UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGON

ELABORACIÓN DE UN PLC DIDÁCTICO MEDIANTE UN MICROCONTROLADOR

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO ELÉCTRICO ELECTRÓNICO

P R E S E N T A

LUIS EDUARDO GONZÁLEZ ARREGUÍN

DIRECTOR DE TESIS

M. en I. MAURICIO ONTIVEROS SALGADO

Cd. Nezahualcoyotl, Edo. de México, 2017

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

VNIVERSIDAD NACIONAL AVFNOMA DE MEXICO

Facultad de Estudios Superiores Aragón

DIRECCIÓN

LUIS EDUARDO GONZALEZ ARREGUIN

Presente

Con fundamento en el punto 6 y siguiente, del Reglamento para Exámenes Profesionales en esta Facultad, y toda vez que la documentación presentada por usted reúne los requisitos que establece el solicitado Reglamento; me permito comunicarle que ha sido aprobado su tema de tesis y asesor.

TÍTULO: "ELABORACIÓN DE UN PLC DIDÁCTICO MEDIANTE UN MICROCONTROLADOR"

M. en I. MAURICIO ONTIVEROS SALGADO

ASESOR:

Aprovecho la ocasión para reiterarle mi distinguida consideración.

Atentamente

"POR MI RAZA HABLARÁ EL ESPÍRITU"

Nezahualcóyotl, Estado de México, 17 de marzo de 2017.

DIRECTOR

M. EN I. FERNANDO MACEDO CHAGOLLA

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN UNAM DIRECCIÓN

C p Secretaria Académica
C p Jefatura de Carrera de Ingeniería Eléctrica Electrónica
C p Asesor de Tesis

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

INGENIERÍA ELÉCTRICA ELECTRÓNICA

VOTO APROBATORIO DE TESIS

M. en I. FERNANDO MACEDO CHAGOLLA
DIRECTOR DE LA FES ARAGÓN
P R E S E N T E

Por medio del presente, me permito comunicar a usted que revisé el trabajo de **tesis** titulado:

ELABORACIÓN DE UN PLC DIDÁCTICO MEDIANTE UN MICROCONTROLADOR

Que presenta el (la):

LUIS EDUARDO GONZÁLEZ ARREGUÍN

Con Número de Cuenta: **41100917-7**

Para obtener el título de:

INGENIERO ELÉCTRICO ELECTRÓNICO

Y considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el Examen Profesional correspondiente, no tengo inconveniente en otorgar mi VOTO APROBATORIO.

A T E N T A M E N T E
" POR MI RAZA HABLARÁ EL ESPÍRITU "
Nezahualcóyotl, Estado de México, a 24 **de** mayo **de 20** 17

ATENTAMENTE

ING. ISAAC ISMAEL LÓPEZ TRUJANO
NOMBRE Y FIRMA DEL PROFESOR

Vo. Bo.

M. en I. FIDEL GUTIÉRREZ FLORES
JEFE DE CARRERA DE INGENIERÍA ELÉCTRICA ELECTRÓNICA

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN
INGENIERÍA ELÉCTRICA ELECTRÓNICA

VOTO APROBATORIO DE TESIS

M. en I. FERNANDO MACEDO CHAGOLLA
DIRECTOR DE LA FES ARAGÓN
P R E S E N T E

Por medio del presente, me permito comunicar a usted que revisé el trabajo de **tesis** titulado:

ELABORACIÓN DE UN PLC DIDÁCTICO MEDIANTE UN MICROCONTROLADOR

Que presenta el (la):

LUIS EDUARDO GONZÁLEZ ARREGUÍN

Con Número de Cuenta: **41100917-7**

Para obtener el título de:

INGENIERO ELÉCTRICO ELECTRÓNICO

Y considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el Examen Profesional correspondiente, no tengo inconveniente en otorgar mi VOTO APROBATORIO.

A T E N T A M E N T E
" POR MI RAZA HABLARÁ EL ESPÍRITU "
Nezahualcóyotl, Estado de México, a 24 **de** mayo **de 20** 17

ATENTAMENTE	Vo. Bo.
ING. RAMÓN PATIÑO RODRÍGUEZ	**M. en I. FIDEL GUTIÉRREZ FLORES**
NOMBRE Y FIRMA DEL PROFESOR	**JEFE DE CARRERA DE INGENIERÍA ELÉCTRICA ELECTRÓNICA**

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN
INGENIERÍA ELÉCTRICA ELECTRÓNICA

VOTO APROBATORIO DE TESIS

M. en I. FERNANDO MACEDO CHAGOLLA
DIRECTOR DE LA FES ARAGÓN
P R E S E N T E

Por medio del presente, me permito comunicar a usted que revisé el trabajo de **tesis** titulado:

ELABORACIÓN DE UN PLC DIDÁCTICO MEDIANTE UN MICROCONTROLADOR

Que presenta el (la):

LUIS EDUARDO GONZÁLEZ ARREGUÍN

Con Número de Cuenta: **41100917-7**

Para obtener el título de:

INGENIERO ELÉCTRICO ELECTRÓNICO

Y considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el Examen Profesional correspondiente, no tengo inconveniente en otorgar mi VOTO APROBATORIO.

A T E N T A M E N T E
" POR MI RAZA HABLARÁ EL ESPÍRITU "
Nezahualcóyotl, Estado de México, a 24 **de** mayo **de 20** 17

ATENTAMENTE

M. en I. MAURICIO ONTIVEROS SALGADO
NOMBRE Y FIRMA DEL PROFESOR

Vo. Bo.

M. en I. FIDEL GUTIÉRREZ FLORES
JEFE DE CARRERA DE INGENIERÍA ELÉCTRICA ELECTRÓNICA

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN
INGENIERÍA ELÉCTRICA ELECTRÓNICA

VOTO APROBATORIO DE TESIS

M. en I. FERNANDO MACEDO CHAGOLLA
DIRECTOR DE LA FES ARAGÓN
P R E S E N T E

Por medio del presente, me permito comunicar a usted que revisé el trabajo de **tesis** titulado:

ELABORACIÓN DE UN PLC DIDÁCTICO MEDIANTE UN MICROCONTROLADOR

Que presenta el (la):

LUIS EDUARDO GONZÁLEZ ARREGUÍN

Con Número de Cuenta: **41100917-7**

Para obtener el título de:

INGENIERO ELÉCTRICO ELECTRÓNICO

Y considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el Examen Profesional correspondiente, no tengo inconveniente en otorgar mi VOTO APROBATORIO.

A T E N T A M E N T E
" POR MI RAZA HABLARÁ EL ESPÍRITU "
Nezahualcóyotl, Estado de México, a 24 **de** mayo **de 20** 17

ATENTAMENTE	Vo. Bo.
DR. OCTAVIO DÍAZ HERNÁNDEZ	M. en I. FIDEL GUTIÉRREZ FLORES
NOMBRE Y FIRMA DEL PROFESOR	JEFE DE CARRERA DE INGENIERÍA ELÉCTRICA ELECTRÓNICA

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN
INGENIERÍA ELÉCTRICA ELECTRÓNICA

VOTO APROBATORIO DE TESIS

M. en I. FERNANDO MACEDO CHAGOLLA
DIRECTOR DE LA FES ARAGÓN
P R E S E N T E

Por medio del presente, me permito comunicar a usted que revisé el trabajo de **tesis** titulado:

ELABORACIÓN DE UN PLC DIDÁCTICO MEDIANTE UN MICROCONTROLADOR

Que presenta el (la):

LUIS EDUARDO GONZÁLEZ ARREGUÍN

Con Número de Cuenta: **41100917-7**

Para obtener el título de:

INGENIERO ELÉCTRICO ELECTRÓNICO

Y considerando que dicha tesis reúne los requisitos necesarios para ser discutida en el Examen Profesional correspondiente, no tengo inconveniente en otorgar mi VOTO APROBATORIO.

A T E N T A M E N T E
" POR MI RAZA HABLARÁ EL ESPÍRITU "
Nezahualcóyotl, Estado de México, a 24 de mayo de 20 17

ATENTAMENTE

ING. OSCAR GUADALUPE MORENO ESPINOZA
NOMBRE Y FIRMA DEL PROFESOR

Vo. Bo.

M. en I. FIDEL GUTIÉRREZ FLORES
JEFE DE CARRERA DE INGENIERÍA ELÉCTRICA ELECTRÓNICA

Para Scipionni,
Nunca dejes de cabalgar

Agradecimientos:

Sin duda, estoy inmensamente agradecido con Dios por brindarme sin fin de personas, que a lo largo de mi vida me han acompañado, guiado y en su momento, regañado. Este trabajo escrito no solo es la culminación de una etapa personal, sino el resultado del apoyo incondicional de mi familia y amigos.

Primeramente, gracias a mi asesor Mauricio Ontiveros, por confiar en el proyecto, proporcionar ideas para que este funcionase y enseñarme a redactar. A mi amigo y compadre Irving, por plantear la posibilidad de realizar un *PLC*, su amistad, cariño y apoyo incondicional a lo largo de la carrera y fuera de esta. Al profesor Oscar Guadalupe, por el asesoramiento a lo largo de la realización del proyecto y el apoyo en las pruebas experimentales.

Le agradezco a mi mejor amiga Valeria Dozzetti, por tantas noches interminables, su amistad incondicional, apoyo en momentos difíciles e innumerables aventuras que pasamos juntos. A Erickito *(CHEP)*, por mostrarme que el honor y lealtad aún existen y que la gallardía e inocencia pueden ir de la mano. A Jorge y Kika, por enseñarme lo grande que puedo llegar a ser, sus consejos sinceros y regaños certeros.

A Lucky, por ser tan empalagosamente tierna y amorosa, porque eres como un golpe de suerte a cualquier vida que llegas. A Lucrecia, por dejarme seguir luchando por mis sueños.

Padre, madre, sin ustedes no sería quien soy ahora, esto es el resultado de todo su trabajo, gracias por no rendirse jamás conmigo, por amarme como a nadie. Padre, eres

el mejor hombre que conozco, el más inteligente, valiente y fuerte. Aún sigo pensando qué, cuando sea grande, quiero ser como tú. Madre, eres el perfecto ejemplo de que el miedo es domable y que el cambio es posible; gracias por escucharme, amarme y enseñarme que los sueños son para realizarse y nada menos.

A mi mejor amigo, Yakin, el mejor hermano que pueda existir. Gracias por soportarme tanto tiempo, por dejarme el departamento cuando te lo pedía, por acompañarme a lo largo de la vida. Te amo y *“SLMC”*. A mis abuelos, tíos, primos, sobrinos. A toda mi familia les agradezco su apoyo a lo largo de mi vida.

Contenido

Índice de figuras

Resumen

En el presente trabajo se describe la propuesta de un *PLC* implementado con un microcontrolador *PIC* de 8-bit basado en la arquitectura *RISC,* el cual contiene las funciones básicas para emular un controlador simple.

En una tarjeta electrónica de bajo costo que contiene los módulos básicos para el estudio del control a través de un *PLC,* se pretende brindar una herramienta de estudio para los alumnos del área de control y automatización de la carrera de Ingeniería Eléctrica y Electrónica.

Las entradas digitales del microcontrolador son aisladas galvánicamente mediante optoacopladores y sus salidas a través de relevadores. Además, se introducen módulos auxiliares para el control de precisión con módulos *PWM* y el sensado a través de puertos *ADC*.

Haciendo uso de un *software* de licencia libre, se programa en lenguaje escalera y se compila un fichero hexadecimal el cual es grabado en el microcontrolador mediante un programa de escritorio desarrollado a través del entorno *VisualBasic.NET*.

Con la finalidad de brindar una implementación rápida, se incorpora una serie de programas pregrabados con las herramientas básicas para el control. El conjunto de las mismas permite controlar y automatizar prácticamente cualquier proceso escolar y realizar prácticas de laboratorio.

Capítulo 1
Introducción

Las necesidades de la industria han cambiado con el pasar de los años, por consiguiente, la implementación de procesos simples incluyendo tecnología de vanguardia, es un impulso para la investigación de nuevas herramientas y dispositivos electrónicos para el control de dichos procesos.

La automatización hoy en día no abarca solamente a los procesos industriales, la necesidad del ser humano por facilitar los trabajos complejos, evitar tareas monótonas y ahorrar tiempo en los procesos, ha puesto a la automatización en un nivel en el que es indispensable para el desarrollo de nuevos sistemas tecnológicos [1].

Un proceso puede ser controlado por diversos sistemas. Uno de estos es el *PLC* (Acrónimo de *Programmable Logic Controller)* que, a pesar de tener algunos años en el mercado, aún abarcan un número importante de procesos industriales como herramienta principal para el control y automatización. Actualmente, estos dispositivos programables están dotados de complejos circuitos y potentes procesadores [4], lo que los mantiene como sistemas embebidos de alta eficiencia, fiabilidad y robustez para el control y automatización de procesos en la industria. Esta alta tecnología implementada ocasiona que sus precios se mantengan altos.

Además del alto costo de adquirir un PLC para fines didácticos, muchos dispositivos requieren del mantenimiento de expertos capacitados por el fabricante debido a la complejidad de sus componentes [7]. Esta es una razón para implementar una herramienta que sea económica tanto en la adquisición como en el mantenimiento de sus dispositivos electrónicos.

La enseñanza del control y la programación de los *PLC* son considerables en las disciplinas técnicas que buscan controlar o automatizar algún proceso. Por lo que el facilitar los medios para una instrucción fácil, económica y sencilla, sea del interés de quienes desean implementar estos dispositivos dentro del control de proceso

1.1 Planteamiento del problema

Un proceso de manufactura básico consiste en una estación de trabajo, manejo y transporte automatizado de piezas y un sistema de control [2]. Al implementar estos sistemas se busca la mayor ganancia con respecto al costo y beneficio de los productos.

Un problema para la instrucción del control por *PLC* proviene de la necesidad industrial de tener un dispositivo con mayor eficiencia, fiabilidad y tecnología. Esto trae consigo sistemas costosos que no son muy eficientes para la enseñanza del control y la automatización.

1.2 Justificación

Dentro del plan de estudios *2007* de la IEE (*Ingeniería Eléctrica-Electrónica*) se cuenta con siete materias enfocadas en el control y automatización de procesos (Automatización industrial, Sistemas de control, Instrumentación electrónica, Instrumentación de procesos industriales, Sistemas de control programable, Control de procesos, y Temas selectos de control), de las cuales solo dos de estas (Sistemas de control e instrumentación de procesos industriales) cuentan con prácticas para de desarrollar en el laboratorio L3.

En la materia de Sistemas de control programable se utilizan los *PLC*. Sin embargo, la disposición para su estudio en el laboratorio es limitado y la adquisición de varios equipos para cubrir la necesidad del alumnado es costoso. Esto no debería de ser impedimento para que los interesados en aprender el control por sistemas programables no lleven a la práctica lo adquirido en la teoría.

De acuerdo a la tesis de maestría de la FI de la UNAM *“Realización de un PLC didáctico”* se propone el diseño de un *PLC* implementando un microcontrolador *PIC* (Peripheral Interface Controller), el cual es programado por medio de una serie de comandos *ASCII,* emulando el lenguaje *IL* (Instruction List).

En este proyecto se pretende continuar con la búsqueda de herramientas fiables para el estudio del control y la automatización.

1.3 Propuesta de solución

Una propuesta para contar con un medio de estudio de *PLC* más económico y amigable para el usuario, es diseñar una tarjeta electrónica implementando un microcontrolador que sea compacta y una aplicación de PC capaz de grabar programas en la tarjeta por el puerto *USB*.

Para obtener un resultado semejante al industrial, en términos de programación, se usará un software de licencia libre, el cuál es un entorno de programación basado en el lenguaje *Ladder* (escalera) y que también genera archivos hexadecimales para microcontroladores.

Además, se implementará una interfaz gráfica dentro de la aplicación para PC que contará con programas pregrabados, los cuáles pueden servir de guía para la implementación de la tarjeta como controlador en procesos simples.

1.4 Objetivos

El objetivo principal es desarrollar una tarjeta electrónica compacta y económica que sea programable vía *USB* mediante una *PC* para emular la programación de un *PLC* comercial.

Desarrollar una aplicación en el ordenador que procese el texto de un archivo hexadecimal para transmitirlo a través de una comunicación serial hacia el microcontrolador y este los grabe en su memoria de programa.

Desarrollar cuatro programas sencillos, que sean configurados desde la aplicación de *PC* y cuenten con tareas básicas para el control de sistemas.

1.5 Metodología

El trabajo comienza con la búsqueda de proyectos similares en la base de datos de tesis de grado de maestría y doctorado de la facultad de ingeniería, para continuar con el desarrollo de material didáctico para la enseñanza del control y automatización.

Posteriormente se busca un software de licencia libre que sea un entorno de programación en *Ladder* para microcontroladores y generare archivos hexadecimales.

Se procede a comparar los microcontroladores soportados por el entorno para elegir el más conveniente, tomando en cuenta su capacidad de procesamiento y almacenamiento en memoria *flash*.

Se comparan las distintas plataformas que brinden herramientas para el uso de programación orientada a objetos, que no requiera de una versión de pago y que sea simple para el diseño de aplicaciones de escritorio.

Se desarrollan los algoritmos para la conversión, procesamiento, y comunicación de los archivos hexadecimales (*.hex*) a través de la aplicación creada en la computadora y se desarrolla el algoritmo de recepción y grabado de *bytes* en la *memoria flash* del microcontrolador (*bootloader).*

Enseguida, se diseña una tarjeta compacta, amigable y lo más completa, manteniendo un bajo costo, la cual debe contar con entradas digitales, entradas analógicas, salidas *PWM* y salidas a relevadores, que sean suficientes para el estudio del control y automatización en diversas áreas.

Por último, se realizan pruebas de funcionamiento del sistema en conjunto y se recopilan los datos y observaciones las cuales están documentadas en este trabajo.

Capítulo 2
Fundamentos básicos

2.1 PLC

Hace algunos años, en la época en que el control de procesos industriales se implementaba mediante el uso de relevadores, temporizadores y una inmensa red de cables, y llegaba a ser requerido algún cambio en estos sistemas de control, se ocasionaban pérdidas de tiempo y recursos humanos en el reacondicionamiento del sistema, esto elevaba considerablemente el costo del producto final [1].

La demanda de productos en masa ha impulsado el desarrollo de nuevos sistemas de control que sustituyan los anticuados y poco fiables sistemas de relevadores y cables. El *PLC* surgió como un dispositivo para remplazar estos circuitos [4], ya que el cambio en las interconexiones lo realiza el procesador siguiendo las instrucciones a ejecutar grabadas en su memoria. Las celdas de manufactura en la actualidad están interconectadas por redes de comunicación, de esta manera un procesador tiene el control sobre los procesos [2].

La *NEMA* (*National Electrical Manufacturers Association*) describe al *PLC* como: un aparato electrónico digital, que utiliza una memoria no volátil en donde almacena instrucciones grabadas anteriormente por un programador, para ejecutar funciones específicas a través de un procesador para el control de máquinas o procesos, implementando módulos digitales o analógicos de entrada y salida [8].

Los controladores programables ocupan microprocesadores de alta velocidad y capacidad de procesamiento, lo que agrega funciones útiles para el control de procesos más complejos. Dado que cuentan con lo último en tecnología y son robustos en su diseño, los hace convenientes para su uso en procesos industriales hostiles como alta temperatura, suministro de potencia eléctrica no confiable, vibraciones mecánicas, etc. [3].

2.1.1 Estructura de un PLC

La estructura del *PLC* se conforma de cuatro partes: las entradas y salidas, el procesador, la fuente de alimentación y la terminal de programación [10], como se observa en la Figura 2.1.

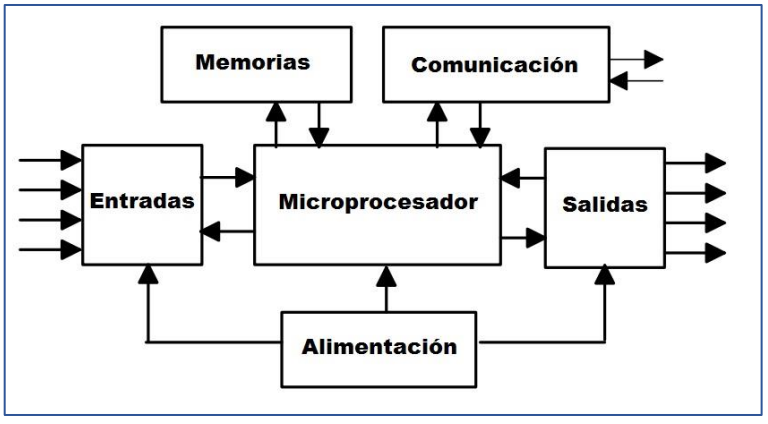


Figura 2.1 Estructura de un *PLC.*

Entradas y Salidas: Esta sección interconecta los dispositivos de alta potencia al sistema de baja potencia [5]. El *PLC* se constituye por módulos separados de entradas y salidas. Generalmente el módulo de entrada de cada terminal recibe señales con tensión de $127V_{AC}$ o bien, $12V_{CD}$ o $24V_{CD}$. Estas tensiones tienen que ser transformadas a voltajes TTL. Para poder hacer esta transformación se utilizan circuitos de rectificación *de señales* u optoacopladores, los cuales aíslan galvánicamente la etapa de alta potencia de la electrónica del procesador [31]. Por otra parte, el módulo de salida opera de manera opuesta, transforma la señal emitida por el procesador a los dispositivos de alta potencia conectados a la terminal de salida. Para la transformación de estas señales se suelen ocupar relevadores u optotriacs [4].

Procesador: La sección del procesador está formada por la memoria de programa (memoria flash), la memoria volátil (memoria *RAM*) y el microprocesador [10]. Este último es el dispositivo electrónico que ejecuta las instrucciones del programa de control, direcciona los datos almacenados en la memoria de sistema y determina el estado de las entradas y salidas tomando como base las condiciones de entrada y la instrucción a realizar según lo indique la memoria de programa. En algunos *PLC* estos

módulos se encuentran separados y no en un solo chip como en los microcontroladores.

Fuente de alimentación: El diseño de esta etapa depende de cada fabricante y la aplicación a la que va enfocado el *PLC* [34], aunque en general tienen la misma función: encargarse de convertir la tensión de entrada en los voltajes requeridos por el procesador y los dispositivos electrónicos.

Terminal de programación: Esta terminal suele ser una interfaz de comunicación conectada al puerto serie (RS-232) o al puerto *USB* de una *PC*. Con el software de comunicación del *PLC* instalado en la computadora, las instrucciones programadas pueden ser transmitidas del ordenador al controlador [6].

2.1.2 Lenguajes de programación

La evolución en la calidad con la que los *PLC* funcionan, trajo consigo una variedad de lenguajes tan amplia como las empresas encargadas de manufacturar estos controladores. Distintos lenguajes traían diversas mejoras individuales, aunque, estas mejoras llegaban a resultar incomodas para la industria a la hora de invertir en una mejora o cambio del proceso [1,10].

Una de las soluciones más eficientes es el uso de estándares que permitan normalizar y simplificar la programación y reutilización de códigos. La *IEC (International Electro-techincal Commission*) publicó el estándar *IEC-1131* para la programación de *PLC* [12], el cual normaliza cuatro lenguajes y un auxiliar como herramienta para el desarrollo de aplicaciones de control:

1. Lista de instrucciones *(IL)*
2. Texto estructurado *(ST)*
3. Diagrama de bloques funcionales *(FBD)*
4. Lenguaje escalera *(Ladder)*
5. Gráfico secuencial de funciones (herramienta *SFC*) [11].

Este estándar especifica la sintaxis, semántica y estructura del lenguaje de programación [9]. El adoptar un estándar regulado por una comisión internacional impactó de manera positiva en la reducción de costos y tiempos en los procesos controlados por un *PLC*.

2.1.3 Lenguaje Ladder

El lenguaje común de programación de *PLC* para lógicas discretas es el lenguaje gráfico *Ladder*, basado en el diseño lógico con relevadores. Se le llama escalera por ser secuencial, siguiendo un orden de izquierda a derecha y de arriba hacia abajo en líneas. Esto quiere decir que la alimentación de energía proviene del lado izquierdo del diagrama y la descarga al lado derecho. Cada línea representa un estado de control iniciando desde la parte alta del programa. La alimentación de la línea puede provenir de la entrada a un pin del procesador o un registro de memoria (relevador interno) [4,10].

El estándar *IEC-1131-3* establece simbología gráfica del lenguaje *Ladder*, en la siguiente figura se muestran algunos símbolos normalizados usados por el entorno de programación *LDmicro* y su descripción [13].

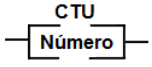
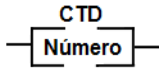
Símbolo	Descripción
	Contacto normalmente abierto
	Contacto normalmente cerrado
	Bobina
	Encendido retardado
	Contador incremental
	Contador decremental

Figura 2.2 Símbolos de lenguaje *Ladder*.

En la Figura 2.3 se muestra un ejemplo de programación en *Ladder* que representa a la función:

$$(A \cdot B + C) \cdot \overline{D} \cdot E \cdot \overline{F} = Q$$

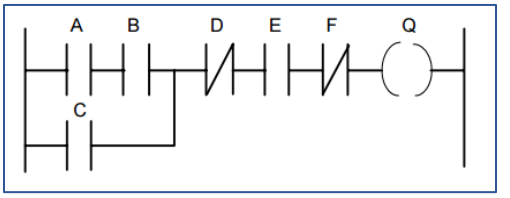


Figura 2.3 Ejemplo de programación con lenguaje *Ladder*.

Este proyecto usará la plataforma *LDmicro* como herramienta para programar en escalera, es software libre bajo la licencia *GPL* de *GNU* desarrollada por Jonathan Westhues. Su uso es sencillo, compila ficheros hexadecimales, contiene una gran variedad de instrucciones para programar y cuenta con un modo simulación que resulta útil para ejecutar programas de prueba.

2.2 Microcontroladores

Los microcontroladores se encuentran prácticamente en cualquier aparato electrónico que ocupemos en nuestra vida diaria. Aunque, el concepto de controlador permanece invariable a través del tiempo; como un dispositivo que se emplea para el manejo de uno o varios procesos [14], los microcontroladores han cambiado a lo largo de décadas, acoplando módulos para el control de sistemas tecnológicos actuales.

Los microcontroladores al igual que la mayoría de los dispositivos basados en microprocesadores, cumplen la función de ejecutar instrucciones de propósito general [17]. Estas instrucciones pueden ser programadas según lo requiera el proceso en el cual se pretenda implementar dicho dispositivo.

En la actualidad, una persona promedio interactúa con alrededor de 400 microcontroladores en un día, su implementación se hace viable por el poco espacio que requieren y el bajo consumo de energía para su funcionamiento [15].

Un sistema embebido es un circuito electrónico computacional capaz de realizar una función específica [16]. Con esto se puede decir que los microcontroladores pueden ser considerados como sistemas embebidos, ya que se componen por un procesador y su diseño solo ejecuta funciones específicas.

2.2.1 Microcontrolador PIC16f887

El *PIC16f887* es un microcontrolador fabricado por *Microchip* de 8 bits disponible en 40 y 44-*pin* que cuenta con tecnología *CMOS* (*Complementary Metal Oxide Semiconductor*) [28]. Este dispositivo implementa algunas características de la estructura *RISC* (*Reduced Instruction Set Computer*), como el uso de la arquitectura *Harvard* (Manejo de memoria de datos e instrucciones por separado), una reducida lista de instrucciones, ciclos simples de procesamiento, entre otras [19]. A pesar de pertenecer a la anticuada familia *16f*, este *PIC* cuenta con los periféricos necesarios para ser implementado en proyectos sencillos o para uso didáctico.

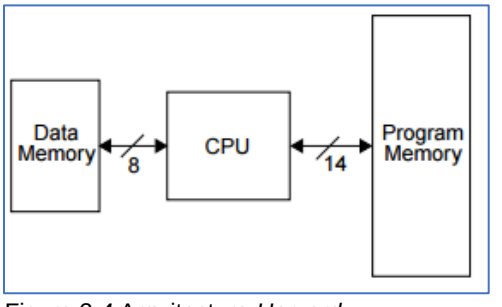


Figura 2.*4* Arquitectura *Harvard*

Entre sus periféricos se encuentran catorce *ADC* (*Analogic/Digital Converter*), cuatro canales de salida *PWM* (*Pulse Width Modulation*), comunicación serial *USART* (*Universal Synchronous Asynchronous Reciver Transmiter*) [28], por mencionar algunos.

En la Figura 2.5 se muestra el *PIC16f887* en versión encapsulada de 40-pin

Figura 2.5 *PIC16f887 DIP40*

2.2.2 Diagrama a bloques del PIC16f887

En la siguiente figura se muestra el diagrama a bloques del *PIC16f887* extraído de su *datasheet*.

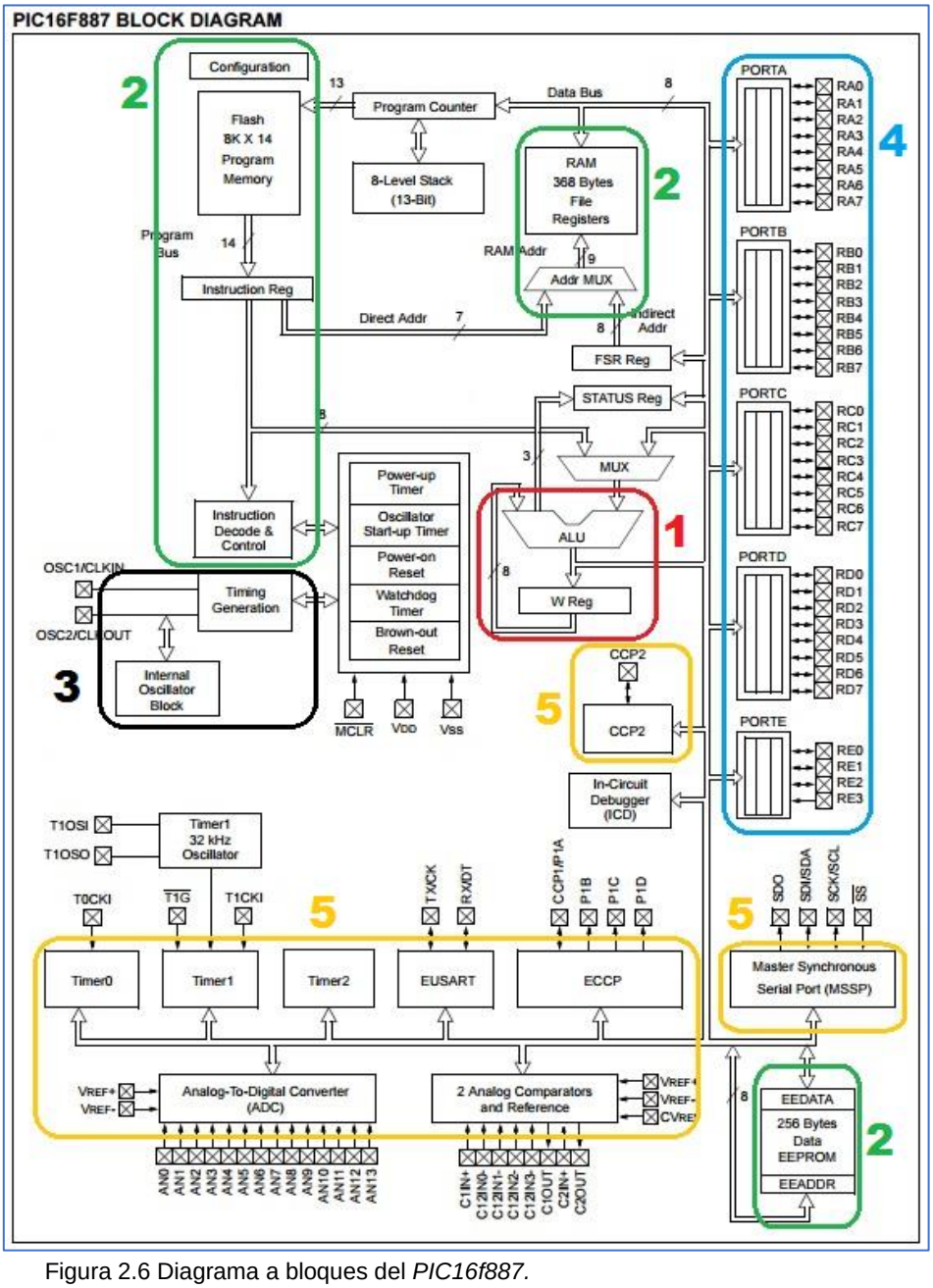


Figura 2.6 Diagrama a bloques del *PIC16f887.*

El diagrama se divide en:

1. Sección de procesamiento (*ALU* y registro *W*) marcada en color rojo.
2. Sección de memoria (*RAM, Flash* y *EEPROM*) marcada en color verde.
3. Sección de oscilación marcada en color negro
4. Sección de puertos de entrada y salida marcada en color azul.
5. Sección de módulos (*ADC, CCP,* comunicación y *Timers*) marcada en color dorado.

2.2.3 Memoria Flash del PIC16f887

Es una memoria no volátil del tipo borrable y programable eléctricamente, con un mínimo uso de circuitos auxiliares [17], bajando su costo considerablemente respecto a otras memorias no volátiles. Hoy en día las memorias *flash* han sustituido a las unidades *EEPROM* (Electrical Ereasable PROM) como unidades para el almacenamiento de programas.

El uso de la memoria *flash* en los microcontroladores brinda la posibilidad del grabado In-Circuit, lo cual se traduce en el grabado y borrado de la memoria a través de pines de comunicación [16]. En el *PIC16f887,* estos pines son los mismos que son usados para la comunicación *USART.* La memoria *flash* en este dispositivo tiene capacidad de 8k x 14, la cual es suficiente para el uso de este *PIC* en este proyecto.

2.2.4 Lenguaje ASM

El lenguaje ensamblador (*ASM*) es una forma en la que los humanos nos comunicamos con los computadores a través de unos y ceros [20]. Al ser el lenguaje nativo de los procesadores, los programas que se suelen implementar suelen ser más rápidos y compactos que los programados en lenguajes de alto nivel (*C, Basic, etc.*), además de aprovechar al máximo la capacidad del procesador.

Su estructura es parecida a una lista secuenciada de instrucciones representada por nemónicos, los cuales son un grupo de

instrucciones predefinidas en cada procesador a la hora de fabricarlo. Cada línea del programa cuenta con un nemónico y un operando sobre el cual se ejecutará la instrucción.

2.3 Programación orientada al objeto

La *OOP* (*Object Oriented Programming*) no es un lenguaje de programación, sino que es considerada como una técnica que puede ser aplicada en cualquier lenguaje [22]. Los entornos de programación actuales incorporan estructuras de la *OOP*.

La evolución de los lenguajes de programación incorporando la esencia de la *OOP* ha sido paulatina e ininterrumpida. El famoso lenguaje *C* ha pasado a la plataforma *C#* incorporando el *Framework .NET* pasando por el exitoso *C++* que por años fue el lenguaje favorito de los programadores que implementaban esta técnica. Por su parte, *Basic* evolucionó a *Visual Basic*, el cual en versiones modernas adoptó el *Framework .NET*, consolidándose como una plataforma a la par del *C#* y dejando de lado la muy criticada falta de *herencia* en sus primeras versiones. Por otra parte, *Java* es la plataforma que ha implementado de manera pura la esencia de la *programación orientada al objeto,* siendo concebida para cubrir los requerimientos de la estructura *OPP* [35].

Esta técnica nació con el propósito de simplificar la descripción de códigos para resolver problemas que cada vez son más complejos. Se le denomina orientada al objeto por la semejanza con el mundo real para la manipulación de objetos [23].

2.3.1 Definición de objeto

A diferencia del mundo real en donde un objeto es cualquier acción, materia física, real o imaginaria, un objeto de *OPP* es un conjunto de datos y métodos [22]. Estos actúan como una sola unidad a la que se le llama objeto, en donde un dato es un calificativo (Color, tamaño, peso) y un método representa una acción (Respirar, masticar, hablar).

En la informática un dato es la clase a la cual pertenece el objeto y el método son los procesos que puede realizar el mismo.

2.3.2 Definición de clase

Una clase sirve para crear muchos objetos independientes pero que tienen las mismas características [25]. Esto quiere decir que si, por ejemplo, creamos un objeto, este puede pertenecer a la clase *SerialPort* (que se usa en este proyecto). Esta clase brinda al objeto creado cualidades que solo ella brinda y que las diferencian de las otras, como la comunicación serial en este caso. Cada objeto puede adquirir esas cualidades según sea requerido por el programador y su objetivo.

2.3.3 Definición de método

El método son las funciones o acciones que lleva a cabo una clase [22], como ya se mencionó, clase y método actúan como una unidad. Continuando con el ejemplo de la clase *SerialPort,* esta posee *métodos* para controlar recursos del *puerto serie* como: *Open* (Abre una nueva conexión del *puerto serial*), *WriteLine* (Escribe una línea a través del *puerto serie*), *ReadLine* (Lee una línea del puerto serie), por mencionar algunas [36].

2.3.4 Definición de herencia

La herencia permite reutilizar el código escrito en clases superiores y reescribir las diferencias que existen entre ellas [22]. Al igual que en la naturaleza los hijos heredan ciertos rasgos y características de los padres para adaptarse y sobrevivir, en la informática estos rasgos heredados son fragmentos de código de una o varias clases.

Esta herramienta es lo que vuelve a la *OPP* una técnica productiva y eficiente para resolver problemas de gran envergadura. Siguiendo con la clase *SerialPort,* esta heredó rasgos de una clase “superior” o padre. En la siguiente figura se muestra la jerarquía de herencia de la clase *SerialPort*.

Figura 2.7 Jerarquía de herencia clase *SerialPort*.

Como se observa, la clase *SerialPort* heredó características de la clase *Component* hasta llegar a *Object* [36].

Otra ventaja de la herencia es el reacondicionamiento de códigos, si se hace un cambio en *Component*, este cambio también afectará a *SerialPort* y a las *subclases* que se hayan creado a partir de ella, esto es una ventaja para la actualización de sistemas basados en *OPP*.

El uso de herencia múltiple trae grandes ventajas como la facilidad, flexibilidad y simplicidad en la definición de nuevas clases, sin embargo, no es recomendable su uso en exceso, ya que provoca sistemas de comportamiento impredecible y el aumento de tiempo en el proceso de programas [23].

Esta ciencia es abstracta y compleja, las definiciones escritas anteriormente buscan la neutralidad y simplicidad entre textos de diversos autores. La clase *SerialPort,* métodos y jerarquía de herencia pertenece al *Framework .NET,* por lo que una clase llamada del mismo modo en otro entorno puede tener diferente estructura y jerarquía de herencia.

2.4 Framework .NET

Hace varios años atrás la única forma de programar en *Windows* era haciendo uso de un compilador (*C++*) y de un *API.* Este último es un vasto conjunto de funciones relacionadas que permiten la comunicación con el sistema operativo. A través del *API* de *Win32* se programaban los botones, ventanas, etc. [25]

El extenso número de funciones contenidas en el *API* complicaba la tarea a los programadores, por lo que el desarrollo de nuevos compiladores en busca de facilitar estas problemáticas trajo consigo a *Visual Basic,* el cual trabajaba por encima de *Win32* usando el viejo lenguaje *Basic* y además contaba con un desarrollo

amigable de ventanas, botones, etc. Sin embargo, como se mencionó antes, su falta de herencia evitaba que se pudiera llevar a cabo una *OPP* y a su vez, evitaba el desarrollo de aplicaciones de alta complejidad.

El *Framework .NET* desarrollado por *Microsoft* busca la unificación de las diversas herramientas de desarrollo en un mismo entorno para facilitar la solución de problemas en varios lenguajes [26]. Este entorno brinda a los programadores la comodidad de desarrollar aplicaciones en el lenguaje que más les acomode (*Visual Basic, C++, C#, J#, etc.)* y solucionar problemas con las herramientas que brinda la *OPP*.

2.5 Arquitectura *Framework .NET*

La plataforma *.NET* se divide en dos ramas de programación de interfaz gráfica: *Windows Forms* (Programas de escritorio) y *Web Forms* (Servicios Web) [29], estas dos ramas usan un conjunto de herramientas y servicios único, para el desarrollo de aplicaciones. A continuación, se mencionan cuatro de estas características que son esenciales para esta plataforma.

1. ***CLS*** (*Common Language Specification*), son reglas que los lenguajes deben cumplir, ya que esta herramienta permite la interoperabilidad entre lenguajes.
2. ***BCL*** (*Base Class Library*), es una colección de códigos que puede ser empleada desde cualquier lenguaje *.NET*, define las clases básicas y tipos de datos que se ocupan entre los distintos lenguajes.
3. ***CIL*** (*Common Instruction List*), es un lenguaje intermedio que es leído y ejecutado por el *Runtime*. Esto quiere decir que los programas en *.NET* no se compilan a un código ensamblador para el microprocesador, sino que compilan a un código en *CIL* que es optimizado por el *Runtime* para el dispositivo en el cual se usará.
4. ***CLR*** (*Common Language Runtime*), es un programa que lee un código *CIL* generado por el compilador de algún entorno de programación de la plataforma *.NET*, gestiona al código encargándose de su ejecución en el procesador, optimización, memoria, entre otras [25,35].

En la Figura 2.8 se puede observar de manera resumida la arquitectura del *Framework .NET*.

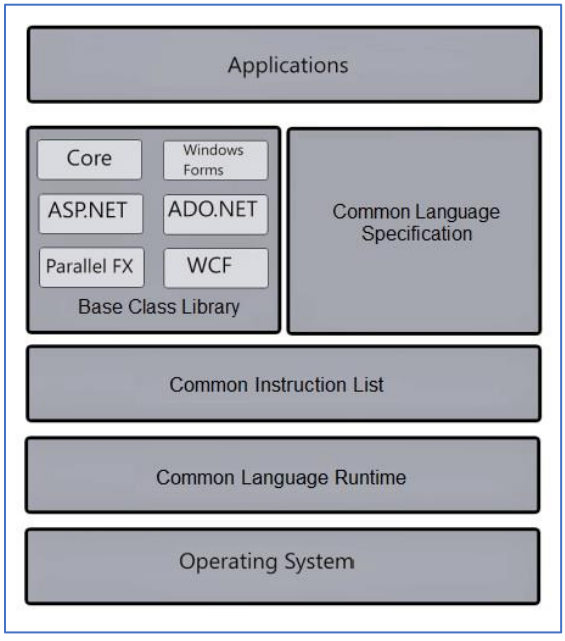


Figura 2.8 Arquitectura del *Framework .NET.*

Estas son solo algunas de las características con las que cuenta la plataforma *.NET.* Hasta el año 2010 se habían desarrollado alrededor de 60 lenguajes que incorporaban *.NET* [29], esto demuestra que se trata de una plataforma robusta y fiable.

Además, es multiplataforma, por lo que podemos programar en alguna plataforma en particular y ser ejecutado en otra que cuente con el *Runtime* del entorno. En otras palabras, un programa diseñado en *Windows* puede ser ejecutado en *Linux* si se cuenta con el *Runtime .NET* actualizado.

Capítulo 3
Diseño y programación del hardware

3.1 Propuesta de diseño

En los objetivos se propuso el diseño de un *PLC* económico y compacto, por lo que la implementación de componentes debe ser de fácil acceso y mantenimiento. Además, para ser compacto, solo cuenta con entradas y salidas necesarias para el uso didáctico.

Para ampliar su alcance en proyectos de diversos enfoques, cuenta con entradas *ADC* y salidas *PWM*. El módulo de comunicación es de fácil conexión y el puente ocupado, también requiere ser útil para otros proyectos.

En la alimentación, se propone un diseño que sea independiente, tanto el microcontrolador como los módulos de entrada y salida. También, se propone que la tarjeta sea alimentada de un tomacorriente de 127 V_{AC} para que su funcionamiento no dependa de la conexión *USB* al ordenador.

3.2 Diseño de entradas y salidas

En este apartado se explicará el diseño electrónico de los módulos de entradas y salidas digitales, módulos *ADC* y *PWM* del *PLC* didáctico que se propone.

3.2.1 Entradas digitales

La sección de entrada puede ser rectificada u optoaislada [1], para este proyecto se usará la optoaislada mediante el uso de optoacopladores. La gran variedad de estos dispositivos en el mercado brinda seguridad en aplicaciones industriales de proceso o tecnología eléctrica [32].

El uso de optoacopladores en los *PLC* les brinda robustez en cuanto a protección del procesador y sus circuitos [10]. El funcionamiento de un optoacoplador en esencia es el mismo para todos los dispositivos que se encuentran en el mercado; está

formado por una fuente emisora de luz y un fotosensor de silicio, la radiación luminosa emite señales de un circuito a otro sin necesidad de conexión eléctrica [14].

Existen muchos tipos de optoacopladores y la diferencia entre ellos radica en los dispositivos de salida que se ocupen. Para este proyecto se usa la configuración básica con un transistor como se observa a continuación:

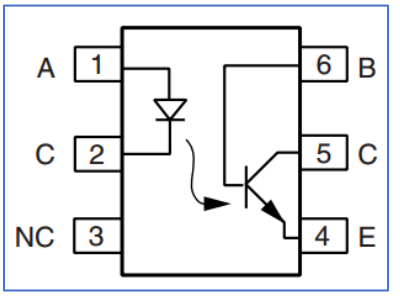


Figura 3.1 Optoacoplador *4n25.*

Los *PLC* comerciales llegan a soportar tensiones de $24V_{DC}$ a la entrada [5], para regular esta tensión a $5V_{DC}$ se propone el uso de un diodo zener de 5.1V a 1/2*W* y un selector de resistencias, el cuál permita elegir la óptima dependiendo de la tensión suministrada a la entrada.

El uso del zener como regulador de voltaje se muestra en el diagrama siguiente [14].

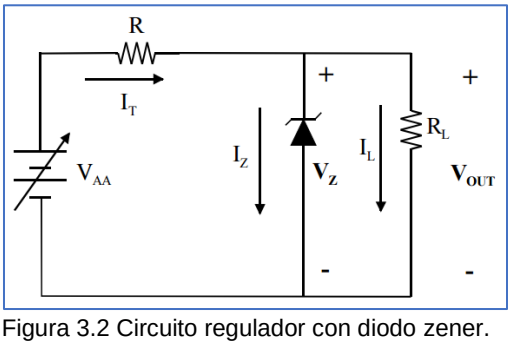


Figura 3.2 Circuito regulador con diodo zener.

Donde:

- **V_{AA}** es una fuente de alimentación *DC.*
- **R** es la resistencia limitadora.
- **I_T** es la corriente total suministrada.
- **I_Z** es la corriente que corre por el zener.

- **I_L** es la corriente de carga.
- **R_L** es la resistencia de carga.

La resistencia **R** debe ser escogida de forma que el zener permanezca en el modo de tensión constante, por lo que el cálculo para seleccionar los dispositivos convenientes viene dado por las siguientes ecuaciones [33].

$$R = \frac{V_{AA\,min} - V_Z}{I_{Z\,min} + I_{L\,max}}$$

$$R = \frac{V_{AA\,max} - V_Z}{I_{Z\,max} + I_{L\,min}}$$

Donde surgen dos incógnitas $I_{Z\,min}$ e $I_{Z\,max}$. Una regla práctica que es aceptable para el cálculo de las mismas es:

$$I_{Z\,max} = 10 \cdot I_{Z\,min}$$

La ecuación para el cálculo de la potencia máxima estimada para la resistencia es:

$$P_R = I_{T\,max}(V_{AA\,max} - V_Z)$$

Y el cálculo de la potencia máxima del *zener* está dada por:

$$P_Z = I_Z \cdot V_Z$$

En la Figura 3.3 se puede observar el diagrama esquemático que se elaboró para este proyecto, el cual solo muestra una entrada digital implementada en esta propuesta de *PLC*. El diseño que se propone cuenta con cinco entradas iguales a la que se muestra.

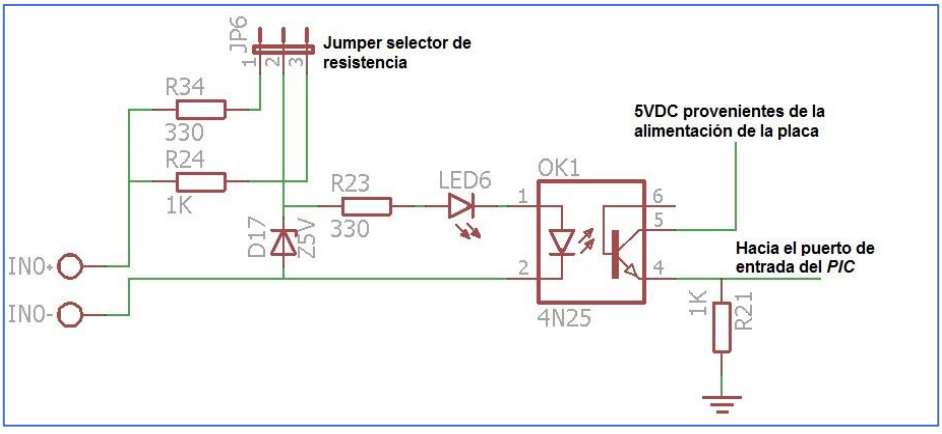


Figura 3.3 Diagrama esquemático de entrada digital.

3.2.2 Módulo ADC

Los módulos de conversión *ADC* implementados en un *PLC* pueden ser de dos tipos:

- Unipolares: Manejan solo valores positivos (0-5V o 0-10V)
- Bipolares: Manejan valores positivos y negativos (-5 a +5V o -10 a +10V) [34].

Para este proyecto se implementarán cuatro módulos *ADC* unipolares de 0 a 5V y al igual que en las entradas digitales, se propone el uso de un diodo zener a 5V con propósitos de protección. El diseño de este circuito es sencillo pero funcional, en la Figura 3.4 se observa el diagrama esquemático de una entrada analógica

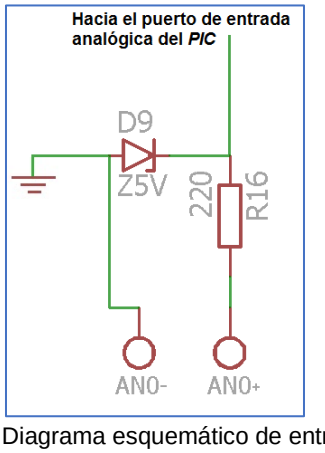


Figura 3.4 Diagrama esquemático de entrada analógica.

3.2.3 Salidas digitales

Comúnmente un *PLC* cuenta con 4, 8, 12 o 16 salidas dependiendo del sistema en el que se implementará. Debido a que este módulo suele estar conectado a dispositivos de potencia, se constituyen por elementos eléctricamente aislados. Los circuitos que generalmente son usados para este propósito son los relevadores, transistores *o* triacs [1,10]. En este proyecto se implementarán las salidas a través de relevadores.

El relevador es un dispositivo electromagnético que permite el paso de corriente de un circuito a otro cuando la bobina que se encuentra en su interior es energizada [38]. En los sistemas modernos de automatización, los relevadores brindan seguridad a los controladores electrónicos en la conexión de periféricos [32].

Se propone además de los relevadores, el uso de un transistor en *Darlington* del tipo *NPN* por la robustez en su diseño. Además, el uso de un par de diodos para evitar el flujo de corrientes en sentidos no deseados.

El circuito equivalente del *TIP122* se muestra a continuación:

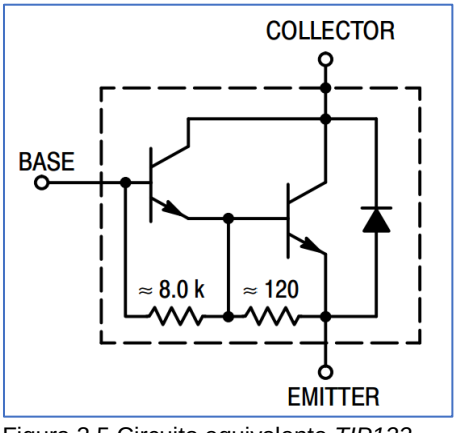


Figura 3.5 Circuito equivalente *TIP122.*

Se puede observar que la ganancia β será mucho mayor que un solo transistor y, además, el voltaje V_{BE} será el doble. Las ecuaciones que representan esta configuración son las siguientes:

$$\beta_{Darlington} = \beta_1 \cdot \beta_2 + \beta_1 + \beta_2$$

$$V_{BE} = V_{BE1} + V_{BE2} \approx 2V_{BE1}$$

La tensión de saturación también aumenta:

$$V_{CE2} = V_{BE2} + V_{CE1}$$

Esto eleva drásticamente el valor de saturación hasta aproximadamente 2V en el *TIP122*, teniendo en cuenta que normalmente este voltaje es de 0.2V para un único transistor de silicio.

La velocidad de conmutación de esta configuración suele ser lenta, sin embargo, se ha acoplado un par de resistencias en derivación para agilizar la descarga de la carga acumulada entre la base y emisor [18,37].

En la Figura 3.5 se muestra el diagrama esquemático de una salida digital propuesta en este proyecto de *PLC.* Tomando en cuenta que su uso será didáctico, el diseño final solo cuenta con cuatro salidas idénticas a la mostrada, las cuáles son suficientes para la emulación de un controlador pequeño.

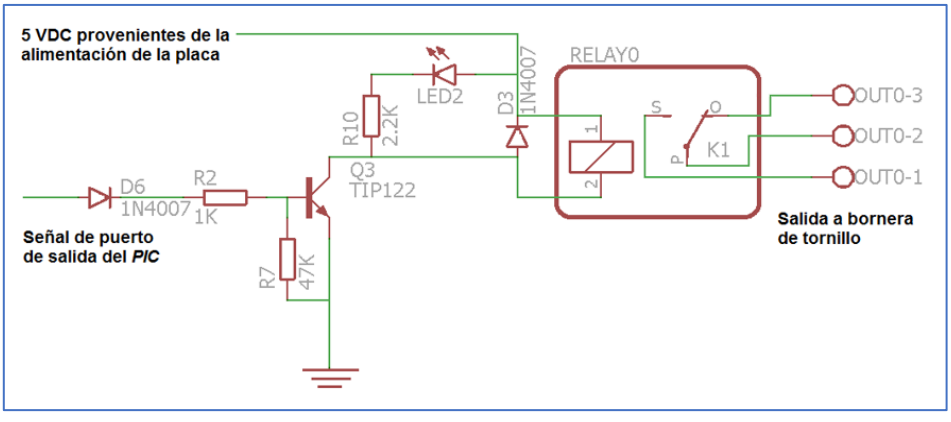


Figura 3.6 Diagrama esquemático de salida digital.

3.2.4 Módulo PWM

La mayoría de los procesos industriales solo requieren de dos estados (*on/off)*, esto es limitado para un control que requiere ser más minucioso. El uso del *PWM* sirve para regular la intensidad de energía aplicada en un sistema [3]. Usualmente esta modulación

se ocupa para el control de motores de corriente directa en procesos industriales.

La modulación por ancho de pulso, como su nombre lo indica, regula el suministro de energía dependiendo de la proporción en el estado alto respecto al bajo como se puede observar en la siguiente figura, el promedio de esa relación es la tensión que proveerá el circuito [10].

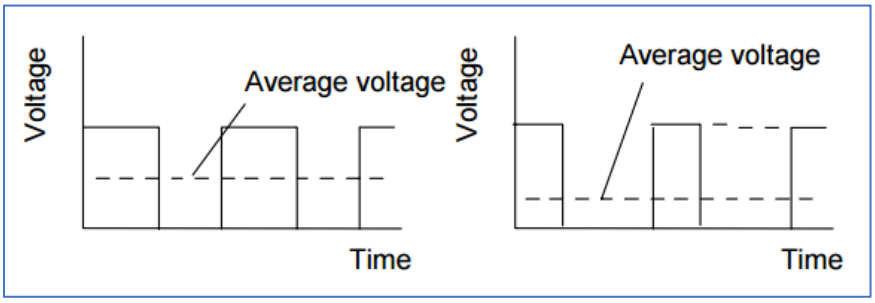


Figura 3.7 Modulación por ancho de pulso.

La generación de estas señales en el *PIC* requiere de tres factores: el periodo, el ancho de pulso y la proporción del ciclo de trabajo. Cada uno de estos se calcula con las siguientes ecuaciones [28]:

$$T_{PWM} = [PR2 + 1] \cdot 4 \cdot T_{OSC} \cdot TMR2_{Prescaler}$$

$$Ancho\ de\ pulso = (RegistroCCP) \cdot T_{OSC} \cdot (TMR2_{Prescaler})$$

$$Proporción\ de\ ciclo\ = \frac{RegistroCCP}{4\ (PR2 + 1)}$$

Donde:

- ***PR2*** es el registro donde se especifica el periodo del *Timer2*
- ***Tosc*** es el periodo del oscilador principal
- ***TMR2Prescaler*** es el valor del preescalador del *Timer2*
- ***RegistroCCP*** es el registro donde se almacena el valor del ciclo de trabajo.

La resolución del PWM puede ser cambiada ajustando el valor de *PR2,* sin embargo, hay que tener en cuenta que la resolución máxima soportada por el *PIC* para el módulo *PWM* es de 10 *bits.* La siguiente ecuación muestra el cálculo para la resolución:

$$Resolución = \frac{\log[4(PR2+1)]}{\log(2)}\ bits$$

Los *PLC* generalmente no cuentan con estos módulos, sin embargo, la implementación de estos, es una herramienta básica para aquellos que deseen aprender el control de sistemas precisos.

El diseño en este proyecto cuenta con dos salidas de señales *PWM* las cuales no contemplan algún dispositivo de protección, por lo que, si se desea conectar algún actuador, el circuito de protección debe ser implementado fuera de la placa.

3.3 Módulo de programación

Los *PLC* tienen varias formas de ser programados, la más usual de ellas es por medio de una *PC*. La comunicación se lleva a cabo a través del puerto serie *RS-232* (*hyperterminal*) o de algún puerto *USB.* Para poder entablar comunicación, cada dispositivo cuenta con los controladores necesarios para ser reconocido por el computador. Una vez entablada la comunicación, se graban las instrucciones de manera secuencial en la memoria de programa del *PLC* para que este las ejecute cuando se encuentre en el modo ejecución [4,6].

Para este proyecto, el módulo de programación estará formado por tres partes:

1. Puente *USB* para entablar comunicación con la tarjeta.
2. *Firmware* en forma *bootloader* para grabar las instrucciones en la memoria de programa del *PIC*.
3. *Software* para transmisión de instrucciones en formato hexadecimal.

La implementación del último punto será explicada detalladamente en el **Capítulo 4**.

3.3.1 Puente USB

La elección de este dispositivo tuvo como criterio la búsqueda de un circuito que fuese barato, de instalación y montaje sencillos y además pudiera ser usado en otros circuitos además de esta propuesta de *PLC*.

Se optó por un puente *USB-UART* con *FT-232*, ya que cumple con todos los criterios antes mencionados. Su precio oscila entre los $130 MXN en MercadoLibre en noviembre del 2016. Por otra parte, puede ser implementado en otros proyectos de manera fácil y rápida. En la siguiente figura se muestra este dispositivo.

Figura 3.8 Puente *USB-UART* con *FT-232.*

Este dispositivo crea un puerto *COM* en la *PC* que permite la comunicación serial a través de sus pines (5V, 3.3V, Tx, Rx, GND). La conexión con la tarjeta deberá ser de la siguiente forma:

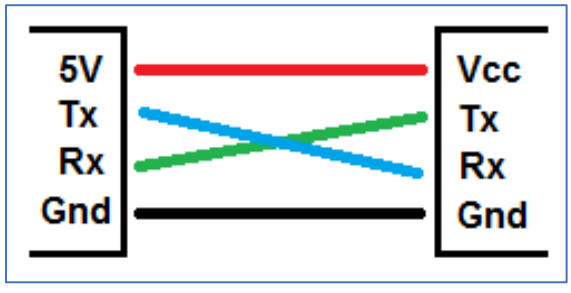


Figura 3.9 Conexión del puente y la tarjeta.

La instalación del *driver* necesario para su funcionamiento se explica en el Anexo.

3.3.2 Introducción al bootloader

Un *bootloader* o arrancador es un software que se ejecuta después de un reset de sistema, este *firmware* ofrece la posibilidad a la plataforma de realizar diferentes funciones grabadas en diferentes partes de la memoria *flash* [40].

Un *bootloader* como grabador, es un firmware que es capaz de grabar datos en la memoria de programa de un microcontrolador y ejecutarlos [30]. En otras palabras, es un programa que guarda programas en la memoria *flash*. Su uso no es nuevo, existen plataformas que lo usan como base para su funcionamiento, como Arduino.

El uso de esta herramienta de grabado es útil para este proyecto, ya que el módulo de programación es similar al de un *PLC* comercial, ambos contando con modo de programación y ejecución, comunicación serial y grabado secuencial de instrucciones en la memoria de programa.

3.3.3 Estructura de bootloader para grabado

Para crear un *bootloader* con la función de grabado hay que tener en cuenta varias consideraciones que son requeridas para que tenga un óptimo funcionamiento:

1. Es necesario un módulo de comunicación que interactúe con el microcontrolador y la *PC*.
2. Tener un método para la selección de grabado o ejecución.
3. Contar o crear algún programa de *PC* que procese y transmita los *bytes* de un archivo hexadecimal hacia el *PIC*.
4. El microcontrolador debe tener la posibilidad de escribir, leer y borrar la memoria *flash* y *EEPROM* sin necesidad de un grabador convencional.
5. El *firmware* debe contar con una protección de ejecución de grabado y de programa principal.
6. Contar con una protección de sobre escritura en el fragmento de memoria *flash* en donde se almacena el *bootloader* [39].

Generalmente el *bootloader* se almacena en la parte baja de la memoria de programa como se muestra en la Figura 3.10.

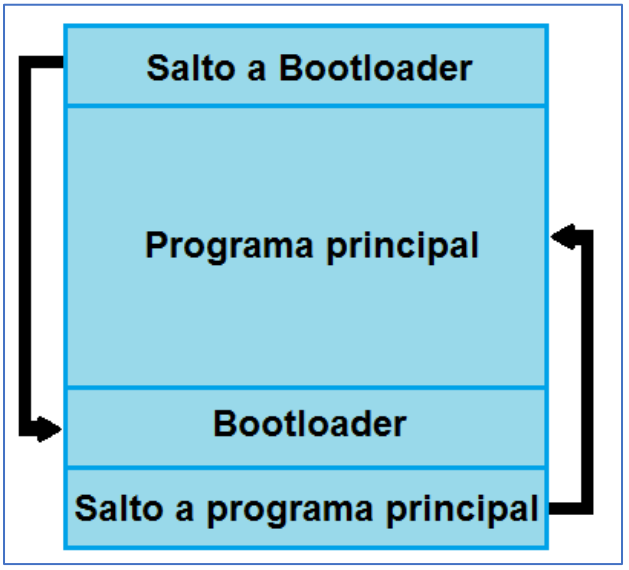


Figura 3.10 Estructura de *bootloader.*

Como se observa, un *bootloader* se divide en cuatro secciones, las cuales se explicarán siguiendo el orden de las flechas de la figura:

1. **Salto a *bootloader*:** Esta sección contiene el cambio a la página de memoria donde se localiza el *bootloader* y el salto a la dirección donde se almacena.
2. ***Bootloader:*** Aquí se almacenan las configuraciones, la rutina de recepción de *bytes*, el grabado de la memoria *flash* y el método de selección.
3. **Salto a programa principal:** Este salto es resultado de la decisión tomada en el método de selección y es el cambio al modo ejecución.
4. **Programa principal:** En esta sección de la memoria se almacena el programa grabado anteriormente por el *bootloader*. Es el programa principal que ejecutará el microcontrolador.

Complementando, un cambio de página en la memoria, es un incremento en el registro *PCLATH* el cual complementa a *PCL*, en ambos registros se almacena el contador de programa. Ya que *PCL* tiene un ancho de 8 *bits,* cada cambio de página se dará

cuando este alcance la dirección *0xFF* hasta llegar al tope de la memoria *flash.* Para el *PIC16f887,* la memoria flash es de 8k, por lo que la última dirección es *0x1FFF* [28]. Con esto se concluye que se cuenta con *0x1F* o 31 páginas. En la Figura 3.9 se muestra la distribución de registros del *PC* (*Program Counter*).

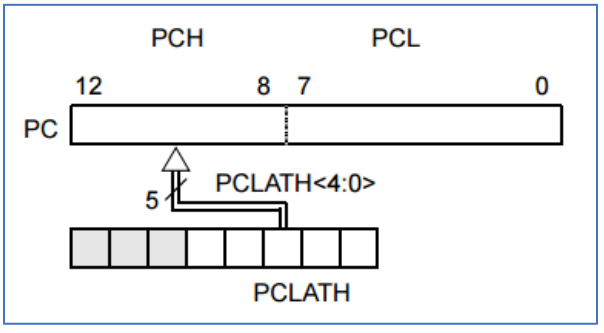


Figura 3.11 Distribución de *PC.*

Los cambios de página cuando el *PIC* está en funcionamiento son realizados automáticamente, sin embargo, para realizar saltos hacia una página de memoria distinta a la que se encuentra en ejecución en ese momento, es necesario modificar manualmente el valor del registro *PCLATH* en el código *ASM*.

El método de selección de ejecución, es una toma de decisión que ejecuta el *PIC* para hacer el salto a la rutina de grabado de la memoria *flash* o al programa de usuario. Esta selección puede ser a través de software como lo realiza Arduino o a través de hardware, lo más usual para esta tarea suele ser a través de hardware, tomando la decisión si el estado de un pin se encuentra en uno u otro [30,39]. En este proyecto se implementará a través de hardware con un interruptor deslizable de dos estados. El diagrama implementado para este selector se puede observar en el **Tema 3.5** en la Figura 3.16.

3.3.4 Propuesta de bootloader

El arrancador que se implementa en este proyecto es programado en la plataforma *PIC Simulator IDE 6.83* con la herramienta *assembler*. En los siguientes diagramas se describe el diseño del *bootloader* que se propone.

Hay algunos datos importantes por mencionar:

- La configuración del módulo *USART* es: BaudRate a 9600, 8 bits de datos, sin paridad y 1 bit de parada.
- El método de selección de ejecución está remarcado en color azul.
- El protocolo de escritura en la memoria *flash* está remarcado en color rojo, este se describirá más adelante.

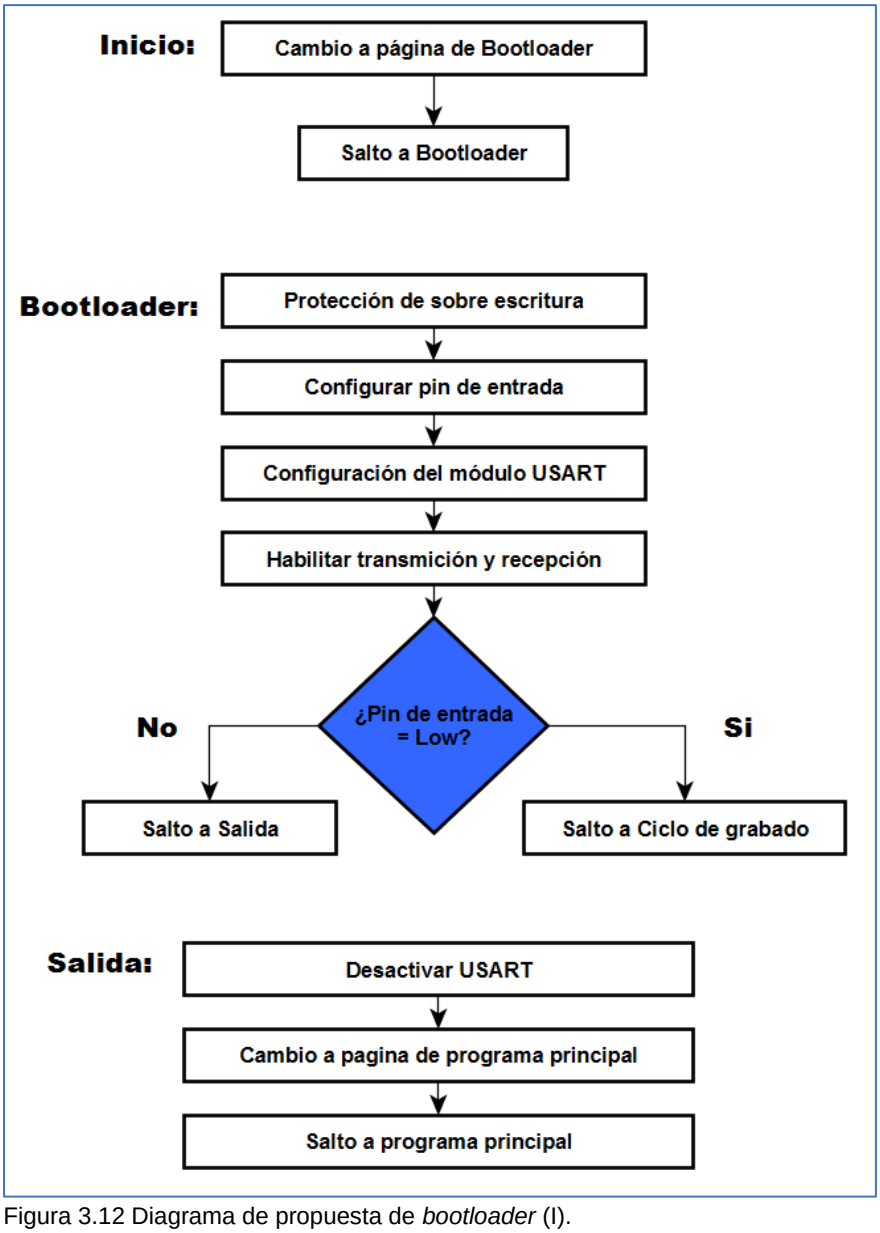


Figura 3.12 Diagrama de propuesta de *bootloader* (I).

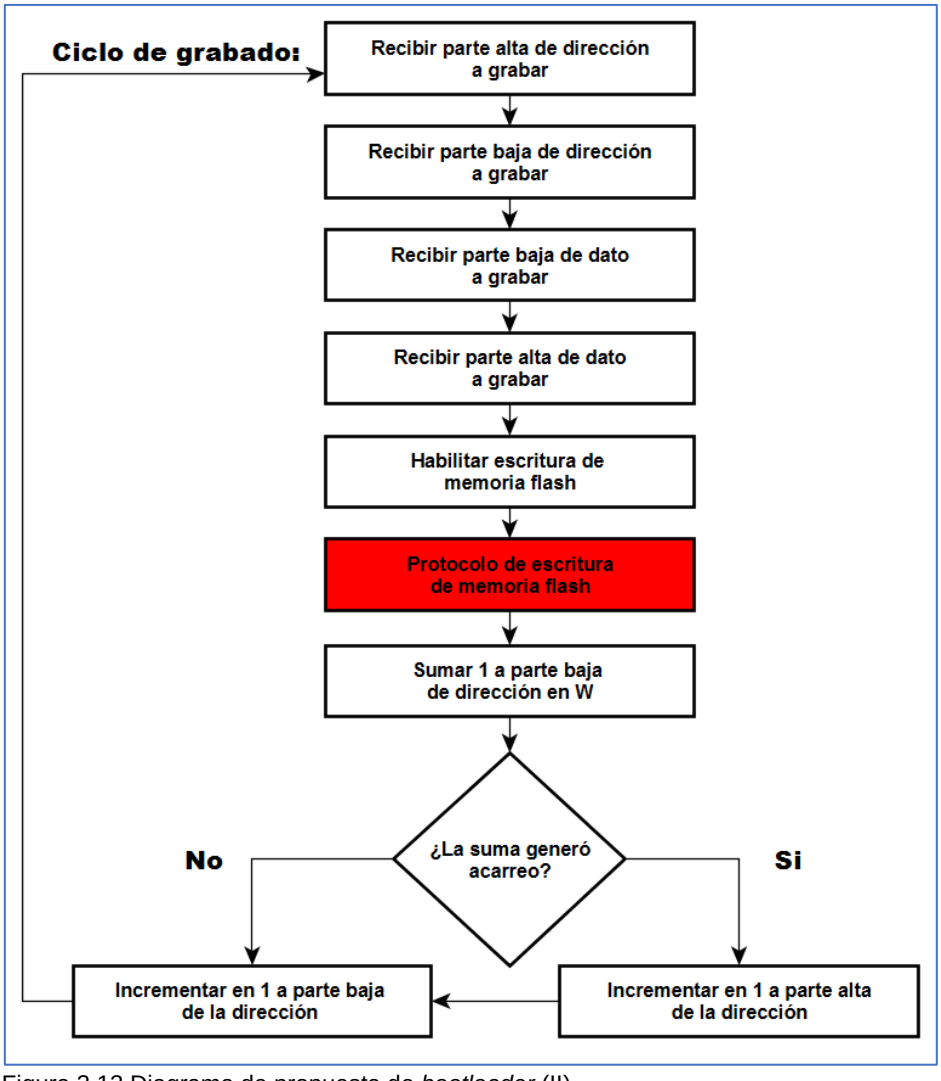


Figura 3.13 Diagrama de propuesta de *bootloader* (II).

El diseño de este *bootloader* cumple con los elementos necesarios presentados en el **Subtema 3.3.3.** El diseño es sencillo, pero cumple con su función correctamente, se han realizado alrededor de 100 pruebas de grabado con el *bootloader* terminado y en todas se ha presentado una escritura exitosa en la memoria *flash*.

3.3.5 Escritura en memoria flash

El *datasheet* del *PIC16f887* menciona que; la programación en la memoria *flash* se escribe por bloques de dieciséis palabras con

dirección secuencial. La dirección se almacena en los registros *EEADR* y *EEADRH,* mientras que cada palabra que será grabada, en los registros *EEDATA* y *EEDATH* [28]. Estos cuatro registros se localizan en el *Banco2* de los registros de funciones especiales del microcontrolador. En la Figura 3.14 se muestra un fragmento de estos registros del *PIC* y se señalan en color rojo los mencionados anteriormente.

PIC16F887 SPECIAL FUNCTION REGISTERS

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.	00h	Indirect addr.	80h	Indirect addr.	100h	Indirect addr.	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	WDTCON	105h	SRCON	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	CM1CON0	107h	BAUDCTL	187h
PORTD(2)	08h	TRISD(2)	88h	CM2CON0	108h	ANSEL	188h
PORTE	09h	TRISE	89h	CM2CON1	109h	ANSELH	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDAT	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved	18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh	Reserved	18Fh
T1CON	10h	OSCTUNE	90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	WPUB	95h		115h		195h
CCPR1H	16h	IOCB	96h	General Purpose Registers	116h	General Purpose Registers	196h
CCP1CON	17h	VRCON	97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h	16 Bytes	119h	16 Bytes	199h

Figura 3.14 Fragmento de registros especiales del *PIC16f887.*

Como protección la memoria *flash* requiere de una secuencia o protocolo de eventos antes de iniciar la escritura:

1. Poner a 1 el bit *EEPGD* del registro *EECON1.*
2. Escribir 55h (55 hexadecimal), después AAh en el registro *EECON2.*
3. Poner a 1 el bit *WR* del registro *EECON1*.
4. Poner en no operación el procesador durante dos ciclos [28].

Los dos ciclos que el procesador se encuentra en no operación es el tiempo que se requiere para grabar o escribir en la memoria *flash.* Los registros utilizados para el protocolo se localizan en el *Banco3* y se muestran en la Figura 3.14 señalados en color azul.

En la siguiente figura se muestra un fragmento del código de ejemplo para grabado de memoria *flash* extraído del *datasheet* del *PIC16f887*, en donde se puede observar este protocolo.

```
BSF     EECON1,EEPGD        ; Point to program memory
BSF     EECON1,WREN         ; Enable writes
BCF     INTCON,GIE          ; Disable interrupts (if using)
MOVLW   55h                 ; Start of required write sequence:
MOVWF   EECON2              ; Write 55h
MOVLW   AAh                 ;
MOVWF   EECON2              ; Write AAh
BSF     EECON1,WR           ; Set WR bit to begin write
NOP                         ; Any instructions here are ignored as processor
                            ; halts to begin write sequence
NOP                         ; processor will stop here and wait for write complete
                            ; after write processor continues with 3rd instruction
BCF     EECON1,WREN         ; Disable writes
```

Figura 3.15 Protocolo de grabado de memoria *flash.*

Como se mencionó, la escritura se hace a través de bloques, los cuales están distribuidos en ocho *buffers* de datos (*EEDATH* y *EEDATA*). La escritura de cada uno de estos se logra sí y solo sí se han completado los ocho *buffers*, de lo contrario la grabación no se realiza y el contenido de ese bloque de memoria se vuelve 0 en cada uno de los bits. La distribución de los *buffers* se puede observar en la Figura 3.16 [28].

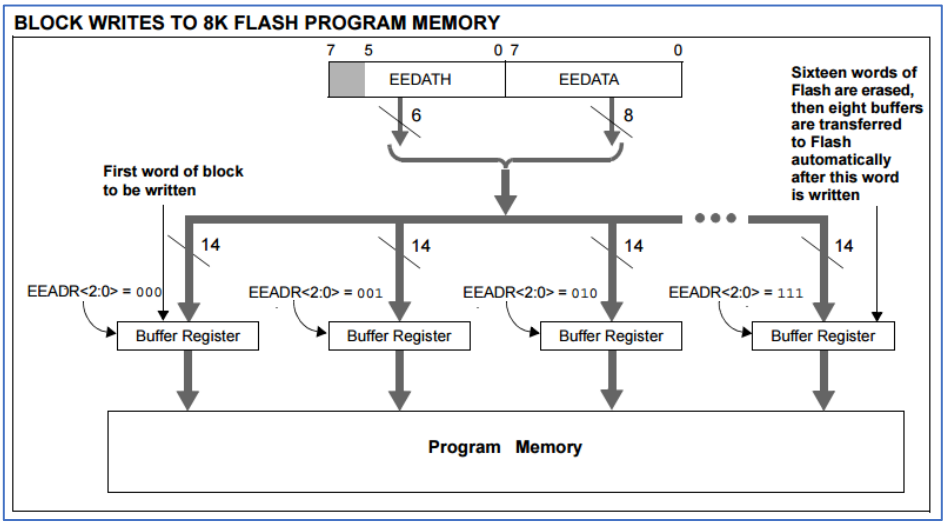


Figura 3.16 Distribución de *buffers.*

3.4 Descripción de *pines*

El *PIC16f887* en la versión *P-DIP* cuenta con 40 pines, se optó por este microcontrolador ya que es el *PIC* que tiene más capacidad de memoria *flash* y puertos CCP para *PWM,* de los dispositivos

soportados por el entorno *LDmicro* [13]. La descripción de los pines de interconexión con la tarjeta electrónica para este proyecto se muestra a continuación.

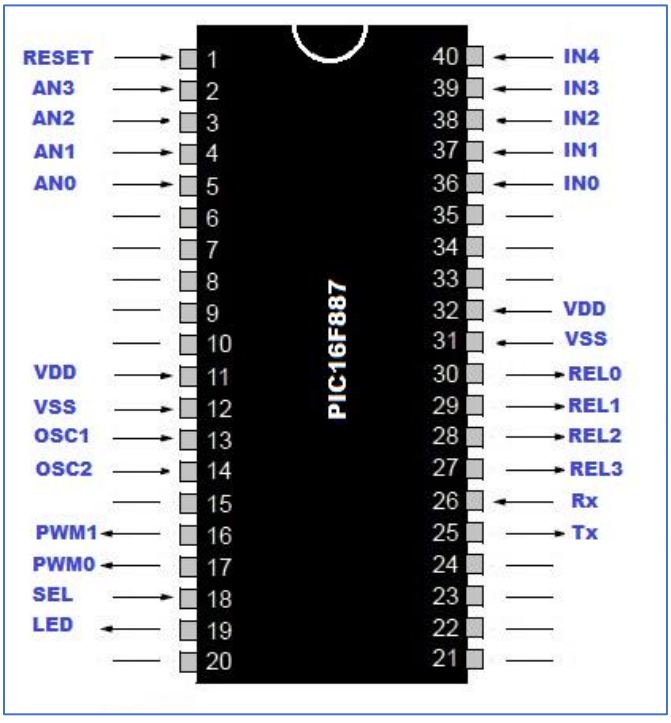


Figura 3.17 Descripción de *pines*.

Donde:

- ***RESET*****,** es el pin de reset.
- ***ANx,*** son las entradas del módulo *ADC.*
- ***Vxx,*** Son los *pines* de alimentación del *PIC.*
- ***OSCx,*** son las conexiones al oscilador externo.
- ***PWMx*****,** son las salidas al módulo *PWM.*
- ***SEL,*** es el pin de selección de grabado o ejecución.
- ***LED,*** es un indicador luminico el cual enciende cuando se está en modo grabación.
- ***INx,*** son las entradas digitales optocopladas.
- ***RELx,*** son las salidas digitales a relevador.
- ***Rx* y *Tx,*** son los pines del módulo *USART*.

3.5 Diagrama esquemático del *PIC*

Con la descripción del uso de pines, el diagrama esquemático del *PIC* queda de la siguiente forma:

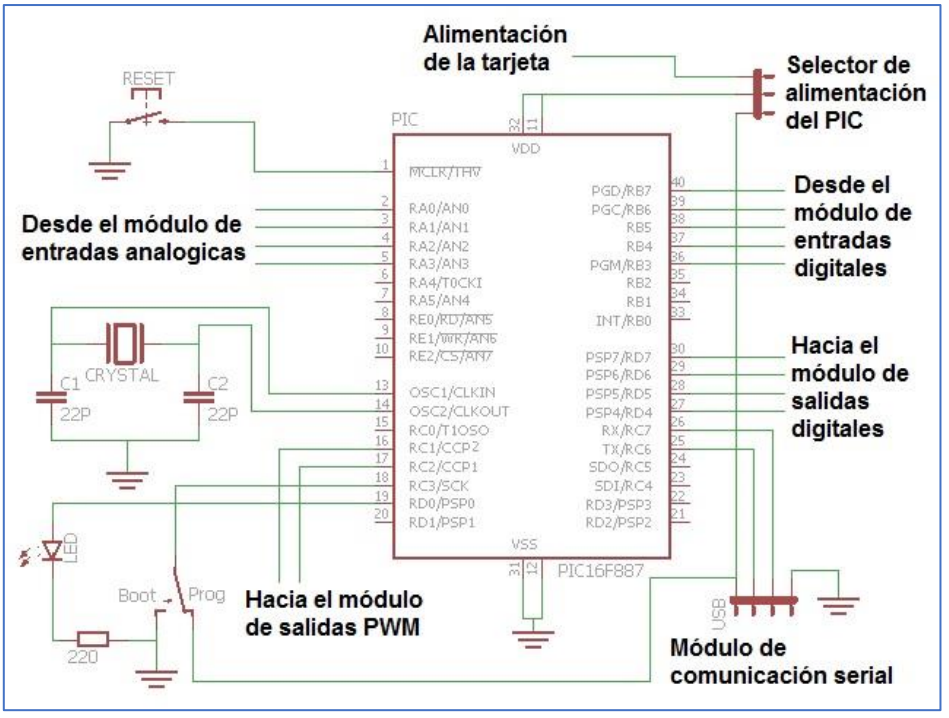


Figura 3.18 Diagrama esquemático del *PIC16f887.*

Donde:

- El valor del cristal de la tarjeta es de 20Mhz.
- El módulo de comunicación son cuatro *jumpers* macho en el orden de izquierda a derecha: *Vcc, Tx, Rx, Gnd*.
- El selector de alimentación permite elegir entre la alimentación de la placa o de módulo de comunicación, para suministrar energía al *PIC*.

La alimentación del puerto serie (*Vcc*) es recomendable usarla sólo si se utiliza el modo grabación del *bootloader*, para usar el *PLC* en modo ejecución se recomienda hacer uso la alimentación de la tarjeta.

3.6 Fuente de alimentación

Como se mencionó en el **Capítulo 2**, esta sección de un *PLC* se encargará de convertir los $127V_{AC}$ de la acometida eléctrica a $5V_{Dc}$. El diseño de esta sección debe ser práctica, que no ocupase mucho espacio y fuese de bajo costo. Por lo que implementar una fuente en la placa no era conveniente, ya que el transformador y la circuitería ocuparían bastante espacio y elevaría el costo de la placa.

La solución por la que se optó para resolver esta problemática fue el uso de un eliminador con puerto *USB* que se conecta a la tarjeta a través de un conector tipo B. Estos eliminadores son incluidos en la mayoría de los teléfonos celulares al momento de adquirirlos actualmente. Se optó por este tipo de alimentación ya que estos circuitos suministran $5V_{DC}$, su corriente máxima llega a variar desde 400*mA* hasta 2*A*. Ya que por ahora no se cuenta con una protección para sobre voltajes o sobre corrientes, el uso de un eliminador que suministre específicamente estos valores es necesario para el funcionamiento del circuito y evitar que los dispositivos electrónicos se dañen. El diagrama propuesto para alimentar la placa se muestra a continuación:

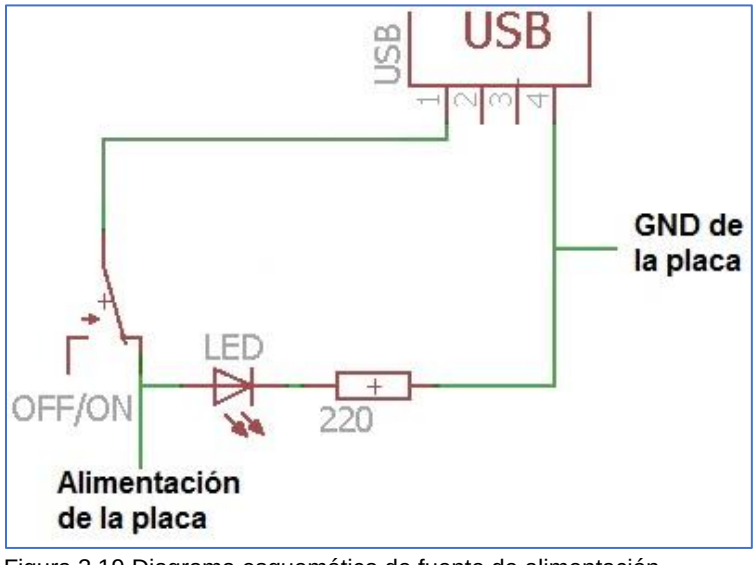


Figura 3.19 Diagrama esquemático de fuente de alimentación.

3.7 Circuito implementado

Unificando todo lo redactado con anterioridad, se tiene una tarjeta electrónica compacta, versátil y relativamente rápida. Esta placa se puede describir en bloques como se muestra en la siguiente figura:

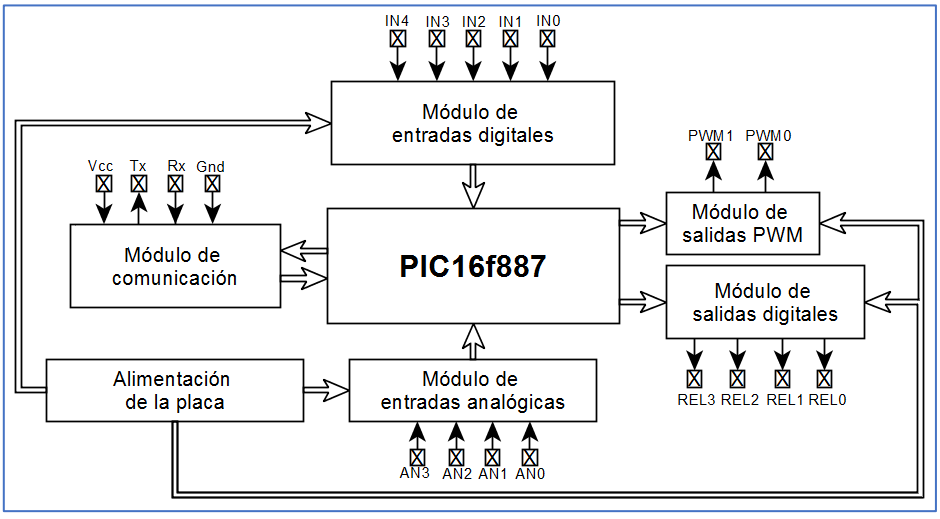


Figura 3.20 Diagrama a bloques de *PLC* propuesto.

A continuación, se muestra la tarjeta electrónica ensamblada y el orden de entradas y salidas.

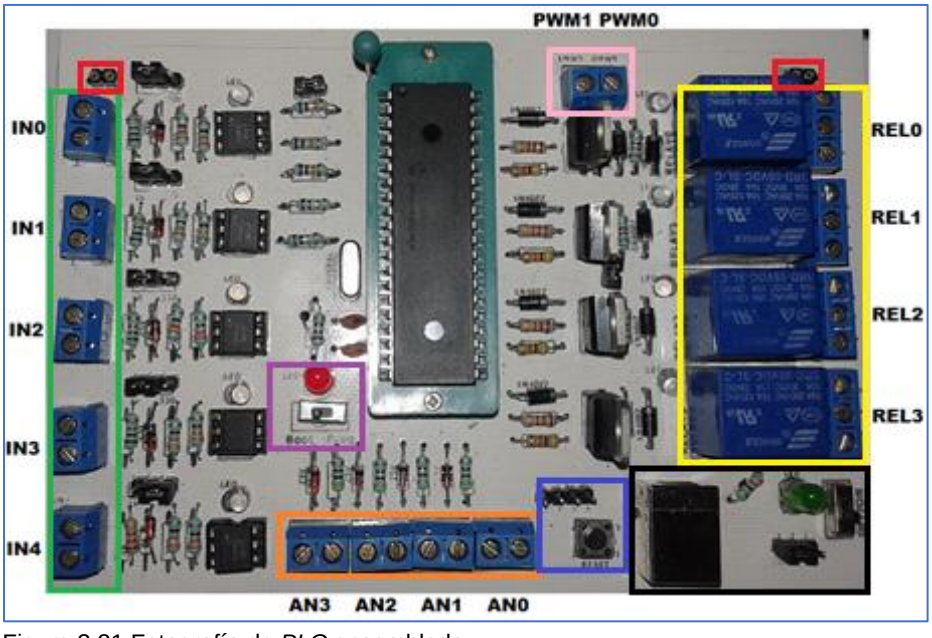


Figura 3.21 Fotografía de *PLC* ensamblado.

Donde los recuadros representan según su color:

- **Verde:** Entradas digitales.
- **Naranja:** Entradas analógicas.
- **Amarillo:** Salidas digitales.
- **Rosa:** Módulos de salida *PWM*.
- **Rojo:** Conexiones de alimentación de la placa.
- **Morado:** Led de estado de *bootloader* y el interruptor de selección.
- **Azul:** *Jumpers* para el módulo de comunicación y el botón de *RESET.*
- **Negro:** La alimentación de la placa y el selector de alimentación del *PIC*

La alimentación mediante un eliminador y un conector USB se presenta en la siguiente imagen:

Figura 3.22 Fotografía de fuente de alimentación.

Las pistas y la localización de componentes se muestran en las siguientes figuras. El diseño de la tarjeta se realizó en *Eagle v7.6*, los archivos para impresión se localizan en el enlace https://goo.gl/JhX1p4

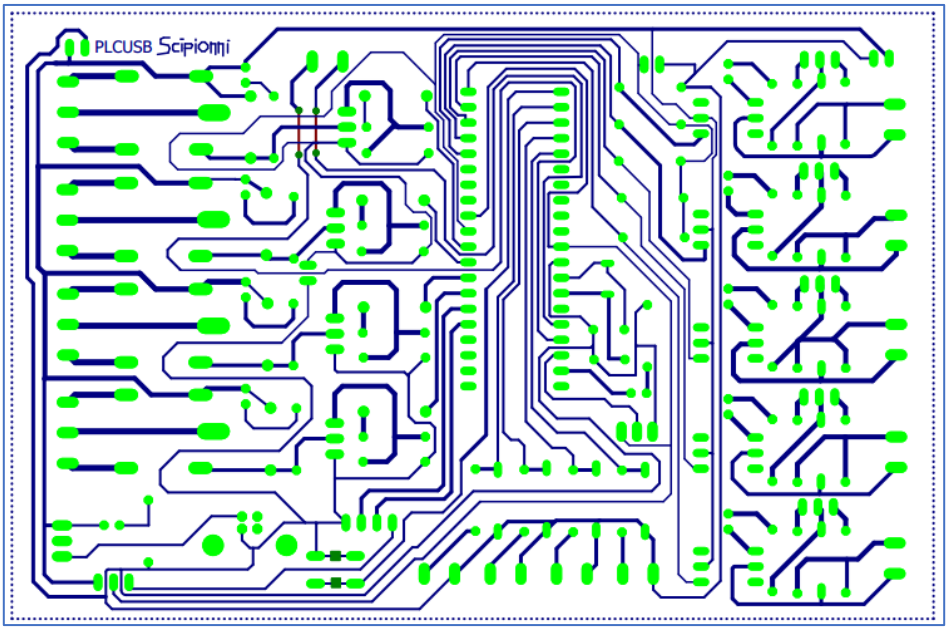


Figura 3.23 Pistas de la tarjeta

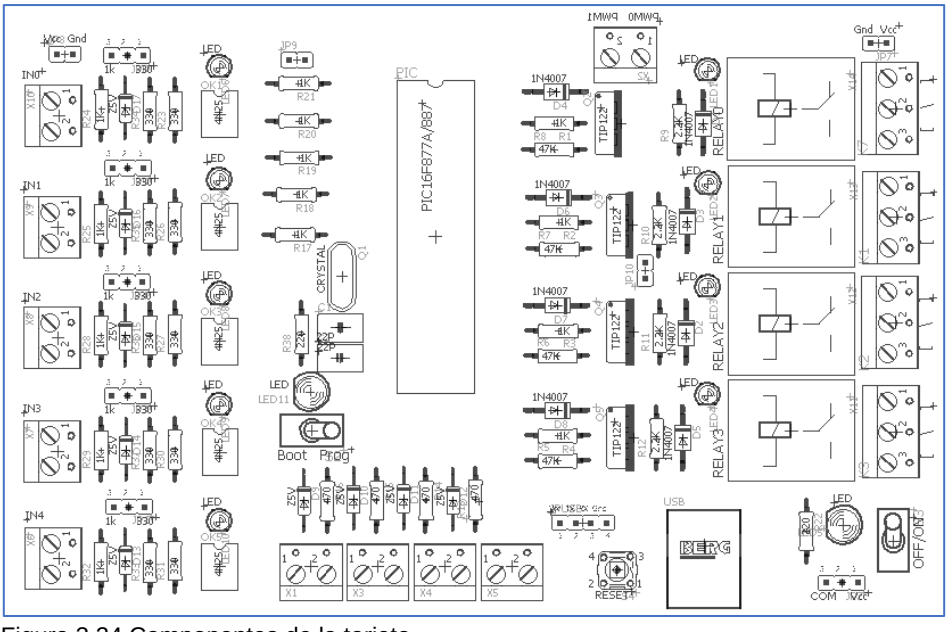


Figura 3.24 Componentes de la tarjeta

Capítulo 4
Programación gráfica del PLC

4.1 Visual Basic .NET 2010 Express

Hoy en día existen varias opciones de compiladores de *Windows Forms*, para este proyecto se usará *Visual Basic 2010* en su versión *Express*. La versión profesional de este *software* tiene costo, pero la versión *Express* es gratuita y se puede descargar directamente en el enlace *www.microsoft.com/*express.

La versión 2010 E*xpress* cuenta con las herramientas de la versión profesional para el desarrollo de *Windows Forms*, su limitación radica en herramientas para equipos de desarrollo, como la inclusión de *SQL,* plantillas de bibliotecas de control web, *Crystal Reports* de *ASP.NET* [41], por mencionar algunas. Para este proyecto no es necesario contar con estas características extras. Por lo que el uso de la versión *Express* basta.

La aplicación se desarrolló en *Visual Basic 2010*, sin embargo, esta puede ser visualizada y modificada en versiones posteriores. Para que la aplicación pueda ser ejecutada, es necesario contar con la versión *4.0* o posterior del *Framework .NET* instalada en el ordenador.

4.1.1 Aplicación desarrollada

En este apartado se explica el funcionamiento y programación de la aplicación creada en *Visual Basic .NET*. El código sobrepasa las 4500 líneas de programación, por esta razón solo se comentan los procesos más importantes con diagramas de bloques.

4.1.2 Ventana principal

En la Figura 4.1 se observa la ventana principal de la aplicación que se desarrolló como interfaz de comunicación para el *PLC* propuesto.

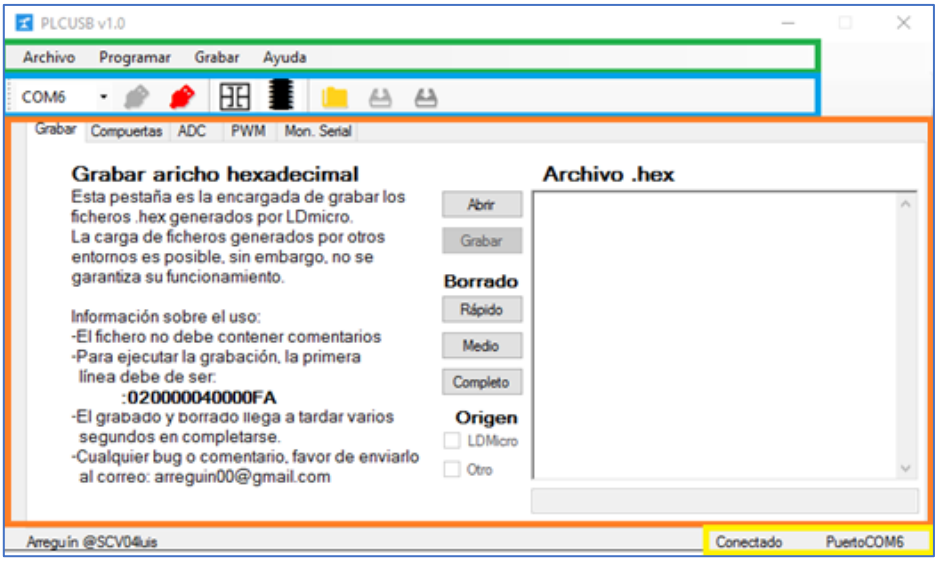
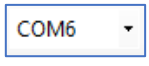


Figura 4.1 Ventana principal.

Podemos observar:

- **Barra de menús:** marcada con color verde. Esta barra contiene lo menús “Archivo, Programar, Grabar y Ayuda”:
  - ❖ Archivo: Tiene la opción de “Conectar, Desconectar y Salir”.
  - ❖ Programar: Se puede ejecutar “LDmicro” y además desplegar la “Descripción de pines”.
  - ❖ Grabar: Permite “Abrir. HEX” y grabar ficheros de “LDmicro” u “Otro” entorno de forma genérica.
  - ❖ Ayuda: Contiene el “Manual” para el uso del programa e información “Acerca de” la aplicación.
- **Barra de herramientas:** marcada en color azul. Esta barra contiene una serie de botones y un *ComboBox*.
  - ❖ 

Este *ComboBox* despliega los puertos *COM* disponibles en el ordenador para entablar comunicación.
  - ❖  Este botón conecta la aplicación con el puerto *COM* seleccionado en el *ComboBox*
  - ❖  Desconecta la aplicación del puerto *COM*
  - ❖  Abre el entorno de programación *LDmicro*

- ❖  Despliega la ventana de visualización de los pines del microcontrolador y el *PLC*
- ❖  Abre una ventana para elegir el fichero *.hex* que desea ser grabado
- ❖  Procesa el fichero con el formato que genera el entorno *LDmicro*
- ❖  Procesa el fichero de forma genérica

- **Pestañas de área de trabajo:** Marcada en color naranja. Esta área contiene cinco pestañas (Grabar, Compuertas*, ADC, PWM* y Mon. Serial), de las cuales se explicará más adelante su funcionamiento.
- **Estado y puerto:** Marcado en color amarillo. Aquí se observa si la aplicación está conectada y en que puerto lo está.

4.2 Pestañas

Las pestañas permiten interactuar entre las diversas funciones y programas pregrabados que ofrece esta aplicación. Su diseño brinda un uso sencillo. En los siguientes subtemas se explicarán cada una de estos apartados.

Los programas pregrabados están formados por dos partes, la transmisión de las configuraciones seleccionadas en la aplicación y el programa que ejecuta el microcontrolador. La transmisión de configuraciones se explica en este tema y la ejecución de estas configuraciones en el *PIC,* en el **Tema 4.4.** Además, para ejecutar uno de estos programas, es necesario tener el *PLC* en modo grabación, ya que se han transmitido las configuraciones se puede pasar a modo ejecución y ejecutar los programas.

4.2.1 Pestaña Compuertas

Esta pestaña tiene la función de programar al *PLC* para que emule una serie de compuertas lógicas (*AND, OR, XOR* y *NOT*) en la tarjeta. En el *ComboBox* se puede elegir una combinación entre las entradas y dependiendo la interacción de la combinación, las

salidas reflejarán el resultado. A continuación, se muestra este apartado.

Figura 4.2 Pestaña compuertas.

El botón “Habilitar” permite la manipulación las características de esta y las otras pestañas, además, graba el programa pregrabado en el *PIC* necesario para interactuar con la aplicación.

Cada una de las combinaciones tiene un orden fijo entre las entradas y salidas. Por lo que, no es posible cambiarlas.

Ya que se seleccionó una combinación se puede pulsar el botón “Grabar”, el cual transmite las configuraciones necesarias para que el microcontrolador pueda ejecutar la combinación seleccionada. En la siguiente figura se muestra que carácter se transmite dependiendo la combinación seleccionada.

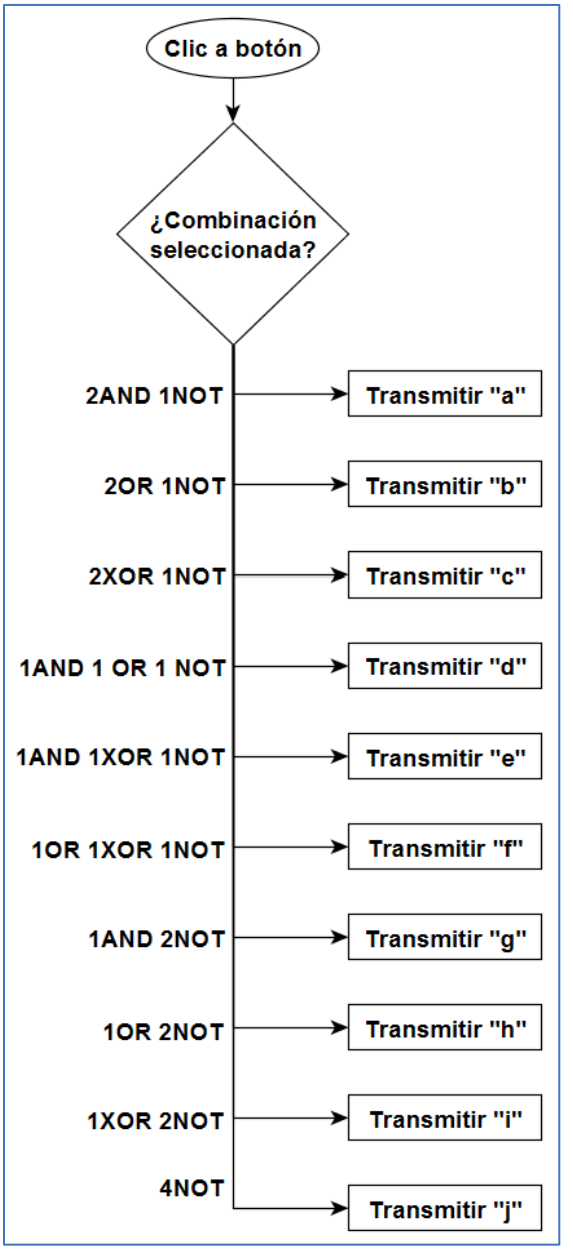


Figura 4.3 Diagrama de botón “Grabar” *Compuertas.*

Para la transmisión de datos a través del puerto serie hacia el microcontrolador se ocupó la clase *SerialPort,* como se mencionó anteriormente, esta cuenta con varios métodos para la transmisión (*Write, Write(Char), Write(Byte)* y *WriteLine*) [36], el envío de caracteres de configuración se lleva a cabo a través del método *Write* [24].

4.2.2 Pestaña ADC

La pestaña *ADC* pretende brindar una herramienta de fácil implementación. Como se observa en la Figura4.4, el diseño es práctico y además cuenta con la posibilidad de visualizar el valor convertido por el microcontrolador.

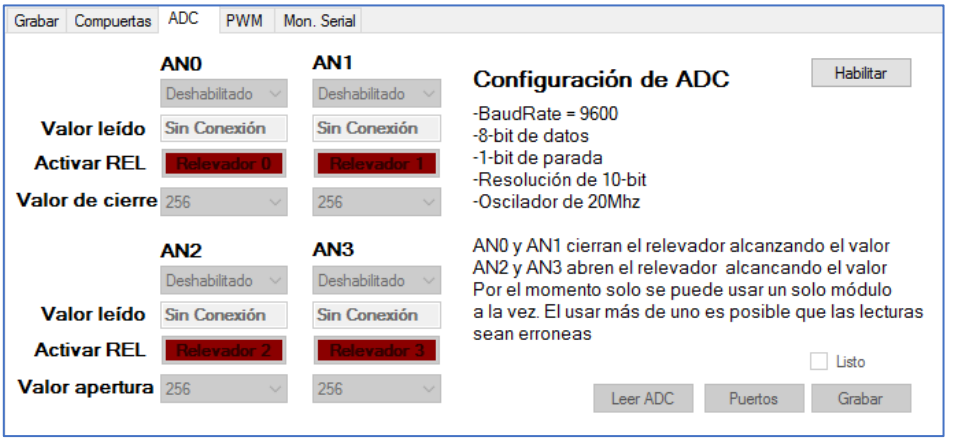


Figura 4.4 Pestaña *ADC.*

Al igual que la pestaña Compuertas, *PWM* y Mon.Serial, se necesita habilitar la pestaña para utilizarla. También, graba un fichero hexadecimal en el *PIC* para que se lleve a cabo la interacción.

Cada módulo de conversión cuenta con su *ComboBox* para seleccionar si está habilitado o deshabilitado, un *TextBox* donde se muestra el valor leído por el *ADC*, el botón para activar o desactivar un relevador y el valor de cierre o apertura de dicho relevador.

Una vez que se selecciona el estado “Habilitado” de un módulo *ADC*, este permite elegir si se activa un relevador o no. En caso de que no se desee interactuar con un relevador, el botón de “Leer *ADC”* permite visualizar el valor leído por el microcontrolador.

Por otra parte, al pulsar el botón “RerlevadorX”, se habilita el uso de un relevador. Además, al habilitarlo, permite seleccionar el valor de cierre o apertura del mismo. Los módulos están acomodados de forma que *AN0* y *AN1* cierran, mientras que *AN2* y *AN3* abren sus respectivos relevadores.

Por ejemplo, se elige el módulo *AN2* con relevador y apertura de 256. Cuando la lectura del *ADC* iguale o sobrepase el valor de

apertura, el *PIC* cambiará el estado de su pin asignado al *Relevador2* de 1 a 0. Por otra parte, si se hubiese elegido el módulo *AN1*, este cerraría el relevador o cambiaría el estado de 0 a 1.

Para desbloquear el botón “Grabar” es necesario marcar el *CheckBox* “Listo”, para eso hay que tener activado al menos un módulo. Una vez pulsado este botón, el programa procesa el estado de cada módulo para transmitir las configuraciones de ejecución al microcontrolador.

En la siguiente figura se muestra el diagrama a bloques del botón “Grabar”, solo se muestra el proceso de *AN0*, sin embargo, los demás módulos siguen la misma secuencia.

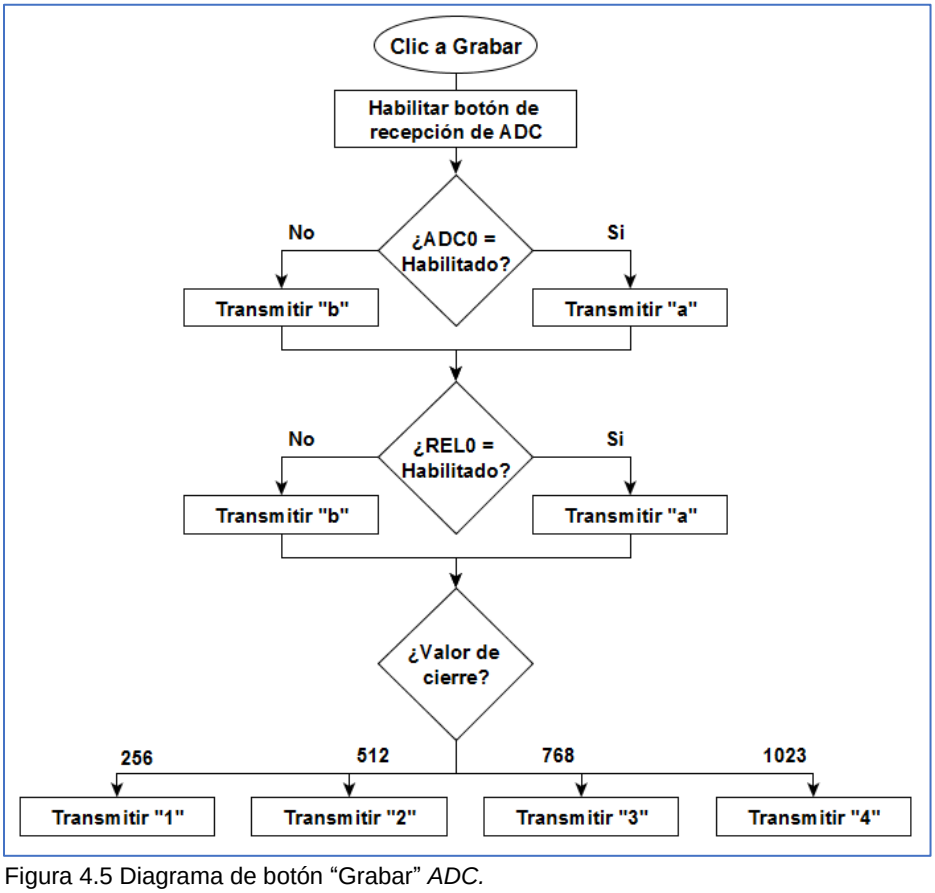


Figura 4.5 Diagrama de botón “Grabar” *ADC.*

Ya que se transmitieron las configuraciones, el botón “Leer *ADC*” se desbloquea. Al pulsarlo se ejecuta una serie de procesos para mostrar el valor de cada módulo correspondiente.

Para poder llevar a cabo esta tarea, se implementó un *Timer*, el cual ejecuta un algoritmo cada vez que se desborda [27]. Para poder distinguir hacia donde se dirige un dato recibido entre la pestaña Mon. Serial y *ADC*, se creó una variable para distinguir entre una y otra. En la Figura 4.6 se muestra el diagrama a bloques de la recepción del valor leído por el *ADC*, solo se muestra la operación del *ADC0*, pero, se realiza lo mismo para los otros módulos.

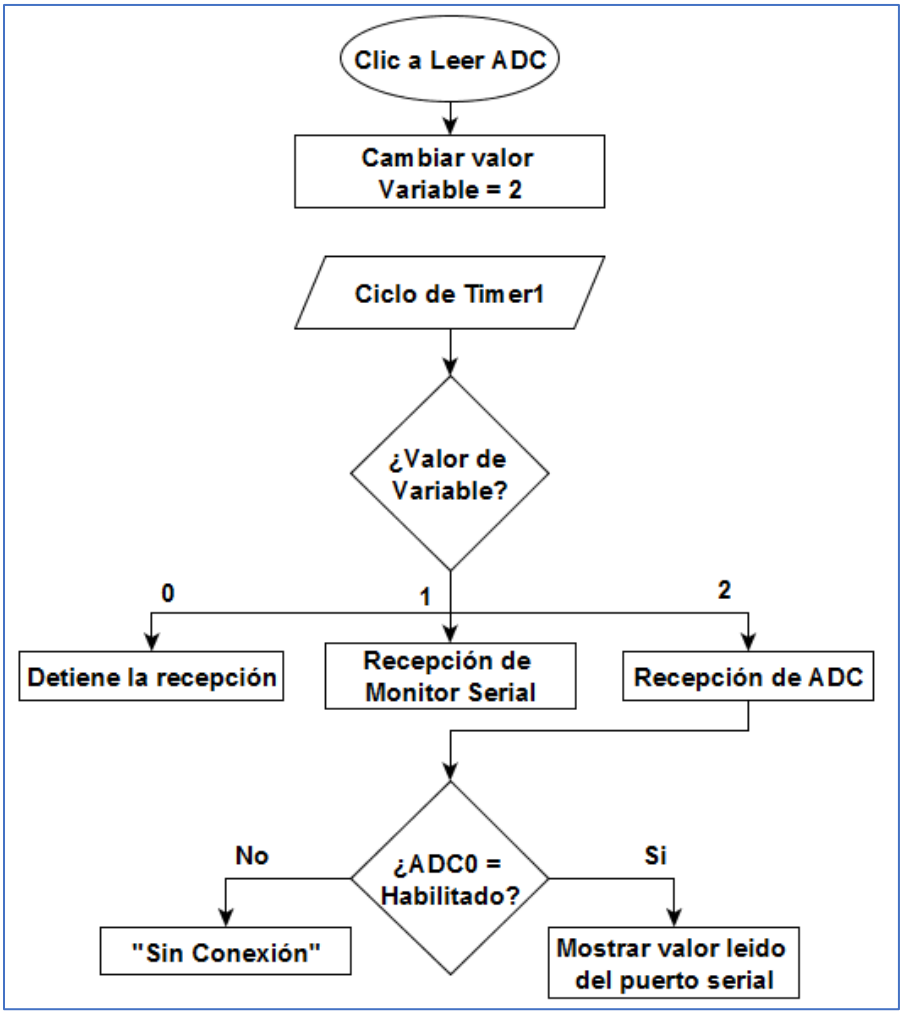


Figura 4.6 Diagrama de botón "Leer *ADC".*

Actualmente su uso está limitado a un puerto a la vez. Se hicieron varias pruebas y en cuanto a las lecturas, al usar más de un módulo, estas son erróneas, aun así, la interfaz cuenta con los cuatro módulos con el propósito de implementarlo con programas desarrollados en *LDmicro*.

4.2.3 Pestaña PWM

En esta pestaña se configuran los módulos *CCP* del *PIC* en su modalidad *PWM*. En la Figura 4.7 se observa esta pestaña:

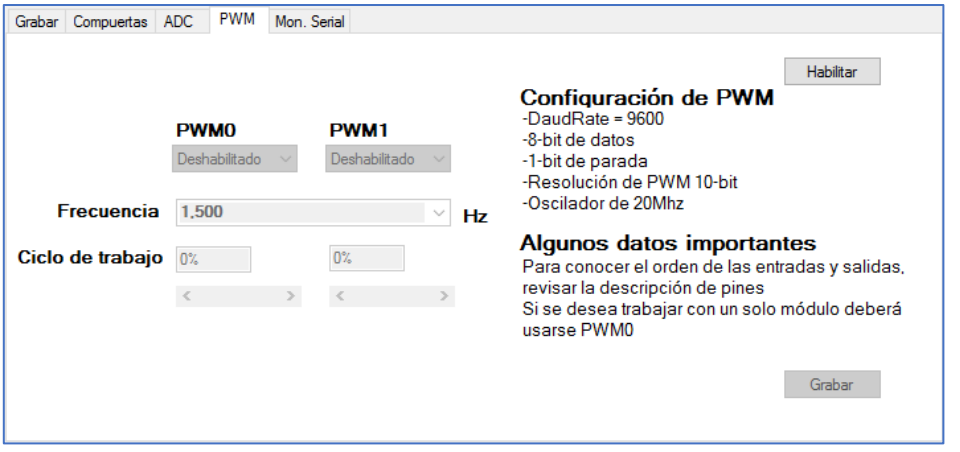


Figura 4.7 Pestaña *PWM.*

Ambos módulos cuentan con un *ComboBox* en el cual se habilita cada uno. La frecuencia es la misma para ambos módulos y se cuenta con tres configuraciones 1500, 2500 y 3500Hz.

Se puede usar ambos módulos independientemente, sin embargo, si se desea hacer uso de un solo módulo, este deberá ser el rotulado como *PWM0*. El uso del segundo módulo solo se utiliza si se desea modular por dos canales.

El botón “Grabar” en esta pestaña se desbloquea cuando se tiene habilitado al menos el módulo *PWM0.* Una vez que se haya pulsado se ejecutará la secuencia de transmisión de las configuraciones hacia el *PIC*, en la Figura 4.8 se representa un diagrama a bloques de este proceso.

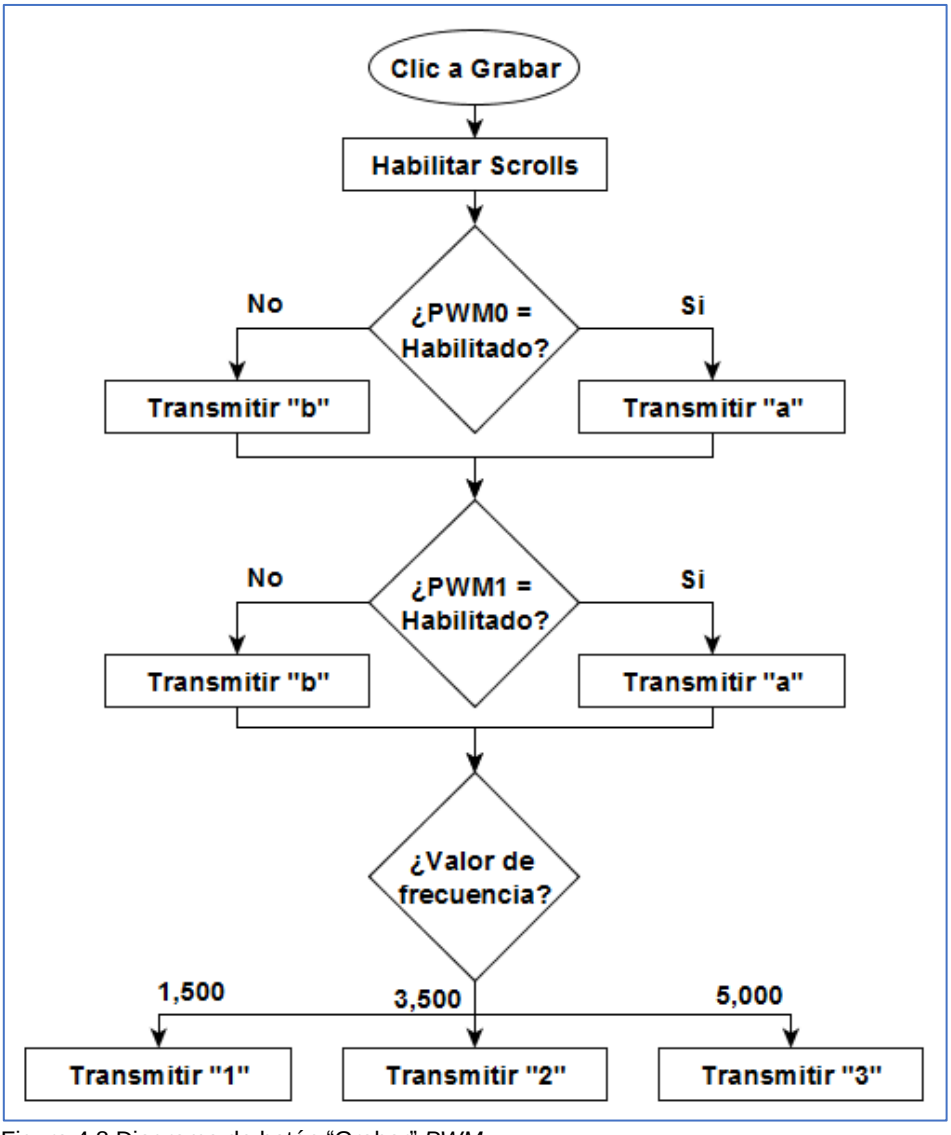


Figura 4.8 Diagrama de botón “Grabar” *PWM.*

Al deslizar la *ScrollBar* se transmite el valor de este en decimal. Fue configurado con límite a diez y cuenta de uno en uno por lo que se tiene once posibilidades de caracteres para transferir (0-10). El *PIC* al recibir cualquiera de estos valores, lo procesa para modificar el ciclo de trabajo deseado en proporciones de 10% en 10% del ciclo de trabajo efectivo.

4.2.4 Pestaña Mon. Serial

La pestaña de Monitor Serial proporciona una herramienta de comunicación efectiva. Surgió como una interfaz de prueba para el *bootloader*, por esta razón su diseño solo transmite caracteres de 1 byte. El enviar más de un carácter colapsa la aplicación por desbordamiento, sin embargo, la recepción no tiene límite de caracteres. A continuación, se muestra esta pestaña.

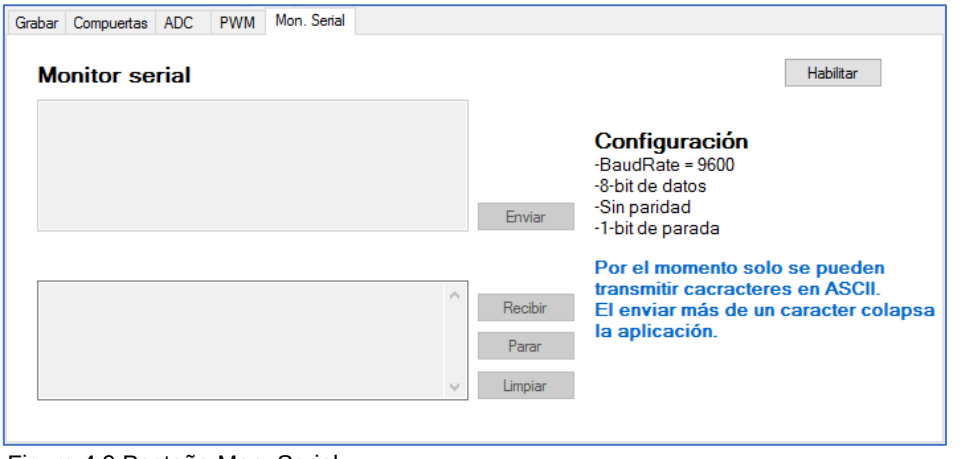


Figura 4.9 Pestaña Mon. Serial.

El botón “Habilitar” en esta pestaña solo activa los botones y cajas de texto implementadas, a diferencia de las otras, no graba ningún programa en el *PIC*.

Para poder entablar comunicación se debe respetar la configuración mostrada en la figura anterior. También, para mostrar datos se hace uso del mismo *Timer* de la pestaña *ADC*, sin embargo, como se observa en la Figura 4.10, el valor de la variable, al pulsar el botón recibir, será 1.

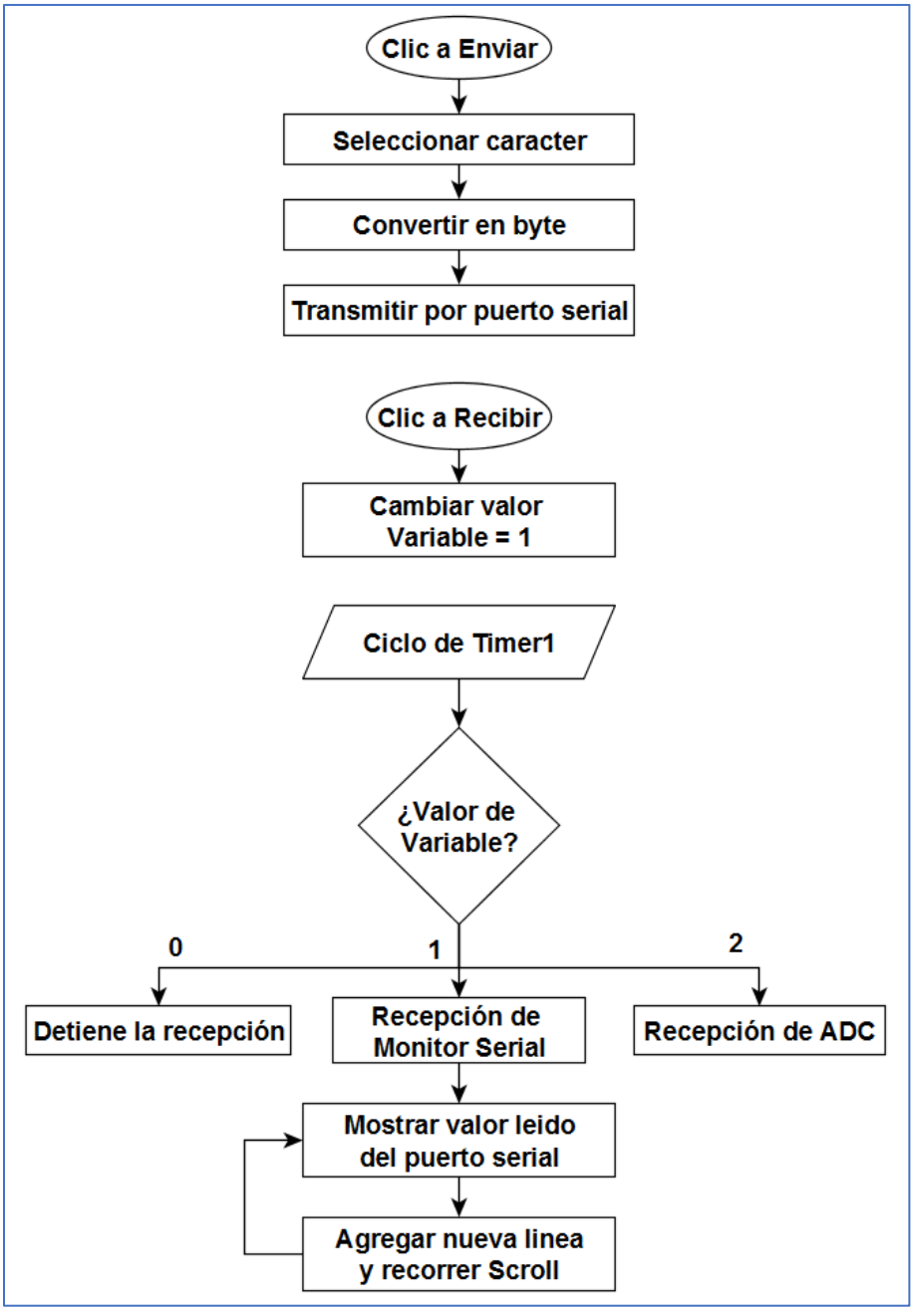


Figura 4.10 Diagrama de botón "Enviar" *y "*Recibir" Mon*.* Serial*.*

El *Scroll* de un *TextBox*, es la barra de desplazamiento. Recorrer *Scroll*, como se muestra en el diagrama anterior, hace referencia a tener siempre en primera línea el último dato recibido [24].

4.3 Interfaz de bootloader

El *bootloader* de un microcontrolador necesita de una interfaz que transmita los datos de un fichero hexadecimal. Esta se encarga de procesar el contenido, hacer las conversiones necesarias para que el microcontrolador interprete los datos, transmita la información y en el caso de este proyecto, implemente la protección contra sobre escritura [39].

La pestaña “Grabar” en la aplicación que se desarrolló para este proyecto, es la interfaz gráfica para el *bootloader*. A continuación, se presenta esta pestaña.

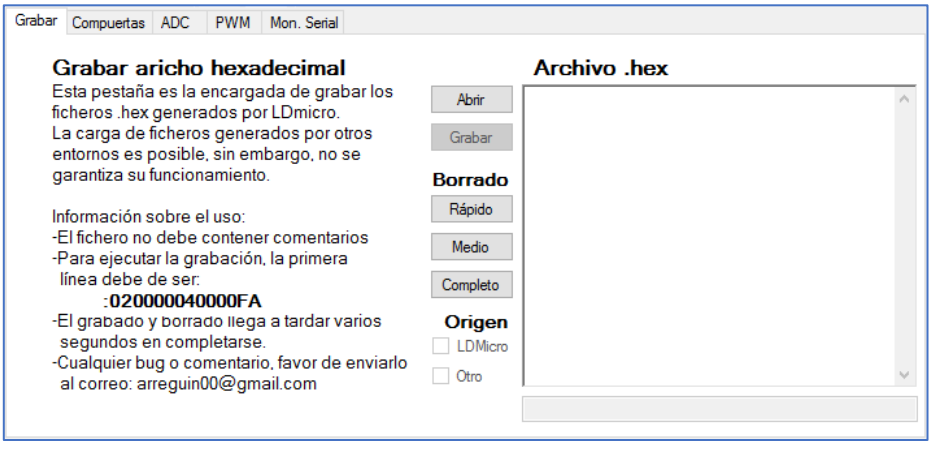


Figura 4.11 Pestaña “Grabar”.

Como se observa, en la parte izquierda se menciona información sobre su uso. Cada una de estas consideraciones se abordarán a lo largo de este tema.

Por otra parte, en el costado derecho de la pestaña, se observa un *TextBox*, varios botones y un par de *CheckBox*. El conjunto de estas herramientas conforma la interfaz gráfica del *bootloader.*

El funcionamiento de cada botón es el siguiente:

- **Abrir*:*** Despliega una ventana de dialogo para elegir el fichero hexadecimal que se quiera grabar en el microcontrolador
- **Grabar*:*** Al pulsar este botón se inicia el procesamiento del fichero y la transmisión de las direcciones y datos a grabar.

- **Rápido*:*** Se carga el valor *0x3FFF* hasta la dirección *0x01A0* que equivale aproximadamente al 5% de la memoria de programa.
- **Medio*:*** Este botón alcanza hasta la dirección *0x0FF8* que sería algo parecido al 50% de la memoria disponible.
- **Completo*:*** Carga el valor *0x3FFF* en todas las direcciones de la memoria *flash* disponible sin borrar el *bootloader*.

Los tres botones de borrado, no suprimen en si el contenido de la memoria *flash*, sino que lo sustituyen por el valor *0x3FFF* el cual el *PIC* interpreta como borrado. El proceso de grabado y borrado toma cierto tiempo, entre más largo sea el fichero, más tardará en grabar. Para el borrado medio, se tarda entre uno y dos minutos y el completo poco más de cuatro. Se recomienda solo hacer uso del borrado necesario para evitar pérdida de tiempo.

4.3.1 Procesamiento de fichero hexadecimal

Los *CheckBox* funcionan como un condicional para grabar y además discriminar el procesamiento del fichero según su origen. Es necesario tener uno marcado para proceder a la grabación.

El rotulado con *"LDmicro"* borra el primer bloque del fichero, o, en otras palabras, borra las primeras dos líneas de datos del archivo, se hace esto porque en el primer bloque de la memoria está almacenado el salto hacia la estructura principal del *bootloader* y para evitar problemas con los saltos en el programa.

El operador *28xx* de un fichero hexadecimal para *PIC* de 8-bit es el encargado de indicar los saltos. En la Figura 4.12 se observa un ejemplo de este operador.

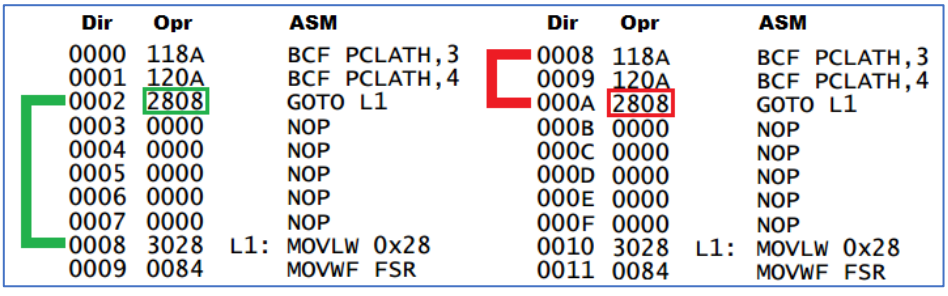


Figura 4.12 Ejemplos de operador *28xx.*

Como se aprecia, las “xx” son el valor hexadecimal de la parte baja de la dirección hacia dónde se dirige el salto. En la parte izquierda de la figura se observa un salto que se ejecuta de manera correcta. Del lado derecho se muestra un salto erróneo; que desde luego que al compilar en *ASM* direccionaría a *0x0010*, pero, este desplazamiento ocurre al usar el grabado de la memoria *flash* a través de un *bootloader*.

Como se mencionó, la solución a este problema fue eliminar el primer bloque del fichero de *LDmicro*, ya que el encabezado para todos sus archivos generados es el mismo, el cual no contiene configuraciones o procesos del programa, sino que limpia la memoria *RAM* para poder ocuparla para variables generadas en el código de programación, sin embargo, al ser una memoria volátil, el contenido de sus celdas de uso general será el de fábrica [28], por lo que quitar este bloque del fichero no perjudica a la ejecución del programa.

El segundo *CheckBox* rotulado con *“*Otro” indica que la grabación será genérica y que solo deberá ejecutar la protección contra sobre escritura.

Por otra parte, para iniciar el procesamiento del fichero y grabar el programa dentro del *PIC*, es necesario que la primera línea del fichero sea ***“:020000040000FA”***, ya que los archivos generados por *LDmicro* incluyen esta línea, por lo que el procesamiento inicia con esta condicionante, tanto en *“LDmicro”* como en *“*Otro*”*.

En las Figuras 4.13 y 4.14 se muestran los diagramas que representan el procesamiento de un fichero *hexadecimal* pulsando el botón “Grabar” de la interfaz. El procesamiento y transmisión se hace de la misma manera usando la grabación a través de la barra de menús o los botones de la barra de herramientas. Lo único que cambia es el condicionante de los *CheckBox*, ya que, dependiendo del tipo de grabación que se elija, se ejecuta la sentencia de la condicionante que se muestra en la Figura 4.13 pero de forma secuencial. Además, se han resaltado en color rojo las protecciones contra sobre escritura del *bootloader* y en azul la protección de los *fuses CONFIG*.

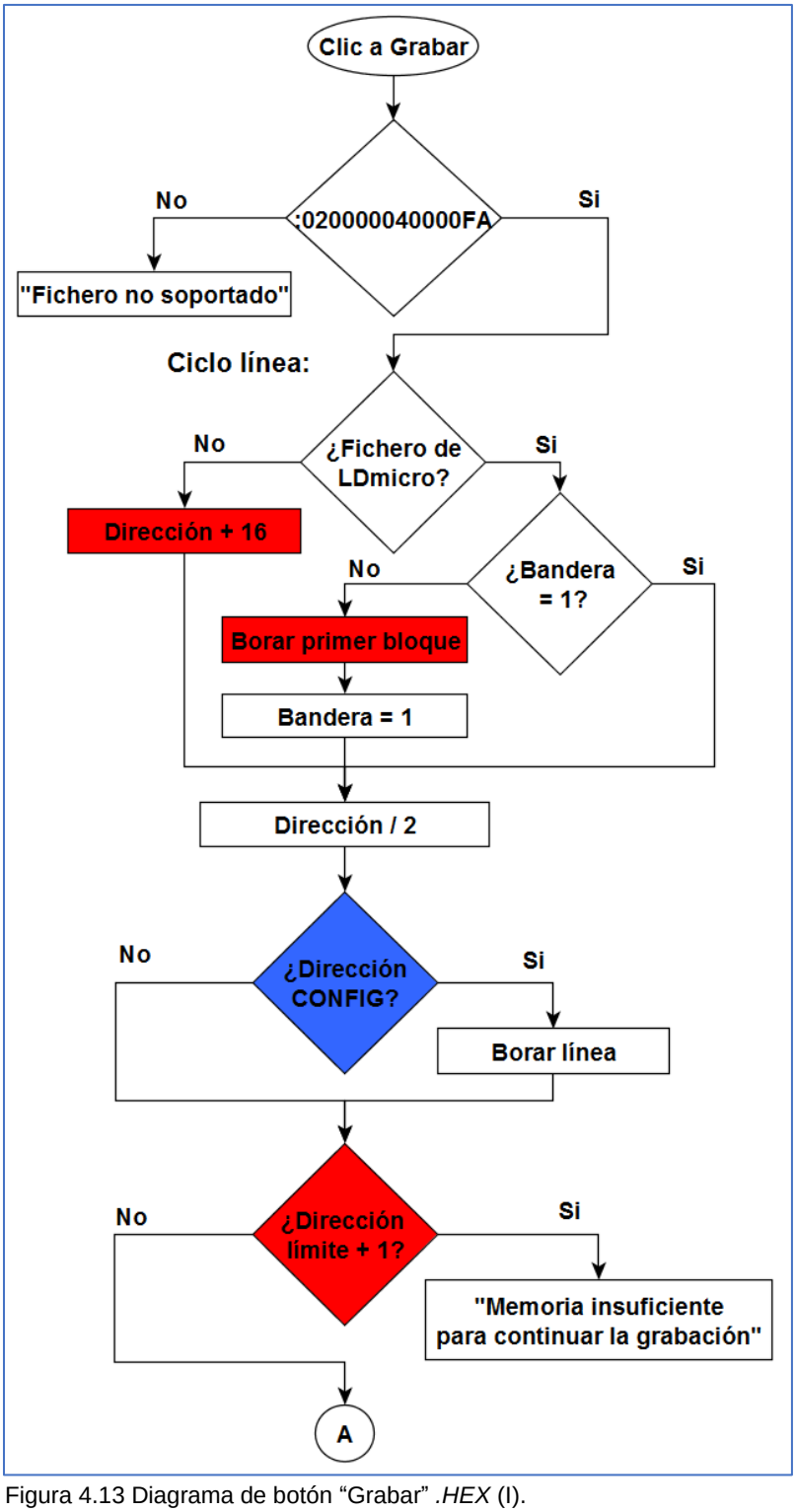


Figura 4.13 Diagrama de botón “Grabar” *.HEX* (I).

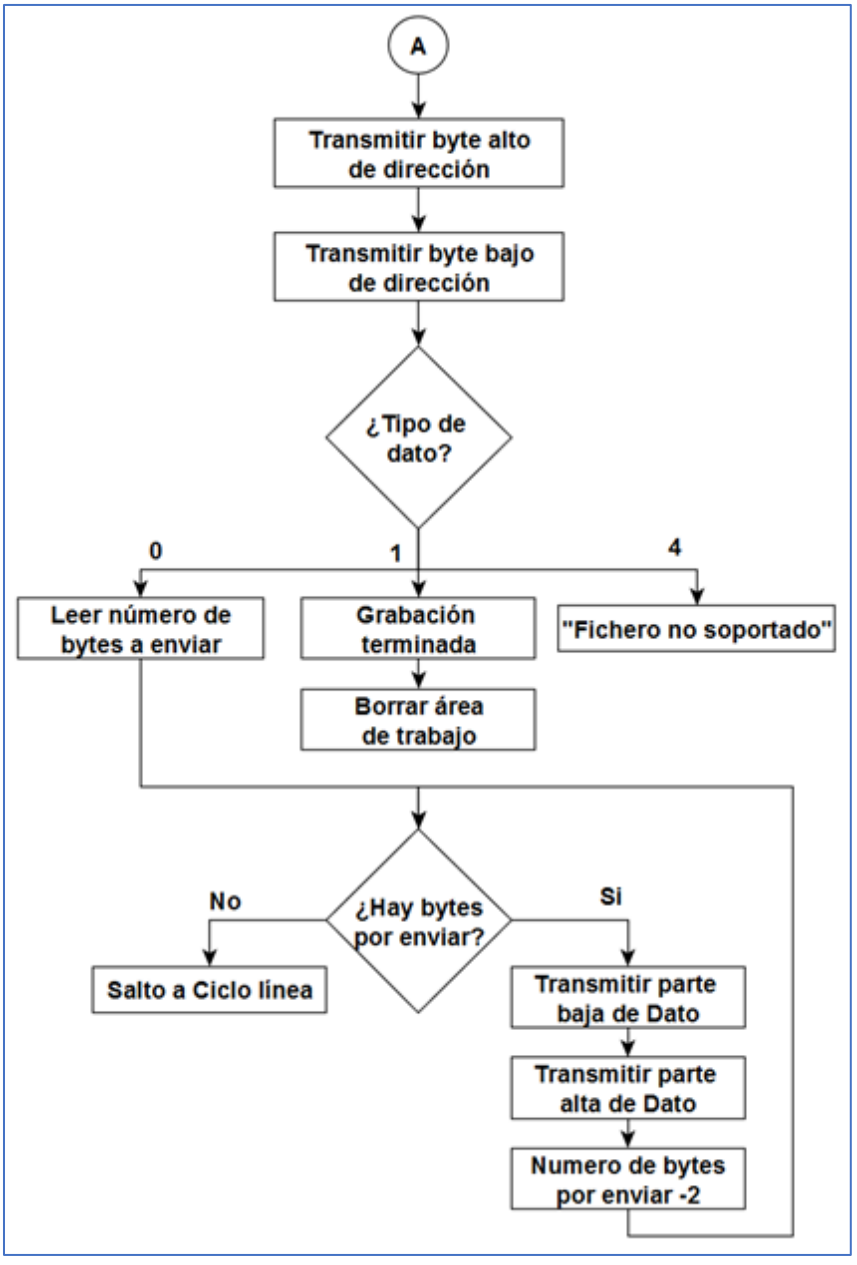


Figura 4.14 Diagrama de botón “Grabar” *.HEX* (II).

Los tipos de datos o grabación indican al ordenador que proceso se va a realizar con la línea del fichero. Del tipo *0* indica que son datos por grabar, *1* fin del fichero y *4* complemento a 32-bit. El *4* no se implementará ya que el *PIC* usado es de 8-bit y no son necesarios complementos tan amplios. Además, los tipos *2* y *3* solo son usados por procesadores de 32-bit [42].

4.3.2 Protección contra sobre escritura

Como se pudo observar en la Figura 4.13, se protege el bloque donde se almacena el salto al *bootloader* con “Dirección +16” y “Borrar primer bloque”. La protección del arrancador se lleva a cabo con “¿Dirección límite +1?”, esta toma de decisión revisa que no se grabe sobre el bootloader ya que la dirección límite es un bloque anterior al que se almacena.

El sumar 16 a la dirección es básicamente el mismo principio que el de eliminar el primer bloque del fichero, solo que en vez de ignorar las primeras dos líneas, el sumar 16 las recorre. Si se desea ocupar el grabado genérico hay que tener en cuenta que:

- El desplazamiento de la grabación puede ocasionar que los saltos no se realicen correctamente.
- Los *fuses CONFIG* no pueden ser modificados.
- La configuración del *fuse* del oscilador es *HS.*

El desplazamiento puede ser solucionado con la función *NOP* en el caso de *ASM*, o haciendo el desplazamiento desde el entorno de programación para que los saltos se realicen correctamente.

Los *fuses CONFIG* no pueden ser modificados para mantener la integridad del *bootloader*, por lo que el pin 1 será *RESET* y el oscilador funcionará en *HS*, como se mencionó, además no se podrá incluir *WatchDog* ya que este está desactivado.

4.3.3 Transmisión de bytes.

El protocolo de escritura del *PIC* toma un tiempo considerable, por lo que la aplicación también requiere esperar para transmitir el siguiente valor.

Tomando en cuenta que el *PIC* ocupa cuatro ciclos de reloj por cada instrucción y se ocupa un oscilador de 20Mhz, el tiempo de ejecución por cada instrucción es de 200ns [21].

$$t_{ins} = \frac{4}{20\ Mhz} = 200ns$$

Teniendo en cuenta que el proceso de grabación y su regreso a la espera de un dato recibido es de 36 instrucciones, el tiempo que deberá esperar la aplicación para transmitir cada byte es de 7.2ms, sin embargo, se programó con un tiempo de espera de 10ms mediante el método *Sleep* de la clase *Thread* la cual permite retardos sin necesidad de saturar al procesador con saltos [24].

4.4 Programas pregrabados

Los programas pregrabados fueron creados como una herramienta para implementar el *PLC* de manera rápida y como ejemplos sencillos para su implementación. Estos programas fueron escritos en el entorno *CCS* que es una plataforma de programación en lenguaje C estándar para microcontroladores *PIC*.

El diseño de estos programas mantiene comunicación serial con la aplicación que se desarrolló en *Visual Basic*. A través de la cual se configura el PIC para sus diferentes modalidades de uso en el *PLC*.

4.4.1 Programa de compuertas lógicas

En este programa, la configuración del *PIC* se lleva a cabo a través de un caracter, como se observó en la Figura 4.3, la combinación de compuertas elegida en la aplicación transmite dicho caracter, el cual ejecuta una función de manera cíclica como se muestra a continuación:

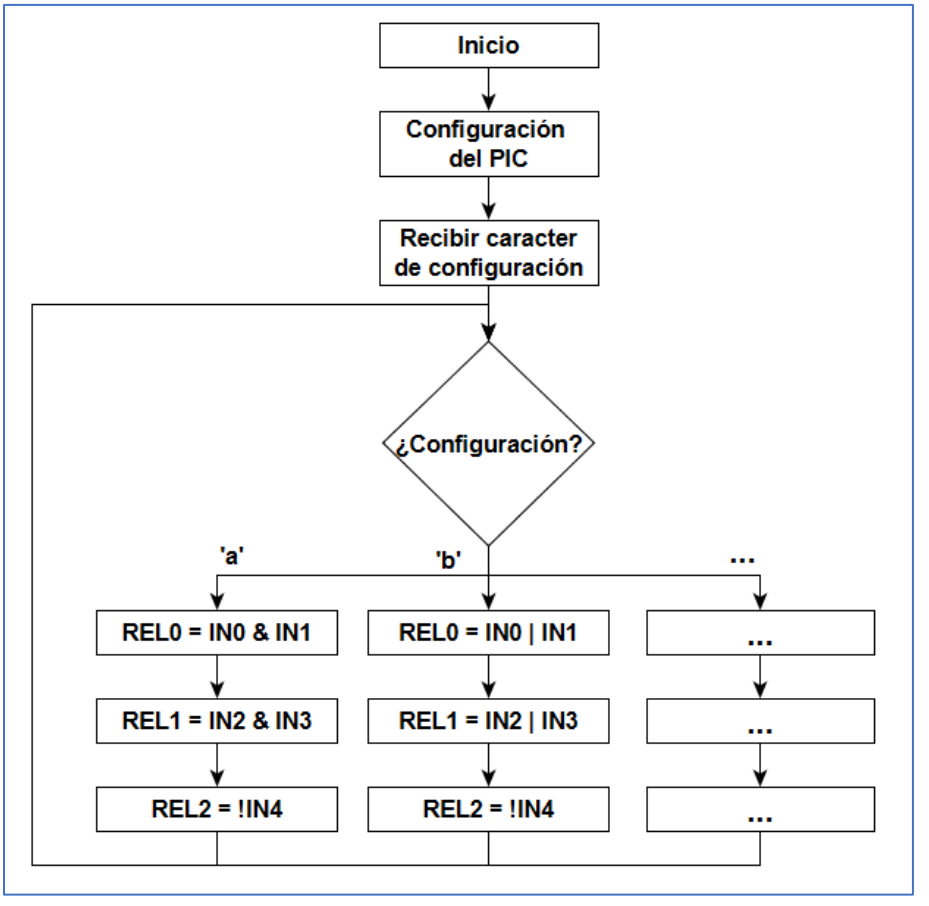


Figura 4.15 Diagrama de compuertas lógicas en el *PIC.*

Una vez recibido el carácter de configuración, no se podrá cambiar la combinación de las entradas y salidas. Para cambiar de combinación será necesario reiniciar el *PIC*.

4.4.2 Programa ADC

De la misma manera que el programa de compuertas lógicas, el *PIC* recibe caracteres para configurarse, para este caso, son tres. Estos caracteres alfanuméricos configuran el módulo *ADC*, relevador y valor de cierre.

En la Figura 4.16 se muestra el diagrama del programa diseñado, solo se percibe el funcionamiento del *ADC0*, sin embargo, la configuración y el ciclo infinito para los otros módulos se realiza de la misma manera.

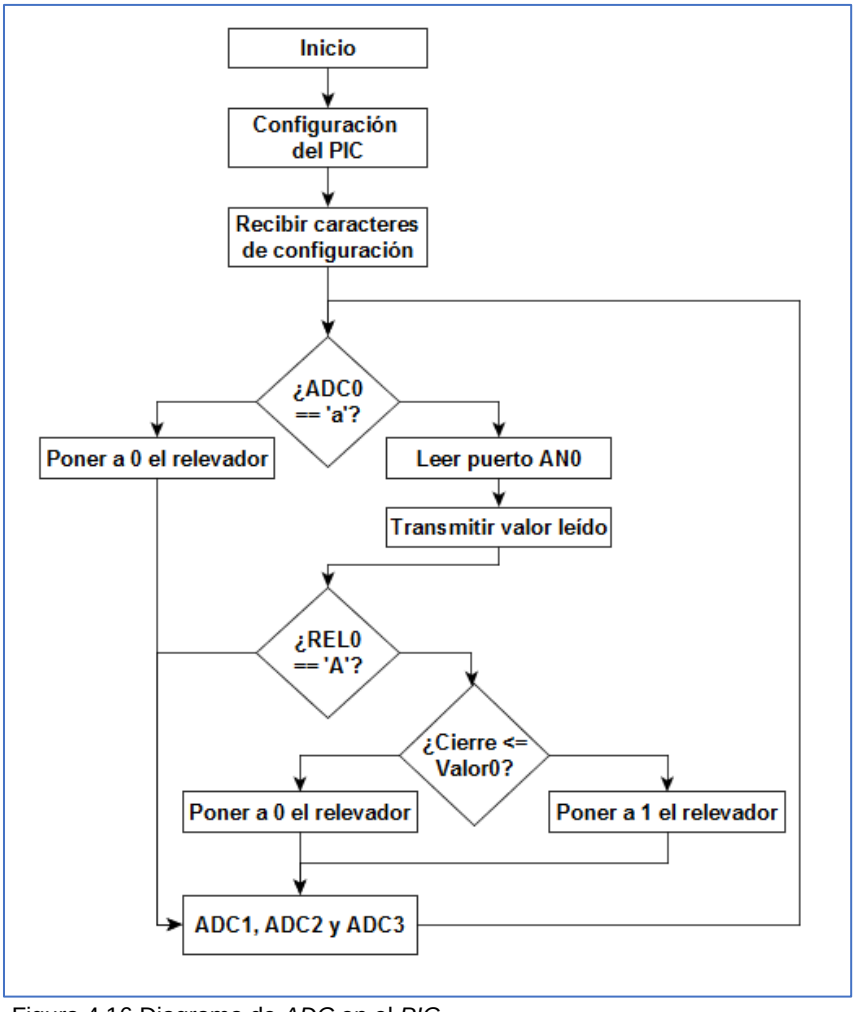


Figura 4.16 Diagrama de *ADC* en el *PIC.*

Como se mencionó anteriormente, estas lecturas al implementar más de un módulo son erróneas cuando se convierte el valor implementando el algoritmo presentado anteriormente, sin embargo, se comprobó con instrumentos de laboratorio (osciloscopio y multímetro) el valor del voltaje transmitido por el dispositivo analógico, el cual decae 20mV al encender un relevador. De cualquier forma, el valor convertido no debería de variar drásticamente el valor de la conversión como se ha observado en las pruebas.

4.4.3 Programa PWM

La configuración de este programa se realiza de la misma manera que los dos anteriores, sin embargo, la relación del ciclo de trabajo puede ser modificada a través de la aplicación. En la siguiente figura se muestra el diagrama a bloques de este programa.

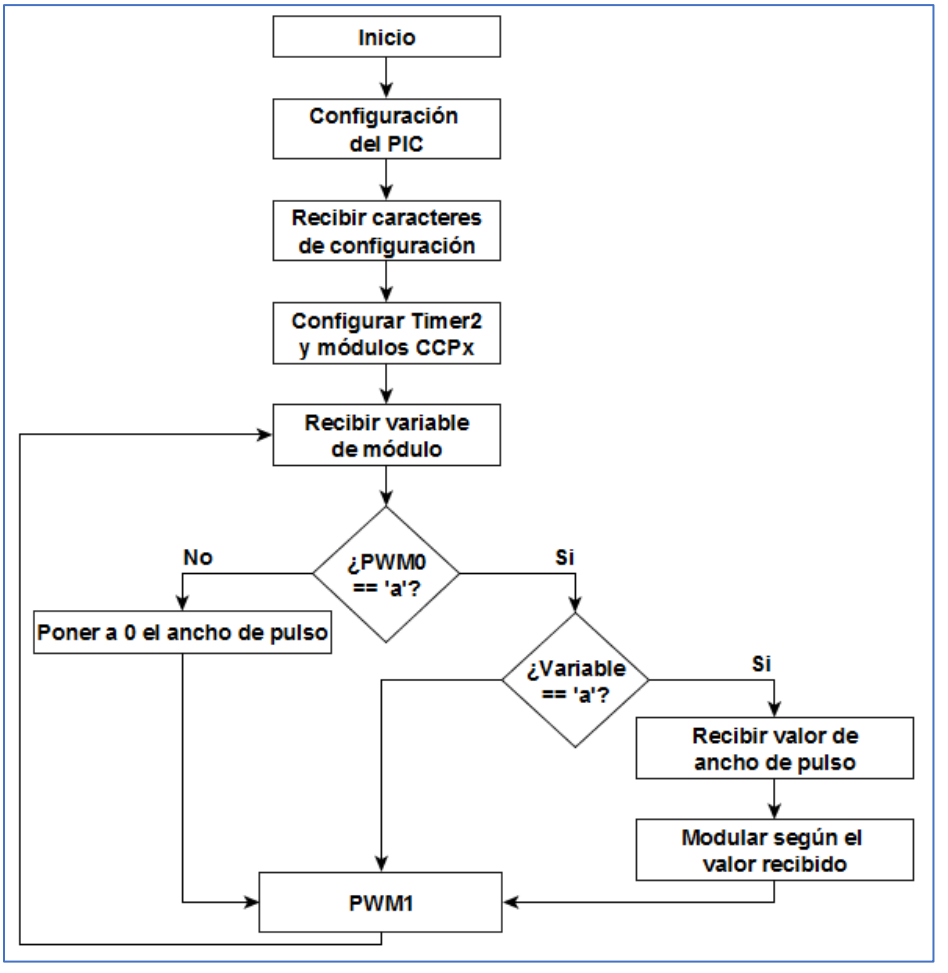


Figura 4.17 Diagrama de *PWM* en el *PIC.*

Capítulo 5
Pruebas y resultados

5.1 Prueba de programa creado en LDmicro

En la siguiente figura se muestra un programa en *Ladder* creado en la plataforma *LDmicro* exportado a texto para ser visualizado de mejor manera. El propósito de este programa es el de encender y apagar un par de relevadores dependiendo de la señal a la entrada del *PIC*.

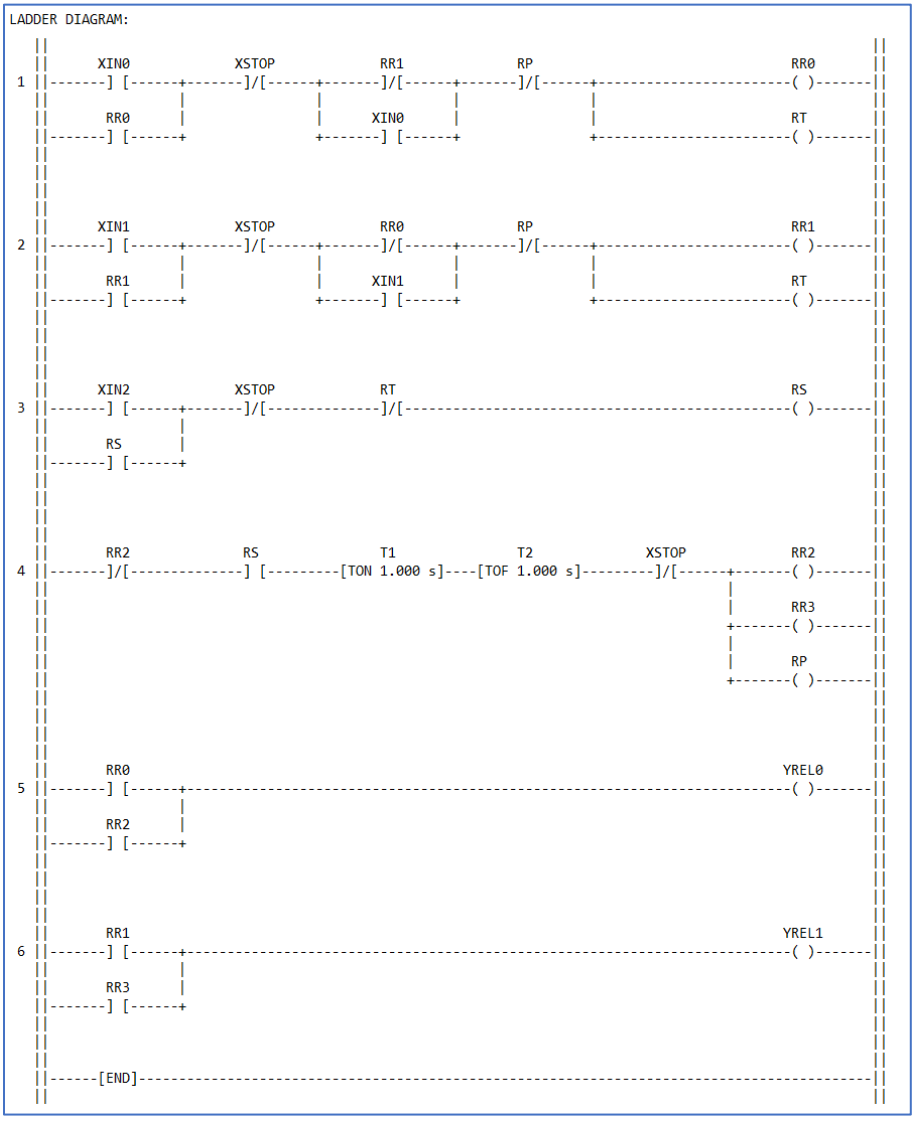


Figura 5.1 Programa de prueba en *LDmicro.*

De acuerdo con el programa, al activar *XIN0* se energiza la salida *YREL0*, por otra parte, al activar *XIN1* se activa *YREL1*. La entrada *XSTOP* detiene cualquier proceso que se esté ejecutando.

También, se observa que en la línea 4 se dispone de un encendido con retardo (*TON*) y apagado con retardo (*TOF*), la conexión entre ellos provoca que *YRELO* y *YREL1* enciendan y apaguen intermitentemente cada segundo al activar *XIN2*.

La configuración del cristal, el retardo entre ciclo y la disposición de pines del microcontrolador se muestran a continuación.

```
LDmicro export text
for 'Microchip PIC16F887 40-PDIP', 20.000000 MHz crystal,
10.0 ms cycle time

E/S ASIGNACIÓN:
  Nombre                    | Tipo               | Pata
 ---------------------------+--------------------+------
  XIN0                      | entrada digital    | 36
  XIN1                      | entrada digital    | 37
  XIN2                      | entrada digital    | 38
  XSTOP                     | entrada digital    | 39
  YREL0                     | salida digital     | 30
  YREL1                     | salida digital     | 29
  RP                        | rele interno       |
  RR0                       | rele interno       |
  RR1                       | rele interno       |
  RR2                       | rele interno       |
  RR3                       | rele interno       |
  RS                        | rele interno       |
  RT                        | rele interno       |
  T1                        | activar retardo    |
  T2                        | desactivar retardo |
```

Figura 5.2 Configuración de *PIC.*

5.1.1 Grabación del fichero en el PIC

El paso siguiente fue cargar el fichero que se creó dentro de la interfaz del *bootloader,* como se visualiza a continuación:

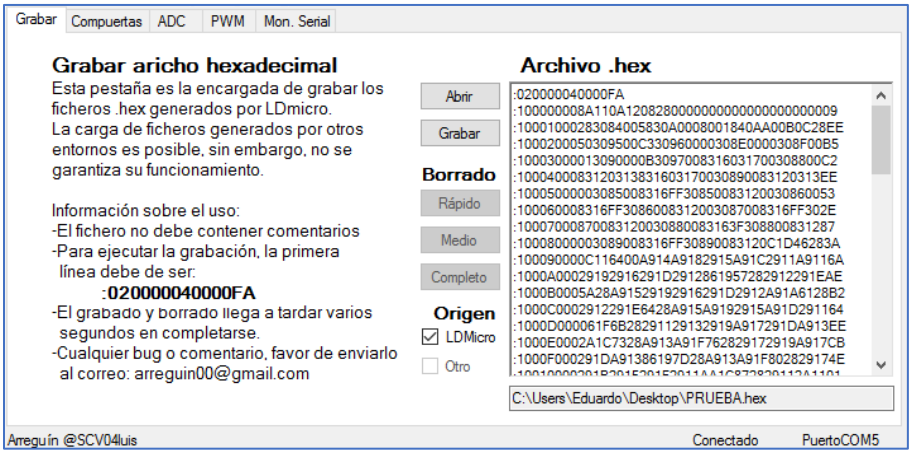


Figura 5.3 Fichero abierto en la aplicación.

Al ser un programa compilado en *LDmicro* se selecciona el *CheckBox* correspondiente y se pasa a grabar el fichero. Una vez grabado se despliega el mensaje de que todo fue transmitido correctamente.

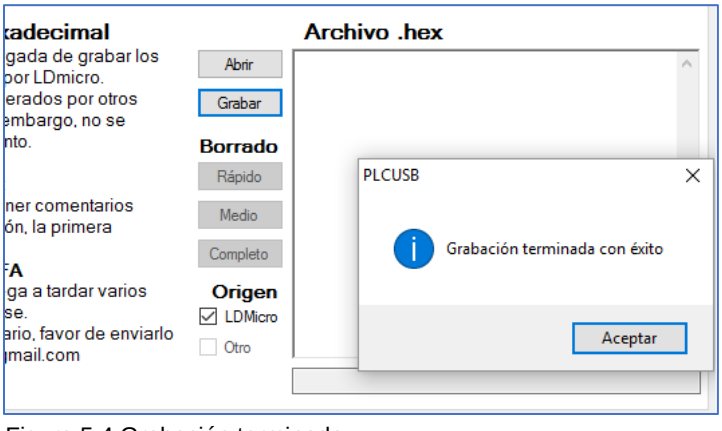


Figura 5.4 Grabación terminada.

5.1.2 Circuito de prueba

Las señales de las cuatro entradas provienen de cuatro interruptores normalmente abiertos, que al ser pulsados energizarán con $13V_{DC}$ las entradas, por lo que la resistencia del *zener* a la entrada debe ser la de 1kΩ. En ambos relevadores se conectan dos focos incandescentes de 25W a $127V_{AC}$ en la conexión NA, como se muestra a continuación:

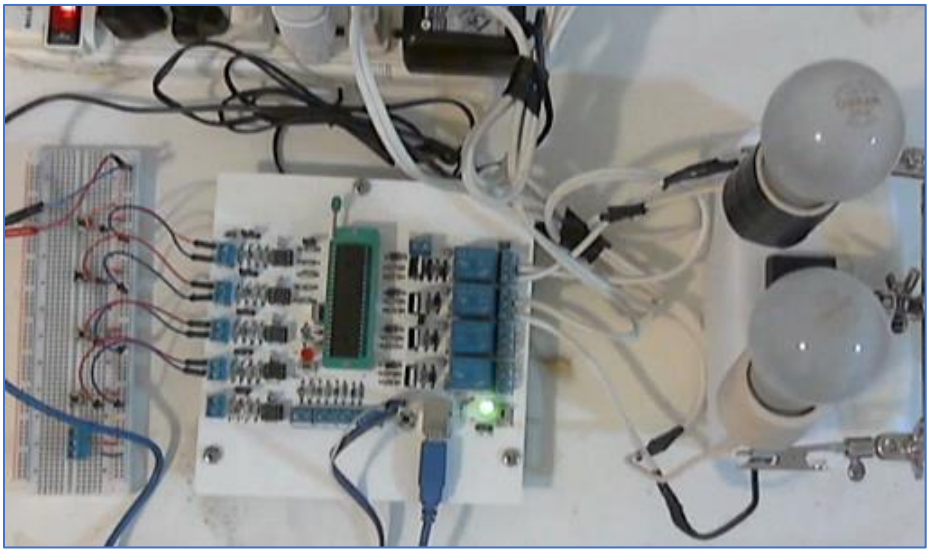


Figura 5.5 Circuito de prueba *LDmicro* implementado.

La conexión hacia los relevadores es la siguiente:

Figura 5.6 Disposición de conectores a relevador.

Como se observa, la conexión izquierda es el contacto NA, la central es el común y la de la derecha el NC. Esta disposición es la misma para los cuatro relevadores. Tomando en cuenta esto, la conexión será entre NA y común.

5.1.3 Resultados de la prueba

A continuación, se presentan los resultados obtenidos en la prueba:

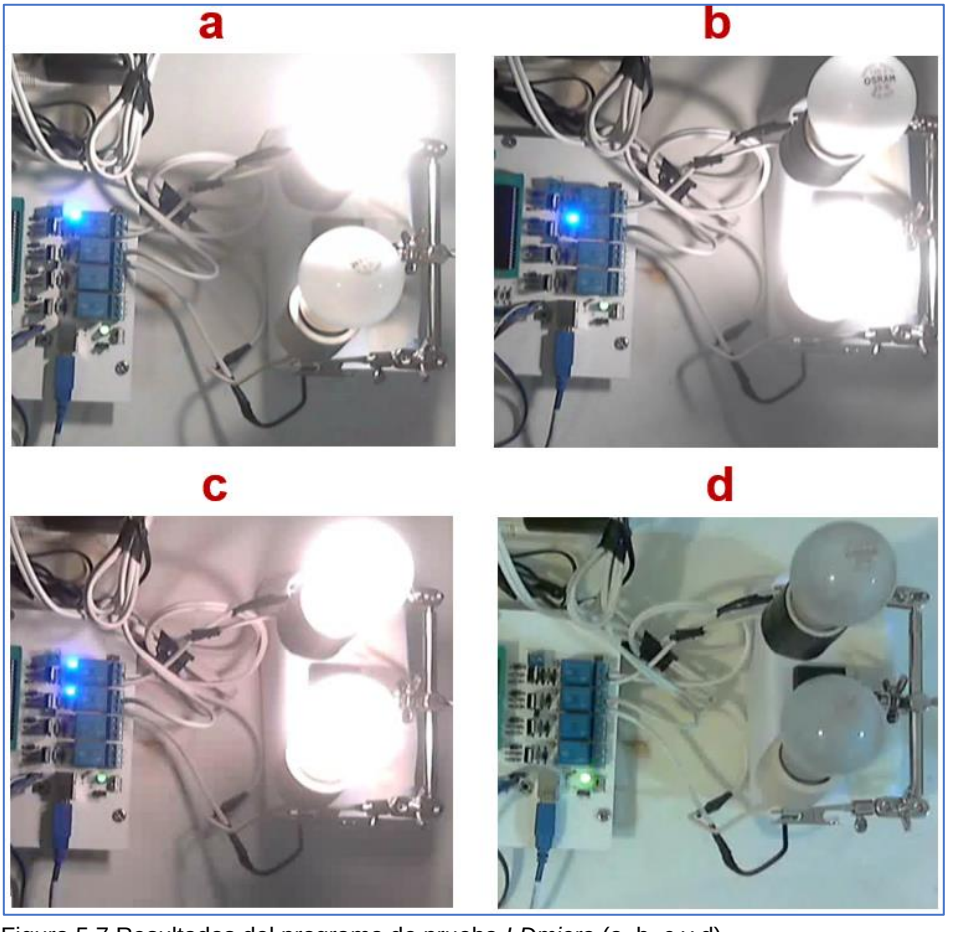


Figura 5.7 Resultados del programa de prueba *LDmicro* (a, b, c y d).

Donde:

- **a:** es el resultado de pulsar el botón acoplado a *XIN0* y permanece encendido indefinidamente hasta recibir otra instrucción
- **b:** es el resultado de pulsar el botón acoplado a *XIN1* y de la misma forma que "*a"* permanece encendido
- **c y d:** son el resultado de pulsar el botón *XIN3*, y su cambio de encendido a apagado y viceversa dilata un segundo

5.2 Prueba de programa pregrabado Compuertas

A continuación, se muestra la configuración de la pestaña “Compuertas” que será usada en esta prueba.

Figura 5.8 Configuración de pestaña “Compuertas”.

Tomando en cuenta esta configuración, la tabla de verdad es la siguiente:

Entradas	REL0	REL1	REL2
$\overline{IN0}\&\overline{IN1}$	0	*	*
$IN0\&\overline{IN1}$	0	*	*
$\overline{IN0}\&IN1$	0	*	*
$IN0\&IN1$	1	*	*
$\overline{IN2}\&\overline{IN3}$	*	0	*
$IN2\&\overline{IN3}$	*	1	*
$\overline{IN2}\&IN3$	*	1	*
$IN2\&IN3$	*	1	*
$!\overline{IN4}$	*	*	1
!IN4	*	*	0

Figura 5.9 Tabla de verdad de prueba compuertas.

5.2.1 Circuito de prueba

Para realizar esta prueba, se suministran $5V_{DC}$ provenientes de la alimentación de la placa, las variaciones en el suministro son a través de interruptores normalmente abiertos al igual que en la prueba anterior. Las salidas son visualizadas a través del *LED* de

cada relevador, el cual enciende cada vez que el pin de salida del microcontrolador se encuentra en estado alto.

5.2.2 Resultados de la prueba

En la siguiente figura se muestran los resultados obtenidos en la prueba para algunas combinaciones, sin embargo, las predicciones de la tabla de verdad se cumplen satisfactoriamente.

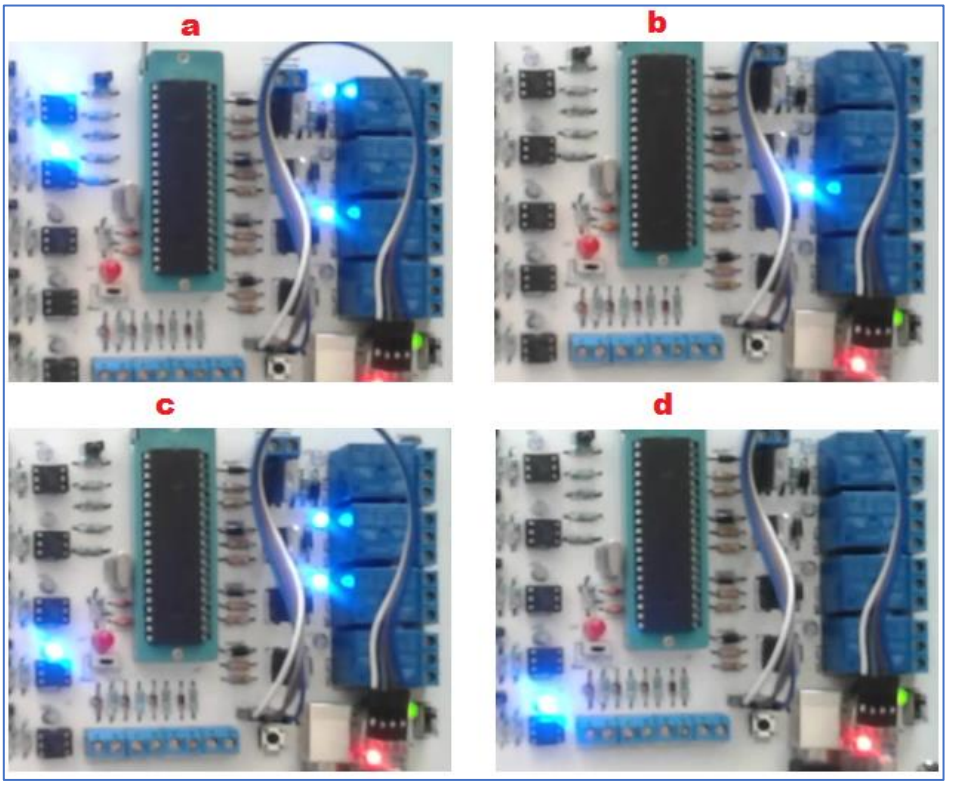


Figura 5.10 Resultados de prueba *Compuertas* (a, b, c y d).

Donde:

- **a:** representa a IN0&IN1
- **b:** representa a $\overline{\text{IN2}}$&$\overline{\text{IN3}}$
- **c:** representa a $\overline{\text{IN2}}$&IN3
- **d:** representa a !IN4

5.3 Prueba de programa pregrabado ADC

La configuración de esta pestaña para la prueba realizada se muestra a continuación:

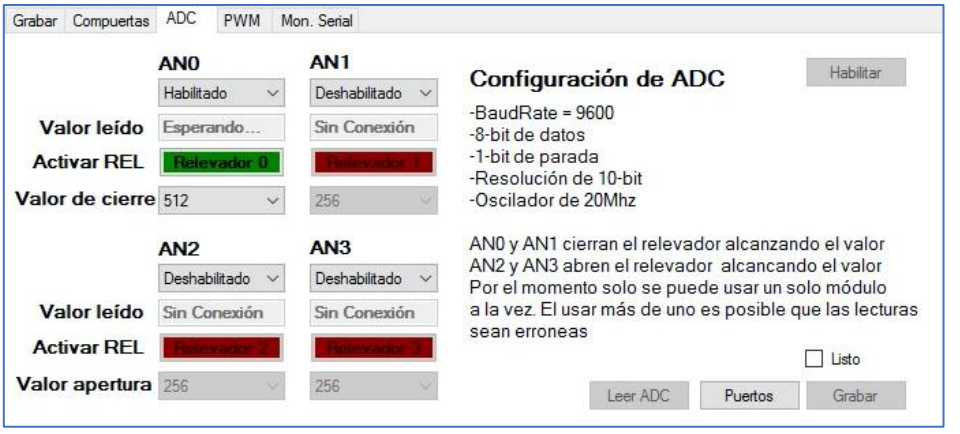


Figura 5.11 Configuración de pestaña *ADC.*

Se observa que se ocupa la entrada *AN0* la cual cerrará el relevador al alcanzar el valor 512 en las lecturas del puerto *ADC*.

5.3.1 Circuito de prueba

Para esta prueba se implementa un resistor variable y de la misma manera que en la prueba de compuertas, la visualización de la salida es a través del *LED* del relevador. En la siguiente figura se muestra el circuito de prueba implementado.

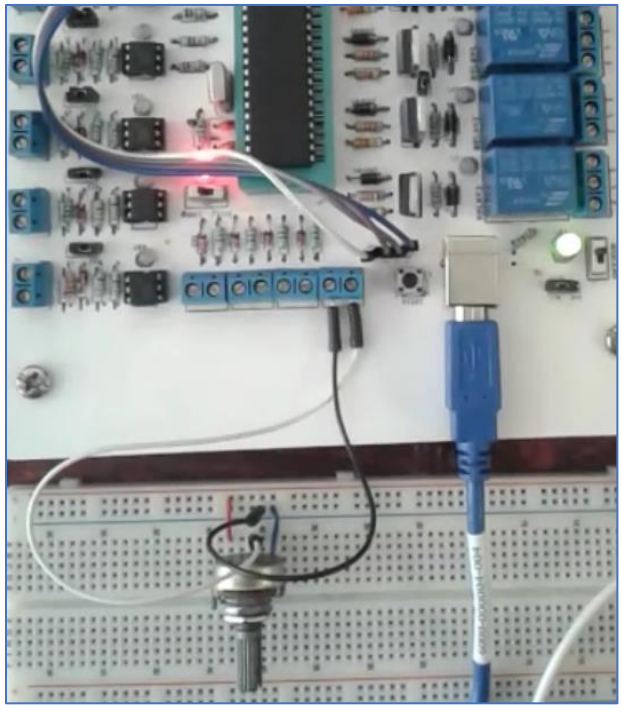


Figura 5.12 Circuito de prueba *ADC.*

5.3.2 Resultados de la prueba

Como se visualiza a continuación, el comportamiento del programa es el adecuado:

Figura 5.13 Resultados de prueba *ADC* (a y b).

Donde:

- **a:** el valor de cierre se ha superado y la bobina del relevador se ha energizado.
- **b:** el valor convertido es menor al de cierre y la bobina ha dejado de energizarse

5.4 Prueba de programa pregrabado PWM

Para este caso se ocupan ambos módulos con la frecuencia a 1500Hz como se presenta a continuación:

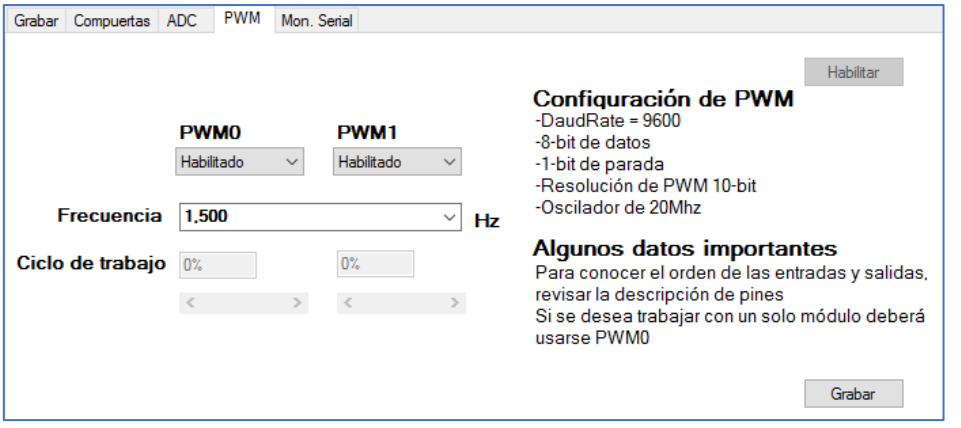


Figura 5.14 Configuración de pestaña *PWM.*

5.4.1 Circuito de prueba

En la siguiente imagen, la visualización de la señal emitida es mostrada en un osciloscopio el cual está conectado directamente al módulo *PWM* y *GND* de la tarjeta.

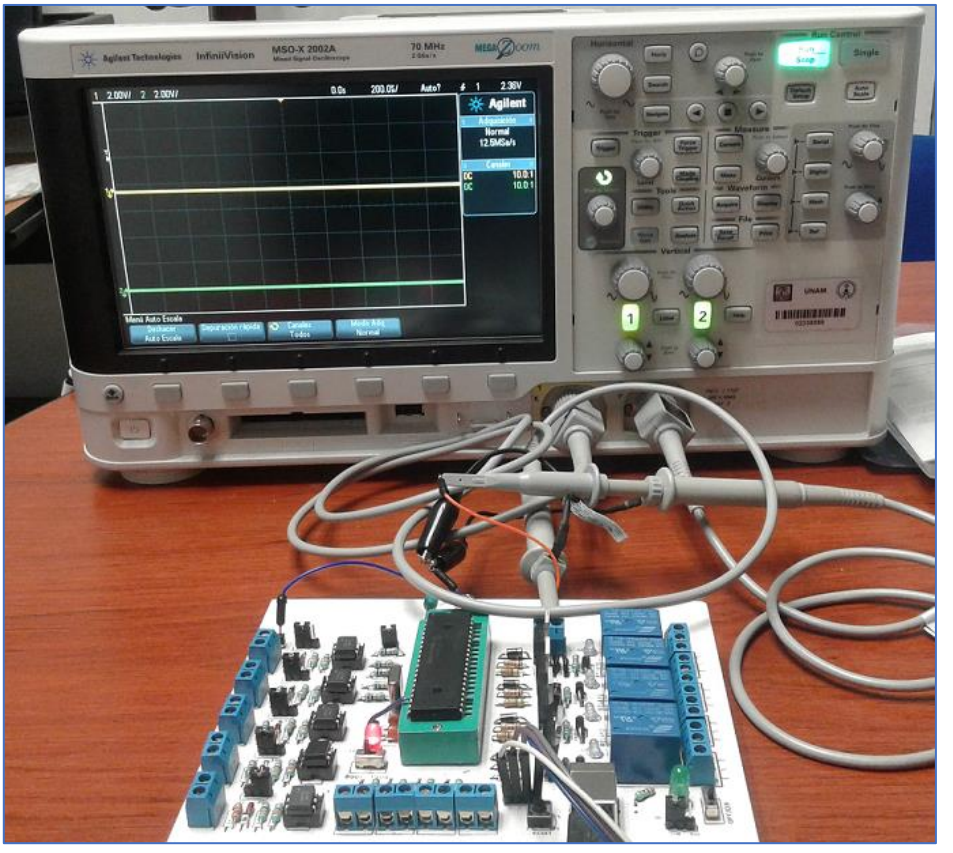


Figura 5.15 Conexión de prueba *PWM.*

5.4.2 visualización en osciloscopio

Los resultados obtenidos de la modulación fueron satisfactorios como. Se presentan en la siguiente figura:

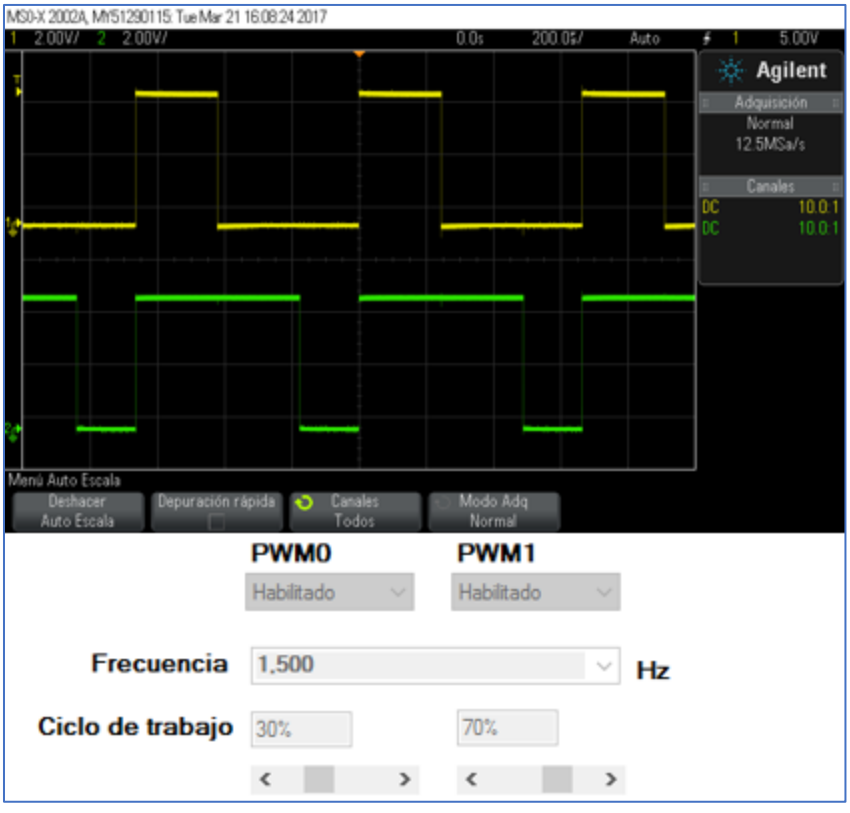


Figura 5.16 Visualización en osciloscopio.

La modulación se refleja correctamente a la salida. La línea superior corresponde al módulo *PWM0* y la inferior al módulo *PWM1.*

Por último, cada una de estas pruebas han sido documentadas en video y pueden obtenerse en el enlace: https://goo.gl/JhX1p4 en la carpeta “Pruebas”. La mayoría de las figuras presentadas en este capítulo son capturas de dichos videos.

Capítulo 6
Conclusiones

Conforme a los objetivos planteados, se desarrolló una tarjeta electrónica capaz de emular a un *PLC* comercial con un bajo costo de adquisición, tamaño reducido y programable vía *USB*, implementando un microcontrolador *PIC*.

También, se desarrolló una aplicación para *Windows*, que entabla comunicación con el microcontrolador de manera eficiente para transmitir los *bytes* de un fichero hexadecimal, el cual graba en la memoria *flash* de este dispositivo. Además, brinda cuatro programas pregrabados que funcionan como un auxiliar para implementar tareas sencillas de control dentro del mismo entorno desarrollado.

De acuerdo con las pruebas realizadas a la tarjeta, la grabación de la memoria *flash* a través del *bootloader* se realiza correctamente, el circuito propuesto trabaja de la manera esperada con los programas pregrabados y los programados en *Ladder*.

Con un microcontrolador de 8 bits basado en *RISC,* se concluye que puede ser implementado en un circuito que funcione como herramienta de estudio, para el diseño de sistemas embebidos de control a nivel industrial. En conjunto con una aplicación desarrollada para PC e implementando un *Bootloader,* se cuenta con las características básicas de un *PLC*.

6.1 Trabajo futuro

Debido a que el diseño busca ser semejante a un *PLC* comercial, el trabajo que queda por realizarse es basto. Respecto al circuito digital, se pueden adaptar las conexiones para que sean soportados microcontroladores de 18 o 28 pines. Se propone incorporar un algoritmo de verificación del *checksum* de cada bloque grabado en la memoria *flash*. Además de implementar un *Bootloader* en otros microcontroladores *PIC*.

En cuanto a la aplicación, acoplar programas pregrabados con diferentes propósitos para el control, así como un algoritmo de calibración para los dispositivos conectados a los módulos.

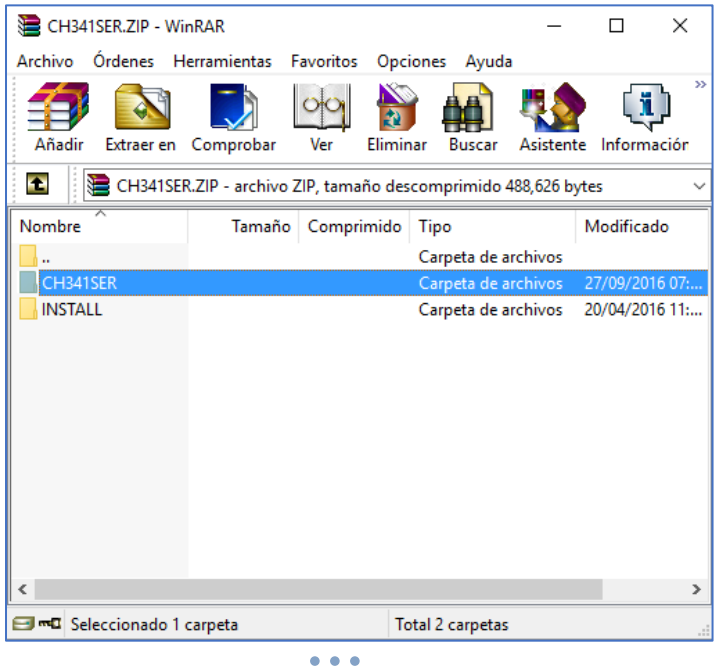
También, la incorporación de un entorno de programación en *Ladder* y otros lenguajes estandarizados dentro de la misma aplicación. El procesaLa interfaz del *bootloader* puede incorporar un algoritmo para procesar el operador *28xx* y evitar los problemas causados por el desplazamiento de los saltos y se pueda asegurar la buena ejecución de programas creados en otros entornos.

Anexo

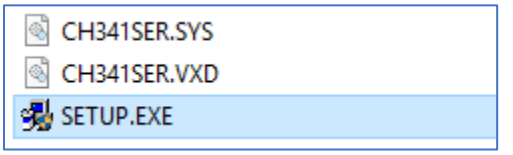
La instalación del *driver* para el puente *USB* se lleva a cabo de la siguiente manera:

1. Ingresar al siguiente enlace y descargar el fichero Zip pulsando el botón *"DOWNLOAD"*: http://www.wch.cn/download/CH341SER_ZIP.html

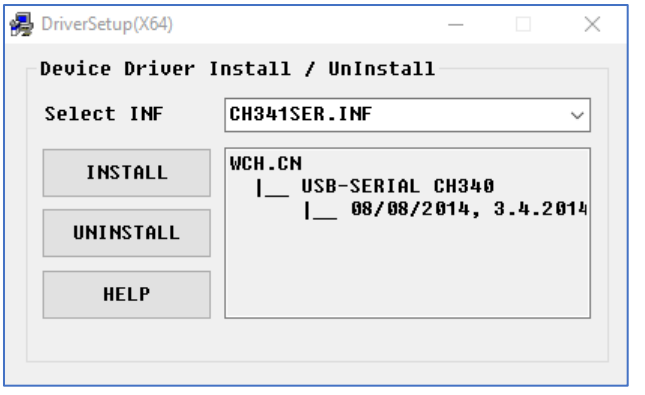
2. Al abrir el fichero nos encontraremos con dos carpetas, solo es necesario descomprimir la llamada "CH341SER"



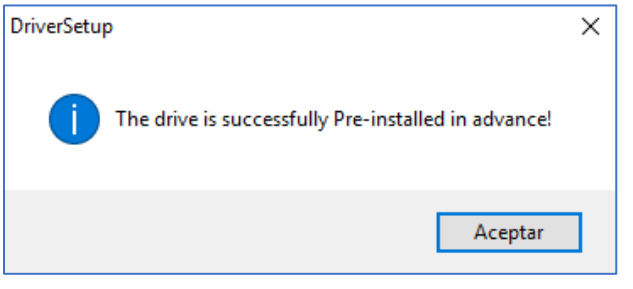
3. Al abrir la carpeta que se descomprimió se encontrará una carpeta y varios archivos, ejecutar el llamado “SETUP”



4. Al ejecutar el archivo se desplegará una pequeña pestaña como la que se muestra a continuación:



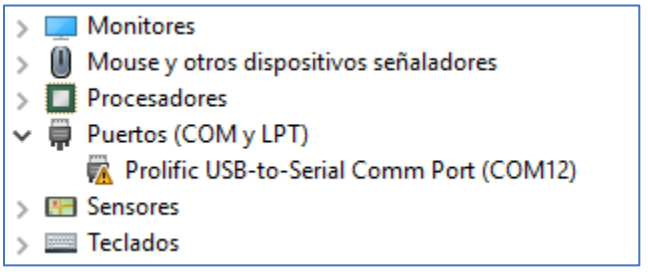
5. Para instalar el *driver* solo es necesario pulsar en el botón “INSTALL”. Se iniciará la copia de los archivos necesarios para el óptimo funcionamiento y al terminar se desplegará la siguiente ventana:



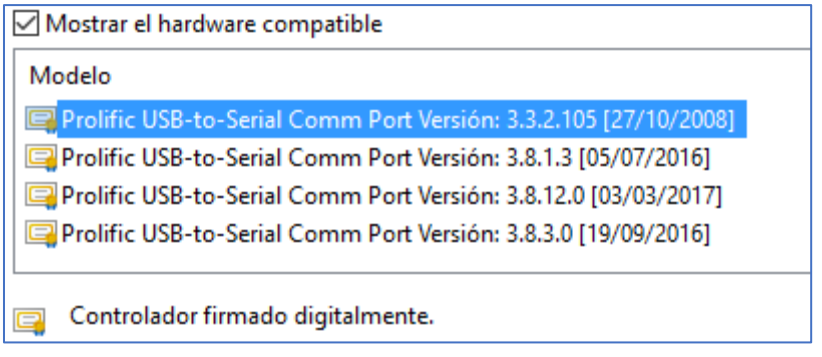
6. Una vez que se desplegó la ventana, la instalación del *driver* se ha completado y se pueden cerrar las ventanas ocupadas anteriormente.

Actualmente se instala un *driver* automáticamente, sin embargo, este no es funcional para el *FT-232*, si se presenta la situación en la que no se puede crear el puerto *COM* virtual, proceder con los siguientes pasos:

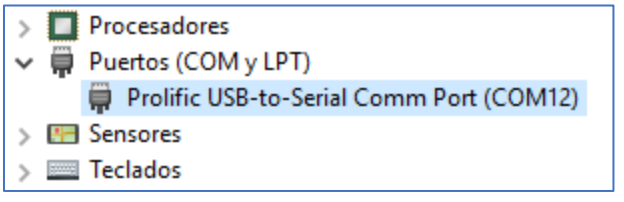
1. Abrir el administrador de dispositivos el cual se puede desplegar desde el panel de control en el apartado *Sistema* o a través del buscador "*Cortana*".
2. Desplazarse hacia el apartado "Puertos COM y LPT" y observar si el dispositivo llamado "Prolific USB-to-Serial Comm Port" se encuentra como se muestra a continuación.



3. Pulsar clic derecho sobre el dispositivo, después pulsar en "Actualizar software de controlador…", "Buscar Software de controlador en el equipo" y por último "Elegir en una lista de controladores de dispositivo en el equipo".
4. Se desplegará una lista con los controladores que se encuentran instalados en el equipo, elegir el fechado con [27/10/2008], como es mostrado a continuación:



5. Pulsar el botón "Siguiente", se instalará el controlador y posteriormente cerrar la ventana. Si el procedimiento se realizó correctamente, el dispositivo deberá observarse de la siguiente manera:



Algunas veces el ordenador instala automáticamente el controlador más reciente, sin embargo, estos no funcionan para el puente que se ocupa en este proyecto. Si eso ocurre solo es necesario repetir los cinco pasos mostrados anteriormente.

El instalador del *driver* es compatible para sistemas operativos de 64 y 32 bits desde *Windows XP* hasta *10*. Si se desea instalar en otro sistema operativo, en la página de descarga se puede seleccionar las diferentes versiones para descargar.

Referencias bibliográficas

[1] Molero, R. (2006). *Realización de un PLC didáctico* (Maestría). Universidad Nacional Autónoma de México.

[2] Ontiveros, M. (2016). *Reconocimiento de objetos y manejo de un brazo robótico mediante un procesador Sitara* (Maestría). Universidad Nacional Autónoma de México.

[3] Maloney, T. (2006). *Electrónica industrial moderna* (5th ed.). México: PEARSON EDUCACIÓN.

[4] Bryan, L. & Bryan, E. (1997). *Programmable controllers: theory and implementation* (2nd ed.). United States of America: Industrial Text Company.

[5] Allen Bradley. (2016). *MicroLogix 1400 Programmable Controllers.* Recuperado en febrero del 2017, de: http://literature.rockwellautomation.com

[6] Vallejo, H. (2013). *PLC: Los controladores Lógicos programables*. Saber Electrónica versión argentina, 166.

[7] Jiménez, A. (2015). *Desarrollo de un sistema en FPGA para ensamble robotizado guiado por visión* (Maestría). Universidad Nacional Autónoma de México.

[8] NEMA. (2013). *ICS 61131-1-2005 (R2013).* National Electrical Manufacturers Association. Recuperado en enero del 2017, de: http://www.nema.org

[9] PLCOpen. (2014). *PLCopen on IEC 61131-3*. PLCOpen. Recuperado en diciembre del 2016, de: http://www.plcopen.org

[10] Bolton, W. (2006). *Programmable Logic Controllers* (4th ed.). Burlington: Elsevier.

[11] PLCOpen. (2013). *IEC 61131-3: a standard programming resource*. PLCOpen. Recuperado en enero del 2017, de: http://www.plcopen.org

[12] Van der Wal, E. *IEC 1131 or 61131: Status of the Standard*. PLCOpen. Recuperado en diciembre del 2016, de: http://www.plcopen.org

[13] Westhues, J. (2009). LDmicro: *Ladder Logic for PIC and AVR* (6th ed.). Seattle: LDmicro. Recuperado en diciembre del 2016, de: http://cq.cx/ladder.pl#dl

[14] De la Peña, B. & Díaz, F. (2009). *Elementos de electrónica*. Cuautitlán: Universidad Nacional Autónoma de México.

[15] Galeano, G. (2009). *Programación de sistemas embebidos en C* (1st ed.). México: Alfaomega.

[16] Salas, S. (2015). *Todo sobre sistemas embebidos* (1st ed.). Lima: Universidad Peruana de Ciencias Aplicadas.

[17] Di Lella, D. *Curso introductorio sobre Microcontroladores Familias HC705 y HC908 de Freescale*. Edudevices. Recuperado en enero del 2017, de: http://www.edudevices.com.ar

[18] ON Semiconductor (2014). *TIP120, TIP121, TIP122 (NPN); TIP125, TIP126, TIP127 (PNP)*. USA. ON Semiconductor

[19] Reyes, C. (2006). *Microcontroladores PIC Programación en Basic* (2nd ed.). Quito: RISPERGRAF.

[20] Medina, R. (1992). *Programación avanzada en lenguaje ensamblador*. Bolivia: Facultad Nacional de Ingeniería. Recuperado en octubre del 2016, de: https://inf3530.files.wordpress.com/2011/09/tutorial-completo-assembler.pdf

[21] Valdés, F & Pallás, R. (2007). *Microcontroladores: Fundamentos y aplicaciones con PIC.* España: Marcombo

[22] Morero, F. (2000). *Introducción a la OPP* (1st ed.). Grupo EIDOS.

[23] Joyanes, L. (1996). *Programación orientada a objetos* (1st ed.). España: McGraw-Hill.

[24] Halvorson, M. (2010). *Visual Basic 2010 Step by Step* (1st ed.). Washington: Microsoft Press.

[25] Landa, N. (2010). *C#* (1st ed.). Buenos Aires: USERS.

[26] *Introducción a Visual Studio* .NET. Developer Network. Recuperado en octubre del 2016, de: https://msdn.microsoft.com/es-MX

[27] Blanco, L. (2002). *Programación en Visual Basic .NET* (1st ed.). Madrid: Grupo EIDOS.

[28] Microchip. (2007). *PIC16f882/883/884/886/887 Data Sheet*. USA. Microchip Technology Inc.

[29] Rodríguez, L. *Conceptos de la arquitectura .NET Framework.* Madrid: Universidad Pontifica de Salamanca. Recuperado en enero del 2017, de: http://www.colimbo.net/i208/docu.htm

[30] *Arduino - Bootloader*. Arduino.cc. Recuperado en febrero del 2017, de: https://www.arduino.cc

[31] CEL. (2016). *Optocouplers Product Selection Guide*. California. CEL.

[32] WAGO. (2015). *RELAYS AND OPTOCOUPLERS Product Overview*. Alemania. WAGO Kontakttechnik GmbH & Co.

[33] Universidad del Cauca. *Práctica 4. Características del diodo Zener*. Colombia. Universidad del Cauca. Recuperado en diciembre del 2016, de: http://www.rftorrent.com/caracteristicas_deL_diodo_zener.pdf

[34] Hackworth, J. & Hackworth, F. (2004). *Programmable Logic Controllers: Programming Methods and Applications* (1st ed.). Pearson/Prentice Hall.

[35] India Community Initiative. *.NET Tutorial for Beginners*. India Community Initiative. Recuperado en noviembre del 2016, de: https://www.computer-pdf.com/programming

[36] *Clase SerialPort*. Developer Network. Recuperado en octubre del 2016, de: https://msdn.microsoft.com/es-es

[37] Pérez, F & Ávila, D. *Amplificador Darlington*. SENA (CEET). Recuperado en marzo del 2017, de: https://es.scribd.com

[38] Instituto Nacional para la Educación de los Adultos. (2008). *CONTROL DE EMISIONES 14. Relevadores*. México. Instituto Nacional para la Educación de los Adultos en Coordinación del Consejo Nacional del Educación para la Vida y el Trabajo

[39] Beningo, J. (2015). *Bootloader Design for Microcontrollers in Embedded Systems*. BENINGO ENGINEERING. Recuperado en diciembre del 2016, de: http://beningo.com/

[40] Anton, O. Gelineau, B. & Sauget, J. (2012). *Firmware and bootloader*. rose.telecom-paristech.fr. Recuperado en febrero del 2017, de: https://rose.telecom-paristech.fr/2012/wp-content/uploads/

[41] *Comparación de Visual Studio y Visual Web Developer Express.* Developer Network. Recuperado en febrero del 2017, de: https://msdn.microsoft.com/es-es

[42] Márquez, Diego. (2008) *El fichero HEX explicado.* PicManía by RedRaven. Recuperado en septiembre del 2016, de: http://www.picmania.garcia-cuervo.net