MICROCONTROLADORES

GUILLERMO TERÁN
JUAN CARLOS COBOS
MARCO A. CHILUISA

Microcontroladores

Editorial Universitaria, 2022
Guillermo Terán
Juan Carlos Cobos
Marco Chiluisa

ISBN | 978-9942-7004-0-7
1.ª edición | enero de 2022

Diseño y diagramación | Christian Echeverría
Corrección de textos | Marcelo Acuña
Diseño de portada | Christian Echeverría

Editorial Universitaria
Ciudadela Universitaria, avenida América, s. n.
Quito, Ecuador
+593 (02) 2524 033
editorial@uce.edu.ec

Contenido

Presentación

El fin que persigue esta obra es elaborar una guía básica del funcionamiento de los microcontroladores. Para conseguirlo, se ha intentado ofrecer una explicación sobre la construcción textual y gráfica, a fin de hacerlo más asequible a quienes desconocen por completo sobre estos dispositivos electrónicos.

Para que el lector adquiera los imprescindibles conocimientos prácticos en la programación y construcción se ha ejemplificado el diseño de circuitos electrónicos, con este fin se ha procurado elegir material electrónico fácil de adquirir y económico.

La estructuración de la obra está orientada especialmente hacia la nanoelectrónica, basada en los microcontroladores y los nuevos conceptos que conlleva la microinformática. En este sentido, se ha eliminado el estudio de la electrónica digital y analógica, que ya han dejado de usarse, y se ha potenciado la microelectrónica.

El temario general abarca tres capítulos relevantes: el primero hace referencia a la arquitectura, el segundo capítulo trata sobre la programación y el tercer capítulo sobre la construcción de circuitos.

El ofrecer la obra en estos capítulos tiene doble finalidad. En primer lugar, sirve para escalonar el aprendizaje, de una forma metodológica y sencilla, procurando que cada capítulo muestre un tema completo e independiente, para facilitar una progresiva introducción a la nanoelectrónica. En segundo lugar, se permite la consulta del código fuente de los circuitos realizados

Introducción

En los avances tecnológicos, los microcontroladores son las herramientas fundamentales para este logro, sumados a la versatilidad de los dispositivos electrónicos que trabajan a grandes velocidades y con varias aplicaciones, de acuerdo a las necesidades del usuario.

Los ejercicios aquí expuestos son las prácticas realizadas con los microcontroladores en las diversas actividades de programación y sistemas electrónicos. En primera instancia, queremos dar a conocer las bondades de los microcontroladores basados en los fundamentos científicos que han desarrollado las familias tecnológicas de PIC y AVR, cuya popularidad es muy amplia entre los diseñadores; esto se debe a que requieren un rendimiento alto y bajo costo, el seleccionar entre los PIC y AVR se debe a su nivel de integración, arquitectura, la disponibilidad de recursos de su lenguaje de programación.

En este libro nos dedicaremos a resolver los diferentes ejercicios con la utilización de circuitos integrados programables, que de aquí en adelante los denominaremos PIC.

Los PIC son circuitos integrados que tienen una estructura similar a las de las computadoras, es decir, tienen una memoria RAM, un espacio de almacenamiento, terminales de entrada y de salida; lo que no tienen son los monitores, pero para la demostración de las diferentes programaciones se utilizan los llamados elementos de respuesta, que son: parlantes, diodos led, display, entre otros; asimismo, utilizan el hardware y el software; en el hardware se emplean elementos electrónicos activos y pasivos, en lo que se refiere al software tiene un IDE netamente para los microcontroladores.

Para las simulaciones de los circuitos se utiliza el software PROTEUS, en sus diferentes versiones, esto hace que el estudiante, de acuerdo a la arquitectura y sistema operativo de su computador, pueda instalarlo adecuadamente, así como también la grabación de

los PIC que, por lo general, es una tarjeta con dispositivos electrónicos y un microcontrolador con salida USB con algunas directivas para su configuración.

En un acápite hablamos de la forma de instalar el software para la programación de los microcontroladores, además presentamos varios ejercicios realizados con los PIC.

Finalmente, dejamos planteados algunos ejercicios para que puedan ser desarrollados de acuerdo a los avances tecnológicos y científicos adquiridos por los estudiantes.

Capítulo I

Estructura de los microcontroladores

Generalidades de los microcontroladores

Los primeros microcontroladores fueron inventados por Texas Instruments en la década de 1970; en esa misma época se inventaron los microprocesadores en Intel. Los primeros microcontroladores tenían las características con la función de memorias RAM y ROM, pero con el pasar del tiempo, los microcontroladores fueron desarrollando una amplia gama de dispositivos diseñados para aplicaciones de sistemas integrados específicos en dispositivos como automóviles teléfonos móviles y electrodomésticos (ver Figura 1).

El primer microcontrolador creado con el nombre de TMS, que tenía como características cuatro bits con funciones específicas en la memoria ROM y RAM, era utilizado internamente por Texas Instruments en sus productos de cálculo, desde 1972 hasta 1974.

Pero la empresa Intel también desarrolló muchos microcontroladores importantes, dos de los cuales, el 8048 y el 8051, fueron utilizados en diferentes circuitos, como el procesador del teclado de la computadora personal IBM. El microcontrolador 8051 siguió usándose hasta 1980 y se convirtió en una de las familias más populares.

Hoy en día aún se conservan las variaciones de la arquitectura del 8051, pero es sumamente importante resaltar que esta arquitectura es uno de los diseños electrónicos más longevos de la historia.

Con el paso del tiempo, en la década de 1990, los microcontroladores eran memorias ROM (EEPROM) que se podían programar y borrar eléctricamente, como las memorias flash USB que se utilizan actualmente. Estos microcontroladores pueden ser programados, borrados y volver a ser reprogramados, utilizando solo señales

eléctricas; antes, los microcontroladores necesitaban de programación especializada y hardware para realizar el proceso de borrado, por lo que se requería que el dispositivo sea retirado del circuito para poder efectuar este proceso de grabar y borrar.

Con los avances tecnológicos, y una vez eliminada esta limitación, los microcontroladores hoy son capaces de ser programados y reprogramados en los mismos circuitos y actualizados con los nuevos software, sin tener que ser devueltos a los fabricantes, tanto Atmel como Microchip (ver Figura 2).

Diferencias de controladores y microcontroladores

Una de las principales diferencias de los controladores y microcontroladores, es que el controlador tiene programado una función específica en el funcionamiento de un circuito y no puede ser reprogramado, mientras que el microcontrolador puede realizar varias funciones y además ser reprogramado sin necesidad de ser extraído del circuito; con los avances tecnológicos, hoy en día los microcontroladores con soldaduras superficiales son cada vez más pequeños y muy versátiles.

Otra de las diferencias de estos dispositivos electrónicos es la estructura interna, los microcontroladores tienen entradas, salidas y un espacio de memoria para almacenamiento, los controladores tienen circuitos electrónicos diseñados para una función específica (ver Figura 3).

Un circuito integrado *controlador*, conocido también como tarjeta controladora, es el conjunto de dispositivos electrónicos que se encuentran previamente diseñados para realizar una determinada acción de entrada y salida con aplicaciones específicas de potencia u otras aplicaciones como radio frecuencia (RF), video frecuencia (VF), audio frecuencia (AF), en sistemas analógicos y digitales (ver Figura 4). Según HETPRO (2021) el A/D funciona con el método de aproximaciones sucesivas. El voltaje de referencia de este dispositivo puede ser ajustado para codificar diferentes rangos de voltajes.

Como podemos observar, tanto el microcontrolador como el controlador tienen similar estructura física, pero la diferencia es que

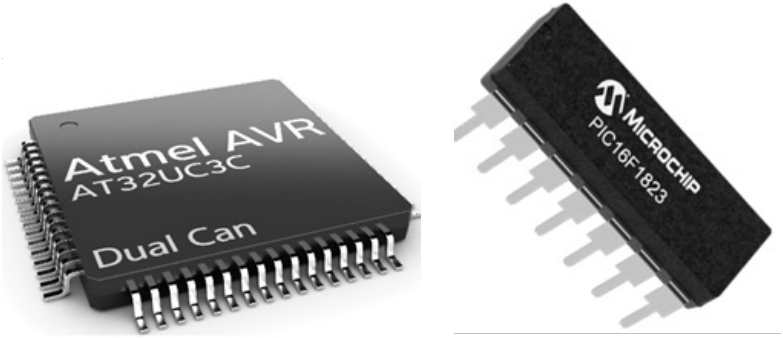


Figura 1. Forma física de los microcontroladores en AVR y PIC
Nota. El microcontrolador AVR con tecnología SMD y el PIC de tecnología común, que se inserta en las placas electrónicas. Tomada de (Direct Industry, 2021)

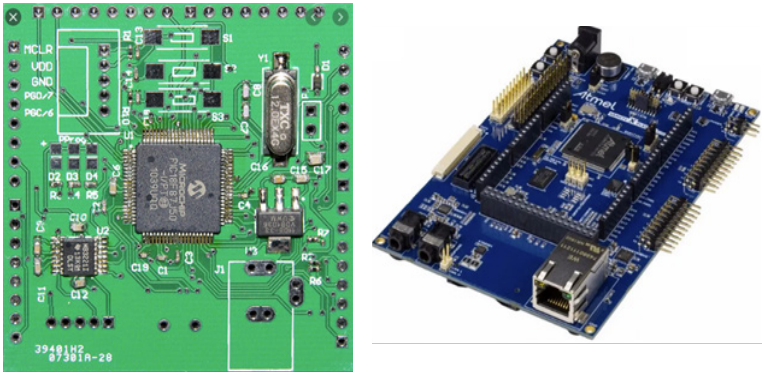


Figura 2. Los microcontroladores en las placas electrónicas
Nota. Los microcontroladores AVR y PIC con tecnología SMD (Direct Industry, 2021).

en el microcontrolador se puede borrar y programar acciones específicas que el usuario necesita que realice un determinado dispositivo, mientras que el controlador está diseñado para que realice una función específica y no se puede reprogramar este tipo de dispositivos electrónicos.

En la actualidad existen microcontroladores con tecnología SMD que trabajan a velocidades de nanosegundos y son los elementos principales para el funcionamiento de un determinado dispositivo electrónico, por ejemplo: el celular, el microondas, la videocámara, entre otros.

Según Surtel electrónica (2019), un componente tipo SMD (*surface mounting device*) se suelda de forma directa a la superficie de la PCB a través de los pads, dicha tecnología es denominada SMT, frente a los componentes de tecnología de agujeros pasantes o throughole que se fabrican con terminales que se sueldan en la parte contraria donde se inserta el componente. Los componentes electrónicos tradicionales se están abandonando paulatinamente, haciendo uso extensivo de los componentes SMD.

El controlador sigue siendo parte de una tarjeta que tiene incluido un microcontrolador para realizar una determinada acción, por ejemplo, la tarjeta de red, tarjetas USB, tarjetas lectoras de memoria, entre otras (ver Figura 5).

El microcontrolador

Es un elemento electrónico que tiene la misma forma física que un circuito integrado, el microcontrolador tiene una estructura similar a la de un computador, es decir, tiene puertos o terminales de entrada y salida y una memoria de almacenamiento y control (ver Figura 6).
Para el funcionamiento de los microcontroladores es necesario analizar que el hardware ya viene integrado en un solo chip, y que para usar un microcontrolador se debe especificar su funcionamiento por software a través de programas que indiquen las instrucciones que el microcontrolador realizará.

En una memoria se guardan los programas y un elemento llamado CPU se encarga de procesar paso por paso las instrucciones

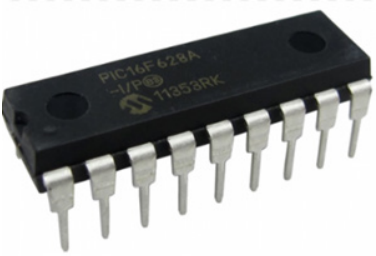


Figura 3. Microcontrolador PIC
Nota. El PIC más utilizado, con las características físicas reales (Urbina, 2012).

Figura 4. Controlador analógico digital
Nota. CI convertidor analógico/digital ADC0804 de 8 bits (Hetpro, 2021).

del programa, los lenguajes de programación que se usan para este fin son ensambladores y C, pero antes de grabar un programa en el microcontrolador hay que compilarlo a hexadecimal, que es el formato con el que funciona. Con las actuales tecnologías la grabación de los microcontroladores se puede realizar en el mismo circuito sin tener que extraer al PIC, todo dependerá del tipo de grabador de micros que esté utilizando.
Los bloques funcionales básicos de los microcontroladores son:

- CPU (unidad central de procesamiento)
- Memoria ROM (memoria de solo lectura)

- Memoria RAM (memoria de acceso aleatorio)
- Líneas de entrada y salida (periféricos)

La unidad central de proceso posee, de manera independiente, una memoria de acceso rápido para almacenar datos denominados registros, que pueden ser de 8 o 32 bits; a esto se debe que se les denomine microcontroladores de 8 bits o de 32 bits (ver Figura 7).

La forma en que interactúan estos bloques antes mencionados dependerá de la arquitectura; las dos principales arquitecturas son: Von Neumann y Harvard .

Estas arquitecturas pueden tener procesadores de tipo CISC o RISC, cada una con sus propias características y formas de trabajo.

Arquitectura básica de los microcontroladores

La arquitectura de un microcontrolador permite definir la distribución de su funcionamiento, en la fabricación de los microcontroladores se utilizan la arquitectura Von Neumann y la arquitectura Harvard, estas arquitecturas pueden tener procesadores tipo CISC o RISC. Esto se debe al número de instrucciones que se necesite y la versatilidad de ejecución (ver Figura 8).

Cuando se empezó a utilizar los microcontroladores tenían la arquitectura Von Neumann, que se caracteriza por tener una sola memoria que almacena datos, así como también instrucciones; para acceder a esta información se procede a través de un bus constituido únicamente para datos y control

La arquitectura Harvard dispone de dos memorias independientes: una que contiene solo instrucciones, y otra únicamente datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias (ver Figura 9).

Es importante hacer un análisis de la arquitectura de los microcontroladores porque en los actuales momentos los programas son cada vez más grandes y complejos, esto demanda mayor velocidad en el proceso de la información, lo que implica la utilización de microprocesadores rápidos y eficientes. Los avances tecnológicos de

Figura 5. Tarjeta de red
Nota. Tarjeta electrónica incorporadas al microcontrolador y el circuito integrado controlador (Nobbot, 2020).

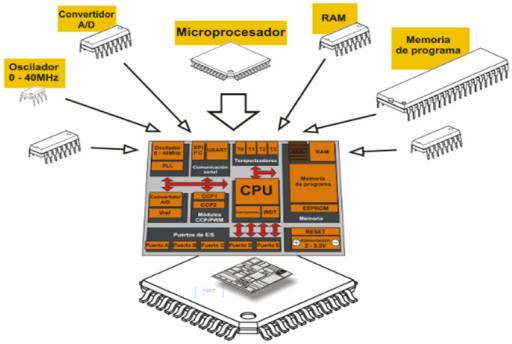


Figura 6. Estructura del microcontrolador
Nota. Los módulos importantes de un microcontrolador (Aprendiendo Arduino, 2019).

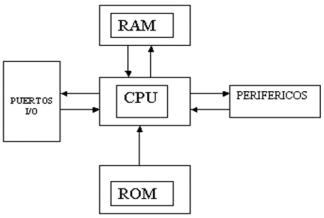


Figura 7. Partes de un microcontrolador
Nota: El microcontrolador y microprocesador en un solo circuito integrado (Ingeniería Mecafenix, 2019).

los semiconductores, han reducido las diferencias en la velocidad de procesamiento de los microprocesadores con la velocidad de las memorias.

Con la finalidad de tener un panorama general empezaremos indicando el significado de los términos CISC y RISC:

CISC (Complex instruction set computer) computadoras con un conjunto de instrucciones complejo.

RISC (Reduce instruction set computer) computadoras con un conjunto de instrucciones reducido (ver Figura 10).

Para seleccionar la tecnología del procesador se debe considerar las diferentes situaciones en que va a ser utilizado el microcontrolador, esto hace suponer que en un caso RISC reemplazará al CISC; otro factor muy importante para aplicar una determinada arquitectura del microprocesador son las condiciones técnicas de trabajo y, sobre todo, la rentabilidad, incluyendo los costos de software.

El objetivo principal es incrementar el rendimiento del procesador, ya sea optimizando alguno existente o creando uno nuevo. Para esto se debe considerar tres áreas principales que el microprocesador en su diseño debe tener, éstas son:

- Arquitectura
- Tecnología del proceso
- Encapsulado

En cuanto a la tecnología de proceso, se refiere a los materiales y técnicas utilizados en la fabricación del circuito integrado, que infiere mucho en el encapsulado, determinando cómo se integra un procesador de un sistema funcional, que de alguna manera establece la velocidad del dispositivo.

Con el anterior contexto podemos determinar que la tecnología de proceso y el encapsulado son importantes en la elaboración de procesadores muy rápidos, pero la arquitectura del procesador es lo que hace la diferencia entre el rendimiento de una y otra en la CPU (Central Processing Unit).

Existen tres tipos de conjuntos de instrucciones para definir cómo el procesador almacena los operadores en la CPU, éstos son:

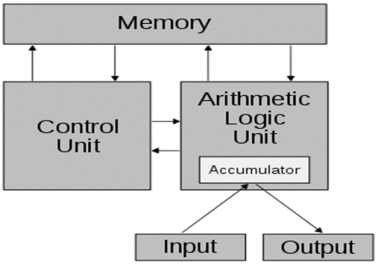


Figura 8. Arquitectura Von Neumann
Nota. Es la combinación de la arquitectura Von Neumann (Wikipedia, 2019).

Figura 9. Estructura Harvard
Nota. En el gráfico se puede observar dos memorias separadas, pero con un solo bus de conexión de entrada y salida (Lifeder.com, 2019).

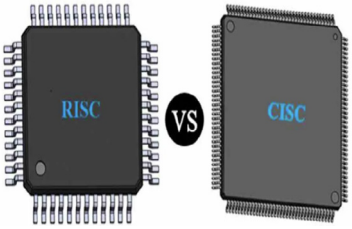


Figura 10. Arquitectura RISC vs. CISC
Nota. Los tipos de encapsulado y los complejos programas para que se ejecuten a grandes velocidades (Wikipedia, 2019).

- En pilas
- En acumulador
- Basadas en registros

En relación con los microcontroladores, su arquitectura está definida por parámetros específicos:

- Un sistema de procesamiento
- Memorias no volátiles, de lectura y escritura
- Elementos adicionales para su funcionamiento

En los microcontroladores, cuando nos referimos a los recursos auxiliares, podríamos nombrar a:

- Circuitos de reloj
- Temporizadores
- Conversores AD y DA
- Comparadores analógicos (ver Figura 11)

Los microcontroladores utilizados en este libro son de la familia 16FXX

Los microcontroladores PIC

Como habíamos indicado, existen varias empresas en la actualidad que se encuentran fabricando microcontroladores, llamados también minicomputadores, los cuales poseen variedad de modelos que facilitan la construcción de diversos proyectos, profundizaremos el estudio de los microcontroladores PIC de Microchip Tecnology Inc.

En definitiva, un PIC es un circuito integrado programable, capaz de ejecutar la programación grabada en su memoria,

El PIC 16F628A

El microcontrolador utiliza una arquitectura Harvard con la finalidad de optimizar la lectura y escritura simultáneamente en 2 memorias, y la versatilidad del procesamiento de instrucciones y datos, esto conlleva a la utilización de las memorias independientes con un método de acceso por buses con características específicas, 8 líneas para datos y 14 para instrucciones. (ver Figura 12). En este

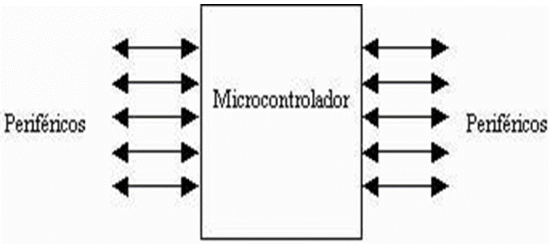


Figura 11. El microcontrolador
Nota. Los periféricos pueden ser programados como entradas o salidas. (Microcontroladores Microchip, 2019)

tipo de microcontroladores el proceso de información se realiza por el método del paralelismo, ajustándose a la velocidad de funcionamiento de un determinado circuito electrónico (ver Figura 13).

Los microcontroladores necesitan de circuitos externos para su funcionamiento, como la fuente de alimentación, el reloj, el cual indica la velocidad con que el microcontrolador debe trabajar. Los circuitos osciladores externos están conformados por elementos pasivos como resistencias y condensadores, y el elemento principal que es el cristal, así tenemos el RC oscilador resistencia condensador, XT condensadores y cristal (ver Figura 14).

En el diseño de circuitos con osciladores externos no se ha tomado en cuenta que con el uso van teniendo ligeras variaciones y que llegan en un momento determinado a cambiar de frecuencia, por eso en los diseños es muy recomendable tener los osciladores internos, ya que existe una gran variedad de encapsulados (ver Figura 15).

En la práctica encontramos una necesidad de conocer la estructura de un microcontrolador y la nomenclatura de cada uno de los terminales del circuito integrado o encapsulado del PIC. Estos terminales pueden servir como entradas o salidas, dependiendo el diseño y la programación que realicemos.

El microcontrolador PIC 16F628A utiliza la arquitectura RISC con un set de 35 instrucciones con una gama media de la familia de los

PIC, es muy notorio analizar la memoria de datos, la memoria del programa, además se puede diferenciar las conexiones de estas memorias, y la distribución de los terminales.

En cuanto se refiere a la memoria ROM, una parte no debe ser volátil con relación a la memoria de instrucciones, puesto que está destinada a contener el programa de instrucción que caracteriza la aplicación, y otra parte de la memoria será tipo RAM, en este caso volátil, cuya función principal será guardar las variables y los datos. El PIC tiene una arquitectura de 8 bits, con una velocidad de operación desde DC hasta 20 mHz, tienen 16 terminales de entrada y salida disponible, una memoria del programa flash de 2048 words (2 k a 14 bits), SRAM de 224 bytes (ver Figura 16).

La tecnología del PIC es CMOS, esto quiere decir que consume bajísima corriente y es susceptible a los daños de la estática. Se recomienda utilizar un voltaje estable de cinco voltios, esto se puede lograr con la utilización del circuito integrado regulador 7805, también se recomienda no sobrepasar los niveles de corriente, tanto en la entrada como en la salida de los microcontroladores, esto quiere decir no sobrepasar los 25 Ma (ver Figura 17).

El microcontrolador PIC 16F877A

Se caracteriza por poseer una memoria de programa de 8192 words, memoria EEPROM de 256 bytes, memoria RAM de 368 bytes y 33 terminales de entrada y salida, los cuales se dividen en cinco puertos: A, B, C, D, E, estos puertos trabajan con la siguiente distribución de bits:

- Puerto A trabaja a 6 bits
- Puerto B trabaja a 8 bits
- Puerto C trabaja a 8 bits
- Puerto D trabaja a 8 bits
- Puerto E trabaja a 3 bits

Otra característica importante es que tiene 8 conversores analógico-digitales A/D, este microcontrolador se utiliza para proyectos avanzados que requieren mayor número de entradas y salidas (ver Figuras 18 y 19).

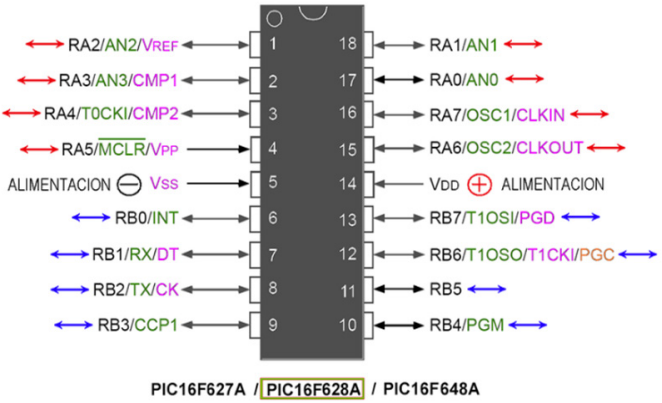


Figura 12. Estructura física de los PIC 16fxx
Nota. En la figura se puede notar la nomenclatura de los terminales para su programación Microcontroladores Microchip (2019).

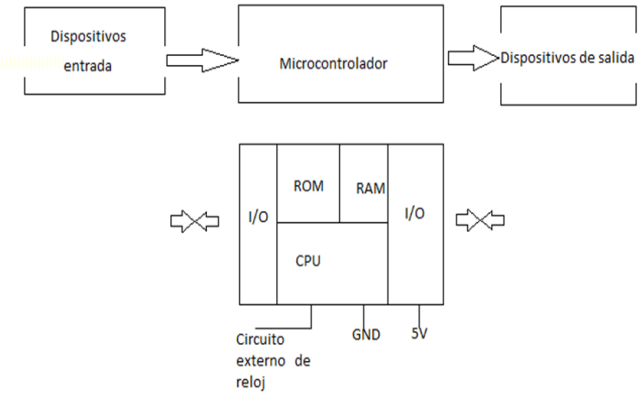


Figura 13. Estructura interna del microcontrolador
Nota. El microcontrolador necesita de circuitos externos para su funcionamiento (Electrónicas y Chiluisa (2017).

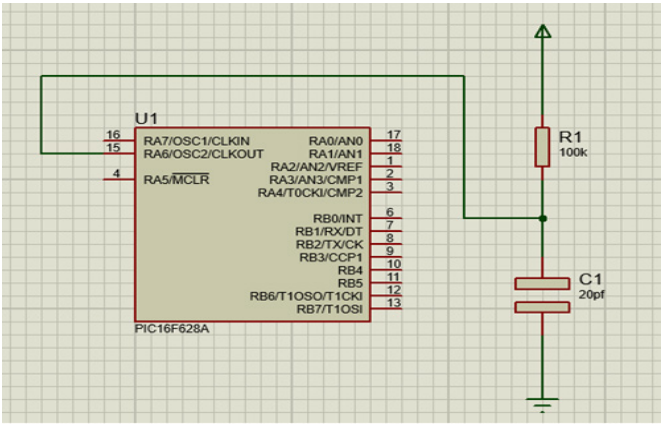


Figura 14. Oscilador externo RC
Nota. Formado por un circuito resistivo capacitivo, la frecuencia dependerá de los valores y la fuente de alimentación (Electrónicas y Chiluisa, 2017).

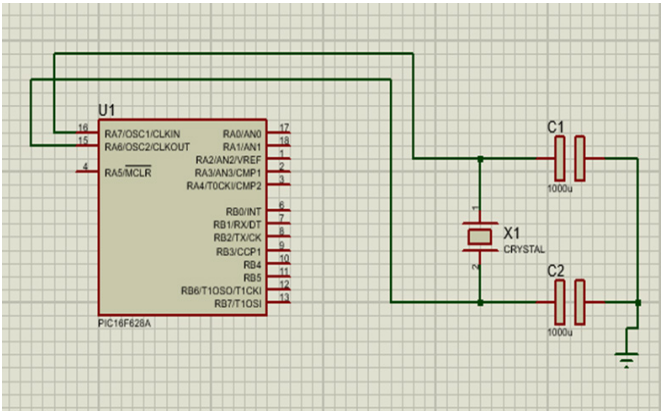


Figura 15. Oscilador externo XT
Nota. Dependerá mucho de los valores de los condensadores para su frecuencia en HS Cristal de alta velocidad y LP cristal de baja frecuencia y bajo consumo de potencia (Electrónicas y Chiluisa, 2017).

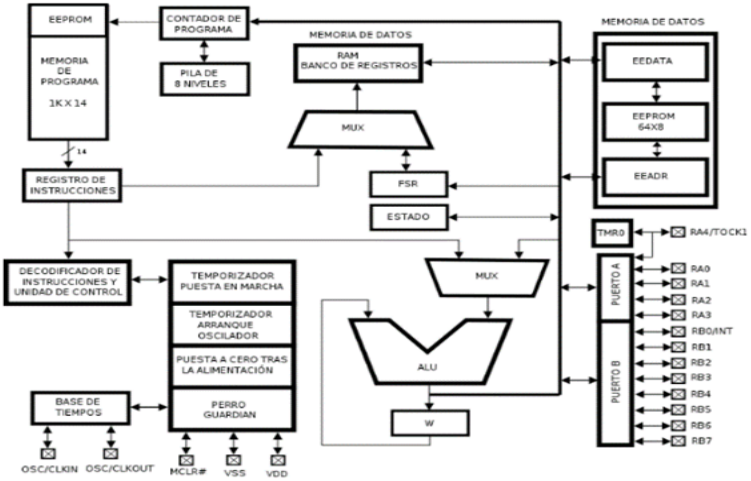


Figura 16. El microcontrolador y sus bloques

Nota. Estructura de los puertos A-B que son los terminales de entrada y salida (Electrónicas y Chiluisa, 2017).

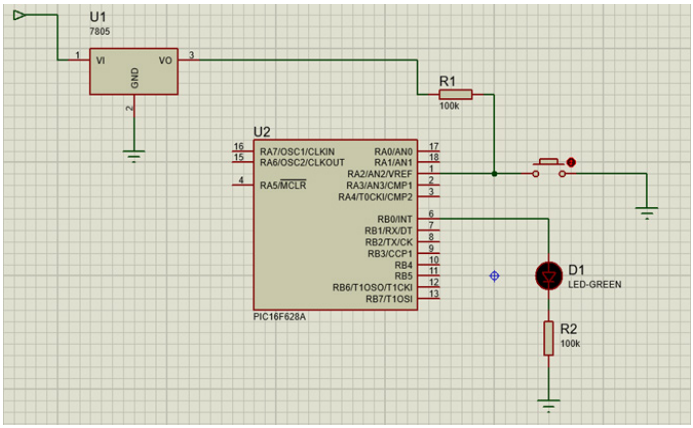


Figura 17. Diagrama básico de un PIC

Nota. Diagrama de un microcontrolador con un diodo led, pulsador y el integrado regulador de 5 voltios (Electrónicas y Chiluisa (2017).

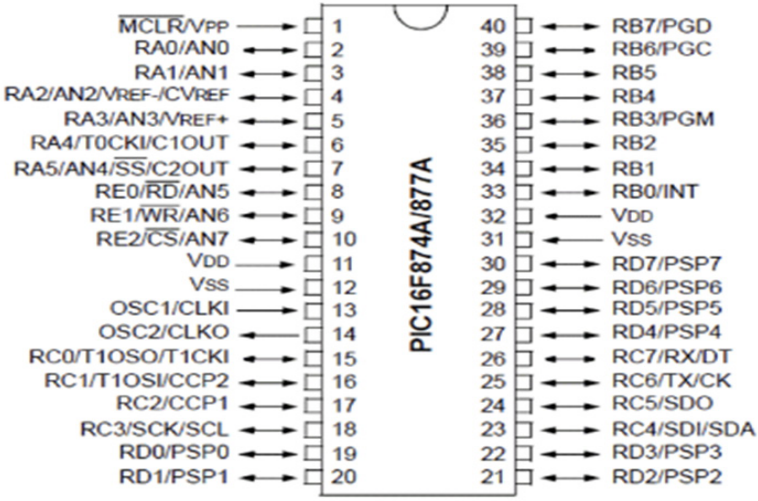


Figura 18. Distribución de los terminales del pic
Nota. El PIC con la nomenclatura de los puertos y las entradas y salidas de los terminales (Components101, 2019).

Figura 19. Encapsulado del PIC 16F877A
Nota. Es la forma física real del pic (Components101, 2019).

Capítulo II

Programación de los microcontroladores

Software para los microcontroladores

Para el diseño y programación de dos microcontroladores se utilizan dos software importantes: el software PROTEUS, que sirve para realizar las simulaciones y diagramas esquemáticos electrónicos, y el software MicroCode Studio, que servirá para su programación.

Instalación del software para los PIC

Para la instalación del software MicroCode Studio, se deben seguir los siguientes pasos:

- Ingrese a la página de MicroCodeStudio.com
- Descargue el software
- Descomprima en una carpeta con el nombre de su interés
- Instale (ver Figuras 20, 21, 22 y 23).

En este momento se debe esperar unos instantes para que se instale el software, posteriormente se debe seleccionar las dos carpetas MPA y PBP y copiar en el lugar que se instaló el software MicroCode Studio (ver Figuras 24, 25 y 26).

En la figura se muestra cómo se copia las dos carpetas necesarias para las librerías y PCB de los microcontroladores. Es recomendable tener instalado el software en el disco donde se encuentra el sistema operativo, para su óptimo funcionamiento (ver Figura 27).

Los procedimientos recomendados son con la finalidad de que el software MicroCode Studio sea una herramienta útil para la programación de los microcontroladores, también este procedimiento se puede revisar en la página oficial de los microcontroladores.

Finalmente, se debe configurar el IDE de MicroCode Studio para poder obtener los archivos necesarios para su grabación y su simulación. Las simulaciones de las grabaciones de los microcontroladores se pueden realizar en el software PROTEUS en sus diferentes versiones, de acuerdo a las tecnologías de los equipos informáticos de los usuarios.

En las siguientes figuras se procede a indicar cómo debe estar configurado el IDE de MicroCode Studio con la finalidad de realizar las prácticas de investigaciones necesarias con los microcontroladores de PIC (ver Figura 28).

Si usted ha logrado obtener el icono de MicroCode Studio como se muestra en la figura 28 significa que ha realizado correctamente la instalación, ahora se empezará a realizar la configuración de IDE de los microcontroladores para poder generar los archivos necesarios. Estos archivos son x.asm, x.bas, x.hex, x.mac x.PWI, donde x es el nombre del proyecto.

El archivo x.hex es el que nos permite realizar la grabación del microcontrolador, así como también la simulación en el software PROTEUS. Es decir, este archivo que nos proporciona el software MicroCode Studio es el más importante para realizar la práctica y la simulación.

Si no obtenemos estos archivos, indica que el software MicroCode está mal instalado, y se tendrá que volver a hacerlo. Para instalarlo nuevamente, primero se tiene que borrar los archivos de registro, el no hacerlo ocasiona grandes problemas en el software (ver Figuras 29, 30 y 31).

En este momento se procede a la configuración para compilar y programar las directrices del proyecto a realizarse (ver Figuras 32, 33, 34, 35 y 36).

Programado de los PIC

Para empezar a programar en los microcontroladores se debe conocer un conjunto de instrucciones que permitan interactuar, estas son: encendido, apagado, espera; por lo general, estas acciones se utilizan para poner en funcionamiento, esperar y finalmente apagar, la espera o pausa es de milisegundos, también se debe consi-

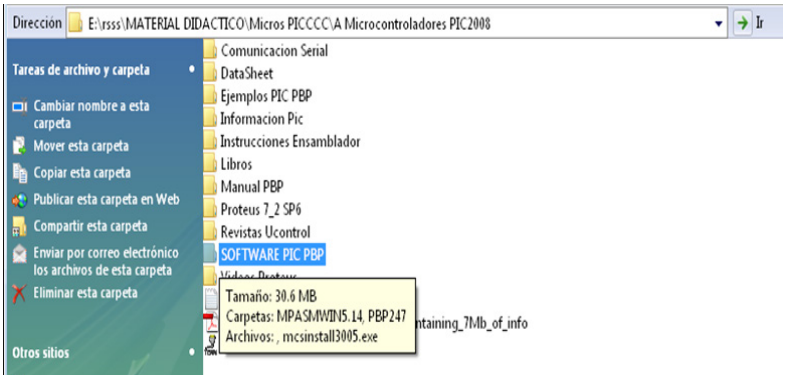


Figura 20. Carpeta de contenido del software
Nota. El software debe ser copiado en el HD del computador (Electrónicas y Chiluisa, 2017).

Figura 21. Contenido del software IDE MicroCode Studio
Nota. Se tiene que ejecutar el archivo para realizar la instalación (Electrónicas y Chiluisa, 2017).

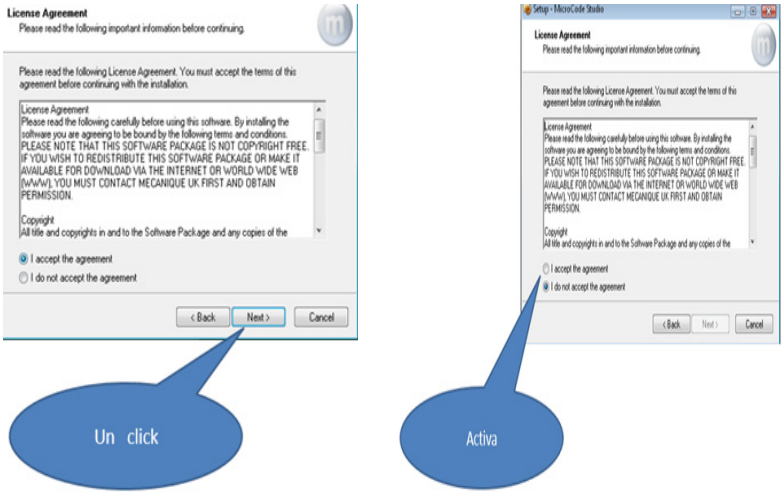


Figura 22. Parámetros legales de MicroCode
Nota. Primero tiene que dar un clic en siguiente y posteriormente digitar en la opción de aceptar los parámetros del software (Electrónicas y Chiluisa, 2017).

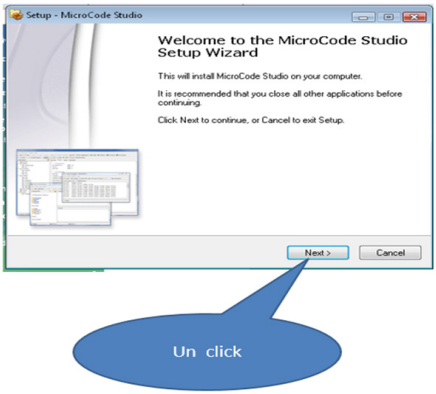


Figura 23. Portada de bienvenida al software MicroCode Studio
Nota. Es el inicio de la instalación de MicroCode (Electrónicas y Chiluisa, 2017).

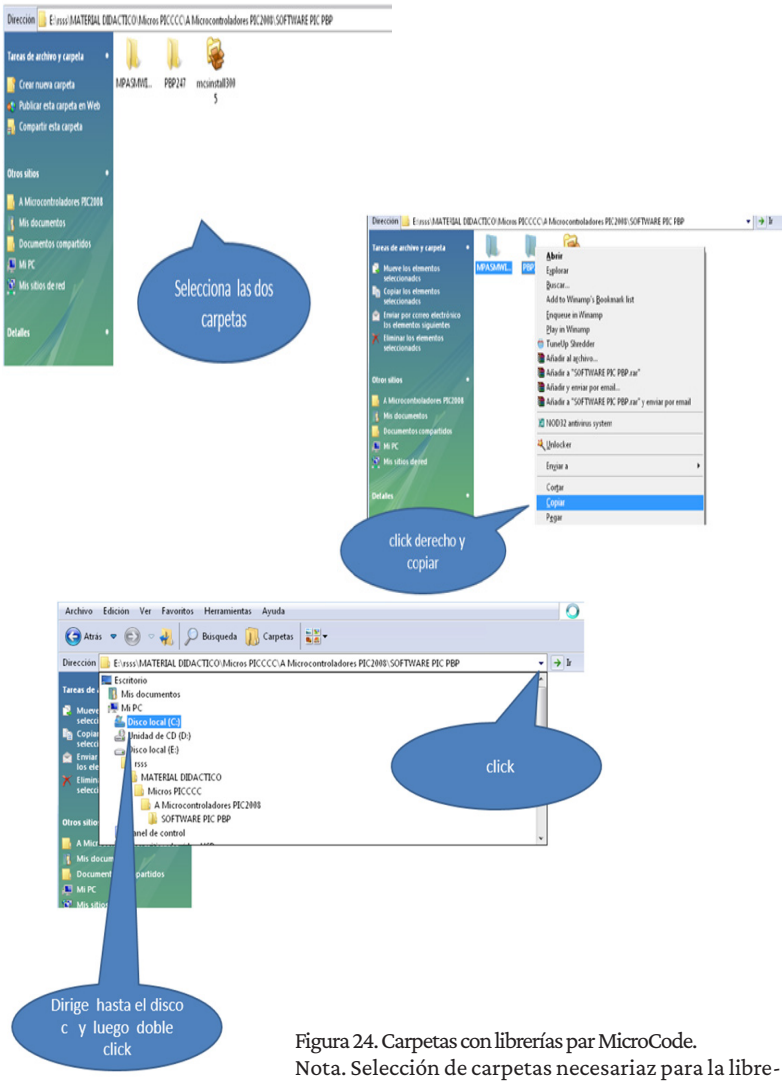


Figura 24. Carpetas con librerías par MicroCode.
Nota. Selección de carpetas necesariaz para la librería de MicroCode (Electrónicas y Chiluisa, 2017).

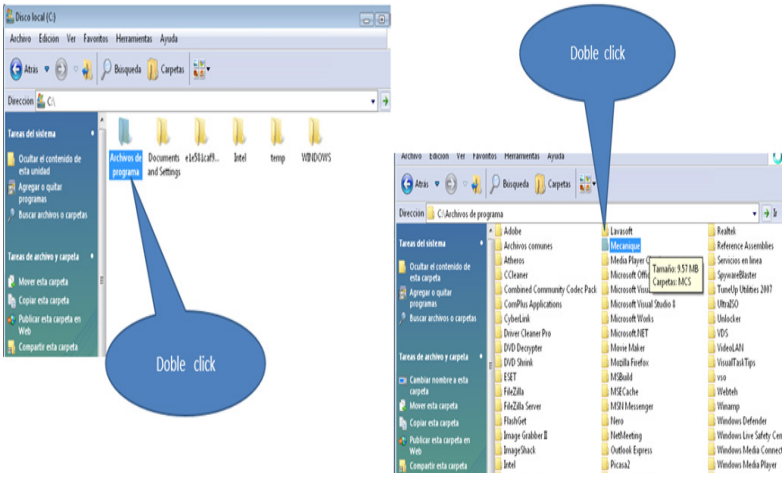


Figura 25. Lugar para pegar las carpetas de las librerías
Nota. Se sugiere realizar este procedimiento con la finalidad de que MicroCode tenga las librerías actualizada de los diversos modelos de microcontroladores (Electrónicas y Chiluisa, 2017).

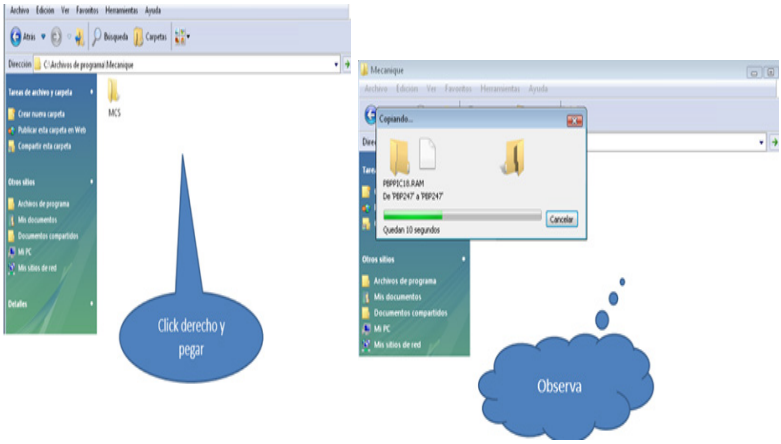


Figura 26. Carpeta donde se instaló el software MicroCode
Nota. Se debe pegar las librerías seleccionadas (Electrónicas y Chiluisa, 2017).

Figura 27. MicroCode y las librerías
Nota. La carpeta que instaló el software, aparecerá con el nombre Mecanique debe contener MCS y las dos carpetas de las librerías.

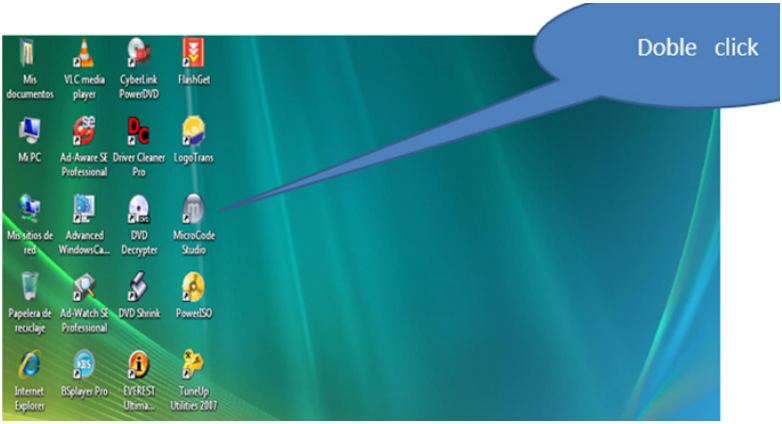


Figura 28. Icono de MicroCode Studio
Nota. Cuando se instala MicroCode Studio, por defaul tenemos un archivo ejecutable en el escritorio (Electrónicas y Chiluisa, 2017).

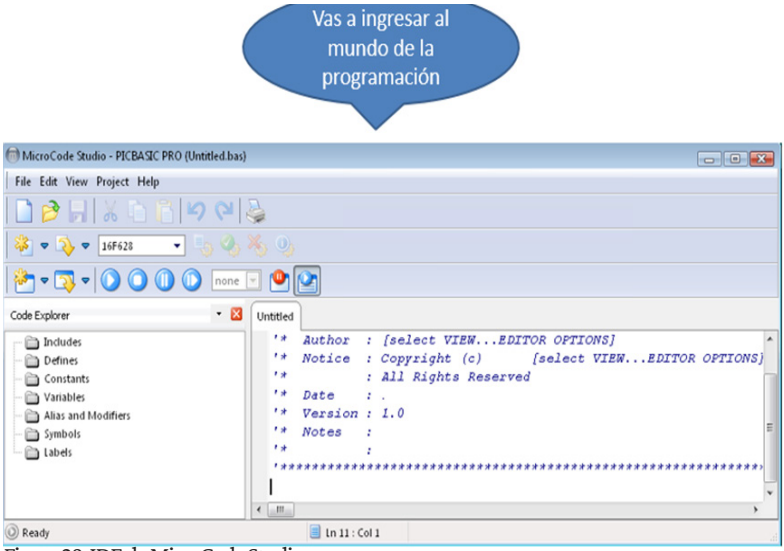


Figura 29. IDE de MicroCode Studio
Nota. Para que este IDE esté operable se debe configurar (Electrónicas y Chiluisa, 2017).

Figura 30. Configuración de IDE MicroCode Studio
Nota. Debe dirigirse a la opción de View y luego a la opción Editor Option, para que se desplieguen las opciones necesarias para la configuración (Electrónicas y Chiluisa, 2017).

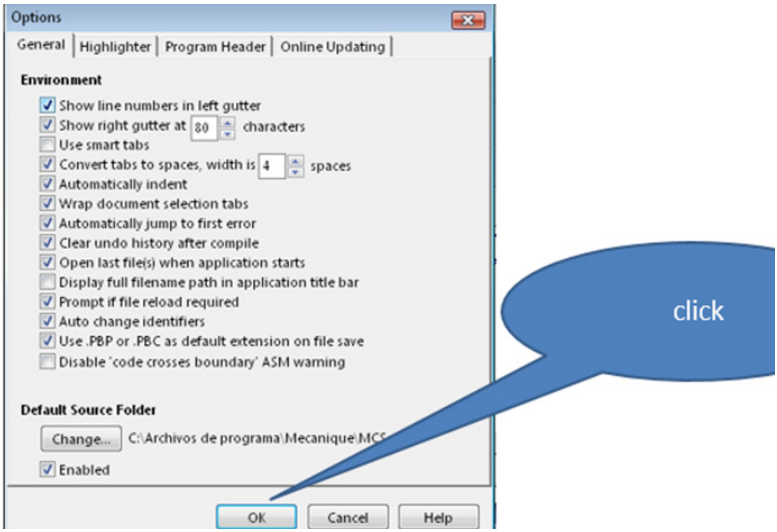


Figura 31. Características del IDE
Nota. Seleccionar las opciones de la figura (Electrónicas y Chiluisa (2017).

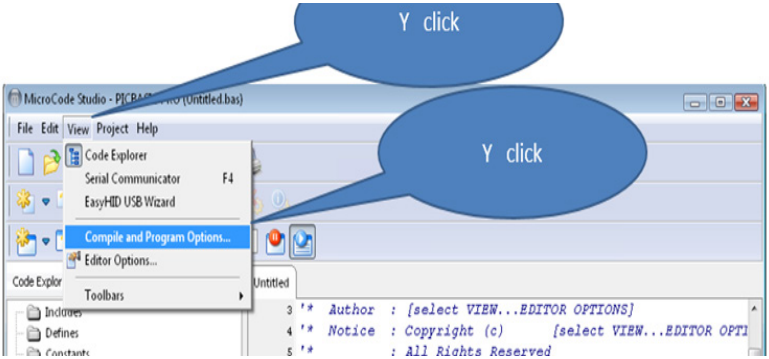


Figura 32. Compilador y grabación
Nota. Debe seleccionar la opción compiler y program para poder seleccionar el ensamblador (Electrónicas y Chiluisa, 2017).

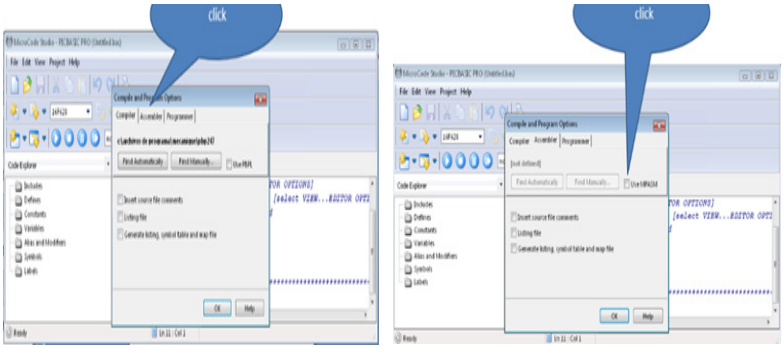


Figura 33. Configuración de IDE MicroCode Studio
Nota. En las figuras se indica el procedimiento a seguir para que el IDE pueda generar el código del ensamblador (Electrónicas y Chiluisa, 2017).

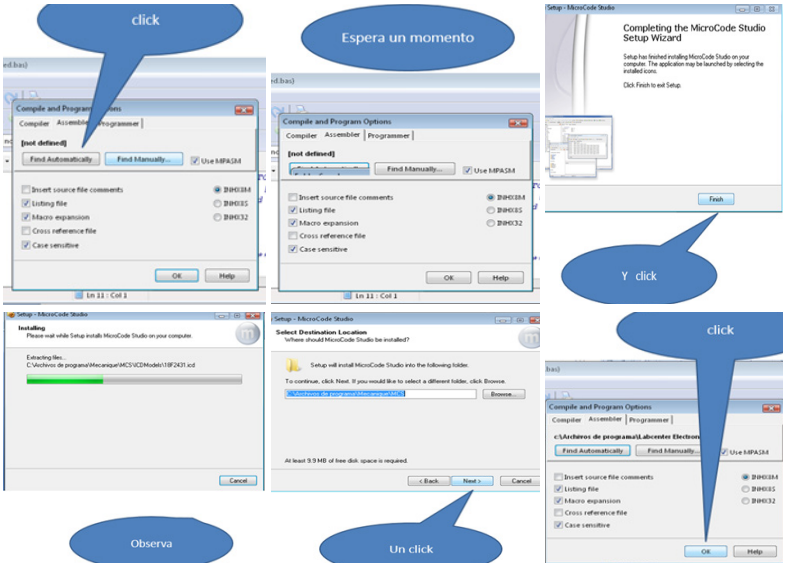


Figura 34. Configuración del programador
Nota. Se debe seguir los pasos para que el IDE MicroCode genere el código. hex

Figura 35. Archivos que debe generar MicroCode
Nota. Si se encuentra bien instalado el software en el momento de compilar se generará los archivos que se indica en la figura (Electrónicas y Chiluisa, 2017).

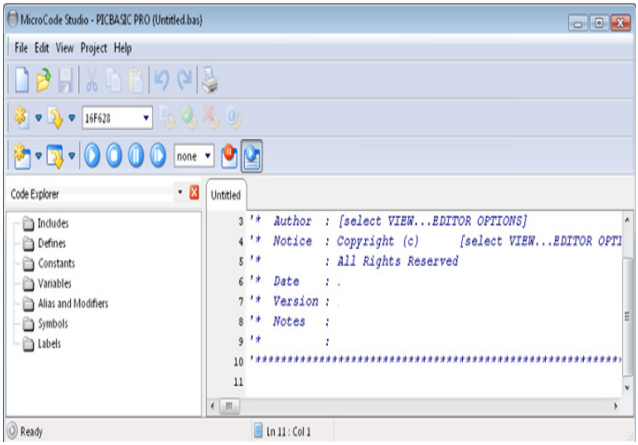


Figura 36. IDE MicroCode
Nota. El IDE MicroCode estudio listo para realizar los ejercicios de programación (Electrónicas y Chiluisa, 2017).

derar los puertos necesarios para las entradas y salidas, ejemplo:

PORTB.0; indica que se ha seleccionado el puerto B.0

PORTA.0; indica que sea seleccionado el puesto A.0

Como podemos observar la sintaxis radica en llamar al puerto con la letra que especifica el nombre seguido de un punto y una parte numeral, en este caso es cero.

Una vez que se ha definido los puertos que va a utilizar, se puede también modificar las acciones que realizarán estos puertos, es decir, si pueden estar prendidos o apagados, si son entradas o salidas.

Realizaremos un ejercicio de encendido y apagado de un diodo LED con una pausa de 1000 milisegundos (ver Figura 37).

En este ejemplo se puede observar que se cumplen las tres características fundamentales de una programación:

- El nombre = Inicio,
- Acción = prendido y apagado uno LED,
- Fin de la programación = se repita al inicio

En el ejemplo se identifica que está programado en el puerto B y se utiliza las sentencias de HIGH, LOW, PAUSE, GOTO, esta última sirve para que se repita las acciones mientras esté alimentado con el voltaje respectivo (ver Figura 38).

En la programación existen diferentes caminos que llegan al mismo objetivo, la clave es escribir correctamente la sintaxis, a continuación, un ejemplo a seguir (ver Figura 39).

Para emplear este tipo de sintaxis debemos utilizar la sentencia TRIS seguido del puerto que se va a utilizar A, B, C; TRISA, TRISB, TRISC...

Los PIC y las subrutinas

Las subrutinas en los microcontroladores se refieren a una parte de la programación, especificando que el microcontrolador realice una determinada acción, es decir, que puede cumplir dos o más trabajos en un mismo programa, las instrucciones en el código de programación son: GOTO, GOSUB.

En el IDE de MicroCode Studio se reconocen fácilmente y tienen una función importante en las líneas de programación de repeticiones.

```
Inicio:
        HIGH portb.1      ;enciende el LED
        Pause 1000        ;espera un segundo
        LOW portb.1       ; apaga el LED
        Pause 1000        ;espera un segundo
GOTO inicio               ; se repite el programa
```

Figura 37. Programación de un diodo led
Nota. Todo programa de encendido y apagado de un diodo led (Urbina, 2012).

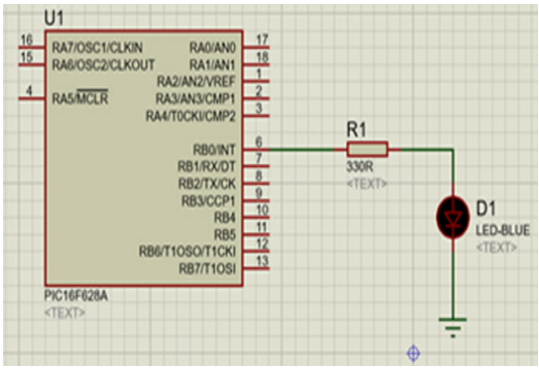


Figura 38. El microcontrolador 16F628A y un diodo led
Nota. Diagrama esquemático realizado en PROTEUS (Electrónicas y Chiluisa, 2017).

```
11
12 TRISB=%0 ; hacemos salida el bit B.0 del puerto B
13
14 INICIO:  ; ETIQUETA PARA UN LAZO REPETITIVO
15
16          PORTB=%0 ; sacamos 0L por el puerto B.0
17          PAUSE 1000 ; PAUSA DE 1 SEGUNDO
18          PORTB=%1 ; sacamos 1L por el puerto B.0
19          PAUSE 1000 ; PAUSA DE 1 SEGUNDO
20
21 GOTO INICIO   ; SALTA A LA ETIQUETA INICIO
22
```

Figura 39. Programación
Nota. Se utiliza codificación binaria para el encendido y apagado de los diodos led.

La aplicación de la instrucción GOTO es utilizada para repetir una acción mientras tenga energía, generalmente es utilizada para establecer un tipo de frecuencia de onda cuadrada, se puede mencionar una similitud de trabajo como del circuito integrado 555 en modo estable que genera pulsos, esta acción se realiza con los microcontroladores utilizando las sentencias HIGH, LOW, PAUSE y GOTO.

El conjunto de las sentencias antes mencionadas forma parte de una acción de la programación y se mantienen ejecutándose mientras tenga un fluido eléctrico. En programación se le conoce como subrutina, y para saltar a otra acción necesita la vinculación de otras sentencias como el FOR, IF, ELSE, etc. (ver Figura 40).

Analizaremos las líneas de la programación, en primera instancia en la línea número 11 está definido el nombre del programa (mk) seguido de dos puntos, y en la línea doce utilizamos la sentencia HIGH seguido de nombre del terminal portb.0 indicando que el diodo LED está prendido, en la línea trece se utiliza la sentencia PAUSE seguido del tiempo (200), en la línea catorce la sentencia LOW seguido del nombre del terminal que indica que el diodo LED se encuentra apagado, en la línea quince nuevamente utilizamos la sentencia PAUSE seguido del tiempo (200) y en la línea dieciséis se utilizan la sentencia GOTO seguido del nombre del programa (mk).

De esta manera podemos evidenciar que se puede programar el tipo de frecuencia y la velocidad que necesitamos que esté oscilando.

GOSUB

Esta instrucción permite realizar un salto de línea en la programación a una etiqueta que esté asignada, es muy útil GOSUB para realizar programas con varias etiquetas que cumplan diversas acciones, en la práctica se utiliza al microcontrolador para alarmas o función específica de un dispositivo en una determinada acción (ver Figura 41).

La utilización de las declaraciones GOTO y GOSUB, como podemos observar al momento que se utiliza GOSUB, direcciona inmediatamente a la etiqueta que le acompañe y siempre RETURN para que regrese, el GOTO regresa a la etiqueta principal. Se utilizan estas declaraciones en proyectos electrónicos que efectúan actividades de forma

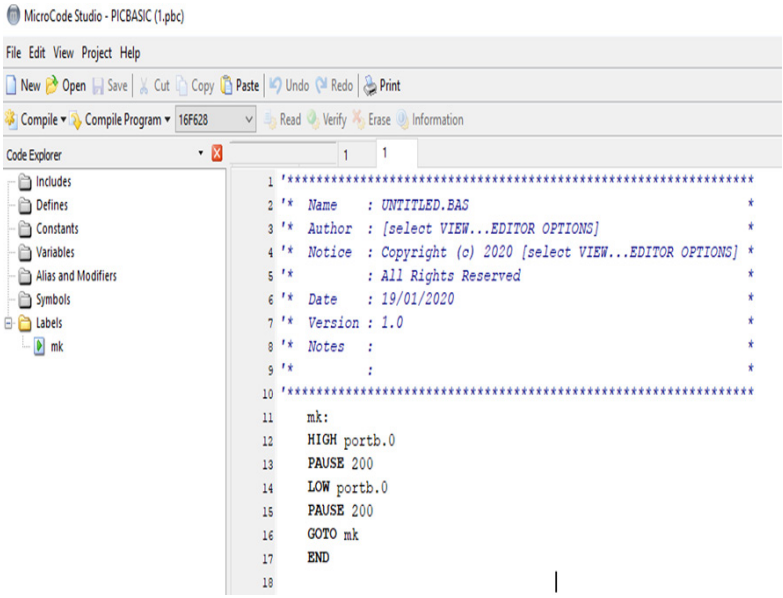


Figura 40. Subrutinas
Nota. Con la utilización de las sentencias se puede lograr un generador de pulsos.

progresiva, ejemplo: encienda 3 diodos LED, active un motor por 5 minutos, y posteriormente active una sirena.

Las etiquetas utilizadas serán 3, una para cada acción, se realizará acción por acción de forma secuencial, sin necesidad de que el usuario intervenga (ver Figura 42).

Uso de variables

En forma general, existen 2 tipos de variables en programación: las numéricas y las alfanuméricas, a + b = N.

En donde a y b son números y N es el número resultado de la operación matemática, en este caso la suma. Como podemos ob-

servar, las variables, como su nombre lo indica, pueden ser cualquier número, pero estos números tienen que tener una acción con un operador matemático: suma, resta, multiplicación, división.

En esta analogía cada variable adopta un valor, y el nombre de la etiqueta de la programación también puede variar o ser modificado.

Las variables BIT, BYTE, WORD, son creadas para guardar datos en la memoria RAM (random access memory), los valores de estas variables son:

BIT= 2^1 el valor es 0,1
BYTE= 2^8 el valor es de 0-255
WORD = 2^{16} el valor es de 0-65535

La memoria RAM trabaja únicamente cuando el microcontrolador está alimentado con energía eléctrica, en el momento que se desconecta el fluido eléctrico al microcontrolador los datos de la memoria RAM se borran. Al momento de programar podemos cambiar los nombres de los terminales del microcontrolador por un nombre que se necesite, acompañado del espacio requerido, ejemplo:

- led VAR.BIT
- led1 VAR portb.1

En la primera línea asignamos el espacio de memoria, y en la segunda línea cambiamos el nombre del terminal del microcontrolador por led1, los nombres led, led1 son nombres aleatorios pero la palabra VAR es la declaración para signar el espacio en memoria o cambio de nombre del terminal.

Sem VAR BIT crea una variable y asigna un tamaño de un bit, es decir, 0 o 1.

Sem VAR BYTE crea una variable y asigna un tamaño de 8 bits, es decir, de 0 a 255.

Sem VAR WORD crea una variable y asigna un tamaño de 2 bits, es decir, de 0 a 65535.

Potenciómetro VAR PORTB.0 ésta es la sintaxis para cambiar el nombre de los puertos en la programación (ver Figura 43).

En programación se debe asignar nombres a los puertos para permitir personalizar los nombres de los terminales de microcontrolador.

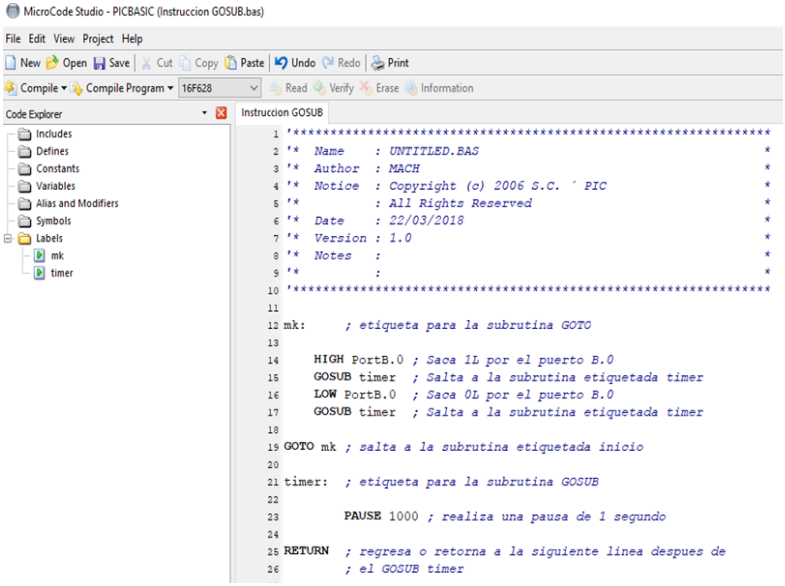


Figura 41. Subrutina GOSUB

Nota. Sentencia GOSUB para la programación de un microcontrolador en el encendido y apagado de un diodo led (Electrónicas y Chiluisa, 2017).

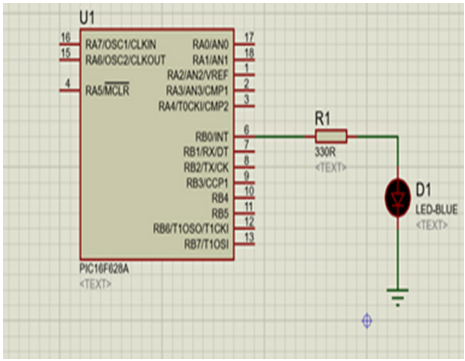


Figura 42. Diagrama esquemático de ON-OFF de un led

Nota. El microcontrolador PIC programado en el puerto B.0 y un diodo led (Electrónicas y Chiluisa, 2017).

Instrucciones de repeticiones

Permite repetir instrucciones, existe una gran variedad de declaraciones para realizar estas acciones como son: FOR, NEXT, IF, ELSE.

Las diversas herramientas trascendentales de PIC BASIC PRO permiten escribir varias instrucciones en una misma línea solo con digitar (:) ejemplo HIGH LED: LOW LED 2.

Cuando se está elaborando proyectos con display se recomienda utilizar la declaración LOOKUP (ver Figura 44), que sirve para obtener un valor constante en una tabla, esto se realiza según el número de veces que se repita el FOR NEXT.

```
Li VAR BYTE     ; crea una variable Li
Car VAR BYTE    ; crea una variable Car
TRISB = 0       ; todo el puerto B como salida
Mk:
   FOR Li=0 to 15  ; repeticiones de 0 a 15
    LOOKUP Li, [64,121,36,48,25,18,2,120,0,16,8,3,70,33,6,14],Car ; toma
uno por uno cada valor de la tabla constante y lo guarda en la variable Car.
     portb = Car     ; sacar el contenido Car por el puerto B
     PAUSE 100       ; espera de 0,1 seg
      NEXT Li        ; siguiente repetición
GOTO Mk
END
```

Los pulsadores y switch

(ver Figura 45)

Instrucciones condicionantes

En los dispositivos electrónicos es muy rutinario manipular pulsadores que realicen una cierta actividad ON-OFF, es decir, siempre estaremos comparando, se utilizan las instrucciones IF, ELSE, ENDIF, para realizar preguntas, en los microcontroladores se los utiliza para los

```
LED VAR PORTB.0 ; CAMBIO DE NOMBRE AL PUERTO B.0 POR LED
R VAR WORD ; VARIABLE R DE TAMAÑO 65536

INICIO:   ; ETIQUETA DE SALTO LLAMADA INICIO

        HIGH LED ; 1L POR EL PUERTO B.0
        GOSUB TIMER ; SALTA A REALIZAR UNA PAUSA DE 5 MINUTOS
        LOW LED ; 0L POR EL PUERTO B.0
        GOSUB TIMER ; SALTA A REALIZAR UNA PAUSA DE 5 MINUTOS

GOTO INICIO ; SALTA A LA ETIQUETA INICIO

TIMER:  ; ETIQUETA DE SALTO LLAMADA TIMER

        FOR R=1 TO 300  ; INICIO DEL LAZO REPETITIVO DE 1 A 300

            PAUSE 1000  ; PAUSA DE 1 SEGUNDO

        NEXT  ; CONTINUA AL LAZO FOR HASTA QUE R=300

RETURN  ; REGRESA A LA SIGUIENTE LINEA DONDE SE QUEDO
```

Figura 43. Utilización de la sentencia VAR
Nota. En el gráfico se puede apreciar la forma de utilizar la sentencia VAR (Electrónicas y Chiluisa, 2017).

```
'*****************************************************
DI VAR BYTE    ;CREA VARIABLE DI
DAT VAR BYTE  ;CREA VARIABLE DATA
TRISB=0         ;TODO EL PUERTO COMO SALIDA
MI:
FOR DI= 0 TO 15   ;PARA REPETIR DE 0 A 15
LOOKUP DI,[64,121,36,48,25,18,2,120,0,16,8,3,70,33,6,14],DAT
          ;TOMA UNO POR UNO CADA VALOR CONSTANTE Y LO GUARDA EN DATA
PORTB=DAT;SACA EL VALOR DE DATA POR EL PUERTO B
PAUSE 500
NEXT DI     ;SIGUIENTE REPETICION
GOTO MI
END
```

Figura 44. Utilización de LOOKUP
Nota. Utilización de LOOKUP, FOR (Electrónicas y Chiluisa, 2017)

proyectos electrónicos con pulsadores, teclados, sensores, y otros dispositivos que cumplan la función de un interruptor o switch.

Este tipo de sentencias se utilizan para programaciones de comparación o, también, que se ejecuten acciones mientras se cumpla con la condición, su sintaxis es:

IF pregunta si es correcto THEN entonces
Realice la programación
ELSE, caso contrario, si no es correcto realice ...
ENDIF termina la instrucción IF

Un ejercicio del empleo de IF, THEN, ELSE, ENDIF es la utilización de un pulsador y un diodo LED, que al momento de presionar el pulsador se prenda el LED. Para la programación de esta acción se realiza la siguiente programación:

```
LED VAR PORTB.0                 ; se cambia el nombre del terminal
                                  PORTB.0 por LED
Pulsador VAR PORTB.1            ; se cambia el nombre del terminal
                                  PORTB.1 por pulsador
Mk:                             ;  etiqueta de inicio de la programación.
    IF pulsador =0 THEN         ; pregunta si el pulsador es 0 entonces
            GOTO prendido       ; salte a la etiqueta prendido
            ELSE                ; caso contrario, si el pulsador sigue en
                                  1, entonces
            GOTO apagado        ; salta a la etiqueta apagado
            ENDIF               ; termina la instrucción IF
            GOTO Mk             ; salta al inicio
    prendido:                   ; etiqueta de salto
        HIGH LED                ; saque 1 lógico por el puerto B.0
        GOTO Mk                 ; etiqueta de salto
    Apagado:                    ; etiqueta de salto
        LOW LED                 ; saque 0 lógico por el puerto B.0
GOTO Mk                         ; salta al inicio
END                             ; fin del programa.
```

```
; LED INTERMITENTE CON VELOCIDAD VARIABLE
sube VAR portb.2
baja VAR portb.1
led VAR portb.0
mj VAR BYTE
mk VAR BYTE
mk= 100

MACH:
      HIGH led
      GOSUB mate
      LOW led
      GOSUB mate
      GOTO MACH

      mate:
           IF sube=0 THEN GOSUB restar
           IF baja=0 THEN GOSUB sumar
           FOR mj= 1 TO mk
           PAUSE 5
           NEXT
           RETURN

        sumar:
           IF mk>1500 THEN RETURN
           mk= mk+5
           RETURN

        restar:
           IF mk< 10 THEN RETURN
           mk=mk-5
           RETURN

          END
```

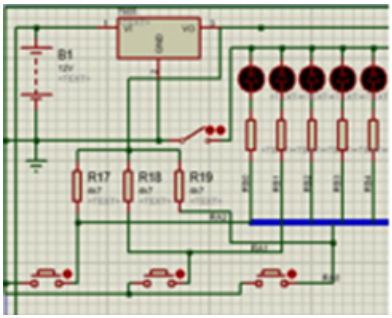


Figura 45. Ejercicio con pulsadores

Nota. Proyecto de luces que aumenta y disminuye su velocidad (Electrónicas y Chiluisa, 2017).

Como se dijo anteriormente, cuando se utilizan pulsadores intervienen las declaraciones IF, pero siempre estará seguida de THEN (ver Figura 46), en el caso de que se utilice el GOTO sale directamente a la línea donde se encuentra la etiqueta que esté asignada; esta etiqueta contiene líneas de programación para que realice un determinado juego de instrucciones con 1 lógico y 0 lógico, es decir, que pueden entrar normalmente cerrados o normalmente abiertos, en los circuitos electrónicos se puede lograr esta configuración de acuerdo al diseño de nuestro diagrama esquemático (ver Figura 47).

```
11 cmcon=7                    ; convierte todo el puerto en digital
12 mk:
13    IF portb.3= 0 THEN kl   ; pregunta si portb.3=0 para ir a kl
14    GOTO mk
15 kl:
16    HIGH porta.0
17    PAUSE 100
18    LOW porta.0
19    PAUSE 100
20 GOTO kl
21 END
```

Figura 46. Ejercicio del IF

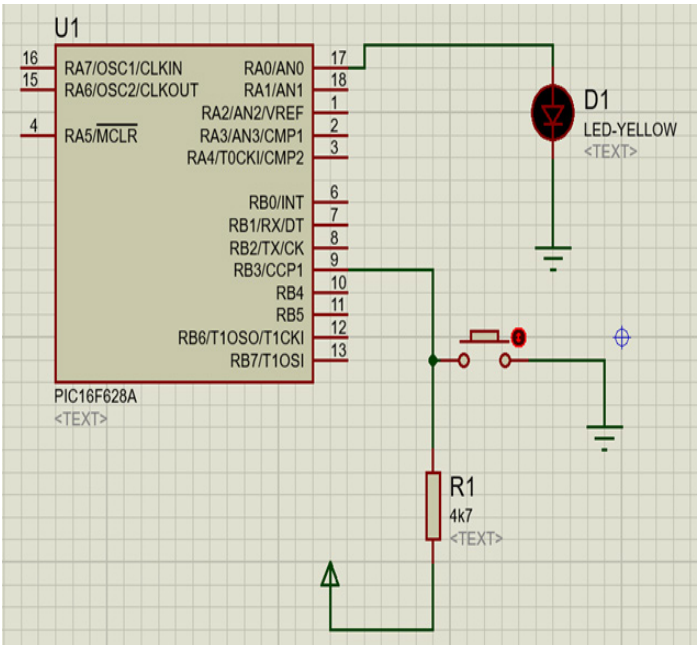


Figura 47. Pulsadores
Nota. El pulsador está configurado como 1 lógico (Electrónicas y Chiluisa, 2017).

Capítulo III

Circuitos electrónicos con los microcontroladores

Encendido de luces en forma secuencial de izquierda a derecha y viceversa

(ver Figuras 48 y 49)

Interface

Se refiere a la conexión de los microcontroladores a grandes potencias, esto se logra con elementos electrónicos como los relee o relay, SCR, TRIAC. En varias ocasiones se han realizados diseños en plantillas protoboar o placas de simulaciones electrónicas, donde la corriente que se utiliza es muy baja en el rango de los miliamperios, la difícil tarea es conectar cargas que funcionen con altas corrientes y voltajes como pueden ser de 220 V o 110 V a 10 amperios, para realizar la conexión del microcontrolador a estas cargas se necesita de un relee que realizará la conexión de forma analógica y electrónica (ver Figuras 50 y 51).

Una vez que se ha programado al microcontrolador para el encendido y apagado de un diodo LED del mismo terminal del microcontrolador se deriva con una resistencia a la base de un transistor bipolar NPN, el emisor se encuentra conectado a tierra, y el colector del transistor es el que se conecta a un extremo del terminal del relay, y el otro extremo se conectará a la fuente de voltaje que queremos aplicar, puede ser de 12, 5, 24 voltios.

El transistor trabaja en dos puntos: el de saturación y el de corte, activando y desactivando al relay que trabaja por electromagnetismo, cuando llega voltaje al embobinado del relay se activan las placas internas, esto permite que funcione un determinado sistema electrónico, esta acción del relay se le conoce como interruptor analógico (ver Figura 52).

Cuando se utiliza este tipo de elementos electrónicos se debe tomar en cuenta que sufren un desgaste los contactos de las placas del relay, todo dependerá de la carga de funcionamiento que realice este dispositivo electrónico (ver Figuras 53, 54 y 55).

Interface 4 salidas

(ver Figura 56)

Interface

(ver Figura 57)

Dispositivo LCD

El módulo de cristal líquido es utilizado para mostrar mensajes numéricos, alfa numéricos o caracteres especiales de forma gráfica con una gama de colores, dependiendo de los materiales de construcción de la LCD, existen diversos tipos de pantallas de cristal líquido que se utilizan como medio de comunicación entre la programación y la acción que deben realizar los microcontroladores.

Otra de las funciones importantes de las LCD es que sirven como monitor del funcionamiento eléctrico y electrónico, esto se debe al alcance y versatilidad del microcontrolador, dependerá de la programación y de las familias de los PIC

Es importante informar que las LCD permiten la comunicación entre la máquina y el ser humano de una forma visual, a través de cualquier carácter ASCII, su consumo es muy bajo en relación a los display de 7 segmentos.

La declaración LCDOUT sirve para mostrar la programación en la pantalla de cristal líquido, la sintaxis es la siguiente: LCDOUT, seguido de $FE y posteriormente el comando a utilizar; a continuación, presentamos una tabla de comandos y su operación (ver Tabla 1).
La velocidad del mensaje que se obtiene al conectar un microcontrolador y la LCD dependerá del bus de datos que puede ser de 4 o de 8 bits. Cuando se utiliza de 4 bits el mensaje se demora en llegar porque el trabajo lo realiza primero enviando 4 bis más altos y lue-

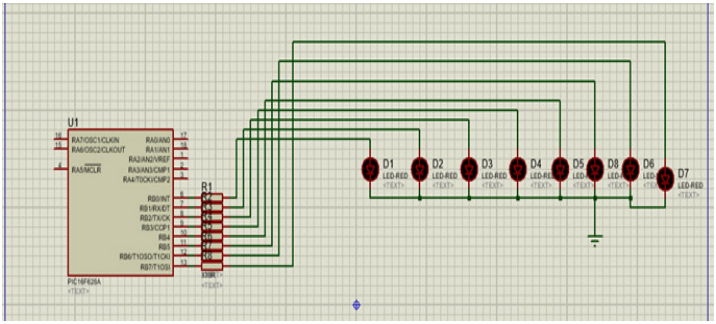


Figura 48. Diagrama de 8 led
Nota. Circuito diseñado para un juego de luces (Electrónicas y Chiluisa, 2017).

```
;Programa para encender leds de izquierda a derecha y viceversa
x       VAR     BYTE        ;creamos la variable x y tamaño de 255
LEDS    VAR     PORTB       ;todo el puerto B se llamará LEDS
TRISB = 0                   ; hacemos salidas a todo el puerto B
    LEDS = 1     ;Cargamos la variable LEDS con 1 (%00000001)

PROG:
    FOR x = 1 TO 7          ;repetir 7 veces
        LEDS = LEDS << 1    ;desplazar uno a uno a la izquierda
        PAUSE 200           ;esperar 200 milisegundos
        NEXT                ;repetir hasta que x sea = a 7

    FOR x = 1 TO 7          ;repetir 7 veces
        LEDS = LEDS >> 1    ;desplazar uno a uno a la derecha
        PAUSE 200           ;esperar 200 milisegundos
        NEXT                ;repetir hasta que x sea = a 7

GOTO PROG                   ; ir a PROG
        END
```

Figura 49. Programación de encendido de diodos led

```
CMCON=7 ; CAMBIAMOS EL PUERTO A COMO DIGITAL

PULSADOR VAR PORTA.0 ; CAMBIO DE NOMBRE AL BIT A.0 POR PULSADOR
RELE VAR PORTB.0 ; CAMBIO DE NOMBRE AL BIT B.0 POR RELE

INICIO: ; ETIQUETA PARA SALTO

        IF PULSADOR=0 THEN HIGH RELE : GOTO INICIO ; SI EL PULSADOR ES
        ; PRESIONADO, PRENDA EL RELE, Y SALTE A INICIO
        LOW RELE ; SI DEJA DE SER PULSADOR ENTONCES APAGUE EL RELE

GOTO INICIO ; SALTE A INICIO
```

Figura 50. Encendido de un diodo led como un pulsador
Nota. En esta programación se demuestra la importancia de una de las condicionantes.

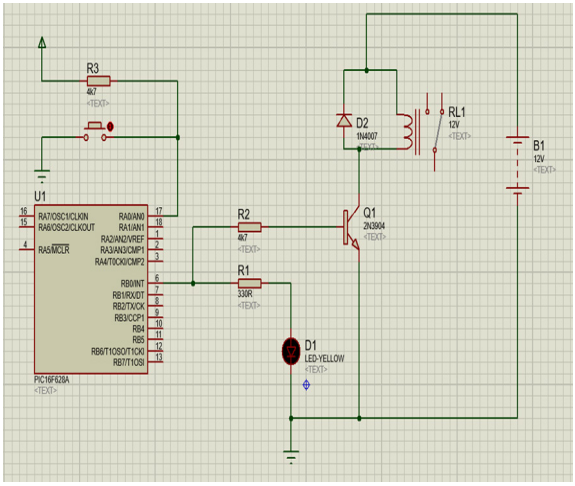
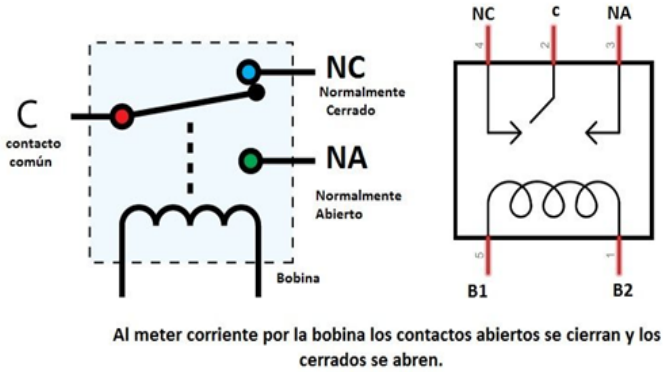


Figura 51. Esquema de una interface
Nota. El transistor trabaja como interruptor con un relay y éste puede tener como carga sistemas de grandes potencias.



Figuras 52. Esquema interno de un relay
Nota. Se puede apreciar en el bobinado y los contactos que tiene un relay (Ingeniería Mecafenix, 2019).

Figura 53. Relay
Nota. Figura física de un relay de 5 v cd. (Hetpro, 2021).

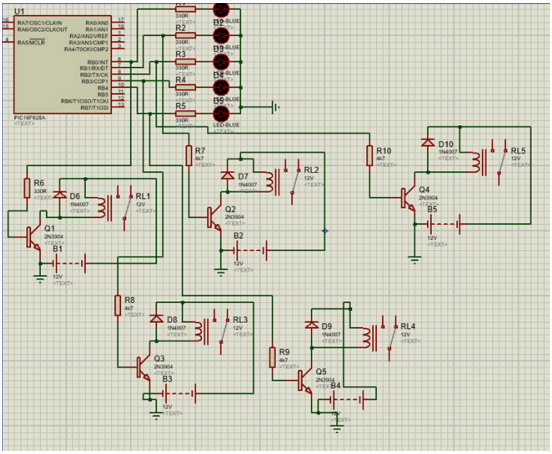


Figura 54. Tablero de control
Nota. Circuito utilizado en las empresas (Electrónicas y Chiluisa, 2017).

```
11 inicio:
12 HIGH  portb.0
13 PAUSE 100
14 LOW   portb.0
15 PAUSE 100
16 HIGH  portb.1
17 PAUSE 100
18 LOW   portb.1
19 PAUSE 100
20 HIGH  portb.2
21 PAUSE 100
22 LOW   portb.2
23 PAUSE 100
24 HIGH  portb.3
25 PAUSE 100
26 LOW   portb.3
27 PAUSE 100
28 HIGH  portb.4
29 PAUSE 100
30 LOW   portb.4
31 PAUSE 100
32 GOTO  inicio:
33 END
```

Figura 55. Programación
Nota. La programación con el HIGH, LOW, PAUSE, parámetros básicos (Electrónicas y Chiluisa, 2017).

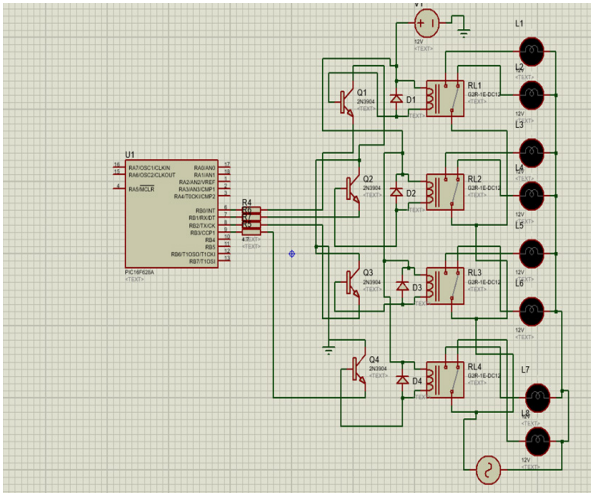
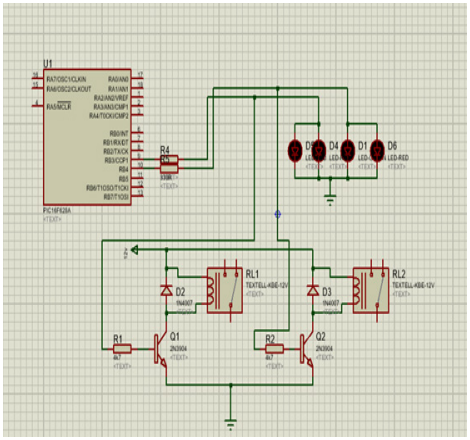


Figura 56. Voltajes continuos y alternos
Nota. Este circuito reemplaza a los PLC (Electrónicas y Chiluisa, 2017).




```
jorge:
HIGH portb.3
PAUSE 100
LOW portb.3
 HIGH portb.4
PAUSE 100
LOW portb.4
 HIGH portb.3
PAUSE 100
LOW portb.3
 HIGH portb.4
PAUSE 100
LOW portb.4
GOTO jorge
END
```

Figura 57. Del microcontrolador y un sw con transistor para aplicar a grandes potencias
Nota. Este circuito está diseñado con el transistor bipolar como interruptor, las líneas de programación con las sentencias de HIGH y LOW (Electrónicas y Chiluisa, 2017).

Tabla 1. Comandos de la LCD

LÍNEA	OPERACIÓN	COMANDO
1	Limpia el mensaje de la LCD	$FE,1
2	Apaga el cursor	$FE,$0E
3	Regresa al inicio del mensaje	$FE,2
4	Titila el cursor	$FE,$0F
5	Resalta el cursor activo	$FE,$0E
6	Mueve el cursor una posición a la izquierda	$FE,$10
7	Mueve una posición a la derecha	$FE,$14
8	Mueve el cursor a la primera línea	$FE,$80
9	Mueve el cursor al inicio de la segunda línea	$FE,$C0
10	Mueve el cursor al inicio de la tercera línea	$FE,$94
11	Mueve el cursor al inicio de la cuarta línea	$FE,$D4

Nota. En la tabla se puede apreciar el comando y la operación a realizar en la LCD (Microcontroladores Microchip, 2019).

go los 4 bits más bajos, pero si se utiliza 8 bits se envía al mismo tiempo todo (ver Figura 58).

Programación en la LCD

Antes de realizar la programación se debe verificar qué tipo de LCD se utilizará en el proyecto, esto implica que en algunos LCD no requieren poner al inicio PAUSE, tan solo requieren tener unos milisegundos para estar listos (ver Figuras 59, 60 y 61); para la sintaxis se utiliza la declaración DEFINE LCDOUT seguido de un comando $FE..., a continuación, algunas líneas de programación:

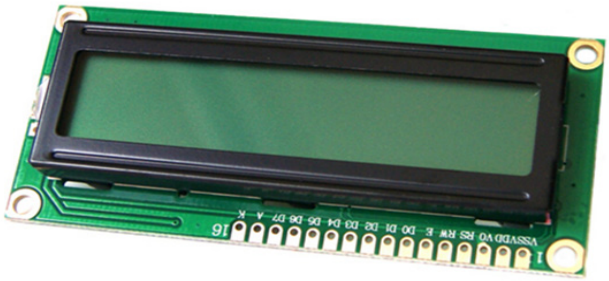


Figura 58. Pantalla de cristal líquido
Nota. Forma física de una lcd (Aprendiendo Arduino, 2019).

```
LCDUT $FE,1                    ; limpia la pantalla y ubica al cursor
                                 en el inicio.
LCDOUT $FE,86, "Marco"         ; escribe en la primera línea salta el
                                 cursor a la 7.ª casilla.
LCDOUT $FE,$C4, "Programar"    : salta a la casilla 5 de la 2.ª línea  y
                                 escribe Programar.

PAUSE                          ; retardo para esperar que funcione la LCD
LCDOUT $FE,1, "micro"          ; limpia la pantalla y escribe la pala
                                 bra  micro
LCDOUT $FE,$C0, "Programar"    ; salta al comienzo de la segunda lí-
                                 nea y escribe Programar
END                            ; fin de instrucciones
```

El optocupler conocido también como optoaislador, que trabaja como un interruptor activado mediante la luz infrarroja, es un elemento electrónico importante para este tipo de proyectos que necesitan verificar la velocidad, esto se logra gracias al infrarrojo incorporado (ver Figura 62).

El PIC y el DS1307

Una de las aplicaciones importantes de los microcontroladores es la realización de un reloj digital. Para realizar este tipo de proyectos se utiliza el integrado DS1307 que es el elemento principal para el funcionamiento de un reloj, este circuito integrado tiene incorporado un sensor de energía que detecta los fallos de alimentación y cambia automáticamente a la fuente de respaldo.

El acceso de datos se obtiene mediante la aplicación de una condición de inicio (start) y la presentación de un código de identificación del dispositivo, seguido de una división de registro. El integrado DS1307 cuenta con el formato BCD y utiliza un cristal externo de 32.768 kHz, el circuito oscilador no necesita de resistencias o condensadores para su funcionamiento (ver Figuras 63 y 64).

Para su programación se utilizará varias sentencias de control, así como también la lógica binaria de 0, 1 y los caracteres necesarios para la LCD. Como elementos externos, en este proyecto se utiliza el transistor bipolar SW como indicador de energía y un potenciómetro para calibrar las variaciones del reloj, en el diagrama se puede observar que un cristal externo se conecta sin ningún tipo de resistencias ni condensadores, esto hace que tenga una frecuencia regularizada. Así tenemos las siguientes líneas de programación. (ver Figura 65)

Otro tipo de proyecto de reloj es utilizando solo el microcontrolador, como elemento principal y como elementos externos tendremos un potenciómetro y una LCD, optimizando recursos de dispositivos electrónicos (ver Figura 66).

Las líneas de programación cuando solo se utiliza un microcontrolador se simplifican a la siguiente forma (ver Figura 67).

Proyectos con display

Para realizar este tipo de proyectos empezaremos dando a conocer qué son los display. Este dispositivo electrónico tiene la función de emitir una luz, existen dos tipos de ánodo común y cátodo común que, de acuerdo al diseño, pueden ser utilizados sin alterar su

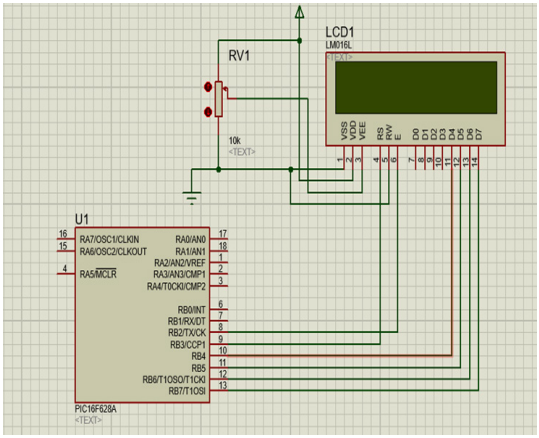


Figura 59. LCD y PIC

Nota. Circuito esquemático de un PIC y su controlador de luminosidad (Electrónicas y Chiluisa, 2017).

```
DEFINE LCD_DREG PORTB ; Definición para utilizar 4 bits del puerto B para
                      ; transmición de datos
DEFINE LCD_DBIT 4     ; desde el BIT B.4 hasta el B.7
DEFINE LCD_RSREG PORTB ; Definición para utilizar el registro de control/datos
                      ; en el puerto B
DEFINE LCD_RSBIT 3    ; en el BIT B.3
DEFINE LCD_EREG PORTB ; Definición para utilizar el enable en el puerto B
DEFINE LCD_EBIT 2     ; en el BIT B.2
R VAR BYTE ; X de tamaño de 256
DATOS VAR BYTE ; DATOS de tamaño de 256
LCDOUT $FE,$7 ; Configura al LCD para que realice el desplazamiento izquierdo
LCDOUT $FE,1 ; Limpia el visor del LCD


INICIO: ; Etiqueta para salto

        LCDOUT $FE,$90 ; Ubica el cursor en la celda 17
        FOR R=0 TO 24 ; Lazo FOR desde R=0 hasta 23
                LOOKUP R,["ELECTRONICA-PRACTICA-"],DATOS
;Toma caracter por caracter de la cadena y lo guarda en la variable DATOS
                LCDOUT , DATOS ; Saque en el LCD el contenido de DATOS
                PAUSE 200 ; Pausa de 200 milisegundos
        NEXT ; Continue con el Lazo FOR hasta que R sea igual a 24

GOTO INICIO ; Salte a la etiqueta INICIO
```

Figura 60. Programación de LCD

Nota. Ejercicio realizado para la escritura de la palaba electrónica en la LCD.

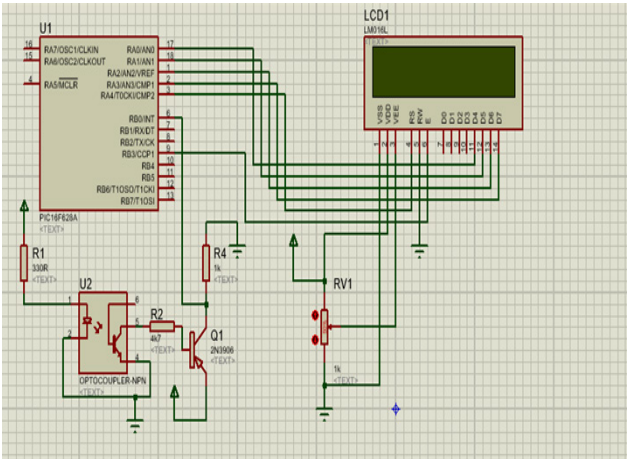


Figura 61. Medidor de velocidad con LCD
Nota. Aplicación del PIC con LCD y un optocupler (Prácticas Electrónicas, 2017).


```
cmcon=7
revo VAR WORD

jorge:
COUNT portb.0,1000,revo
revo=revo*60
LCDOUT$fe,1,"motor girando a:"
LCDOUT$fe,$c3, DEC revo
LCDOUT$fe,$c9," RPM "
GOTO jorge
END
```

Figura 62. Forma física de un optoacoplador
Nota. Dispositivos electrónicos que trabajan con infrarrojo (Deltakit, 2019).

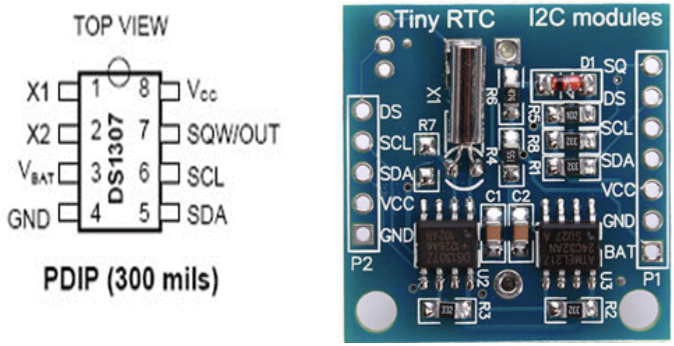


Figura 63. Estructura del DS1307
Nota. Distribución de los terminales y aplicación con SMD (EPA, 2018).

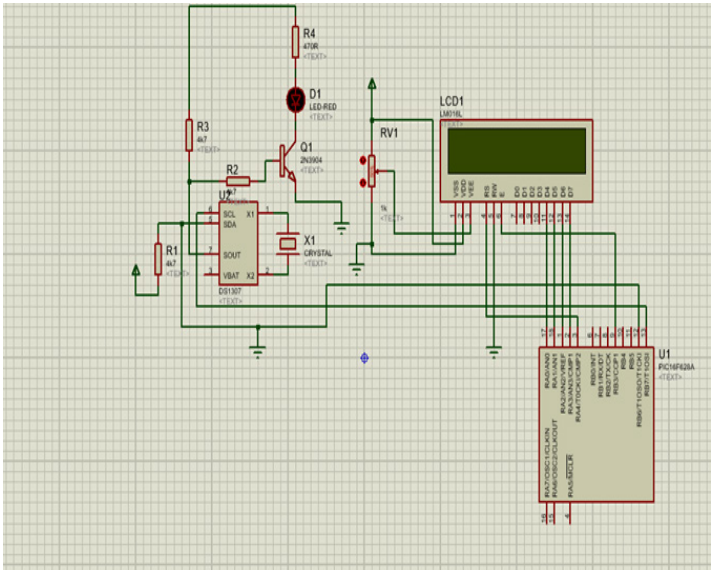


Figura 64. Diagrama esquemático
Nota. Circuito completo con un PIC (Electrónicas y Chiluisa, 2017).

```
DEFINE i2c_sclout 1

cpin VAR portb.7
dpin VAR portb.6

segu VAR BYTE
minu VAR BYTE
hora VAR BYTE
dias VAR BYTE
mes VAR BYTE
anio VAR BYTE

actualizado VAR BIT

EEPROM 0,[0]

READ 0,actualizado



inicio:
I2CREAD dpin,cpin,%00010000,0,[segu]
 I2CREAD dpin,cpin,%00010000,1,[minu]
 I2CREAD dpin,cpin,%00010000,2,[hora]
 I2CREAD dpin,cpin,%00010000,3,[dias]

 I2CREAD dpin,cpin,%00010000,5,[mes]
 I2CREAD dpin,cpin,%00010000,6,[anio]

 LCDOUT$fe,1,HEX2 hora,":",HEX2 minu,":",HEX2 segu

 LCDOUT$fe,$c0


 IF dias=$1 THEN LCDOUT"Dom."
 IF dias=$2 THEN LCDOUT"Lun."
 IF dias=$3 THEN LCDOUT"Mar."
 IF dias=$4 THEN LCDOUT"Mie."
 IF dias=$5 THEN LCDOUT"Jue."
 IF dias=$6 THEN LCDOUT"Vie."
 IF dias=$7 THEN LCDOUT"Sab."


 LCDOUT$fe,$cB,"/20",HEX2 anio

```

```
55
56  LCDOUT$fe,$c8
57  IF mes=$1 THEN LCDOUT"ene"
58  IF mes=$2 THEN LCDOUT"feb"
59  IF mes=$3 THEN LCDOUT"mar"
60  IF mes=$4 THEN LCDOUT"abr"
61  IF mes=$5 THEN LCDOUT"may"
62  IF mes=$6 THEN LCDOUT"jun"
63  IF mes=$7 THEN LCDOUT"jul"
64  IF mes=$8 THEN LCDOUT"ago"
65  IF mes=$9 THEN LCDOUT"sep"
66  IF mes=$10 THEN LCDOUT"ovt"
67  IF mes=$11 THEN LCDOUT"nov"
68  IF mes=$12 THEN LCDOUT"dic"
69  PAUSE 500
70
71  GOTO inicio
72  '****************subrutina grabar*******************
73 grabaRTC:
74
75 I2CWRITE dpin,cpin,%00010000,0,[$00]
76 PAUSE 10
77  I2CWRITE dpin,cpin,%00010000,1,[$30]
78 PAUSE 10
79  I2CWRITE dpin,cpin,%00010000,2,[$13]
80 PAUSE 10
81  I2CWRITE dpin,cpin,%00010000,3,[$2]
82 PAUSE 10
83  I2CWRITE dpin,cpin,%00010000,4,[$27]
84 PAUSE 10
85  I2CWRITE dpin,cpin,%00010000,5,[$9]
86 PAUSE 10
87  I2CWRITE dpin,cpin,%00010000,6,[$04]
88 PAUSE 10
89 I2CWRITE dpin,cpin,%00010000,7,[$10]
90 PAUSE 10
91
92 WRITE 0,1
93
94 GOTO inicio
95 END
96
```

Figura 65. Líneas de programación en el PIC 16F628A
Nota. Programación en el IDE de MicroCode (Electrónicas y Chiluisa, 2017).

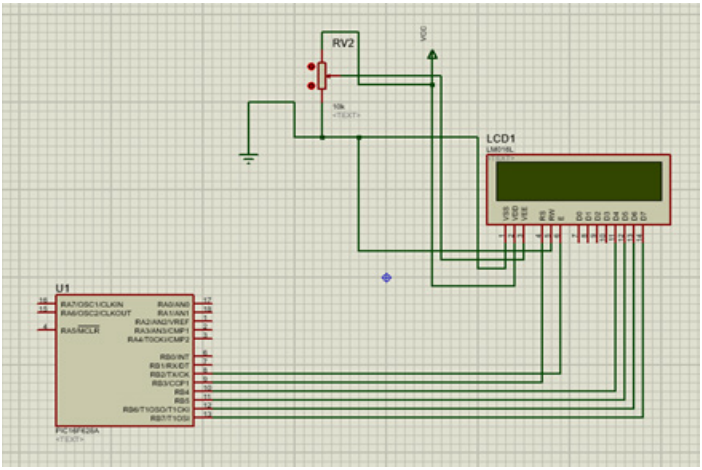


Figura 66. Diagrama esquemático
Nota. Diagrama de un reloj con el PIC (Electrónicas y Chiluisa, 2017).

```
'*****************************************************************
DEFINE LCD_DREG PORTB  ;Definición para utilizar 4 bits del puerto b para
                       ; transmición de datos
DEFINE LCD_DBIT 4      ; desde el bit B.4 hasta el bit B.7
DEFINE LCD_RSREG PORTB ; definicón para utilizar el registro de control/datos
                       ; en el puerto B
DEFINE LCD_RSBIT 3     ; En el bit B.3
DEFINE LCD_EREG PORTB  ; definición para utilizar el enable en el puerto B
DEFINE LCD_EBIT 2      ; en el bit B.2

H VAR BYTE  ; cambio de variable a H de tamaño 256
            ; variable para las HORAS
M VAR BYTE  ; cambio de variable a M de tamaño 256
            ; variable para los MINUTOS
S VAR BYTE  ; cambio de variable a S de tamaño 256
            ; variable para los SEGUNDOS
  LCDOUT $FE,1,"LA HORA ES: GSM"   ; limpia el visor del lcd y se
 ; posiciona al principio de la primera fila, y escribe
 ;RELOJ DIGITAL


INICIO:   ; etiqueta para salto

FOR H=0 TO 23  ;lazo for para las horas
FOR M=0 TO 59  ;lazo for para los minutos
FOR S=0 TO 59  ;lazo for para los segundos
LCDOUT $FE,$c4,DEC2 H,":",DEC2 M,":",DEC2 S
        ; saca por la segunda linea dos decimales de llas horas,
        ; 2 puntos, 2 decimales para los minutos,
```

```
39          ; 2 puntos y 2 decimales para los segundos
40          ; correspondientes a las variables de H,M,S respectivamente
41 PAUSE 395    ;pausa de 395 milisegundos
42 LCDOUT $FE,$c4,DEC2 H," ",DEC2 M," ",DEC2 S
43          ; saca la segunda línea dos decimales de las horas,
44          ; espacio en blanco, 2 decimales para los minutos
45          ; espacio en blanco y 2 decimales para los segundos
46          ; correspondientes a las variables de H,M,S respectivamente
47 PAUSE 395    ; pausa de 395 milisegundos
48 NEXT         ; continúe con el lazo hasta que S=59
41 PAUSE 395    ;pausa de 395 milisegundos
42 LCDOUT $FE,$c4,DEC2 H," ",DEC2 M," ",DEC2 S
43          ; saca la segunda línea dos decimales de las horas,
44          ; espacio en blanco, 2 decimales para los minutos
45          ; espacio en blanco y 2 decimales para los segundos
46          ; correspondientes a las variables de H,M,S respectivamente
47 PAUSE 395    ; pausa de 395 milisegundos
48 NEXT         ; continúe con el lazo hasta que S=59
49 NEXT         ; continúe con el lazo hasta que M=59
50 NEXT         ; continúe con el lazo hasta que H=23
51
52 GOTO INICIO  ; salte a la etiqueta de inicio
53
54
55
```

Figura 67. Programación de un reloj
Nota. Programación realizada en el IDE MicroCode (Electrónicas y Chiluisa, 2017).

resultado, pero en lo que se refiere al funcionamiento interno del circuito varía porque el uno trabaja con tensión positiva y el otro con tensión negativa. Existe una gran variedad de circuitos que son muy utilizados por los usuarios en el diario vivir (ver Figura 68).

El propósito de este circuito es que pueda contar desde cero hasta 99, para esto utilizaremos las siguientes líneas de programación (ver Figuras 69 y 70).

Para la programación se utiliza directivas de PIC (ver Figuras 71, 72 y 73)

Micro con teclado y display

En los proyectos con teclados, es necesario conocer cómo funcionan los teclados y los diferentes tipos que existen.

Un teclado denominado matricial es un conjunto de pulsadores que forman una matriz, y éstos, a su vez, realizan una determinada acción. Los teclados están conformados por filas y columnas internamente (ver Figura 74).

La forma en que trabajan los teclados matriciales es por filas, del 1 hasta el 4, que serán en punto común, sus acciones las realizarán las columnas, desde 5 hasta el 8 (ver Figuras 75 y 76).

Activación de teclado e interface

(ver Figura 77)

Motores

La utilización de los motores en la industria se vuelve cada vez más necesaria para el funcionamiento de muchos dispositivos electrónicos digitales y analógicos. Existe gran variedad de tipos de motores de corriente continua y corriente alterna, así como también servomotores, motores paso a paso, motores DC.

Los motores paso a paso son utilizados en dispositivos que necesitan, por precisión de movimiento, un determinado grado y velocidad, en las industrias es muy común observar en las fresadoras, tornos, en la parte informática este tipo de motores se encuentran en las unidades de CD-ROM, en las impresoras de matriz, entre otras.

La principal diferencia de los motores paso a paso y los motores de corriente continua o corriente alterna consiste en el grado de control que se tiene en la velocidad de rotación.

En los motores paso a paso existen dos grandes clasificaciones: motores bipolares y unipolares (ver Figura 78).

Para manejar este tipo de motores con los microcontroladores se necesita de un circuito electrónico conocido también como interfaz, porque la corriente con que trabaja el microcontrolador es muy débil, y la corriente de alimentación de los motores es muy grande, estos circuitos integrados por transistores de potencia ayudan a controlar el funcionamiento de los motores.

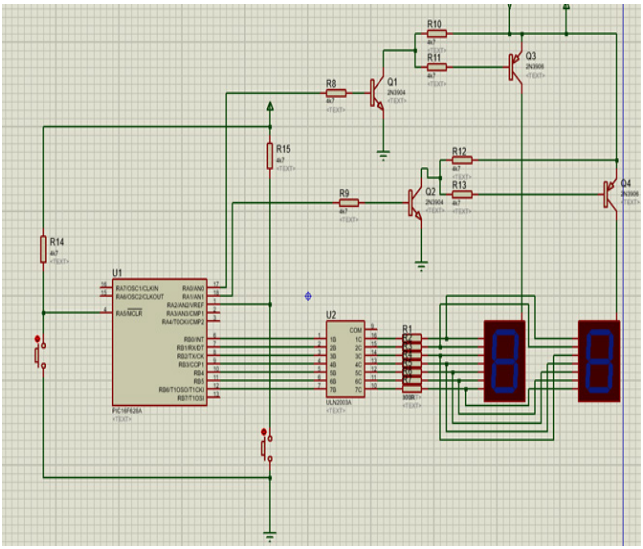


Figura 68. Contador digital
Nota. Diagrama esquemático de ánodo común (Electrónicas y Chiluisa, 2017).

```
13 PORTA=0
14 TRISB=%00000000
15 PORTB=0
16
17 VEC VAR BYTE(10)
18
19          GFEDCBA
20 VEC(0)=%0111111
21 VEC(1)=%0000110
22 VEC(2)=%1011011
23 VEC(3)=%1001111
24 VEC(4)=%1100110
25 VEC(5)=%1101101
26 VEC(6)=%1111101
27 VEC(7)=%0000111
28 VEC(8)=%1111111
29 VEC(9)=%1101111
```

```
30
31 NUM VAR BYTE
32 UNI VAR BYTE
33 DECE VAR BYTE
34 UNIDAD VAR BYTE
35 DECENA VAR BYTE
36 I VAR BYTE
37 NUM=0
38 UNIDAD=VEC(0)
39 DECENA=VEC(0)
40
41 MAIN:
42     IF PORTA.2=1 THEN
43          REPEAT
44            GOSUB BARRIDO
45          UNTIL PORTA.2=0
46          GOSUB SUMAR
47     ENDIF
48     GOSUB BARRIDO
49 GOTO MAIN
50
51 BARRIDO:
52          HIGH PORTA.1
53          PORTB=UNIDAD
54          PAUSE 20
55          LOW PORTA.1
56          HIGH PORTA.0
57          PORTB=DECENA
58          PAUSE 20
59          LOW PORTA.0
60 RETURN
61
62 SUMAR:
63          NUM=NUM+1
64          UNI=NUM DIG 0
65          SELECT CASE UNI
66               CASE 0:
67                    UNIDAD=VEC(0)
68               CASE 1:
69                    UNIDAD=VEC(1)
70               CASE 2:
71                    UNIDAD=VEC(2)
```

```
72                CASE 3:
73                     UNIDAD=VEC(3)
74                CASE 4:
75                     UNIDAD=VEC(4)
76                CASE 5:
77                     UNIDAD=VEC(5)
78                CASE 6:
79                     UNIDAD=VEC(6)
80                CASE 7:
81                     UNIDAD=VEC(7)
82                CASE 8:
83                     UNIDAD=VEC(8)
84                CASE 9:
85                     UNIDAD=VEC(9)
86           END SELECT
87           DECE=NUM DIG 1
88            SELECT CASE DECE
89            CASE 0:
90               DECENA=VEC(0)
91            CASE 1:
92               DECENA=VEC(1)
93            CASE 2:
94               DECENA=VEC(2)
95            CASE 3:
96               DECENA=VEC(3)
97            CASE 4:
98               DECENA=VEC(4)
99            CASE 5:
100              DECENA=VEC(5)
101           CASE 6:
102              DECENA=VEC(6)
103           CASE 7:
104              DECENA=VEC(7)
105           CASE 8:
106              DECENA=VEC(8)
107           CASE 9:
108              DECENA=VEC(9)
109     END SELECT
110 RETURN
```

Figura 69. Líneas de programación
Nota. Programación de un contador en el IDE MicroCode (Electrónicas y Chiluisa, 2017).

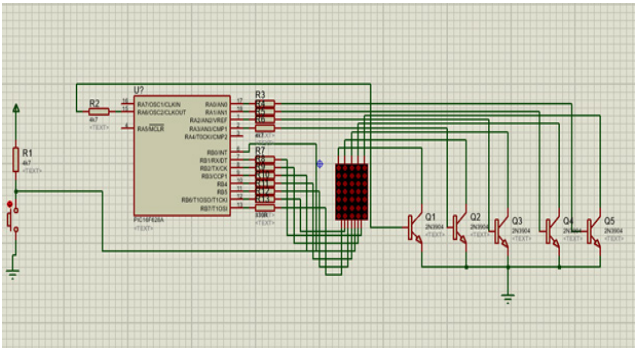


Figura 70. Animación con una matriz
Nota. Cumple la función de realizar una figura en movimiento (Electrónicas y Chiluisa, 2017).

```
[DeviceName]
Value=P16F628A
[DeviceClock]
Value=8
[MainUnit]
Value=Munieco_Animado.pbas
[DeviceFlags]
COUNT=4
Value0=_WDT_OFF = $3FFB
Value1=_LVP_OFF = $3F7F
Value2=_MCLRE_OFF = $3FDF
Value3=_INTRC_OSC_NOCLKOUT = $
[ProjectFiles]
COUNT=0
[OtherFiles]
COUNT=1
Value0=C:\Munieco_Animado.pbas
[TabIndex]
Value=0
[SearchPath]
COUNT=5
Value0=C:\Defs\
```

Figura 71. Líneas de programación
Nota. Programación de figura en el IDE MicroCode (Electrónicas y Chiluisa, 2017).

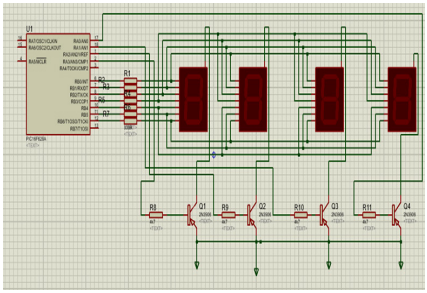


Figura 72. Ejercicio con display
Nota. Diagrama esquemático de ánodo común (Electrónicas y Chiluisa, 2017).

```
cmcon=7
Trisb=0
Trisa=0

texto:
porta=7
portb=70
PAUSE 2000
porta=11
portb=15
PAUSE 2000
porta=13
portb=79
PAUSE 2000
porta=14
portb=63
PAUSE 2000
porta=7
portb=36
PAUSE 2000
porta=11
portb=64
PAUSE 2000
porta=13
portb=64
PAUSE 2000
porta=14
portb=56
PAUSE 3000
GOTO texto
END
```

```
GK VAR BYTE
led0 VAR portb.0
G1:
FOR GK=1 TO 15
HIGH led0
PAUSE 500
LOW led0
PAUSE 500
NEXT
PAUSE 3000
GOTO G1
END
```

Figura 73. Líneas de programación para que funcionen los display formando letras
Nota. Este tipo de programación sirve para formar textos completos en letras digitales (Electrónicas y Chiluisa, 2017).

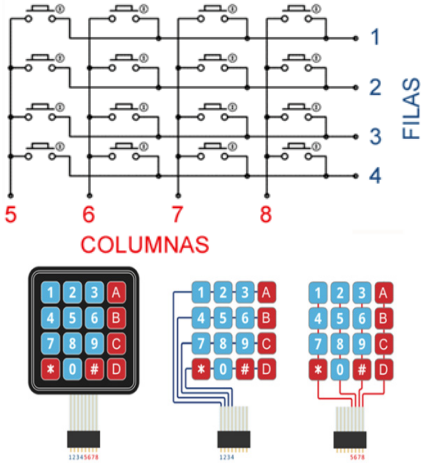


Figura 74. Esquema de un matriz
Nota. Configuración interna y externa de un teclado matricial (Luis Llamas, 2020).

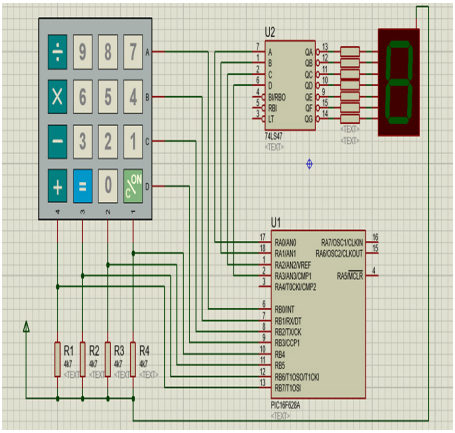


Figura 75. Diagrama esquemático de una matriz
Nota. Aplicación de una matriz con su visualización en el display (Electrónicas y Chiluisa, 2017).

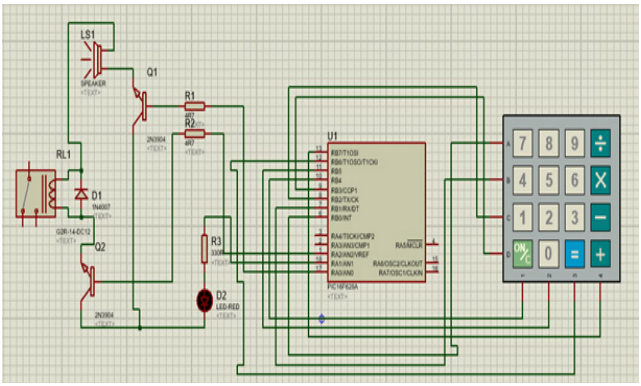
```
 CMCON=7
TrisA=%0000 ; Hacemos salida a los bits
            ; A.3, A.2, A.1, A.0
x VAR BYTE  ; x de tamaño de 256
R VAR BYTE  ; R de tamaño de 256
x=0 ; inicializamos la variable x con cero
GOSUB VISUALIZAR ; salte a visualizar el número cero
FILA A VAR PORTB.0 ; cambio de nombre al puerto B.0 por FILA A
FILA B VAR PORTB.1 ; cambio de nombre al puerto B.1 por FILA B
FILA C VAR PORTB.2 ; cambio de nombre al puerto B.2 por FILA C
FILA D VAR PORTB.3 ; cambio de nombre al puerto B.3 por FILA D
COL UNO VAR PORTB.4 ; cambio de nombre al puerto B.0 por COL_UNO
COL DOS VAR PORTB.5 ; cambio de nombre al puerto B.0 por COL_DOS
COL TRES VAR PORTB.6 ; cambio de nombre al puerto B.0 por COL_TRES
COL CUATRO VAR PORTB.7 ; cambio de nombre al puerto B.0 por COL CUATRO

INICIO: ; Etiqueta para salto
        LOW FILA A ; censamos la FILA A
            IF COL_UNO=0 THEN  x=7 : GOSUB VISUALIZAR ; Si la
            ;columna uno es presionada cargue a x con 7 salte a visualizar
            ; el dato
            IF COL_DOS=0 THEN  x=8  : GOSUB VISUALIZAR
            IF COL_TRES=0 THEN x=9  : GOSUB VISUALIZAR
            IF COL CUATRO=0 THEN x=0 : GOSUB VISUALIZAR
        HIGH FILA A ; fin del censo de la FILA A
        LOW FILA B ; censamos la FILA B
            IF COL_UNO=0 THEN  x=4 : GOSUB VISUALIZAR
            IF COL_DOS=0 THEN  x=5 : GOSUB VISUALIZAR
            IF COL_TRES=0 THEN  x=6 : GOSUB VISUALIZAR
            IF COL CUATRO=0 THEN  x=0 : GOSUB VISUALIZAR
        HIGH FILA B ; fin del censo de la FILA B
        LOW FILA C ; censamos la FILA C
            IF COL_UNO=0 THEN  x=1 : GOSUB VISUALIZAR
            IF COL_DOS=0 THEN  x=2 : GOSUB VISUALIZAR
            IF COL_TRES=0 THEN  x=3 : GOSUB VISUALIZAR
            IF COL CUATRO=0 THEN  x=0 : GOSUB VISUALIZAR
        HIGH FILA C ; fin del censo de la FILA C
        LOW FILA D ; censamos la FILA D
            IF COL_UNO=0 THEN  x=0 : GOSUB VISUALIZAR
          IF COL_DOS=0 THEN  x=0 : GOSUB VISUALIZAR
          IF COL_TRES=0 THEN  x=0 : GOSUB VISUALIZAR
          IF COL CUATRO=0 THEN  x=0 : GOSUB VISUALIZAR
       HIGH FILA D ; fin del censo de la FILA D

GOTO INICIO ; salto a la etiqueta inicio
VISUALIZAR: ; etiqueta de salto llamada visualizar

              PORTA=x ; el puerto A se carga con la variable x
              IF COL_UNO=0 THEN GOTO VISUALIZAR
              IF COL_dOs=0 THEN GOTO VISUALIZAR
              IF COL_tres=0 THEN GOTO VISUALIZAR
              IF COL cuatro=0 THEN GOTO VISUALIZAR
RETURN ; retorna a donde se quedó
```

Figura 76. Programación de un PIC
Nota. Programación de filas y columna (Electrónicas y Chiluisa, 2017).



Programación:

```
11
12 NUMERO VAR BYTE
13 R VAR BYTE
14 BIP VAR PORTA.0
15 LED VAR PORTA.1
16 DOOR VAR PORTA.2
17 A VAR PORTB.0
18 B VAR PORTB.1
19 C VAR PORTB.2
20 D VAR PORTB.3
21 UNO VAR PORTB.4
22 DOS VAR PORTB.5
23 TRES VAR PORTB.6
24 CUATRO VAR PORTB.7
25 INICIANDO:
26 HIGH LED:HIGH BIP
27 PAUSE 500
28 LOW LED: LOW BIP
29 GOTO TECLAUNO
30 BARRIDO:
31 LOW A
32 IF UNO =0 THEN NUMERO=1: RETURN
33 IF DOS =0 THEN NUMERO=2: RETURN
34 IF TRES =0 THEN NUMERO=3: RETURN
35 IF CUATRO =0 THEN NUMERO=10: RETURN
36 HIGH A
37 LOW B
38 IF UNO =0 THEN NUMERO=4 :RETURN
39 IF DOS =0 THEN NUMERO=5 :RETURN
40 IF TRES =0 THEN NUMERO=6: RETURN
41 IF CUATRO =0 THEN NUMERO=11: RETURN
42 HIGH B
43 LOW C
44 IF UNO =0 THEN NUMERO=7 :RETURN
```

Figura 77. Activador con el teclado

Nota. Proyecto para activar por medio de las teclas (Electrónicas y Chiluisa, 2017)

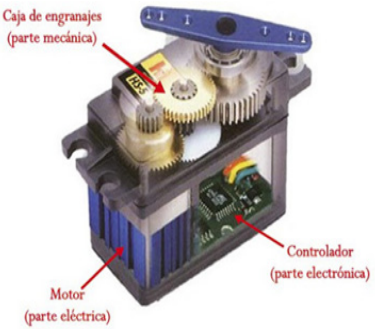


Figura 78. Los motores
Nota. Motores paso a paso y servomotor (Wikipedia, 2019)

Motores de corriente continua

Los motores de CD se utilizan para trabajos de potencia a grandes velocidades, y pueden funcionar con los microcontroladores utilizando una interface analógica o digital (ver Figura 79).

En el diagrama se observa la instalación de 4 motores en paralelo con la salida del microcontrolador (ver Figura 80), estos motores son de bajo consumo, del orden de los 500 miliamperios, funcionan sin ningún tipo de problema ya que el PI proporciona independientemente el voltaje, los pulsadores sirven para dar la orden de ejecución de un determinado motor, este tipo de circuito es utilizado en la industria con sus respectivas interfaces de potencia (ver Figuras 81 y 82).

La programación utiliza como punto de partida señales digitales, es decir, 1 y 0 para el encendido y apagado de los motores, con las condicionantes controlamos los pulsadores y una sentencia de repetición con el GOTO, esto se cumplirá mientras esté con un fluido eléctrico. Para proteger al PIC se recomienda instalar transistores de potencia u optoacopladores (ver Figuras 83, 84 y 85).

Motores paso a paso

Los motores paso a paso son muy utilizados en la actualidad (ver Figuras 86, 87, 88, 89, 90 y 91).

Detector de objetos

(ver Figuras 92 y 93)

Aplicación de sonido

Otra de las fortalezas que tienen los microcontroladores, es que permiten generar sonidos de ocho bits, el sonido va en un rango de 0 a 255 (ver Figura 94). En muchas ocasiones se puede aplicar directamente en un zumbador o parlante, el sonido es bastante bajo con el riesgo de que el microcontrolador llegue a un estado de saturación, por eso es muy recomendable instalar como mínimo un transistor para que trabaje como una interface y regule el voltaje y la corriente proporcionada por el microcontrolador con los voltajes de la carga aplicada.

Tienen una estructura de bobina y un núcleo, se caracteriza por transformar las corrientes eléctricas en vibraciones sonoras, de ahí el nombre de transductores. Existe una amplia gama de parlantes, zumbadores.

Para la programación se utilizan la sentencia SOUND seguido del nombre del puerto a dónde va conectado, y posteriormente los números que indicarán el tono que se generará.

SOUND PORTB.0,[120,8,100,20] que significa que saque un tono de 120 por el puerto B, mantenga el tono por 8 milisegundos, y luego un tono de 100 durante 20 milisegundos.

Proyecto con sonido

La utilización de varias sentencias tiene como objetivo construir una alarma que genere un sonido, así como también se accionen los diodos LED; en el ejercicio, a los sensores se ha colocado pulsadores (ver Figuras 95 y 96).

Se maneja el estado 1 lógico y 0 lógico, así como también las condicionantes, un lazo repetitivo. De esta forma se puede construir una

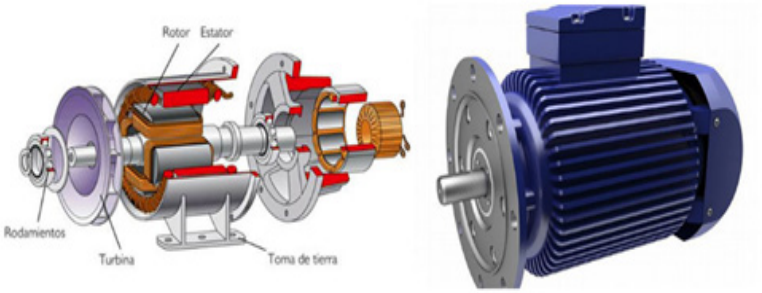


Figura 79. Motor de corriente continua
Nota. En la figura se puede aprecia cómo está constituido (Taller Electrónicos Brim, 2020).

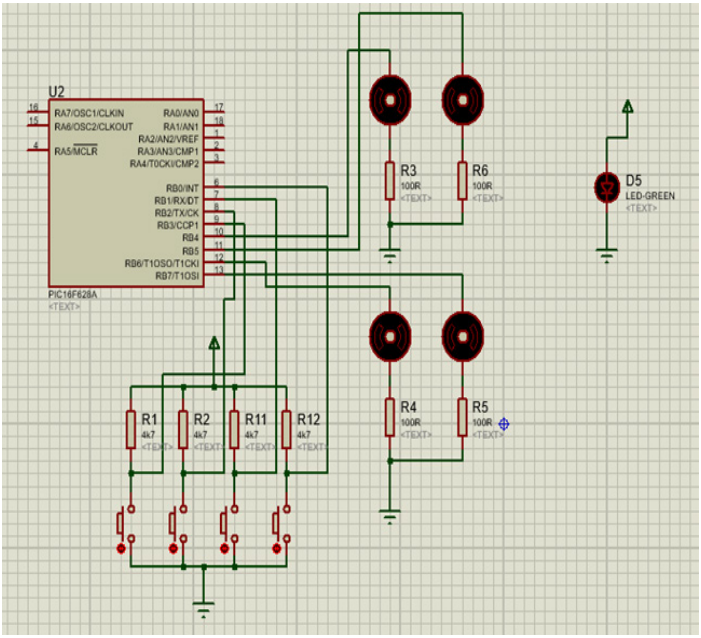


Figura 80. Motor de corriente continua
Nota. Diagrama de 4 motores de CD con pulsadores (Electrónicas y Chiluisa, 2017).

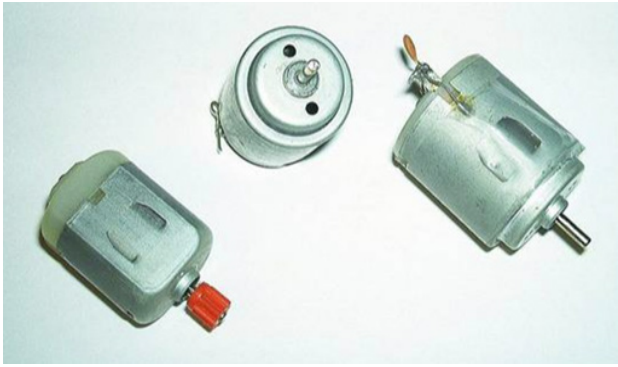
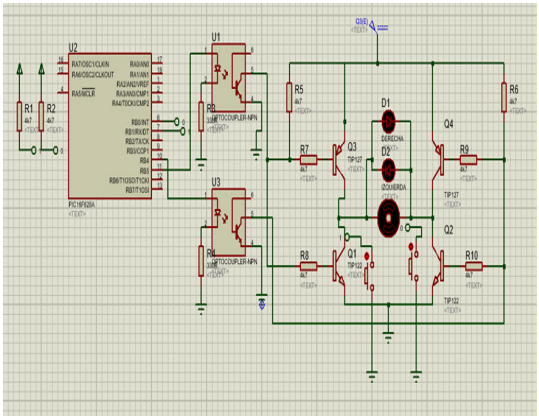


Figura 81. Motores de CD
Nota. Estos motores son de bajo consumo y trabajan con corriente directa CD (Taller Electrónicos Brim, 2020).

```
TRISB = %00001111
PORTB = 0

MAIN:
IF PORTB.0 = 0 THEN
    PAUSE 250
    HIGH PORTB.5
    LOW PORTB.4
    PAUSE 1000
    LOW PORTB.5 : LOW PORTB.4
ENDIF
IF PORTB.1 = 0 THEN
    PAUSE 250
    LOW PORTB.5
    HIGH PORTB.4
    PAUSE 1000
    LOW PORTB.5 : LOW PORTB.4
ENDIF
IF PORTB.2 = 0 THEN
    PAUSE 250
    HIGH PORTB.7
    LOW PORTB.6
    PAUSE 1000
    LOW PORTB.7 : LOW PORTB.6
ENDIF
IF PORTB.3 = 0 THEN
    PAUSE 250
    LOW PORTB.7
    HIGH PORTB.6
    PAUSE 1000
    LOW PORTB.6 : LOW PORTB.7
ENDIF
GOTO MAIN
```

Figura 82. Líneas de programación
Nota. Programación de un contador en el IDE MicroCode (Electrónicas y Chiluisa, 2017).



El circuito conformado

```
cmcon=7
Trisb=0
Trisa=0

texto:
porta=7
portb=70
PAUSE 2000
porta=11
portb=15
PAUSE 2000
porta=13
portb=79
PAUSE 2000
porta=14
portb=63
PAUSE 2000
porta=7
portb=36
PAUSE 2000
porta=11
portb=64
PAUSE 2000
porta=13
portb=64
PAUSE 2000
porta=14
portb=56
PAUSE 3000
GOTO texto
END
```

Figura 83. Motores con circuito puente H
Nota. Programación de un contador en el IDE MicroCode (Electrónicas y Chiluisa, 2017).

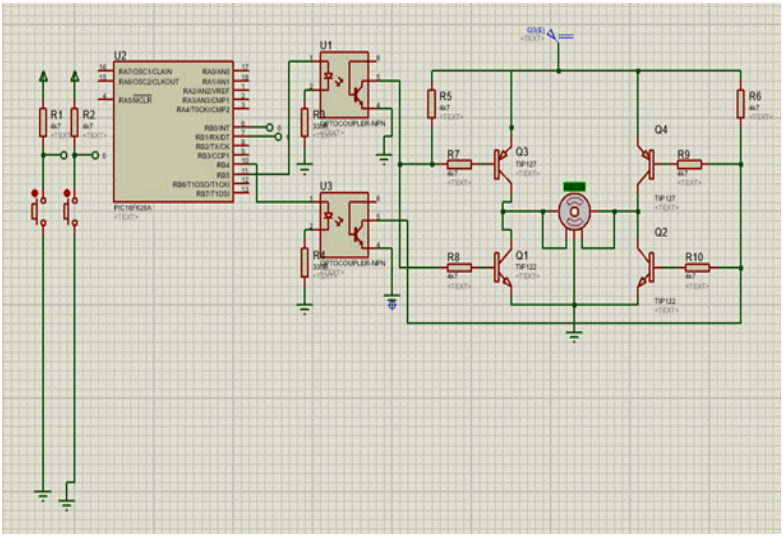


Figura 84. Aplicaciones de optoacoplador
Nota. Circuito controlado con los pulsadores para cambiar de giro (Electrónicas y Chiluisa, 2017).

```
12
13 TRISB = %00001111
14 PORTB = 0
15
16 MAIN:
17 IF PORTB.0 = 0 THEN
18     PAUSE 250
19     HIGH PORTB.5
20     LOW PORTB.4
21     PAUSE 1000
22     LOW PORTB.5 : LOW PORTB.4
23 ENDIF
24 IF PORTB.1 = 0 THEN
25     PAUSE 250
26     LOW PORTB.5
27     HIGH PORTB.4
28     PAUSE 1000
29     LOW PORTB.5 : LOW PORTB.4
30 ENDIF
31 GOTO MAIN
```

Figura 85. Programación del PIC
Nota. Motor de doble giro controlado por los pulsadores (Electrónicas y Chiluisa, 2017)

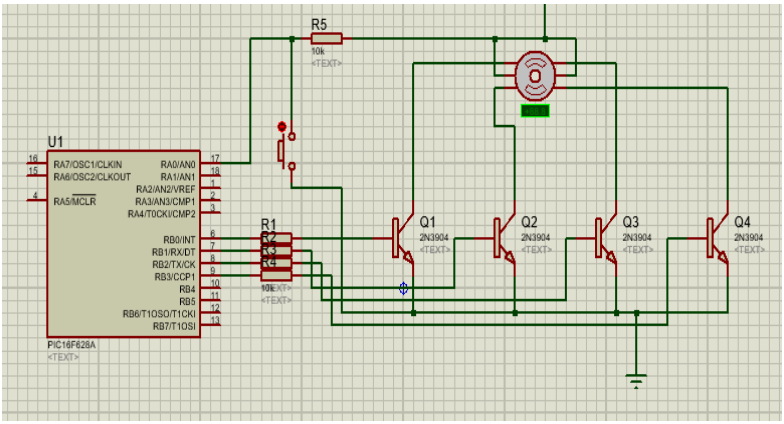


Figura 86. Variador de velocidad con el pulsador

```
[DeviceName]
Value=P16F628A
[DeviceClock]
Value=8
[MainUnit]
Value=PROYECTO1.pbas
[DeviceFlags]
COUNT=4
Value0=_WDT_OFF = $3FFB
Value1=_LVP_OFF = $3F7F
Value2=_MCLRE_OFF = $3FDF
Value3=_INTRC_OSC_CLKOUT = $3FFD
[ProjectFiles]
COUNT=0
[OtherFiles]
COUNT=0
[TabIndex]
Value=0
[SearchPath]
COUNT=3
Value0=C:\Defs\
Value1=C:\p16\
Value2=C:\5 PROYECTOS MOTOR\PROYECTO1\
[EEPROMinfo]
isused=0
```

Figura 87. Líneas de programación

Nota. Circuito que funciona con el puente H para invertir el giro (Electrónicas y Chiluisa, 2017).

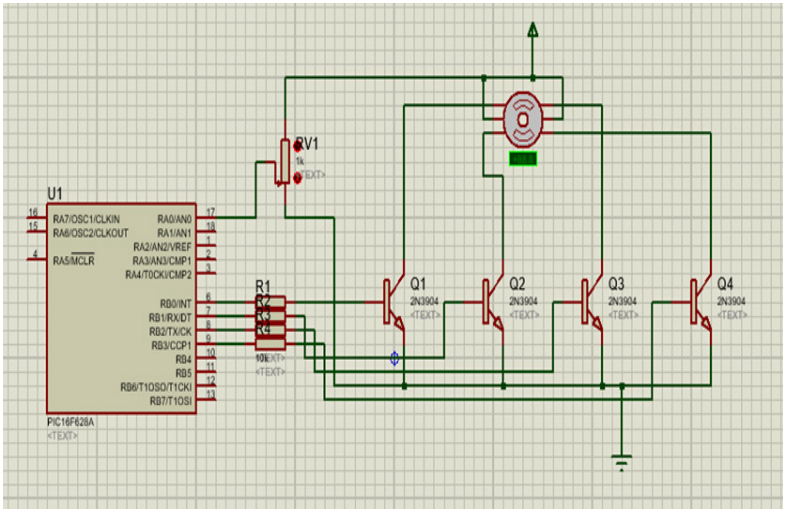


Figura 88. Inversor de giro con potenciómetro
Nota. Se controla el giro con el potenciómetro (Electrónicas y Chiluisa, 2017).

```
[DeviceName]
Value=P16F628A
[DeviceClock]
Value=8
[MainUnit]
Value=PROYECTO2.pbas
[DeviceFlags]
COUNT=4
Value0=_WDT_OFF = $3FFB
Value1=_LVP_OFF = $3F7F
Value2=_MCLRE_OFF = $3FDF
Value3=_INTRC_OSC_CLKOUT = $3FFD
[ProjectFiles]
COUNT=0
[OtherFiles]
COUNT=0
[TabIndex]
Value=0
[SearchPath]
COUNT=3
Value0=C:\Defs\
Value1=C:\p16\
Value2=C:\5 PROYECTOS MOTOR\PROYECTO2\
[EEPROMinfo]
isused=0
```

Figura 89. Programación con directivas de C
Nota. MicroCode Studio reconoce algunas sentencias (Electrónicas y Chiluisa, 2017).

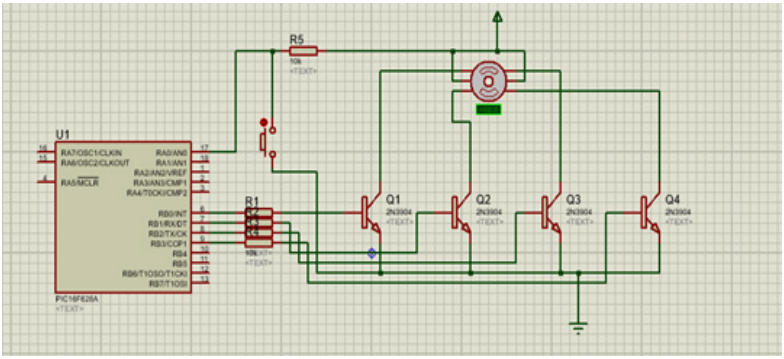


Figura 90. Circuito que controla la velocidad con el pulsador a 10 segundos
Nota. Este circuito está programado para que funcione 10 segundos con una dirección y luego cambie de dirección.

```
[DeviceName]
Value=P16F628A
[DeviceClock]
Value=8
[MainUnit]
Value=PROYECTO3.pbas
[DeviceFlags]
COUNT=4
Value0=_WDT_OFF = $3FFB
Value1=_LVP_OFF = $3F7F
Value2=_MCLRE_OFF = $3FDF
Value3=_INTRC_OSC_CLKOUT = $3FFD
[ProjectFiles]
COUNT=0
[OtherFiles]
COUNT=2
Value0=C:\PROYECTO3.pbas
Value1=C:\PROYECTOS\5 proyectos infrarojo\Contador_lcd.pbas
[TabIndex]
Value=0
[SearchPath]
COUNT=4
Value0=C:\Defs\
Value1=C:\p16\
Value2=C:\5 PROYECTOS MOTOR\PROYECTO3\
Value3=C:\5 PROYECTOS MOTOR\PROYECTO3\
[EEPROMinfo]
isused=0
```

Figura 91. Líneas de programación
Nota. Este circuito tiene incorporado el puente H para invertir el giro (Electrónicas y Chiluisa, 2017).

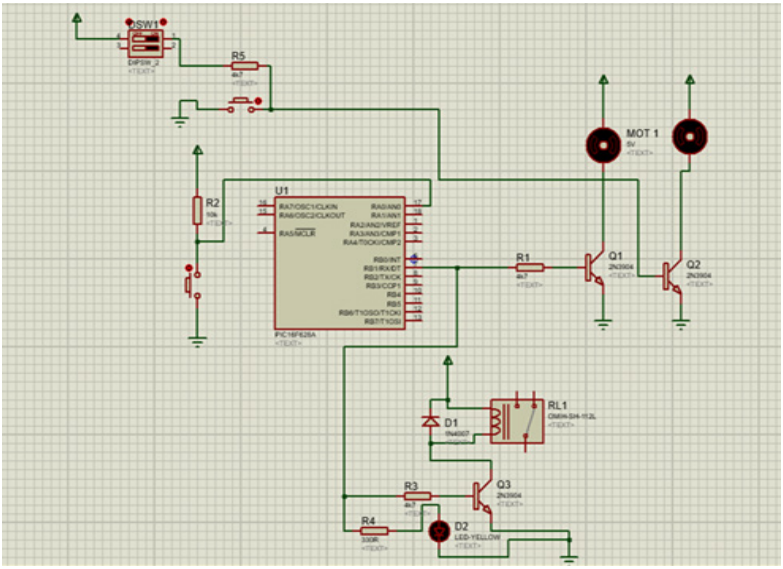


Figura 92. Circuito diseñado con 2 motores de corriente continua
Nota. Este es un prototipo para un robot (Electrónicas y Chiluisa, 2017).

```
 2
 3 [DeviceName]
 4 Value=P16F628A
 5 [DeviceClock]
 6 Value=8
 7 [MainUnit]
 8 Value=MOTOR.pbas
 9 [DeviceFlags]
10 COUNT=4
11 Value0=_WDT_OFF = $3FFB
12 Value1=_LVP_OFF = $3F7F
13 Value2=_MCLRE_OFF = $3FDF
14 Value3=_INTRC_OSC_NOCLKOUT = $3FFC
15 [ProjectFiles]
16 COUNT=0
17 [OtherFiles]
18 COUNT=0
19 [TabIndex]
20 Value=0
21 [SearchPath]
22 COUNT=3
23 Value0=C:\Defs\
24 Value1=C:\p16\
25 Value2=C:\PROYECTOS\5 proyectos infrarojo\
26 [EEPROMinfo]
27 isused=0
```

Figura 93. Programación
Nota. Este es un prototipo para un robot (Electrónicas y Chiluisa, 2017).

Figura 94. Estructura del parlante
Nota. El parlante es el dispositivo de salida para las programaciones de SOUND (Taller Electrónicos Brim, 2020).

alarma básica con la programación del microcontrolador. La selección de un microcontrolador dependerá del tipo de proyecto que se va a construir para garantizar el software o líneas de programación.

Alarma de aviso

(ver Figuras 97 y 98)

Alarma con microcontrolador 16F877

Para la utilización de un determinado PIC, se debe considerar la complejidad del proyecto y los servicios que estarán programados (ver Figuras 99 y 100). El proyecto alarma contra siniestros natura-

les y producidos por el hombre, que pueden ser temblores, incendios, intrusos, tiene las siguientes características:

Elementos electrónicos de bajo costo, y que existen en el mercado

Este proyecto tiene incorporado varios sensores, su función es accionar el encendido y apagado de focos, en el momento que exista movimiento accionar una alarma, abrir o cerrar cortinas, y además tiene incorporados sensores electromagnéticos que serán instalados en diferentes puntos importantes, pueden ser las ventanas o las puertas.

Este proyecto tiene como objetivo general controlar la seguridad de una casa y que sea monitoreada desde un celular (ver Figura 101).

Elementos necesarios para el proyecto

En forma general, se necesita elementos eléctricos, elementos electrónicos activos y pasivos que se detalla a continuación:

Un microcontrolador 16F877A que es el elemento fundamental para el funcionamiento del proyecto, una pantalla de cristal líquido de 20*4 la que servirá como monitor, 1 circuito integrado que trabaja como sensor de temperatura LM35, una fotorresistencia conocida también con el nombre de LDR, un ventilador de 6200 r. p. m. a 12 voltios, 300 miliamperios, 1 motor de corriente continua de 5 voltios/300 miliamperios, una sirena de 12 voltios/200 miliamperios, tres sensores con características especiales para cada función que son: sensor de movimiento PIR, sensor de humo y gas Q-4 , sensor de vibración, un switch de 1 amperio dos posiciones, 8 pulsadores normalmente abiertos, 4 diodos led de colores rojo, verde, amarillo que son utilizados como indicadores de funcionamiento, 3 potenciómetros simples para variar la sensibilidad y voltajes, 15 resistencias de ¼ de vatio, 6 diodos rectificadores 1N4005 o similares, 8 capacitores cerámicos del rango de picofaradios, un circuito integrado regulador LM358 , zócalos para el microcontrolador, 3 focos de 110 60 vatios, 2 TRIAC BT 138 que sirven como interruptores electrónicos.

Los materiales son de bajo costo y no tienen características especiales para su adquisición, se pueden conseguir en cualquier tienda comercial de electrónica.

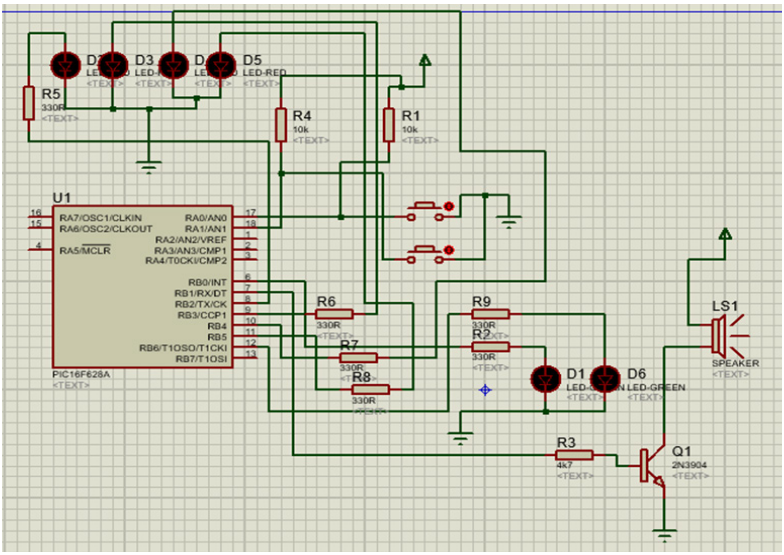


Figura 94. Estructura del parlante

Nota. El parlante es el dispositivo de salida para las programaciones de SOUND (Taller Electrónicos Brim, 2020).

```
cmcon=7
;xy var byte
Larma:
    HIGH portb.2
    PAUSE 200
    LOW portb.2
    PAUSE 200
    HIGH portb.3
    PAUSE 200
    LOW portb.3
    PAUSE 200
    HIGH portb.4
    PAUSE 200
    LOW portb.4
    PAUSE 200
    HIGH portb.5
    PAUSE 200
    LOW portb.5
    PAUSE 200
    HIGH portb.4
    PAUSE 200
    LOW portb.4
    PAUSE 200
    HIGH portb.3
    PAUSE 200
    LOW portb.3
    PAUSE 200
    HIGH portb.2
    PAUSE 200
    LOW portb.2
    PAUSE 200
      IF porta.0 =0 THEN A
      GOTO Larma
      IF porta.1 =0 THEN B
      GOTO Larma
A:
   HIGH portb.0
   PAUSE 100
   ;low portb.0
   ;pause 100
   HIGH portb.6
   PAUSE 300
   LOW portb.6
   PAUSE 300
   SOUND portb.1,[125,4,123,5]
   IF porta.1 =0 THEN B
   GOTO A
B:
    LOW portb.0
    PAUSE 200
    LOW portb.1
    PAUSE 200
  GOTO Larma

   END
```

Figura 96. Líneas de programación

Nota. En las líneas de programación se puede mostrar la utilización de las sentencias para el sonido (Electrónicas y Chiluisa, 2017).

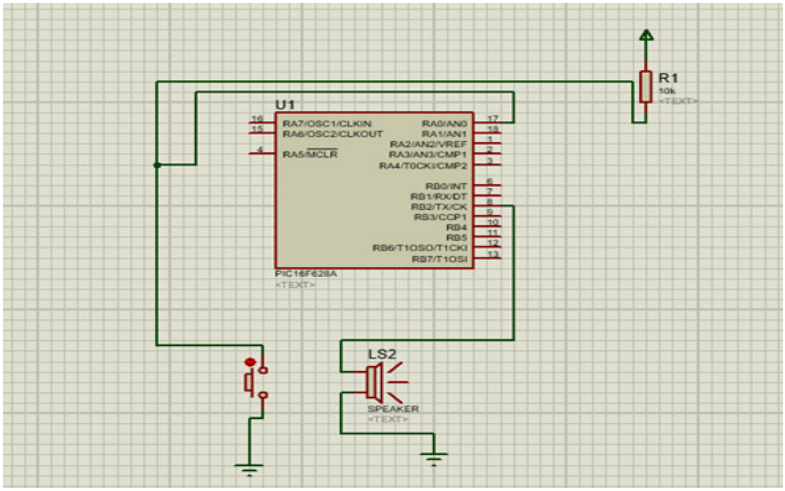


Figura 97. Alarma de sonido

Nota. Proyecto generador de sonido cuando se activa el pulsador (Electrónicas y Chiluisa, 2017).

```
[DeviceName]
Value=P16F628A
[DeviceClock]
Value=8
[MainUnit]
Value=Sonido.pbas
[DeviceFlags]
COUNT=3
Value0=_WDT_OFF = $3FFB
Value1=_LVP_OFF = $3F7F
Value2=_INTRC_OSC_NOCLKOUT = $3FFC
[ProjectFiles]
COUNT=0
[OtherFiles]
COUNT=0
[TabIndex]
Value=0
[SearchPath]
COUNT=3
Value0=C:\Defs\
Value1=C:\p16\
Value2=C:\PROYECTOS\5 proyectos infrarojo\
[EEPROMinfo]
isused=0
```

Figura 98. Programación de una alarma básica

Nota. También puede utilizar otras directivas para la programación (Electrónicas y Chiluisa, 2017).

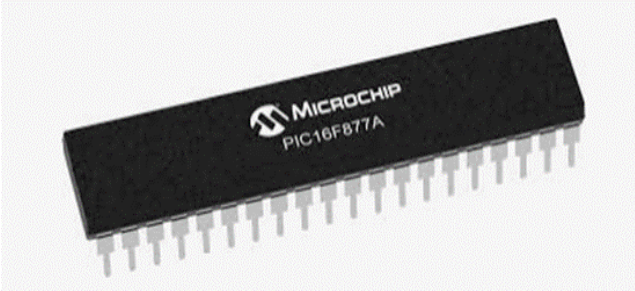


Figura 99. El microcontrolador
Nota. Dispositivo electrónico para ser programado (Microcontroladores Microchip, 2019).

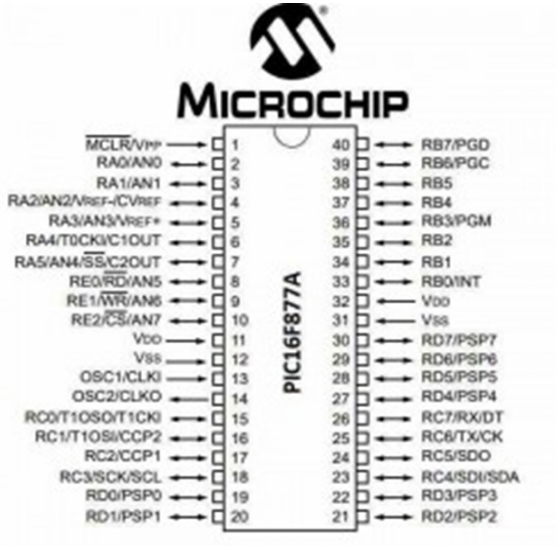


Figura 100. Estructura del microcontrolador
Nota. Nombres de los terminales del microcontrolador (Microcontroladores Microchip, 2019).

El software PROTEUS es utilizado para simular y diseñar las placas electrónicas, lo que permite optimizar recursos y tiempo. Las pruebas necesarias con los sensores e interruptores también son realizadas con el software.

La programación del microcontrolador está realizada en el ID MicroCode, y la grabación del PIC en el grabador universal.

Estructura de la alarma

(ver Figura 102)

Etapa de potencia

(ver Figura 103)

Etapa del puente H

(ver Figura 104)

Etapa de sensores

(ver Figura 105)

Placas electrónicas

El software PROTEUS es un programa completo que permite diseñar y simular circuitos electrónicos; la placa electrónica que se observa en la figura es una demostración del gran aporte de este software. Tenemos el diseño del puente H conectado a algunas borneras que servirán para conectar el motor y los pulsadores (ver Figura 106).

Las líneas de conducción están diseñadas a un solo lado con sus respectivas dimensiones para soportar el amperaje respectivo. La dimensión de los orificios de los componentes electrónicos nos proporciona automáticamente PROTEUS.

Una vez concluido el diseño se debe realizar el proceso de la elaboración de la placa electrónica, que consiste en la utilización de ácido férrico y la baquelita sin perforar.

A continuación, se muestra las figuras de las etapas en la alarma diseñadas en PROTEUS.

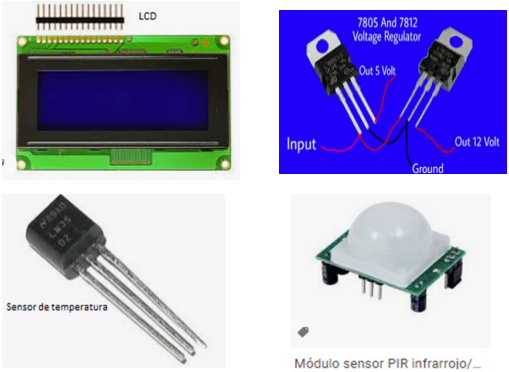


Figura 101. Principales elementos utilizados en el proyecto
Nota. Elementos electrónicos LCD para la visualización, IC reguladores para su correcto funcionamiento, sensores de temperatura y de movimientos. (Ingeniería Mecafenix, 2019)

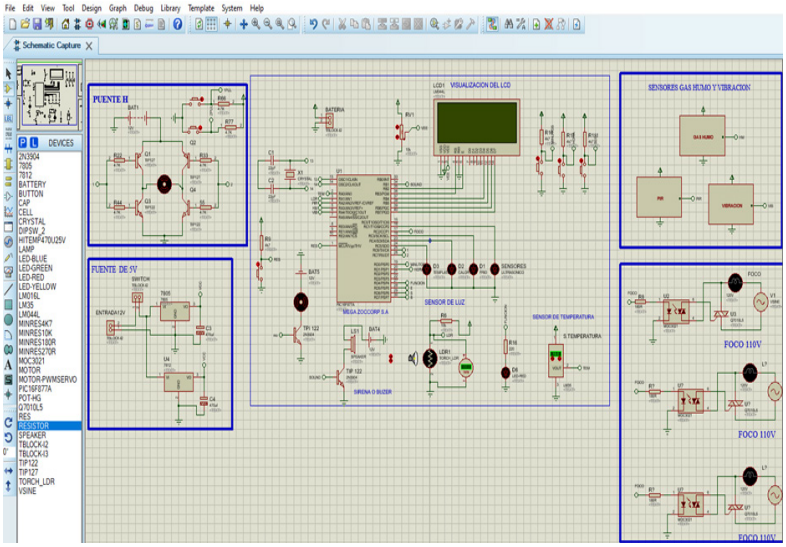


Figura 102. Diagrama total de la alarma
Nota. Diagrama esquemático con sus respectivas etapas (Electrónicas y Chiluisa, 2017)

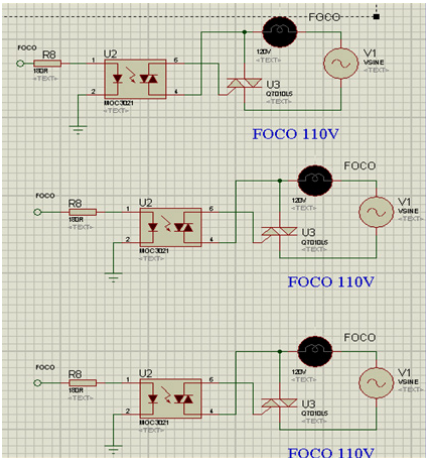


Figura 103. Diagrama de dispositivos de potencia
Nota. Se utiliza un optoacoplador (Electrónicas y Chiluisa, 2017).

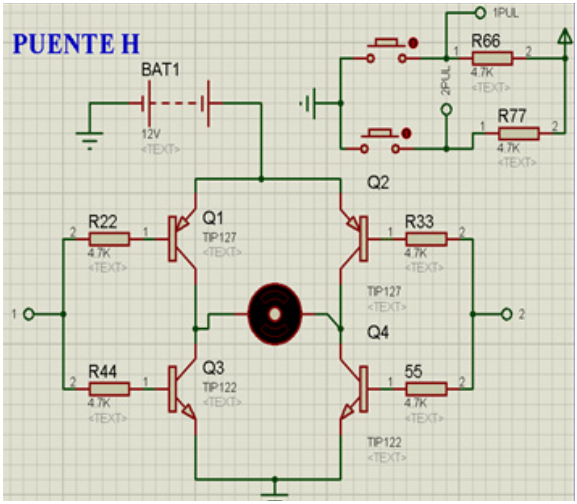


Figura 104. Configurados con transistores bipolares
Nota. La función principal es invertir el giro del motor (Electrónicas y Chiluisa, 2017).

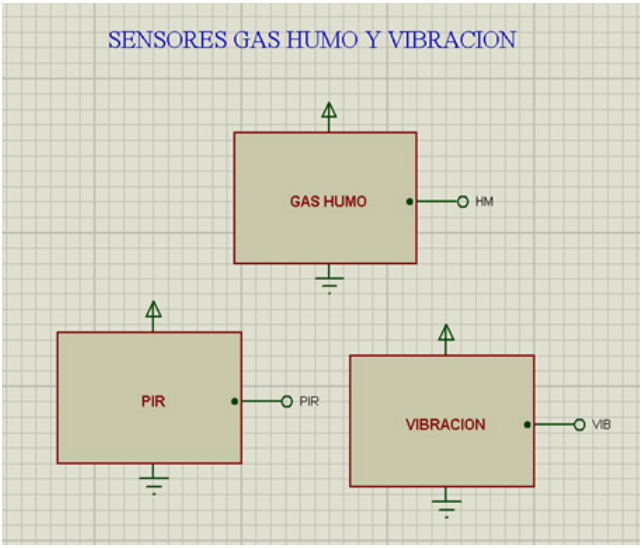


Figura 105. Esquema en bloque de sensores
Nota. Los sensores utilizados en PROTEUS (Electrónicas y Chiluisa, 2017).

Placa electrónica de voltajes

En esta sección la regulación de voltaje es la más importante para el funcionamiento de todo el proyecto, la tensión que regulan es 12 v con un amperaje de 800 miliamperios, 5 v con una corriente de 500 miliamperios. Estos voltajes y amperajes son los óptimos para el funcionamiento del microcontrolador y las interfaces (ver Figura 107).

La placa electrónica está diseñada para insertar borneras, con sus respectivos acoples eléctricos.

Placa electrónica de potencia

(ver Figura 108)

Placa electrónica del microprocesador

El funcionamiento del circuito electrónico y eléctrico es un éxito en el simulador PROTEUS. Se pudo realizar las correcciones necesarias en las diferentes etapas que conforman el proyecto (ver Figura 109).

Para la programación se utilizó MicroCode Studio; a continuación, presentamos el código fuente de este proyecto.

Programación

(ver Figura 110)

Ejercicios propuestos

Diseñar un circuito electrónico de 32 LED, 16 diodos LED que se enciendan de forma sincronizada de izquierda a derecha, y los otros 16 diodos LED de derecha a izquierda.

Diseñar un circuito electrónico con dos pulsadores y dos motores, que con el primer pulsador los motores trabajen en el sentido horario en un tiempo de 500 ms, cuando se digite el siguiente pulsador funcionen los motores en sentido antihorario en un tiempo de 1000 milisegundos. Para repetir este proceso se necesita digitar los dos pulsadores juntos, caso contrario se queda bloqueado.

Diseñar un circuito con una LCD y un mensaje de bienvenida. En el momento que se digite un pulsador se despliegue otro mensaje por un tiempo de 300 milisegundos.

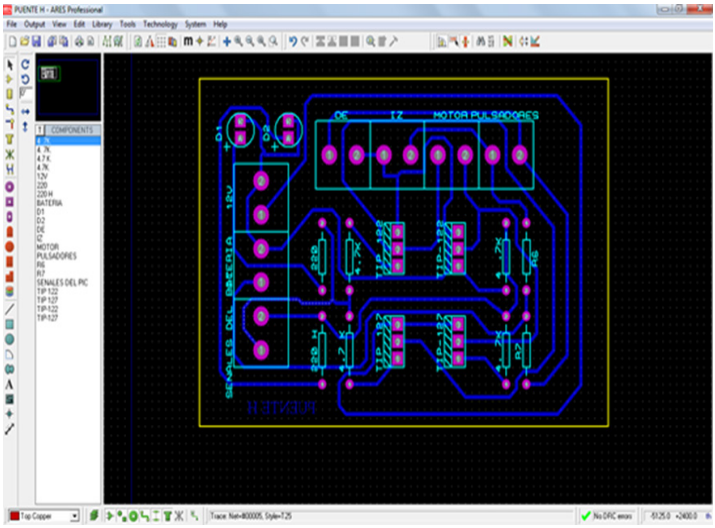


Figura 106. Diseño de la placa del puente H
Nota. Este diseño es realizado en PROTEUS (Electrónicas y Chiluisa, 2017)

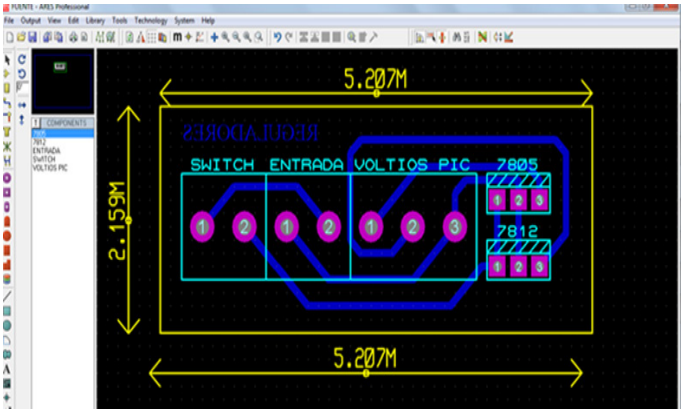


Figura 107. Regulador
Nota. Etapa reguladora de voltaje de 5 v y 12 v (Electrónicas y Chiluisa, 2017).

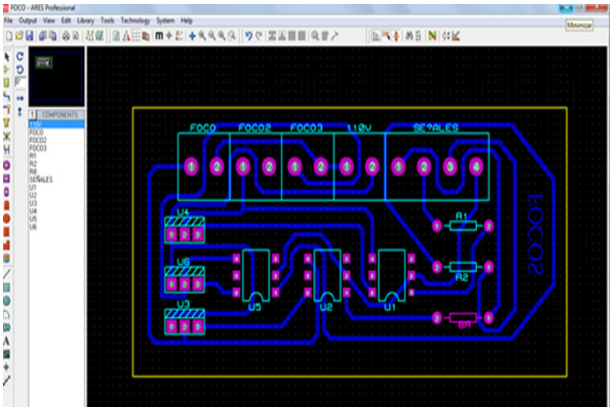


Figura 108. Circuito de SCR y optoacopladores
Nota. El circuito está compuesto por 3 SCR, 3 optoacopladores y las borneras (Electrónicas y Chiluisa, 2017).

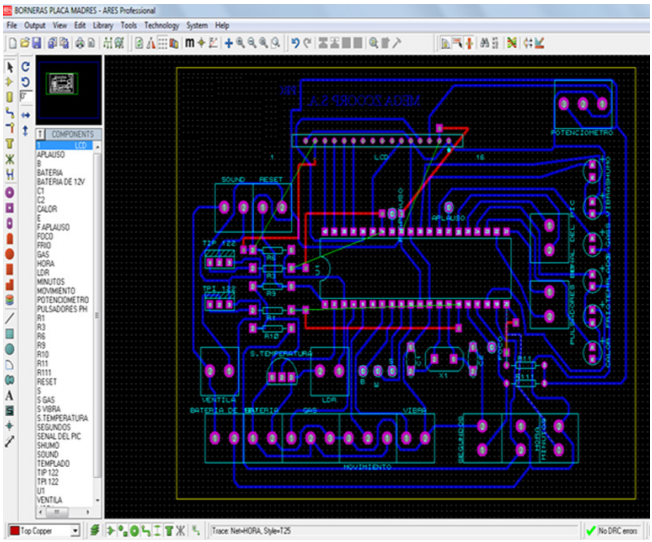


Figura 109. Circuito CPU con el PIC16F877A
Nota. Etapa controladora de la alarma con el PIC16F877A (Electrónicas y Chiluisa, 2017).

```
11   @ DEVICE HS_OSC
12 DEFINE OSC 10
13 1.  DEFINE LCD_DREG PORTC
14 2.  DEFINE LCD_DBIT 4
15 3.  DEFINE LCD_RSREG PORTD
16 4.  DEFINE LCD_RSBIT 1
17 5.  DEFINE LCD_EREG PORTD
18 6.  DEFINE LCD_EBIT 0
19 7.  DEFINE ADC_BITS 8        ' Set number of bits in result
20 8.  DEFINE ADC_CLOCK 3     ' Set clock source (rc = 3)
21 9.  DEFINE ADC_SAMPLEUS 50
22 10. ADCON1=2
23 11. TRISA=255
24 12. NUMERO VAR BYTE
25 13. SONIDO VAR BYTE
26 14. Z VAR BYTE
27 15. A VAR PORTB.0
28 16. B VAR PORTB.1
29 17. C VAR PORTB.2
30 18. D VAR PORTB.3
31 19. UNO VAR PORTB.4
32 20. DOS VAR PORTB.5
33 21. TRES VAR PORTB.6
34 22. CUATRO VAR PORTB.7
35 23. RELE VAR PORTD.6
36 24. BUZZER VAR PORTD.7
37 25. Y VAR BYTE
38 26. C1 VAR BYTE
39 27. C2 VAR BYTE
40 28. C3 VAR BYTE
41 29. C4 VAR BYTE
42 30. T1 VAR BYTE
43 31. T2 VAR BYTE
44 32. T3 VAR BYTE
45 33. T4 VAR BYTE
46 34. AUX1 VAR BYTE
47 35. AUX2 VAR BYTE
48 36. AUX3 VAR BYTE
```


```
46 34. AUX1 VAR BYTE
47 35. AUX2 VAR BYTE
48 36. AUX3 VAR BYTE
```


```
49 37. AUX 4 VAR BYTE
50 38. HUMO VAR BYTE
51 39. PIR VAR BYTE
52 40. VIBRA VAR BYTE
53 41. S VAR BYTE
54 42. S=15
55 43. M VAR BYTE
56 44. M=30
57 45. H VAR BYTE
58 46. H=12
59 47. X VAR BYTE
60 48. LED1 VAR PORTC.0
61 49. LED2 VAR PORTC.1
62 50. ST VAR BYTE
63 51. ST=0
```

```
52. ERROR VAR BYTE
53. ERROR=0
54. HIGH LED1
55. HIGH LED1
56. LCDOUT $FE,1,"COL. TEC. IND 6TO B1"
57. LCDOUT $FE,$C0,"MIGUEL DE SANTIAGO"
58. LCDOUT $FE,$94," PROYECTO DE GRADO"
59. LCDOUT $FE,$D4,"ALARMA DE SEGURIDAD"
60. PAUSE 3000
61. FOR X= 1 TO 5
62. HIGH BUZZER:PAUSE 50:LOW BUZZER:PAUSE 50
63. NEXT X
64. PAUSE 3000
65. LOW LED1
66. LOW LED2
67. READ 0,C1
68. READ 1,C2
69. READ 2,C3
70. READ 3,C4
71. DESARMADO:
72. LOW BUZZER
```

```
70. READ 3,C4
71. DESARMADO:
72. LOW BUZZER
73. LCDOUT $FE,1,"**DIGITE CLAVE**"
74. PAUSE 500
75. GOSUB BARRIDOS:GOSUB ESPACIO
76. T1=NUMERO
77. GOSUB BARRIDOS:GOSUB ESPACIO
78. T2=NUMERO
79. GOSUB BARRIDOS:GOSUB ESPACIO
80. T3=NUMERO
81. GOSUB BARRIDOS:GOSUB ESPACIO
82. T4=NUMERO
83. IF T1=C1 AND T2=C2 AND T3=C3 AND T4=C4 THEN ;***CLAVE
84. LCDOUT $FE,1,"   CLAVE CORRECTA"
85. LCDOUT $FE,$C0,"  ***BIENVENIDO***"
86. LCDOUT $FE,$94," SISTEMA TOTALMENTE"
87. LCDOUT $FE,$D4,"       ACTIVADO"
88. PAUSE 3000
89. FOR X= 1 TO 2
90. HIGH BUZZER:PAUSE 80:LOW BUZZER:PAUSE 80
91. NEXT X
92. HIGH RELE
93. PAUSE 3000
94. GOTO ARMADO
95. ELSE
96. LCDOUT $FE,1,"CLAVE INCORRECTA"
97. HIGH BUZZER
98. PAUSE 3000
99. LOW BUZZER
100.    ENDIF
101.    GOTO DESARMADO
102.    ARMADO:
103.    LCDOUT $FE,1,"DIGIT CLAVE ",DEC2 H,":",DEC2 M,":",DEC2 
104.    GOSUB BARRIDOS2:GOSUB ESPACIO
```

```
105.    IF NUMERO=10 THEN MCLAVE
106.    IF NUMERO=11 OR NUMERO=12 THEN
107.    LCDOUT $FE,$94," SISTEMA SECUNDARIO"
108.    LCDOUT $FE,$D4,"     DESACTIVADO"
```

```
107.    LCDOUT $FE,$94," SISTEMA SECUNDARIO"
108.    LCDOUT $FE,$D4,"     DESACTIVADO"
109.    LOW RELE
110.    PAUSE 5000
111.    ENDIF
112.    GOTO ARMADO
113.    MCLAVE:
114.    LOW BUZZER
115.    LCDOUT $FE,1,"***CAMBIO DE****"
116.    LCDOUT $FE,$C0,"      CLAVE"
117.    PAUSE 1000
118.    READ 0,C1
119.    READ 1,C2
120.    READ 2,C3
121.    READ 3,C4
122.    LCDOUT $FE,1,"  DIGITE SU CLAVE"
123.    GOSUB TECLADO 22:GOSUB SOLTAR 2:AUX1=NÚMERO:LCDOUT $FE,$C5,"*"
124.    IF NUMERO=10 THEN DESARMADO
125.    GOSUB TECLADO 22:GOSUB SOLTAR 2:AUX2=NÚMERO:LCDOUT $FE,$C6,"*"
126.    GOSUB TECLADO 22:GOSUB SOLTAR 2:AUX3=NÚMERO:LCDOUT $FE,$C7,"*"
127.    GOSUB TECLADO 22:GOSUB SOLTAR 2:AUX4=NUMERO:LCDOUT $FE,$C8,"*"
128.    IF C1=AUX1 AND C2=AUX2 AND C3=AUX3 AND C4=AUX4 THEN
129.    LCDOUT $FE,1," DIGITE NUEVA CLAVE"
130.    GOSUB TECLADO22:GOSUB SOLTAR2:C1=NUMERO:LCDOUT $FE,$C5,"*"
131.    GOSUB TECLADO22:GOSUB SOLTAR2:C2=NUMERO:LCDOUT $FE,$C6,"*"
132.    GOSUB TECLADO22:GOSUB SOLTAR2:C3=NUMERO:LCDOUT $FE,$C7,"*"
133.    GOSUB TECLADO22:GOSUB SOLTAR2:C4=NUMERO:LCDOUT $FE,$C8,"*"
134.    LCDOUT $FE,1,"CONFIRME NUEVA CLAVE"
135.    GOSUB TECLADO22:GOSUB SOLTAR2:AUX1=NUMERO:LCDOUT $FE,$C5,"*"
136.    GOSUB TECLADO22:GOSUB SOLTAR2:AUX2=NUMERO:LCDOUT $FE,$C6,"*"
137.    GOSUB TECLADO22:GOSUB SOLTAR2:AUX3=NUMERO:LCDOUT $FE,$C7,"*"
138.    GOSUB TECLADO22:GOSUB SOLTAR2:AUX4=NUMERO:LCDOUT $FE,$C8,"*"
139.    IF C1=AUX1 AND C2=AUX2 AND C3=AUX3 AND C4=AUX4 THEN
140.    LCDOUT $FE,1,"   CAMBIO DE CLAVE"
141.    LCDOUT $FE,$C0,"      EXITOSO"
142.    PAUSE 4000
143.    WRITE 0,C1
144.    WRITE 1,C2
144.    WRITE 1,C2
145.    WRITE 2,C3
146.    WRITE 3,C4
147.    GOTO ARMADO
148.    ENDIF
149.    ENDIF
150.    LCDOUT $FE,1," CLAVE INCORRECTA"
151.    LCDOUT $FE,$C0,"        ERROR"
152.    PAUSE 4000
153.    GOTO MCLAVE
154.    BARRIDOS:
155.    LCDOUT $FE,1,"DIGIT CLAVE ",DEC2 H,":",DEC2 M,":",DEC2 S
156.    LCDOUT $FE,$C0," PROYECTO DE GRADO"
157.    LCDOUT $FE,$94,"  SISTEMA ACTIVADO"
158.    LCDOUT $FE,$D4,"****CASA SEGURA*****"
```

```
159.    ADCIN 0,HUMO
160.    ADCIN 1,PIR
161.    ADCIN 2,VIBRA
162.    IF HUMO>200 THEN
163.    LCDOUT $FE,1
164.    LCDOUT $FE,$C0,"HUMO DETECTADO: ", DEC HUMO
165.    HIGH BUZZER :HIGH LED2
166.    S=S+1
167.    PAUSE 1000
168.    LOW BUZZER  :LOW LED2
169.    ENDIF
170.    IF PIR<10 THEN
171.    LCDOUT $FE,1
172.    LCDOUT $FE,$94,"PIR DETECTADO: ",DEC PIR
173.    HIGH BUZZER: HIGH LED2
174.    S=S+1
175.    PAUSE 1000
176.    LOW BUZZER:LOW LED2
177.    ENDIF
178.    IF VIBRA>=175 THEN
179.    LCDOUT $FE,1
180.    LCDOUT $FE,$D4,"VIBRA DETECTADA: ",DEC VIBRA
181.    HIGH BUZZER:HIGH LED2
182.    S=S+1
183.    PAUSE 1000
184.    LOW BUZZER:LOW LED2
185.    ENDIF
186.    PAUSE 990
187.    IF PORTD.4=0 THEN
188.    M=M+1
189.    ENDIF
190.    IF PORTD.5=0 THEN
191.    H=H+1
192.    ENDIF
193.    LOW A
194.    IF UNO =0 THEN NUMERO = 1:RETURN
195.    IF DOS =0 THEN NUMERO =4:RETURN
196.    IF TRES =0 THEN NUMERO =7:RETURN
197.    HIGH A
198.    LOW B
199.    IF UNO=0 THEN NUMERO =2:RETURN
200.    IF DOS=0 THEN NUMERO =5:RETURN
201.    IF TRES=0 THEN NUMERO =8:RETURN
202.    IF CUATRO=0 THEN NUMERO =0:RETURN
203.    HIGH B
204.    LOW C
205.    IF UNO=0 THEN NUMERO =3:RETURN
206.    IF DOS=0 THEN NUMERO =6:RETURN
207.    IF TRES=0 THEN NUMERO =9:RETURN
208.    HIGH C
209.    LOW D
210.    IF UNO=0 THEN NUMERO =10:RETURN
211.    IF DOS=0 THEN NUMERO =11:RETURN
212.    IF TRES=0 THEN NUMERO =12:RETURN
213.    HIGH D
```

```
214.    S=S+1
215.    IF S>59 THEN
216.    M=M+1
217.    S=0
218.    ENDIF
219.    IF M>59 THEN
220.    M=0
221.    H=H+1
222.    ENDIF
223.    IF H>23 THEN
224.    H=0
225.    ENDIF
226.    GOTO BARRIDOS
227.    BARRIDOS2:
228.    LCDOUT $FE,1,"DIGIT CLAVE ",DEC2 H,":",DEC2 M,":",DEC2 S
229.    LCDOUT $FE,$C0," PROYECTO DE GRADO"
230.    LCDOUT $FE,$94," SISTEMA TOTALMENTE"
231.    LCDOUT $FE,$D4,"      ACTIVADO"
232.    ADCIN 0,HUMO
233.    ADCIN 1,PIR
234.    ADCIN 2,VIBRA
235.    ADCIN 3,SONIDO
236.    IF HUMO>200 THEN
237.    LCDOUT $FE,1
238.    LCDOUT $FE,$C0,"HUMO DETECTADO: ", DEC HUMO
239.    HIGH BUZZER :HIGH LED2
240.    S=S+1
241.    PAUSE 1000
242.    LOW BUZZER  :LOW LED2
243.    ENDIF
244.    IF PIR<10 THEN
245.    LCDOUT $FE,1
246.    LCDOUT $FE,$94,"PIR DETECTADO: ",DEC PIR
247.    HIGH BUZZER: HIGH LED2
248.    S=S+1
249.    PAUSE 1000
250.    LOW BUZZER:LOW LED2
251.    ENDIF
252.    IF VIBRA>=175 THEN
253.    LCDOUT $FE,1
254.    LCDOUT $FE,$D4,"VIBRA DETECTADA: ",DEC VIBRA
255.    HIGH BUZZER:HIGH LED2
256.    S=S+1
257.    PAUSE 1000
258.    LOW BUZZER:LOW LED2
259.    ENDIF
260.    FOR Y= 1 TO 100
261.    ADCIN 3,SONIDO
262.    IF PORTD.3=0 THEN
263.    LCDOUT $FE,$94," SERVOMOTOR GIRANDO"
264.    LCDOUT $FE,$D4,"SENTIDO ANTIHORARIO"
265.    PAUSE 500
266.    FOR Z=1 TO 30
267.    HIGH PORTC.3:PAUSEUS 2500:LOW PORTC.3:PAUSE 20
268.    NEXT Z
269.    ENDIF
270.    IF PORTD.2=0 THEN
```

```
271.    LCDOUT $FE,$94," SERVOMOTOR GIRANDO"
272.    LCDOUT $FE,$D4,"  SENTIDO HORARIO"
273.    PAUSE 500
274.    FOR Z= 1 TO 30
275.    HIGH PORTC.3:PAUSEUS 500:LOW PORTC.3:PAUSE 20
276.    NEXT Z
277.    ENDIF
278.    IF SONIDO>=40 THEN
279.    IF ST=0 THEN
280.    ST=1
281.    HIGH LED1
282.    PAUSE 500
283.    GOTO BARRIDOS2
284.    ENDIF
285.    IF ST=1 THEN
286.    ST=0
287.    LOW LED1
288.    PAUSE 500
289.    GOTO BARRIDOS2
290.    ENDIF
291.    ENDIF
292.    PAUSE 9
293.    NEXT Y
294.    IF PORTD.4=0 THEN
295.    M=M+1
296.    ENDIF
297.    IF PORTD.5=0 THEN
298.    H=H+1
299.    ENDIF
300.    LOW D
301.    IF UNO=0 THEN NUMERO =10:RETURN
302.    IF DOS=0 THEN NUMERO =11:RETURN
303.    IF TRES=0 THEN NUMERO =12:RETURN
304.    HIGH D
305.    PAUSE 990
306.    S=S+1
307.    IF S>59 THEN
308.    M=M+1
309.    S=0
310.    ENDIF
311.    IF M>59 THEN
312.    M=0
313.    H=H+1
314.    ENDIF
315.    IF H>23 THEN
316.    H=0
317.    ENDIF
318.    GOTO BARRIDOS2
319.    BARRIDO:
320.    LOW A
321.    IF UNO =0 THEN NUMERO = 1:RETURN
322.    IF DOS =0 THEN NUMERO =4:RETURN
```

```
335 323.    IF TRES =0 THEN NUMERO =7:RETURN
336 324.    HIGH A
337 325.    LOW B
338 326.    IF UNO=0 THEN NUMERO =2:RETURN
339 327.    IF DOS=0 THEN NUMERO =5:RETURN
340 328.    IF TRES=0 THEN NUMERO =8:RETURN
341 329.    IF CUATRO=0 THEN NUMERO =0:RETURN
342 330.    HIGH B
343 331.    LOW C
344 332.    IF UNO=0 THEN NUMERO =3:RETURN
345 333.    IF DOS=0 THEN NUMERO =6:RETURN
346 334.    IF TRES=0 THEN NUMERO =9:RETURN
347 335.    HIGH C
348 336.    LOW D
349 337.    IF UNO=0 THEN NUMERO =10:RETURN
350 338.    IF DOS=0 THEN NUMERO =11:RETURN
351 339.    IF TRES=0 THEN NUMERO =12:RETURN
352 340.    HIGH D
353 341.    PAUSE 200
354 342.    GOTO BARRIDO
355 343.    TECLADO22:
356 344.    LOW A
357 345.    IF UNO =0 THEN NUMERO = 1:RETURN
358 346.    IF DOS =0 THEN NUMERO =4:RETURN
359 347.    IF TRES =0 THEN NUMERO =7:RETURN
360 348.    HIGH A
361 349.    LOW B
362 350.    IF UNO=0 THEN NUMERO =2:RETURN
363 351.    IF DOS=0 THEN NUMERO =5:RETURN
364 352.    IF TRES=0 THEN NUMERO =8:RETURN
365 353.    IF CUATRO=0 THEN NUMERO =0:RETURN
366 354.    HIGH B
367 355.    LOW C
368 356.    IF UNO=0 THEN NUMERO =3:RETURN
369 357.    IF DOS=0 THEN NUMERO =6:RETURN
370 358.    IF TRES=0 THEN NUMERO =9:RETURN
371 359.    HIGH C
372 360.    LOW D
373 361.    IF UNO=0 THEN NUMERO =10:RETURN
374 362.    IF DOS=0 THEN NUMERO =11:RETURN
375 363.    IF TRES=0 THEN NUMERO =12:RETURN
376 364.    HIGH D
377 365.    PAUSE 200
378 366.    GOTO TECLADO22
379 367.    ESPACIO:
380 368.    HIGH PORTD.7
381 369.    IF UNO =0 THEN ESPACIO
382 370.    IF DOS =0 THEN ESPACIO
383 371.    IF TRES =0 THEN ESPACIO
384 372.    IF CUATRO =0 THEN ESPACIO
385 373.    PAUSE 25
386 374.    LOW PORTD.7
387 375.    RETURN
```

```
376.    SOLTAR2:
377.    HIGH PORTD.7
378.    IF UNO =0 THEN SOLTAR2
379.    IF DOS =0 THEN SOLTAR2
380.    IF TRES =0 THEN SOLTAR2
381.    IF CUATRO =0 THEN SOLTAR2
382.    PAUSE 25
383.    LOW PORTD.7
384.    RETURN
```

Figura 110. Líneas de programación
Nota. Son 384 líneas de programación de la alarma para un correcto funcionamiento de los sensores, optoacopladores (Electrónicas y Chiluisa, 2017).

Referencias

Aprendiendo Arduino. (2019). *Microcontroladores*. https://aprendiendoarduino.wordpress.com/2016/06/26/microcontroladores-2/

Cabrera J. I. (27 de marzo de 2020). ¿Qué es una tarjeta de red y para qué sirve? Nobbot. https://www.nobbot.com/redes/tarjeta-de-red-que-es/

Components101. (2019). *Microcontrolador PIC16F877A, figura*. https://components101.com/pic16f877a-pin-diagram-description-features-datasheet

Deltakit. (2019). *Optoacopladores*. https://www.deltakit.net/product/speed-sensor-module-tacho-sensor-slot-optocoupler/

Direct Industry. (2021). *Atmel*. https://www.directindustry.es/prod/atmel/product-13779-584868.html

Electrónicas, P., & Chiluisa, M. A. (2017). *Microcontroladores*. Quito, Pichincha, Ecuador.

EPA. (2018). *reloj*. Obtenido de Electrónica práctica aplicación.

Helmut Sy Corvo. (2019). *Arquitectura Harvard: origen, modelo, cómo funciona*. Lifeder. https://www.lifeder.com/arquitectura-harvard/

HETPRO. (2021). *Convertidor analógico digital ADC ADC0804*. https://hetpro-store.com/adc0804/

Ingeniería Mecafenix. (2019). *Microcontrolador PIC [Partes y aplicaciones]*. https://www.ingmecafenix.com/electronica/microcontrolador-pic-partes-aplicaciones/

Llamas Luis. (18 de enero de 2020). *Teclado matricial*. Ingeniería, informática y diseño. https://www.luisllamas.es/arduino-teclado-matricial/#:~:text=Como%20hemos%20dicho%2C%20un%20teclado,las%20pulsaci%C3%B3n%20de%20las%20teclas.

Microcontroladores Microchip. (2019). *Arquitectura de los PIC*. https://www.google.com/search?q=microcontrolador&tbm=isch&ved=2ahUKEwjIxOS0kbbuAhWmr1kKHRAsAGUQ2-cCegQIABAA&oq=microcontrolador&gs_lcp=CgNpbWcQAzIECCMQJzIECCMQJzIECAAQQzICCAAyAggAMgIIADICCAAyAggAMgIIADICCAA6BQgAELEDOgcIIxDqAhAnOgcIABCxAxBDOggIABCxAxCDAVCW

Sensagent. (28 de abril de 2013). *Microcontrolador*. Wikipedia. http://diccionario.sensagent.com/Microcontrolador/es-es/

Surtel electrónica. (19 de abril de 2019). *¿Qué son los componentes SMD?* ht-

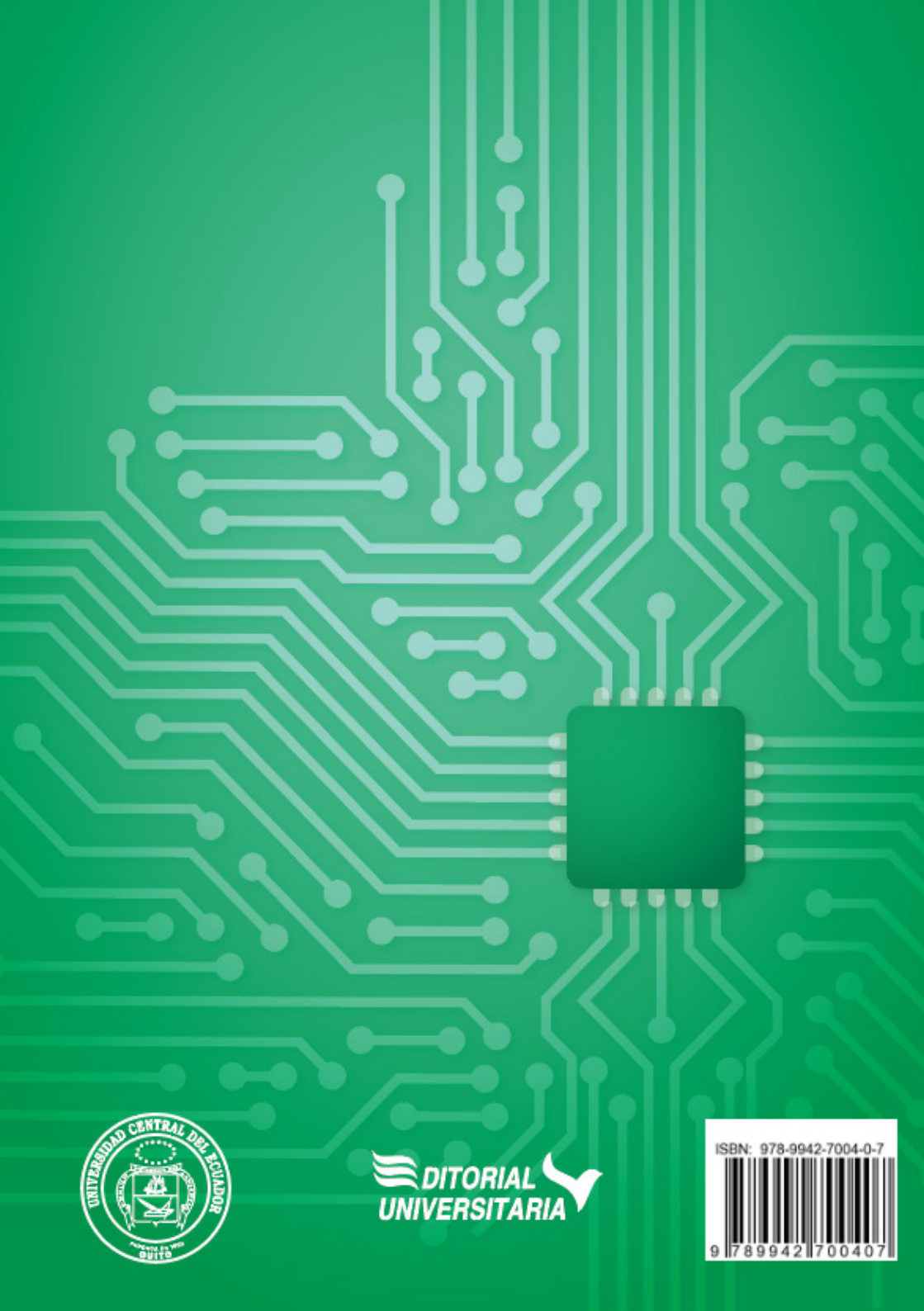
tps://www.surtel.es/blog/que-son-los-componentes-smd/

Talleres Eléctricos Brim. (2020). *Motores eléctricos de corriente continua*. https://tallereselectricosbrim.com/ca/blog-cat/84-motores-electricos-de-corriente-continua-c-c

Urbina, C. (3 de mayo de 2012). *Instrucciones de microcontroladores clásicos*. http://cecilia-urbina.blogspot.com/2012/02/instrucciones-de-microcontroladores.html

Wikipedia. (22 de noviembre de 2019). *Arquitectura de Von Neumann-figura*. https://es.wikipedia.org/wiki/Arquitectura_de_Von_Neumann

Esta revista, que usó tipografía Crimson Pro tamaño 11, se terminó de diagramar para su versión digital en Editorial Universitaria en el mes de enero de 2022 siendo rector de la Universidad Central del Ecuador el Dr. Fernando Sempértegui Ontaneda y director de Editorial Universitaria el Prof. Gustavo Pazmiño.



ISBN: 978-9942-7004-0-7