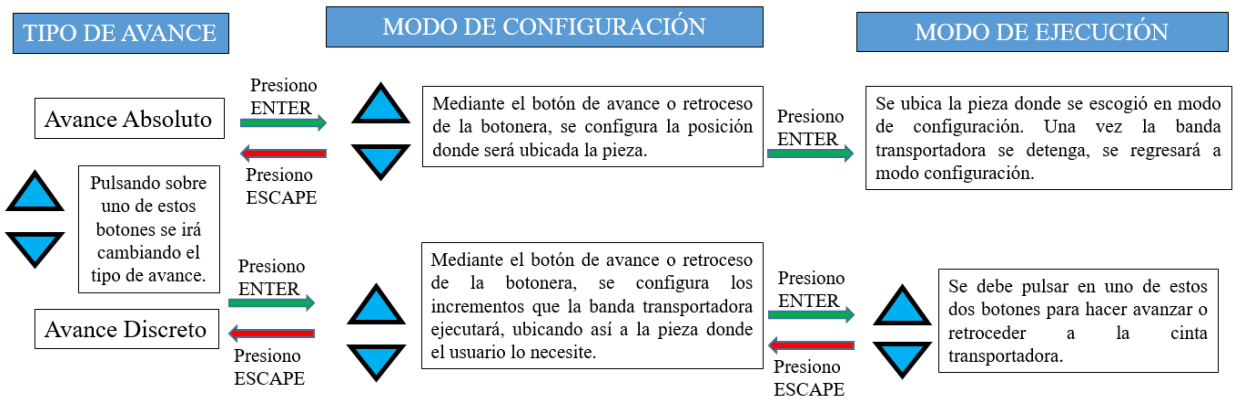


Figura 13: Modo de funcionamiento según el tipo de avance

3.3 Modo de operación del usuario con el sistema

La persona que manipule el sistema deberá colocar de manera manual la pieza sobre la banda transportadora y al inicio de la misma, es decir en el eje X igual cero y en el eje Y igual a cero (Figura 14), dicha posición es donde se encuentra fijo un calibrador digital dispuesto de tal manera que el usuario pueda manipularlo para dejar la pieza ubicada en el eje X (Figura 15), teniendo como siguiente acción a ejecutar la pulsación de los botones. Hay que considerar y tener muy en cuenta que la acción con los botones debe realizarse siempre que la banda se encuentre detenida.

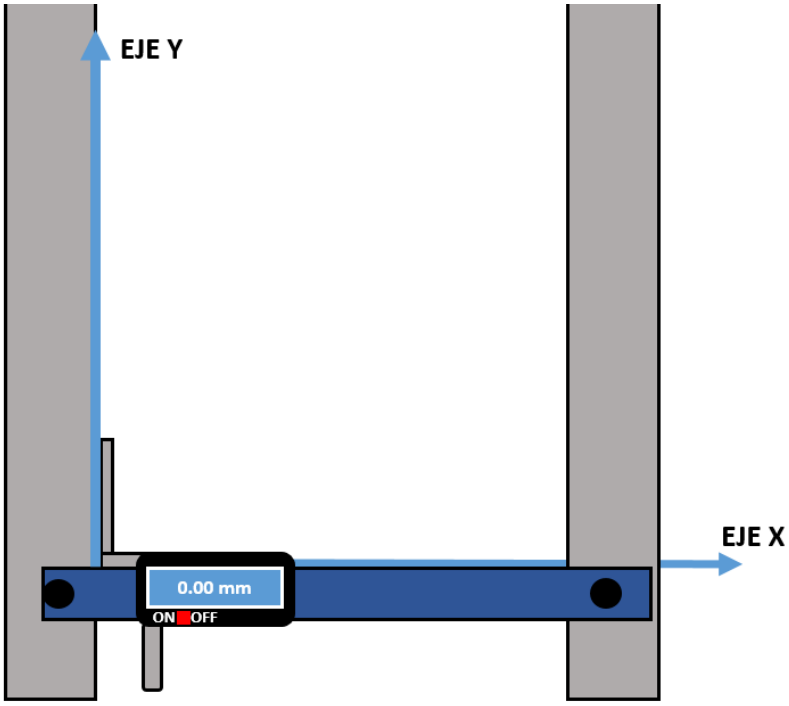


Figura 14: Posición origen del calibrador digital, eje X cero y eje Y cero

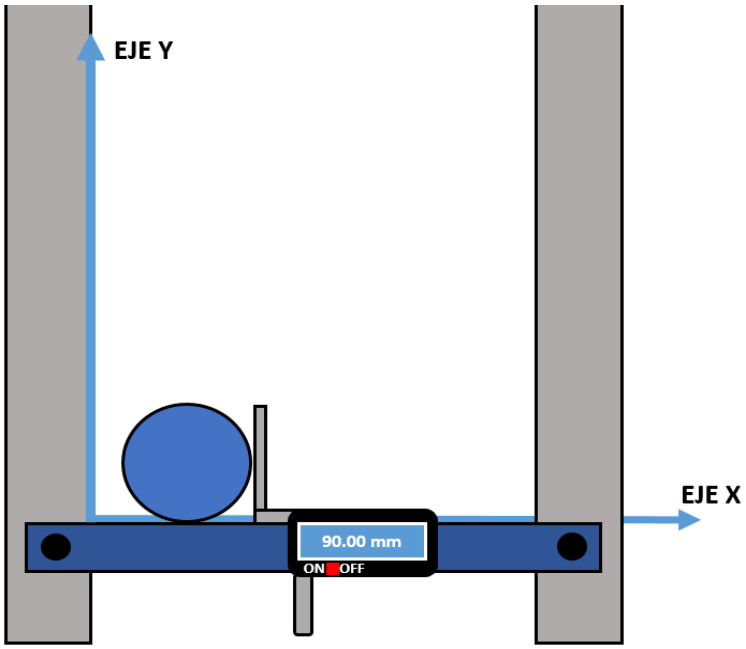


Figura 15: Pieza ubicada en el calibrador, eje X 90mm y eje Y 0mm

La pieza al movilizarse por la banda transportadora quedará por encima de la medida que se haya indicado para el posicionamiento en el eje Y, por ejemplo, para una medida indicada en Y de 200mm se obtendría lo indicado en la Figura 16.

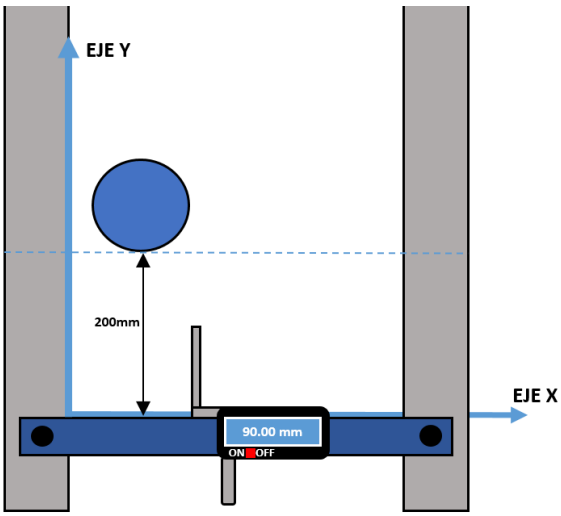


Figura 16: Pieza ubicada en la banda transportadora, eje X 90mm y eje Y 200mm

3.3.1 Respuesta del sistema estando en modo de Avance Absoluto

Una vez escogido el modo de avance absoluto, la presentación del mensaje en el LCD sería igual al que se presenta en la Figura 17 (Posición Inicial), se indica la posición en X y en Y actual de la pieza, el usuario deberá pulsar los botones de avanzar o retroceder para con ello indicar al sistema donde ubicar a la pieza, estos cambios que el usuario realice serán observados en el LCD a la derecha de las letras PY (posición en Y), a continuación se presenta un ejemplo de uso para clarificar la acción y respuesta que vería el operador.

Si la pieza está ubicada al inicio de la banda transportadora, el usuario moverá el calibrador digital ubicando en el eje X por ejemplo 75mm y en eje Y al estar en el inicio de la banda transportadora tendría 0mm, como acción siguiente el usuario debería pulsar los botones de avanzar **(▲)** o retroceder **(▼)** para indicar un valor al que deberá llegar la pieza, por ejemplo, un valor en PY de 300mm, para confirmar la ejecución de la banda transportadora se hará una pulsación sobre enter, que hará mover a la banda transportadora hasta llegar al punto de destino como indica la Figura 17.

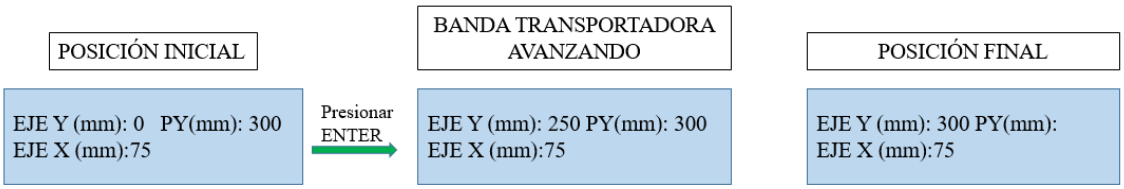


Figura 17: Avance absoluto, posición inicial en Y de 0mm, posición final en Y de 300mm

Una vez ubicada la pieza en la posición que se indicó en Y se tiene la opción de hacer avanzar o retroceder a la misma, cada vez que la pieza llegue a su posición final el valor en PY se borra en el LCD para escoger un nuevo valor, para este ejemplo se hará retroceder a la pieza hasta un valor de 200mm (Figura 18), y luego se la hará avanzar hasta los 500mm (Figura 19).

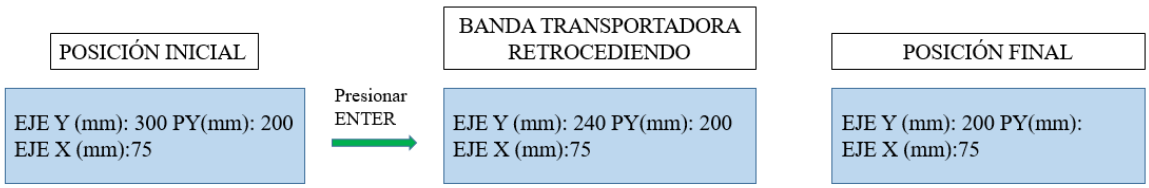


Figura 18: Avance absoluto, posición inicial en Y de 300mm, posición final en Y de 200mm

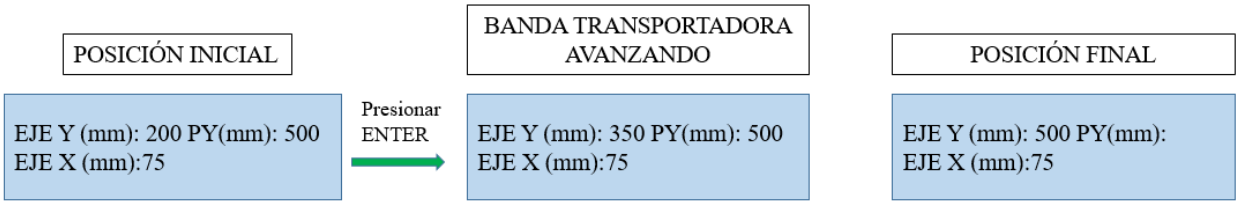


Figura 19: Avance absoluto, posición inicial en Y de 200mm, posición final en Y de 500mm

3.3.2 Respuesta del sistema estando en modo de Avance Discreto

Al estar dentro de este modo el operador del sistema deberá pulsar sobre avanzar **(▲)** o retroceder **(▼)** para elegir los incrementos que se realizarán, esto cambios en la LCD se visualizarán a la derecha de las letras IY(Incrementos en Y), para este ejemplo usaremos una posición inicial en Y de 0mm, posición en X de 80mm, con un incremento inicial de 15mm. La acción siguiente será pulsar el botón enter que permitirá pasar al modo de ejecución estando es este modo los botones avanzar **(▲)** o retroceder **(▼)** no producirán cambios en IY lo que realizarán ahora será la función de habilitación para que la banda transportadora empiece a ir hacia adelante o hacia atrás respectivamente, para este ejemplo al pulsar avanzar provocará que la pieza llegue a su posición final de 15mm, todo esto como se indica en la Figura 20. Una vez aquí de ser el caso se puede pulsar sobre avanzar nuevamente, lo que haría llegar a la pieza a la posición de 30mm (Figura 21), si pulso una vez más sobre avanzar la pieza llegaría hasta los 45mm en Y (Figura 22).

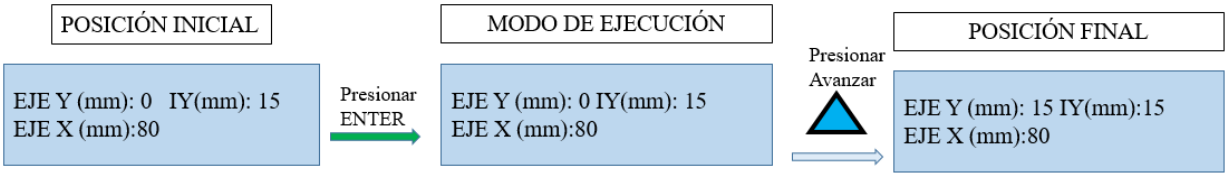


Figura 20: Avance discreto, posición inicial en Y de 0mm, posición final en Y de 15mm

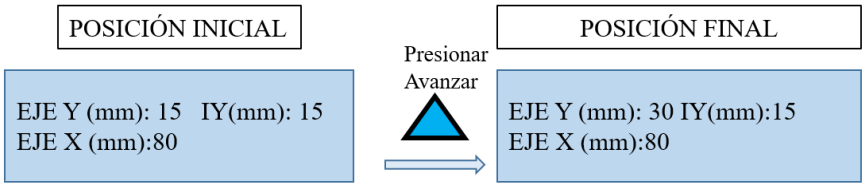


Figura 21: Avance discreto, posición inicial en Y de 15mm, posición final en Y de 30mm

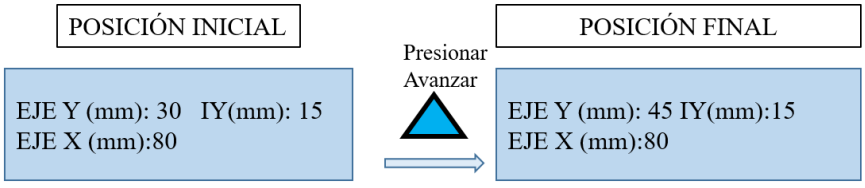


Figura 22: Avance discreto, posición inicial en Y de 30mm, posición final en Y de 45mm

Si el usuario en este caso de ejemplo estando en la posición en Y de 45mm hiciera una pulsación con el botón retroceder **(▼)** llevaría a la pieza a ubicarse en la posición final en Y de 30mm, como se muestra en la *Figura 23*.

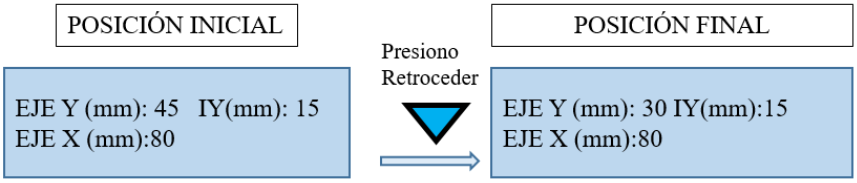


Figura 23: Avance discreto, posición inicial en Y de 45mm, posición final en Y de 30mm

Si el operador requiere de un incremento diferente a 15mm, lo que debería hacer es pulsar el botón de escape, que llevaría al sistema a que regrese al modo de configuración, una vez en este modo se tiene la opción de usar los botones de avanzar o retroceder para establecer un nuevo valor de incremento, para este ejemplo se estableció un valor en IY de 20mm, que al pasar al modo de ejecución mediante la pulsación sobre enter y que al pulsar sobre el botón de avanzar haría que la pieza llegue hasta la posición de 50mm, al presionar nuevamente sobre avanzar la pieza alcanzaría la posición de 70mm, y por último se ha presionado sobre el botón retroceder provocando que la pieza se posicione en 50mm, esto tal y como se indica en la *Figura 24*.

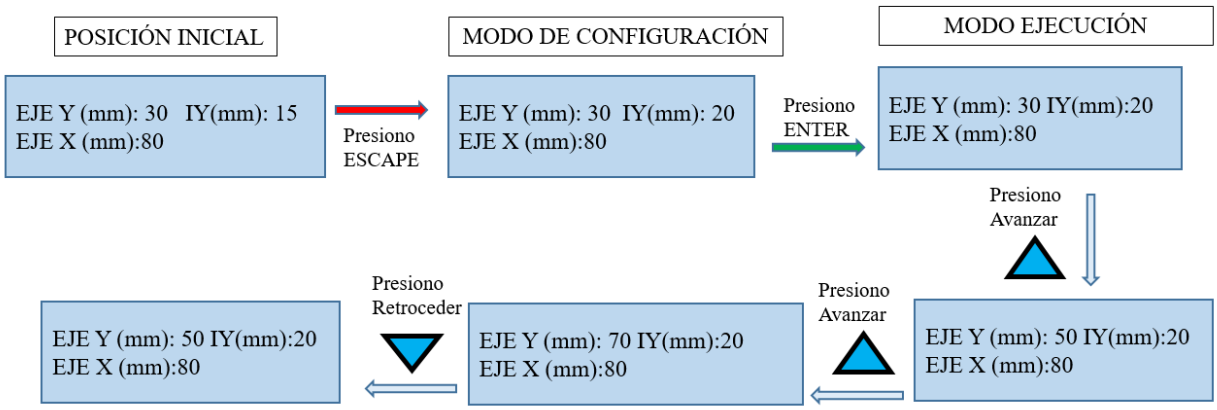


Figura 24: Avance discreto, cambio entre modos de configuración y ejecución

3.4 Modo de funcionamiento autónomo operación del robot ABB con el sistema

A continuación, se detallará las opciones que se tienen al utilizar el modo de posicionamiento de manera autónoma, el robot ABB dispondrá de entradas y salidas a 24VDC para manipular el accionamiento de la banda transportadora. Con el interruptor de la botonera activado para el cambio a modo de funcionamiento remoto los botones ENTER, ESCAPE, avanzar **(▲)** y retroceder **(▼)** quedarán deshabilitados y no se podrá obtener respuesta alguna por parte del sistema, si se está en modo de funcionamiento local las señales enviadas serán únicamente leídas por el robot ABB.

La función en modo de operación remoto está pensada para que el robot ABB comande el accionamiento de la banda transportadora. El microcontrolador del sistema envía y recibe señales que facilitan la comunicación con el robot ABB, las mismas se detallan a continuación:

3.4.1 Señales de salida del microcontrolador hacia el robot ABB

- **Posición en X:** La posición en X será transmitida mediante señal de tipo PWM, el pulso entregado en alto representa la medida que será leída por el robot ABB, teniendo en cuenta que la posición en X puede variar de 0 a 100mm, a continuación, se indica en la Figura 25 a modo de ejemplo la transmisión de la posición 40mm en eje X, este valor escalado en tiempo representa a 800ms en alto y 1200 ms en bajo, teniendo en cuenta un período de 2000ms, dependiendo de la posición en X que se quiera enviar se tendrá mayor o menor tiempo de pulsación en alto.

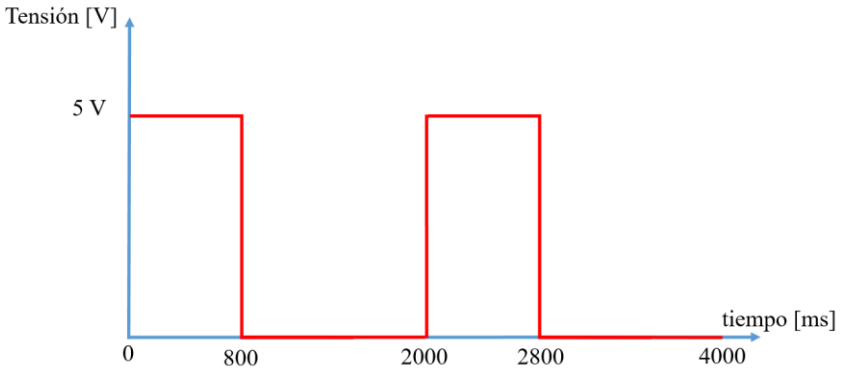


Figura 25: Envío de posición en X hacia el robot ABB mediante señal PWM

- **Posición en Y:** La posición en Y al igual que el envío de posición realizada en X, será transmitida mediante señal de tipo PWM, el pulso entregado en alto representa la medida que será leída por el robot ABB. Teniendo en cuenta que la posición en Y puede variar de 0 a 800mm, a continuación, se indica en la Figura 26 a modo de ejemplo la transmisión de la posición 600mm en eje Y, este valor escalado en tiempo representa a 1500ms en alto y 500 ms en bajo, teniendo en cuenta un período de 2000ms, dependiendo de la posición en Y que se quiera enviar se tendrá mayor o menor tiempo de pulsación en alto.

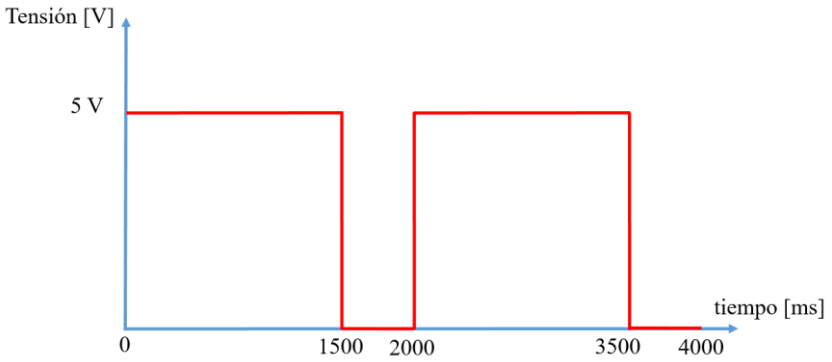


Figura 26: Envío de posición en Y hacia el robot ABB mediante señal PWM

- **E0 y E1:** Estos estados indican cambios producidos en el sistema mientras está en funcionamiento, representa una codificación conjunta a dos bits. Si E0=E1=0, éste estado indicaría que la banda está en reposo y se está configurando una orden. Si E0=0, E1=1, se haría referencia a que se está en ejecución de una orden. Si E0=1, E1=0, indicaría un fallo producido en el sistema. Si E0=1, E1=1, indicaría que se ha realizado un reseteo en la lectura del Encoder del sistema.
- **FO:** Indica la activación del sensor fotoeléctrico, cuando esta señal está a 1 el sensor está activo, si por el contrario se encuentra en 0 el sensor está desactivado.
- **Em:** Indica el accionamiento de la seta de emergencia, si esta señal se encuentra en 1 la seta de emergencia ha sido activada, y si este estado está en 0 la seta de emergencia no se encuentra activada.
- **Auto/Manual:** Esta señal muestra que se ha pasado del modo de funcionamiento automático a manual.
- **Local/ Remoto:** Con esta señal se reconocerá si el sistema está habilitado para trabajar de manera local o de manera remota.

3.4.2 Señales de entrada del microcontrolador desde el robot ABB

- **Referencia Y (REF Y):** Señal de tipo PWM que el robot ABB está enviando hacia el microcontrolador del sistema, pulsos en alto y bajo que representarán la medida de referencia en Y.

Esta señal se ocupará cuando se quiera trabajar en modo de avance absoluto.

- **ΔY:** Señal de tipo PWM que el robot ABB está enviando hacia el microcontrolador del sistema, pulsos en alto y bajo que representarán los incrementos que se requieren para avances en Y cuando se trabaje en modo de avance discreto.
- **Ejecución (EJEC):** Para el modo de avance absoluto, cada flanco de subida por parte de EJEC haría posicionarse a la pieza donde se haya indicado en referencia en Y (REF Y). Si se necesita por el contrario del avance discreto se mantendrá REF Y en alto todo el tiempo y se codificará ΔY, cuando se reciba un flanco de subida en EJEC se ejecutará la orden de posicionamiento en modo discreto.
- **Avance:** Señal que provocaría un avance de la banda transportadora cuando se detecte en el microcontrolador una pulsación constante en alto.
- **Retroceso:** Señal que provocaría un retroceso de la banda transportadora cuando se detecte en el microcontrolador una pulsación constante en alto.

3.5 Accionamiento de sensor fotoeléctrico en el sistema

La banda transportadora tiene acoplado a su estructura un sensor que permitirá la detección de piezas que se encuentren frente a este, si el sensor se encuentra activo la banda transportadora se detendrá y no se podrá mover hasta que la pieza salga de la zona de detección del sensor fotoeléctrico. La posición a la que llegue finalmente la pieza servirá como referencia para que el robot ABB mediante programación realice tareas de recolección de objetos.

4 DESARROLLO

En este capítulo se indicarán los diferentes pasos realizados hasta poder llegar a obtener una interfaz terminada del proyecto. De las diferentes pruebas que se realizaron se puede mencionar, el uso de pantalla LCD, pulsadores de manipulación simple y pulsadores de manipulación con anti rebote, control de motor de corriente continua mediante un transistor TIP120, lectura de señales de encoder, captación de señales obtenidas del calibrador digital, lecturas de señal de sensor fotoeléctrico, manejo del driver L298N. Todas estas pruebas se irán detallando a continuación de manera individual y luego se indicará la integración de las mismas para obtener el modo de operación de posicionamiento de piezas manual y el modo de operación autónomo desde el robot.

4.1 Uso de pantalla LCD

Para empezar a utilizar la pantalla LCD se debe tener en cuenta las conexiones respectivas que debe tener el Arduino Mega 2560 con el LCD, para lo cual se indica a continuación en la Figura 27 la manera correcta de conexión de la misma.

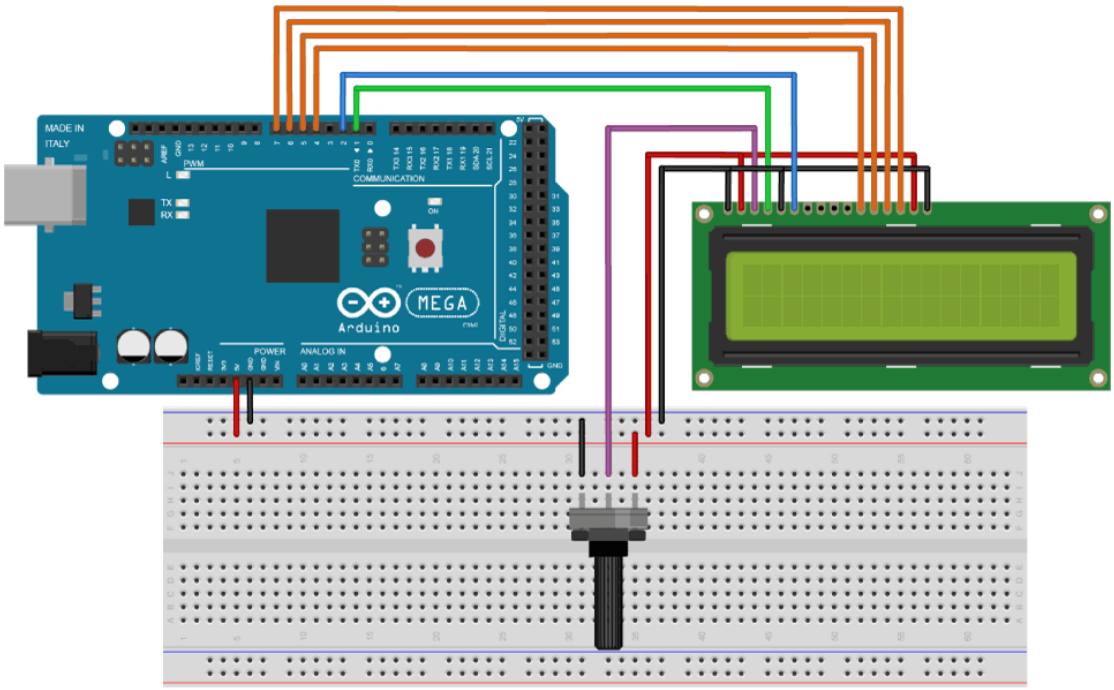


Figura 27: Conexionado de Arduino Mega 2560 con LCD 16x2. [2]

Luego de realizar esto lo que se hace es abrir el programa de arduino que debe estar previamente instalado en la computadora utilizada para realizar la programación respectiva del arduino Mega 2560, el programa se lo puede descargar de la página oficial de Arduino, el archivo de instalación no demanda de un espacio de memoria considerable y es muy intuitivo al momento de instalarlo, además si se llegara a tener alguna duda sobre como instalar el mismo se tienen infinidad de tutoriales en la red que indican la manera de hacerlo.

Mediante la utilización de las librerías con las que cuenta el software de arduino y los programas guía que se tienen como respaldo para empezar a realizar un programa, se puede detallar a continuación un programa simple de programación que presenta ciertos mensajes en la pantalla LCD, el mismo detalla en cada línea de su código la función que realiza en el programa.

```
#include <LiquidCrystal.h> //Inclusión de la librería para utilizar la LCD

//inicialización de la librería asociada
//Pines a ser conectados y utilizados de la LCD con el Arduino
const int rs = 1, en = 2, d4 = 4, d5 = 5, d6 = 6, d7 = 7;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {



//----------PANTALLA LCD----------------
  lcd.begin(16, 2);// Setear el número de columnas y filas de la LCD
  lcd.setCursor(0, 0);//posiciona el cursor en 0,0
  lcd.print("TFM ANDRES TAPIA");// Impresión de mensaje en la LCD
  delay(3000); // espera un tiempo de 3 sg
  lcd.clear(); // borra la pantalla
}

void loop() {
  lcd.setCursor(0, 1);// Setea el cursor en la columna cero, fila uno
  lcd.print("HOLA HOLA");//imprime un mensaje indicado entre las comillas
}
```

4.2 Pulsadores de manipulación simple, pulsadores de manipulación con anti rebote.

Para utilizar un pulsador con Arduino se debe tener en cuenta la conexión del mismo como se especifica en la Figura 28, en dicha figura también se incluye la conexión que se realiza para la utilización del indicador LED.

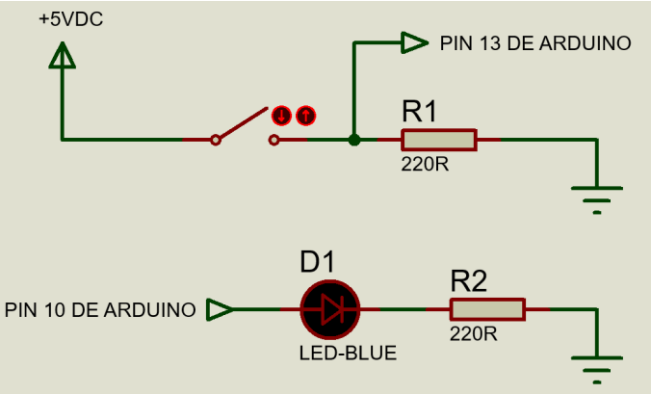


Figura 28: Conexión de un pulsador (input Arduino) y diodo led (output Arduino). [3]

El programa indicado a continuación representa al modo de utilización simple de un pulsador mediante la programación en Arduino.

```
//Este programa enciende el diodo LED mientras se esté pulsando el botón
//cuando se deja de pulsar el botón el diodo LED se apaga
int estado =0; //guarda el estado del botón
int BOTON= 13;// Variable declarada para utilizar con el botón
int LED= 10;//Variable declarada para hacer uso del LED

void setup(){
  Serial.begin(9600);//Velocidad de comunicación
  //BOTON Y LED
  pinMode(BOTON,INPUT);//Declaración de botón a utilizarse como una entrada
  pinMode(LED,OUTPUT);//Declaración de LED a utilizarse como una salida

}

void loop(){
 estado = digitalRead(BOTON);// Igualación de la variable estado con la lectura del botón
  if(estado == HIGH) {
  digitalWrite(LED, HIGH); //Encender el LED
 }
 else{
  digitalWrite(LED, LOW);//Apagar el LED
}
```

Este programa solo considera la activación del indicador LED cuando está pulsado el botón, y cuando el botón se deja de presionar, el LED indicador se apaga, ahora lo siguiente a realizar es un programa que detecte una pulsación para activar un diodo LED pero sin tener que estar presionando el botón todo el tiempo, aquí utilizamos un pulsador con anti rebote, el rebote se produce al momento de accionar el botón y es detectado por el arduino en tan solo milisegundos, quizá el cambio de estado bajo a alto es lo único que inicialmente se podría considerar, pero hay estados intermedios que se producen cuando se acciona un botón y que produciría estados erróneos al momento de querer activar una salida, para ello se hace un arreglo en programación que considera un estado actual y un estado anterior del botón, así como también un tiempo de retardo para evitar el rebote, todo esto se puede ver de manera más clara en el código siguiente:

```
int estado =0;    //guarda el estado del botón
int estadoAnterior=0;   //guarda el estado anterior del botón
int s=0;   //0=LED está apagado, 1= LED esta encendido

int BOTON= 13; // Variable declarada para utilizar con el botón
int LED= 10; //Variable declarada para hacer uso del LED
```

```
void setup(){
   Serial.begin(9600); //Velocidad de comunicación
  //BOTON Y LED
  pinMode(BOTON,INPUT); //Declaración de botón a utilizarse como una entrada
  pinMode(LED,OUTPUT); //Declaración de LED a utilizarse como una salida
}




void loop(){
 estado = digitalRead(BOTON);// Igualación de la variable estado con la lectura del botón


 //Comparación que permite saber el estado actual del botón
 if((estado == HIGH) && (estadoAnterior == LOW)){
  s=1-s;
  delay(20);
 }
 estadoAnterior = estado; //guarda el estado actual


 if (s == 1){
  digitalWrite(LED, HIGH); //ENDENDER EL LED
 }
 else{
  digitalWrite(LED, LOW);//APAGAR EL LED
 }
}
```

4.3 Control de motor de corriente continua mediante Arduino Mega 2560

Las conexiones respectivas para poder realizar el control del motor de corriente continua se las observa en la Figura 29, aquí se hace uso de un botón que al ser presionado hará funcionar al motor haciéndolo rotar y al mismo tiempo se encenderá un indicador LED, cuando el botón se pulse nuevamente provocará que el motor se detenga y el LED se apague, en el circuito de conexionado también se indica el uso del transistor NPN TIP120 que posibilita el control del motor, cabe mencionar que esta prueba de funcionamiento se realizó de manera inicial para luego pasar a controlar el motor de corriente continua mediante el driver L298N.

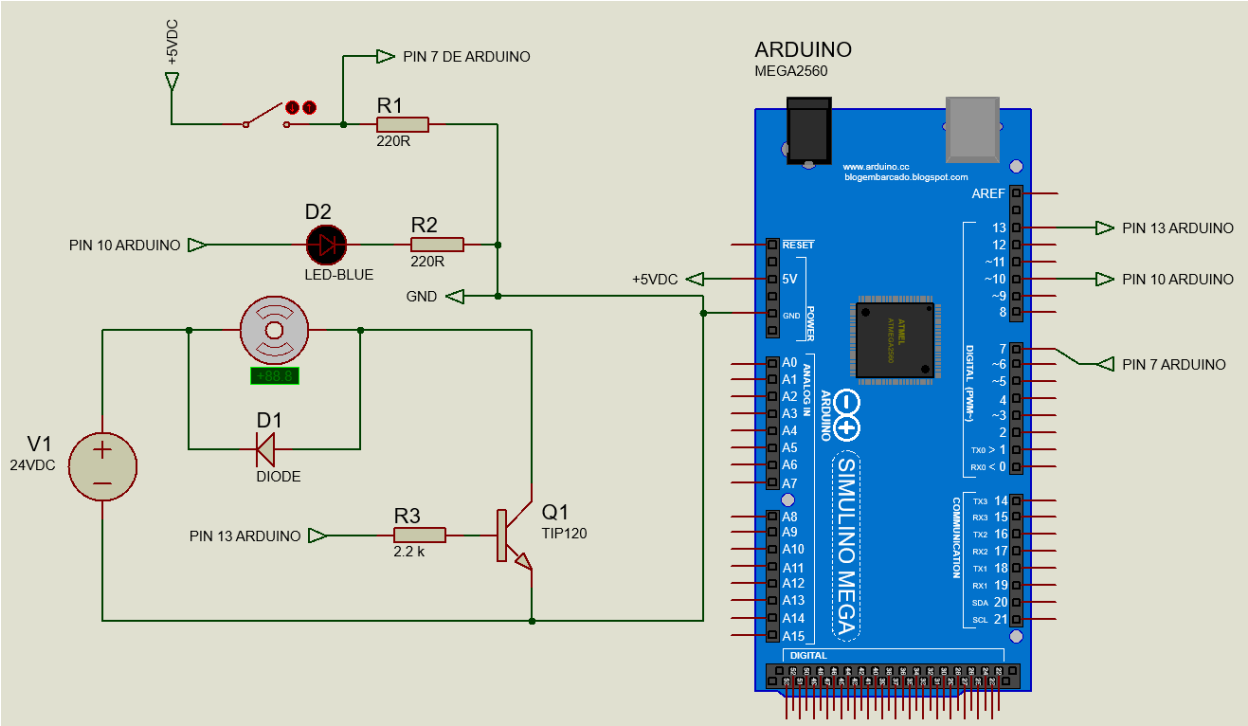


Figura 29: Conexionado de motor DC mediante uso de transistor TIP120. [3]

El código de programación utilizado para poder tener el control del motor de corriente continua mediante Arduino es el que se presenta en las siguientes líneas:

```
int estado =0;         //guarda el estado del botón
int estadoAnterior=0; //guarda el estado anterior del botón
int s=0;               //0=LED está apagado, 1= LED está encendido

int MOTOR=13;
int BOTON= 7;
int LED= 10;

void setup(){
  Serial.begin(9600);//Velocidad de comunicación
  //BOTON Y LED
  pinMode(BOTON,INPUT);//Declaración de botón como entrada
  pinMode(LED,OUTPUT);//Declaración de LED como salida

  //MOTOR
  pinMode(MOTOR, OUTPUT);//Declaración de Pin de conexión con motor como salida
  analogWrite(MOTOR, 0);// Inicialización en cero
}

void loop(){
 estado = digitalRead(BOTON); // Lectura del estado del botón
```

```
 if((estado == HIGH) && (estadoAnterior == LOW)){
  s=1-s;
  delay(20);
 }
 estadoAnterior = estado; //guarda el estado actual

 if (s == 1){
  digitalWrite(LED, HIGH); //Encender el LED
  analogWrite(MOTOR, 120);// Activación de Motor con salida PWM en valor igual a 120
 }
 else{
  digitalWrite(LED, LOW);//Apagar el LED
  analogWrite(MOTOR, 0);// Salida que detiene la rotación del motor
 }

}
```

4.4 Lectura de las señales de Encoder Rotativo Incremental

La estructura de la banda transportadora de cual se hace uso para la realización de este proyecto cuenta con un encoder incremental el mismo que gira al igual que lo hace el motor de corriente continua, si el motor se detiene el encoder también lo hará, teniendo esto en cuenta se puede hacer uso del mismo y sus conexiones respectivas serían iguales a las que se muestran en la Figura 30, el encoder dispone de cuatro cables que se diferencian por sus colores, el cable de color rojo será por donde se realice la conexión de alimentación de tensión igual a +5VDC la cual será proporcionada por el mismo arduino, el cable de color negro servirá para realizar la conexión a GND, los dos cables restantes representan a las señales con las que se trabajará para reconocer la posición en la que se encuentra el motor de corriente continua, el cable de color verde (Fase A) y cable de color blanco (Fase B) serán conectados a pines digitales del Arduino Mega 2560.

La programación usada para la lectura de los datos que proporciona el encoder y que se puede ver reflejada en el monitor Serie del programa de Arduino es indicada a continuación:

```
volatile unsigned int temp, counter = 0; //Variables usadas para el incremento o
decremento en la rotación del encoder


void setup() {
  Serial.begin (9600);//Velocidad de comunicación


  pinMode(2, INPUT_PULLUP); // Pin 2 como entrada, uso de pullup interno
  pinMode(3, INPUT_PULLUP); // Pin 3 como entrada, uso de pullup interno
```

```
  //Configuración de Interrupción
  //AttachInterrupt asigna la función que se ejecutará siempre que una interrupción externa se genere en el pin especificado
  attachInterrupt(0, ai0, RISING);// Uso de AttachInterrupt con 0 (pin2) para la función ai0, con modo de interrupción RISING
  attachInterrupt(1, ai1, RISING);// Uso de AttachInterrupt con 1(pin3) para la función ai1, con modo de interrupción RISING
  }

  void loop() {
  // Envío de valores al contador
  if( counter != temp ){
  Serial.println (counter);//Impresión serial de variable counter
  temp = counter;
  }
  }

  void ai0() {
  //ai0 es activada si el pin Digital 2 va de LOW a HIGH
  // Se verifica el pin3 para determinar la dirección
  if(digitalRead(3)==LOW) {
  counter++;
  }else{
  counter--;
  }
  }

  void ai1() {
  //ai1 es activada si el pin Digital 3 va de LOW a HIGH
  // Se verifica el pin2 para determinar la dirección
  if(digitalRead(2)==LOW) {
  counter--;
  }else{
  counter++;
  }
  }
```

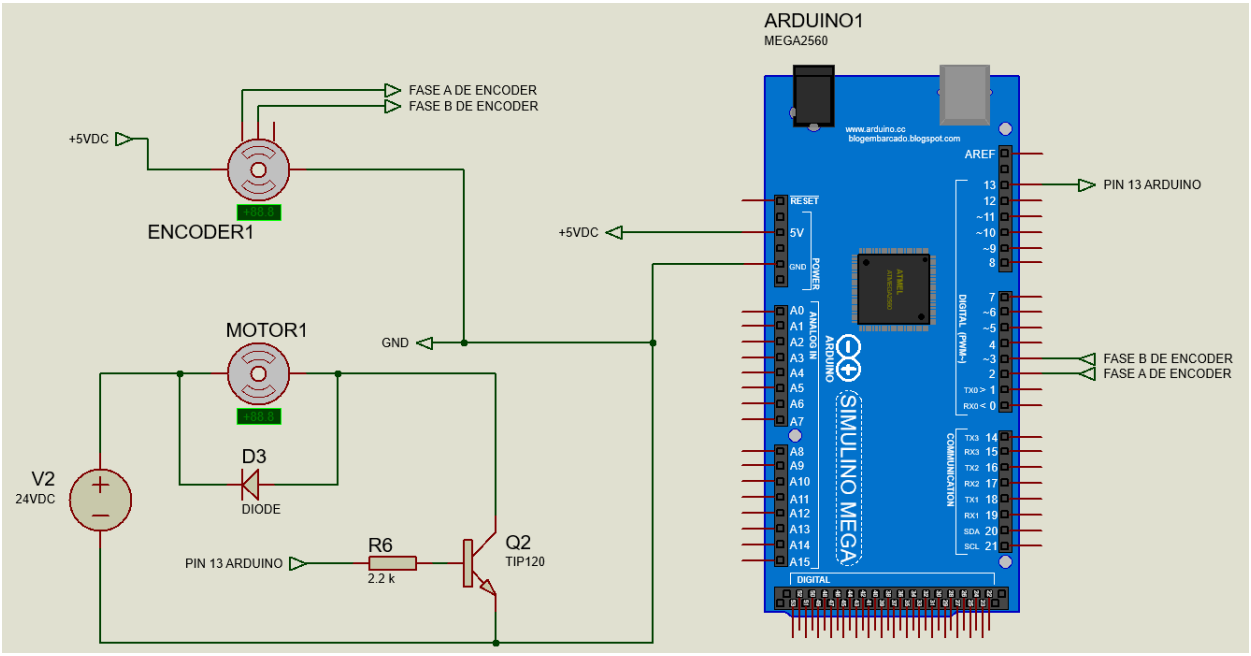


Figura 30: Conexión de motor de corriente continua con Encoder Rotativo Incremental. [3]

4.5 Lectura de señales obtenidas del calibrador digital

Lo primero a realizar para leer las señales que posee el calibrador (CLK y DATA) es la amplificación de las mismas ya que se requiere elevar su nivel lógico de 1.5VDC a 5VDC y así el arduino pueda detectarlas, esto se puede realizar de una manera fácil al hacer uso de dos resistencias de 10 Komhios y un transistor NPN 2N2222A para cada señal, tal y como se puede observar en el conexionado indicado en la Figura 31. Adicionalmente en este circuito se adiciona la utilización de una LDC 16x2 que servirá para ir presentando los datos que se van obteniendo del calibrador digital.

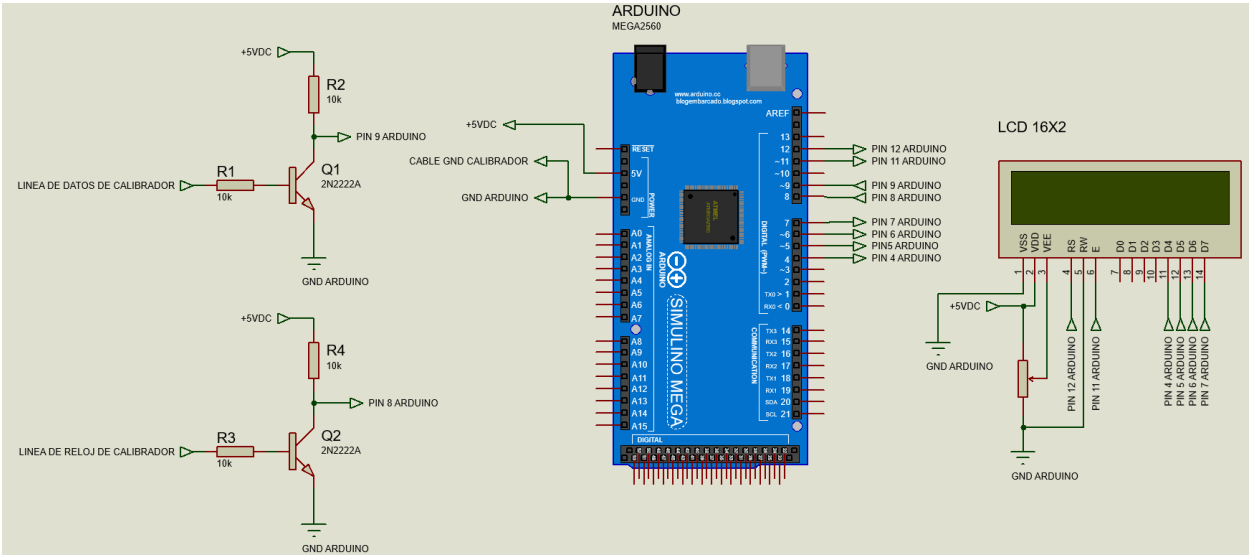


Figura 31: Circuito de conexión para obtener los datos provenientes del calibrador digital. [3]

Hay que mencionar que en la señal de datos los bits que son mayores a 21 se mantienen en bajo, y no considera los cambios que se producen en el calibrador cuando este se encuentra en una nueva posición, los bits que contienen los datos se van presentando en grupos de a cuatro por ejemplo del bit 0 al 3, del bit 4 al 7 y así consecutivamente hasta conformar un total de cinco grupos, dichos grupos son los que contienen la medida del calibrador el primer grupo representa la medida de 0.01mm, el segundo grupo corresponde a la medida de 0.1mm, el tercer grupo representa la medida de 1mm, y de esta manera hasta completar los cinco grupos que presentan la medición mediante el calibrador.

El código ocupado para la obtención de las señales del calibrador y presentación de la medición respectiva en una pantalla LCD 16x2 es el que se indica seguidamente:

```
// Librerías utilizadas para realizar comunicación I2C de LCD 16x2 con Arduino
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);   // establece la dirección de LCD 0x27 para 16 caracteres y 2 líneas de display


int i;//Variable usada para leer los 24 bits del calibre digital
int sign1;// Variable usada para detección de signo del dato 21 del calibrador
long value1;//
float result1;//Variable usada para presentar la medición en milímetros
int clockpin = 8;  //calibre1  8----Amarillo clock
int datapin = 9;   //calibre1  9----Blanco data


unsigned long tempmicros;// Variable usada contar tiempo en microsegundos de ejecución de sketch
void setup() {
  lcd.init(); //Inicialización de LDC
  lcd.backlight();//Encender Led de fondo de LCD I2C
  Serial.begin(9600);//Velocidad de comunicación
  pinMode(clockpin, INPUT);//Declaración de señal de reloj como entrada
  pinMode(datapin, INPUT);// Declaración de señal de dato como entrada


}


 void loop () {
  while (digitalRead(clockpin)==HIGH) {}//Si el reloj esta en bajo espera hasta que el reloj cambie a alto
  tempmicros=micros();//La función micros() devuelve el número de microsegundos desde que arduino empezó a ejecutar el sketch.
  while (digitalRead(clockpin)==LOW) {}//Espera el final del pulso en alto
  if ((micros()-tempmicros)>500) {//si el pulso en alto era más largo que 500 micros estamos al comienzo de la nueva secuencia de bits
     calibre1();
 }


 //escribe en LCD
 lcd.setCursor(0,0);// Posicionamiento de cursor en 0,0
 lcd.print("MEDIDA ");// Impresión de mensaje MEDIDA en LCD
 lcd.setCursor(9,0);//Posicionamiento de cursor en 9,0
```

```
 lcd.print(result1,2);//Presentación de resultado con dos decimales



}

void calibre1() {
  sign1=1;
  value1=0;
  for (i=0;i<23;i++) {
    while (digitalRead(clockpin)==HIGH) { }//espera hasta que el reloj regrese a alto
    while (digitalRead(clockpin)==LOW) {}//espera hasta que el reloj regrese a bajo
    if (digitalRead(datapin)==LOW) {
      if (i<20) {
        value1|= 1<<i; // desplazamientos a la izquierda hasta completar el resultado
de la medida
      }
      if (i==20) {
        sign1=-1;
      }
    }
  }
   result1=(value1*sign1)/100.00;
}
```

4.6 Lecturas de señal de sensor fotoeléctrico

Hay que tener en cuenta que el sensor fotoeléctrico representa a un interruptor que se va cerrando y abriendo conforme el mismo se active o desactive al momento de que un objeto este frente a este o no, el modo de utilización del sensor fotoeléctrico con el programa de Arduino es similar al visto anteriormente en el apartado 4.2 considerando para la conexión que donde iba conectado el interruptor ahora se conectará los cables de salidas a relé del sensor fotoeléctrico y los cables correspondientes a la alimentación que permitirá que se encienda el mismo.

4.7 Manejo del Driver L298N

Este módulo ayudará a realizar el control del motor de DC, en el apartado 4.3 se había realizado el control del motor utilizando un transistor TIP120, desde el punto de vista de eficiencia resulta mejor controlar el motor de corriente continua con el uso del driver L298N, al igual que aporta de protección adicional tanto para el motor, así como también a la tarjeta Arduino, al hacer uso de este módulo se tiene control sobre una señal de habilitación permitiendo realizar arreglos en programación para que el motor por ejemplo vaya activándose con diferentes velocidades si se requiere (señal PWM), y posibilita para que se pueda efectuar la inversión de giro del motor lo que no se puede hacer si se tiene conectado el motor como se estaba haciendo en el apartado 4.3, el módulo a ocupar será alimentado a través de una fuente externa de 24VDC, esta tensión permitirá el accionamiento del motor de corriente continua. El circuito de conexionado electrónico para la utilización con este módulo es igual al indicado en la Figura 32, a parte de las conexiones de tensión también se indica dónde

deben ir conectados los pines del arduino hacia el módulo y las respectivas salidas del módulo hacia el motor (OUT3 y OUT4). Se puede ocupar la salida OUT1 y OUT2 con la señal de habilitación ENA si se requiriera conectar otro motor.

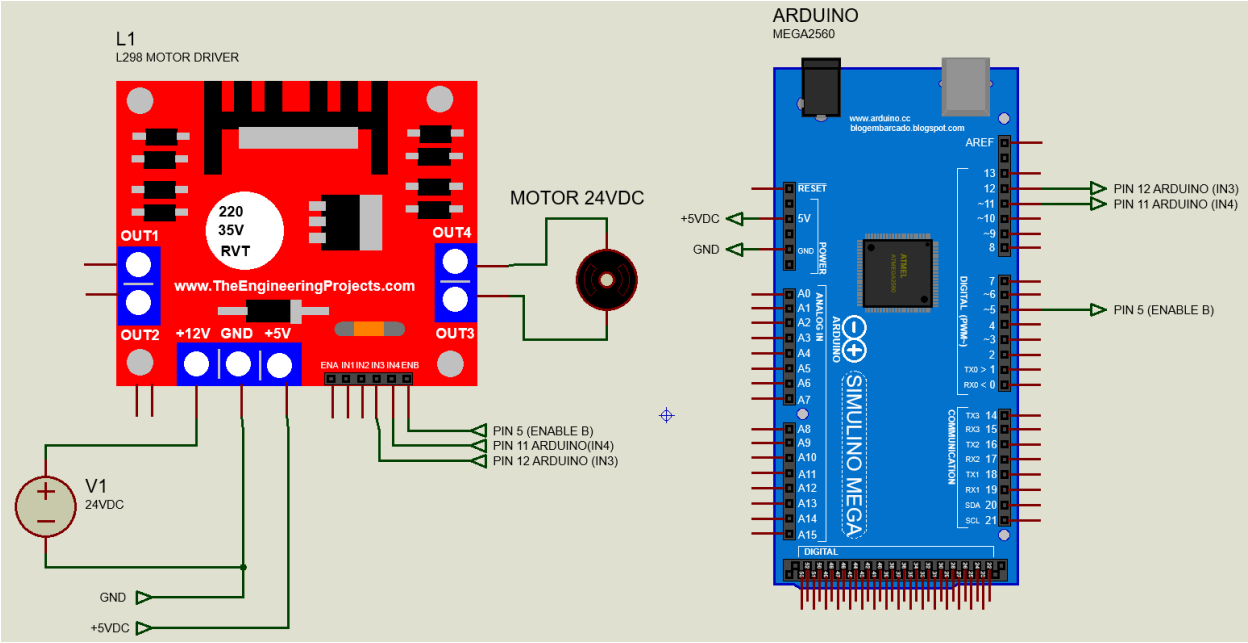


Figura 32: Circuito de conexión de conexión de driver L298N, Arduino Mega2560 y motor DC. [3]

La programación con la que se cargará al arduino para poder observar el funcionamiento del módulo con el motor es igual al que se presenta seguidamente, este programa produce que el motor vaya aumentando de velocidad cada dos segundos y se detenga por cinco segundos.

```
int IN3 = 12;     // Input3 conectada al pin 5
int IN4 = 11;     // Input4 conectada al pin 4
int ENB = 5;     // ENB conectada al pin 3 de Arduino
void setup()
{
 pinMode (ENB, OUTPUT);// Declaración de señal de habilitación ENB como salida
 pinMode (IN3, OUTPUT);//Declaración de IN3 como salida
 pinMode (IN4, OUTPUT);// Declaración de IN4 como salida
}
void loop()
{
  // Escritura en IN3 e IN4 para que el motor gire en un sentido
  digitalWrite (IN3, HIGH);
  digitalWrite (IN4, LOW);

  // Aplicamos PWM al pin ENB, haciendo girar el motor, cada 2 seg aumenta la velocidad
  analogWrite(ENB,80);//Valor 80 de PWM aplicado a ENB
  delay(2000);// tiempo de retardo por 2sg
  analogWrite(ENB,100);// Valor 100 de PWM aplicado a ENB
```

```
  delay(2000);//tiempo de retardo por 2sg
  analogWrite(ENB,150);//Valor 150 de PWM aplicado a ENB
  delay(2000);//tiempo de retardo por 2sg

  analogWrite(ENB,0);//Escritura en ENB que produce que el motor se detenga
  delay(5000);//retardo por 5sg
}
```

Después de realizar esta prueba para el uso de señal PWM con el motor, lo que se considera ahora será realizar un código que facilite la inversión de giro en el motor, dicho código se indica a continuación:

```
int IN3 = 12;    // Input3 conectada al pin 5
int IN4 = 11;    // Input4 conectada al pin 4
int ENB = 5;    // ENB conectada al pin 3 de Arduino



void setup()
{
 pinMode (ENB, OUTPUT); // Declaración de señal de habilitación ENB como salida
 pinMode (IN3, OUTPUT);//Declaración de IN3 como salida
 pinMode (IN4, OUTPUT);// Declaración de IN4 como salida
}
}

void loop()
{
  // Escritura en IN4 e IN3 para que el motor gire en un sentido
  digitalWrite (IN4, HIGH);
  digitalWrite (IN3, LOW);
  // Aplicamos PWM al pin ENB, haciendo girar el motor
  analogWrite(ENB,68);//Valor 68 de PWM aplicado a ENB
  delay(2000);// tiempo de retardo por 2sg
  analogWrite(ENB,0);//Escritura en ENB que produce que el motor se detenga
  delay(5000);// tiempo de retardo por 5sg



  //Motor girando en sentido contrario
  digitalWrite (IN4, LOW);
  digitalWrite (IN3, HIGH);
  analogWrite(ENB,68);//Valor 68 de PWM aplicado a ENB
```

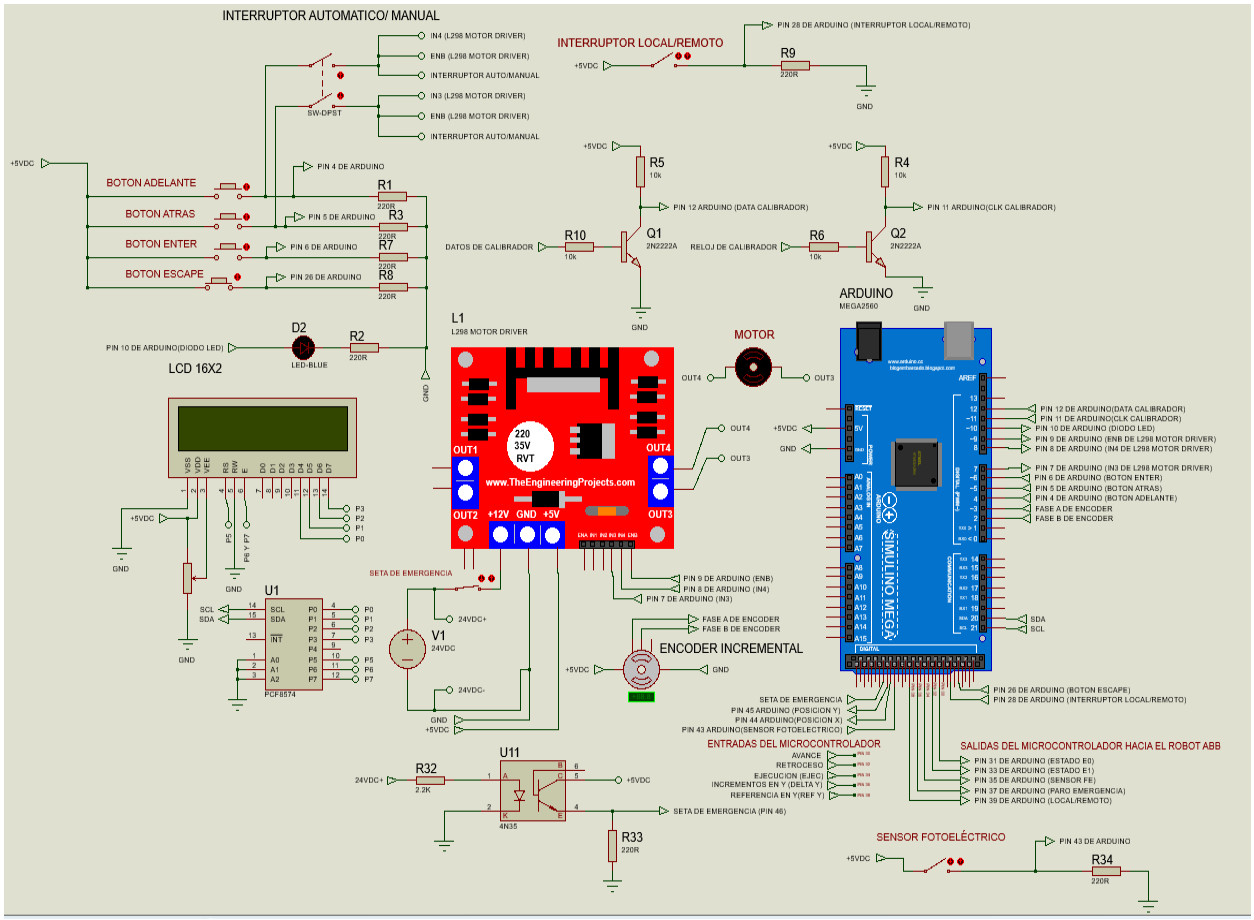
```
  delay(2000);// tiempo de retardo por 2sg
  analogWrite(ENB,0);//Escritura en ENB que produce que el motor se detenga
  delay(3000);// tiempo de retardo por 3sg
}
```

Después de haber realizado las anteriores pruebas de funcionamiento con los distintos elementos en hardware que se ocupa para este proyecto, lo que se realizará a continuación será trabajar en los diferentes modos de operación con los que debe contar la banda transportadora.

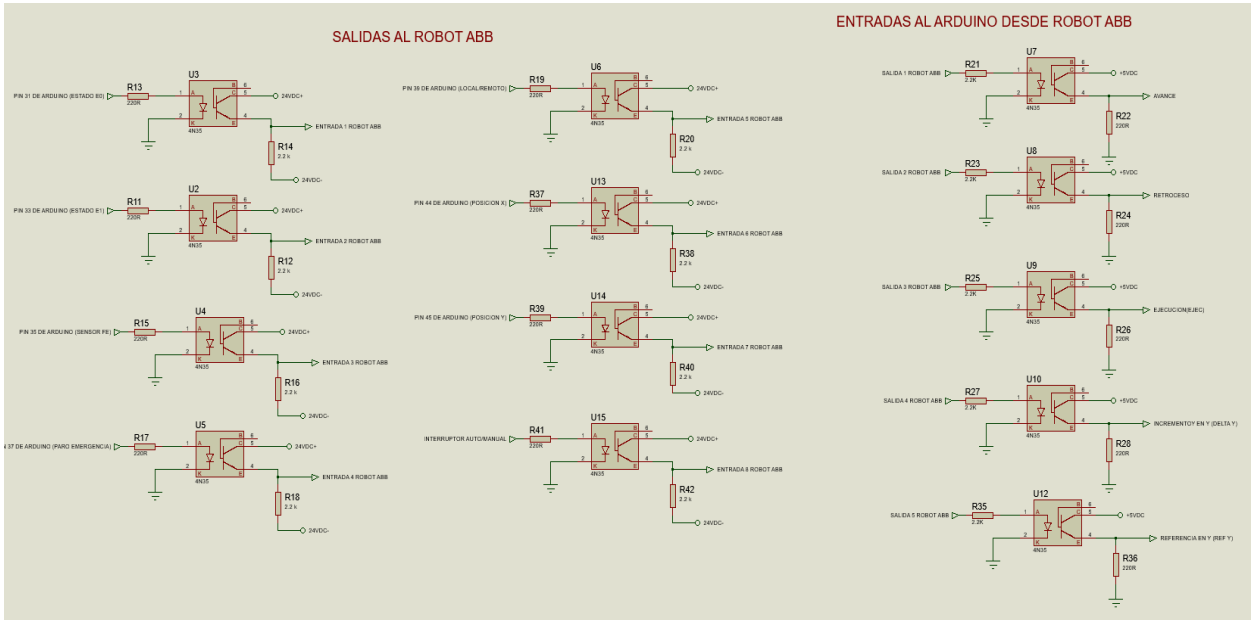
4.8 Sistema completo para posicionamiento de piezas de manera manual y autónoma

Las piezas a utilizar serán colocadas sobre la banda transportadora de manera manual por una persona, el usuario tiene la posibilidad de medir la posición en la que se encuentra la pieza con el uso del calibrador digital, esta posición será indicada como medida en X, el avance que se realice con la banda transportadora será indicada por su parte como la posición Y. Las medidas respectivas en ejes X e Y serán representadas mediante el uso de arduino Mega2560 a una LCD 16x2 con comunicación I2C.

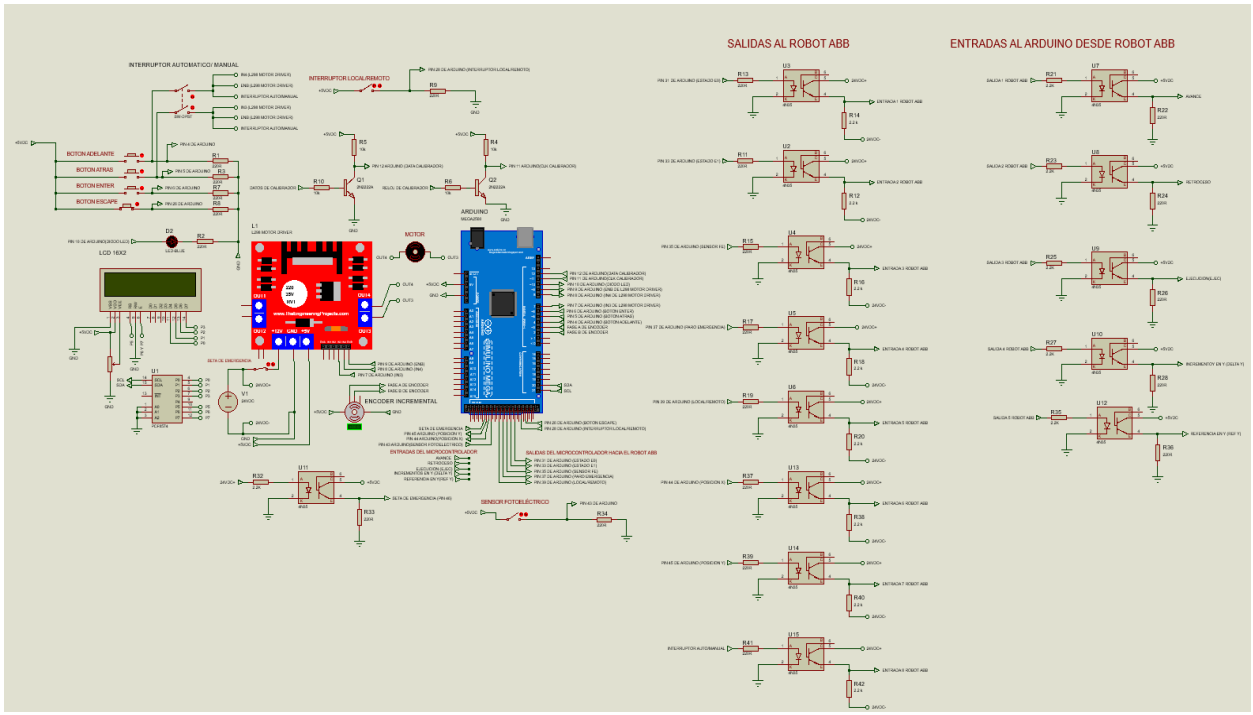
La conexión de los respectivos componentes electrónicos que son ocupados para este trabajo se indican tal y como se muestra en la Figura 33.



(a)



(b)



(c)

Figura 33: Circuito de elementos electrónicos del sistema de posicionamiento de piezas. a) Conexiones en el microcontrolador del sistema, b) Conexión de señales de microcontrolador a robot ABB, c) Diagrama completo de todos los elementos en el proyecto. [3]

El código de programación para la realización de este proyecto con los diferentes modos de funcionamiento es el que se indica a continuación:

```
// Librerías utilizadas para realizar comunicación I2C de LCD 16x2 con Arduino
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);// establece la dirección de LCD 0x27 para 16 caracteres y 2 líneas de display



volatile unsigned int temp, counter = 0; //Variables usadas para el incremento o decremento en la rotación del encoder
float ejey=0;//VARIABLE PARA MEDIR EN cm POSICION EN EJE Y


int bandera=0;
//Indicador Led del Sistema
int LED=10;


//VARIABLES PARA LECTURA DEL CALIBRADOR DIGITAL
int i;
int g;
int sign1;
long value1;
float result1;
int clockpin = 11;  //calibre1  pin 11, cable amarillo clock
int datapin = 12;   //calibre1  pin 12, cable blanco data
unsigned long tempmicros;

// VARIABLES PARA CONTROL DE MOTOR (MODULO L298)
int IN3 = 7;    // Input3 conectada al pin 5
int IN4 = 8;    // Input4 conectada al pin 4
int ENB = 9;    // ENB conectada al pin 3 de Arduino



//Pulsadores del sistema
const int pulsADELANTE = 4; //Pulsador adelante
const int pulsATRAS = 5;// Pulsador atrás
const int pulsENTER=6; //Pulsador Enter
const int pulsESCAPE=26;//Pulsador Escape
//Variables utilizadas para la habilitación de menús y submenús
int avance=0,retro=0,aux=0;
```

```
int confirmacion=0, aux2=0;

//INTERRUPTORES
int LOCAL;
int REMOTO;
int INTERRUPTOR1=28; //INTERRUPTOR PARA DEFINIR LOCAL O REMOTO

//SENSOR FOTOELÉCTRICO DE ENTRADA
int sensor=43;//Variable de sensor fotoeléctrico
int estadosensor;
int estadosensorAnterior;
int activacionsensor=0;



//Botón hacia adelante
const int tiempoAntirrebote = 10;// tiempo en ms para evitar el rebote
int cuenta = 0;
int estadoBoton;
int estadoBotonAnterior;
//PARA AVANCES DISCRETOS
int ESTADOADELANTE;
int ESTADOADELANTEANTERIOR;

//Botón hacia atrás
const int tiempoAntirrebote2 = 10;// tiempo en ms para evitar el rebote
int cuenta2 = 0;
int estadoBoton2;
int estadoBotonAnterior2;

//VARIABLES PARA AVANCES DISCRETOS
int ESTADOATRAS;
int ESTADOATRASANTERIOR;

//Botón ENTER
const int tiempoAntirrebote3 = 10;// tiempo en ms para evitar el rebote
int cuenta3 = 0;
int estadoBoton3;
int estadoBotonAnterior3;
```

```
 //Botón ESCAPE
const int tiempoAntirrebote4 = 10;// tiempo en ms para evitar el rebote
int cuenta4 = 0;
int estadoBoton4;
int estadoBotonAnterior4;



String Anuncio1;
String Anuncio2;

//Menús
void menu();
void AvanceAbsoluto();
void AvanceDiscreto();



//*************************************************************
//VARIABLES PARA CONTROL REMOTO (ENTRADAS RECIBIDAS DESDE PLC)
//*************************************************************
int avanceABB=30;//avance del posicionador por estado de salida de ABB
int retrocesoABB=32;//avance discreto por flanco de salida de ABB
//REFENCIAS EN Y DESDE ABB O PLC HACIA ARDUINO
int ejecucionABB=34; // VARIABLE PARA INDICAR LA EJECUCIÓN
int ejecucion=1;
int avancediscretoABB=36;//VARIABLE PARA REALIZAR AVANCE DISCRETO
int REFERENCIAY=38;//VARIABLE PARA DAR UNA REFERENCIA DE POSICION EN Y



//VARIABLES PARA LEER ms EN ALTO y ms EN BAJO QUE ME ENVIA EL ROBOT ABB
volatile unsigned int counter2 = 0; //PARA DETECCION DE PULSOS EN ALTO (REFERENCIA DE
Y)
volatile unsigned int counter3 = 0; //PARA DETECCION DE PULSOS EN BAJO (REFERENCIA DE
Y)


volatile unsigned int counter4 = 0; //PARA DETECCION DE PULSOS EN ALTO(INCREMENTOS EN
Y)
volatile unsigned int counter5 = 0; //PARA DETECCION DE PULSOS EN BAJO(INCREMENTOS EN
Y)


//TIEMPOS QUE VOY A LEER PARA REFERENCIA Y
unsigned long daletiempo;
```

```
unsigned long daletiempo2;
unsigned long diferenciaTiempo = 0;
long TALTOINT;
long TBAJOINT;
long PERIODOINT;
long TALTOBIEN;
long PERIODOBIEN;
int MEDIDA_RECIBIDA_REFY;


//TIEMPOS QUE VOY A LEER PARA INCREMENTOS Y (DELTAY)
unsigned long daletiempo3;
unsigned long daletiempo4;
unsigned long diferenciaTiempo2 = 0;
long TALTOINT2;
long TBAJOINT2;
long PERIODOINT2;
long TALTOBIEN2;
long PERIODOBIEN2;
int MEDIDA_RECIBIDA_INCREY;


//Variables para avance estado de salida de ABB
int estadosalidaABB1;
int estadosalidaABBAnterior1;

//Variables para retroceso por estado de salida de ABB
int estadosalidaABB2;
int estadosalidaABBAnterior2;

//Variables para REFERENCIA DE Y desde ABB
int estadosalidaABB3;
int estadosalidaABBAnterior3;

//Variable para AVANCES DISCRETOS desde ABB
int estadosalidaABB4;
int estadosalidaABBAnterior4;
//*************************************************************************
//VARIABLES PARA CONTROL REMOTO (SALIDAS ARDUINO(ENTRADAS A ROBOT ABB))
//*************************************************************************
```

```
const int POSICIONX=44;//ENVIO DE POSICION EN X
const int POSICIONY=45;//ENVIO DE POSICION EN Y
//ESTADOS DE E0, E1, SENSOR FOTOELECTRICO, SETA DE EMERGENCIA
int ESTADOE0=31;
int ESTADOE1=33;
int SENSORFE=35;
int PAROEMERGENCIA=37;
//ESTADOS DE ACTIVACION DE INTERRUPTORES
int LOCAL_REMOTO=39;
int AUTO_MANUAL=41;

//ENTRADA PARA CONOCER LA ACTIVACION DE SETA DE EMERGENCIA
int SETAEMERGENCIA=46;
//VARIABLES PARA REALIZAR AVANCES Y RETROCESOS

int av=0;
int avanterior=0;
int medidaavance=0;
int medidaavanceanterior;
int confirmacionadelante=0;
int confirmacionatras=0;
int adentroabsoluto=0;

//VARIABLES PARA ENVIAR TIEMPOS EN ALTO Y BAJO (MEDIDA EN Y)
int talto=0;
int tbajo=0;
boolean estadoLed = true;  //guarda el estado de encendido o apagado
unsigned long tiempoAnteriorEncendido = 0;  //guarda tiempo de referencia para comparar
unsigned long tiempoAnteriorApagado = 0;



//VARIABLES PARA ENVIAR TIEMPOS EN ALTO Y BAJO (MEDIDA EN Y)
int talto2=0;
int tbajo2=0;
boolean estadoLed2 = true;  //guarda el estado de encendido o apagado
unsigned long tiempoAnteriorEncendido2 = 0;    //guarda tiempo de referencia para
comparar
unsigned long tiempoAnteriorApagado2 = 0;
```

```
//VARIABLE PARA SABER SI LA BANDA ESTA EN REPOSO
float ejeyanterior;



void setup() {

  //------VELOCIDAD DE COMUNICACION
  Serial.begin (9600);

  //------INICIALIZACION LCD I2C
  lcd.init();//INICIALIZACION PARA USO DE LCD I2C
  lcd.backlight();//ENCEDER LED DE LCD I2C

  //------DECLARACION DE PULSADORES
  pinMode(pulsADELANTE, INPUT);
  pinMode(pulsATRAS, INPUT);
  pinMode(pulsENTER, INPUT);
  pinMode(pulsESCAPE, INPUT);

  //------INTERRUPTORES
  pinMode(INTERRUPTOR1, INPUT);



  //Declaración de pines como entradas de señales de ABB
  pinMode(avanceABB, INPUT);
  pinMode(retrocesoABB, INPUT);
  pinMode(ejecucionABB, INPUT);

  pinMode(avancediscretoABB, INPUT);
  pinMode(REFERENCIAY, INPUT);



  //DECLARACION DE PINES PARA SALIDA HACIA ABB
  pinMode(POSICIONX, OUTPUT);
  pinMode(POSICIONY,OUTPUT);
  pinMode(ESTADOE0, OUTPUT);
  pinMode(ESTADOE1,OUTPUT);
  pinMode(SENSORFE, OUTPUT);
  pinMode(PAROEMERGENCIA,OUTPUT);
  pinMode(AUTO_MANUAL, OUTPUT);
```

```
  pinMode(LOCAL_REMOTO,OUTPUT);



  //DECLARACION DE SENSOR COMO ENTRADA
  pinMode(sensor,INPUT);

  //PARA ACTIVACION DE SETA DE EMERGENCIA
  pinMode(SETAEMERGENCIA,INPUT);

  //------DECLARACION DE INDICADOR
  pinMode(LED,OUTPUT);//INICIALIZACION DE LED COMO SALIDA

  //DECLARACION DE DATOS PARA CALIBRADOR DIGITAL
  pinMode(clockpin, INPUT);//INICIALIZACION DE SEÑAL DE RELOJ DE CALIBRADOR1 COMO ENTRADA
  pinMode(datapin, INPUT);//INICIALIZACION DE SEÑAL DE DATOS DE CALIBRADOR1 COMO ENTRADA

  //INICIALIZACION DE PINES PARA CONTROL DE MOTOR
   pinMode (ENB, OUTPUT);//INICIALIZACION DE VARIABLE COMO SALIDA QUE SERVIRA DE ENABLE PARA PUENTEH L298
   pinMode (IN3, OUTPUT);//INICIALIZACION DE VARIABLE COMO SALIDA (ENTRADA IN3 EN PUENTEH L298)
   pinMode (IN4, OUTPUT);//INICIALIZACION DE VARIABLE COMO SALIDA (ENTRADA IN4 EN PUENTEH L298)

  //PINES PARA LECTURA DE ENCODER
  pinMode(2, INPUT_PULLUP);
  pinMode(3, INPUT_PULLUP);
  attachInterrupt(0, ai0, RISING);
  attachInterrupt(1, ai1, RISING);
}

  void loop() {

  /*
   //------LECTURA DE DATOS DEL CALIBRADOR DIGITAL
  while (digitalRead(clockpin)==HIGH) {}
  tempmicros=micros();
  while (digitalRead(clockpin)==LOW) {}
  if ((micros()-tempmicros)>500) {
```

```
  calibre1();
  }
    if (digitalRead(SETAEMERGENCIA)==0){
    digitalWrite(PAROEMERGENCIA,HIGH);
    } else{
    digitalWrite(PAROEMERGENCIA,LOW);
    }*/
//*****************************************************************************
    //------------------SALIDAS DEL ARDUINO(ENTRADAS ABB)-----------------------

//*****************************************************************************


   //***************************************************************************
    //ESCALADOS DE TIEMPO EN ALTO ms y BAJO EN ms PARA ENVIAR MEDIDA EN Y
    //***************************************************************************
    talto=map(ejey,0,80,0,2000);
    tbajo=2000-talto;
    //Serial.println(talto);


    if((millis()-tiempoAnteriorEncendido>=talto)&&estadoLed==true){         //si     ha
transcurrido el periodo programado
     estadoLed=false;  //actualizo la variable para poner en false
     digitalWrite(POSICIONY,estadoLed);  //Escribo en POSICIONY
     tiempoAnteriorApagado=millis();  //guarda el tiempo actual para comenzar a contar
el tiempo apagado
    }
    if((millis()-tiempoAnteriorApagado>=tbajo)&&estadoLed==false){          //si     ha
transcurrido el periodo programado
     estadoLed=true;  //actualizo la variable para encender el led
     digitalWrite(POSICIONY,estadoLed);  //enciendo el led
     tiempoAnteriorEncendido=millis();    //guarda  el  tiempo  actual  para  comenzar  a
contar el tiempo encendido
    }


    //*********************************************************************
    //ESCALADOS PARA ENVIA MEDIDA EN X DE TIEMPOS EN ALTO Y TIEMPOS EN BAJO
    //*********************************************************************
    talto2=map(result1,0,100,0,2000);
    tbajo2=2000-talto2;
    //Serial.println(talto2);
```

```
    if((millis()-tiempoAnteriorEncendido2>=talto2)&&estadoLed2==true){        //si    ha
transcurrido el periodo programado
     estadoLed2=false;  //actualizo la variable para poner en false
     digitalWrite(POSICIONX,estadoLed2);  //Escribo en POSICIONX
     tiempoAnteriorApagado2=millis();  //guarda el tiempo actual para comenzar a contar
el tiempo apagado
    }
    if((millis()-tiempoAnteriorApagado2>=tbajo2)&&estadoLed2==false){        //si    ha
transcurrido el periodo programado
     estadoLed2=true;  //actualizo la variable para encender el led
     digitalWrite(POSICIONX,estadoLed2);  //enciendo el led
     tiempoAnteriorEncendido2=millis();   //guarda  el  tiempo  actual  para  comenzar  a
contar el tiempo encendido
    }


 //************************************************************
 //------------------MODO DE CONTROL LOCAL------------------
 //************************************************************


  if (digitalRead(INTERRUPTOR1)==1)
  {
  //************************************
  //ACCIONES ESTANDO EN LOCAL
  //************************************
  digitalWrite(LOCAL_REMOTO,LOW);



  //--------LECTURAS DE POSCION DE ENCODER
  if( counter != temp ){
  temp = counter;
  //TRANSFORMACION DE BITS A centímetros
    ejey=map(counter,0,8260,0,80);
    //LIMITACIÓN DE ENCODER PARA NO TENER MEDIDAS ERRONEAS
    if (counter > 20000){
    counter=0;
    }
  }
  //-----------BOTÓN ADELANTE
  estadoBoton = digitalRead(pulsADELANTE);
  if(estadoBoton != estadoBotonAnterior){
    if(antirrebote(pulsADELANTE)){
```

```
      if (aux==0){
      cuenta++;
      }else{
       avance++;
      }
    }
  }
  estadoBotonAnterior = estadoBoton;

  //-----------BOTÓN ATRAS
   estadoBoton2 = digitalRead(pulsATRAS);
   if(estadoBoton2 != estadoBotonAnterior2){
     if(antirrebote2(pulsATRAS)){
      if (aux==0){
        cuenta--;
      }else{
        avance--;
      }
     }
   }
   estadoBotonAnterior2 = estadoBoton2;
//-----------BOTON ENTER
estadoBoton3 = digitalRead(pulsENTER);
if(estadoBoton3 != estadoBotonAnterior3){
  if(antirrebote3(pulsENTER)){

    if(aux2==0){
      cuenta3=1;
     }else{
        confirmacion=1;
      }

   }
  }
  estadoBotonAnterior3 = estadoBoton3;

//----------BOTÓN ESCAPE
  estadoBoton4 = digitalRead(pulsESCAPE);
  if(estadoBoton4 != estadoBotonAnterior4){
    if(antirrebote4(pulsESCAPE)){
```

```
        if(confirmacion==1){//INDICACION EN MODO DISCRETO DE EJEUCION
          cuenta=2;//SETEO PARA ELEGIR AVANCE DISCRETO
          cuenta3=1;//SETEO PARA INGRESAR A MENU AVANCE DISCRETO
          confirmacion=0;//RESETEO DE CONFIRMACIÓN PARA ESTAR EN MODO DE CONFIGURACIÓN
NUEVAMENTE
          Serial.println("ESTAS CONFIGURANDO");
          }
        else{
        cuenta4=1;
        }
      }

    }
    estadoBotonAnterior4 = estadoBoton4;



//-------------------MENÚ DE SELECCIÓN
   if (cuenta >2)
   {
    cuenta=1;
    }
   if (cuenta <1)
   {
    cuenta=2;
    }

  if(cuenta==1){
  if (cuenta3==HIGH)
  {
  aux=1;
  aux2=1;
   AvanceAbsoluto();
  }
  else{
  aux=0;
  aux2=0;
  //adentroabsoluto=0;
  avance=0;
```

```
  retro=0;
  confirmacion=0;
  Anuncio1="   PULSE ENTER  ";
  Anuncio2=" AVANCE ABSOLUTO  ";
  menu();
  }
  }


  if(cuenta==2){

  if (cuenta3==HIGH)
  {
  aux=1;
  aux2=1;
  AvanceDiscreto();
  }
  else{
  aux=0;
  aux2=0;
  avance=0;
  retro=0;
  confirmacion=0;
  Anuncio1="   PULSE ENTER  ";
  Anuncio2=" AVANCE DISCRETO  ";
  menu();
  }
  }
}


//************************************
//ACCIONES ESTANDO EN REMOTO
//************************************
else{
    //LED INDICADOR SALIDA CUANDO ESTA EN REMOTO
     digitalWrite(LOCAL_REMOTO,HIGH);

  //--------LECTURAS DE POSCIÓN DE ENCODER
  if( counter != temp ){
```

```
  temp = counter;
  //TRANSFORMACIÓN DE BITS A centímetros
    ejey=map(counter,0,8260,0,80);
    //LIMITACIÓN DE ENCODER PARA NO TENER MEDIDAS ERRÓNEAS
    if (counter > 20000){
    counter=0;
    }
  }


    //PRESENTACIÓN DE DATOS DE CALIBRADOR EN LCD
    lcd.setCursor(6,1);
    lcd.print(result1,2);
    if (result1 <= 100){
    lcd.setCursor(11,1);
    lcd.print ("    "); //Impresión de espacios en blanco para que no se sobrescriba
sobre el dato leido
    }


//*****************************************************************************
    //------------------ENTRADAS DEL ARDUINO(SALIDAS PLC O ABB)----------------------

//*****************************************************************************


    //-----------------REALIZAR AVANCE-----------
    estadosalidaABB1 = digitalRead(avanceABB);
    if (estadosalidaABB1==1 && estadosalidaABBAnterior1==0 && activacionsensor==0)
    {
       digitalWrite(LED, HIGH); //ENDENDER EL LED
       //Preparamos la salida para que el motor gire en un sentido
       digitalWrite (IN4, HIGH);
       digitalWrite (IN3, LOW);
       // Aplicamos PWM al pin ENB, haciendo girar el motor
       analogWrite(ENB,100);

       //ESTADOS EO Y E1 ENVIADOS
       digitalWrite(ESTADOE0,LOW);
       digitalWrite(ESTADOE1,HIGH);
    }
     if (estadosalidaABB1==0 && estadosalidaABBAnterior1==1)
    {
```

```
      digitalWrite(LED, LOW); //APAGAR EL LED
      analogWrite(ENB,0);

      //ESTADOS EO Y E1 ENVIADOS
      digitalWrite(ESTADOE0,LOW);
      digitalWrite(ESTADOE1,LOW);
   }
   estadosalidaABBAnterior1= estadosalidaABB1;


   //--------------REALIZAR RETROCESO-----------
   estadosalidaABB2 = digitalRead(retrocesoABB);
   if (estadosalidaABB2==1 && estadosalidaABBAnterior2==0)
   {
      digitalWrite(LED, HIGH); //ENDENDER EL LED}
      digitalWrite (IN4, LOW);
      digitalWrite (IN3, HIGH);
      // Aplicamos PWM al pin ENB, haciendo girar el motor
      analogWrite(ENB,100);

      //ESTADOS EO Y E1 ENVIADOS
      digitalWrite(ESTADOE0,LOW);
      digitalWrite(ESTADOE1,HIGH);

  }
    if (estadosalidaABB2==0 && estadosalidaABBAnterior2==1)
   {
      digitalWrite(LED, LOW); //APAGAR EL LED
      analogWrite(ENB,0);

      //ESTADOS EO Y E1 ENVIADOS
      digitalWrite(ESTADOE0,LOW);
      digitalWrite(ESTADOE1,LOW);
  }
  estadosalidaABBAnterior2= estadosalidaABB2;


  //--------------Activación de Sensor fotoeléctrico--------
   estadosensor = digitalRead(sensor);
    if(estadosensor==1 && estadosensorAnterior==0)
```

```
   {
      digitalWrite(SENSORFE, HIGH);
      digitalWrite(LED, LOW);
      activacionsensor=1;
      analogWrite(ENB,0);
    }

  if (estadosensor==0 && estadosensorAnterior==1)
    {
      digitalWrite(SENSORFE, LOW);
      activacionsensor=0;
    }
  estadosensorAnterior= estadosensor;

 //----------------CONFIRMACIÓN DE EJECUCIÓN--------------------
   if (estadosalidaABB4==1 && estadosalidaABBAnterior4==0){
    ejecucion=1;
  }
  if (estadosalidaABB4==0 && estadosalidaABBAnterior4==1){
    ejecucion=0;
  }
  estadosalidaABBAnterior4= estadosalidaABB4;

//-------------------------------------------------------------------------
//DETECCIÓN DE SEÑAL QUE ESTA LLEGANDO DEL ROBOT ABB PARA REFERENCIA EN Y
//-------------------------------------------------------------------------

if(digitalRead(REFERENCIAY)==HIGH) {
counter2++;
daletiempo=millis();
}

if(digitalRead(REFERENCIAY)==LOW) {
counter3++;
daletiempo2=millis();
}

  estadosalidaABB3 = digitalRead(REFERENCIAY);
  if (estadosalidaABB3 ==0 && estadosalidaABBAnterior3 ==1)
  {
```

```
    //TIEMPO QUE VA AVANZANDO EN ALTO CONFORME EL TIEMPO
    TALTOINT=daletiempo;

     //TIEMPO QUE PERMANECE REALMENTE EN ALTO LA SEÑAL
     TALTOBIEN=TALTOINT-PERIODOINT;
  }
    if (estadosalidaABB3==1 && estadosalidaABBAnterior3==0)
   {
    //PERIODO DE LA SEÑAL DE ENTRADA QUE VA AVANZANDO CONFORME EL TIEMPO
    PERIODOINT=daletiempo2;
    TBAJOINT=PERIODOINT-TALTOINT;
    PERIODOBIEN=TALTOBIEN+TBAJOINT;

  //MEDIDA DE SEÑAL RECIBIDA DE REFERENCIA Y
  MEDIDA_RECIBIDA_REFY=map(TALTOBIEN,0,PERIODOBIEN,0,80);
  Serial.println("MEDIDA REFERENCIA EN Y");
  Serial.println(MEDIDA_RECIBIDA_REFY);

  }
   estadosalidaABBAnterior3 = estadosalidaABB3;


//--------------------------------------------------------------------------------
//DETECCIÓN DE SEÑAL QUE ESTA LLEGANDO DEL PLC O ABB PARA INCREMENTOS DE Y(DELTAY)
//--------------------------------------------------------------------------------

if(digitalRead(avancediscretoABB)==HIGH) {
counter4++;
daletiempo3=millis();
}

if(digitalRead(avancediscretoABB)==LOW) {
counter5++;
daletiempo4=millis();
}

  estadosalidaABB4 = digitalRead(avancediscretoABB);
  if (estadosalidaABB4 ==0 && estadosalidaABBAnterior4 ==1)
  {
   //TIEMPO QUE VA AVANZANDO EN ALTO CONFORME EL TIEMPO
```

```
     TALTOINT2=daletiempo3;
     //TIEMPO QUE PERMANECE REALMENTE EN ALTO LA SEÑAL
      TALTOBIEN2=TALTOINT2-PERIODOINT2;
   }
     if (estadosalidaABB4==1 && estadosalidaABBAnterior4==0)
    {
     //PERIODO DE LA SEÑAL DE ENTRADA QUE VA AVANZANDO CONFORME EL TIEMPO
     PERIODOINT2=daletiempo4;
     TBAJOINT2=PERIODOINT2-TALTOINT2;
     PERIODOBIEN2=TALTOBIEN2+TBAJOINT2;

   //MEDIDA DE SEÑAL RECIBIDA DE REFERENCIA Y
   MEDIDA_RECIBIDA_INCREY=map(TALTOBIEN2,0,PERIODOBIEN2,0,80);
   Serial.println("INCREMENTOS EN Y");
   Serial.println(MEDIDA_RECIBIDA_INCREY);

   }
    estadosalidaABBAnterior4 = estadosalidaABB4;


   //----------REFERENCIAS DADAS EN Y---------------
    lcd.setCursor(0,0);
    lcd.print("Y(cm):");
    lcd.setCursor(6,0);
    lcd.print(ejey);
    if (ejey<=9.9)
   {
    lcd.setCursor(10,0);
    lcd.print(" ");
    }
    lcd.setCursor(0,1);
    lcd.print("X(mm):");
    lcd.setCursor(14,1);
    lcd.print("  ");
    lcd.setCursor(11,0);
    lcd.print ("PY:");
 }

}
```

```
void menu(){
   lcd.setCursor(0,0);
   lcd.print(Anuncio1);
   lcd.setCursor(0,1);
   lcd.print(Anuncio2);
}
void AvanceAbsoluto(){
  //adentroabsoluto=1;
    if (cuenta4==1){
      cuenta3=0;
      cuenta4=0;
      lcd.clear();
     }

    lcd.setCursor(0,0);
    lcd.print("Y(cm):");
    lcd.setCursor(6,0);
    lcd.print(ejey);
    if (ejey<=9.9)
   {
    lcd.setCursor(10,0);
    lcd.print(" ");
    }

    lcd.setCursor(0,1);
    lcd.print("X(mm):");
    lcd.setCursor(14,1);
    lcd.print("  ");
    lcd.setCursor(11,0);
    lcd.print ("PY:");

    //PRESENTACION DE DATOS DE CALIBRADOR EN LCD
    lcd.setCursor(6,1);
    lcd.print(result1,2);
    if (result1 <= 100){
    lcd.setCursor(11,1);
    lcd.print ("    "); //Impresión de espacios en blanco para que no se sobreescriba
sobre el dato leido
    }
    if (avance < 0){
```

```
      avance=0;
      lcd.setCursor(14,1);
      lcd.print("M0");
 }
lcd.setCursor(14,0);
lcd.println(avance);
if(avance<10){
lcd.setCursor(15,0);
lcd.println(" ");
}


 if(confirmacion==1){
  if(ejey<avance){
  Serial.println("avanzando");
  digitalWrite(LED,HIGH);
  //Preparamos la salida para que el motor gire en un sentido
  digitalWrite (IN4, HIGH);
  digitalWrite (IN3, LOW);
  // Aplicamos PWM al pin ENB, haciendo girar el motor
  analogWrite(ENB,100);
   //ESTADOS EO Y E1 ENVIADOS
   digitalWrite(ESTADOE0,LOW);
   digitalWrite(ESTADOE1,HIGH);
  }


  if(ejey>avance){
  Serial.println("retrocediendo");
  digitalWrite(LED,HIGH);
  //Preparamos la salida para que el motor gire en sentido contrario
  digitalWrite (IN4, LOW);
  digitalWrite (IN3, HIGH);
  // Aplicamos PWM al pin ENB, haciendo girar el motor
  analogWrite(ENB,100);

   //ESTADOS EO Y E1 ENVIADOS
   digitalWrite(ESTADOE0,LOW);
   digitalWrite(ESTADOE1,HIGH);
  }
```

```
     if(ejey==avance){
     confirmacion=0;
     digitalWrite(LED,LOW);
    // Aplicamos PWM al pin ENB para detener al motor
     analogWrite(ENB,0);

      //ESTADOS EO Y E1 ENVIADOS
      digitalWrite(ESTADOE0,LOW);
      digitalWrite(ESTADOE1,LOW);
     }
    }
    if (avance > 80){
       avance=80;
    }
    medidaavance=ejey;
}

void AvanceDiscreto(){
   //VARIABLES PARA PODER REGRESAR EN EL MENU
    if (cuenta4==1){
          cuenta3=0;
          cuenta4=0;
          lcd.clear();

     }
   //IMPRESIÓN DE MENSAJES PARA VISUALIZAR MEDIDAS EN LCD
   lcd.setCursor(0,0);
   lcd.print("Y(cm):");
   lcd.setCursor(6,0);
   lcd.print(ejey);
    if (ejey<=9.9)
   {
    lcd.setCursor(10,0);
    lcd.print(" ");
    }

   lcd.setCursor(11,0);
   lcd.print ("IY:");
   lcd.setCursor(0,1);
```

```
   lcd.print("X(mm):");
   lcd.setCursor(14,1);
   lcd.print("  ");
    //PRESENTACION DE DATOS DE CALIBRADOR EN LCD
    lcd.setCursor(6,1);
    lcd.print(result1,2);
    if (result1 <= 100){
    lcd.setCursor(11,1);
    lcd.print ("    "); //Impresión de espacios en blanco para que no se sobrescriba sobre el dato leído
    }

    lcd.setCursor(14,0);
    lcd.print(avance);
    if (avance<=9.9)
   {
    lcd.setCursor(15,0);
    lcd.print(" ");
    }

    if (avance<=0){
    avance=0;
    }



    if (medidaavance<=0){
    medidaavance=0;
    }

   //AVANCE
   if(confirmacion==1){ //LA CONFIRMACION EN 1 REPRESENTA CUANDO SE AH PULSADO SOBRE ENTER
     Serial.println("ESTOY EN EJECUCION");

     //AVANCE
     if (estadoBoton){ // SE LEE ESTADOBOTON PARA CONFIRMAR AVANCE HACIA ADELANTE
       avance=avance-1;
       medidaavance=medidaavance+avance;
       Serial.println(medidaavance);
       confirmacion=0;
```

```
      confirmacionadelante=1;
     }
    //RETROCESO
    if(estadoBoton2){
    avance=avance+1;
    medidaavance=medidaavance-avance;
    Serial.println(medidaavance);
    confirmacion=0;
    confirmacionatras=1;
   }
  }
   //ACTIVACION AVANCE
   if (confirmacionadelante==1){
   if(ejey<medidaavance){
   digitalWrite(LED,HIGH);

   //Preparamos la salida para que el motor gire en un sentido
    digitalWrite (IN4, HIGH);
    digitalWrite (IN3, LOW);
    // Aplicamos PWM al pin ENB, haciendo girar el motor
    analogWrite(ENB,100);

     //ESTADOS EO Y E1 ENVIADOS
     digitalWrite(ESTADOE0,LOW);
     digitalWrite(ESTADOE1,HIGH);
   }
   else{
   digitalWrite(LED,LOW);
   confirmacion=1;//SIGUE ESTANDO EN MODO DE EJECUCION
   confirmacionadelante=0;//SE RESETEA EL AVANCE CUANDO LLEGA AL PUNTO DE DESTINO
    // Aplicamos PWM cero al pin ENB, haciendo detener el motor
    analogWrite(ENB,0);
         //ESTADOS EO Y E1 ENVIADOS
     digitalWrite(ESTADOE0,LOW);
     digitalWrite(ESTADOE1,LOW);
    }
   }
   //ACTIVACION RETROCESO
   if(confirmacionatras==1){
   if(ejey>medidaavance){
```

```
    digitalWrite(LED,HIGH);
    //Preparamos la salida para que el motor gire en un sentido
    digitalWrite (IN4, LOW);
    digitalWrite (IN3, HIGH);
    // Aplicamos PWM al pin ENB, haciendo girar el motor
    analogWrite(ENB,100);
      //ESTADOS EO Y E1 ENVIADOS
      digitalWrite(ESTADOE0,LOW);
      digitalWrite(ESTADOE1,HIGH);
    }
    else{
    digitalWrite(LED,LOW);
    confirmacion=1;//SIGUE ESTANDO EN MODO DE EJECUCION
    confirmacionatras=0;
    // Aplicamos PWM cero al pin ENB, haciendo detener el motor
    analogWrite(ENB,0);
    //ESTADOS EO Y E1 ENVIADOS
    digitalWrite(ESTADOE0,LOW);
    digitalWrite(ESTADOE1,LOW);
     }
    }
    if (avance>40){
    avance=40;
    }
}
  // Función que evita el rebote del pulsador ADELANTE
  boolean antirrebote(int pin)
  {
    int contador = 0;
    boolean estado;
    boolean estadoAnterior;

    do
    {
      estado = digitalRead(pin);
      if(estado != estadoAnterior)
      {
        contador = 0;
        estadoAnterior = estado;
      }
```

```
    else
    {
      contador = contador+ 1;
    }
    delay(1);
  } while(contador < tiempoAntirrebote);

  return estado;
}
// Función que evita el rebote del pulsador ATRAS
boolean antirrebote2(int pin2)
{
  int contador2 = 0;
  boolean estado2;
  boolean estadoAnterior2;
  do
  {
    estado2 = digitalRead(pin2);
    if(estado2 != estadoAnterior2)
    {
      contador2 = 0;
      estadoAnterior2 = estado2;
    }
    else
    {
      contador2 = contador2+ 1;
    }
    delay(1);
  } while(contador2 < tiempoAntirrebote2);

  return estado2;
}
// Función que evita el rebote del pulsador ENTER
boolean antirrebote3(int pin3)
{
  int contador3 = 0;
  boolean estado3;
  boolean estadoAnterior3;
  do
  {
```

```
    estado3 = digitalRead(pin3);
    if(estado3 != estadoAnterior3)
    {
      contador3 = 0;
      estadoAnterior3 = estado3;
    }
    else
    {
      contador3 = contador3+ 1;
    }
    delay(1);
  } while(contador3 < tiempoAntirrebote3);

  return estado3;
}
  // Función que evita el rebote del pulsador ESCAPE
boolean antirrebote4(int pin4)
{
  int contador4 = 0;
  boolean estado4;
  boolean estadoAnterior4;
  do
  {
    estado4 = digitalRead(pin4);
    if(estado4 != estadoAnterior4)
    {
      contador4 = 0;
      estadoAnterior4 = estado4;
    }
    else
    {
      contador4 = contador4+ 1;
    }
    delay(1);
  } while(contador4 < tiempoAntirrebote4);

  return estado4;
}
```

```
   //----------FUNCIÓN ai0 y ai1 PARA DETECCIÓN DE POSICIÓN DE ENCODER
  void ai0() {
  //ai0 es activada si el pin Digital 2 va de LOW a HIGH
  // Se verifica el pin3 para determinar la dirección

  if(digitalRead(3)==LOW) {
  counter++;
  }else{
  counter--;
  }
  }
  void ai1() {
  //ai1 es activada si el pin Digital 3 va de LOW a HIGH
  // Se verifica el pin2 para determinar la dirección

  if(digitalRead(2)==LOW) {
  counter--;
  }else{
  counter++;
  }
  }
//FUNCION PARA DEVOLVER LOS DATOS QUE SE PRESENTAN EN EL CALIBRADOR DIGITAL
 void calibre1() {
    sign1=1;
    value1=0;
    for (i=0;i<23;i++) {
      while (digitalRead(clockpin)==HIGH) { }
      while (digitalRead(clockpin)==LOW) {}
      if (digitalRead(datapin)==LOW) {
      if (i<20) {
        value1|= 1<<i;
      }
      if (i==20) {
        sign1=-1;
      }
    }
  }
   result1=(value1*sign1)/100.00;
}
```

La LCD 16x2 a ocupar tiene acoplado el circuito integrado PCF8574 mediante un módulo, la pantalla requiere entonces solo la conexión para la alimentación (5VDC, GND) y se dispone de dos pines (SDA y SCL) para realizar la comunicación I2C, esto como se indica en la Figura 34.

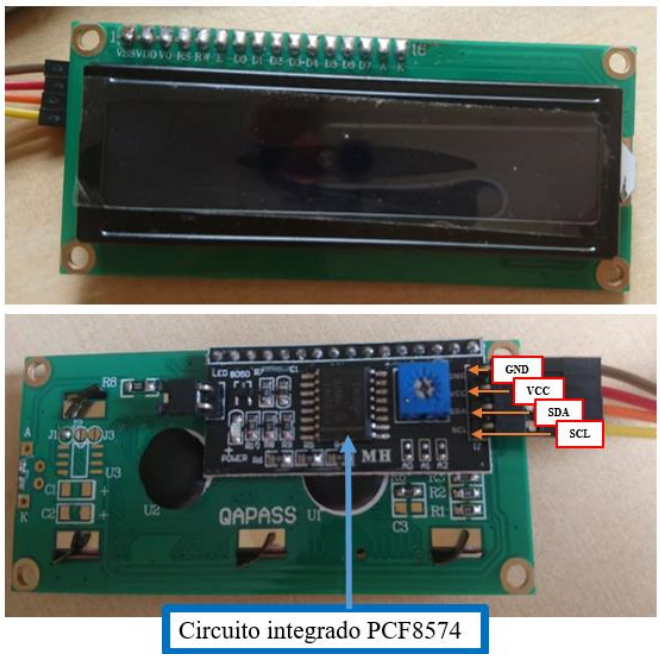


Figura 34: LCD 16x2 con comunicación I2C

El proyecto tuvo pruebas de funcionamiento inicial únicamente ocupando por separado los elementos del sistema tales como un motor de 24VDC, encoder incremental, puente HL298, pantalla LCD, fuente de 24VDC, computadora portátil todo esto como se indica en la Figura 35, para luego realizar las respectivas pruebas con la banda transportadora ubicada en el laboratorio de automática de la Universidad (Figura 36).

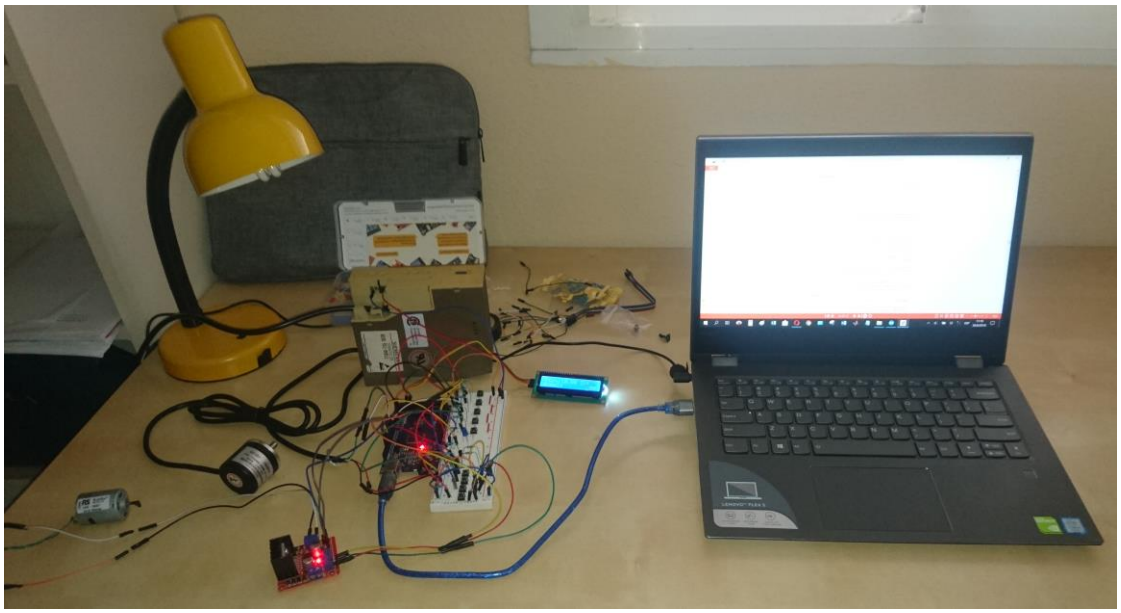


Figura 35: Pruebas iniciales del sistema

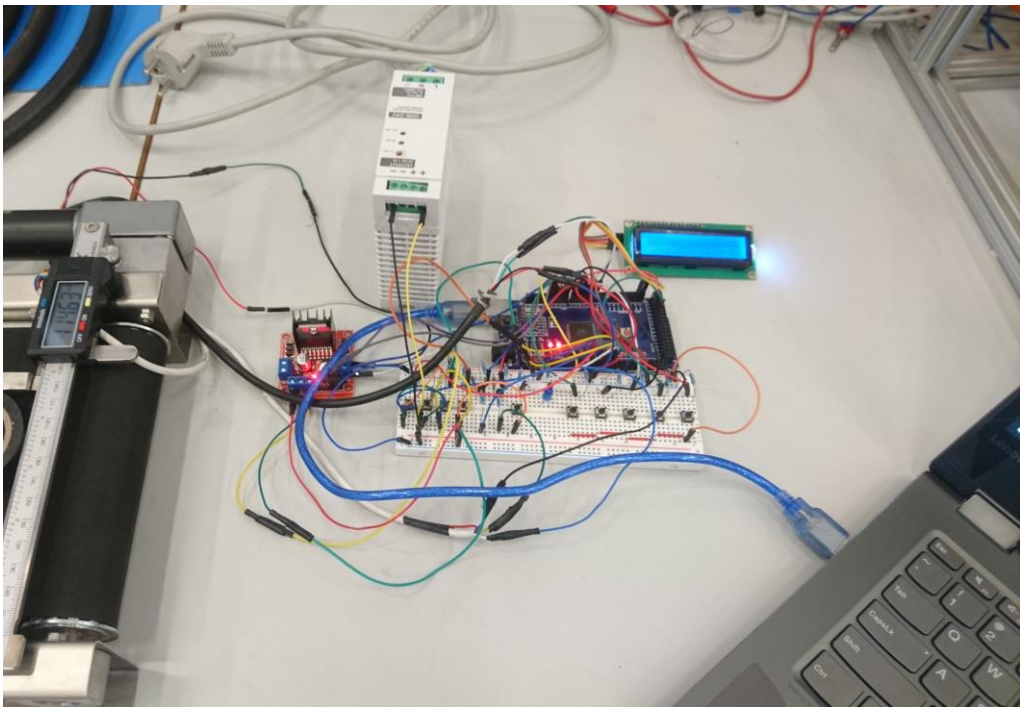


Figura 36: Pruebas de funcionamiento inicial con la banda transportadora

Para enviar y recibir las señales del microcontrolador que servirán para la comunicación con el robot ABB, se añadió un segundo protoboard para las pruebas de funcionamiento realizadas por separado de la estructura de la banda transportadora (Figura 37), luego de esto se incoroporó dicho circuito a la banda transportadora que se ubica en el laboratorio de Automática de la Universidad (Figura 38).

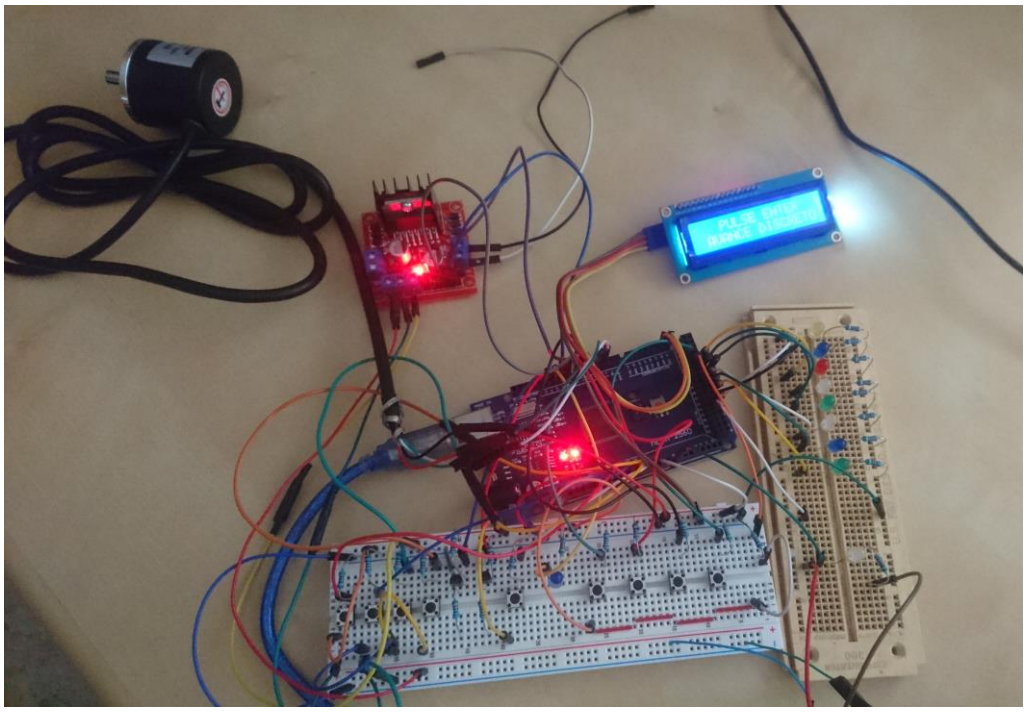


Figura 37: Pruebas para comunicación de microcontrolador con robot ABB

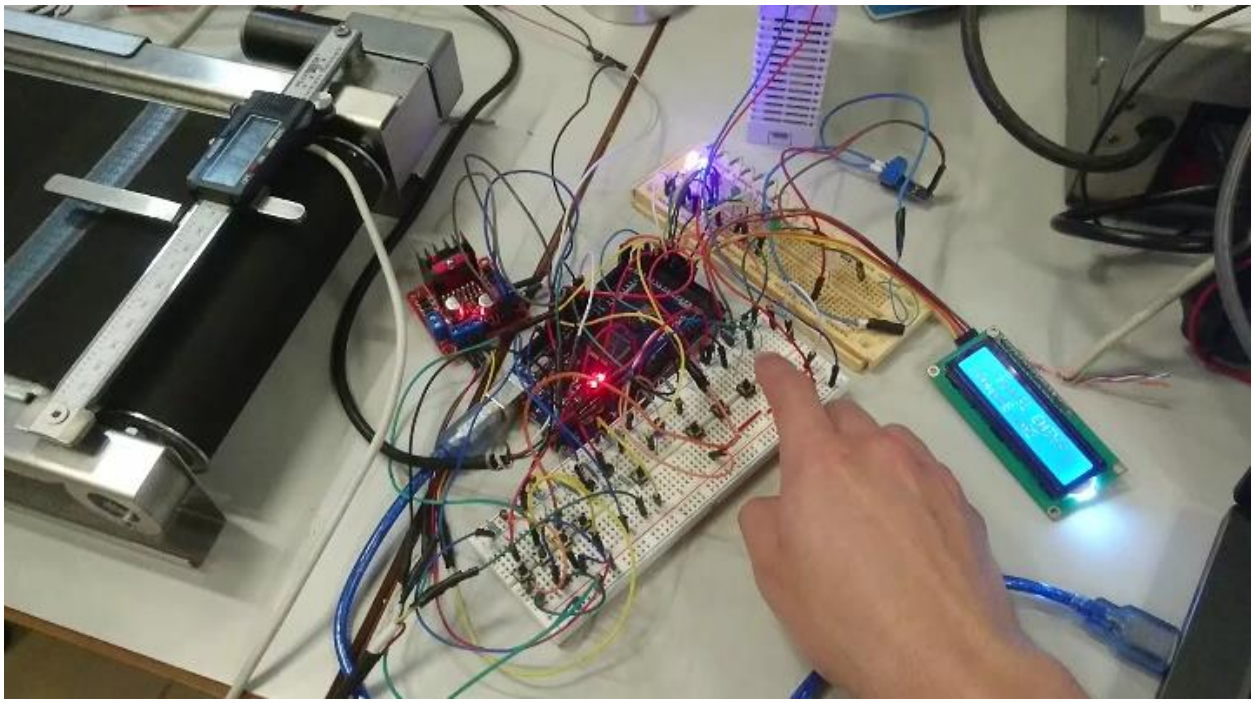


Figura 38: Pruebas de funcionamiento del sistema con la banda transportadora.

Se procede a indicar a continuación como se debe posicionar una pieza dentro de la estructura de la cinta transportadora realizando una medida con el calibrador digital dispuesto en la misma, tal y como se indica en la Figura 39.

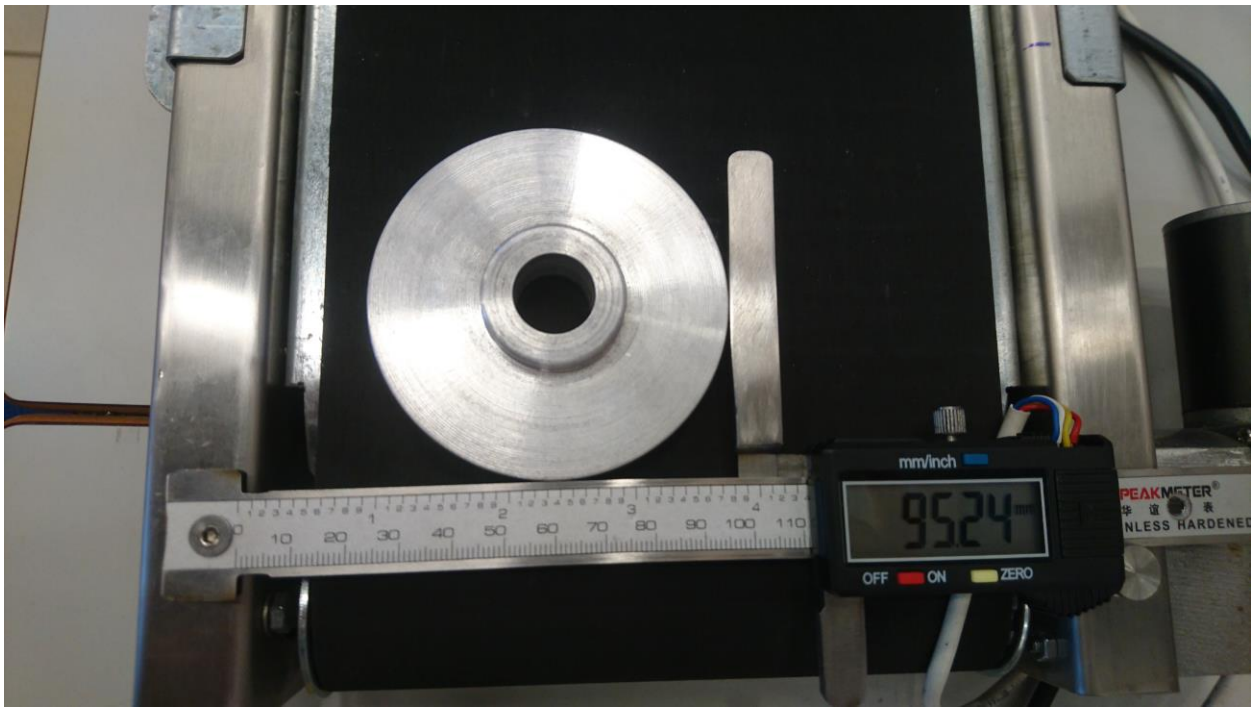


Figura 39: Pieza ubicada al inicio de la banda transportadora con medida en X de 95.24mm

Se ingresó al sistema con las indicaciones anteriormente vistas en el apartado 4.2 para realizar un ejemplo de avance absoluto que posicionó a la pieza ubicada dentro de la banda transportadora en eje X 95mm y en eje Y 100mm, todo esto como se indica en la Figura 40. Todos los cambio realizados se irán visualizando en la LCD del sistema tal y como muestra la Figura 41.

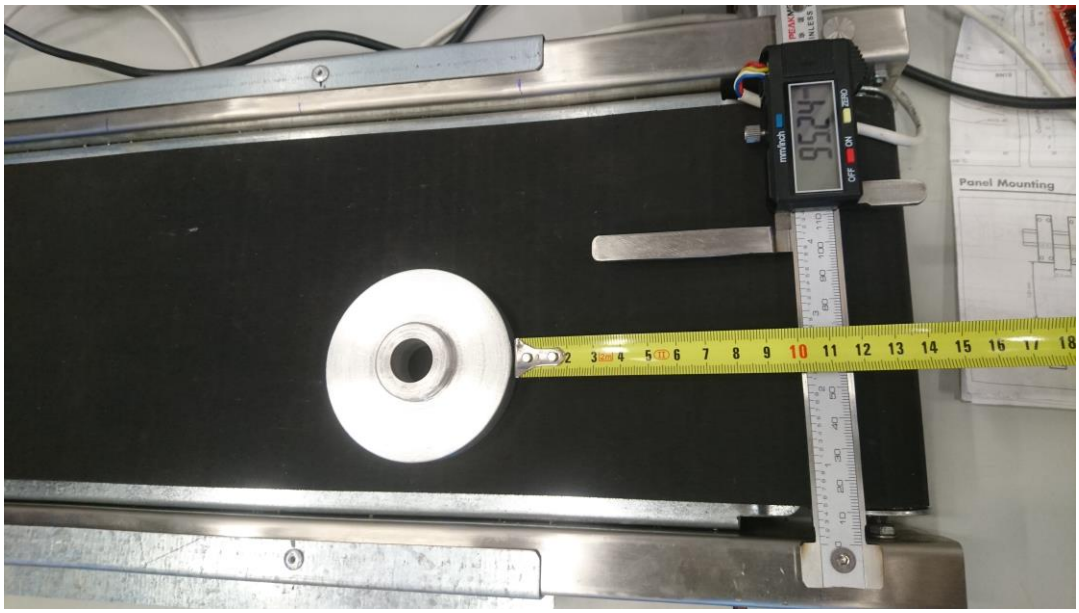


Figura 40: Pieza ubicada en eje X 95mm y en eje Y 100mm

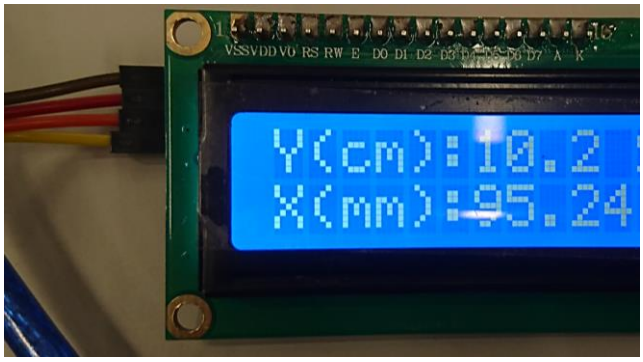


Figura 41: LCD del sistema que indica la ubicación de la pieza en eje X 95mm y en eje Y 100mm

5 CONCLUSIONES Y DESARROLLOS FUTUROS

El uso de cintas transportadoras en la industria ocupa un espacio fundamental, ya que facilita la movilización a grandes o pequeñas distancias de objetos, piezas o cualquier elemento que se coloque sobre estas dependiendo de la aplicación donde trabajen, de igual manera las cintas transportadoras no alteran al producto que se está trasladando y también son fáciles de adaptar a cualquier terreno.

Dentro del desarrollo de este proyecto se puede mencionar que la banda transportadora ayuda de manera óptima para el posicionamiento de piezas dentro del entorno de un robot, dando como referencias posiciones en dos ejes para la utilización que requiera el usuario. Se obtuvo como resultados el poder manipular a la banda transportadora tanto de manera manual con dos modos avance (Absoluto y Discreto) teniendo como interfaz de por medio una botonera con un LCD donde se ingresan las órdenes que serán procesadas por el microcontrolador. La interfaz se desarrolló pensando en acciones simples que ejecutará el operador y no resulten complicadas al momento de trabajar con el sistema.

De igual manera se pudo enviar señales del microcontrolador hacia el robot, para conocer la posición que tiene la pieza de manera remota, así como también si se accionó el sensor en la estructura de la cinta, o la activación de interruptores presentes en la botonera del sistema. Teniendo las señales enviadas por parte del robot hacia el microcontrolador se consiguió mover la cinta transportadora con las referencias que requiera el operador.

El proyecto además permite el uso de la banda transportadora de manera manual, en donde se tiene el control de la misma sin ser necesario el uso del microcontrolador.

Tendiendo como referencia este trabajo se pueden llevar a cabo desarrollo de líneas futuras de estudio, iguales a las que se mencionan a continuación:

- Ampliar el número de bandas transportadoras y brazos robóticos.
- Dotar al sistema realizado de una interfaz que sea de tipo táctil y no requiera del uso de botones físicos para trabajar con el mismo.
- Adicionar a este proyecto sistemas de reconocimiento de imagen que permitan obtener un mejor control en la posición de la pieza.

REFERENCIAS

[1] «Web oficial de Arduino,» [En línea]. Available: www.arduino.cc

[2] *Software Fritzing.*

[3] *Software Proteus 8.*

BIBLIOGRAFÍA Y SITIOS WEB DE CONSULTA

- Web oficial de Arduino - www.arduino.cc
- Web oficial de Adafruit - www. adafruit.com
- Consulta en multitud de foros, video tutoriales y blogs de autores particulares.