
# UNIVERSIDAD TÉCNICA DE AMBATO

## FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

### Carrera de Ingeniería en Electrónica y Comunicaciones

**TEMA:**

---

**PLACA ENTRENADORA PARA LA ENSEÑANZA DIDÁCTICA DE MICROCONTROLADORES EN LA FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL.**

---

Proyecto de Trabajo de Graduación. Modalidad: TEMI. Trabajo Estructurado de Manera Independiente, presentado previo la obtención del título de Ingeniero en Electrónica y Comunicaciones

SUBLÍNEA DE INVESTIGACIÓN: Sistemas Embebidos.

**AUTOR:** Gabriel Eduardo Santamaría Galarza.

**PROFESOR REVISOR:** Ing. Juan Pablo Pallo Mg.

Ambato - Ecuador

Febrero 2014

## APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de investigación sobre el tema: "Placa Entrenadora para la Enseñanza Didáctica de Microcontroladores en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial", del señor Santamaría Galarza Gabriel Eduardo, estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el Art. 16 del Capítulo II, del Reglamento de Graduación para obtener el título terminal de tercer nivel de la Universidad Técnica de Ambato.

Ambato Febrero 21, 2014

EL TUTOR

---

Ing. Juan Pablo Pallo Noroña, Mg.

## AUTORÍA

El presente trabajo de investigación titulado: "Placa Entrenadora para la Enseñanza Didáctica de Microcontroladores en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial". Es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato Febrero 21, 2014

---

Gabriel Eduardo Santamaría Galarza

CC: 180399844-0

## APROBACIÓN DE LA COMISIÓN CALIFICADORA

La Comisión Calificadora del presente trabajo conformada por los señores docentes: Ing. Edison Homero Álvarez Mayorga, Ing. Santiago Manzano Villafuerte, Mg., e Ing. Luis A. Pomaquero Moreno, Mg., revisó y aprobó el Informe Final del trabajo de graduación titulado "Placa Entrenadora para la Enseñanza Didáctica de Microcontroladores en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial", presentado por el señor Gabriel Eduardo Santamaría Galarza, de acuerdo al Art. 18 del Reglamento de Graduación para obtener el título Terminal de tercer nivel de la Universidad Técnica de Ambato.

…….

Ing. Edison Homero Álvarez Mayorga, Mg.

PRESIDENTE DEL TRIBUNAL

…….. ……...

Ing. Santiago Manzano Villafuerte, Mg. Ing. Luis A. Pomaquero Moreno, Mg.

DOCENTE CALIFICADOR DOCENTE CALIFICADOR

**DEDICATORIA**

A mi Madre, apoyo incondicional a lo largo de toda mi vida, ejemplo, luz, sabiduría y esperanza.

A la Vida, que me ha permitido culminar con éxito las etapas que hasta ahora me ha propuesto.

**AGRADECIMIENTO**

A Dios, Padre Eterno,
Arquitecto del Pensamiento.

A mi Madre, sin cuya guía
nada hubiera sido posible.

A mi Padre, que a lo lejos ha
sido el apoyo de este proyecto

A mi familia, quienes a pesar
de todo, siempre me han
comprendido y apoyado.

A mis Maestros, quienes desde
la infancia me han ayudado a
comprender el mundo y sus
devenires.

A mis amigos, familia que he
ido conociendo al andar.

A todos quienes he tenido el
gusto de hallar en el camino, y que
en menor o mayor medida me han
hecho ver lo que soy y lo que sueño
llegar a ser.

## CONTENIDO

## ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

## ÍNDICE DE ECUACIONES

## RESUMEN EJECUTIVO

El propósito del presente proyecto es proponer una alternativa didáctica, que ayude en la formación de Ingenieros en Electrónica y Comunicaciones.

El contenido de la investigación esta puntualizado en 6 capítulos descritos a continuación:

En el Capítulo I, se detalla lo referente a la problemática implícita en la enseñanza de conocimientos técnicos, específicamente programación de microcontroladores.

En el Capítulo II, se recopila, resume y referencia información acerca de la Electrónica y sus definiciones fundamentales; los Circuitos Integrados y Sistemas embebidos; así como un vistazo a las características de los microcontroladores en general.

En el capítulo III, se detalla la metodología utilizada para la recopilación de la información necesaria en el dimensionamiento del proyecto y la definición de los enfoques investigativos.

En el Capítulo IV, se analiza e interpreta la información recopilada en base a encuestas a los estudiantes, fichas de observación a los equipos existentes y la entrevista al docente que imparte la cátedra.

En el Capítulo V, se detallan las conclusiones y recomendaciones obtenidas a través del proceso de observación y análisis de los diversos factores que intervienen en la enseñanza de la cátedra de Microcontroladores.

Finalmente en el Capítulo VI, se desarrolla y la propuesta de la Placa Entrenadora para la Enseñanza Didáctica de Microcontroladores en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial; además de la implementación del prototipo y la liberación de sus archivos fuente para usos no comerciales, utilizando componentes disponibles en el mercado local.

# INTRODUCCIÓN

El ritmo de la evolución de la Electrónica en los últimos 50 años ha sido vertiginoso. Los avances realizados se han convertido en indispensables para la vida moderna. Procesos industriales antes desconocidos han evolucionado de tal manera que ahora son aplicados al día a día de la gente.

Los componentes que hacen esto posible son los microchips. Pequeños módulos de silicio, oro y plata que contienen millones y millones de transistores en su interior. Existen diversas maneras de configurar estos transistores, dependiendo de la utilidad del chip.

Esto ha dado lugar a diversas ramas de especialización dentro de la Ingeniería en Electrónica. En cada una de ellas se encuentran aplicaciones de microprocesadores. Y la gran mayoría de microprocesadores utilizados pertenecen a un microcontrolador. Este hecho hace que éstos sean los dispositivos más comunes en nuestro medio.

En la formación de un Ingeniero en Electrónica, es indispensable tener en claro los conceptos que rigen este mundo, dado que sea cual fuere el lugar en el cual se desempeña, el profesional se encontrará en mayor o menor medida con procesos que utilicen microcontroladores.

La formación en este campo requiere varias etapas, desde el conocimiento del *hardware* hasta el perfeccionamiento de las rutinas y procesos que se han implementado para dar solución a un problema.

Para esto, se pueden tomar una serie de caminos, sin embargo el mejor de ellos en cuanto a cualquier asignatura técnica se refiere, es la combinación de la teoría con la práctica.

Una placa de entrenamiento, permite reutilizar componentes para un gran número de prácticas, de manera fácil y cómoda. En adición, una plataforma abierta, diseñada en GNU/Linux, abre las puertas para que la enseñanza, investigación y desarrollo se hagan de manera profesional, abierta y totalmente legal.

# CAPÍTULO I

# EL PROBLEMA

## 1.1 Tema

PLACA ENTRENADORA PARA LA ENSEÑANZA DIDÁCTICA DE MICROCONTROLADORES EN LA FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

## 1.2 Planteamiento del Problema

### 1.2.1 Contextualización

A nivel mundial, las aplicaciones con microcontroladores, se diseñan con una mayor complejidad, tanto en su construcción, como en su programación. Estos dispositivos se encuentran presentes en la más amplia gama de sistemas, dotando de inteligencia artificial a avanzados circuitos, o ayudando a poner en órbita un satélite. Cada aplicación electrónica que se diseña, tiene en su interior al menos un microprocesador. Un microcontrolador es un microprocesador con los componentes esenciales para su funcionamiento, de allí que sean ampliamente utilizados. Por esta razón, en los centros de educación media y superior, que ofertan un programa de Ingeniería en Electrónica, Comunicaciones, Control y afines; existen asignaturas que se dedican al análisis de estos dispositivos. Y en todos los países existen comunidades tecnológicas que desarrollan proyectos con sistemas embebidos. Sea a nivel profesional o de aficionado, la gente que pertenece a estas comunidades, utiliza diferentes herramientas de diseño, de las cuales, una placa de desarrollo o entrenamiento está entre las más útiles.

A nivel nacional y regional, la aplicación de este conocimiento se evidencia en las diversas exposiciones de tecnología que se dan, como el Concurso

Ecuatoriano de Robótica (CER), que se lleva a cabo anualmente. En estas citas se reúnen aficionados de diversas partes del país y de la región andina. Se exhiben proyectos que representan el trabajo de diversas instituciones. La mayoría de estos desarrollos se hacen sobre placas de entrenamiento, poniendo en clara desventaja a las agrupaciones que no las poseen. Esto demuestra el interés existente sobre el uso de microcontroladores en diferentes aplicaciones electrónicas. Se pueden mencionar también los clubes de robótica que tienen diversas universidades, como la ESPOL, la Universidad Católica sede Quito, la ESPE sede Latacunga. En el Club que promueven los alumnos de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial (FISEI), en la Universidad Técnica de Ambato (UTA), no se ha tenido conocimiento de la utilización de placas entrenadoras.

En Ambato, en la FISEI, el módulo de Microcontroladores es parte de la Carrera de Ingeniería Electrónica en Comunicaciones. Aquí se incluye el tratamiento de la parte teórica sobre su funcionamiento; sin embargo los estudiantes no disponen de suficiente material didáctico apropiado, que pueda ser manipulado libremente y sin temor. Además, en estos módulos se trata sobre entornos de desarrollo integrados, diseño de aplicaciones y soluciones con sistemas embebidos, junto con la transferencia de programas a los dispositivos propiamente dichos, para el posterior análisis de su funcionamiento en prototipos realizados por los estudiantes, en los Laboratorios de Electrónica. No se han registrado casos en los cuales los estudiantes utilicen una placa de entrenamiento o desarrollo específica para microcontroladores.

### 1.2.2 Árbol del Problema

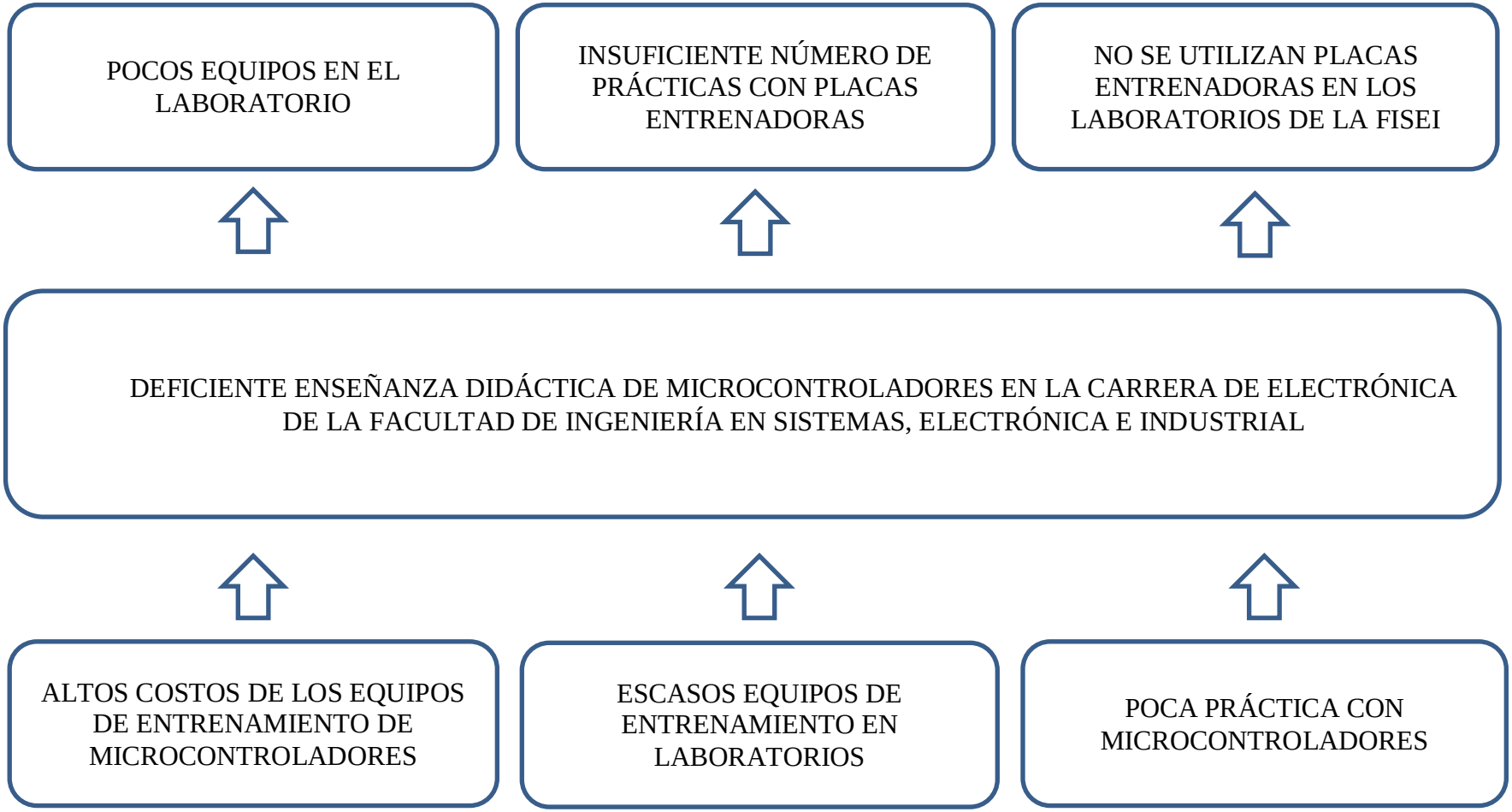



**Figura 1.1** El árbol de problemas

Elaborado por: El investigador.

### 1.2.3 Análisis Crítico

Los altos costos de los equipos de entrenamiento de microcontroladores, se deben a que la gran mayoría de modelos que ofertan fabricantes alrededor del mundo, incluyen, entre otras cosas, una gran variedad de módulos. Algunos de ellos útiles para la iniciación al aprendizaje, como aquellos que contienen leds, pulsadores, relés y algún elemento intermedio adicional. Otros que, utilizados para prácticas más avanzadas, como pantallas táctiles o acelerómetros, etc. A esto, se suman otros rubros, como licencias, distribución, publicidad; debido a estos factores existen pocos equipos en el laboratorio a disposición de los estudiantes.

Debido a los escasos equipos de entrenamiento adecuados en los laboratorios de Electrónica de la FISEI, los estudiantes tienen que reservarlos con tiempo de anticipación; o a su vez gastar tiempo y dinero en construir prototipos que en la gran mayoría de casos tendrán una funcionalidad limitada. Para esto se debe seguir el método de ensayo y error; dicho de otra manera, el estudiante deberá diseñar en papel, armar el circuito en un protoboard, detectar y corregir posibles fallos, diseñar el circuito en un programa de diseño electrónico automatizado (*EDA Electronic Design Automation*), construir la placa de circuito impreso, realizar una nueva etapa de pruebas y finalmente, obtendrá la placa terminada. Por el tiempo que requiere todo este proceso, en el ciclo académico se da un insuficiente número de prácticas con placas entrenadoras.

La poca práctica con microcontroladores, reduce las posibilidades de implementación de proyectos, al hacerse obligatoria la fabricación de circuitos de prueba básicos por parte de los estudiantes, aumentando de esta manera la dificultad para el entendimiento práctico sobre estos dispositivos; así como acrecentando una innecesaria complejidad y disminuyendo la utilidad de los prototipos que se construyen, dedicando una gran cantidad de tiempo al análisis teórico, método de prueba y error, sin la realización de una completa abstracción de los procesos dentro de un circuito micro

controlado, resultando la no utilización de placas entrenadoras en los laboratorios de la FISEI como la herramienta apropiada para la enseñanza didáctica del módulo de microcontroladores.

### 1.2.4 Prognosis

De no dar una solución eficaz a este problema, se seguirá acortando el tiempo que los tutores de la materia misma y las secuenciales que son parte del módulo formativo; puedan dedicar eficazmente a la enseñanza sobre microcontroladores, por el hecho de que no podrán disponer de la totalidad del tiempo asignado para impartir los contenidos que corresponden curso. Al contrario, se presentarán retrasos y se introducirán puntos de falla inherentes al proceso de diseño de circuitos impresos. Se presentará también una falta de interés investigativo por parte de los estudiantes, cuando por falta de tiempo y resultados satisfactorios en un plazo razonable, no puedan avanzar sobre temas propios de sistemas embebidos, lo cual resultará en una falta de formación eficiente para los estudiantes de Ingeniería Electrónica de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato.

### 1.2.5 Formulación del Problema

¿La inexistencia de una placa entrenadora en los laboratorios de Electrónica de la FISEI, incide en la enseñanza didáctica de Microcontroladores?

### 1.2.6 Preguntas Directrices

¿Cuál es el sistema actual de enseñanza de microcontroladores en los laboratorios de Electrónica de la FISEI de la UTA?

¿Cuál es el material didáctico que se utiliza actualmente para la enseñanza de microcontroladores en los laboratorios de Electrónica de la FISEI de la UTA?

¿Qué propuesta se puede plantear para la enseñanza didáctica de microcontroladores en los laboratorios de Electrónica de la FISEI de la UTA?

### 1.2.7 Delimitación

**DELIMITACIÓN DE CONTENIDO**

**Área académica:** FÍSICA Y ELECTRÓNICA.

**Línea de investigación:** SISTEMAS ELECTRÓNICOS.

**Sublínea:** SISTEMAS EMBEBIDOS.

**DELIMITACIÓN ESPACIAL**

El presente trabajo de investigación se realizará en los laboratorios de Electrónica de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato.

**DELIMITACIÓN TEMPORAL**

El presente proyecto de investigación tendrá una duración de 6 meses, a partir de la aprobación por el Honorable Consejo Directivo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

## 1.3 Justificación

El tema de investigación propuesto será de gran importancia para la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, beneficiándose gran parte de sus integrantes, puesto que una adecuada didáctica en la enseñanza de microcontroladores, logrará mejorar la calidad de la formación de los Ingenieros en Electrónica y Comunicaciones. Además se facilitará el trabajo de los docentes de Microcontroladores y materias relacionadas. Por otra parte, constituirá una valiosa herramienta para garantizar el acceso al conocimiento de los miembros del Club de Robótica de la FISEI.

El desarrollo del presente proyecto de investigación es de interés para el investigador, dado que permitirá poner en práctica los conocimientos adquiridos en el transcurso de la carrera, fundamentarlos en base a una minuciosa actividad investigativa y apoyo bibliográfico, en campos del

conocimiento relacionados a los sistemas embebidos, que son de completo agrado y utilidad para el mismo. Todo esto, permitirá llegar a resultados positivos para todos los involucrados en el presente proyecto de investigación.

Finalmente, la utilidad que se le dará a la placa entrenadora, será la de lograr el esparcimiento de conocimiento sobre microcontroladores de forma eficaz, eliminando los tiempos dedicados al ensayo y error en el diseño de prototipos, proceso anteriormente descrito de proyectar y construir un prototipo, que si bien compete a otras materias de la carrera, no ayuda a impartir eficazmente el módulo de Microcontroladores.

## 1.4 Objetivos

### 1.4.1 General

Analizar los tipos de placas entrenadoras en los laboratorios de Electrónica de la FISEI, y su incidencia en la enseñanza didáctica de Microcontroladores y sus aplicaciones.

### 1.4.2 Específicos

1. Estudiar el sistema actual de enseñanza didáctica de microcontroladores en los laboratorios de Electrónica de la FISEI de la UTA.

2. Analizar el material didáctico que se utiliza actualmente para la enseñanza de microcontroladores en los laboratorios de Electrónica de la FISEI de la UTA.

3. Proponer una placa entrenadora para la enseñanza didáctica de microcontroladores en los laboratorios de Electrónica de la FISEI de la UTA.

# CAPÍTULO II

# MARCO TEÓRICO

## 2.1 Antecedentes Investigativos

Dentro de los registros bibliográficos de la Biblioteca de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, no se encontró un tema similar al planteado.

Dentro de los repositorios digitales de las principales universidades y escuelas politécnicas de la República del Ecuador, (Internet;2013,01,21; http://www.dspace.espol.edu.ec/,repositorio.espe.edu.ec/,bibdigital.epn.edu.ec/, dspace.espoch.edu.ec), no se ha encontrado un trabajo de investigación que se refiera a una Placa Entrenadora para la Enseñanza Didáctica de Microcontroladores.

Sin embargo, en los registros de la biblioteca de la Facultad, consta la existencia de trabajos de graduación que hacen referencia la aplicación de microcontroladores en sistemas embebidos. Entre los más relevantes están los siguientes:

- "Adquisición de datos de un sistema maestro – esclavo utilizando microcontroladores mediante comunicación serial para "M&M Automatización" Ballesteros Jordán (2007); de este trabajo cabe recalcar las siguientes conclusiones:

  "Tanto Philips como otros fabricantes de dispositivos compatibles con I2C disponen de una amplia gama de circuitos integrados, incluyendo memorias RAM y E2PROM, microcontroladores, puertos de E/S, etc.

Incluso, y gracias a que el protocolo es lo suficientemente simple, usualmente se ven dispositivos I2C insertados en sistemas micro controlados que no fueron diseñados con puertos I2C, siendo el protocolo generado por el firmware."

- "Diseño de un circuito electrónico para el monitoreo de switches no administrables utilizando la tecnología del microcontrolador PIC16f877a" Moposita Valencia (2010), las conclusiones relevantes para el presente proyecto son:

  "El módulo SPI (interfaz periférica síncrona) constituye un modo de transmisión serial de datos entre dispositivos que compartan la misma tecnología, ofreciendo características óptimas de velocidad y robustez en la transferencia.

  El PIC16F877A representa un microcontrolador que se puede utilizar en una serie de aplicaciones complejas en las que se necesita que el chip posea prestaciones adicionales de desempeño con nuevas tecnologías respecto a los de gama baja.

  El circuito diseñado aparte de utilizarlo para el proyecto planteado se lo puede enfocar en una serie de aplicaciones…"

- "Diseño y construcción de un brazo robótico industrial comandado mediante un sistema de control inalámbrico" Guilcaso Molina, (2011), las conclusiones de este trabajo son:

  "Mediante la realización del diseño del brazo robótico se puede observar que el torque requerido por los motores es de 13Kg/cm para mover cada de las articulaciones siendo directamente proporcional al peso de las mismas; a más de ello con el fin de disminuir el peso de la estructura se utilizó como material de construcción el Aluminio que es más liviano que el acero.

El movimiento de cada articulación del brazo robótico depende de la pulsación del botón especificado en el control, los cuales están diferenciados por colores; además cabe recalcar que cada el movimiento es mediante la activación de los micro relés, para hacerlo girar en un sentido u otro."

## 2.2 Fundamentación Legal

El presente trabajo se encontrará fundamentado en el vigente:

- REGLAMENTO DE GRADUACIÓN PARA OBTENER EL TÍTULO TERMINAL DE TERCER NIVEL DE LA UNIVERSIDAD TÉCNICA DE AMBATO. (Universidad Técnica de Ambato, 2012)

- Se refiere a La LEY DE PROPIEDAD INTELECTUAL, (Congreso Nacional, 2006), en la sección II, literal g) reza:

  "SECCION II

  OBJETO DEL DERECHO DE AUTOR

  Art. 8. La protección del derecho de autor recae sobre todas las obras del ingenio, en el ámbito literario o artístico, cualquiera que sea su género, forma de expresión, mérito o finalidad. Los derechos reconocidos por el presente Título son independientes de la propiedad del objeto material en el cual está incorporada la obra y su goce o ejercicio no están supeditados al requisito del registro o al cumplimiento de cualquier otra formalidad.

  Las obras protegidas comprenden, entre otras:

  Proyectos, planos, maquetas y diseños de obras arquitectónicas y de ingeniería;"

Dentro de la categoría citada, está el prototipo que se pone a consideración en el presente trabajo, esto para que cualquier uso del presente trabajo con fines de lucro tenga en cuenta al autor.

- Se utiliza software bajo la Licencia General Pública GNU ver. 3; y se pone a disposición de cualquier institución o particular con fines puramente educativos. Esta licencia puede ser encontrada en http://www.gnu.org/copyleft/lgpl.html

## 2.3 Categorías Fundamentales

### 2.3.1 Gráficos de Inclusión Relacionados

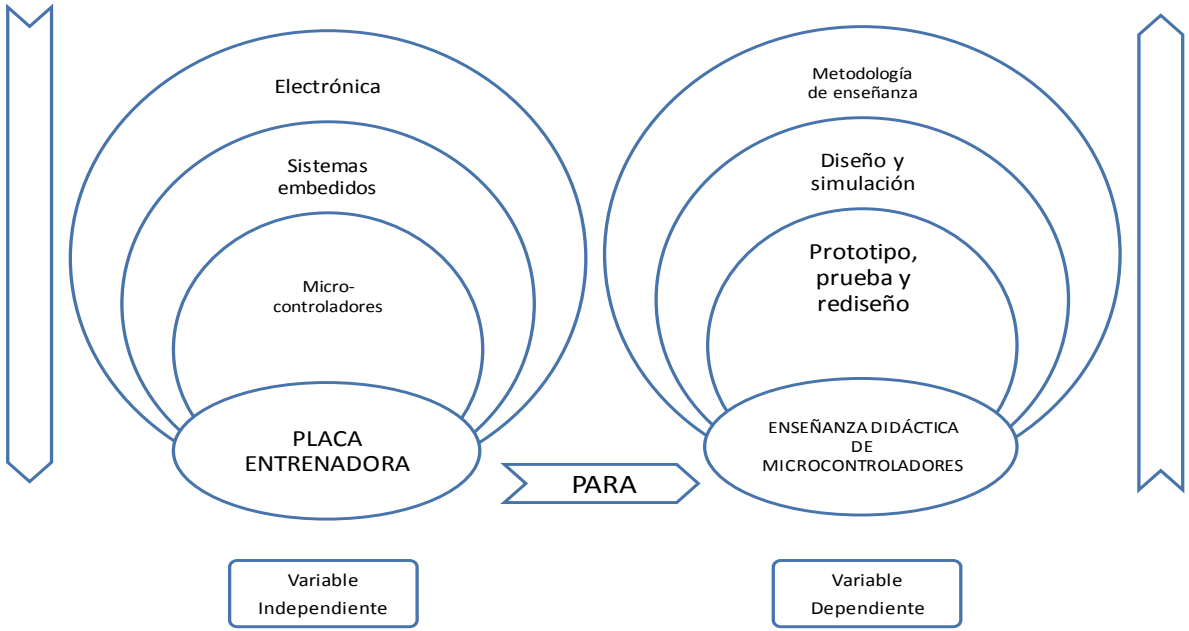



**Figura 2.1.** Gráficos de inclusión relacionados de la variable independiente y dependiente.

Elaborado por: El investigador.

### 2.3.2 Constelación de Ideas

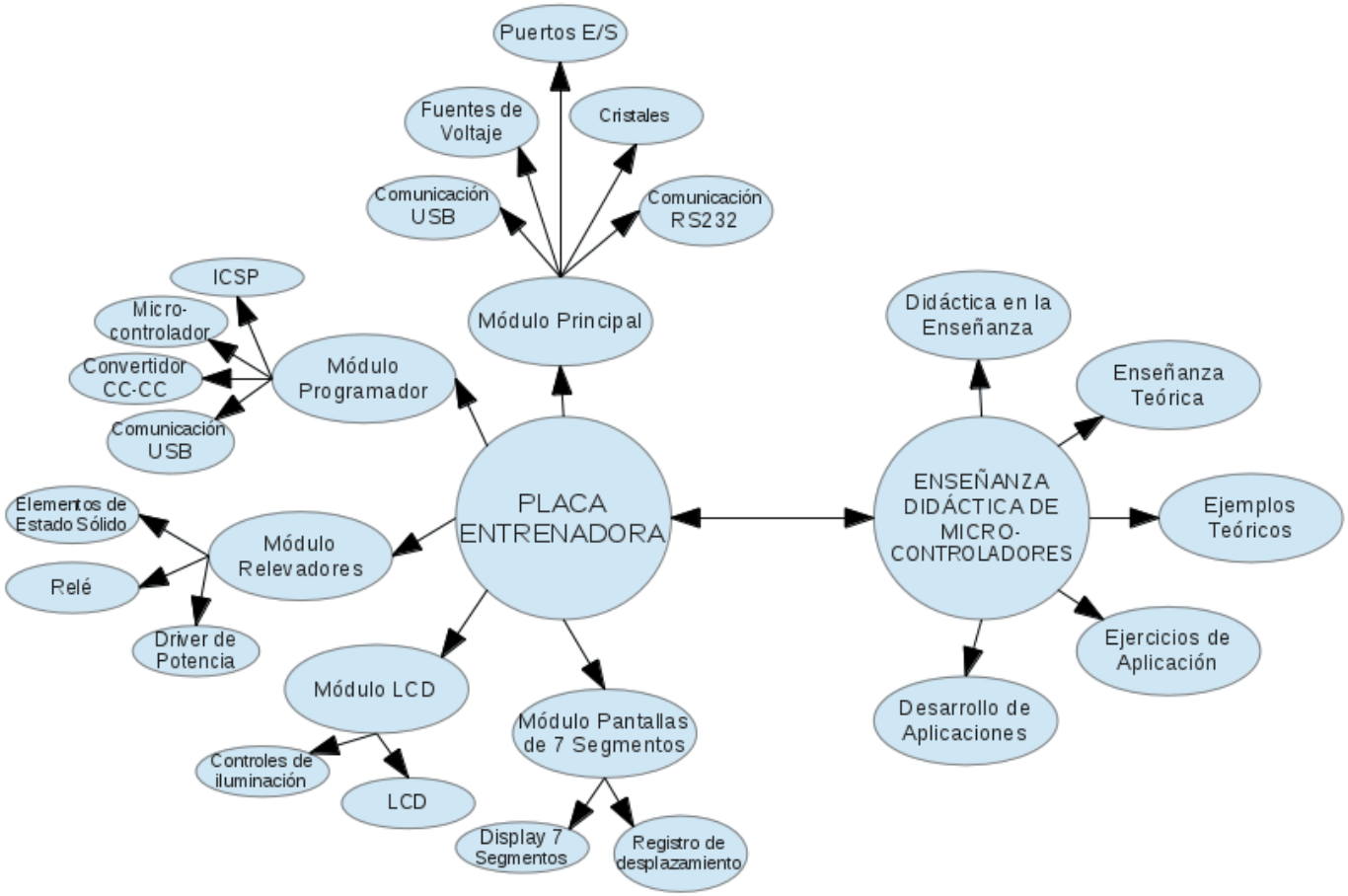



**Figura 2.2.** Constelación de ideas.
Elaborado por: El investigador.

### 2.3.3 Visión Dialéctica de Conceptualizaciones

### 2.3.4 Marco Conceptual de la Variable Independiente

**ELECTRÓNICA**

**Definición y breve historia**

La Electrónica es la ciencia que estudia la interrelación existente entre las cargas eléctricas presentes en los átomos de diferentes materiales, su comportamiento, y la forma de aprovecharlo para beneficio de la humanidad.

Ha tenido una larga evolución, desde la utilización de animales con fines terapéuticos en la Grecia antigua; hasta los experimentos e investigaciones de Ampere, Faraday, Gauss y Ohm por mencionar a algunos, que tuvieron la finalidad de aprovechar de manera práctica los efectos producidos por la circulación de las pequeñas cargas, en los cuales se van descubriendo las diferentes propiedades de los circuitos, como la resistencia, la corriente eléctrica, la inducción, entre otras.

Según escribe Ruiz Robredo, (2001, pág. III) en su libro "Electrónica Básica Para Ingenieros", el verdadero desarrollo comenzó cuando, en 1893 Maxwell reunió los trabajos de varios de estos científicos y postuló las famosas ecuaciones sobre el electromagnetismo, que aún hoy son usadas.

Después de estos tempranos experimentos, se desarrollan diversos componentes en base a las propiedades descubiertas, y se les van dando cierta "utilidad", como por ejemplo a las resistencias de tiza, que se usan para limitar el paso de electrones.

**Circuitos electrónicos**

La definición más simple y acertada, la tenemos en la página electrónica del Premio Nobel, (Internet; 2003,21,01; 2012,12,18; http://www.nobelprize.org/educational/physics/integrated_circuit/history/inde

x.html), “Un circuito electrónico está hecho de diferentes componentes electrónicos, como transistores, resistores, capacitores y diodos, conectados unos con otros de diferentes maneras”.

En una mesa de diseño o en un dispositivo final, se puede encontrar varios ejemplos de aplicación de circuitos electrónicos. Ya sea un medidor de corriente, una simple fuente lineal o un complejo sistema de control, siempre tendrán uno o varios de los componentes anteriormente mencionados. Tienen una funcionalidad determinada, y su diseño corresponderá a las propiedades y limitaciones de los materiales de los cuales está hecho.

**Circuito integrado**

El circuito integrado no es más que un circuito electrónico muy avanzado y miniaturizado**.** Boylestad & Nashelsky, (1998,pág 1), en su libro “Electrónica: Teoría de circuitos y dispositivos electrónicos”, hacen una breve referencia de las dimensiones de los componentes de un circuito electrónico: “Los circuitos integrados de hoy, cuentan con más de 10 millones de transistores en un área no mayor que un pulgar”. Entonces, un circuito integrado es un circuito electrónico que contiene, en escala muy pequeña, los correspondientes componentes eléctricos de un determinado sistema.

Ruiz Robredo (2001, p. IV) escribe: “En 1958 (Jack) Kilby, poco después de incorporarse a la Texas Instrument, concibió la idea de un monolítico, es decir, construir un circuito completo en germanio o silicio. El primer circuito integrado fue un oscilador por rotación de fase que se construyó empleando como material base el germanio, y sobre él, se formaban resistencias, condensadores y transistores, utilizando cables de oro para unir estos componentes.”. En la figura 2.3, se puede apreciar una fotografía del primer circuito integrado de la historia.

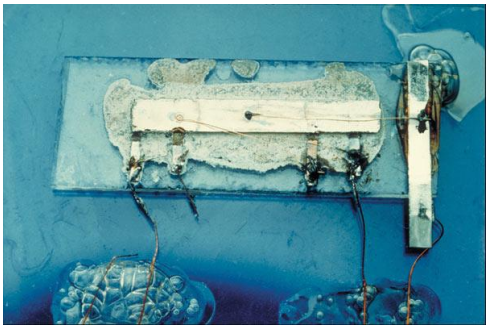


**Figura 2.3.** Primer circuito integrado.
Fuente: Texas Instruments Incorporated

## SISTEMAS EMBEBIDOS

### Definición de sistema embebido

Heat (2003, pág 2), en su libro "*Embedded Systems Desing"*, explica: "un sistema embebido, es un sistema basado en microprocesadores, que está construido para controlar una función o un rango de funciones, pero no está diseñado para ser programado por el usuario final, de la misma manera que una computadora personal". Un sistema embebido también está "inmerso" en el dispositivo o proceso a controlar, es una parte integral de todo el conjunto que lleva a cabo un trabajo programado.

Algo importante es notar que no solo se pueden construir con microcontroladores, ya que existe una amplia variedad de microsistemas capaces de procesar información que pueden ser utilizados, como por ejemplo los FPGA.

Tomando la información que nos presentan, Barrett & Pack (2006), "*Microcontrollers Fundamentals for Engineers and Scientists",* Capítulo 3: *"Microcontroller"* y Heat (2003), *"Embedded Systems Desing",* sección 1: *"What is an embedded system?",* se hace un resumen sobre los sistemas embedidos:

**Características de los sistemas embebidos**

Las principales características de los sistemas embebidos son:

- Reemplazan a los circuitos de lógica discreta. El microprocesador de un sistema embebido se puede programar para que haga las veces de un determinado circuito u otro, sin cambiar de forma radical el diseño electrónico.
- Pueden ser actualizados; al mismo sistema embebido, se le puede programar para que haga tareas más avanzadas, con más eficiencia o corrigiendo posibles fallos.
- Mejora el rendimiento mecánico de otros dispositivos. Al reducir o controlar el número de tareas que realiza un determinado sistema.
- Son confiables, eficientes y abaratan costos.

**Ventajas de los sistemas embebidos**

Entre las ventajas de los sistemas embebidos se puede mencionar:

- Son flexibles y su actualización puede hacerse en poco tiempo.
- Una solución a medida, muchas veces resulta más económica que un sistema comercial completo.
- Tienen un gran poder de procesamiento.
- Son escalables y configurables.

**Desventajas de los sistemas embebidos**

Los sistemas embebidos, para algunas de sus aplicaciones pueden tener ciertas desventajas:

- Requiere la intervención de un equipo especializado para ponerlo en marcha.
- El usuario final siempre dependerá del equipo de desarrollo para añadir funcionalidades o corregir mal funcionamiento.

### Aplicaciones de los sistemas embebidos

Entre las aplicaciones de los sistemas embebidos cabe mencionar algunas categorías y sus usos:

- Industriales, para automatizar, controlar y vigilar procesos.
- Comerciales, al poderse desarrollar y vender soluciones con sistemas embebidos basados en microcontroladores.
- Científicas, al usarse cualquier sistema embebido en investigación.
- Educativas, no solo de estudiantes de electrónica, sino también para cualquier asignatura que maneje procesos.

## MICROCONTROLADORES

### Definición de microcontrolador

Según (Ibrahim, 2008), "Un microcontrolador es una computadora en un chip", tiene todos los componentes necesarios para procesar información. A g*rosso modo* se puede decir que es un sistema completo, que tiene módulos de entrada y salida (PORT A, B S y T en la figura 2.4), diferentes registros, convertidores, un CPU, memoria y algunos circuitos electrónicos con diferente funcionalidad. En dicho sistema, pueden programarse varias tareas, que satisfagan diferentes necesidades.

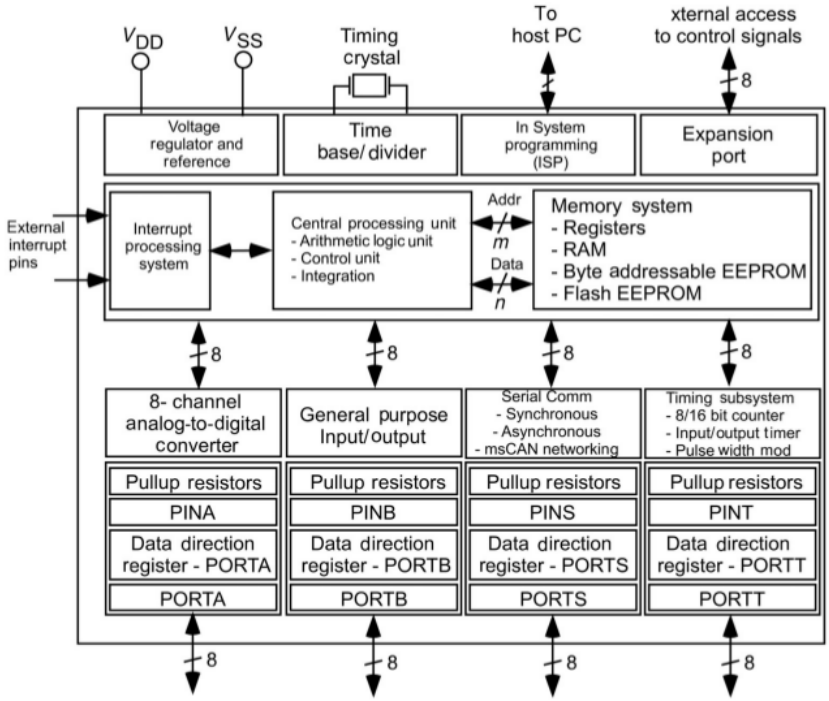



**Figura 2.4.** Diagrama de bloques de un microcontrolador.
Fuente: Microcontrollers Fundamentals for Engineers and Scientists, página 30

De los trabajos de Ibrahim (2008) *"Advanced Pic Microcontroller Projects in C",* sección 1.2: *"Microcontroller Systems"*; y de Barrett & Pack (2006), "*Microcontrollers Fundamentals for Engineers and Scientists",* Capítulo 3: *"Microcontroller",* se ha construido una lista con las principales características, ventajas, desventajas y aplicaciones de los microcontroladores.

### Características de los microcontroladores

Las características de los microcontroladores, varían de un fabricante a otro, sin embargo, se mencionan las siguientes, que pueden estar o no presentes en cierto microcontrolador.

- Fuente de alimentación, la mayoría requiere ser alimentado con un voltaje lógico estándar de 5 V, puede variar desde 2 a 7 V en algunos modelos.

- El oscilador, o reloj. Todo microcontrolador necesita uno para funcionar. Generalmente es un cristal con dos capacitores, o un resonador. Algunos modelos tienen osciladores internos.
- Temporizadores (Timers). Son básicamente contadores que se inicializan por eventos (cambio de estado en una entrada por ejemplo) y desencadenan cierto bloque de programación.
- Perro guardián (Watchdog). Es un tipo especial de contador. Sigue la cuenta hasta que el programa de usuario lo pone a cero. Si esto no ocurre y el contador llega a un valor máximo, el micro se resetea.
- Entrada de reset. Resetea el micro externamente, mediante un cambio lógico en una entrada determinada.
- Interrupciones. Una interrupción hace que el micro responda a un cambio externo de manera casi inmediata, sin importar en que parte del programa se encuentre (excepto en los retardos). Hace que se salte a una parte específica del programa que se denomina *rutina de interrupción.*
- Detectores de bajo voltaje. Resetean o apagan el micro si el voltaje cae bajo un umbral determinado.
- Convertidores análogo – digital. Como su nombre lo indica, se usan para convertir una señal analógica (como el voltaje) en una señal digital.
- Entrada/salida serial. Compatible con RS232. Permite al micro comunicarse con otros dispositivos vía comunicación serial. Algunos modelos incorporan también comunicación SPI (serial peripheral interface) o I2C (integrated interconnect)
- Memoria de datos EEPROM. Es una memoria no volátil que puede cambiar su contenido según la conveniencia del programa.
- Modo de reposo. Es una instrucción especial, que cuando se ejecuta detiene el oscilador interno y por ende cualquier ejecución de instrucciones. Se usa para ahorrar batería.
- Operación de bajo consumo. Especial para aplicaciones portables, que dependen de baterías.

- Interface USB. Interface que permite al micro comunicarse con dispositivos USB a través de sus pines.
- Tienen muchas otras aplicaciones de acuerdo a la utilidad que se le vaya a dar al sistema embebido.

**Ventajas de los microcontroladores**

Entre las ventajas de los microcontroladores se mencionan las más relevantes:

- Se consiguen fácilmente en el mercado ecuatoriano
- Son fáciles de configurar.
- No requieren demasiados componentes adicionales para las aplicaciones más sencillas.

**Desventajas de los microcontroladores**

Los factores negativos a considerar cuando se trabaja con microcontroladores:

- Su memoria flash es limitada.
- Para aplicaciones complejas, requiere componentes adicionales.

**Aplicaciones de los microcontroladores**

Existen varias aplicaciones de microcontroladores, tanto en la vida cotidiana, como en la industria o en la investigación. Por mencionar algunas de ellas.

En la vida cotidiana:

- Reloj despertador electrónico.
- Controles remotos.
- Llaves de autos.
- Sensores en el auto mismo.
- El radio del auto.

En la industria:

- PLC.
- Controles automáticos.
- Sistemas hidráulicos automatizados.
- Sensores.

En la investigación:

- Sistemas de medición.
- Prototipos de prueba.

**Arquitecturas de microcontroladores**

Ibrahim (2008), escribe que existen dos tipos de arquitecturas, que son usadas comúnmente en microcontroladores, como se puede observar en la figura 2.5:

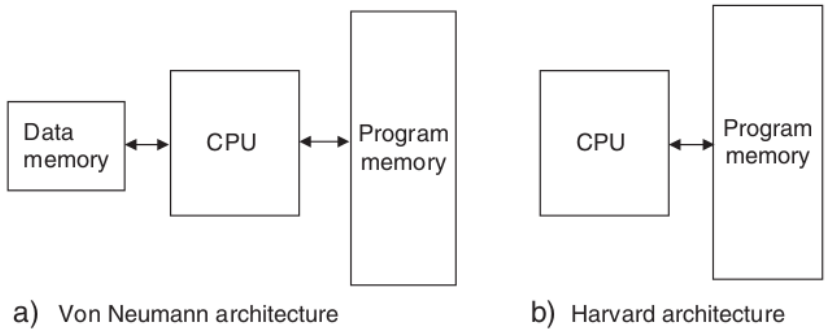



**Figura 2.5.** Diferentes arquitecturas de microcontroladores.
Fuente: Advanced Projects in C, página 13

- Von Neumann, que pone toda la memoria, de programa y de datos en un solo bloque de memoria.
- Harvard, separa los bloques de memoria en un bloque de Programa, que contiene la instrucciones que seguirá el microcontrolador; y un bloque de Datos, que contiene las variables que se van a sobre-escribir con el sistema en funcionamiento. Esto permite un acceso simultáneo a ambos bloques.

**PLACA ENTRENADORA**

Para la investigación; se concluye que una placa entrenadora, también llamada placa de desarrollo, es un circuito impreso que puede ser utilizado con uno o más microcontroladores, conectados apropiadamente a los módulos electrónicos necesarios para utilizarla en el modelamiento de sistemas embebidos.

### 2.3.5 Marco Conceptual de la Variable Dependiente

**METODOLOGÍA DE ENSEÑANZA**

**Introducción**

Comenio, (1998, pág. 8), en su célebre "Didáctica Magna", dice: "El nombre de Erudición comprende el conocimiento de todas las cosas…" y el ingeniero no tiene que aspirar sino a la erudición en los temas referentes a su carrera. El primer paso para la enseñanza de asignaturas técnicas en ingeniería, es el tratamiento de la parte teórica. Esto pudiera parecer contradictorio, sin embargo, la formación integral en microcontroladores, requiere el entendimiento profundo de los procesos que se llevan a cabo dentro de estos circuitos.

Se tratan aspectos desde la electrónica básica, con transistores y resistencias dentro del encapsulado, pasando por arquitecturas, pilas de datos, hasta la manera en que el microcontrolador procesa las instrucciones de una tarea en él programada.

Después, se lleva a cabo la selección de un lenguaje de programación, de la infinidad que existen. Por citar algunos: Ensamblador, Basic, C, JalV2. Cabe mencionar que cualquier lenguaje que se utilice, se deberá compilar adecuadamente, y cargar (o grabar) el hexadecimal resultante en el dispositivo.

**Prototipo**

En diseño electrónico, especialmente cuando trabajamos sobre sistemas embebidos, se convierte en mandatorio el uso de prototipos. Estos se pueden definir como una construcción preliminar, con todas o algunas de las funcionalidades del sistema proyectado. Se usan para comprobar el comportamiento real del circuito, y hacer los ajustes necesarios para su funcionamiento óptimo.

**Diseño**

Hacer un diseño electrónico, es el proceso de establecer las necesidades específicas de un determinado sistema, valorar las posibles soluciones hasta encontrar una que satisfaga las necesidades a un precio razonable, hacer pruebas de concepto y funcionamiento, corregir los posibles fallos en un prototipo y proponer la construcción de una solución eficiente.

**Encapsulados**

Existen varios tipos de encapsulados, difieren tanto por el número de pines, como por su forma de montaje en el circuito, en la práctica, en la FISEI, los que se han utilizados, a lo largo de la Carrera de Electrónica, son:

- *Dual in-line package* (DIP), en la figura 2.6, consiste en dos hileras de pines, que se montan el circuito con el sistema *through hole*, es decir haciendo agujeros en la placa y soldando los componentes al lado contrario del encapsulado.

**Figura 2.6.** Encapsulado DIP
Autor: El investigador.

- *Quad Flat Package* (QFP), en la figura 2.7, el cual tiene pines por los cuatro lados y son para montaje superficial.

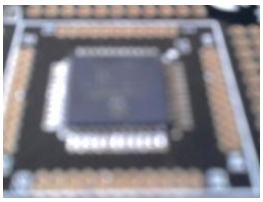


**Figura 2.7.** Encapsulado QFP
Autor: El investigador.

- *El pin grid array* (PGA), que podemos observar en la figura 2.8 consiste en arreglos de pines que se encuentran debajo del encapsulado; generalmente se montan en zócalo, aunque también se pueden soldar directamente en la placa.

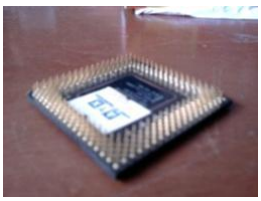


**Figura 2.8.** Encapsulado PGA
Autor: El investigador.

## DISEÑO Y SIMULACIÓN

De acuerdo a la experiencia del investigador en el diseño de prototipos, se sabe que una vez comprendido el proceso que ha de seguir la información dentro de un microcontrolador, y establecido el cómo se desarrollarán los proyectos, se procederá con el aprovechamiento de los recursos. Se hará un diseño en papel, tanto del software como del hardware. Se escoge un lenguaje de programación apropiado. Se traduce el algoritmo a este lenguaje. Se calculan las magnitudes eléctricas, se establecen los elementos necesarios y se diseña el circuito. Se utilizan diferentes programas de modelamiento de microcontroladores, como la suite de diseño Proteus, para simular el comportamiento del programa y el circuito. Se hacen los ajustes necesarios hasta que el modelo satisfaga los requerimientos de diseño.

Es bien conocido en el mundo de la programación el ejemplo "Hola mundo". Que proponen Kernighan & Hill, (1979, pág. 10), en su conocido *"A Tutorial Introduction To The Language B"*, muestran como inicializar dos variables con cadenas de caracteres.

"Las variables externas pueden ser inicializadas con valores de cadenas de caracteres simplemente poniendo a continuación de sus nombres los caracteres, exactamente como para otras constantes. Esas definiciones inicializan a y b:

a "hello"; b "world'*; "

Para un microcontrolador el programa Hello Word es, por ser simple y completo, es el parpadeo de un led.

## PROTOTIPO, PRUEBA Y REDISEÑO

Ibrahim, (2008, pág. 2), menciona que "Básicamente, una microcomputadora ejecuta un programa de usuario que tiene cargado en su memoria de programa. Bajo el control de este programa, los datos son recibidos desde dispositivos externos (entradas), manipulados, y enviados a dispositivos externos (salidas)". Entonces, los sistemas con microcontroladores tienen como objetivo final ser implementados. Pero antes de ponerlos en producción, deben pasar por un proceso de diseño, evaluación y pruebas.

Solamente después de asegurar que el modelo que se ha diseñado cumple con las expectativas se lo transfiere a un circuito de prueba, a un hardware. Puede ser uno específicamente construido para la aplicación. Aquí entran en juego las placas entrenadoras, que facilitan el proceso, al permitir reutilizar la placa en varias ocasiones, haciendo cambios mínimos para que cumpla cualquier funcionalidad. En ese punto, se registrará cualquier eventualidad, se solucionará, y se repetirá el proceso de diseño hasta que el modelo satisfaga los requerimientos de desempeño y confiabilidad.

**ENSEÑANZA DIDÁCTICA DE MICROCONTROLADORES**

Para la presente investigación se concluye que, la enseñanza didáctica de microcontroladores es la enseñanza teórico-práctica del módulo, usando material didáctico de manera apropiada.

## 2.4 Hipótesis

¿Cómo incide el no tener una placa entrenadora de microcontroladores, en la enseñanza didáctica de microcontroladores en la FISEI?

## 2.5 Señalamiento de las Variables de la Hipótesis

### 2.5.1 Dependiente

Enseñanza didáctica de microcontroladores

### 2.5.2 Independiente

Placa entrenadora

# CAPÍTULO III

# METODOLOGÍA

## 3.1 Enfoque

Cuali-cuantitativo, por cuanto se puso énfasis en los procesos de enseñanza que se usan para la Cátedra de Microcontroladores en la FISEI, se estudió el caso desde dentro de su contexto, sin embargo el objetivo final fue buscar la comprensión de los factores que envuelven dicho proceso de enseñanza.

## 3.2 Modalidad Básica de la Investigación

### 3.2.1 De Campo

Al desarrollar esta investigación en base a las necesidades de un caso específico, se definió como una investigación de campo, la cual llevó a cabo un estudio sistemático de la enseñanza de Microcontroladores en los Laboratorios de Electrónica de la FISEI.

### 3.2.2 Bibliográfica – Documental

El presente trabajo tuvo una modalidad de investigación evidentemente bibliográfica – documental, debido a que se utilizaron libros, publicaciones y documentos referentes a la enseñanza y al desarrollo de proyectos con Microcontroladores.

### 3.2.3 Experimental

Se manipuló una variable independiente, con el objeto de establecer el efecto que tiene sobre una variable dependiente, se estableció la relación fundamental entre la inexistencia de una placa entrenadora de microcontroladores y la enseñanza didáctica de la materia.

### 3.2.4 Aplicada

Se resolvió un problema de la vida real, al brindar una alternativa didáctica para impartir la cátedra de Microcontroladores en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

## 3.3 Niveles o Tipos de Investigación

### 3.3.1 Exploratorio

Se exploró el problema que supone el impartir la cátedra de Microcontroladores en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, sin contar con un material didáctico apropiado y suficiente, con el objetivo de desarrollar una alternativa más eficiente.

### 3.3.2 Descriptivo

La presente investigación describió el fenómeno social que implica la enseñanza de un módulo avanzado, sin contar con el material didáctico apropiado, en la FISEI; con el objetivo de estimar los parámetros que intervienen en la enseñanza didáctica de microcontroladores.

### 3.3.3 Correlacional

Por cuanto se estableció la dependencia de la enseñanza didáctica de microcontroladores, con el material didáctico apropiado para la impartición de la cátedra.

### 3.3.4 Explicativo

Se expuso el comportamiento de una variable dependiente, como la enseñanza didáctica de microcontroladores, en función de una placa entrenadora, que representa a una variable independiente.

## 3.4 Población y Muestra

Se realizó una entrevista al docente de la cátedra de Microcontroladores, debido a que es él quien determina los contenidos y la manera de exponerlos

a los alumnos. Además, se aplicó la ficha de observación en la administración de los laboratorios, en donde se almacenan los equipos de la Facultad.

Se aplicó la encuesta a 64 estudiantes de Séptimo, Octavo y Noveno semestres de la Carrera de Ingeniería en Electrónica y Comunicaciones, que representan a la totalidad de alumnos que asisten normalmente a clases, y que han asistido o se encuentran asistiendo a la cátedra de Microcontroladores.

## 3.5 Operacionalización de Variables

**VARIABLE INDEPENDIENTE:** PLACA ENTRENADORA.

**Tabla 3.1** Operacionalización de la variable independiente.

| CONCEPTUALIZACIÓN | CATEGORÍAS | INDICADORES | ÍTEMS | TÉCNICAS E INSTRUMENTOS |
|---|---|---|---|---|
| **Placa de circuito impreso** que contiene uno o más **microcontroladores**, conectados apropiadamente a los **módulos electrónicos** necesarios para utilizarla en el modelamiento de **sistemas embebidos.** | 1. Placa de circuito impreso | 1. Circuito Electrónico | ¿Es capaz usted de definir lo que es un circuito electrónico?<br>( ) Si. ( ) No. ( ) Tal vez.<br><br>¿Puede definir de manera precisa lo que es una placa de circuito impreso?<br>( ) Si. ( ) No. ( ) Tal vez. | 1. Encuesta con un cuestionario dirigido a los alumnos. |
| | 2. Microcontroladores | 2. Microcomputadora | ¿Sabe lo que es y para qué sirve una micro-computadora?<br>( ) Si. ( ) No. ( ) Tal vez.<br><br>Sabe usted lo que es un microcontrolador? Tenga en cuenta que todo PIC es un microcontrolador, pero no todo microcontrolador es un PIC.<br>( ) Si. ( ) No. ( ) Tal vez. | 2. Encuesta con un cuestionario dirigido a los alumnos. |

| | | | | |
|---|---|---|---|---|
| | 3. Módulos electrónicos<br><br>4. Sistemas embebidos | 3. Componentes electrónicos.<br><br>4. Aplicaciones electrónicas | ¿Cuántas aplicaciones con microcontroladores ha implementado usted? Sea como parte de una práctica o fuera de la Universidad.<br>( ) Ninguna ( ) De 0 a 2<br>( ) De 3 a 5 ( ) Más de 5<br><br>¿Tiene claramente definido qué es un componente electrónico?<br>( ) Si. ( ) No. ( ) Tal vez.<br><br>¿Para qué sirven los componentes electrónicos?<br>a)( ) Para construir prototipos.<br>b)( ) Para construir aplicaciones<br>c)( ) Ambas.<br><br>¿Puede usted definir claramente qué es una aplicación electrónica?<br>( ) Si. ( ) No. ( ) Tal vez.<br><br>¿Puede usted definir claramente qué es un sistema embebido?<br>( ) Si. ( ) No. ( ) Tal vez.<br><br>¿Cuántas soluciones con sistemas embebidos ha implementado usted? Sea como parte de una práctica o fuera de la Universidad.<br>( ) Ninguna ( ) De 0 a 2<br>( ) De 3 a 5 ( ) Más de 5 | 3. Encuesta con un cuestionario dirigido a los alumnos.<br><br>4. Encuesta con un cuestionario dirigido a los alumnos. |

Elaborado por: El investigador.

**VARIABLE DEPENDIENTE:** ENSEÑANZA DIDÁCTICA DE MICROCONTROLADORES.

**Tabla 3.2** Operacionalización de la variable dependiente.

| CONCEPTUALIZACIÓN | CATEGORÍAS | INDICADORES | ÍTEMS | TÉCNICAS E INSTRUMENTOS |
|---|---|---|---|---|
| **Enseñanza teórico práctica** del **módulo de microcontroladores**, usando **material didáctico** de manera apropiada. | 1. Enseñanza teórico práctica<br>2. Módulo de microcontroladores<br>3. Material didáctico | 1. Teoría, ejemplos, laboratorios.<br>2. Contenidos que se desarrollan con los estudiantes.<br>3. Interfaces y equipos. | 1. ¿Cómo se imparte teoría de microcontroladores? ¿Qué ejemplos se usan? ¿Cómo se planifican los laboratorios?<br>2. ¿Qué contenidos se desarrollan?<br>3. ¿Qué equipos se encuentran disponibles en el laboratorio? | 1. Entrevista a través de cédula de entrevista al docente.<br>2. Entrevista a través de cédula de entrevista al docente.<br>3. Observación con una ficha, a los laboratorios. |

Elaborado por: El investigador.

## 3.6 Recolección de Información

Se realizó una entrevista con el docente del módulo de microcontroladores.

Para el levantamiento de datos sobre los equipos de entrenamiento de microcontroladores existentes en los laboratorios, se utilizó una ficha de observación.

Para determinar la incidencia de la variable independiente en los estudiantes, se realizó una encuesta referente al módulo de Microcontroladores y el material didáctico.

## 3.7 Procesamiento, Análisis e Interpretación

Una vez aplicados los instrumentos, se procedió al análisis estadístico y a la tabulación de datos correspondiente. Una vez que se tuvieron los datos tabulados, se procedió al análisis, basándose en los criterios desprendidos del marco teórico, y los objetivos de la presente investigación.

# CAPÍTULO IV

# ANÁLISIS E INTERPRETACIÓN DE LOS RESULTADOS

## 4.1 Tabulación de la Encuesta

Se ha planteado una encuesta con el objetivo de detectar la incidencia del material didáctico utilizado en el módulo de microcontroladores, en la enseñanza didáctica del mismo. En el Anexo A se muestra detalladamente la misma.

Para la construcción de los cuadros y gráficos, se ha utilizado la herramienta Microsoft Office Excel 2010, por ser un programa ofimático altamente difundido, y por contar con las herramientas necesarias para una clara representación de la información necesaria en el procesamiento de los datos de la encuesta.

En cada uno de los incisos se realiza el respectivo gráfico y el análisis que permiten comprender la problemática que expone el presente trabajo de investigación.

La población a la cual se aplicó la encuesta, es de 64 personas, que representan a la totalidad de la población estudiantil de los Séptimo, Octavo y Noveno Ciclos Académicos de la Carrera de Ingeniería en Electrónica y Comunicaciones de la FISEI de la UTA, en el período Marzo – Agosto de 2013, presentes en la primera semana del período académico.

Además, la presente investigación se complementa con una Entrevista parcialmente estructurada, que se puede apreciar en su totalidad en el Anexo B, con el Ing. Luis Pomaquero, docente de la cátedra de Microcontroladores en el período antes mencionado, para recoger sus criterios sobre cuestiones específicas acerca de la planificación y ejecución de las clases.

Finalmente, con una ficha de observación, incluida en el Anexo C, aplicada en la Oficina de Control de Laboratorios de Electrónica e Industrial, se establecerá la existencia o inexistencia de material didáctico específico para la cátedra de Microcontroladores.

## 4.2 Análisis de los Resultados

### 4.2.1 Interpretación de Datos de la Encuesta

**Pregunta 1.** ¿Es capaz usted de definir lo que es un circuito electrónico?

**( ) Si. ( ) No. ( ) Tal vez.**

**Tabla 4.1** Tabulación pregunta 1. ¿Es capaz usted de definir lo que es un circuito electrónico?

| Alternativa | Frecuencia (personas encuestadas) | Porcentaje |
|---|---|---|
| Si | 55 | 85,94% |
| No | 2 | 3,13% |
| Tal vez | 7 | 10,94% |
| **Total** | 64 | 100,00% |

Elaborado por: El investigador.

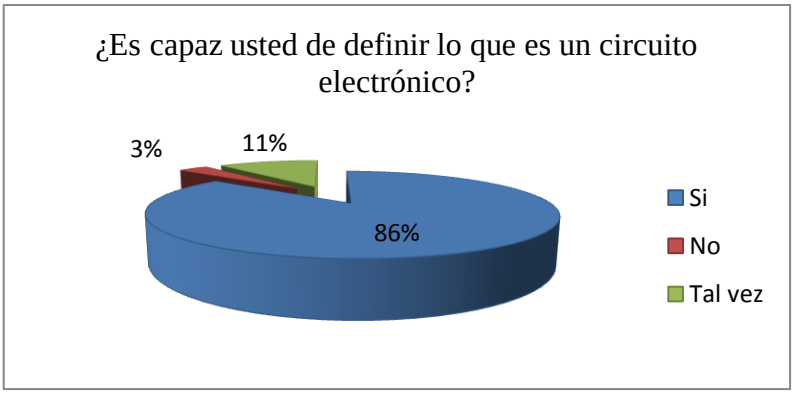



**Figura 4.1** Análisis gráfico pregunta 1. ¿Es capaz usted de definir lo que es un circuito electrónico?
Fuente: Encuesta realizada.
Elaborado por: El investigador.

**Interpretación**

El 85,94% de los encuestados indican que efectivamente pueden definir el significado de circuito electrónico, mientras que el 3,13% de ellos manifiesta

que no puede hacerlo y el 10,94% señala que tal vez puede dar una definición acertada.

**Análisis**

El conocimiento de definiciones simples de aplicaciones electrónicas es alto, pues la mayoría de estudiantes afirma tener en claro lo que es un circuito electrónico. Sin embargo, en los niveles finales de la Carrera de Electrónica y Comunicaciones, aún existen varios aspirantes a la Ingeniería que no sienten tener el dominio suficiente para definir con seguridad un concepto básico.

**Pregunta 2.** ¿Puede definir de manera precisa lo que es una placa de circuito impreso?

**( ) Si. ( ) No. ( ) Tal vez.**

**Tabla 4.2** Tabulación pregunta 2. ¿Puede definir de manera precisa lo que es una placa de circuito impreso?

| Alternativa | Frecuencia (personas encuestadas) | Porcentaje |
|---|---|---|
| Si | 46 | 71,88% |
| No | 0 | 0,00% |
| Tal vez | 18 | 28,13% |
| **Total** | 64 | 100,00% |

Elaborado por: El investigador.

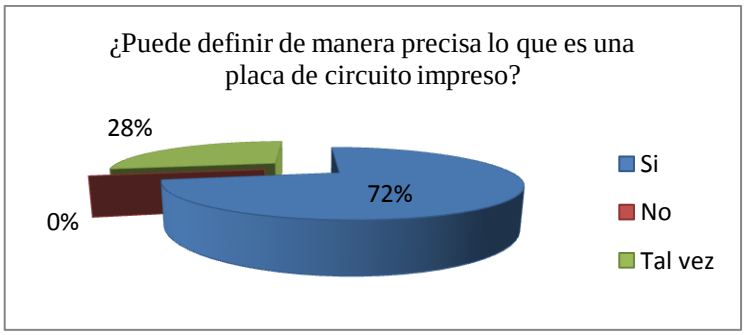



**Figura 4.2** Análisis gráfico pregunta 2. ¿Puede definir de manera precisa lo que es una placa de circuito impreso?
Fuente: Encuesta realizada.
Elaborado por: El investigador.

**Interpretación**

En la segunda pregunta, el 71,88 % de los encuestados indican que dado el caso, pueden definir de manera precisa lo que es una placa de circuito impreso, ninguno de los estudiantes afirma no poder hacerlo, y el 28,13 % tal vez puede dar una definición.

**Análisis**

Al final de la carrera, aún existe cierto temor a hacer aseveraciones acerca de definiciones básicas sobre electrónica. Un porcentaje cercano a la tercera parte de los encuestados tiene dudas al dar una definición de lo que es una placa de circuito impreso, lo cual hace suponer que la formación teórica no se ha visto reflejada en la práctica, especialmente en cuanto al diseño electrónico se refiere.

**Pregunta 3.** ¿Sabe lo que es y para qué sirve una micro-computadora?

**( ) Si. ( ) No. ( ) Tal vez.**

**Tabla 4.3** Tabulación pregunta 3. ¿Sabe lo que es y para qué sirve una micro-computadora?

| Alternativa | Frecuencia (personas encuestadas) | Porcentaje |
|---|---|---|
| Si | 37 | 57,81% |
| No | 7 | 10,94% |
| Tal vez | 20 | 31,25% |
| **Total** | 64 | 100,00% |

Elaborado por: El investigador.

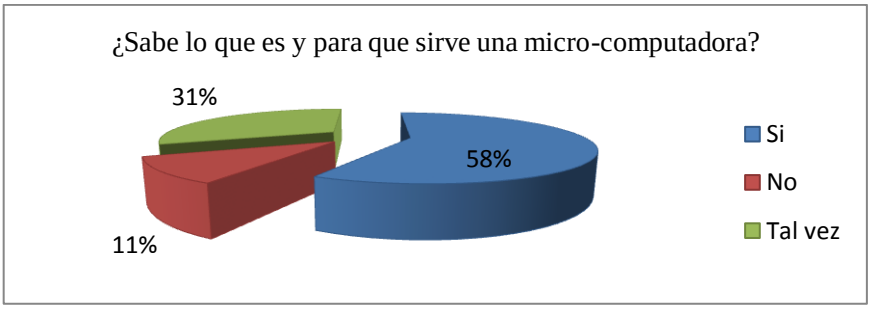



**Figura 4.3** Análisis gráfico pregunta 3. ¿Sabe lo que es y para qué sirve una micro-computadora?
Fuente: Encuesta realizada.
Elaborado por: El investigador.

**Interpretación**

Al plantear la pregunta, el 57,81% de los encuestados afirman saber a qué se refiere el término, entretanto el 10,94% de ellos manifiesta no tener un concepto de ello; finalmente, el 31,25% señala que tal vez tiene idea de lo que es una microcomputadora.

**Análisis**

Más de la mitad de estudiantes afirma saber con certeza lo que es una microcomputadora y las potenciales aplicaciones que puede tener, sin embargo, la otra mitad de estudiantes de últimos niveles de ingeniería, no sabe o no puede afirmar con seguridad lo que representa el término, aun cuando sus aplicaciones en comunicaciones son indispensables para la carrera que cursan.

**Pregunta 4.** ¿Sabe usted lo que es un microcontrolador? Tenga en cuenta que todo PIC es un microcontrolador, pero no todo microcontrolador es un PIC.

**( ) Si. ( ) No.**

**Tabla 4.4** Tabulación pregunta 4. ¿Sabe usted lo que es un microcontrolador?

| Alternativa | Frecuencia (personas encuestadas) | Porcentaje |
|---|---|---|
| Si | 47 | 73,44% |
| No | 17 | 26,56% |
| **Total** | 64 | 100,00% |

Elaborado por: El investigador.

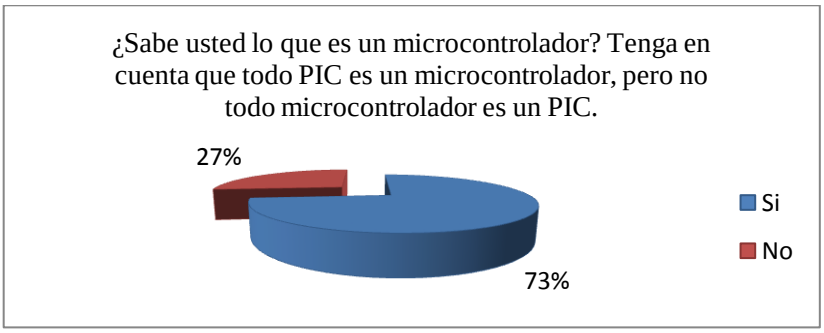



**Figura 4.4** Análisis gráfico pregunta 4. ¿Sabe usted lo que es un microcontrolador?
Fuente: Encuesta realizada.
Elaborado por: El investigador.

### Interpretación

A la pregunta: ¿Sabe usted lo que es un microcontrolador? el 73,44% de los encuestados responden positivamente, mientras que el 26,56% lo hace de manera negativa.

### Análisis

Un elevado número de estudiantes tiene una noción de lo que es un microcontrolador, sin embargo, una cuarta parte de la población estudiantil no está familiarizada con ello, a pesar de contar con una cátedra específica de microcontroladores.

**Pregunta 5.** ¿Cuántas aplicaciones con microcontroladores ha implementado usted? Sea como parte de una práctica o fuera de la Universidad.

**( ) Ninguna ( ) De 0 a 2 ( ) De 3 a 5 ( ) Más de 5**

**Tabla 4.5** Tabulación pregunta 5. ¿Cuántas aplicaciones con microcontroladores ha implementado usted?

| Alternativa | Frecuencia (personas encuestadas) | Porcentaje |
|---|---|---|
| Ninguna | 3 | 4,69% |
| De 0 a 2 | 15 | 23,44% |
| De 3 a 5 | 24 | 37,50% |
| Más de 5 | 22 | 34,38% |
| **Total** | 64 | 100,00% |

Elaborado por: El investigador.

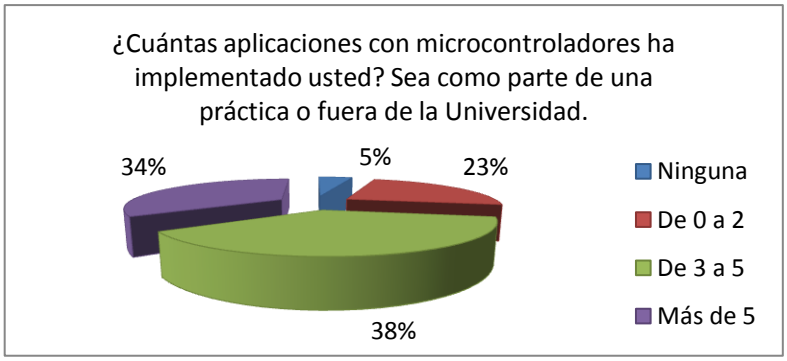



**Figura 4.5** Análisis gráfico Pregunta 5. ¿Cuántas aplicaciones con microcontroladores ha implementado usted?
Fuente: Encuesta realizada.
Elaborado por: El investigador.

### Interpretación

Un 4,69% de los encuestados indican que jamás en su carrera han implementado aplicaciones con microcontroladores, dentro o fuera de la Universidad, un 23,44% ha implementado de 0 a 2 de ellas, mientras que el 37,50% ha implementado de 3 a 5, y un 34,38% ha implementado 5 o más.

### Análisis

Los mismos estudiantes que en las cuatro preguntas anteriores han afirmado no saber o no tener seguridad de los conceptos básicos de electrónica, aseguran haber implementado por lo menos una aplicación con microcontroladores. Esto implica que estas aplicaciones no han sido diseñadas siguiendo principios técnicos, o que no han sido diseñadas por los estudiantes en absoluto; implementando circuitos cuyo comportamiento desconocen, sin prever las posibles consecuencias que ello podría acarrear

para el sistema a controlar o para la seguridad física de las personas que estén cerca de él.

**Pregunta 6.** ¿Tiene claramente definido qué es un componente electrónico?

**( ) Si. ( ) No. ( ) Tal vez.**

**Tabla 4.6** Tabulación pregunta 6. ¿Tiene claramente definido qué es un componente electrónico?

| Alternativa | Frecuencia (personas encuestadas) | Porcentaje |
|---|---|---|
| Si | 54 | 84,38% |
| No | 2 | 3,13% |
| Tal vez | 8 | 12,50% |
| **Total** | 64 | 100,00% |

Elaborado por: El investigador.

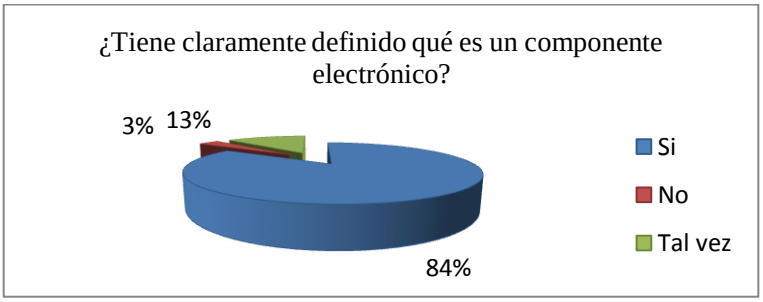



**Figura 4.6** Análisis gráfico pregunta 6. ¿Tiene claramente definido qué es un componente electrónico?
Fuente: Encuesta realizada.
Elaborado por: El investigador.

### Interpretación

El 84,38% de los encuestados afirman que tiene claramente definido el concepto de componente electrónico, mientras que el 3,13% de los estudiantes afirma no poder hacerlo, y el 12,50% tal vez puede dar una definición de ello.

### Análisis

Mientras que la mayoría de estudiantes tiene claro qué es un componente electrónico, existen aún educandos que tienen dudas acerca de este concepto,

lo cual denota una carencia de experticia en el campo práctico de la Electrónica.

**Pregunta 7.** ¿Para qué sirven los componentes electrónicos?

**( ) Para construir prototipos. ( ) Para construir aplicaciones ( ) Ambas.**

**Tabla 4.7** Tabulación pregunta 7. ¿Para qué sirven los componentes electrónicos?

| Alternativa | Frecuencia (personas encuestadas) | Porcentaje |
|---|---|---|
| Para construir prototipos | 15 | 23,44% |
| Para construir aplicaciones | 11 | 17,19% |
| Ambas | 38 | 59,38% |
| **Total** | 64 | 100,00% |

Elaborado por: El investigador.

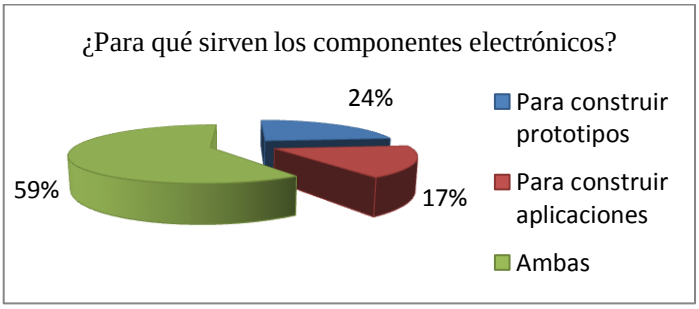



**Figura 4.7** Análisis gráfico Pregunta 7. ¿Para qué sirven los componentes electrónicos?
Fuente: Encuesta realizada.
Elaborado por: El investigador.

### Interpretación

En la pregunta séptima, el 23,44 % de los encuestados considera que los componentes electrónicos sirven para construir prototipos, el 17,19 % de la población de la encuesta considera de sirven para construir aplicaciones, y el 59,38 % considera que sirven para construir ambas.

### Análisis

A pesar de que la mayoría de estudiantes ha afirmado poder definir claramente lo que es un componente electrónico, casi la mitad de los encuestados tiene dudas de su aplicación y utilización real, lo que constituye

un claro indicador de la falta de implementación de soluciones a problemas reales utilizando conceptos y componentes electrónicos.

**Pregunta 8.** ¿Puede usted definir claramente qué es una aplicación electrónica?

**( ) Si. ( ) No. ( ) Tal vez.**

**Tabla 4.8** Tabulación pregunta 8. ¿Puede usted definir claramente qué es una aplicación electrónica?

| Alternativa | Frecuencia (personas encuestadas) | Porcentaje |
|---|---|---|
| Si | 33 | 51,56% |
| No | 9 | 14,06% |
| Tal vez | 22 | 34,38% |
| **Total** | 64 | 100,00% |

Elaborado por: El investigador.

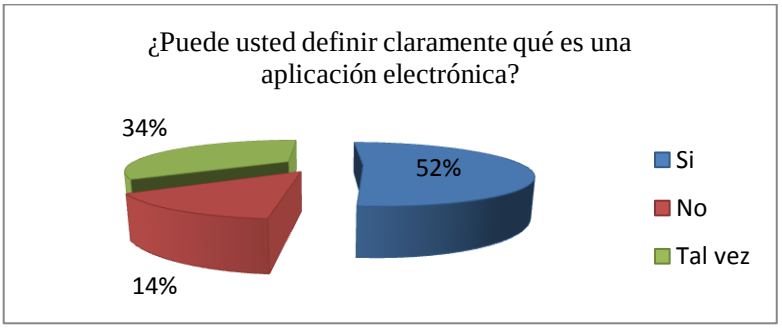



**Figura 4.8** Análisis gráfico pregunta 8. ¿Puede usted definir claramente qué es una aplicación electrónica?
Fuente: Encuesta realizada.
Elaborado por: El investigador.

### Interpretación

El 51,56% de los encuestados consideran que pueden definir claramente qué es una aplicación electrónica, en tanto que el 14,06% ha respondido que no puede hacerlo, y el 34,38% tal vez puede dar una definición.

### Análisis

Un porcentaje aproximado a la mitad de los encuestados no tiene ideas claras acerca de lo que son las aplicaciones electrónicas, acusando la falta de

prácticas con material didáctico adecuado, que les permita ver realmente como se transfiere un concepto práctico en el mundo real.

**Pregunta 9.** ¿Puede usted definir claramente qué es un sistema embebido?

**( ) Si. ( ) No. ( ) Tal vez.**

**Tabla 4.9** Tabulación pregunta 9. ¿Puede usted definir claramente qué es un sistema embebido?

| Alternativa | Frecuencia (personas encuestadas) | Porcentaje |
|---|---|---|
| Si | 6 | 9,38% |
| No | 40 | 62,50% |
| Tal vez | 18 | 28,13% |
| **Total** | 64 | 100,00% |

Elaborado por: El investigador.

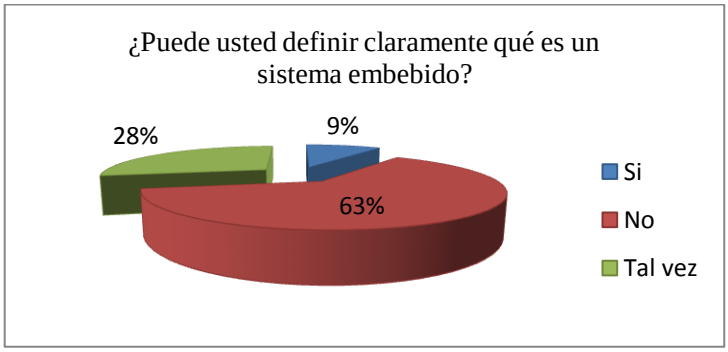



**Figura 4.9** Análisis gráfico pregunta 9. ¿Puede usted definir claramente qué es un sistema embebido?
Fuente: Encuesta realizada.
Elaborado por: El investigador.

### Interpretación

Cuando se ha preguntado si pueden definir claramente qué es un sistema embebido, el 9,38% de los encuestados indican que sí, el 62,50% no pueden definirlo, y el 28,13% considera que tal vez puede hacerlo.

### Análisis

Llegado al caso de la implementación específica de soluciones con microcontroladores, la mayoría de estudiantes no tiene idea de lo que son los

sistemas embebidos, lo cual hace pensar que muy poco, o nunca aplicaron una solución de este tipo.

**Pregunta 10.** ¿Cuántas soluciones con sistemas embebidos ha implementado usted? Sea como parte de una práctica o fuera de la Universidad.

**( ) Ninguna ( ) De 0 a 2 ( ) De 3 a 5 ( ) Más de 5**

**Tabla 4.10** Tabulación pregunta 10. ¿Cuántas soluciones con sistemas embebidos ha implementado usted?

| Alternativa | Frecuencia (personas encuestadas) | Porcentaje |
|---|---|---|
| Ninguna | 48 | 75,00% |
| De 0 a 2 | 13 | 20,31% |
| De 3 a 5 | 3 | 4,69% |
| Más de 5 | 0 | 0,00% |
| **Total** | 64 | 100,00% |

Elaborado por: El investigador.

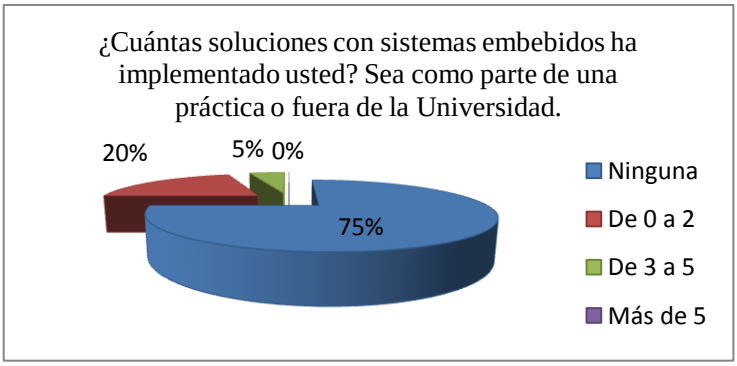



**Figura 4.10** Análisis gráfico pregunta 10. ¿Cuántas soluciones con sistemas embebidos ha implementado usted?
Fuente: Encuesta realizada.
Elaborado por: El investigador.

### Interpretación

Finalmente, a la pregunta, ¿Cuántas soluciones con sistemas embebidos ha implementado usted? Sea como parte de una práctica o fuera de la Universidad, el 75,00% de los encuestados, ha contestado con seguridad que no ha hecho ninguna implementación de este tipo; el 20,31% ha realizado de

0 a 2, el 4,69% de 3 a 5 soluciones, y ninguno de ellos ha implementado más de 5.

**Análisis**

La gran mayoría de los estudiantes ha implementado pocas o ninguna solución integral con sistemas embebidos, tanto dentro de la universidad, como fuera de ella, lo que implica que será considerablemente más difícil para ellos aplicar sus conocimientos teóricos al diseño de soluciones prácticas, quitándoles competitividad frente a otros profesionales del ramo.

### 4.2.2 Entrevista

Se aplicó una entrevista parcialmente estructurada con el docente de Microcontroladores de la Carrera de Ingeniería en Electrónica y Comunicaciones.

Número: 1

Entrevistado: Ing. Luis Pomaquero.

Entrevistador: Gabriel Eduardo Santamaría Galarza

Lugar y fecha: Ambato, jueves 28 de marzo de 2013

Objeto de estudio: Planificación de clases y laboratorios de Microcontroladores.

- **¿Cómo se imparte teoría de microcontroladores**

  "Se estudian los diferentes *datasheet* de cada uno de los microcontroladores en estudio"

- **¿Qué ejemplos se usan?**

  "Se emplea el set de instrucciones de cada uno de los microcontroladores. Se usa la plataforma MPLABX en conjunto con el compilador XC8, ambos de Microchip"

- **¿Cómo se planifican los laboratorios?**

  "Los laboratorios se planifican acorde al avance programático del módulo; en donde cada tema tiene un laboratorio. Se hace hincapié en la utilización de los módulos, especialmente del PIC16F877A y el PIC18F4550"

- **¿Qué contenidos se desarrollan?**

  "Los contenidos desarrollados son:

  - Puertos de entrada y salida.
  - Entradas analógicas y digitales
  - Interrupciones
  - Módulos de CCP (Comparador, capturador y pwm)
  - Comunicación Serial USART y RS485
  - Comunicación SPI e $I^2C$
  - Comunicación USB con PIC18F2550 o PIC18F4550
  - Adquisición de datos
  - Control de procesos
  - Timers y control de frecuencia mediante configuración interna"

### 4.2.3 Valoración de la Ficha de Observación

**Ficha de observación**

**Lugar:** Oficina de control de laboratorios de Electrónica e Industrial

**Fecha:** 21 de Marzo de 2013

**Investigador:** Gabriel Eduardo Santamaría Galarza

**Objeto de evaluación: Existencia de placas entrenadoras de microcontroladores**

Existen dos modelos de tableros lógicos:

- UC – 06 Centronic Connector. (2 unidades)
- IDL – 80 Digital Lab (5 unidades)

Existe un modelo de programador de PICs

- Super Pro (1 unidad)

Cuenta además con una plataforma de desarrollo:

- NI ELVIS II (1 unidad que se encuentra en calidad de préstamo)

**Interpretación – valoración:**

Los tableros digitales proveen herramientas como generadores de señal y alimentación, para aplicaciones electrónicas en general.

El grabador no tiene más funcionalidad que la de transmitir el archivo .hex al microcontrolador.

Ambas herramientas, aunque se pueden utilizar en la cátedra de Microcontroladores, no tienen funcionalidades específicas para ello, no existen en número suficiente para todos los estudiantes, y no son fácilmente reproducibles y menos aún de manera económica.

La unidad NI ELVIS II no cuenta con el módulo básico "*Freescale Microcontroller (MCU) Student Learning Kit (SLK)*" para el diseño de proyectos con microcontroladores.

### 4.2.4 Situación Actual

Los estudiantes que actualmente han atendido o están atendiendo al módulo de microcontroladores, no han fijado de manera apropiada el conocimiento, aun cuando tienen más o menos claros los conceptos básicos, no son capaces de aplicarlos eficientemente en soluciones reales que involucren microcontroladores y sistemas embebidos.

Actualmente, se imparten 3 horas académicas de la cátedra de Microcontroladores, estando a cargo del Ingeniero Luis Pomaquero. Se utilizan al menos dos variedades de PIC de la empresa Microchip, el 16F877A y el 18F2550 para cubrir la planificación.

En la Oficina de control de laboratorios de Electrónica e Industrial, no existe material didáctico necesario ni suficiente para una enseñanza didáctica de Microcontroladores y a pesar de ello, no cuentan con una alternativa barata y de rápida construcción para elaborar material didáctico y propiciar investigaciones propias acerca de los microcontroladores y sistemas embebidos.

# CAPÍTULO V

# CONCLUSIONES Y RECOMENDACIONES

## 5.1 Conclusiones

Una vez analizados los datos de la entrevista y la ficha de observación; tabulados e interpretados los datos de la encuesta, se ha llegado a las siguientes conclusiones:

- Las clases de teoría de Microcontroladores se planifican y se basan en publicaciones oficiales de Microchip, una marca dedicada a la fabricación de estos dispositivos, cubriendo en el tiempo de clases, los aspectos básicos del microcontrolador PIC.

- Todos los estudiantes de Ingeniería en Electrónica y Comunicaciones reciben clases teórico - prácticas sobre Microcontroladores, sin embargo, solo una parte de ellos realiza constantemente prácticas e implementaciones con ellos, debido a la falta de tiempo de prácticas y la necesidad de construir prototipos propios desde su inicio.

- El conocimiento acerca de Microcontroladores se pierde en la mente de los estudiantes conforme pasa el tiempo, debido a la misma falta de implementaciones.

- Existen un total de nueve equipos que pueden usarse para realizar prácticas con microcontroladores, para 64 estudiantes refiriéndose solamente los niveles superiores; sin contar los alumnos de los semestres básicos, por lo que se concluye que ese número de equipos no es suficiente para el número de estudiantes de electrónica.

- No existe ningún diseño abierto pensado tomando en cuenta el mercado local que pueda ser fácilmente reproducido por los estudiantes, de manera económica; que les ayude a realizar prácticas variadas con microcontroladores a bajo costo.

## 5.2 Recomendaciones

- Es recomendable que en las clases de teoría de Microcontroladores se traten temas que exploten las funcionalidades de los mismos, animando a los estudiantes a hacer investigación sobre microcontroladores en general de manera profunda y consciente por su cuenta y también guiados por el docente.

- Se recomienda optimizar el tiempo de prácticas de Microcontroladores, a través de recursos didácticos legales que no necesiten demasiado tiempo para poner a punto las implementaciones.

- Para aumentar la retención del conocimiento acerca de Microcontroladores en la mente de los estudiantes, es recomendable aumentar el número y complejidad de las implementaciones, así como animar a que se realicen más implementaciones propias.

- Se recomienda encontrar alternativas para que cada estudiante posea dispositivos de entrenamiento que complementen a los existentes en los laboratorios de Electrónica.

- Se debe poner disponible un diseño abierto de placa de entrenamiento, que pueda ser fácilmente estudiado, personalizado y construido por los estudiantes, de manera económica.

# CAPÍTULO VI

# PROPUESTA

## 6.1 Datos Informativos

### 6.1.1 Tema de la Propuesta

Placa entrenadora para la enseñanza didáctica de microcontroladores en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

### 6.1.2 Institución Ejecutora

Institución Educativa: Universidad Técnica de Ambato

Tipo de Organización: Pública

Carrera: Electrónica y Comunicaciones

### 6.1.3 Beneficiarios

- Facultad de Ingeniería en Sistemas, Electrónica e Industrial.
- Estudiantes de la Carrera de Electrónica y Comunicaciones.

### 6.1.4 Ubicación

Provincia: Tungurahua

Cantón: Ambato

Dirección: Av. Los Chasquis y Río Guayllabamba.

### 6.1.5 Equipo Responsable

Tutor: Ing. Juan Pablo Pallo Noroña Mg.

Investigador: Gabriel Eduardo Santamaría Galarza.

## 6.2 Antecedentes de la Propuesta

Al hacer la revisión sobre trabajos similares con placas de entrenamiento, en la Biblioteca de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato no se ha encontrado ninguna propuesta similar a la establecida en esta investigación.

El planteamiento del presente proyecto llega a partir de la experiencia del investigador, que ha tomado la cátedra de Microcontroladores sin conocimientos previos y sin contar con el material didáctico adecuado, en ese momento donde se plantea empíricamente la necesidad de desarrollar material didáctico propio, que se adapte a las ofertas del mercado ambateño y a la economía estudiantil, así como a los requerimientos de software libre que tiene actualmente la sociedad.

De acuerdo a la investigación precedente, se ha determinado la necesidad de una propuesta innovadora que permita poner a disposición de cualquier estudiante una placa de entrenamiento a bajo costo y que contribuya al aprendizaje práctico y fijación del conocimiento, tanto de la persona que lo construya, como de las personas que posteriormente accedan a él en los laboratorios.

Existen a nivel mundial placas de entrenamiento de microcontroladores, algunos comerciales, como el "*Freescale Microcontroller (MCU) Student Learning Kit (SLK)*", (Internet; 2013, 04, 04; http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=PBMCUSLK); y otros cuyo diseño es libre y puede reproducirse por cualquiera que tenga a la mano los elementos como el PIC TRAINER (Internet; 2013, 04, 04; http://www.ucontrol.com.ar/wiki/index.php/PIC_TRAINER); sin

embargo, aún se vive en un entorno en donde la disponibilidad limitada, y los altos costos de envío de componentes del extranjero, e inclusive de otras ciudades del Ecuador, dificulta el obtener estos módulos y más aún hacerlos operativos.

## 6.3 Justificación

En base a los estudios llevados a cabo, se ha determinado que el material didáctico disponible para la Carrera de Electrónica es insuficiente para la cantidad de alumnos, convirtiéndose en una necesidad evidente el incrementar el número de estos equipos.

Refiriéndose a la misma fuente, se ha detectado que la insuficiente dotación de equipos de prueba para la realización de prácticas, así como la influencia que tiene este hecho sobre la pobre retención de conocimientos por parte de un elevado porcentaje de los estudiantes, son hechos que afectan al prestigio de la carrera, y deben ser solventados a la brevedad posible.

La propuesta de diseño de una placa entrenadora para la enseñanza didáctica de microcontroladores en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, se justifica por el mismo hecho de ser un centro de enseñanza de Ingeniería, que carece de un proyecto propio, desarrollado localmente, que sirva e incentive a la investigación en las nuevas generaciones de futuros ingenieros.

Otro justificativo, no menos importante, es que en todo momento se toma en cuenta la disponibilidad de componentes en el mercado local, facilitando enormemente el trabajo de quienes se propongan su construcción, y abaratando costos que también es una parte importante del diseño.

En la presente investigación, no se ha mencionado siquiera el diseño electrónico ni la programación de microcontroladores utilizando GNU/Linux o herramientas libres, lo cual es imprescindible en la realidad ecuatoriana, en la cual en varios casos, no cuenta con presupuesto suficiente para utilizar única y exclusivamente software y hardware pagado, con lo cual se justifica

y es meritorio un trabajo investigativo orientado la enseñanza y desarrollo de sistemas embebidos en GNU/Linux.

## 6.4 Objetivos

### 6.4.1 Objetivo General

Diseñar y construir una placa entrenadora para la enseñanza didáctica de microcontroladores en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

### 6.4.2 Objetivos Específicos

1. Plantear un programa de prácticas con microcontroladores, que comprenda el estudio de sus fundamentos; así como la transmisión de datos, aplicación de DSP, robótica y sistemas embebidos.
2. Diseñar un diagrama electrónico que contenga todos los requerimientos de las prácticas antes mencionadas.
3. Diseñar una placa de circuito impreso (PCB) basado en el diagrama electrónico, utilizando herramientas GNU/Linux.
4. Construir un prototipo de placa entrenadora de microcontroladores para los laboratorios de Electrónica de la FISEI.

## 6.5 Análisis de Factibilidad

Esta propuesta es factible según los resultados de la investigación. Para su mejor comprensión, la factibilidad se ha dividido en diferentes áreas, que se detallan a continuación, refiriéndose cada una a un aspecto concreto.

### 6.5.1 Factibilidad Técnica

La construcción de un PCB, que cumpla ciertas condiciones de diseño, como el que se plantea, es técnicamente factible, puesto que se utilizarán recursos existentes en el ámbito ecuatoriano, utilizando partes, componentes y programas ya estudiados o manejados en el transcurso de la carrera.

### 6.5.2 Factibilidad Humana

El investigador posee amplia experiencia diseñando e implementando soluciones basadas en microcontroladores, utilizando GNU/Linux, por lo cual está perfectamente capacitado para el emprendimiento y término exitoso del proyecto.

### 6.5.3 Factibilidad Económica

La factibilidad económica de la presente propuesta es uno de sus pilares fundamentales, al utilizar ingenio para reducir costos, y utilizar software disponible gratuitamente, en su mayoría libre, y componentes baratos, reemplazables, y económicos, existentes en el mercado.

### 6.5.4 Factibilidad Comercial

En el mercado ecuatoriano existen sectores inexplotados, especialmente en instituciones educativas de enseñanza media y superior, que se constituyen en potenciales compradores de la solución.

Sobre sus competidores en el ámbito local, posee la ventaja de que toda la información, tanto de hardware, como de software está incluida en el paquete.

### 6.5.5 Factibilidad Legal

La presente propuesta se basa en el uso de software GNU/Linux, software gratuito distribuible, y las licencias que lo acompañan, siendo totalmente legal su distribución por cualquier medio.

### 6.5.6 Factibilidad Científica

La presente propuesta se basa en publicaciones científicas, libros, revistas, foros y manuales oficiales de diseño electrónico, componentes y recursos informáticos de probada eficacia, por lo cual es perfectamente factible en el plano científico.

## 6.6 Fundamentación

### 6.6.1 Didáctica en la Enseñanza de Microcontroladores

La enseñanza de temas relacionados a la tecnología, es óptima cuando se hace en un ambiente orientado a la práctica. La enseñanza de microcontroladores no es la excepción, puesto que junta el desarrollo de un programa informático al diseño de un medio físico; para de seta manera satisfacer una necesidad específica.

En la experiencia del investigador, la resolución de problemas propuestos es la mejor manera de acometer a la tarea de formar a profesionales en sistemas microcontrolados.

Sin embargo, dada la variedad de alternativas existentes, tanto en lenguajes de programación, como en métodos de implementación; se debe tener en cuenta dos aspectos clave. La curva de aprendizaje del lenguaje en sí, y la experiencia que se va adquiriendo en el diseño del hardware.

Para infundir el conocimiento de manera eficiente, es recomendable partir de un código claramente explicado, e ir reformándolo de manera que el educando llegue a entender qué hace cada una de sus líneas y pueda utilizarlas posteriormente en diseños propios.

De la misma forma, tener una plataforma física probada y confiable, facilita la experimentación y hace que la adquisición de experiencia sea bastante rápida. No obstante a ello, una plataforma educativa eficiente, debe influir de la menor manera posible en la creatividad del usuario, y en los diferentes enfoques que de él pueden surgir.

### 6.6.2 Prácticas Básicas

Como su nombre lo indica, son las prácticas que contienen los conceptos básicos e iniciales de microcontroladores. Es esencial consultar la hoja de especificaciones para conocer las capacidades y limitaciones del dispositivo a usar.

Los principios fundamentales de la programación de microcontroladores pueden resumirse en el manejo de los periféricos que posea el dispositivo elegido. Por regla general, cualquiera de ellos tendrá por lo menos un conjunto de salidas y entradas digitales. Con ellas podrá interactuar con el proceso.

La placa entrenadora no está limitada a ningún lenguaje en particular, sin embargo con motivos didácticos, en el presente texto se utilizarán los diferentes herramientas que provee la casa Microchip, de manera gratuita. Estos son MPLAB X IDE Ver 1.85 y el compilador XC8 1.20. También se utilizará el compilador Protón IDE.

MPLAB X es un entorno de desarrollo multiplataforma (Linux, Mac y Windows) para PICs. Está programado sobre NetBeans y puede ser obtenido gratuitamente de http://www.microchip.com/pagehandler/en-us/family/mplabx/

El compilador, gratuito de la misma manera; es un intérprete de ANSI C. su descarga está disponible en el enlace mencionado anteriormente. Se ha escogido estas herramientas por ser las que distribuye el fabricante y por contar con numerosos ejemplos.

También se recomienda estudiar compiladores como GCBasic, SDCC o Jalv2 todos ellos gratuitos. Los compiladores pagados CCS o Protón (con algunos ejemplos incluidos en los anexos) son también alternativas válidas, así como considerar la amplia variedad de desarrollos existentes.

En la gran mayoría de lenguajes, el primer paso es declarar el tipo de dispositivo para el cual se va a compilar.

En XC8:
*#include <xc.h>*

En Protón IDE:
*Device = 18F2550*

El siguiente paso es el llenado de diversos registros de configuración, que afectarán el comportamiento del microcontrolador. Cada uno de los registros y su funcionamiento está ampliamente explicado en la hoja de especificaciones (*datasheet*). Son comúnmente llamados *fuses* y entre los más importantes se tiene:

Para usar el oscilador interno en XC8:
*__CONFIG FOSC_INTOSCIO*

Para usar el oscilador interno en Protón IDE:
*Config INTRC_OSC_NOCLKOUT*

Para usar cristal externo de 4 Mhz en XC8:
*pragma config FOSC = XT_XT*
#define _XTAL_FREQ 400000

Para usar cristal externo de 4 Mhz en Protón IDE:
*XTAL = 20*

Los puertos pueden ser utilizados como entrada o como salida. Es absolutamente necesario declarar el uso que se hará para evitar funcionamiento indeseado del programa. Para esto se utilizará el nombre de

los registros de configuración de puertos TRISx. En donde la x representa el puerto deseado. Para declarar el pin del puerto como salida, se usa el 0. Para declararlo como entrada se utiliza el 1.

En XC8:

*TRISA=0xFF;  //Todo PORTA como entrada.*

En Protón IDE:

*TRISC=%00000000  " Poner PortC como salida, el símbolo de porcentaje representa un número binario. Cada posición representa un pin, siendo el LSB el último cero a la derecha.*

Algunas directivas de configuración no serán necesarias, dependiendo del caso. En este punto, se debe recalcar el hecho de que la hoja de datos contiene toda la información necesaria para las implementaciones deseadas.

Para asignar los valores respectivos en los puertos existe PORTx, en donde x representa al puerto en cuestión, por ejemplo:

En XC8:

*PORTB=0x00;  // Todo PORTB con valor cero.*
*PORTBbits.RB0=1;  // Solo RB0 con valor lógico alto.*

En Protón IDE:

*PORTB=%00000000  " Todo PORTB con valor cero.*
*PORTB.0=0  " Solo RB0 con valor lógico alto.*

Cada acción que pueda realizar el microcontrolador se representa por el valor en un registro específico, y el resultado de cada operación que realice se almacenará en un registro especializado. Se pueden ejecutar procesos desde los más simples, como el explicado en este apartado, hasta los más complejos imaginables.

Por ejemplo, es posible hacer que el micro responda ante una interrupción (*INTCON*), discriminar por tipo de interrupción (*PIR*), habilitar el módulo

USART (*RCSTA*) y una infinidad de combinaciones de acuerdo a las necesidades de implementación.

Un programa adecuado de prácticas básicas contempla el estudio de los registros de configuración, las posiciones de memoria; y un adecuado manejo de los puertos de entrada y salida de acuerdo con la hoja de especificaciones del dispositivo seleccionado.

La placa de entrenamiento debe contar con un acceso fácil a los periféricos del microcontrolador, así como contar con elementos ópticos como leds para una mejor comprensión del funcionamiento del programa.

### 6.6.3 Prácticas de Transmisión de Datos

Los protocolos de comunicación son, en esencia, un sistema estandarizado de control de señales, ya sean esta eléctricas, luminosas o de otros tipos. Los microcontroladores son capaces de manejar estándares, ya sea por módulos de hardware como el USART de los PICs o mediante un programa grabado en él. Incluso con alguna implementación adicional se pueden ampliar las capacidades del dispositivo.

En cualquier microcontrolador, así no posea un módulo especializado, es posible lograr diversos tipos de transmisión de datos. Existen librerías especializadas para este propósito, como las siguientes:

En XC8:
*#include <plib/usart.h> // Librería para manejo del módulo USART*

En Protón IDE:
*SerIn PORTA.0 , 9600 , [VariableDatos]*
*SerOut PORTB.0 , 9600 , [@i," UTA - FISEI",13]*

Ambas son maneras de transmitir y recibir datos se manera serial.

Teniendo en cuenta los niveles de voltaje que entrega el PIC, de 3.3 a 5 v generalmente, se puede realizar implementaciones de todo tipo de protocolos.

Un estándar de transmisión serie ampliamente utilizado es el RS-232 (*Electronic Industries Alliance Recommended Standard-232, TIA/EIA-232*), sin embargo, en el documento "*Comparing Bus Solutions*" de Texas Instruments, (2004, pág 28) reza: el estándar RS-232 "tiene altas amplitudes de señal, de ± (5 v a 15 v) a la salida del driver". Estas magnitudes de voltaje destruirían la mayoría de microcontroladores, circunstancia que hace necesaria la implementación de drivers que hagan las veces de intermediarios entre el pic y el dispositivo RS-232.

Por otro lado, la movilidad actual, hace que no siempre se cuente con una interface serie estándar. En cambio, los ordenadores, sobe todo los portátiles, tienen interfaces USB, más avanzadas y mas accesibles. El USB trabaja en niveles de voltaje TTL (de 0v a 5v). Esta circunstancia puede ser aprovechada incluso para alimentar la placa, teniendo en cuenta el consumo máximo de la misma.

Existen microcontroladores, como los de la familia 18Fxx5x de Microchip o el ATMEGA8, que soportan comunicación USB por hardware. Están también proyectos como el V-USB para AVR que no posean soporte nativo.

Para trabajar con USB, Microchip provee las *microchip-libraries-for-applications* en su versión 2013-06-15, disponibles para descarga gratuita en http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&dDocName=en547784

Un programa adecuado de prácticas de transmisión de datos debe incluir el manejo de los periféricos de comunicaciones que tienen los microcontroladores. Conjuntamente, debe estudiar comunicaciones por software y estándares que pueden ser implementados con dispositivos embebidos.

Por lo anteriormente expuesto, una placa de entrenamiento debe contener un acceso fácil tanto a los pines de transmisión por hardware, como a los potenciales pines de transmisión por software. Asimismo, debe contar con una interfaz estándar RS232 y una USB, ya que son las más utilizadas para implementar protocolos de comunicaciones. Sin embargo, los pines no deben estar permanentemente conectados esta interfaz, para que la entrenadora sea flexible y práctica.

Esta placa puede utilizarse en la cátedra de Interfaz de PC, ya que provee excelentes puntos de prueba, tanto para medir magnitudes eléctricas en las comunicaciones, como para visualizar tramas de estándares de comunicación en el osciloscopio.

### 6.6.4 Prácticas de Aplicación de Teoría de DSP

DSP *Digital Signal Processing* o Procesamiento Digital de Señales, se refiere al análisis de técnicas y procedimientos para el tratamiento de la información digital. Todas las señales que son procesadas en un microcontrolador son digitales. Esto, independientemente de cómo hayan sido adquiridas.

Existen alternativas de microcontroladores o FPGAs especialmente diseñadas para DSP, como el dsPIC 30F2010 o el Xilinx Spartan 3e. Las principales diferencias son el poder de procesamiento, y la manera en que tratan la información. Éstos están diseñados para dar tratamiento a la información en tiempo real, en implementaciones que requieren mínima tolerancia a los retardos. Por otra parte, los costos de ejecución son un limitante para que estudiantes puedan hacer implementaciones de este tipo sin conocimiento previo.

Justamente en este aspecto, la utilidad de una placa básica de entrenamiento es evidente, debido al hecho que, por un costo significativamente menor, se pueden hacer prácticas iniciales sobre el funcionamiento de sistemas microcontrolados aplicados a estos campos, como se detalla a continuación.

Algunos modelos de microcontroladores, como el PIC18F2550 tienen módulos conversores A/D. Estos muestrean una señal analógica y la convierten en un número digital correspondiente de 10 bits.

Como el microcontrolador es capaz de realizar operaciones matemáticas con el número binario que representa las señales, es posible aplicar diversos métodos y técnicas para procesar la señal a conveniencia.

Por ejemplo, realizando un sencillo procedimiento lógico para negar cada una de las posiciones de memoria con los valores obtenidos de los registros de la transformación del conversor A/D (ADRESL y ADRESH); es posible encontrar la Función espejo inversa de la señal de entrada.

Un ejemplo para una implementación avanzada es aplicar el algoritmo Split-Radix para obtener la Transformada Discreta de Fourier de una señal. Esto es realmente útil para analizarla en el campo de la frecuencia.

Un programa adecuado de prácticas de aplicación de teoría de DSP con microcontroladores, abarca aspectos que van desde la adquisición de datos, hasta la visualización de resultados, ya sean de forma numérica o gráfica.

La placa necesita maneras de visualizar los datos resultantes de estas transformaciones de señales. Existen muchas maneras de efectuar este procedimiento. Ya sea con un LCD para un sistema *stand alone* (que solamente necesita las placas con microcontroladores), o comunicando los datos a un pc para su visualización y análisis en algún programa más complejo. Este cometido se logra con periféricos de comunicación como el USB o el puerto serie tratados en el apartado anterior.

Junto con lo anteriormente expuesto, se puede utilizar la placa para aplicar conocimientos de Comunicaciones Analógicas para filtrar señales de entrada o salida.

### 6.6.5 Prácticas de Robótica

Los sistemas microcontrolados dotan de poder de procesamiento a un robot o autómata. Aún con alternativas más potentes como los FPGAs, los robots con microcontroladores son una estupenda aproximación para el estudiante recién iniciado.

Una vez más, la primera explicación pertinente es aplicar los conceptos básicos para una implementación en un campo específico. Por ejemplo; al utilizar las entradas del microcontrolador para conectar sensores, ya sean éstos analógicos, como el LM35, o digitales como el DS1820; se dota al sistema con maneras de escanear los cambios en su entorno.

Esta circunstancia se hace evidente al incorporar actuadores, como motores con sus respectivos drivers, a los pines configurados como salidas. El conjunto hace que la implementación sea capaz de reaccionar de manera autónoma a circunstancias cambiantes, lo cual es la base de la robótica.

A continuación se describe un ejemplo de implementación simple, utilizando un par de sensores CNY70 debidamente conectado a pines de entrada del microcontrolador. Las entradas se pondrán en estado lógico alto cuando la superficie sea oscura y en bajo cuando sea clara. Realizando un pequeño programa que corrija el rumbo del autómata se logra un seguidor de línea con microcontrolador.

En el otro extremo está la utilización de varios microcontroladores conectados en paralelo, para repartirse tareas. Por ejemplo, mientras un PIC18F2550 se encarga de transmitir datos por USB, un PIC16F876A toma muestras de señales analógicas con visualización en leds y un PIC16F88 se encarga de manejar un LCD; formando así un clúster de microprocesadores.

En un programa adecuado de prácticas de robótica, es imprescindible incluir el manejo de sensores y actuadores. Además, se pueden incluir prácticas para el ahorro de energía, como el uso de la función de *sleep*. Las

comunicaciones entre microcontroladores también deben ser tomadas en cuenta.

En la placa entrenadora, es necesaria la presencia de un fácil acceso a todos los puertos del microcontrolador, tanto para conectar sensores, como para diseñar módulos apilables que permitan constituir un clúster de microcontroladores.

La aplicación puede hacerse incluso con competencias de la carrera como Instrumentación y control de procesos o sistemas de control, para perfeccionar las respuestas del autómata, incluyendo por ejemplo métodos de control de histéresis y similares.

### 6.6.6 Prácticas de Sistemas Embebidos

La esencia de los microcontroladores es su utilización en sistemas embebidos. Casi la totalidad de sistemas embebidos y sensores comercialmente disponibles, basan su funcionamiento en microcontroladores. La razón para ello es el bajo costo que representan las implementaciones de este tipo.

En un sistema embebido, el o los microcontroladores realizarán tareas específicas. Por ejemplo, en un circuito sensor de humedad en lugares lejanos, se puede tener un microcontrolador sencillo que gobierne los sensores y transmita las mediciones por cualquier medio, mientras que uno más potente recibe los datos y los procesa, o a su vez los retransmite a una estación de control.

Otro ejemplo sencillo son los sensores de movimiento con luz, que acoplan un sensor PIR (*Passive InfraRed sensor*), que cuenta con tres terminales. Dos de ellos son para la alimentación y el restante devuelve un estado lógico alto cuando detecta calor. Simplemente conectando una interface de potencia, como un relé o un TRIAC, es posible controlar una bombilla. Incluso se puede llevar más lejos la aplicación transmitiendo los datos a un sistema de detección de intrusos o una implementación por el estilo.

Un programa adecuado de prácticas en sistemas embebidos, incluye el manejo de interfaces de potencia, comunicaciones, sensores y actuadores, tales como motores, relés o dispositivos de estado sólido.

Para ser utilizada en sistemas embebidos, la placa de entrenamiento debe tener acceso a las entradas del microcontrolador para conectar sensores y actuadores. También es deseable que la placa acepte un módulo con interfaces de potencia, para control de dispositivos con tensiones mayores aniveles TTL.

Es factible en este apartado, combinar la placa madre con los estudios de Electrónica de Potencia, diseñando módulos con DIACS, TRIACS o cualquier interface de estado sólido.

### 6.6.7 Esquema de la Propuesta

Una vez tomadas en cuenta las consideraciones principales de diseño, se ha decidido dividir la placa entrenadora en sendos módulos que cumplirán funciones específicas. Lo cual permitirá cumplir con los objetivos de manera eficiente, se mentalizó la propuesta como un sistema modular.

Estos factores posibilitan que la solución final sea un sistema económico y manejable tanto en el laboratorio como en el aula e incluso en el domicilio. El resultado es un producto totalmente ampliable y modificable, así como también se tiene la opción de construir los módulos según se vayan requiriendo en la Carrera, modificarlos a conveniencia, o diseñando módulos personalizados.

A continuación se presenta un diagrama de bloques de lo que es el sistema propuesto y los diferentes módulos que lo componen.

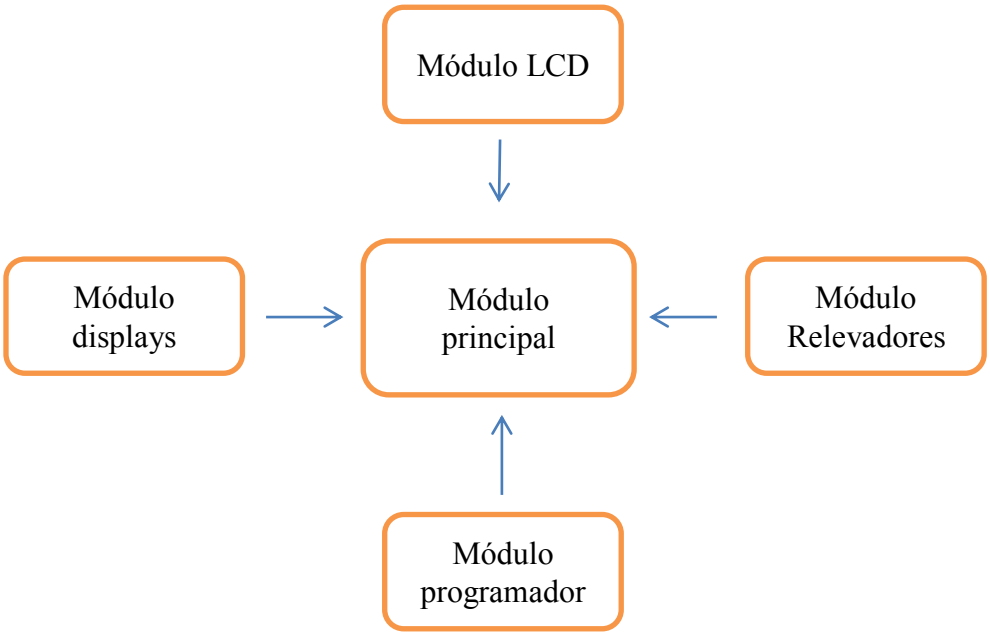



**Figura 6.1** Esquema de la propuesta.
Autor: El investigador

El módulo principal es el único absolutamente necesario para comenzar con las prácticas, que incluye todos los elementos necesarios para el funcionamiento del microcontrolador. Así mismo, contiene las interfaces básicas de comunicaciones, y acceso cómodo a sus periféricos.

El resto de módulos no es vital para la mayoría de prácticas, sin embargo, son especialmente útiles en aplicaciones avanzadas que requieran varios componentes adicionales. Esto, para abaratar costos y alentar al estudiante a construir el prototipo.

### 6.6.8 Elementos Electrónicos Básicos

- **Resistencia**

Según escribe Prat Viñas (1998, pág 36), en su obra "Circuitos y dispositivos electrónicos", "La *resistencia lineal ideal* es un elemento de circuito cuya característica (i,v) es una recta que pasa por el origen", como lo muestra la **Figura 6.2** Característica (i, y) de la resistencia y la relación matemática de dicha recta es la conocida Ley de Ohm, cuya fórmula es:

$$I = \frac{v}{R}$$

**Ecuación 6.1** Ley de Ohm.
Fuente: Circuitos y dispositivos electrónicos, página 36.

En donde:

**I** Es la intensidad de corriente.

***v*** Es el voltaje

***R*** Es la resistencia

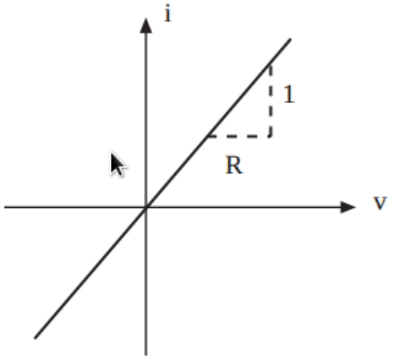



**Figura 6.2** Característica (i, y) de la resistencia.
Fuente: Circuitos y dispositivos electrónicos, página 36.

La resistencia es, para lo que nos compete, un elemento electrónico cuya reacción a una corriente que lo atraviesa, es inversamente proporcional a dicha corriente, y directamente proporcional al voltaje generado por ella. En palabras más simples, es un elemento al cual cuando se le aplica una corriente, genera una caída de voltaje entre sus terminales.

La unidad de medida de la resistencia eléctrica es el ohmio, representado por la letra griega omega (Ω). Suelen comercializarse en valores estandarizados, teniendo un código de colores o de números impreso para identificar los valores de los componentes.

De la *Ley de Ohm,* se desprende el siguiente análisis dimensional:

$$A = \frac{v}{\Omega}$$

**Ecuación 6.2** Análisis dimensional de la ley de Ohm.
Fuente: Circuitos y dispositivos electrónicos, página 36.

En donde:

| | |
|---|---|
| $\boldsymbol{A}$ | Símbolo de Amperio. |
| $\boldsymbol{v}$ | Símbolo de Voltio |
| Ω | Símbolo de Ohmios |

La forma en la que se comercializan es diversa, desde resistencias de cobre para PCB multicapa, hasta grandes resistencias de tiza que son capaces de dispersar gran cantidad de energía.

En gran parte de los programas de Diseño Electrónico Automatizado, concretamente en Kicad, se utilizan por defecto símbolos europeos, siendo el de la resistencia el que se muestra en a la **Figura 6.3** Resistencia en el esquemático de Kicad**:**

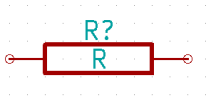



**Figura 6.3** Resistencia en el esquemático de Kicad.
Autores: Kicad Developers Team

- **Diodo**

Es un dispositivo semiconductor, que solamente permite el paso de la corriente en un sentido. Las características de un diodo ideal se muestran en la **figura 6.4** Cuando la corriente fluye en el sentido de polarización del diodo, y supera el voltaje de polarización; típicamente 0,7 v en los diodos comunes, se comporta como un circuito cerrado. Caso contrario, si la corriente fluye en contra del sentido de polarización, se comportará como un circuito abierto.

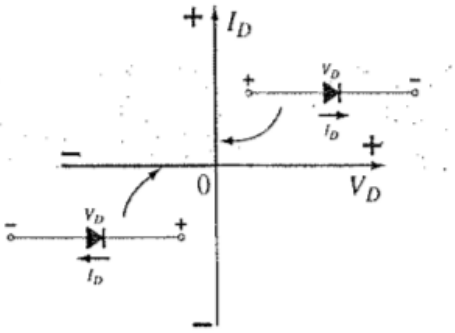



**Figura 6.4** Características de un diodo ideal.
Fuente: Electrónica: Teoría de circuitos y dispositivos electrónicos, página 1

En el software de diseño Kicad, el símbolo que representa el diodo en el esquema eléctrico es el que se detalla en la **figura 6.5**:

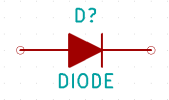



**Figura 6.5** Diodo en el esquemático de Kicad.
Autores: Kicad Developers Team.

- **Diodo Led**

Citando a Prat Viñas (1998, pág 329), en su obra "Circuitos y dispositivos electrónicos", "LED (iniciales de Light Emitting Diode), es un diodo de unión P-N que emite luz cuando está polarizado en directa. La intensidad de la luz emitida es aproximadamente proporcional a la intensidad de la corriente que atraviesa el diodo. La longitud de onda de la luz emitida (color) depende del material con el que está fabricado el diodo."

Es decir, es similar al elemento anterior, con la particularidad de que emitirá luz en cierta longitud de onda, cuando esté polarizado directamente, osea, cuando actúe como circuito cerrado.

En Kicad, el símbolo que representa el diodo en el esquema eléctrico es el que se puede ver en la **figura 6.6:**

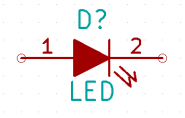



**Figura 6.6** Diodo led en el esquemático de Kicad.
Autores: Kicad Developers Team

- **Condensador**

Es un componente electrónico que tiene la capacidad de almacenar una carga eléctrica. El voltaje entre sus terminales es directamente proporcional a la carga almacenada. La fórmula

$$vc = \frac{q}{C}$$

**Ecuación 6.3** Valor del voltaje entre los terminales del capacitor.
Fuente: Circuitos y dispositivos electrónicos, página 103.

En donde:

| | |
|---|---|
| $\boldsymbol{vc}$ | Voltaje entre terminales. |
| $\boldsymbol{q}$ | Carga almacenada. |
| $\boldsymbol{C}$ | Capacitancia. |

Aplicando el respectivo análisis dimensional, se tiene que:

$$f = \frac{C}{v}$$

**Ecuación 6.4** Análisis dimensional de la capacitancia.
Fuente: Circuitos y dispositivos electrónicos, página 103

En donde:

| | |
|---|---|
| $\boldsymbol{f}$ | Faradio. |
| **C** | Coulomb. |
| $\boldsymbol{v}$ | Voltio. |

Es decir, que un faradio es la capacidad que tiene un condensador, cuando entre sus terminales hay un voltio, y su carga es de un Coulomb.

Al tener la capacidad de almacenar energía fe forma constante, el capacitor puede usarse como un filtro para prevenir variaciones indeseadas de voltaje en puntos críticos de los circuitos.

Los elementos electrónicos más comunes de este tipo son los capacitores polarizados y los cerámicos. En Kicad, los símbolos de capacitor son respectivamente, en la **figura 6.7**:

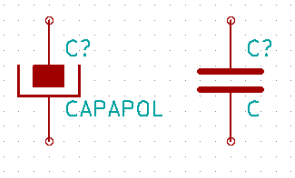



**Figura 6.7** Capacitor polarizado y cerámico en Kicad.
Autores: Kicad Developers Team.

- **Inductor**

Es un elemento electrónico que tiene la capacidad de almacenar energía mediante la creación de campo magnético, cuando circula una corriente a través de él. El tratamiento matemático que se le da es muy similar al de los capacitores, cambiando las magnitudes respectivamente, como lo indica la ecuación:

$$i_l = \frac{q}{L}$$

**Ecuación 6.5** Valor de la corriente entre los terminales de la bobina.
Fuente: Circuitos y dispositivos electrónicos, página 103.

En donde:

| | |
|---|---|
| $\boldsymbol{i_l}$ | Corriente entre terminales. |
| $\boldsymbol{q}$ | Carga almacenada. |
| $\boldsymbol{L}$ | Coeficiente de autoinducción. |

En la suite de diseño electrónico utilizada, el símbolo para el esquemático de la bobina es el que se muestra en la **figura 6.8**:

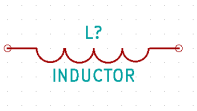



**Figura 6.8** Inductor en Kicad.
Autores: Kicad Developers Team.

- **Transistor Bipolar**

De la obra "Teoría de circuitos y dispositivos electrónicos" de Boylestad & Nashelsky (1998, pág. 132) se toma el estracto textual "El transistor es un dispositivo semiconductor de tres capas que consta de ya sea dos capas de material tipo *n* y una capa tipo *p*, o bien de dos capas de material tipo *p* y una tipo *n*.", por lo tanto se tienen dos tipos de transistores, uno *npn* y el otro *pnp*. En la **figura 6.9:**

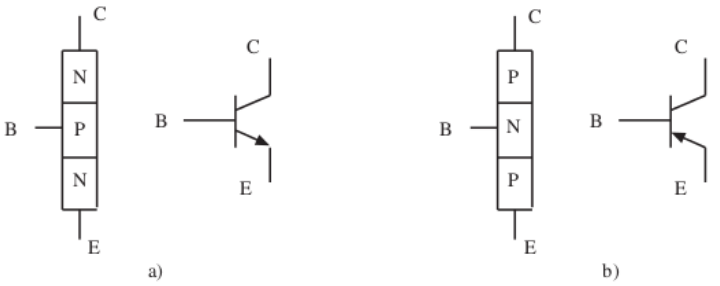



**Figura 6.9** Tipos y símbolos de transistores bipolares a) transistor *npn* b) transistor *pnp*.
Fuente: Electrónica: Circuitos y dispositivos electrónicos, página 205.

El transistor fue desarrollado en los laboratorios Bell, y su primera aplicación fue la de amplificador de corriente. Sin embargo, dependiendo de la configuración de la electrónica, se puede utilizar también como conmutador, en circuitos osciladores y rectificadores.

Para efectos de diseño, el símbolo del transistor se muestra en la **figura 6.10:**

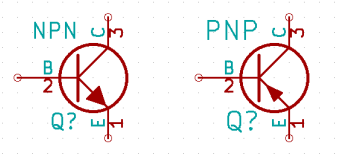



**Figura 6.10** Transistores en Kicad. Izquierda *npn*. Derecha *pnp*.
Autores: Kicad Developers Team.

- **Oscilador y Cristal**

El oscilador es un elemento imprescindible en el funcionamiento de cualquier microprocesador; debido a que esté es el corazón del microcontrolador, hace obligatoria la presencia de un oscilador en un sistema embebido. El rango de la frecuencia de oscilación que soporta un microcontrolador viene dado en su respectiva hoja de datos.

Aunque en algunos casos el oscilador está incluido en el micro, esta no es la norma. Existen diversos tipos de osciladores, siendo el más común el de cristal de cuarzo, cortado de tal manera que al ser excitado vibra de manera predeterminada y a su vez emite pulsos eléctricos a una frecuencia específica.

El símbolo que será utilizado para identificar el cristal en el diseño del esquemático será el de la **figura 6.11**:

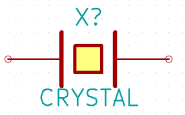



**Figura 6.11** Cristal en Kicad.
Autores: Kicad Developers Team.

- **LM7805**

Este circuito integrado es un regulador de voltaje positivo, que tiene una salida fija de 5v. Puede manejar corrientes superiores al 1ª. Consta de tres terminales como se muestra en la **figura 6.12**:

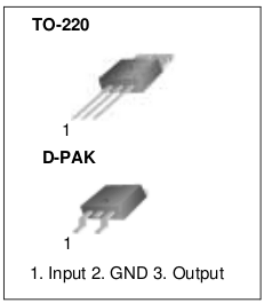



Figura 6.12 **Encapsulados del LM7805.**
Fuente: MC78XX/LM78XX/MC78XXA Datasheet

En la **figura 6.13** se muestra un diagrama interno simplificado del integrado:

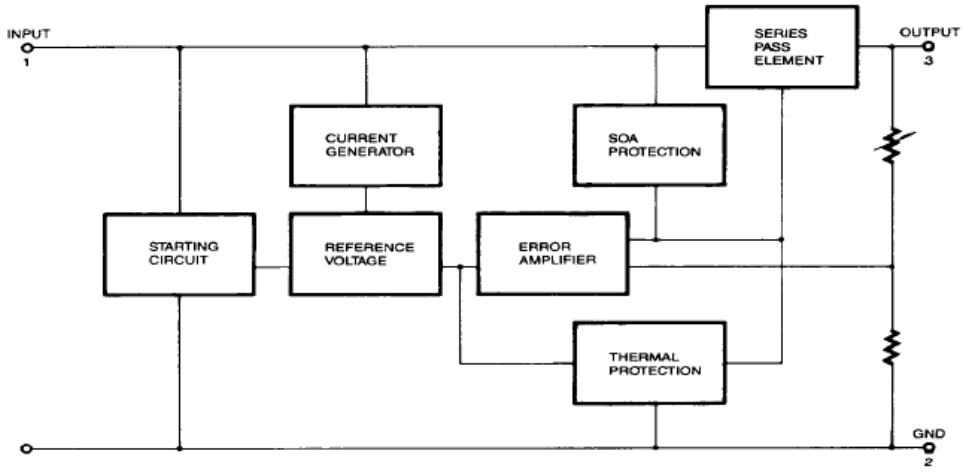



**Figura 6.13** Diagrama simplificado del LM7805.
**Fuente:** MC78XX/LM78XX/MC78XXA Datasheet.

Como se puede observar, las tres patillas tienen funciones perfectamente definidas. La número uno es para el Voltaje de entrada, que puede ir desde 7 a 35 v. La número 2 es para GND, y entre GND y la tercera patilla tenemos la salida de 5v. Típicamente se eliminan ruidos no deseados tanto a la entrada como en la salida conectando en paralelo a cada una de ellas un par de capacitores de baja denominación, tal como se aprecia en la **figura 6.14:**

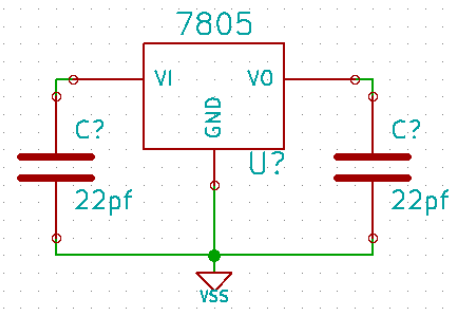



**Figura 6.14** Símbolo y conexión típica del LM7805.
Autor: El investigador

- **LM317**

De forma similar al integrado anterior, es un regulador de voltaje, con la diferencia que a su salida no tiene un voltaje fijo, pudiendo variar desde 1,2v a 37v, manejando una corriente de hasta 1A.

De su hoja de datos se desprende la implementación mostrada en la **figura 6.15:**

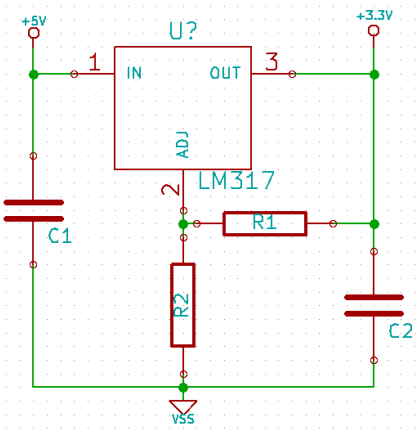



**Figura 6.15** Símbolo y conexión típica del LM317.
Autor: El investigador.

Los valores de R1 y R2 se calculan con la ecuación:

$$v_{OUT} = 1.25v\left(1 + \frac{R2}{R1}\right)$$

**Ecuación 6.6** Cálculo de los valores de Resistencia en función del voltaje de salida.
Fuente: LM117 LM317A LM317 3-Terminal Adjustable Regulator Datasheet.

### 6.6.9 Microcontroladores PIC

Para el presente trabajo, de entre la amplia gama de microcontroladores existentes en el mercado, se ha escogido trabajar con los PIC® de Microchip Technology Inc., siguiendo criterios de disponibilidad, eficiencia y economía.

En el mercado local, los PIC® se pueden encontrar con relativa facilidad y a costos que oscilan entre los tres y treinta dólares. Otros fabricantes, como Atmel o Freesscale son bastante desconocidos en el medio, y a pesar de que se pueden encontrar a precios relativamente bajos, los costos de envío hacen

que por ahora cuesten alrededor de tres veces más que sus contrapartes de Microchip.

Los microcontroladores PIC® más comunes en el mercado local, son los pertenecientes a las familias PIC16F87XA y PIC18F2455/2550/4455/4550, cuyas características, tomadas de sus respectivas hojas de datos se detallan a continuación:

- **PIC16F87XA**

Es un microcontrolador cuya arquitectura es de 8 bits, es decir, utiliza registros de memoria de esta longitud, por lo cual todos los datos e instrucciones serán procesados de 8 en 8 bits.

En la **figura 6.16** se muestra el diagrama de pines del PIC16F877A en encapsulado DIP40:

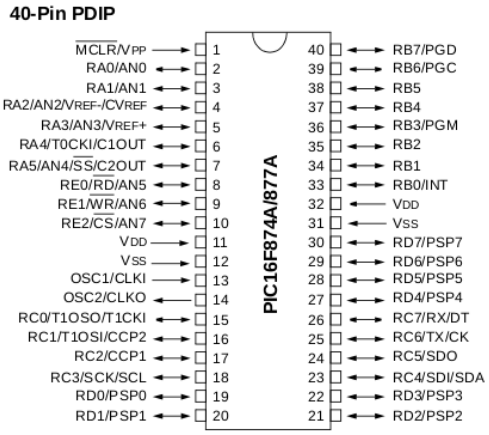



**Figura 6.16** Diagrama de pines 16F877A.
Fuente: PIC16F87XA Datasheet.

Según su hoja de datos, son microcontroladores de gama media, cuyas características se detallan en la **tabla 6.1**:

**Tabla 6.1.** Características del dispositivo PIC16F87XA

| **Características** | **PIC18F873A** | **PIC18F874A** | **PIC16F876A** | **PIC16F877A** |
|---|---|---|---|---|
| Frecuencia de operación | DC - 20 Mhz | DC - 20 Mhz | DC - 20 Mhz | DC - 20 Mhz |
| Restablecer (y Retardo) | POR, BOR, (PWRT, OST) | POR, BOR, (PWRT, OST) | POR, BOR, (PWRT, OST) | POR, BOR, (PWRT, OST) |

| | | | | |
|---|---|---|---|---|
| Memoria de datos (Bytes) | 192 | 192 | 368 | 368 |
| Memoria de datos EEPROM (Bytes) | 128 | 128 | 256 | 256 |
| Fuentes de interrupción | 14 | 15 | 14 | 15 |
| Puertos de Entrada/Salida | Puertos A, B, C | Puertos A, B, C, D, E | Puertos A, B, C | Puertos A, B, C, D, E |
| Temporizadores | 4 | 4 | 4 | 4 |
| Módulos Captura/Comparación/PWM | 2 | 2 | 2 | 2 |
| Módulos de Captura/Comparación/PWM | 0 | 0 | 1 | 1 |
| Comunicaciones seriales | MSSP, USART Mejorado | MSSP, USART Mejorado | MSSP, USART Mejorado | MSSP, USART Mejorado |
| Comunicaciones paralelas | - | PSP | - | PSP |
| Módulo Analógico a Digital de 10 Bits | 5 Canales de entrada | 8 Canales de entrada | 5 Canales de entrada | 8 Canales de entrada |
| Comparadores | 2 | 2 | 2 | 2 |
| Juego de instrucciones | 35 Instrucciones | 35 Instrucciones | 35 Instrucciones | 35 Instrucciones |
| Encapsulados | 28 pines PDIP<br>28 pines SOIC<br>28 pines SSOP<br>28 pines QFN | 40 pines PDIP<br>44 pines PLCC<br>44 pines TQFN<br>44 pines QFN | 28 pines PDIP<br>28 pines SOIC<br>28 pines SSOP<br>28 pines QFN | 40 pines PDIP<br>44 pines PLCC<br>44 pines TQFN<br>44 pines QFN |

Fuente: PIC16F87XA Datasheet, página 5.
Traducción: El investigador.

Entre la información que cabe destacar de la tabla, se tiene la memoria de programa flash de 8K palabras hexadecimales de 14 bits, suficiente para escribir un gran número de aplicaciones. Además, tiene 5 puertos de entrada y salida, lo cual brinda un buen número de pines disponibles para estos menesteres. Incorpora comunicación serial por hardware. Posee también convertidores analógicos – digitales para entradas de este tipo. Esto, entre muchas otras características.

- **Conjunto de Instrucciones**

El conjunto o *set* de instrucciones, consiste en una serie de palabras reservadas que permiten al programador hacer que el procesador ejecute los procesos deseados para una implementación. El código que esté escrito

exclusivamente con estas instrucciones se denomina escrito en *lenguaje ensamblador,* y va a poder ser traducido directamente a código hexdecimal.

Según Microchip, (2013, pág. 159), en su "*PIC16F87XA Datasheet*", la familia PIC16F de Microchip Inc., tiene un conjunto de instrucciones de 35 palabras, "cada una de 14 bits, divididas en un **código de operación**, que especifica el tipo de instrucción y uno o más operadores, que especifican la operación de la instrucción".

Es importante no confundir esta longitud en bits con la de los registros de memoria, dado que el procesador atenderá a estas instrucciones dividiéndolas, de una u otra forma, en palabras de memoria de 8 bits.

La **tabla 6.2** contiene el set de instrucciones para la familia PIC16F87XA

**Tabla 6.2.** Conjunto de instrucciones de la familia PIC16F87XA

| Mnemotécnico, operandos | | Descripción | Ciclos | Código de 14 bits | | | Estado afectado | Notas |
|---|---|---|---|---|---|---|---|---|
| | | | | Msb | | Lsb | | |
| **Operaciones en registros, orientados a bytes** | | | | | | | | |
| ADDWF | f, d | Sumar W y f | 1 | 00 | 0111 | dfff ffff | C,DC,Z | 1,2 |
| ANDWF | f, d | AND W con f | 1 | 00 | 0101 | dfff ffff | Z | 1,2 |
| CLRF | f | Limpiar f | 1 | 00 | 0001 | lfff ffff | Z | 2 |
| CLRW | - | Limpiar W | 1 | 00 | 0001 | 0xxx xxxx | Z | |
| COMF | f, d | Complemento f | 1 | 00 | 1001 | dfff ffff | Z | 1,2 |
| DECF | f, d | Decrementar f | 1 | 00 | 0011 | dfff ffff | Z | 1,2 |
| DECFSZ | f, d | Decrementar f, Saltar si es 0 | 1(2) | 00 | 1011 | dfff ffff | | 1,2,3 |
| INCF | f, d | Incrementar f | 1 | 00 | 1010 | dfff ffff | Z | 1,2 |
| INCFSZ | f, d | Incrementar f, Saltar si es 0 | 1(2) | 00 | 1111 | dfff ffff | | 1,2,3 |
| IORWF | f, d | OR Inclusiva W con f | 1 | 00 | 0100 | dfff ffff | Z | 1,2 |
| MOVF | f, d | Mover f | 1 | 00 | 1000 | dfff ffff | Z | 1,2 |
| MOVWF | f | Mover W to f | 1 | 00 | 0000 | lfff ffff | | |
| NOP | - | No hacer nada | 1 | 00 | 0000 | 0xx0 0000 | | |
| RLF | f, d | Rotar a la izquierda f con acarreo | 1 | 00 | 1101 | dfff ffff | C | 1,2 |
| RRF | f, d | Rotar a la derecha f con acarreo | 1 | 00 | 1100 | dfff ffff | C | 1,2 |
| SUBWF | f, d | Substraer W de f | 1 | 00 | 0010 | dfff ffff | C,DC,Z | 1,2 |
| SWAPF | f, d | Intercambiar *nibbles* en f | 1 | 00 | 1110 | dfff ffff | | 1,2 |
| XORWF | f, d | OR Exclusiva de W con f | 1 | 00 | 0110 | dfff ffff | Z | 1,2 |

| Operaciones en registros, orientados a bits | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| BCF | f, b | Bit Clear f | 1 | 01 | 00bb | bfff ffff | | 1,2 |
| BSF | f, b | Bit Set f | 1 | 01 | 01bb | bfff ffff | | 1,2 |
| BTFSC | f, b | Bit Test f, Saltar si está limpio | 1 (2) | 01 | 10bb | bfff ffff | | 3 |
| BTFSS | f, b | Bit Test f, Saltar si está listo | 1 (2) | 01 | 11bb | bfff ffff | | 3 |
| **Operaciones de control y con literales** | | | | | | | | |
| ADDLW | k | Sumar un literal y W | 1 | 11 | 111x | kkkk kkkk | C,DC,Z | |
| ANDLW | k | AND Literal con W | 1 | 11 | 1001 | kkkk kkkk | Z | |
| CALL | k | Llamar a subrutina | 2 | 10 | 0kkk | kkkk kkkk | | |
| CLRWDT | - | Limpiar Timer de Watchdog | 1 | 00 | 0000 | 0110 0100 | TO,PD | |
| GOTO | k | Ir a dirección | 2 | 10 | 1kkk | kkkk kkkk | | |
| IORLW | k | OR Inclusiva literal con W | 1 | 11 | 1000 | kkkk kkkk | Z | |
| MOVLW | k | Mover Literal a W | 1 | 11 | 00xx | kkkk kkkk | | |
| RETFIE | - | Regresar de Interrupción | 2 | 00 | 0000 | 0000 1001 | | |
| RETLW | k | Regresar con Literal en W | 2 | 11 | 01xx | kkkk kkkk | | |
| RETURN | - | Regresar desde Subrutina | 2 | 00 | 0000 | 0000 1000 | | |
| SLEEP | - | Entrar en modo Standby | 1 | 00 | 0000 | 0110 0011 | TO,PD | |
| SUBLW | k | Substraer W de Literal | 1 | 11 | 110x | kkkk kkkk | C,DC,Z | |
| XORLW | k | OR Exclusiva de Literal con W | 1 | 11 | 1010 | kkkk kkkk | Z | |

**Notas**

**1** Cuando un registro de entrada/salida es modificado con un función de sí mismo (ej. MOVF PORTB, 1), el valor usado va a ser el valor presente en los pines. Por ejemplo, si el dato en el latch es "1" para un pin configurado como entrada y es llevado a cero por un dispositivo externo, el dato va a ser escrito con un "0".

**2** Si esta instrucción es ejecutada en el registro TMR0 (y cuando sea aplicable, d=1), el pre escalador va a ser limpiado si es que está asignado al módulo Timer0.

**3** Si el contador de programa es modificado, o un test condicional es Verdadero, la instrucción requiere dos ciclos. El segundo ciclo es ejecutado como NOP

Fuente: PIC16F87XA Datasheet, página 160.
Traducción: El investigador

- **PIC18F2455/2550/4455/4550**

Son microcontroladores de alto rendimiento, manejan registros de memoria de 8 bits, con características mejoradas respecto a la serie 16F; incorporan diversos periféricos más que los PIC® de gama media, a un costo razonable en el mercado local.

El diagrama de pines para la versión de 28 patillas de estos microcontroladores, es el estándar de facto que ha asumido Microchip para sus dispositivos, que se muestra en la **figura 6.17:**

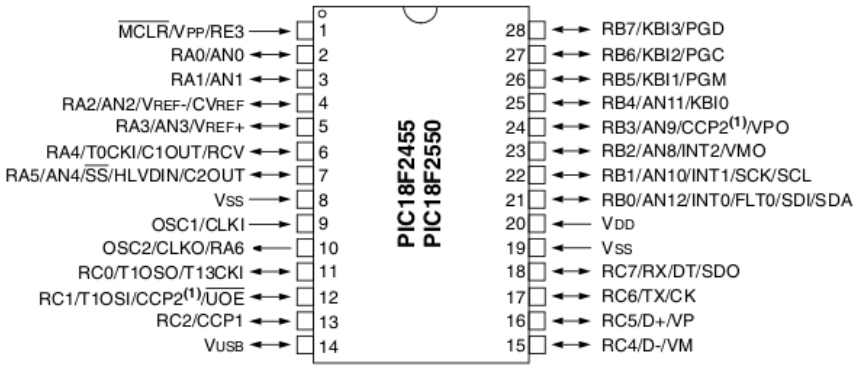



**Figura 6.17** Diagrama de pines 16F877A.
Fuente: PIC18F2455/2550/4455/4550 Datasheet.

La **tabla 6.3** muestra las características de la familia de microcontroladores PIC18F2455/2550/4455/4550:

**Tabla 6.3.** Características del dispositivo PIC18F2550

| **Características** | **PIC18F2455** | **PIC18F2550** | **PIC18F4455** | **PIC18F4550** |
|---|---|---|---|---|
| Frecuencia de operación | DC -48 Mhz | DC -48 Mhz | DC -48 Mhz | DC -48 Mhz |
| Memoria de programa (Bytes) | 24576 | 32768 | 32768 | 32768 |
| Memoria de programa (Instrucciones) | 12288 | 16384 | 12288 | 16384 |
| Memoria de datos (Bytes) | 2048 | 2048 | 2048 | 2048 |
| Memoria de datos EEPROM (Bytes) | 256 | 256 | 256 | 256 |
| Fuentes de interrupción | 19 | 19 | 20 | 20 |
| Puertos de Entrada/Salida | Puertos A, B, C, (E) | Puertos A, B, C, (E) | Puertos A, B, C, E | Puertos A, B, C, E |
| Temporizadores | 4 | 4 | 4 | 4 |
| Módulos Captura/Comparación/PWM | 2 | 2 | 2 | 2 |
| Módulos mejorados Captura/Comparación/PWM | 0 | 0 | 1 | 1 |
| Comunicaciones seriales | MSSP, USART Mejorado | MSSP, USART Mejorado | MSSP, USART Mejorado | MSSP, USART Mejorado |
| USB (Bus serial universal) | 1 | 1 | 1 | 1 |
| Puerto paralelo de transmisión (SPP) | No | No | Si | Si |
| Módulo Analógico a Digital de 10 Bits | 10 Canales de entrada | 10 Canales de entrada | 13 Canales de entrada | 13 Canales de entrada |
| Comparadores | 2 | 2 | 2 | 2 |

| | | | | |
|---|---|---|---|---|
| Restablecer (y Retardo) | POR, BOR, Instrucción RESET, pila completa, desbordamiento de pila, (PWRT, OST), MCLR opcional, WDT | POR, BOR, Instrucción RESET, pila completa, desbordamiento de pila, (PWRT, OST), MCLR opcional, WDT | POR, BOR, Instrucción RESET, pila completa, desbordamiento de pila, (PWRT, OST), MCLR opcional, WDT | POR, BOR, Instrucción RESET, pila completa, desbordamiento de pila, (PWRT, OST), MCLR opcional, WDT |
| Detección de Bajo Voltaje Programable | Si | Si | Si | Si |
| Restablecimiento por caída de tensión programable | Si | Si | Si | Si |
| Juego de instrucciones | 75 Instrucciones; 83 con el Juego de instrucciones extendido habilitado | 75 Instrucciones; 83 con el Juego de instrucciones extendido habilitado | 75 Instrucciones; 83 con el Juego de instrucciones extendido habilitado | 75 Instrucciones; 83 con el Juego de instrucciones extendido habilitado |
| Encapsulados | 28 pines PDIP<br>28 pines SOIC | 28 pines PDIP<br>28 pines SOIC | 40 pines PDIP<br>44 pines QFN<br>44 pines TQFP | 40 pines PDIP<br>44 pines QFN<br>44 pines TQFP |

Fuente: PIC18F2455/2550/4455/4550 Datasheet
Traducción: El investigador

Como se puede observar, las implementaciones de sistemas embebidos que se desarrollan con estos dispositivos, pueden tener exigencias bastantes más altas que con uno de gama media. El módulo que le ha dado la popularidad a este micro, es el USB integrado, que proporciona una vía de comunicación estandarizada para la gran mayoría de dispositivos actuales, a un costo razonablemente bajo.

- **Conjunto de Instrucciones**

El conjunto de instrucciones para esta familia, según Microchip (2009, pág 316, 317, 318), en la Hoja de especificaciones de los PIC18F2455/2550/4455/4550, consiste en 75 palabras, más 8 palabras si se usa del conjunto extendido. Esta gama usa palabras de instrucción de 16 bits de longitud, excepto por cuatro instrucciones que usan dos locaciones de memoria.

Los sets de instrucciones han sido diseñados para ser compatibles, con las restricciones propias del uso de tecnologías nuevas. La mayoría de las 35 instrucciones enumeradas anteriormente sirven para implementaciones con

esta familia. Para un uso más complejo, u operaciones que lo requieran, las tablas completas pueden consultarse en las respectivas hojas de especificaciones.

### 6.6.10 Diseño de PCB

Desde los inicios de la electrónica moderna, se presentó la necesidad de interconectar elementos entre sí, para dar un tratamiento a las señales eléctricas, de tal forma que cumplan una función determinada, como la amplificación, o la representación de números binarios, entre muchas otras.

Se idearon varias maneras, entre otras la de pasar las patillas de los elementos por una tarjeta perforada y unirlas con cables; sin embargo, la técnica que se ha impuesto hasta nuestros días es la de diseñar un circuito impreso en material conductor, el cual brinda la conectividad eléctrica entre elementos, que se encuentra sobre un material no conductor, que provee el adecuado soporte mecánico.

El material conductor es generalmente una capa de cobre, cuyo espesor es expresado por el peso del cobre (onzas por pie cuadrado). En la práctica se consideran espesores mínimos que van desde los 5µm como lo muestra la **tabla 6.4**

**Tabla 6.4.** Requerimientos de la capa de cobre.

| **Tipo de cobre** | **Clase 1-3** |
|---|---|
| Capa de cobre inicial mínima - externa | 1/8 oz/ft2 (5µm) |
| Capa de cobre inicial mínima - interna | 1/4 oz/ft2 (9µm) |
| Filme inicial de cobre (semiaditivo) | 5µm |
| Filme final de cobre (aditivo) | 15-20µm |

Fuente: Generic Standard on Printed Board Design, página 21.

Traducción: El investigador.

El material no conductor se puede presentar de diversas formas, dependiendo del uso y las características del diseño electrónico. Para circuitos multicapa es

común el uso de resina, llamada comúnmente Pértinax, por ser su nombre comercial. En circuitos rígidos se usan materiales de fibras de vidrio, conocidos como FR4; siendo también de uso extendido el compuesto plástico conocido como baquelita; sin embargo el FR4 es de mayor durabilidad y brinda mejor soporte mecánico, aun siendo más difícil de mecanizar.

El concepto de diseño es simple, unir elementos de acuerdo a un diagrama eléctrico, mediante pistas de material conductor, como en la **figura 6.18:**

**Figura 6.18** Circuito impreso diseñado de forma manual.
Autor: El investigador

Éste es un diseño funcional, que cumple su cometido, sin embargo no tiene en cuenta las normas antes mencionadas por lo cual el trabajo no es de calidad y es poco probable que resulte comercialmente viable.

### 6.6.11 Estándares para el diseño de PCB

Los estándares nacen a partir de la necesidad de contar con elementos que sean compatibles, y diseños que sirvan a funcionalidades generales, reduciendo el tiempo de desarrollo a niveles aceptables económica y prácticamente. De no existir estándares, existirían tantas formas y medidas de los componentes y PCB, como fabricantes hay en el mundo; haciendo extremadamente difícil la socialización de la tecnología en general.

A través del tiempo, se han formado varias asociaciones e institutos para la normalización o estandarización en el diseño de PCB, algunos de ellos tan conocidos como ANSI (*American National Standards Institute*), o la EIA

(*Electronic Industries Alliance*), que han generado grupos de normas para el diseño de PCB.

Uno de los conjuntos de recomendaciones más difundidos es el del IPC (*Institute for Printed Circuits*), que mantiene sedes alrededor de todo el mundo; cuyas recomendaciones son compatibles con casi todos los demás estándares, razón por la cual han sido seleccionadas para guiar el diseño de la presente placa de entrenamiento.

La serie correspondiente al diseño de circuitos impresos rígidos es la IPC-2220, siendo el conjunto de normas IPC-2221*"Generic Standard on Printed Board Design"* la norma específica que contiene las directrices necesarias para la placa de circuito impreso aquí propuesta.

### 6.6.12 Estándar IPC-2221

Éste es un estándar producido por el IPC o "*The institute for interconnecting and packaging electronic circuits*", en el cual se trata de establecer requerimientos genéricos para el diseño de placas de circuito impreso. La sección siguiente es un resumen del mencionado estándar, que abarca las partes a ser aplicadas en el diseño. Además es una traducción por parte del investigador, libre pero apegada a lo que predica el original.

- **Disposición de Elementos**

Las recomendaciones hechas por el IPC-D-275 Task Group, (1998, pág. 12):

- Asegurarse de que los componentes tengan puntos de prueba accesibles.
- Tener agujeros de alimentación y componentes a una distancia prudente del borde, para tener holgura suficiente
- Que la placa de producción sea fabricada con el mismo tipo de medidas utilizado en el diseño.
- Planificar que las diversas partes del diseño se puedan aislar para facilitar pruebas y diagnósticos.

- Cuando sea posible, agrupar puntos de prueba y jumpers.
- Considerar poner los componentes en zócalos para que puedan ser fácilmente reemplazados.

- **Holgura**

Este apartado se refiere al espacio de material no conductivo que debe existir entre las pistas y entre ellas y los planos de señales, como lo muestra la **figura 6.19**.

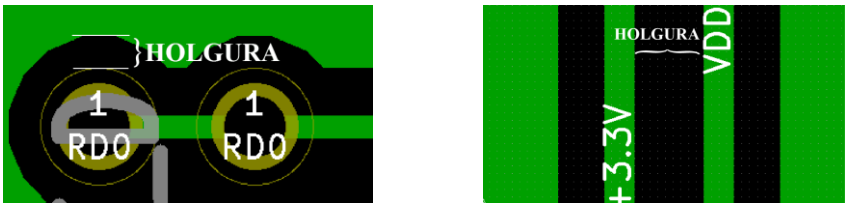



**Figura 6.19** Izquierda: Holgura entre pista y plano. Derecha: Holgura entre pistas
Autor: El investigador

De acuerdo las normas del IPC-D-275 Task Group (1998, pág 39): "El espacio entre los conductores debe maximizarse en donde quiera que sea posible" pero también, provee la **tabla 6.5** para espacios mínimos entre conductores:

**Tabla 6.5.** Espacio mínimo entre conductores (holgura).

| **Voltaje entre conductores (en DC o picos de AC)** | **Espacio mínimo** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Placa Desnuda** | | | | Ensamblaje | | |
| | **B1** | **B2** | **B3** | **B4** | **A5** | **A6** | **A7** |
| 0 - 15 | 0,05 mm | 0,1 mm | 0,1 mm | 0,05 mm | 0,13 mm | 0,13 mm | 0,13 mm |
| 16 - 30 | 0,05 mm | 0,1 mm | 0,1 mm | 0,05 mm | 0,13 mm | 0,25 mm | 0,13 mm |
| 31 - 50 | 0,1 mm | 0,6 mm | 0,6 mm | 0,13 mm | 0,13 mm | 0,4 mm | 0,13 mm |
| 51 - 100 | 0,1 mm | 0,6 mm | 1,5 mm | 0,13 mm | 0,13 mm | 0,5 mm | 0,13 mm |
| 101 150 | 0,2 mm | 0,6 mm | 3,2 mm | 0,4 mm | 0,4 mm | 0,8 mm | 0,4 mm |
| 151 - 170 | 0,2 mm | 1,25 mm | 3,2 mm | 0,4 mm | 0,4 mm | 0,8 mm | 0,4 mm |
| 171 - 250 | 0,2 mm | 1,25 mm | 6,4 mm | 0,4 mm | 0,4 mm | 0,8 mm | 0,4 mm |
| 251 - 300 | 0,2 mm | 1,25 mm | 12,5 mm | 0,4 mm | 0,4 mm | 0,8 mm | 0,8 mm |
| 301 - 500 | 0,25 mm | 2,5 mm | 12,5 mm | 0,8 mm | 0,8 mm | 1,5 mm | 0,8 mm |
| > 500 | 0,0025 mm/volt | 0,005 mm/volt | 0,0025 mm/volt | 0,00305 mm/volt | 0,00305 mm/volt | 0,00305 mm/volt | 0,00305 mm/volt |

B1 Conductores internos

| | |
|---|---|
| B2 | Conductores externos, descubiertos, desde el nivel del mar hasta 3050 m |
| B3 | Conductores externos, descubiertos, sobre 3050 m |
| B4 | Conductores externos, con recubrimiento de polímero a cualquier altura |
| A5 | Conductores externos, con recubrimiento hecho después del ensamblaje a cualquier altura |
| A6 | Componentes externos, con terminación de plomo, descubiertos |
| A7 | Componentes externos, con terminación de plomo, y recubiertos |

Fuente: Generic Standard on Printed Board Design, página 39.

Traducción: El investigador.

- **Perforaciones**

Todo el material circundante a las perforaciones debe ser maximizado en donde sea posible y estar de acuerdo con la holgura especificada más arriba. Para determinar el ancho mínimo de material, se debe hacer con la **ecuación 6.7**:

$$Ancho\ mínimo = a + 2b + c$$

**Ecuación 6.7** Ancho mínimo del área a perforar.

Fuente: Generic Standard on Printed Board Design, página 70.

En donde:

***a*** Diámetro mínimo del agujero.

***b*** Requerimientos mínimos del área a perforar (**tabla 6.6**).

***c*** Tolerancia estándar (**tabla 6.7**).

**Tabla 6.6.** Requerimientos mínimos del área a perforar.

| Anillo | Clase 1,2 y 3 |
|---|---|
| Apoyado interno | 0,03 mm |
| Apoyado externo | 0,05 mm |
| No apoyado externo | 0,15 mm |

Fuente: Generic Standard on Printed Board Design, página 71.

Traducción: El investigador.

**Tabla 6.7.** Requerimientos mínimos del área a perforar.

| **Nivel A** | **Nivel B** | **Nivel C** |
|---|---|---|
| 0,4 mm | 0,25 mm | 0,2 mm |

Para pesos de cobre mayores a 1 oz. añadir 0,05 mm por onza

Para más de 8 capas añadir 0,05 mm

Fuente: Generic Standard on Printed Board Design, página 70.

Traducción: El investigador.

Nivel A: Complejidad de diseño general – Preferida

Nivel B: Complejidad de diseño moderada – Estándar

Nivel C: Complejidad de diseño Alta – Reducida

- **Thermal Relief**

A falta de una traducción más adecuada, son "alivios térmicos", zonas en las cuales implica soldar componentes a planos grandes. Estas zonas deben estar unidas a la superficie grande por medio de pequeñas pistas, para que el calor que conduce el cobre, no dificulte la soldadura. Se muestran en la **figura 6.20,** y siempre deben estar presentes en estos casos.

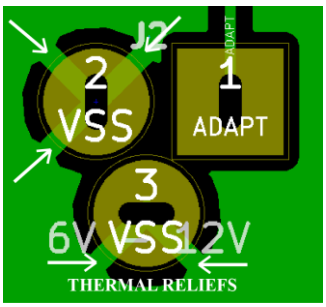



**Figura 6.20** *Thermal reliefs* o "alivios térmicos".

Autor: El investigador.

- **Características de los Conductores**

Las pistas en un circuito impreso pueden tomar una gran variedad de formas, tanto como simples conductores, hasta grandes planos de alimentación o señales. El IPC-D-275 Task Group (1998, pág. 73) dice: "El espesor de las pistas en la placa terminada, debe ser determinado en base a las

características de la señal.” Y también reza: “El espesor mínimo de un conductor terminado debe ser no menor a 0.1 mm”.

Las pistas deben ser lo más cortas posible, y privilegiar el enrutamiento en los ejes x e y, y los ángulos en 45 grados.

Sin embargo, el estándar provee la **tabla 6.8** de espesores mínimos:

**Tabla 6.8.** Espesor mínimo de los conductores externos.

| Capa de cobre base | Mínimo |
|---|---|
| 1/8 oz | 20 µm |
| 1/4 oz | 20 µm |
| 3/8 oz | 25 µm |
| 1/2 oz | 33 µm |
| 1 oz | 46 µm |
| 2 oz | 76 µm |
| 3 oz | 107 µm |
| 4 oz | 137 µm |

Para cada onza de cobre a partir de ahí, aumentar el espesor mínimo en 30,0 um

Fuente: Generic Standard on Printed Board Design, página 73.

Traducción: El investigador.

Después de consultar esta tabla de espesores mínimos, se debe tener en cuenta la corriente máxima que ha de circular por los conductores y el aumento máximo de temperatura en ellos.

El aumento máximo de temperatura está definido como la diferencia entre la temperatura segura de operación del material de la placa y el aumento máximo de temperatura del ambiente en donde funcionará la placa.

La **figura 6.21**, establece una relación entre el área de las pistas de cobre, el amperaje que ésta puede soportar, y las temperaturas de funcionamiento.

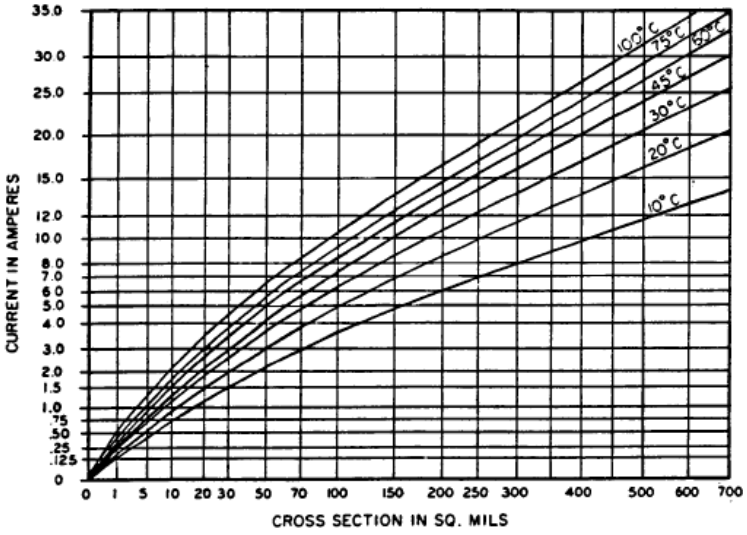



**Figura 6.21** Área en función de amperaje para conductores externos.
Fuente: Generic Standard on Printed Board Design, página 38.

La **figura 6.22** es el gráfico del ancho de pista en función del área de cobre, la misma determinada en la **figura 6.22**:

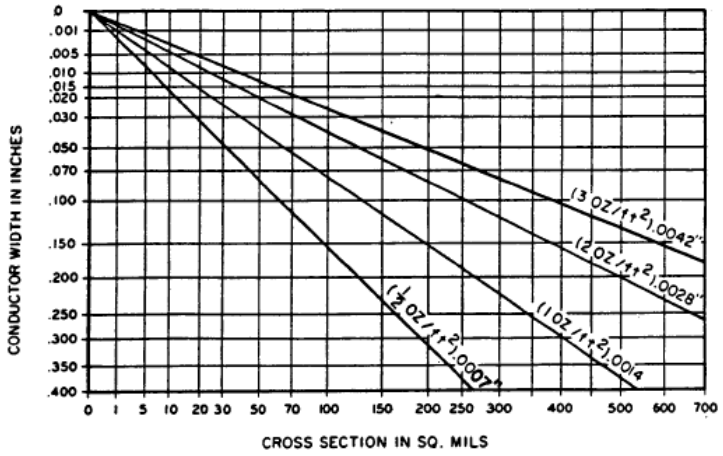



**Figura 6.22** Área en función del ancho de pista.
Fuente: Generic Standard on Printed Board Design, página 38.

Con esta información, el cálculo del ancho de las pistas es sencillo; se necesitan tres variables dimensionadas por el diseñador: el amperaje máximo, el aumento de temperatura y el espesor de la capa de cobre.

Se usa la **figura 6.21** para determinar el área de cobre en función del amperaje y el aumento de temperatura. Con este valor, se selecciona el ancho

de pista de la **figura 6.22,** en función del espesor de la capa de cobre específica de la placa a utilizar.

- **Documentación**

La documentación generalmente consiste en el dibujo principal de los circuitos, ya sea en papel o en formato digital; así como los esquemáticos, la lista de partes y el dibujo de las pistas.

### 6.6.13 Placas de Entrenamiento o Prototipo de Microcontroladores

En resumen, son placas, ya sea de circuito impreso, flexibles, perforadas, etc., que contienen los elementos necesarios para el funcionamiento de los microcontroladores. Generalmente se hacen para aplicaciones simples y estándar, aumentando su precio con la complejidad de implementaciones que se puedan realizar en ellas.

### 6.6.14 Módulo Principal

El diseño entero de este proyecto se basa en el concepto de modularidad. Esto significa que cada etapa puede ser interconectada con las otras para sumar funcionalidades. La manera en que se logró esto es utilizando un método de apilamiento, al estilo de cubos para armar. En la **figura 6.23** se muestra el conjunto de conectores lineales (en adelante *headers*) que permitieron hacer esto posible:

**Figura 6.23** *Headers* macho y hembra para conexión modular.
Autor: El investigador.

El módulo principal está dividido en diferentes zonas específicas. Ninguna de ellas es dependiente de otra, esto, con el objetivo de que pueda ser modificado para desarrollos específicos futuros.

- **Zona de Zócalos**

El módulo se diseñó para que sea capaz de albergar diversas variedades de PIC®. Aprovechando el hecho de que la gran mayoría de ellos son compatibles pin a pin, se instalaron dos zócalos interconectando las patillas equivalentes en los encapsulados DIP40 y DIP28 con los que trabaja Microchip.

En la **figura 6.24** se aprecia la parte el esquemático de esta zona:

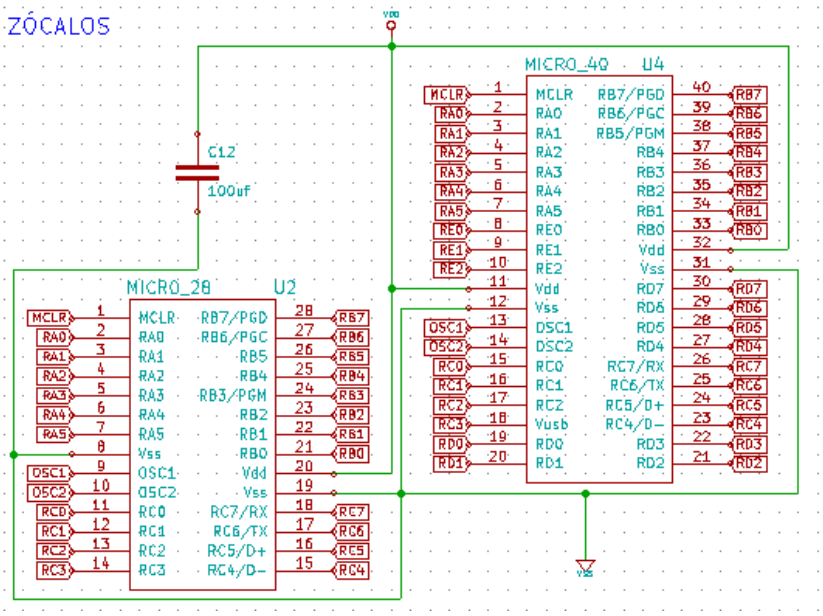



**Figura 6.24** Esquemático zona de zócalos.

Autor: El investigador.

Las etiquetas globales permiten liberar espacio y evitar marañas de cables, para hacer el diseño fácilmente entendible.

El capacitor de 100uf se utiliza para filtrar las líneas de alimentación de los integrados, y se debe ubicar lo más cerca posible de ellos.

- **Zona de Puertos**

Los microcontroladores tienen un conjunto de pines que hacen las veces de entradas o salidas según la capacidad física y la configuración lógica. A estos

se les llaman puertos. En el diseño se utilizan líneas dobles de *headers* para permitir a cualquier módulo comunicarse con el principal.

En la **figura 6.25** se muestra el esquemático de los puertos:

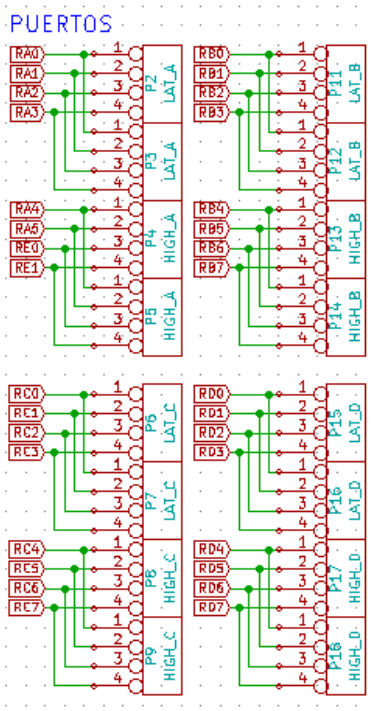



**Figura 6.25** Esquemático zona de puertos.

Autor: El investigador.

- **Zona de Pulsadores**

Una de las maneras de introducir datos en un pin de entrada es a través de un pulsador. En dependencia de la manera en que se ha conectado, se mantendrá un estado lógico bajo o cero lógico y cuando se active el pulsador, llevará al puerto a un estado lógico alto o uno lógico. Esta es la configuración que se ha escogido para el módulo, ya que un es un sistema más natural para quienes no tienen experiencia en sistemas embebidos.

En la **figura 6.26** se encuentra representada la zona de pulsadores

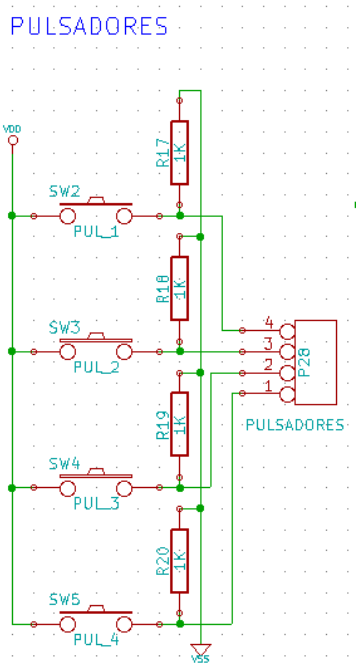



**Figura 6.26** Esquemático zona de pulsadores.

Autor: El investigador.

El cálculo del valor de la resistencia se hace en función de la corriente que ha de atravesar por ella. Estas resistencias, ya que establecen una referencia a masa se denominan de *pull-down*. Para comunicarse con los pines de entrada, se dispone de un *header* hembra, con cuatro espacios, que deberán ser cableados a los respectivos puertos.

Para intercambiar la relación entre estados lógicos, basta con conectar la parte que está a VDD a VSS y viceversa. En este caso las resistencias se denominarán de *pull-up*.

- **Zona de Leds**

Uno de los indicadores visuales básicos en cualquier implementación electrónica es el diodo led.

La **figura 6.27** muestra la implementación de leds en cátodo común que se ha elegido para esta zona.

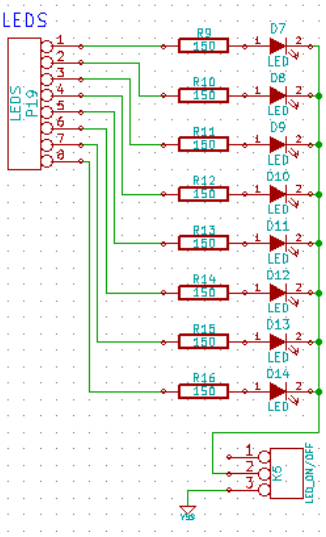



**Figura 6.27** Esquemático zona de leds.

Autor: El investigador.

Dispone de un jumper K5, para decidir si se conectan o no a tierra los cátodos, en función si se utilizan o no los leds.

El cálculo de las resistencias obedece a la aplicación de la ley de Kirchhoff, en el circuito de la **figura 6.28**:

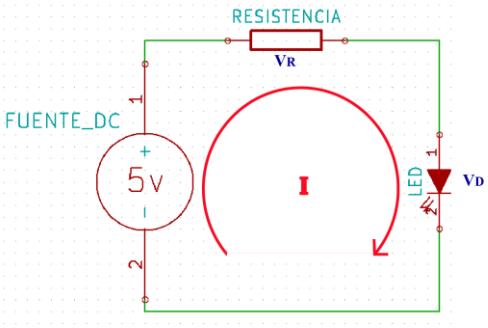



**Figura 6.28** Circuito serie para cálculo de la resistencia.

Autor: El investigador.

Según Sadiku & Alexander, (2004, pág. 39), la ley de voltajes de Kirchhoff reza: "La suma algébrica de todos los voltajes en un circuito cerrado es igual a cero". Aplicando este postulado se tiene que:

$$v_{FUENTE} - v_R - v_D = 0$$

Reemplazando y reordenando:

$$v_{FUENTE} - IxR - v_D = 0$$

$$I = \frac{(v_{FUENTE} - v_{caida\ en\ led})}{R}$$

**Ecuación 6.8** Cálculo de la corriente Aplicando ley de voltajes de Kichhoff

Despejando R:

$$R = \frac{(v_{FUENTE} - v_{caida\ en\ led})}{I}$$

Haciendo un compendio de las especificaciones generales para los leds verdes de 3 mm utilizados en este módulo, éstos necesitan una corriente desde 10 a 20 mA para funcionamiento óptimo, a un voltaje directo promedio de 2.2 v, entonces:

$$R = \frac{(v - v_{caida\ en\ led})}{I}$$

$$R = \frac{(5 - 2.2)v}{20\ mA}$$

$$R = 140\ \Omega$$

Siendo éste el valor óptimo de la resistencia a 5v. Ahora se realizará el cálculo para 3.3 v:

$$R = \frac{(3.3 - 2.2)v}{20\ mA}$$

$$R = 55\ \Omega$$

Teniendo en cuenta que a menor resistencia, mayor será la intensidad de corriente, se consultan los valores comerciales de resistencias, para elegir un valor nominal igual o superior al calculado, siendo apropiado el de 150 Ω, que aplicando la **ecuación 6.8**, hará circular aproximadamente 18,7 mA a 5v

y casi 8 mA a 3.3 v; capaces de proporcionar al led un brillo adecuado, y suficientemente bajo para no acortar demasiado la vida útil del elemento.

Aplicando este procedimiento, con valores normales para los colores de leds utilizados en el presente trabajo, se tiene la **tabla 6.9:**

**Tabla 6.9.** Valores de resistencias para leds

| Color | Valor típico de voltaje (v) | Valor típico de corriente (A) | Resistencia calculada 5v | Resistencia calculada 3.3v | Resistencias comerciales al 5% (Ω) | | Resistencia óptima | Corriente calculada 5v (A) | Corriente calculada 3,3v (A) |
|---|---|---|---|---|---|---|---|---|---|
| Verde | 2,2 | 0,02 | 140 | 55 | 33 | 150 | 150 | 0,019 | 0,007 |
| Amarillo | 2,1 | 0,025 | 116 | 48 | 33 | 120 | 120 | 0,024 | 0,010 |
| rojo | 2 | 0,025 | 120 | 52 | 33 | 120 | 120 | 0,025 | 0,011 |

Autor: El investigador

- **Zona de Fuentes de Voltaje**

En la etapa de alimentación del circuito, existen dos opciones; conectar un adaptador de 6 a 12 voltios y máximo un amperio; o bien, conectar directamente un cable USB al conector tipo B en el módulo principal. Para la segunda opción hay que tener muy en cuenta el consumo de la aplicación que se vaya a implementar, debido a que los mencionados puertos tienen una corriente máxima soportada.

Para la mayoría de placas se habla de valores alrededor de 500 mA, esto es cierto para el controlador raíz de puertos, estando dividido para las entradas USB existentes. Dicho esto, se recomienda limitar la implementación a un consumo de 200 mA.

La **figura 6.28** detalla la zona de alimentación:

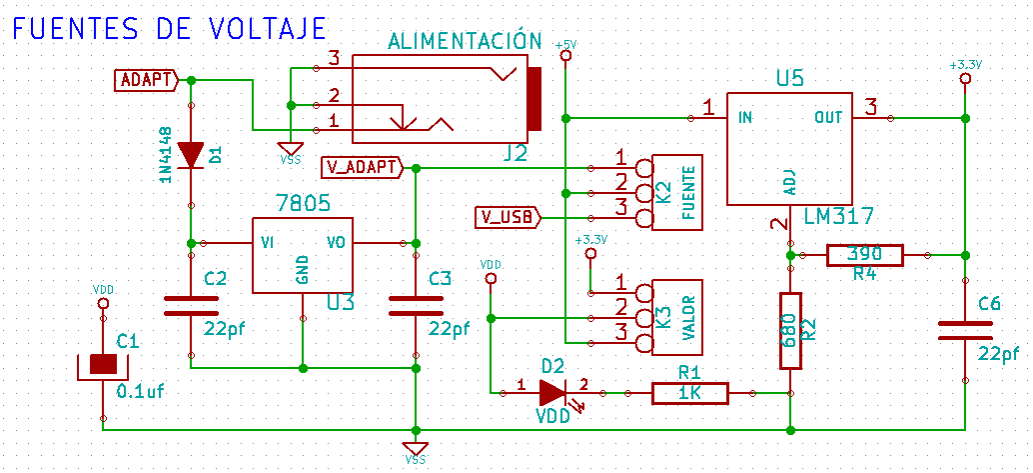



**Figura 6.29** Esquemático zona de fuentes de voltaje.

Autor: El investigador.

Cambiando de posición el jumper *K2*, se selecciona si el módulo va a ser alimentado por adaptador o por USB. Ambos entregan al circuito dos valores de voltaje, habituales en aplicaciones con microcontroladores.

La fase alimentada por adaptador, se vale de un LM7805 para entregar 5v de corriente continua, cuyas características se han explicado en un apartado anterior.

Si trabaja con la alimentación del USB se utiliza directamente para 5 v. El jumper *K3* sirve para conmutar entre una tensión de alimentación de ese valor o 3,3 v para la alimentación en el resto del circuito; esto para maximizar el número de aplicaciones que se puedan implementar en el prototipo.

Los 3,3 v se consiguen utilizando el integrado LM317, que tiene resistencias conectadas de acuerdo a la hoja de datos, y cuyos valores se calculan con la Ecuación 6.6 como se muestra a continuación y en la **figura 6.30**:

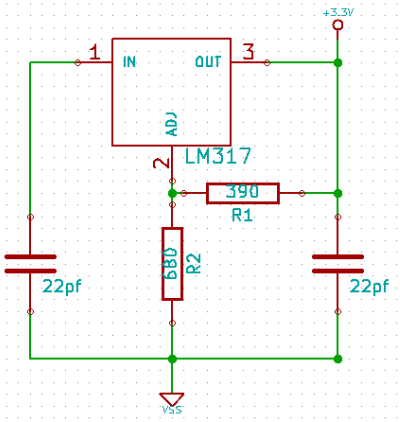



**Figura 6.30** Conexión del LM317 para 3.3 v a la salida.

Autor: El investigador.

$$v_{OUT} = 1.25v\left(1 + \frac{R2}{R1}\right)$$

Para 3,3 voltios se despeja cualquiera de las resistencias, en este caso R1:

$$R1 = \left(\frac{1.25v * R2}{v_{OUT} - 1.25v}\right)$$

En este punto, es necesario buscar un valor comercial de resistencia para asignar a R2 y calcular un valor aproximado para R1. Por conveniencia se escoge 680 Ω, entonces:

$$R1 = \left(\frac{1.25v * 680\ \Omega}{3.3v - 1.25v}\right)$$

$$R1 = 414{,}63\ \Omega$$

Los valores comerciales más cercanos son los de 390 Ω y 430 Ω. Sin embargo se escoge el primero para asegurarse que exista voltaje suficiente para aplicaciones de 3.3 v.

- **Zona de Comunicación RS232**

Esta zona se basa en un MAX232 para convertir los niveles de voltaje TTL (5 v) a niveles de voltaje del estándar RS232 (9 a 12 v).

La **figura 6.31** consta de un esquemático de esta zona:

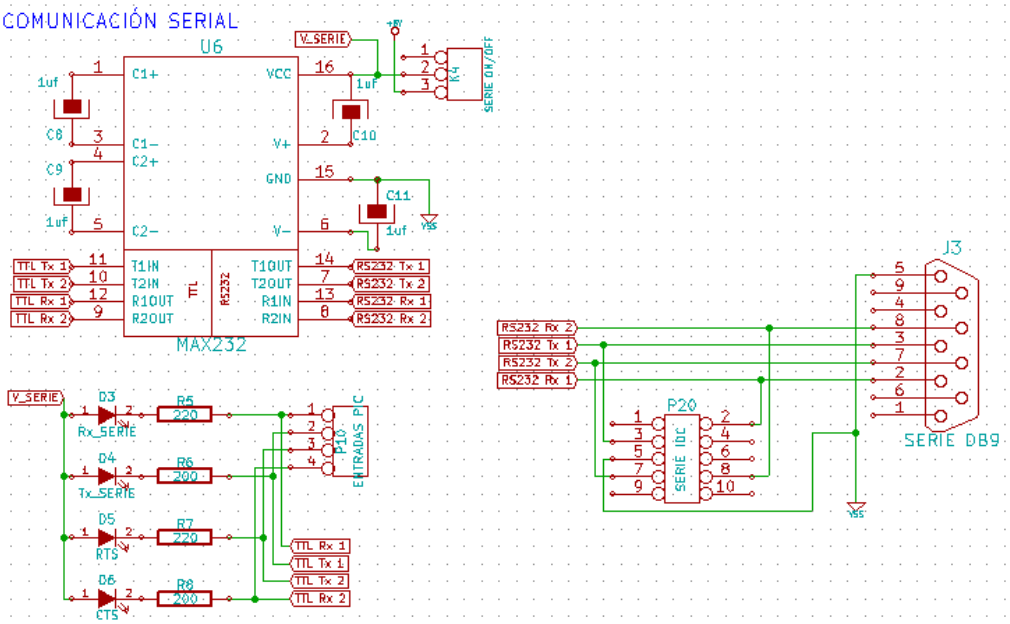



**Figura 6.31** Esquemático de zona RS232.

Autor: El investigador.

Como ya se dijo la parte principal es un MAX232, conectado de acuerdo a su respectiva hoja de datos con capacitores de 1uf a 25 v, para su funcionamiento óptimo. La configuración del puerto es DTE (equipo terminal de datos) por lo cual se usa un conector DB9 macho para la placa. Esto se hizo con la finalidad de contar con un COM virtual manejado por el microcontrolador, en caso de querer utilizarlo como interfaz de PC.

Posee además un *header* hembra, para conectar la respectiva salida del PIC®, dependiendo si se usa el USART o comunicación por software. Una salida estándar IDC10 completa el conjunto; las salidas de la misma son equivalentes al conector macho.

Finalmente, se tiene un led por cada entrada y salida de comunicación, lo cual es especialmente útil a la hora de depurar errores y visualizar la comunicación de manera didáctica.

- **Zona de Extensiones**

Esta es una zona común para todos los módulos, de manera que extienden al nivel superior la alimentación, brindan soporte mecánico y poseen un anillo conectado a VSS para facilitar las pruebas. La **figura 6.32** muestra esta zona:

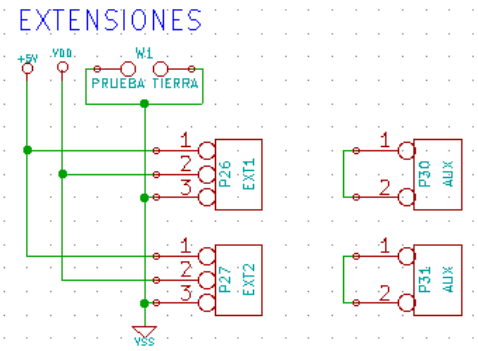



**Figura 6.32** Esquemático de zona RS232.

Autor: El investigador.

Ext1 y Ext2 tienen VSS; la alimentación que haya sido escogida con *K3*, además de los 5v de la fuente seleccionada con *K2*.

Los conectores marcados como AUX, sirven para brindar soporte mecánico en la esquina superior izquierda de las placas.

- **Potenciómetro**

Es un potenciómetro incorporado, que puede servir para simular entradas analógicas. La **figura 6.33** lo muestra:

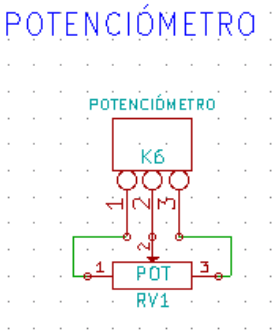



**Figura 6.33** Esquemático de conexión del potenciómetro.

Autor: El investigador.

- **Zona de USB**

Como lo muestra la **tabla 6.2,** los microcontroladores de la serie 18F, tienen incorporados módulos de comunicación USB, cuyo hardware se encarga de estabilizar las señales hasta para comunicación USB 2.0. La implementación es bastante simple en hardware y está en la **figura 6.34**:

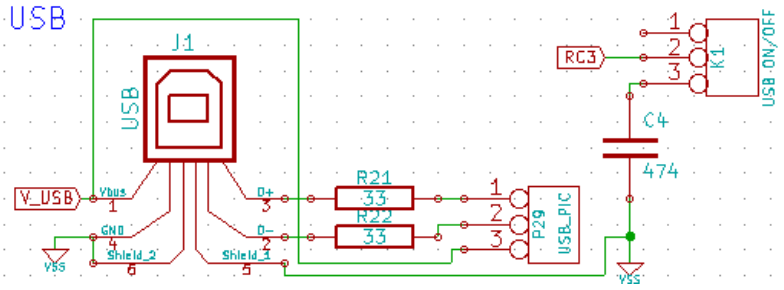



**Figura 6.34** Esquemático de zona USB.

Autor: El investigador.

El conector USB tiene un pin D+ y uno D-, conectados a *R21* y *R22*, que son resistores de pull-up externos, por lo cual no se hará uso de los resistores internos del microcontrolador, esto para simplificar la programación al máximo. Las conexiones llegan a un *header*, desde donde deberán ser cableados directamente a sus equivalentes en el PIC®.

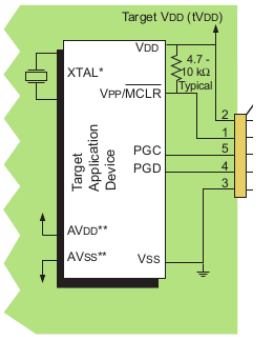
Además es necesaria la conexión de un capacitor a tierra en el pin Vusb para que el regulador interno trabaje a los 3.3v del estándar, circunstancia que se ha aprovechado para colocar el jumper K1¸que activa o no la comunicación USB.

- **Zona de ICSP (*In-Circuit Serial Programming*)**

O programación serial en el circuito, es un estándar para programar microcontroladores introducido por Microchip Inc., pero utilizado en sus propias versiones por casi todos los fabricantes; que permite programar el dispositivo sin removerlo de la tarjeta de la aplicación.

Esto es especialmente útil a la hora del desarrollo, debido a que los integrados no salen fácilmente de sus zócalos, y pueden arruinarse en el proceso.

La **figura 6.35** muestra la conexión típica en un circuito de aplicación tomada del Pickit3 In-Circuit Serial Debugger Poster de Microchip, (2008):




| Pin | Signal |
|---|---|
| 1 | MCLR/VPP |
| 2 | VDD Target |
| 3 | Vss Ground |
| 4 | ICSPDAT/PGD |
| 5 | ICSPCLK/PGC |
| 6 | LVP |

**Figura 6.35** Utilización del ICSP.
Fuente: Pickit3 In-Circuit Serial Debugger Poster.

Aquí son mostrados tanto los pines que se utilizan, como la numeración de los mismos. La **figura 6.36** contiene la implementación en el proyecto:

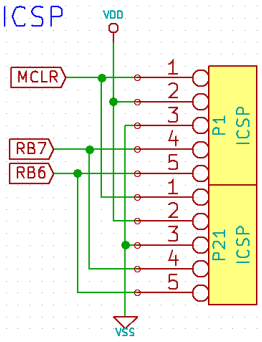



**Figura 6.36** Esquemático de zona ICSP.
Autor: El investigador.

- **Zona De Cristales**

Posee un zócalo troquelado, para que pueda ser insertado un cristal del valor conveniente para la aplicación o incluso para que los microcontroladores que tenga esa capacidad, puedan funcionar con su oscilador interno.

La **figura 6.37** contiene el esquemático:

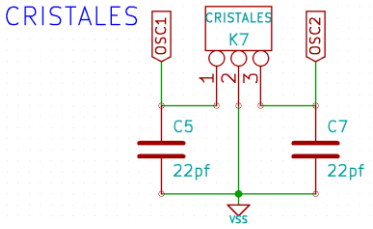



**Figura 6.37** Esquemático de zona de cristales.

Autor: El investigador.

Además, esta zona contiene dos capacitores C5 y C7, de valor bajo conectados a tierra, para filtrar el ruido que pudiera ocasionarse. El pin 2 de *K7* está a tierra para darle mayor sujeción al zócalo.

- **Zona de Botón de Reset**

Los PIC® tienen la función de reinicio cuando el pin MCLR es puesto a tierra. De hecho es necesario ponerlo en un estado alto para que el programa corra adecuadamente. Por ello se utiliza una resistencia de *pull-up* de 10k que es el valor recomendado por el fabricante.

Para un restablecimiento rápido, se ha incluido un pulsador que lleva momentáneamente MCLR a tierra. La **figura 6.38** muestra la zona del botón de reset.

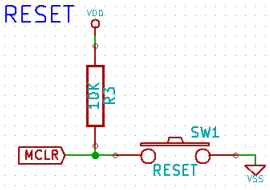



**Figura 6.38** Esquemático de zona de reset.

Autor: El investigador.

- **Consumo Total del Módulo**

Para determinar el consumo de cada módulo, se han utilizado los valores de sus respetivas hojas de especificaciones, en el peor de los escenarios, es decir,

cuando estén al máximo de su consumo. Se han incluido en el análisis los dispositivos de mayor consumo, para simplificar.

La **tabla 6.10** Contiene el consumo individual de cada elemento, así como el consumo total del módulo principal.

**Tabla 6.10.** Consumo del módulo principal.

| ITEM | DETALLE | CANTIDAD | UNIDAD | CONSUMO UNITARIO (mA) | SUBTOTAL (mA) |
|---|---|---|---|---|---|
| **6** | led 3 mm | 13 | c/u | 20 | 260 |
| **9** | Max232 | 1 | c/u | 30 | 30 |
| **11** | PIC18F2550 | 1 | c/u | 75 | 75 |
| | | | | **TOTAL (mA)** | 365 |

Autor: El investigador.

### 6.6.15 Módulo Programador

El programador es una versión reducida del Pickit 2, por ser uno de los diseños más simples que se han liberado para construir programadores USB. Consta de muy pocos elementos y por no ser el tema de este trabajo la construcción de un programador de microcontroladores; se hará solamente una breve reseña de su construcción.

El módulo ha sido realizado a partir de los esquemáticos de Microchip. Esta reducción consiste en implementar solamente las partes esenciales para la programación, que se detallan a continuación:

- **Microcontrolador**

El diseño se basa en un PIC18F2550, que corre el firmware que provee la empresa para este Programador, los pines están nombrados basándose en los esquemáticos que distribuye Microchip, (2008, pág. 77,78) en su *PICkit™ 2 USER'S GUIDE,* el cual gestiona la comunicación con el ordenador, y la programación con el microcontrolador objetivo, es decir, en el que se va a grabar el archivo correspondiente.

En la **figura 6.39,** consta el esquemático de las conexiones del PIC18F2550:

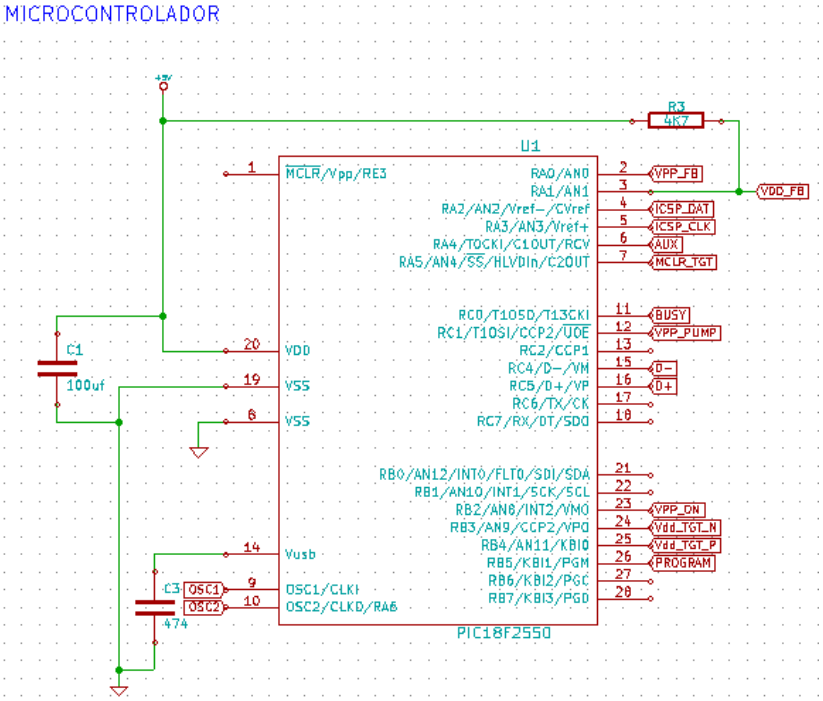



**Figura 6.39** Esquemático del PIC18F2550 para el programador.

Autor: El investigador.

- **Convertidor CC-CC**

Los PIC® objetivo necesitan una tensión de 13 a 14 voltios en el in MCLR para entrar en modo de programación, por lo cual se hace imprescindible un circuito que eleve los 5 v del puerto USB. Para ello se hace uso de un convertidor – elevador de voltaje CC.

La **figura 6.40,** es el esquemático de Microchip para elevar el voltaje en su *Pickit 2:*

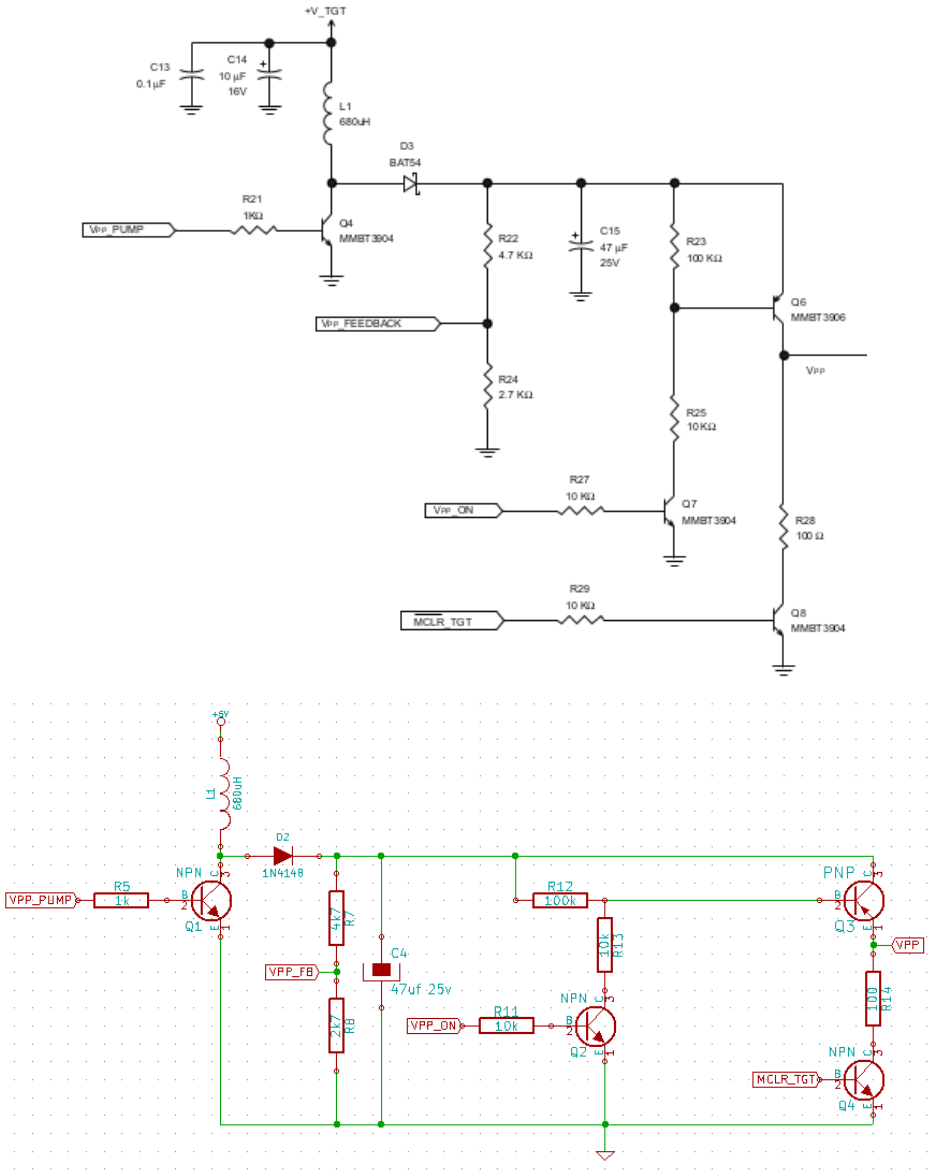



**Figura 6.40** Arriba: Esquemático del PIC18F2550 para el programador. Abajo: Diagrama del convertidor reducido

Fuente: PICkit™ 2 USER'S GUIDE e Investigador.

Para las referencias se utilizará la figura el diagrama reducido. La parte principal de este elevador, es el inductor ***L1*** de 680uH, conectado por una parte a los 5v del USB y la otra al transistor ***Q1***, el cual está controlado por un PWM generado en el PIC18F2550, alternando entre estado de saturación y corte, cuando enviamos la orden de programar desde el software.

Cuando ***Q1*** está en saturación, el voltaje en la bobina tiende a cero y la corriente tiende a incrementarse. Cuando ***Q1*** está en corte, la corriente no

desaparece instantáneamente sino que le toma cierto tiempo desvanecerse. Esto, producido en las rápidas transiciones del PWM, da lugar a picos de voltaje que superan los 13 voltios deseados.

El divisor de tensión formado por ***R7*** y ***R8***, censa estas variaciones para controlar a su vez el PWM. ***C4*** filtra esta señal. Si el voltaje es suficiente, el PIC activa el transistor ***Q2*** mediante la señal VPP_ON, lo cual a su vez hace que ***Q3*** entre en saturación, enviando el voltaje de programación al micro objetivo. MCLR_TGT, sirve para restablecer el objetivo mediante software, llevando su MCLR a un estado bajo.

Debido a que Microchip libera solamente el archivo hexadecimal para el PIC18F2550, es imposible saber a ciencia cierta cómo trabaja exactamente esta parte del programa; sin embargo, este es un buen resumen de esta etapa del circuito.

- **USB**

La conexión física del USB en la familia 18F es sencilla como se ha visto anteriormente. La **figura 6.41** es el esquemático de la etapa USB del módulo programador:

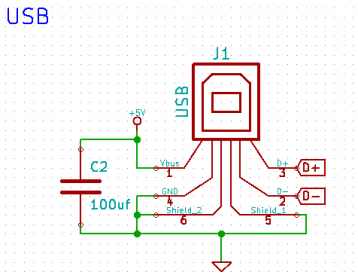



**Figura 6.41** USB del módulo programador

Autor: El investigador

El capacitor C2, es un filtro, ya que la salida de 5v y VSS alimentan todo el circuito.

- **Extensiones**

Esta es una parte común a todos los módulos, que permitirá la interconexión y el soporte mecánico entre ellos. El módulo programador solamente utiliza las señales del ICSP.

- **Salida ICSP**

La **figura 6.42** muestra las salidas en el orden estándar de Microchip, haciendo este módulo compatible para toda la gama de microcontroladores que soporta el Pickit 2 original.

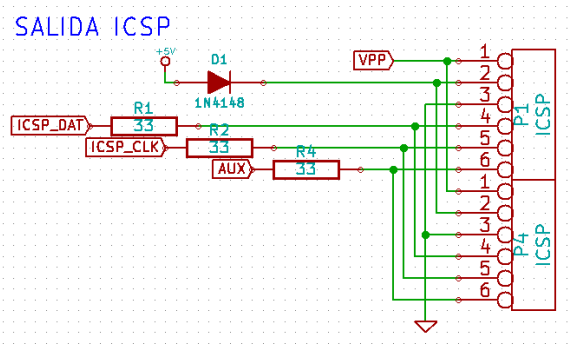



**Figura 6.42** Salida ICSP del programador

Autor: El investigador

La señal VPP ya ha sido explicada en un apartado anterior. Las señales ICSP_DAT, ICSP_CLK y AUX salen directamente del PIC18f2550; las resistencias ***R1, R2 Y R3***, sirven para asegurar que exista tensión en ellas respectivamente. El diodo ***D1,*** protege al circuito cuando existe alimentación en la placa objetivo.

- **Switch**

El pulsador ***SW1*** de la **figura 6.43** tiene dos funciones. En la versión original se usa para descargar el programa de las memorias EEPROM, que no han sido implementadas en el módulo programador y para que el PIC18F2550 entre en modo *bootloader* y poder actualizar el firmware. Esto se logra manteniendo presionado el pulsador mientras se conecta el programador al

puerto USB; el led BUSY quedará parpadeando para demostrar que está en modo *bootloader*.

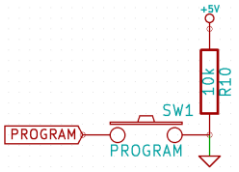



**Figura 6.43** Pulsador para entrar en modo *bootloader*.
Autor: El investigador

- **Indicadores**

Son tres leds de que muestran el estado de diferentes funciones del programador. En la **figura 6.44** se puede apreciar el esquemático correspondiente:

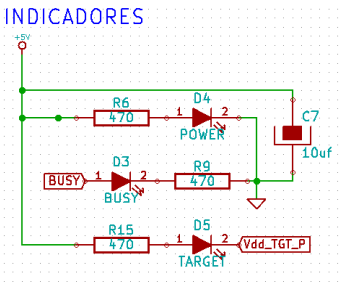



**Figura 6.44** Esquemático de los indicadores
Autor: El investigador

***D4 Power*** indica que el programador está alimentado a través del puerto USB. ***D3 Busy*** se enciende cuando el programador está ocupado en alguna tarea. ***D5 Target*** indica que el microcontrolador objetivo está alimentado a través del programador.

- **Cristal**

Para el funcionamiento del PIC18F2550 se utiliza un cristal de 20 Mhz, conectado con sus dos capacitores a tierra para filtrar posibles ruidos, como consta en la **figura 6.45:**

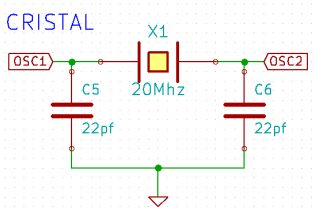



**Figura 6.45** Pulsador para entrar en modo *bootloader*.

Autor: El investigador

- **Consumo Total del Módulo**

La **tabla 6.11** muestra el consumo total de este módulo

**Tabla 6.11.** Consumo del módulo programador

| ITEM | DETALLE | CANTIDAD | UNIDAD | CONSUMO UNITARIO (mA) | SUBTOTAL (mA) |
|---|---|---|---|---|---|
| **3** | cristal | 1 | c/u | 0,2 | 0,2 |
| **7** | led 3 mm | 3 | c/u | 20 | 60 |
| **8** | PIC18F2550 | 1 | c/u | 25 | 25 |
| **12** | transistores | 4 | c/u | 20 | 80 |
| | | | | **TOTAL (mA)** | 165,2 |

.Autor: El investigador.

### 6.6.16 Módulo Pantalla de 7 Segmentos

Una de las maneras más extendidas de visualizar datos en una implementación con microcontroladores, es mediante la utilización de las pantallas de 7 segmentos.

- **Pantallas 7 Segmentos**

Son arreglos de leds, dispuestos de tal manera que se pueden formar números, y ciertas letras si son encendidos de forma adecuada. Los leds de los cuales están formados, tienen un extremo común, interconectados entre sí, ya sean ánodos o cátodos, dando lugar a las respectivas denominaciones *display ánodo común* o *display cátodo común.*

La **figura 6.46,** es una pantalla de 7 segmentos, cuya notación utiliza las letras de la ***a*** hasta la ***g*** *(o DP por Decimal Point)*,

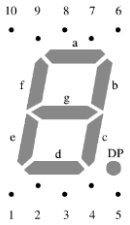



**Figura 6.46** Notación del display de 7 segmentos.
Fuente: Standard 7– Segment Display Datasheet, página 8.

En donde:

**1** Segmento e
**2** Segmento d
**3** Conector común (ánodo o cátodo)
**4** Segmento c
**5** Segmento g (o *DP*)
**6** Segmento b
**7** Segmento a
**8** Conector común (ánodo o cátodo)
**9** Segmento g
**10** Segmento f

- **Utilización**

La técnica es sencilla; se polarizan adecuadamente los segmentos para que aquellos que estén encendidos representen el símbolo deseado. Aquellos que tengan la configuración de cátodo común encenderán sus segmentos con un estado alto en el pin correspondiente y viceversa para los de ánodo común.

Como existen ocho canales de control por cada *display*, en caso de hacer una implementación con varios de ellos, se utilizarían demasiadas patillas del microcontrolador, por lo cual se acostumbra a utilizar diferentes técnicas; una de las cuales consiste en controlar cada pantalla con un registro de desplazamiento, la cual se ha usado en esta implementación.

- **CD4094BC**

Es un registro de desplazamiento de ocho bits, suficientes para controlar los ocho leds de un *display*, como consta en la **figura 6.47:**

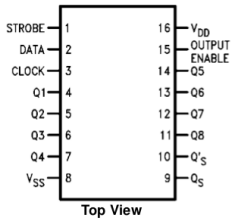



**Figura 6.47** Esquema de pines CD4094BC.
Fuente: CD4094BC 8-Bit Shift Register Datasheet, página 1

Utilizando solamente tres pines del microcontrolador (Strobe, Data y Clock), se puede controlar varios *displays*, conectando estos integrados en cascada, para obtener un registro de desplazamiento más grande. Esto se entiende mejor echando un vistazo a la **figura 6.48:**

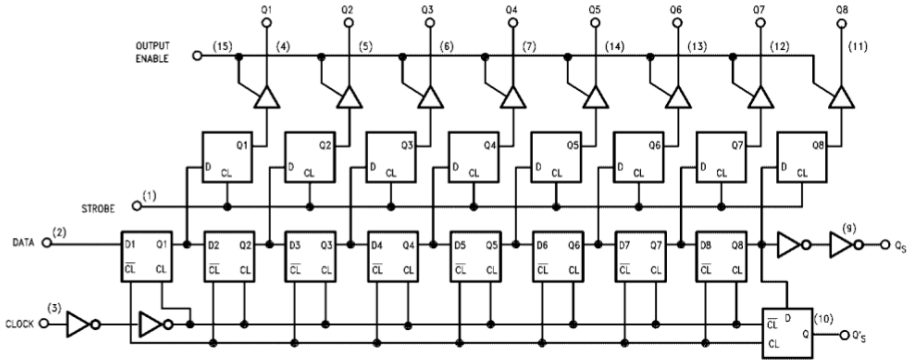



**Figura 6.48** Diagrama de bloques CD4094BC.
Fuente: CD4094BC 8-Bit Shift Register Datasheet, página 2

Conectando en la salida del último registro (QS) a Data del siguiente 4094, resulta un efecto cascada, el cual permite ampliar el registro. Por motivos de espacio, en este módulo, se han implementado tres conjuntos registro – *display*, por lo cual la programación deberá tomar en cuenta este registro de 24 bits.

El primer bit que sea enviado a través del pin Data va a ir pasando con cada pulso de reloj (en flanco de subida, o en lenguaje coloquial cuando el pin Clock pase de cero a uno). Entonces, el primer bit que el programa envíe, después de ocho transiciones positivas va a parar en Q8, el segundo en Q7 y así sucesivamente. Basta con extrapolar esta explicación para el análisis de los dispositivos en cascada.

- **Conjunto Display y Registro**

La **figura 6.49**, contiene la implementación que en el presente proyecto se ha hecho, de tres *displays* con su respectivo registro de desplazamiento:

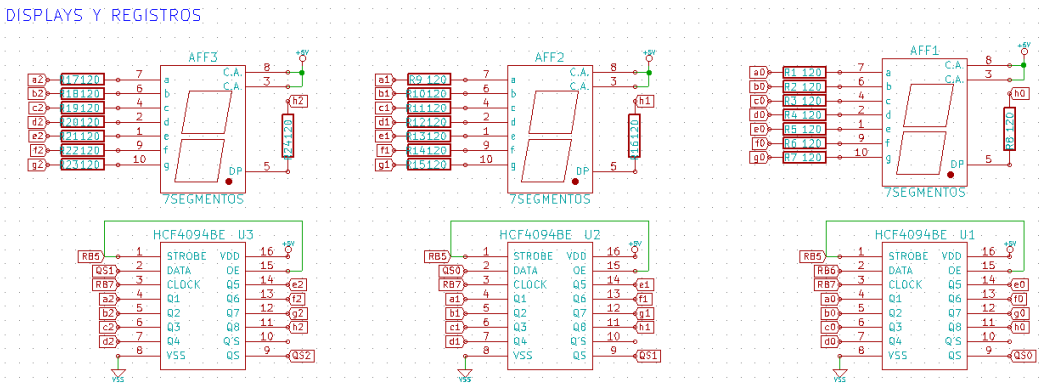



**Figura 6.49** Diagrama esquemático CD4094BC.

Autor: El investigador.

Comenzando desde la derecha para el número menos significativo, cada segmento es conectado a su respectivo pin del registro de la siguiente manera: a Q0, b a Q1 y así sucesivamente hasta h (*DP*) al pin Q7, resultando la **tabla 6.11** con los valores seriales que deberán ser enviados desde el microcontrolador, para *displays* cátodo común, cuyos segmentos se encenderán cuando el estado en el registro sea bajo, es decir cero lógico:

**Tabla 6.12** Valores para representar números en el módulo *Displays.*

| | **Último bit** | | | | | | **Primer bit** | | **Número** |
|---|---|---|---|---|---|---|---|---|---|
| **Registros** | **Q0** | **Q1** | **Q2** | **Q3** | **Q4** | **Q5** | **Q1** | **Q7** | |
| **Segmentos** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | |
| | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | **1** |
| | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | **2** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | **3** |
| | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | **4** |
| | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | **5** |
| | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | **6** |
| | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | **7** |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **8** |
| | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | **9** |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | **0** |

Autor: El investigador.

Finalmente, el cálculo de las resistencias se ha realizado de igual manera que con cualquier led; al consultar la hoja de especificaciones para el *display* rojo que se ha utilizado, la corriente recomendada es de 25 mA y 1.2 v, entonces aplicando la **ecuación 6.8**, es de aproximadamente 180 Ω por segmento.

- **Consumo Total del Módulo**

Los elementos ópticos pueden llegar a exigir una gran cantidad de corriente al circuito, así que cuando se construyan conjuntos más grandes en cascada, este hecho deberá tomarse en cuenta.

La **tabla 6.13** contiene el consumo de este módulo:

**Tabla 6.13.** Consumo del módulo.

| ITEM | DETALLE | CANTIDAD | UNIDAD | CONSUMO UNITARIO (mA) | SUBTOTAL (mA) |
|---|---|---|---|---|---|
| **1** | CD4094 | 3 | c/u | 0,64 | 1,92 |
| **2** | display 7 segmentos | 3 | c/u | 160 | 480 |
| | | | | **TOTAL (mA)** | 481,92 |

Autor: El investigador.

### 6.6.17 Módulo LCD

Una manera de superar las limitaciones de interfaz que tienen los sistemas embebidos, es incluir un LCD (Liquid Crystal Display), o pantalla de cristal líquido en la implementación. Esta clase de pantallas sirven para representar una mayor cantidad de caracteres alfanuméricos que los *displays*.

El funcionamiento, a grandes rasgos, se basa en la polarización de la luz que pasa a través de una serie de capas de material translúcido. Cuando están en

reposo, la luz del medio ambiente se refleja y es visible al ojo humano. Con una pequeña corriente la polarización de estas capas cambia y no permite reflejar la luz, por lo cual el píxel no enciende. El aprovechamiento de este efecto, en un patrón adecuado, permite formar caracteres. De todo esto se encargan circuitos y memorias incluidas en el dispositivo mismo.

Utilizan generalmente un driver Hitachi HD44780U, y una memoria ROM en donde son almacenados los caracteres, en concreto, según su hoja de especificaciones, son 240 en total. Sin embargo, cabe decir que se puede generar cualquier animación que soporte el LCD desde el microcontrolador de la aplicación.

En la **figura 6.50,** se puede apreciar la distribución de pines típica de un LCD comercial:

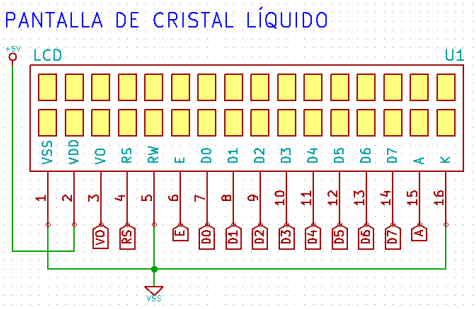



**Figura 6.50** Diagrama esquemático LCD.

Autor: El investigador.

En donde:

| | | |
|---|---|---|
| **1** | **Vss** | Común o tierra. |
| **2** | **Vdd** | Alimentación. |
| **3** | **Vo** | Ajuste de contraste |
| **4** | **RS** | Selección de registro de datos. (Alto o bajo) |
| **5** | **RW** | Selección de modo lectura o escritura. |
| **6** | **Enable** | Señal de activación |
| **7** | **Data 0** | Bus de datos. |
| **8** | **Data 1** | Bus de datos. |

| | | |
|---|---|---|
| **9** | **Data 2** | Bus de datos. |
| **10** | **Data 3** | Bus de datos. |
| **11** | **Data 4** | Bus de datos. |
| **12** | **Data 5** | Bus de datos. |
| **13** | **Data 6** | Bus de datos. |
| **14** | **Data 7** | Bus de datos. |
| **15** | **Ánodo** | Negativo de iluminación de fondo. |
| **16** | **Cátodo** | Positivo de iluminación de fondo. |

Como el driver se encarga de los procesos internos para la generación de caracteres en la pantalla; en esta etapa, el microcontrolador tiene la tarea de enviar la información que se ha de representar, así como de habilitar el cambio en el estado de las capas con los datos nuevos. Además, tiene un línea que le permite leer los datos que están en el *display*, y un control de estado para esta función.

- **Puertos**

La mayor parte de los lenguajes de alto nivel para microcontroladores, tienen librerías específicas para manejar LCD; también es posible una implementación de bajo nivel, sin importar el método escogido, en casi todos los casos se permite cambiar los pines designados para las diversas tareas que supone poner en funcionamiento un LCD.

La **figura 6.51** pone de manifiesto la utilización del puerto B del microcontrolador en la presente implementación, ya que es este puerto el que utilizan por defecto algunos de los lenguajes de programación preferidos por el investigador.

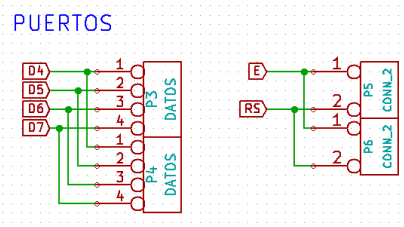



**Figura 6.51** Zona de conexión a puertos LCD.

Autor: El investigador.

Como se observa, en la mayoría de casos son suficientes seis líneas hacia el microcontrolador para utilizar un LCD de los más comunes. Sin embargo, se puede optar por utilizar todos los pines de datos, usar la línea de activación de lectura y escritura; o por el contrario, utilizar implementaciones electrónicas diversas, para utilizarlo aún con menos pines del microcontrolador.

- **Controles de Iluminación**

El contraste de los caracteres que representará el LCD, se controla variando la tensión en el pin 3 ***Vo***; esto puede hacerse por cualquier método, incluso con una resistencia que mantenga un voltaje fijo, o variarlo directamente desde el microcontrolador.

La intensidad de la luz de fondo depende de la misma lógica, aplicada al ánodo de dicha iluminación. En la **figura 6.52** se muestran el esquemático de los potenciómetros escogidos para variar la caída de tensión en estos dos pines del LCD.

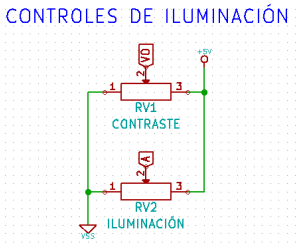



**Figura 6.52** Controles de iluminación.
Autor: El investigador

- **ICSP**

Se trata de una extensión para programación ICSP del microcontrolador en el módulo principal, conectando el respectivo programador.

- **Extensiones**

Son las que se encuentran en todos los módulos.

- **Consumo Total del Módulo**

La **tabla 6.14** muestra

**Tabla 6.14.** Consumo del módulo LCD.

| ITEM | DETALLE | CANTIDAD | UNIDAD | CONSUMO UNITARIO (mA) | SUBTOTAL (mA) |
|---|---|---|---|---|---|
| **3** | lcd | 1 | c/u | 260 | 260 |
| | | | | **TOTAL (mA)** | 260 |

Autor: El investigador.

### 6.6.18 Módulo Relevadores

En muchas de las aplicaciones en las cuales se usan microcontroladores, existe una atapa en la cual la potencia de la implementación es tan alta, que destruiría la parte de control si no estuviera aislada de ella.

- **Relé**

Para evitar situaciones como esta, se han desarrollado diversas técnicas; una de las principales es el uso de relés o relevadores. Estos consisten en interruptores magnéticos, cuya bobina puede ser controlada por un pin del microcontrolador.

La razón de hacerlo en dos etapas, es para no preocuparse de la corriente del dispositivo final que estará mecánicamente aislado de todo el circuito; aunque usando dispositivos de estado sólido se logra un efecto similar, la implementación de relés es mucho más sencilla. Otro factor es que en caso de falla catastrófica (cuando se destruye el dispositivo de estado sólido), las corrientes de la etapa de alta potencia podrían llegar a la etapa de control.

Haciendo un análisis de la **figura 6.53** se puede comprender el funcionamiento de un relé.

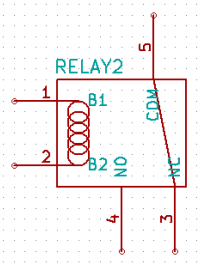



**Figura 6.53** Esquemático de un relé.

Autor: El investigador

Básicamente es un conjunto de dos interruptores, cuando está inactivo, uno permanece normalmente cerrado y el otro normalmente abierto; tienen un pin común a ambos, para conectar la aplicación objetivo. Esta situación da lugar a su nomenclatura. Se tiene el común **COM**, pin cinco en el gráfico; "Normalmente cerrado o **NC**" para el pin tres; y finalmente "Normalmente abierto o **NO**" para el pin cuatro.

Los pines uno y dos, al ser polarizados correctamente, generan un campo magnético alrededor de la bobina, que atrae la pestaña del interruptor, permitiendo alternar entre que el circuito esté cerrado por los pines cinco y tres, o entre cinco y cuatro.

- **ULN2003**

La bobina sin embargo, puede generar corriente suficiente para causar daños en la electrónica del microcontrolador, por lo cual es necesario protegerla. Con un transistor que maneje niveles TTL es suficiente. Existe también una corriente inversa, de la cual se hace cargo un diodo convenientemente colocado.

En lugar de implementarlo con componentes discretos, para el presente proyecto se ha elegido utilizar un circuito integrado ULN2003, específicamente diseñado para estos menesteres, compuesto por siete conjuntos de transistores en configuración Darlington, lo cual permite, según su hoja de especificaciones, manejar corrientes de hasta 600 mA por cada uno.

La **figura 6.54**, muestra el esquemático del interior de un ULN2003.

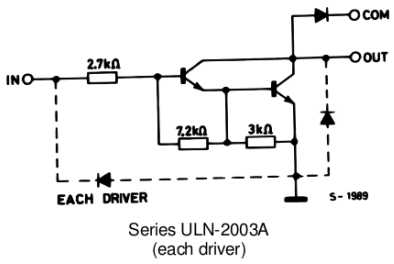



**Figura 6.54** Configuración interna de un conjunto Darlington en el ULN2003.
Fuente: ULN2003A Datasheet, página 2.

Se puede observar que incluye una resistencia de 2.7kΩ, que actúa como *pull-up*, un divisor de tensión que alimenta la base del segundo transistor del par Darlington, y un conjunto de diodos que protegen el montaje, así como una salida común, que en el caso de la placa entrenadora.

La **figura 6.55** es el diagrama de pines del ULN2003

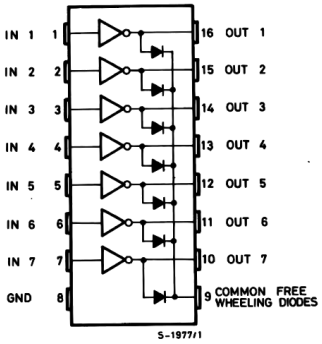



**Figura 6.55** Diagrama de pines del ULN2003.
Fuente: ULN2003A Datasheet, página 2.

Según su hoja de especificaciones, cuando este circuito integrado tiene un uno lógico en una entrada, tendrá cero lógico en la respectiva salida. El pin común se conectará según este criterio.

- **Relevadores**

La **figura 6.56** es la implementación de dos relés en el presente proyecto, utilizando un integrado ULN2003 para su manejo:

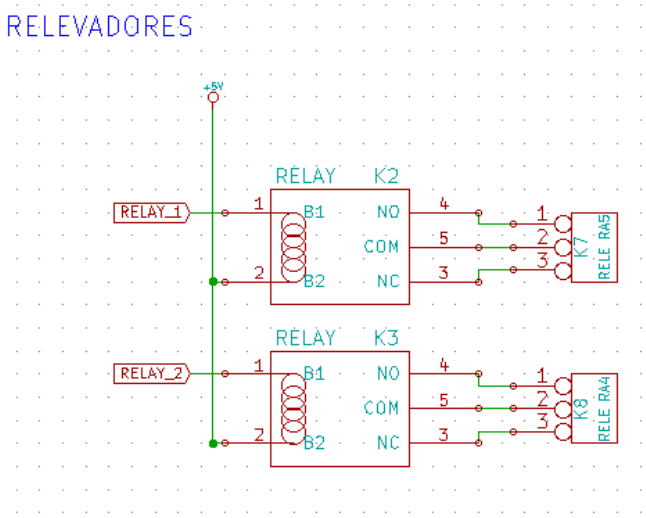



**Figura 6.56** Relés a ser manejados a través del ULN2003.

Autor: El investigador.

Una patilla de la bobina de cada relé está conectada al común del ULN2003, y la otra a una de sus salidas. El par de interruptores está conectado a una bornera, la cual permite fijar los cables del circuito objetivo.

- **Arreglos de Transistores**

La **figura 6.57** contiene el esquemático del ULN2003 utilizados para manejar los transistores

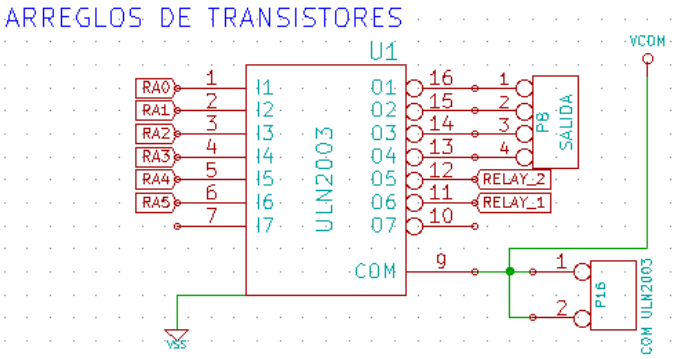



**Figura 6.57** ULN2003 en el módulo relevadores.

Autor: El investigador.

El módulo además, contiene otro integrado de este tipo, para un total de 11 arreglos Darlington libres, que pueden ser utilizadas para el control de otros

dispositivos de potencia, como motores a pasos o aumentar el número de relés.

Las borneras *P10* y P12, son los comunes del integrado uno, aquel que contiene los relés; y el integrado dos respectivamente. El jumper *K10* sirve para seleccionar si se utiliza o no el segundo integrado.

- **Alimentación Objetivo**

Para dotar de una mayor versatilidad al módulo, se le ha implementado un selector para la fuente del voltaje del Común de ambos integrados, lo cual permite trabajar con tensiones diferentes a las de la placa, conectando una fuente externa; o por el contrario, trabajar con los 5v del módulo principal.

La **figura 6.58** muestra la implementación

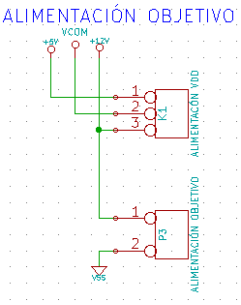



**Figura 6.58** Selector de fuente de alimentación del objetivo.
Autor: El investigador.

- **Puertos**

Para la comunicación por parte del microcontrolador, se han dispuesto los *headers* de tal manera que coincidan con la manera en la que están dispuestos en el módulo principal, como se muestra en la **figura 6.59**:

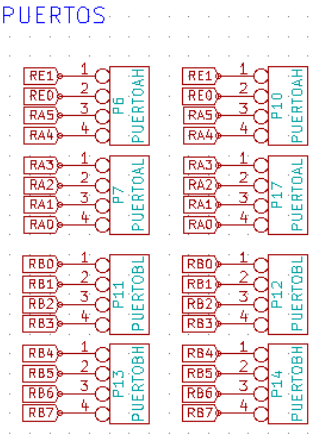



**Figura 6.59** Puertos utilizados para el módulo relevadores.

Autor: El investigador

De tal manera que el puerto B del microcontrolador queda accesible para cualquier aplicación.

- **Puente H**

El puente H es una implementación electrónica que permite controlar el sentido de giro de motores DC. Se utilizan transistores, de los cuales depende la potencia del motor que es capaz de manejar. En el módulo de relevadores, se han escogido los 2N3904 (NPN) y 2N3906 (PNP), capaces según su hoja de especificaciones, de manejar hasta 40 $v$ de voltaje entre Colector y Emisor y 200 $mA$ en su base.

La **figura 6.60** muestra la implementación del Puente H en el módulo de relevadores.

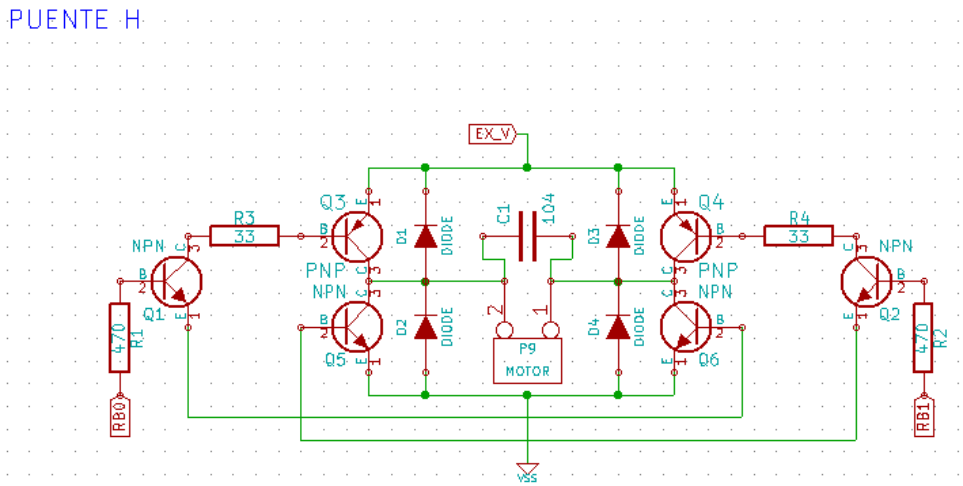



**Figura 6.60** Puente H.

Autor: El investigador

La explicación de su funcionamiento es bastante sencilla: cuando se aplica corriente a la base de Q1, éste junto con Q3 y Q6 saturan, cerrando el circuito y permitiendo que la corriente circule entre ellos, con el positivo del voltaje aplicado al terminal 2 del motor en sentido anti - horario. Lo contrario sucede cuando se aplica corriente a la base de Q2, saturando ahora Q2, Q4 y Q5, con lo cual el positivo se aplica al terminal 1 del motor y gira en sentido horario

- **Extensiones**

Las extensiones son las comunes a todos los módulos.

- **Consumo Total del Módulo**

La **tabla 6.15** muestra el consumo del módulo:

**Tabla 6.15.** Consumo del módulo Relevadores.

| ITEM | DETALLE | CANTIDAD | UNIDAD | CONSUMO UNITARIO (mA) | SUBTOTAL (mA) |
|---|---|---|---|---|---|
| **1** | relé | 3 | c/u | 400 | 1200 |
| **2** | transistores | 6 | c/u | 20 | 120 |
| **3** | ULN2003 | 2 | c/u | 25 | 50 |
| | | | | **TOTAL (mA)** | 1370 |

Autor: El investigador.

Debido al consumo de los relés, es recomendable no activar más de uno al tiempo cuando se usa la alimentación por USB.

### 6.6.19 Análisis Económico

Los precios de los materiales están en dólares y han sido consultados exclusivamente en el mercado local, al 13 de junio de 2013.

- **Módulo Principal**

La **tabla 6.16** es una lista de todos los materiales necesarios para construir el módulo principal, sus precios y el total a invertir en materiales.

**Tabla 6.16.** Lista de materiales módulo principal

| ITEM | DETALLE | CANTIDAD | UNIDAD | VALOR UNITARIO | VALOR DEL RUBRO |
|---|---|---|---|---|---|
| **1** | capacitor cerámico | 7 | c/u | $ 0,10 | $ 0,70 |
| **2** | capacitor electrolítico | 5 | c/u | $ 0,10 | $ 0,50 |
| **3** | diodo 1n4004 | 1 | c/u | $ 0,10 | $ 0,10 |
| **4** | header hembra | 3 | c/u | $ 0,75 | $ 2,25 |
| **5** | header macho | 1 | c/u | $ 0,60 | $ 0,60 |
| **6** | led 3 mm | 13 | c/u | $ 0,08 | $ 1,04 |
| **7** | lm 317 | 1 | c/u | $ 0,60 | $ 0,60 |
| **8** | lm 7805 | 1 | c/u | $ 0,50 | $ 0,50 |
| **9** | Max232 | 1 | c/u | $ 2,30 | $ 2,30 |
| **10** | PIC16F877A | 1 | c/u | $ 7,50 | $ 7,50 |
| **11** | PIC18F2550 | 1 | c/u | $ 11,24 | $ 11,24 |
| **12** | placa fibra FR4 | 0,5 | 50mmx100mm | $ 5,00 | $ 2,50 |
| **13** | Potenciómetro | 1 | c/u | $ 0,30 | $ 0,30 |
| **14** | pulsador | 5 | c/u | $ 0,15 | $ 0,75 |
| **15** | resistencias 1/4 w | 22 | c/u | $ 0,03 | $ 0,66 |
| **16** | USB B conector placa | 1 | c/u | $ 0,75 | $ 0,75 |
| **17** | zócalo 16 pines | 1 | c/u | $ 0,15 | $ 0,15 |
| **18** | zócalo 28 pines | 1 | c/u | $ 0,25 | $ 0,25 |
| **19** | zócalo 40 pines | 1 | c/u | $ 0,30 | $ 0,30 |
| | | | | **TOTAL** | $ 32,29 |

Autor: El investigador

- **Módulo Programador**

La **tabla 6.17** contiene los materiales del módulo programador.

**Tabla 6.17.** Lista de materiales módulo programador.

| ITEM | DETALLE | CANTIDAD | UNIDAD | VALOR UNITARIO | VALOR DEL RUBRO |
|---|---|---|---|---|---|
| **1** | bobina | 1 | c/u | $ 0,45 | $ 0,45 |
| **2** | capacitor electrolítico | 7 | c/u | $ 0,10 | $ 0,70 |
| **3** | cristal | 1 | c/u | $ 0,65 | $ 0,65 |
| **4** | diodo 1n4148 | 2 | c/u | $ 0,10 | $ 0,20 |
| **5** | header hembra | 1 | c/u | $ 0,75 | $ 0,75 |
| **6** | header macho | 1 | c/u | $ 0,60 | $ 0,60 |
| **7** | led 3 mm | 3 | c/u | $ 0,08 | $ 0,24 |
| **8** | PIC18F2550 | 1 | c/u | $ 11,24 | $ 11,24 |
| **9** | placa fibra FR4 | 0,25 | 50mmx100mm | $ 5,00 | $ 1,25 |
| **10** | pulsador | 1 | c/u | $ 0,15 | $ 0,15 |
| **11** | resistencias 1/4 w | 15 | c/u | $ 0,03 | $ 0,45 |
| **12** | transistores | 4 | c/u | $ 0,10 | $ 0,40 |
| **13** | USB B conector placa | 1 | c/u | $ 0,75 | $ 0,75 |
| **14** | zócalo 28 pines | 1 | c/u | $ 0,25 | $ 0,25 |
| | | | | **TOTAL** | $ 17,63 |

Autor: El investigador.

- **Módulo Pantallas de 7 Segmentos**

La **tabla 6.18** contiene los materiales del módulo de pantallas de 7 segmentos:

**Tabla 6.18.** Lista de materiales módulo *display* de 7 segmentos.

| ITEM | DETALLE | CANTIDAD | UNIDAD | VALOR UNITARIO | VALOR DEL RUBRO |
|---|---|---|---|---|---|
| **1** | CD4094 | 3 | c/u | $ 0,45 | $ 1,35 |
| **2** | display 7 segmentos | 3 | c/u | $ 0,60 | $ 1,80 |
| **3** | header hembra | 1 | c/u | $ 0,75 | $ 0,75 |
| **4** | header macho | 1 | c/u | $ 0,60 | $ 0,60 |
| **5** | placa fibra FR4 | 0,5 | 50mmx100mm | $ 5,00 | $ 2,50 |
| **6** | resistencias 1/4 w | 24 | c/u | $ 0,03 | $ 0,72 |
| **7** | zócalo 16 pines | 3 | c/u | $ 0,15 | $ 0,45 |
| | | | | **TOTAL** | $ 8,17 |

Autor: El investigador.

- **Módulo LCD**

La **tabla 6.19** muestra el costo total de los materiales del módulo LCD:

**Tabla 6.19.** Lista de materiales módulo LCD.

| ITEM | DETALLE | CANTIDAD | UNIDAD | VALOR UNITARIO | VALOR DEL RUBRO |
|---|---|---|---|---|---|
| **1** | header hembra | 1 | c/u | $ 0,75 | $ 0,75 |
| **2** | header macho | 1 | c/u | $ 0,60 | $ 0,60 |
| **3** | lcd | 1 | c/u | $ 8,98 | $ 8,98 |
| **4** | placa fibra FR4 | 0,25 | 50mmx100mm | $ 5,00 | $ 1,25 |
| **5** | Potenciómetro | 2 | c/u | $ 0,30 | $ 0,60 |
| **6** | resistencias 1/4 w | 15 | c/u | $ 0,03 | $ 0,45 |
| | | | | **TOTAL** | $ 12,63 |

Autor: El investigador.

- **Módulo Relevadores**

La **tabla 6.20** muestra el costo total de los materiales del módulo Relevadores:

**Tabla 6.20.** Lista de materiales módulo Relevadores.

| ITEM | DETALLE | CANTIDAD | UNIDAD | VALOR UNITARIO | VALOR DEL RUBRO |
|---|---|---|---|---|---|
| **1** | bornera | 7 | c/u | $ 0,30 | $ 2,10 |
| | diodo 1n4004 | 4 | c/u | $ 0,10 | $ 0,40 |
| **2** | header hembra | 1 | c/u | $ 0,75 | $ 0,75 |
| **3** | header macho | 1 | c/u | $ 0,60 | $ 0,60 |
| **4** | placa fibra FR4 | 0,25 | 50mmx100mm | $ 5,00 | $ 1,25 |
| **5** | rele | 2 | c/u | $ 0,60 | $ 1,20 |
| | resistencias 1/4 w | 4 | c/u | $ 0,03 | $ 0,12 |
| | transistores | 6 | c/u | $ 0,10 | $ 0,60 |
| **6** | ULN2003 | 2 | c/u | $ 0,45 | $ 0,90 |
| | | | | **TOTAL** | $ 7,92 |

Autor: El investigador.

- **Costo Total con Mano de Obra**

Los costos detallados anteriormente contemplan únicamente los materiales, divididos por módulos. La **tabla 6.21** es la suma total de los costos de producción de cada placa de entrenamiento, incluyendo la mano de obra e insumos varios.

**Tabla 6.21.** Costo total de la placa de entrenamiento.

| ITEM | DETALLE | CANTIDAD | UNIDAD | VALOR UNITARIO | VALOR DEL RUBRO |
|---|---|---|---|---|---|
| **1** | Adaptador | 1 | c/u | $ 2,60 | $ 2,60 |
| **2** | Display | 1 | c/u | $ 8,17 | $ 8,17 |
| **3** | estaño | 3 | metro | $ 0,35 | $ 1,05 |
| **4** | Impresión laser | 6 | c/u | $ 0,20 | $ 1,20 |
| **5** | LCD | 1 | c/u | $ 12,63 | $ 12,63 |
| **6** | *Mano de obra** | *10* | *hora* | *$ 8,00* | *$ 80,00* |
| **7** | Módulo principal | 1 | c/u | $ 32,29 | $ 32,29 |
| **8** | Papel cuché | 6 | c/u | $ 0,10 | $ 0,60 |
| **9** | Programador | 1 | c/u | $ 17,63 | $ 17,63 |
| **10** | Relevadores | 1 | c/u | $ 7,40 | $ 7,92 |
| | | | | **TOTAL** | $ 164,09 |

Autor: El investigador.

***Considerar el costo sin mano de obra en el inciso siguiente.*

- **Costo Total sin Mano de Obra**

Si cada estudiante construye su propio módulo, se resta el valor de la mano de obra, con lo cual el precio será de **84,09 dólares**, al 13 de junio de 2013.

### 6.6.20 Implementación del Prototipo

El prototipo se ha diseñado en su totalidad utilizando software gratuito, combinando entre software libre en todos los casos posibles, y freeware, proporcionado por la empresa Microchip Inc. Al estar libres sus terminales para conexiones a conveniencia, el presente diseño no influye negativamente y no limita la creatividad del usuario.

Los diseños se han realizado utilizando el EDA KiCad Build (2012-apr-16-27) stable, corriendo bajo Debian 3.2.46-1 Wheezy, aunque puede instalarse bajo Windows y Mac.

Para programar los microcontroladores se usó el freeware provisto por Microchip pk2cmd, para uso por terminal de comandos en GNU/Linux y Mac; mientras el software gratuito utilizado para esta función en sistemas Windows es el PICkit 2 v2.61 Programmer, este último con interfaz gráfica. Para el uso con *bootloader -USB* se necesita un microcontrolador que tenga módulo USB. El *bootloader* utilizado es el distribuido gratuitamente por Microchip en su microchip-libraries-for-applications-v2013-06-15, y el cargador *Microchip USB HID Bootloader v2.6* para sistemas basados en Windows, y el programa de línea de comandos *hid_bootloader*, escrito por el usuario *arhi*, que puede ser descargado en el siguiente enlace: http://elco.crsndoo.com/wordpress/2011/03/microchip-hid-bootloader-from-linux/. Las pruebas de concepto se realizaron directamente sobre protoboard.

La placa es fácilmente construible de manera artesanal, objetivo principal con el cual se ha planificado. Sin embargo, también puede encargarse su construcción profesional, dado que reúne criterios estandarizados para producción, cumple normas y recomendaciones IPC-2221; y el presente proyecto contiene toda la documentación necesaria para tal fin.

Se han construido los seis módulos propuestos, utilizando placa de fibra FR4, de una onza de cobre por pie cuadrado ($1\ ^{oz}/_{pie^2}$). El consumo de corriente máximo se ha establecido en 0.5 mA en el peor de los casos.

Utilizando las **tablas 6.21 y 6.22**, tomadas del estándar IPC-2221, se ha determinado que 5 mil (milésimas de pulgada) son el ancho mínimo de una pista, para soportar 0.5 amperios.

Con estas consideraciones, se han utilizado pistas de cobre de tres medidas:

- 500 um equivalente a 20 mil en el sistema imperial, para conductores de alimentación.

- 400 um equivalente a 16 mil en el sistema imperial para las pistas de señal.

- 300 um equivalente a 12 mil en el sistema imperial, para pistas de señal en sectores críticos.

Todos superiores al mínimo recomendado. Además se ha maximizado la capa de cobre para áreas comunes.

Las **figuras** de la **6.61 a la 6.65** muestran los módulos finalizados.

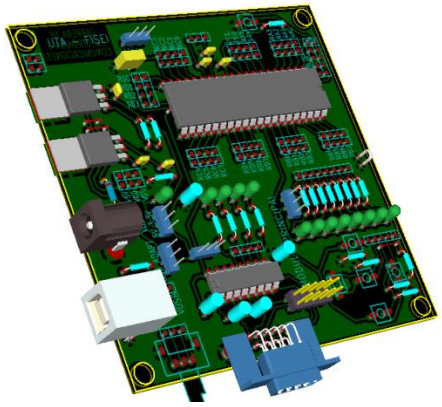


**Figura 6.61** Prototipo Módulo Principal finalizado.
Autor: El investigador.

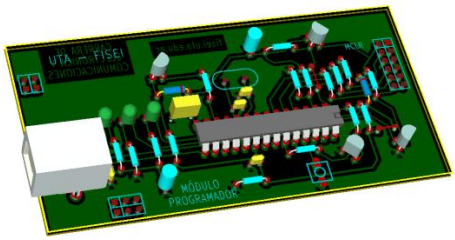



**Figura 6.62** Prototipo Módulo Programador finalizado.
Autor: El investigador.

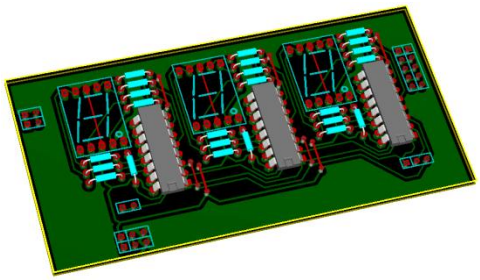


**Figura 6.63** Prototipo Módulo Pantallas de siete segmentos finalizado.

Autor: El investigador.

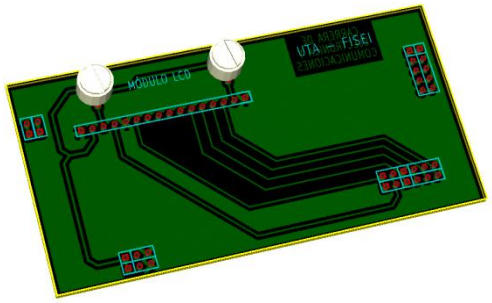



**Figura 6.64** Prototipo Módulo Pantalla LCD finalizado.

Autor: El investigador.

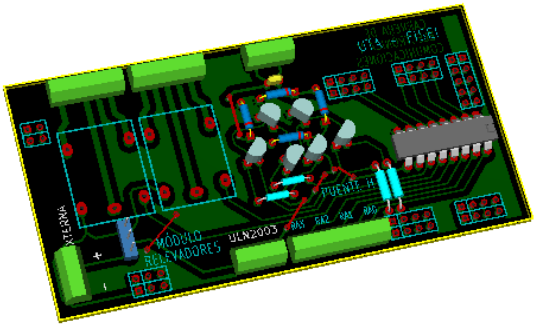



**Figura 6.65** Módulo Relevadores terminado.

Autor: El investigador.

## 6.7 Conclusiones

El diseño e implementación de soluciones con microcontroladores es un mundo complejo que avanza cada día. Para su aprendizaje y enseñanza didáctica hace falta escudriñar profundamente en las bases de su funcionamiento. Una plataforma educativa integral que contemple todos y cada uno de los dispositivos existentes en el mercado es extremadamente difícil de realizar, debido a su cantidad y a la velocidad con la cual se lanzan

nuevos diseños; sin embargo, como resultado de la presente investigación, se han determinado las siguientes conclusiones:

1. Se tienen programas apropiados de prácticas con microcontroladores, a través de la ejemplificación de implementaciones, el manejo de entradas y salidas (Puertos A, B, etc.) y del uso de sus registros de configuración (STATUS, INTCON, PIR, PIE, etc.); tanto para temas fundamentales en la iniciación a su uso, como para su manejo en transmisión de datos, aplicación de DSP, robótica y sistemas embebidos, lo cual permitió delimitar de manera eficaz el diseño del hardware del presente proyecto.

2. Se diseñó un diagrama electrónico de acuerdo los lineamientos previamente establecidos; consiguiendo de esta manera como producto final, un esquemático de implementación electrónica que es ordenado, entendible y modificable, que contiene los elementos básicos para que funcione cualquier microcontrolador Microchip PIC de 28 o 40 pines DIP, sin importar su capacidad o velocidad, que posee además periféricos de comunicaciones (USB, $I^2C$, SPI cuando se usen dispositivos que los soporten, además de RS-232 a través de un MAX232), y pulsadores e indicadores; esto con la finalidad de servir tanto como una herramienta lo más completa posible para la iniciación en las clases de microcontroladores, como también sirva de base a futuros desarrollos de proyectos investigativos en áreas como Automatización, Domótica, etc., en la Facultad y Universidad.

3. Se diseñó un proyecto electrónico completo, usando software libre (EDA KiCad *Build (2012-apr-16-27) stable*, corriendo bajo el sistema operativo Debian 3.2.46-1 *Wheezy*), con el objeto de hacerlo barato y legal, igualmente para alentar el uso de tecnologías disponibles gratuitamente, lo cual contribuye al desarrollo científico de la Facultad y sus integrantes. De esta manera se ha conseguido una plataforma tecnológica educativa fácilmente reproducible; económica, y que sirve para facilitar la socialización del conocimiento en microcontroladores y sus aplicaciones.

4. Se construyó prototipos de manera artesanal, con materiales de calidad (Fibra FR4 por ejemplo), disponibles y fácilmente reemplazables; lo cual amplía el impacto en el campo de investigación local, al tener en cuenta la realidad socioeconómica de los estudiantes y la investigación tecnológica del país. En correspondencia con esto, se han seguido normas de diseño (IPC-2221), que hacen viable también la construcción profesional en masa; para que el presente proyecto contribuya a mantener un equilibrio entre el número de estudiantes y el material didáctico disponible.

5. Una vez diseñada y construida la placa entrenadora para la enseñanza didáctica de microcontroladores en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial; de acuerdo con los resultados de la investigación, detallados previamente en los puntos 1 al 4; se

concluye que el proyecto en su totalidad es factible, aún cuando el diseño deba ser limitado a unos cuantos dispositivos (en este caso a cualquier microcontrolador Microchip PIC de 28 o 40 pines DIP, independiente de sus otras características), que representan los objetivos ideales para el aprendizaje efectivo de esta cátedra.

## 6.8 Recomendaciones

1. Se recomienda profundizar en el aprendizaje de los conceptos fundamentales de programación de microcontroladores, por ejemplo en el uso de los registros de configuración del microcontrolador, de la misma manera se debe ampliar el resto de programas de prácticas; sin importar el lenguaje escogido, ya que la aplicación de soluciones es el fundamento del aprendizaje.

2. Se recomienda profundizar en el conocimiento y uso de herramientas GNU/Linux para Electrónica, ya que existen estupendos recursos educativos de diseño, simulación, prueba y producción (como el mismo KICAD, KTechlab, PicSim), en su mayoría gratuitos y de libre uso.

3. Como recomendación fundamental, está el hacer uso adecuado de la placa entrenadora, teniendo en cuenta sus limitaciones tanto de corriente ($500mA$) como de voltaje ($hasta\ 12v\ con\ el\ adaptador$), observando de igual forma la correcta colocación de los elementos, especialmente el microcontrolador, ya que a pesar de las

protecciones implementadas, cualquier inobservancia en estos aspectos podría destruir todo el conjunto o por lo menos parte de él.

4. Es necesario que al construir artesanalmente las placas, se hagan uso de ciertas medidas básicas de protección (usar guantes y mascarilla), y se ponga mucha atención al uso de sustancias químicas. Al enviar a construir las placas, se debe tener en cuenta la confiabilidad y profesionalismo del fabricante.

5. El diseño debe hacerse manteniendo tanto la sencillez en el esquema electrónico y el PCB, como observando la economía para su construcción, utilizando recursos y materiales accesibles, para que sea fácilmente reproducible y modificable; teniendo siempre en cuenta las vastas posibilidades que la evolución inherente a estas tecnologías, propone a cualquier nuevo desarrollo.

# BIBLIOGRAFÍA

## Libros


Barrett, S. F., & Pack, D. J. (2006). *Microcontrollers Fundamentals for Engineers and Scientists* (Primera ed.). Printed in the United States of America: Morgan & Claypool.

Boylestad, R. L., & Nashelsky, L. (1998). *Teoría de circuitos y dispositivos electrónicos* (Octava ed.). (C. Mendoza Barraza, & A. Suárez Fernández, Trads.) México: Prentice Hall.

Comenio, J. A. (1998). *Didáctica Magna* (Octava ed.). México: Porrúa.

Heat, S. (2003). *Embedded Systems Desing* (Segunda ed.). Oxford: Butterworth-Heinemann.

Herrera, L., Medina, A., & Naranjo, G. (2004). *Tutoría de la Investigación Científica* (Primera ed.). Ambato: Universidad Técnica.

Ibrahim, D. (2008). *Advanced Pic Microcontroller Projects in C* (Primera ed.). Burlington: Elsevier.

Lindgren, B. W. (1960). *Statistical Theory* (Primera ed.). New York: The Macmillan Co.

Prat Viñas, L. (1998). *Circuitos y dispositivos electrónicos* (Quinta ed.). Barcelona, España: Edicions UPC.

Ruiz Robredo, G. A. (2001). *Electrónica básica para ingenieros* (Primera ed.). Santander, España: Universidad de Cantabria.

Sadiku, M. N., & Alexander, C. K. (2004). *Fundamentos de circuitos eléctricos.* España: McGraw-Hill.

Sánchez Miño, A. (2008). *Módulo de estadística y probabilidades* (Primera ed.). Ambato, Ecuador: UTA.


## Informes


Kernighan, B. W., & Hill, M. (1979). *A Tutorial Introduction To The Languaje B.* Bell Laboratories. New Jersey: Bell Laboratories.

**Hojas de especificaciones y manuales**


Fairchild. (1987). *CD4094BC 8-Bit Shift Register/Latch with 3-STATE Outputs.* Fairchild.

Fairchild. (2001). *MC78XX/LM78XX/MC78XXA Datasheet.* Fairchild.

Hitachi. (s.f.). *HD44780U Datasheet.* Hitachi.

Microchip. (2008). *PICkit 2 USER'S GUIDE.* Microchip.

Microchip. (2008). *Pickit 3 Poster.* USA: Microchip.

Microchip. (2013). *PIC16F87XA Datasheet.* Microchip.

Microchip. (2009). *PIC18F2455/2550/4455/4550 Datasheet.* Microchip.

National Semiconductor. (1996). *LM117 LM317A LM317 3-Terminal Adjustable Regulator.* National Semiconductor.

STMicroelectronics. (2002). *ULN2003A Datasheet.* STMicroelectronics.

Texas Instruments. (2004). *Comparing Bus Solutions.* Texas Instruments.

Vishay. (2000). *Standard 7– Segment Display*. Vishay.


**Estándares**


IPC-D-275 Task Group. (1998). *Generic Standard on Printed Board Design.* Northbrook, Illinois: IPC.


**Leyes**


Congreso Nacional. (2006). *Ley de propiedad intelectual (Codificación No. 2006-013).* Quito.


**Tesis**


Ballesteros Jordán, F. M. (2007). *Adquisición de datos de un sistema maestro – esclavo utilizando microcontroladores mediante comunicación serial para "M&M Automatización".* Ambato: Universidad Técnica de Ambato.

Guilcaso Molina, V. A. (2011). *Diseño y construcción de un brazo robótico industrial comandado mediante un sistema de control inalámbrico.* Ambato: Universidad Técnica de Ambato.

Moposita Valencia, G. F. (2010). *"Diseño de un circuito electrónico para el monitoreo de switches no administrables utilizando la tecnología del microcontrolador PIC16f877a".* Ambato: Universidad Técnica de Ambato.

**Links**

nobelprize.org. (5 de Mayo de 2003). *The History of the Integrated Circuit.* Recuperado el 18 de diciembre de 2012, de http://www.nobelprize.org/educational/physics/integrated_circuit/history/index.html

Universidad Técnica de Ambato. (2012). *Reglamento de graduación*. Recuperado el 20 de Mayo de 2012, de http://www.uta.edu.ec/v2.0/index.php?option=com_content&view=article&id=103&Itemid=67

# ANEXOS

| | | |
|---|---|---|
| | **UNIVERSIDAD TÉCNICA DE AMBATO**<br>**FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL** | |

**Encuesta dirigida a los señores estudiantes de séptimo, octavo y noveno semestre de la Carrera de Electrónica en la FISEI**

**OBJETIVO:** Detectar la incidencia del material didáctico utilizado en el módulo de microcontroladores, en la enseñanza didáctica del mismo.

**INSTRUCTIVO:**

- Marque con una X en el paréntesis de la alternativa que usted considere más adecuada.
- Procure ser lo más objetivo y veraz.
- Seleccione solo una de las alternativas por pregunta.
- No existen respuestas malas o equivocadas.

**Preguntas:**

1. ¿Es capaz usted de definir lo que es un circuito electrónico?
   ( ) Si. ( ) No. ( ) Tal vez.
2. ¿Puede definir de manera precisa lo que es una placa de circuito impreso?
   ( ) Si. ( ) No. ( ) Tal vez.
3. ¿Sabe lo que es y para qué sirve una micro-computadora?
   ( ) Si. ( ) No. ( ) Tal vez.
4. ¿Sabe usted lo que es un microcontrolador? Tenga en cuenta que todo PIC es un microcontrolador, pero no todo microcontrolador es un PIC.
   ( ) Si. ( ) No.
5. ¿Cuántas aplicaciones con microcontroladores ha implementado usted? Sea como parte de una práctica o fuera de la Universidad.
   ( ) Ninguna ( ) De 0 a 2 ( ) De 3 a 5 ( ) Más de 5
6. ¿Tiene claramente definido qué es un componente electrónico?
   ( ) Si. ( ) No. ( ) Tal vez.
7. ¿Para qué sirven los componentes electrónicos?
   a) ( ) Para construir prototipos.
   b) ( ) Para construir aplicaciones

c) ( ) Ambas.

8. ¿Puede usted definir claramente qué es una aplicación electrónica?

   ( ) Si. ( ) No. ( ) Tal vez.

9. ¿Puede usted definir claramente qué es un sistema embebido?

   ( ) Si. ( ) No. ( ) Tal vez.

10. ¿Cuántas soluciones con sistemas embebidos ha implementado usted? Sea como parte de una práctica o fuera de la Universidad.

    ( ) Ninguna ( ) De 0 a 2 ( ) De 3 a 5 ( ) Más de 5

Gracias por su colaboración.

| | | |
|---|---|---|
| | **UNIVERSIDAD TÉCNICA DE AMBATO**<br>**FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL** | |

**Pauta para registro de entrevista parcialmente estructurada**.

**Número: _____**

**Entrevistado: _____**

**Entrevistador: _____**

**Lugar y fecha: _____**

**Objeto de estudio: _____**

| **Pregunta** | **Interpretación valoración** |
|---|---|
| 1. ¿Cómo se imparte teoría de microcontroladores | |
| 2. ¿Qué ejemplos se usan? | |
| 3. ¿Cómo se planifican los laboratorios? | |
| 4. ¿Qué contenidos se desarrollan? | |

| | | |
|---|---|---|
| | **UNIVERSIDAD TÉCNICA DE AMBATO**<br>**FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL** | |

**Ficha de observación**

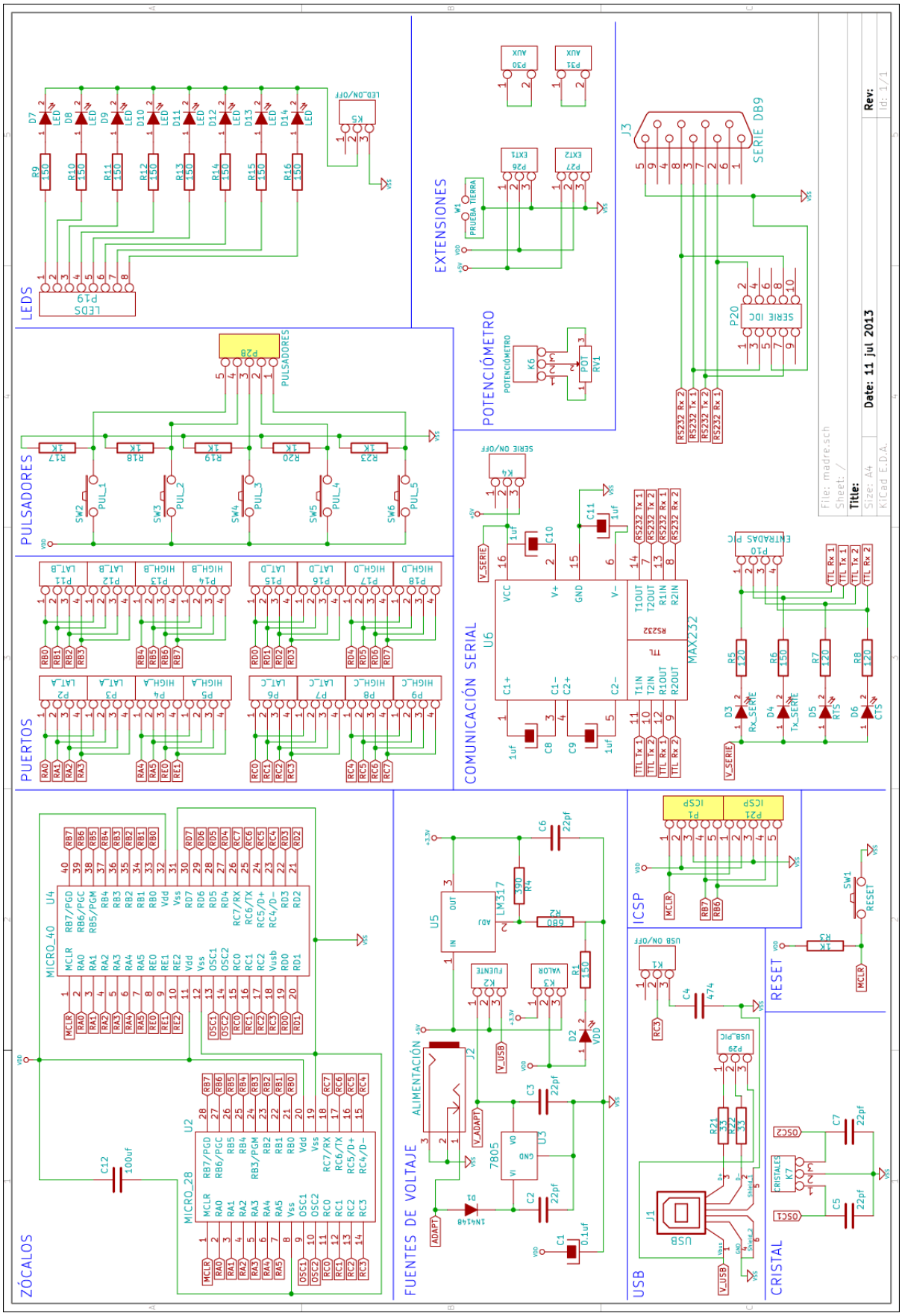
**Lugar: _____**

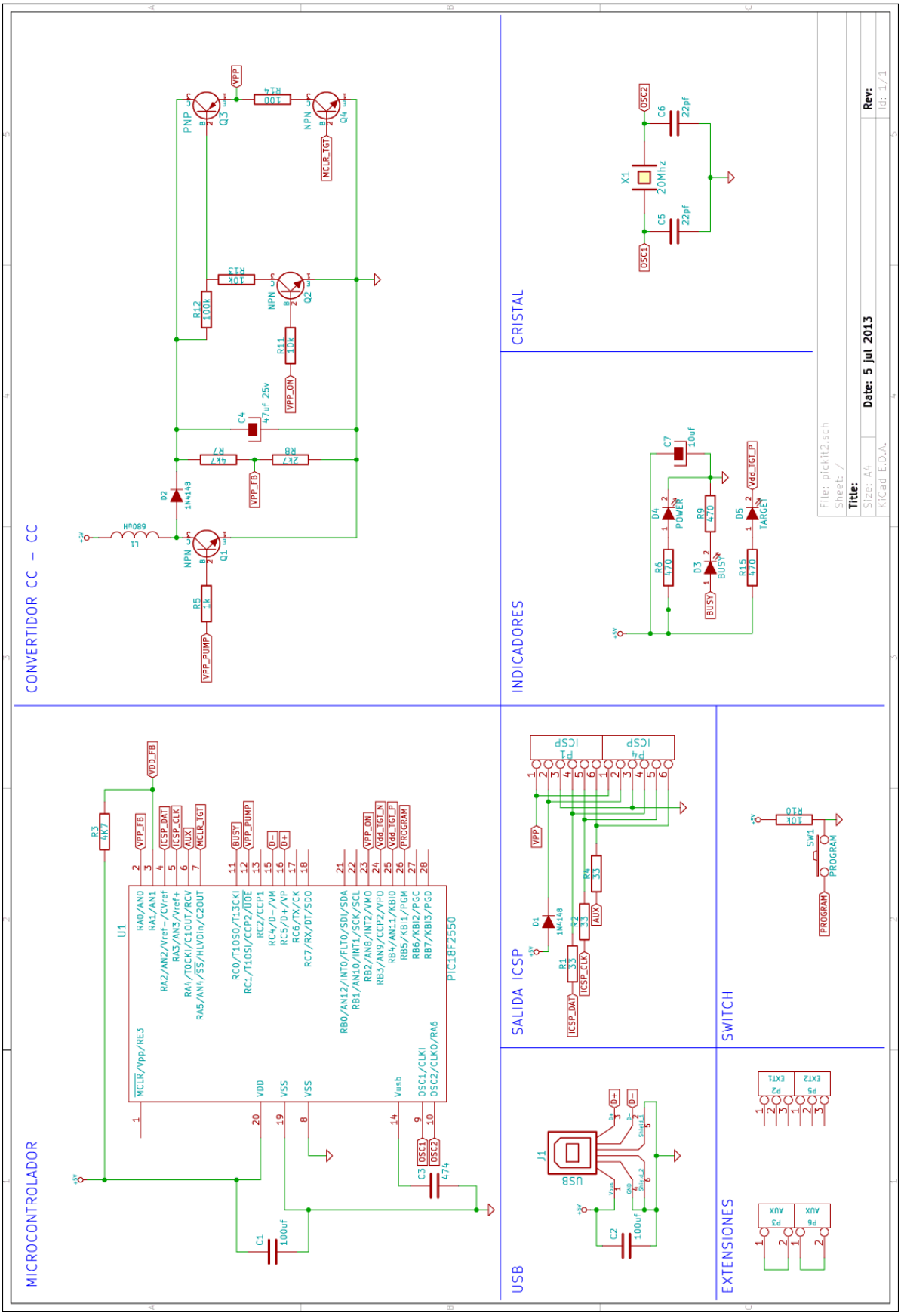
**Fecha: _____**

**Investigador: _____**

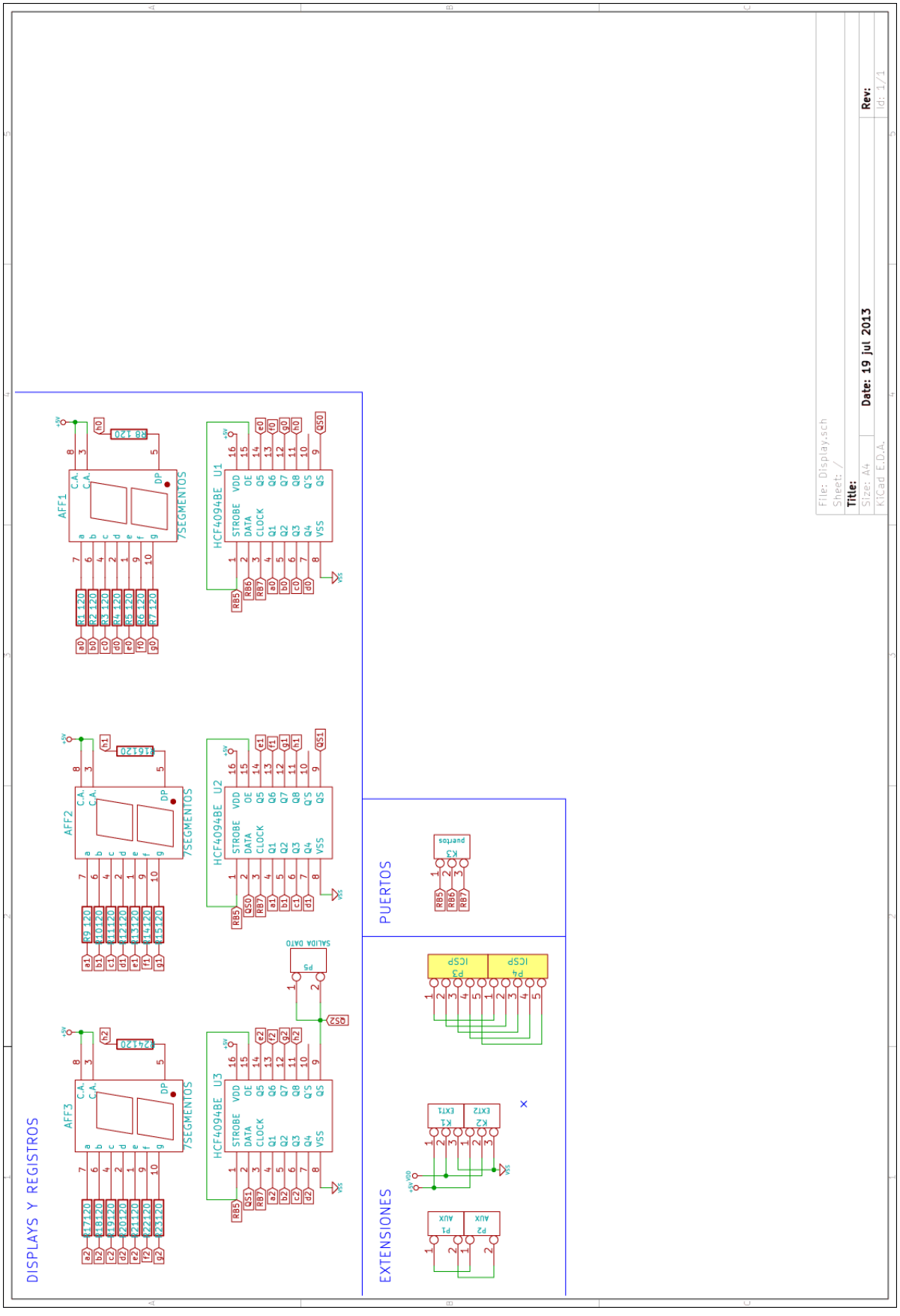
| |
|---|
| **Objeto de evaluación: Existencia de placas entrenadoras de microcontroladores** |
| **Interpretación – valoración:** |

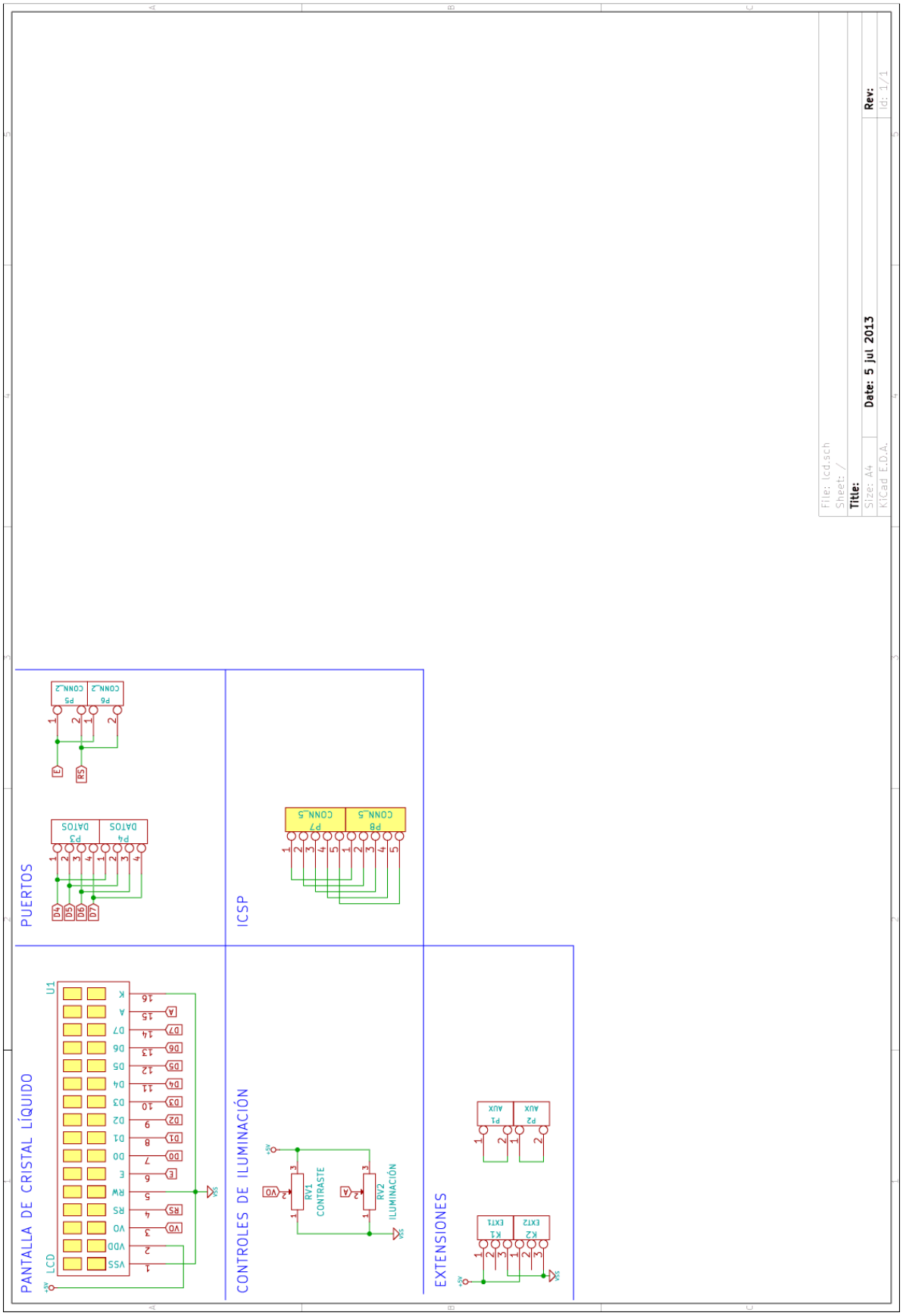
# ANEXO D1: Esquemático Módulo Principal

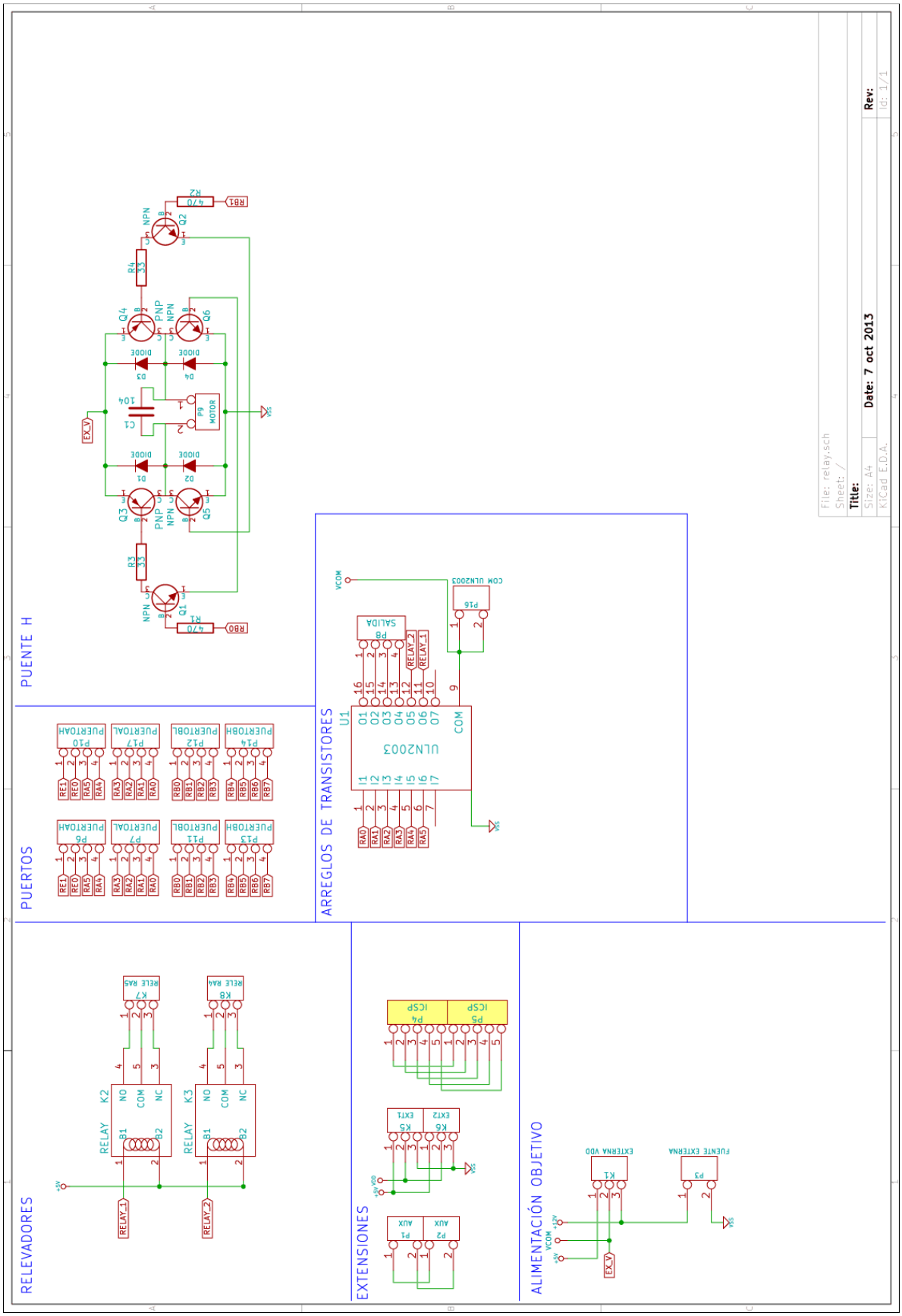



CONVERTIDOR CC – CC
CRISTAL
INDICADORES
MICROCONTROLADOR
SALIDA ICSP
SWITCH
USB
EXTENSIONES
PIC18F2550
Date: 5 jul 2013

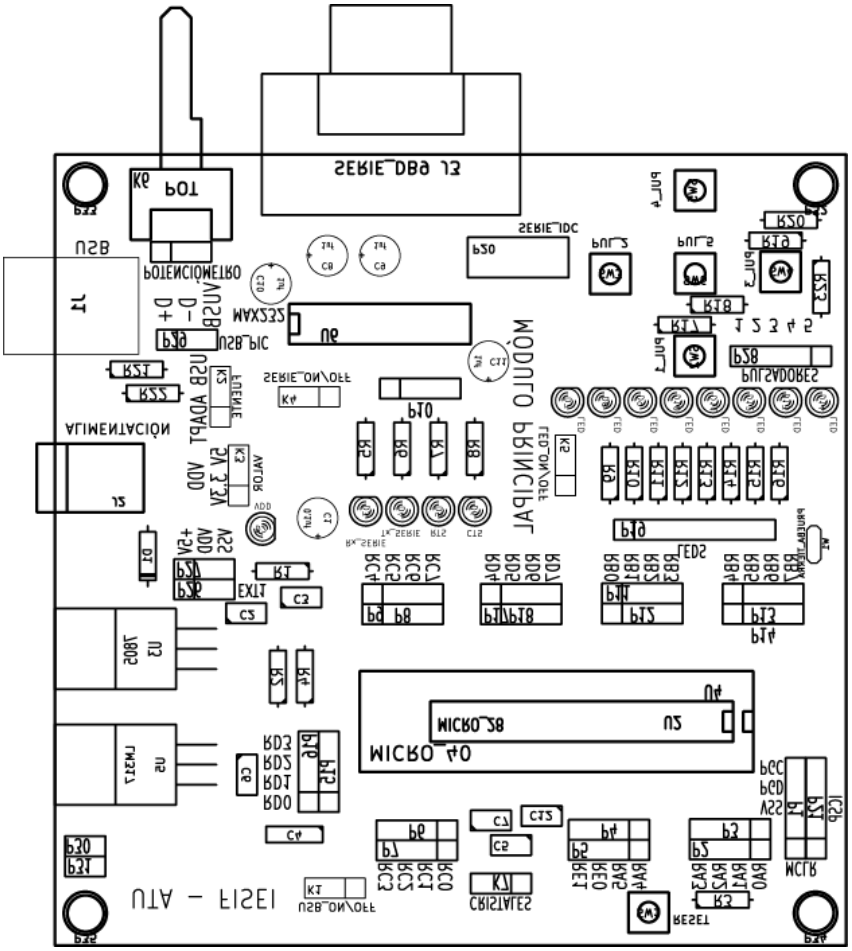
ANEXO D3: Esquemático Módulo Pantallas de 7 segmentos

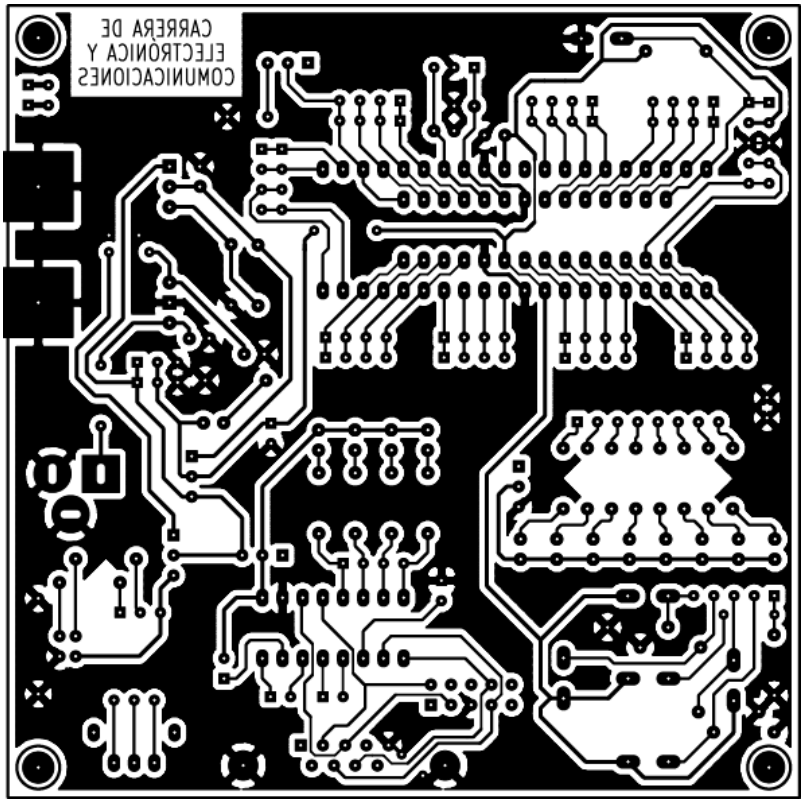



PUERTOS
ICSP
PANTALLA DE CRISTAL LÍQUIDO
CONTROLES DE ILUMINACIÓN
EXTENSIONES
LCD
U1
RV1
CONTRASTE
RV2
ILUMINACIÓN
DATOS
CONN_2
CONN_5
AUX
EXT1
EXT2
File: lcd.sch
Sheet: /
Title:
Size: A4
Date: 5 jul 2013
Rev:
KiCad E.D.A.
Id: 1/1

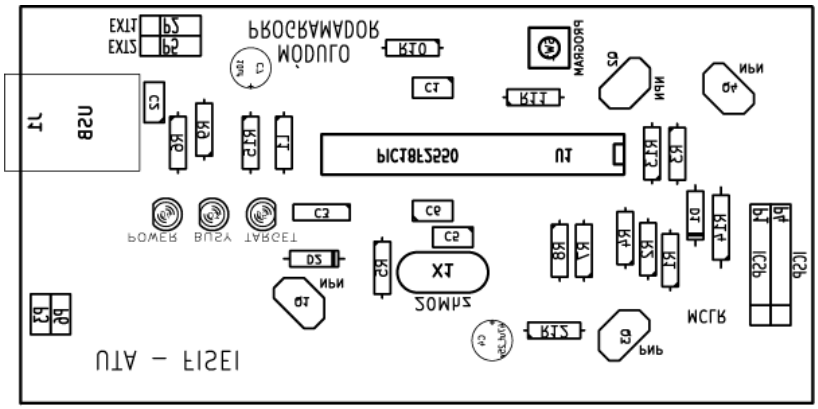

PUENTE H
PUERTOS
ARREGLOS DE TRANSISTORES
RELEVADORES
EXTENSIONES
ALIMENTACIÓN OBJETIVO
ULN2003
COM ULN2003
SALIDA
RELAY
MOTOR
ICSP
AUX
FUENTE EXTERNA
EXTERNA VDD
Title:
Date: 7 oct 2013
Rev:
Size: A4
KiCad E.D.A.
File: relay.sch
Sheet: /
Id: 1/1

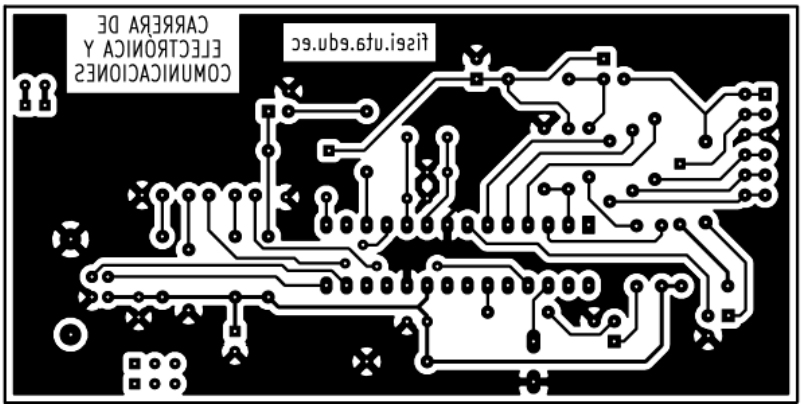
ANEXO E1: Pictograma de elementos y zona de cobre Módulo Principal



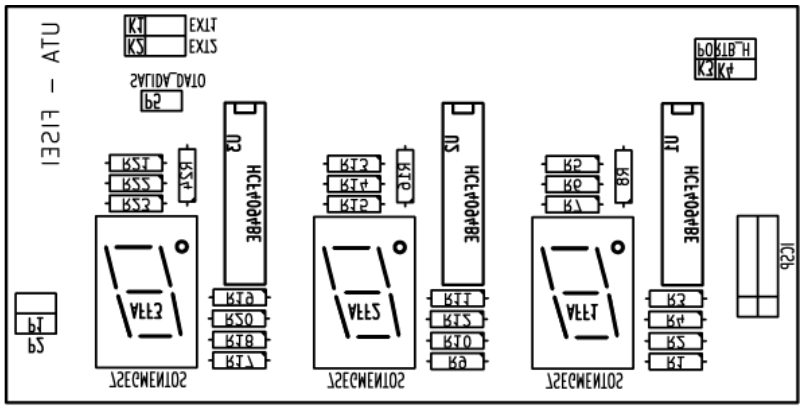



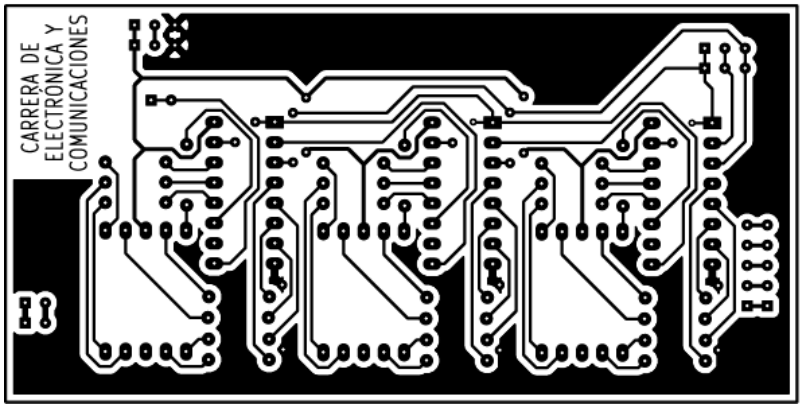
ANEXO E2: Pictograma de elementos y zona de cobre Módulo Programador



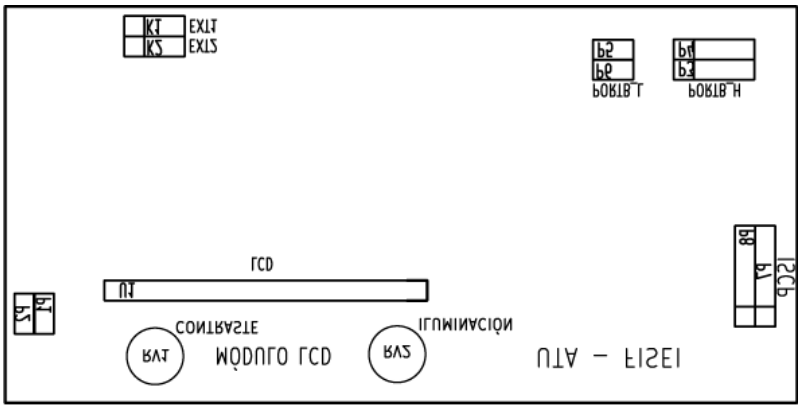


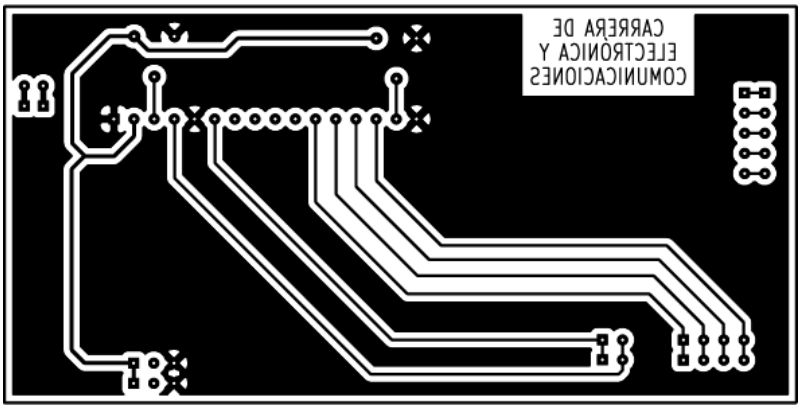
ANEXO E3: Pictograma de elementos y zona de cobre Módulo Pantallas de 7 Segmentos



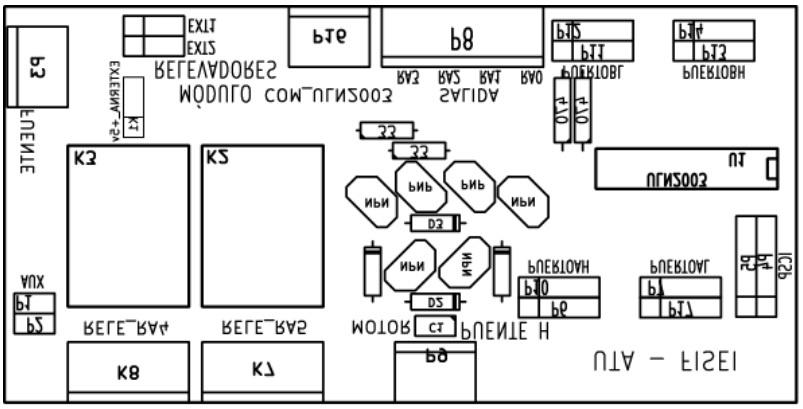


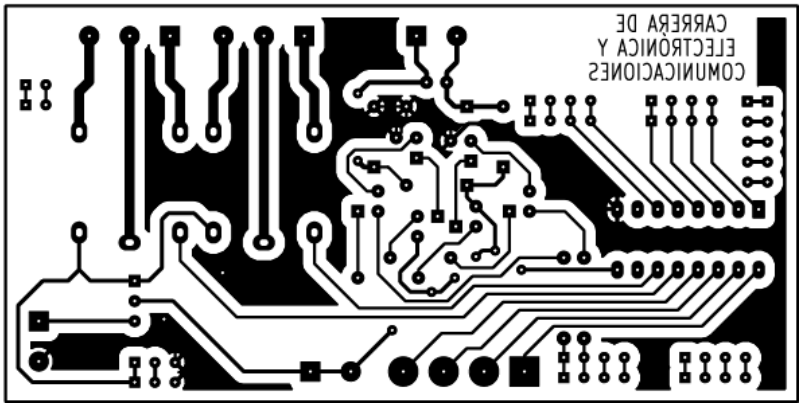
ANEXO E4: Pictograma de elementos y zona de cobre Módulo LCD



ANEXO E5: Pictograma de elementos y zona de cobre Módulo Relevadores



# PRÁCTICAS DE ELECTRÓNICA

**MANUAL DE REFERENCIA PARA**
**PLACA ENTRENADORA UTA - FISEI**

# MANUAL DE REFERENCIA

**GABRIEL E. SANTAMARÍA G.**

**UTA 2013**

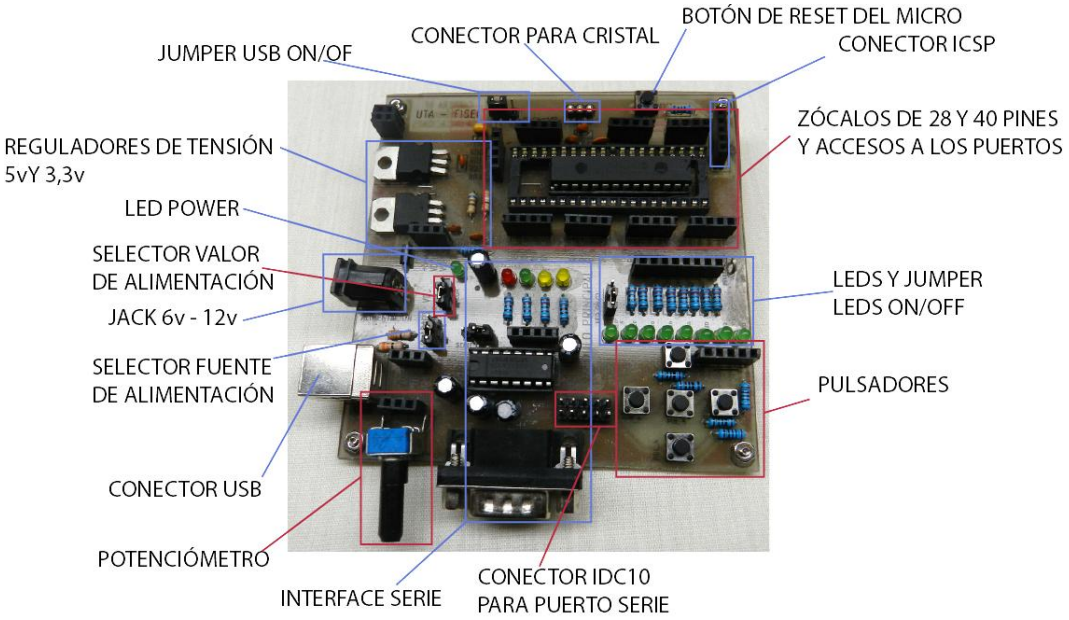
**MÓDULO PRINCIPAL**

El módulo principal está diseñado para llevar a la práctica todos los contenidos de los diversos créditos referentes a electrónica de la Carrera de Electrónica y Comunicaciones de La FISEI.

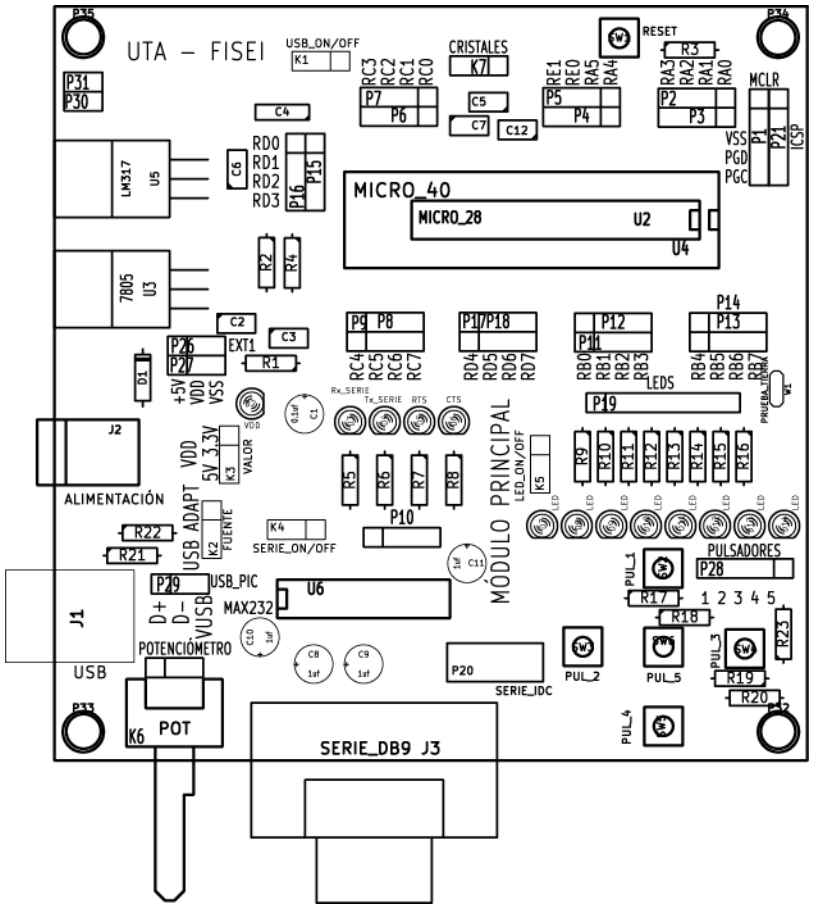
Se han incluido materiales comunes para realizar gran cantidad de prácticas, con poco cableado adicional. Los periféricos del microcontrolador son perfectamente accesibles en grupos de cuatro pines, y permiten el cableado directo con el dispositivo que sea conveniente, incluso fuera de los módulos aquí propuestos.

La alimentación se obtiene de un adaptador externo de 6 a 12 voltios o tomando la energía directamente desde el puerto USB, lo cual le brinda una gran portabilidad.

La figura a continuación, detalla el módulo principal:



El pictograma mostrado, detalla la nomenclatura de los diversos componentes del módulo principal:




Las zonas están perfectamente diferenciadas:

**RA0 a RD7** Accesos a los puertos del microcontrolador.

**ICSP** Es el conector para programación del micro.

**SERIE DB9** Es un conector DB9 macho, conectado al MAX232 con su respectiva electrónica.

**POT** Potenciómetro de 5KΩ.

**J1** Conector USB.

**J2** Jack de alimentación.

**J3** Conector DB9 macho.

**K2** Jumper para seleccionar fuente de alimentación (USB o Adaptador).

**K3** Jumper para seleccionar valor alimentación (3,3v o 5v).

**C1** Capacitor de filtrado de alimentación.

**C8, C9, C10, C11** Capacitores del MAX232.

**P10** Conector para señales seriales del microcontrolador al MAX232.

**P19** Conector de leds para cualquier puerto.

**P28** Conector de pulsadores para cualquier puerto.

**P29** Conector para señales de USB al microcontrolador.

**P26 y P27** Fuentes de alimentación y GND.

**R1 a R23** Resistores.

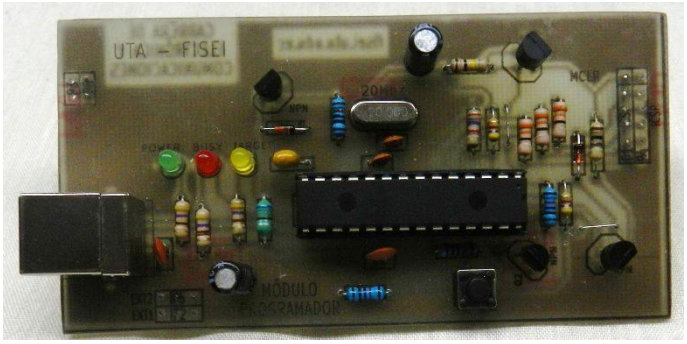
**SW1 a SW5** Pulsadores.

**K7** Conector de cristales (Puede dejarse sin cristal para usar oscilador interno).

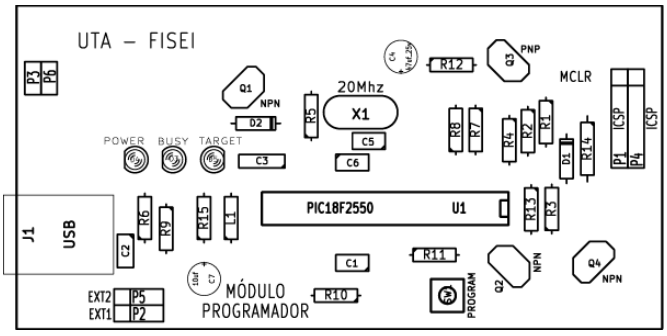
**ICSP** Conector estándar ICSP Microchip.

**MÓDULO PROGRAMADOR**

Es un Pickit2 genérico, con el firmware provisto por Microchip.




El diagrama pictórico se muestra a continuación:




En donde:

**J1** Conector USB.

**Q1, Q2, Q4** Transistores NPN.

**Q4** Transistor PNP.

**POWER** Led de alimentación. Se enciende cuando se ha conectado a un puerto USB.

**BUSY** El programador está realizando una tarea.

**TARGET** El programador está alimentando el objetivo.

**R1 a R15** Resistores.

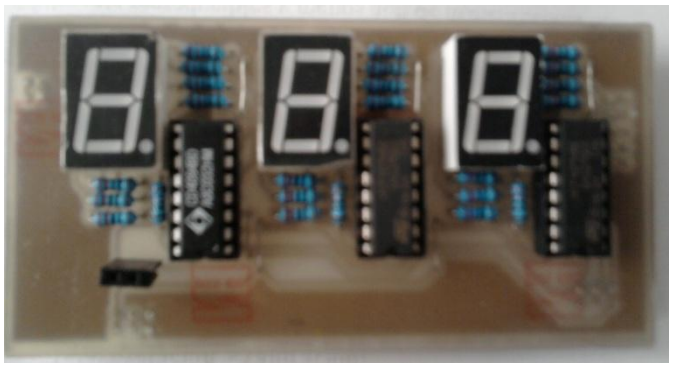
**PROGRAM** Manteniéndolo presionado mientras se conecta al USB hace entrar al PIC en modo *bootloader*.

**X1** Cristal de 20 Mhz.

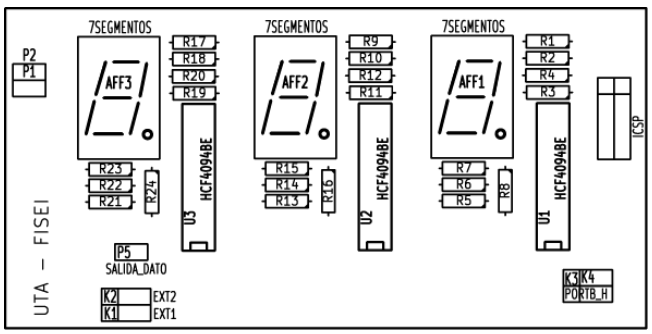
**ICSP** Conector estándar ICSP Microchip.

**MÓDULO PANTALLAS DE 7 SEGMENTOS**

Consiste en tres pantallas de 7 segmentos estándar, ánodo común. Cada una está conectada a un registro de desplazamiento CD4094 o equivalente. Posee la salida del último registro para conexión en cascada.



El diagrama pictórico muestra los componentes del módulo:




En donde:

**AF1 a AF3** Pantallas de 7 segmentos.

**K3, K4** Conectores hacia el módulo principal.

**U1 a U3** CD4094.

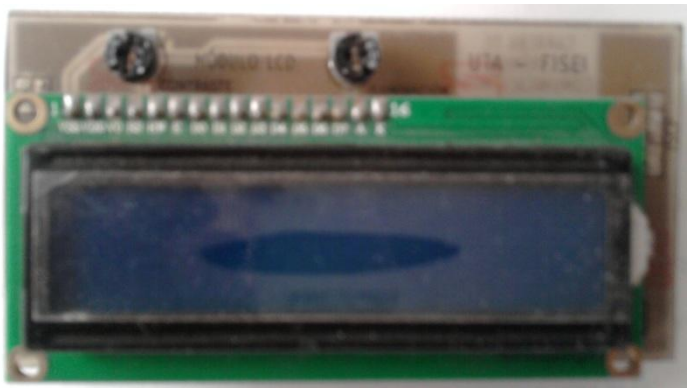
**R1 a R24** Resistores.

**P5** Salida del último CD4094 para conectar en cascada.

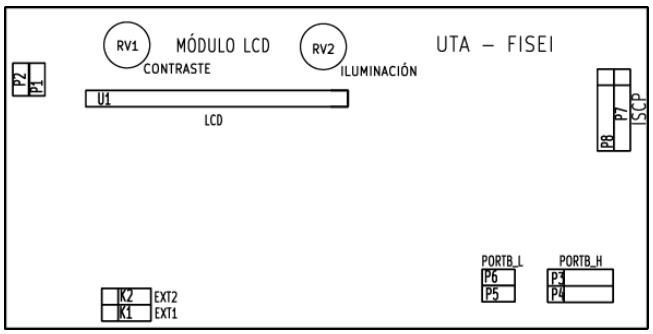
**ICSP** Conector estándar ICSP Microchip.

**MÓDULO LCD**

Posee dos potenciómetros, uno para control de contraste de los caracteres mostrados; mientras el segundo controla la intensidad de la luz de fondo del LCD. Consta además de un *header* hembra para conectar LCD con una salida de pines incompatible.



En el pictograma se puede apreciar la constitución del módulo:




En donde:

**U1** *Header* hembra para conexión de LCD.

**RV1** Control contraste carácter LCD.

**RV2** Control iluminación posterior LCD.

**P3 a P6** Conectores hacia el módulo principal.

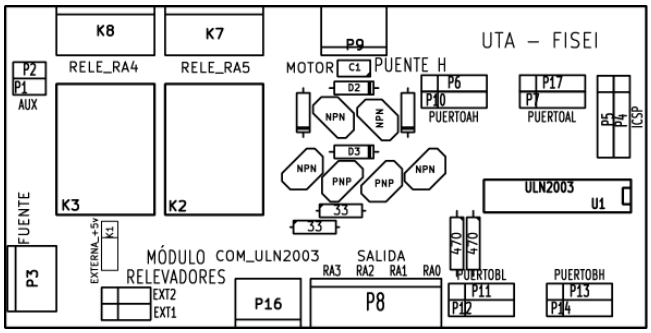
**ICSP** Conector estándar ICSP Microchip.

**MÓDULO RELEVADORES**

Consiste en un driver ULN2003 con siete salidas para control de potencia. Dos de ellas están conectadas a sendos relevadores de 5v; mientras las cuatro restantes tienen salida a borneras para conexión a dispositivos externos. La alimentación de los dispositivos controlados por el ULN2003 puede ser tomada de la placa, tomando en cuenta que el consumo no debe ser mayor a $400mA$.

Además posee un puente H controlado por PORTB.0 Y PORTB.1, con salida a bornera P9.

El diagrama pictórico es el siguiente:




En donde:

**U1** Driver ULN2003.

**K1** Selector alimentación ULN2003.

**K3, K2** Relevadores de 5v.

**K7, K8** Borneras de salida para relevadores.

**P3** Bornera para alimentación externa de ULN2003.

**P8** Bornera de salida pines libres ULN2003.

**P6, P7** Conectores hacia el módulo principal.

**P9** Bornera de salida de puente H

**P10, P17**Conectores hacia el módulo principal.

**P16** Borneras para COMÚN de ULN2003.

**ICSP** Conector estándar ICSP Microchip.

**BOOTLOADER USB**

El *bootloader* es un programa residente en el microcontrolador. Aunque pueden funcionar tanto por puerto serie como por USB, se ha escogido este último por cuanto está presente en la mayoría de dispositivos modernos. El *bootloader* utilizado es el distribuido gratuitamente por Microchip en su *microchip-libraries-for-applications-v2013-06-15*, y el cargador *Microchip USB HID Bootloader v2.6* para sistemas basados en Windows, y el programa de línea de comandos hid_bootloader, de *arhi*, que puede descargarse de http://elco.crsndoo.com/wordpress/2011/03/microchip-hid-bootloader-from-linux/.

Este *bootloader* en particular ha sido compilado para usarse en el PIC18F2550, habilitarse mediante un pulsador activo alto en PORTC.2 y tener dos leds indicadores en PORTC.0 y PORTC.1.

Para utilizarlo en la placa entrenadora se debe grabar el archivo *BOOTLOADER.hex* con el módulo programador por única vez. Además de hacer las conexiones correspondientes desde los *headers* libres; se debe asegurar que el *jumper USB* esté en posición *ON y* la fuente seleccionada sea USB. Las salidas PORTC.4 a D- y PORTC.5 a D+. Una vez cargado el programa, se pueden deshacer todas estas conexiones.

El programa residente ocupa las primeras posiciones de memoria, así que cuando se compilen los programas, se deberá comenzar en la línea de memoria 1000. Además, el programa deberá estar preparado para usar oscilador de 48Mhz. Estos son los requisitos para utilizar el *bootloader*.

# PRÁCTICAS DE ELECTRÓNICA

# MANUAL DE PRÁCTICAS PARA PLACA ENTRENADORA UTA - FISEI

# APLICACIONES CON MICROCONTROLADORES

**GABRIEL E. SANTAMARÍA G.**

**UTA 2013**

**INTRODUCCIÓN.**

Los microcontroladores están presentes en una gran cantidad de sistemas, desde los más simples, hasta las más complejas aplicaciones.

La presente guía de prácticas, pretende utilizar la Tarjeta Entrenadora de la UTA – FISEI, para que la primera aproximación del estudiante a los sistemas embebidos con microcontroladores sea una experiencia didáctica y entretenida. Las prácticas en su mayoría requieren cableado mínimo y permiten al usuario decidir el ajuste de las características tanto de hardware como de software según convenga.

Los ejemplos suponen un pequeño conocimiento previo de programación y abarcan tres lenguajes ampliamente utilizados: C, Basic y ensamblador. También se toman en cuenta compiladores gratuitos y comerciales, para demostrar la versatilidad de la placa entrenadora. Sin embargo, pueden traducirse fácilmente de unos a otros.

Los programas de igual manera, se encuentran comentados extensivamente, para que su funcionamiento sea fácilmente entendible y didácticamente aplicable. El estudiante será capaz de entender el código aquí expuesto, la manera en la que funciona en la entrenadora; y podrá aplicar el conocimiento en implementaciones propias, tanto en la tarjeta, como en diseño diferente de hardware.

Se ha dividido en cinco secciones, para discernir mejor el campo de aplicación del código específico; sin embargo, es importante anotar que se pueden utilizar las secciones de código a conveniencia, independientemente del uso que se le vaya a dar a la aplicación que se esté desarrollando.

Finalmente, el código aquí presentado, es solamente una pequeñísima muestra de lo que se puede llegar a lograr con la entrenadora, dejando abierto el camino a quien la utilice; para mejorarla y adaptarla a sus propios gustos y ocupaciones. Las fuentes de consulta principales son los documentos de la página de Microchip Inc. www.microchip.com, y los diferentes foros como www.todopic.com.ar, picmania.garcia-cuervo.net, pic-linux.foroactivos.net, www.circuitvalley.com y la gran variedad de información académica que hoy en día ofrece internet.

La sección de prácticas propuestas, es una pequeña muestra de las prácticas que pueden escribirse para esta placa entrenadora, sin embargo, de ninguna manera se restringe a ellas, siendo el límite la imaginación del usuario.

## CONTENIDO

**PRÁCTICAS DESARROLLADAS**

# PRÁCTICAS PROPUESTAS

## SECCIÓN 1: PRÁCTICAS BÁSICAS

### 1. BOOTLOADER

Algunos microcontroladores, como los de las series 18Fxx5x, tienen la capacidad de escribir en su propia memoria flash, derivada de la EEPROM; pero que puede escribir en varios sectores en una misma operación. Esta característica es utilizada para ponerlos en modo de recepción de datos a través de un cambio externo (como un pulsador). Finalmente, se transmite el nuevo programa a través de alguno de sus periféricos de comunicaciones, como son el módulo USART o USB si el microcontrolador lo posee.

### 2. DISPLAY DE 7 SEGMENTOS

Los displays de 7 segmentos son elementos ópticos ampliamente utilizados para mostrar datos. La técnica recomendada para su manejo con microcontroladores, es la multiplexación a través de registros de desplazamiento, como el CD4094; para hacer un uso eficiente de los pines disponibles.

### 3. CONTADORES DE DOS Y TRES DÍGITOS CON DISPLAY DE 7 SEGMENTOS

Conectando registros de desplazamiento entre sí, se logra multiplexar una gran cantidad de datos. Esta técnica es comúnmente llamada conexión en cascada, y los registros de desplazamiento que vienen en encapsulado, como el CD4094, están diseñados de tal manera que permiten transmitir los datos de unos a otros de manera sencilla.

### 4. MASTER CLEAR Y PERRO GUARDIÁN

Los PICs tienen diversos mecanismos de protección para no caer en lazos infinitos o quedarse en partes indeseadas del programa. De los más importantes son el *Master Clear*, reinicio por cambio de estado en hardware; y el *Watch Dog* o Perro Guardián, un contador interno que, de no realizarse instrucción alguna en un determinado tiempo, reinicia el microcontrolador.

### 5. HOLA MUNDO EN LCD

El LCD es un modo de mostrar información detallada de los procesos implementados. Los diferentes lenguajes que existen, tienen diversas formas de manejar estos dispositivos.

### 6. GRABAR CARACTERES EN LA MEMORIA DEL LCD

Los LCD tienen una memoria CG ROM incorporada que guarda permanentemente los juegos de caracteres a mostrarse. Cada vez que uno de ellos tiene que ser mostrado en pantalla, una memoria DD RAM lo consulta a la CG ROM. Los LCD más comunes en el mercado local poseen un driver Hitachi HD44780, que maneja una tanto una memoria CG ROM como una CG RAM,

esta última posee ocho espacios que pueden ser grabados con caracteres personalizados que pueden ser consultados por la DD RAM mediante sus locaciones en memoria.

**7. TECLADO MATRICIAL**

Es una manera de introducir datos a los procesos del microcontrolador. Un teclado matricial se llama así, debido a que usa una matriz de entradas para determinar el dato que se desea introducir, usando la menor cantidad de pines, para la mayor cantidad de datos posibles.

**8. GRAFICAR EN LCD**

Con los caracteres incluidos en la CG ROM y los personalizados en la CG RAM, es posible enviar puntos a conveniencia a la DD RAM, lo cual permite formar en el LCD gráficos o representaciones gráficas de procesos en el sistema implementado.

**9. CALCULADOR USANDO TECLADO Y LCD**

Implementar un calculador con teclado y LCD, resume una parte importante de los procesos básicos de un sistema embebido.

**10. REGISTROS DE CONFIGURACIÓN**

Los PICs tienen locaciones especiales de memoria llamados registros de configuración, como *INTCON*, que controla la configuración general de interrupciones. Este contenido puede ser llenado por el usuario, y los valores ahí asignados afectan el comportamiento del microcontrolador. Los nombres de los registros y el cómo afectan al PIC se encuentran ampliamente explicados en las hojas de especificaciones.

## SECCIÓN 2: TRANSMISIÓN DE DATOS

**1. ESTÁNDAR RS232**

Las comunicaciones seriales pueden llevarse a cabo de varias maneras. Sin embargo existen estándares cuyo uso se ha generalizado, como el USB (*Universal Serial Bus*) o el que es objeto de esta práctica, el RS232. Maneja niveles de voltaje de -15 a +15 v, algo que el PIC no puede soportar; por lo tanto necesita un *driver* o controlador que ajuste dichos niveles. La fase con el MAX232 es ideal para entender cómo funciona el estándar.

**2. COMUNICACIÓN RS232 CON HANDSHAKING CONTROLADO POR PIC**

La comunicación RS232 puede lograrse con tres líneas: GND, Tx y Rx. Una forma completa del estándar adiciona por lo menos otras dos líneas: CTS (*Clear to Send*) que dice que el dispositivo está libre para recibir datos y RTS (*Request To Send*) que es una petición para enviar datos. Esto sirve para evitar pérdidas de

información y cada línea puede implementarse con una salida del microcontrolador.

## 3. COMUNICACIÓN SERIE ASÍNCRONA: CREAR UN PROTOCOLO PROPIO

Una vez entendido el proceso que un dispositivo utiliza para transmitir un dato, el reto es crear un estándar propio, con niveles de voltaje TTL, y los tiempos y señales que defina el usuario. Esto sirve para fijar el conocimiento y despejar dudas.

## 4. COMUNICACIÓN I2C

Es un protocolo de comunicaciones muy usado en la industria, para comunicar microcontroladores entre sí y con sus periféricos. Utiliza tres líneas SDA (*Serial Data*) SCL (*Serial Clock*) y GND: tierra, para conectar una gran cantidad de dispositivos (Depende de las características de los elementos y la aplicación). Una gran cantidad de modelos de PICs traen un módulo que soporta esta clase de comunicación.

## 5. PROTOCOLO MODBUS

Un protocolo industrial que trabaja en modo maestro - esclavo, puede tener hasta 247 dispositivos conectados. Aun cuando no especifica el tipo de red, el protocolo MODBUS dicta la manera en que el maestro y el esclavo intercambian datos, el formato que deben tener estos datos, y el tratamiento de errores. Todo este procedimiento puede ser implementado en microcontroladores.

## 6. COMUNICACIÓN SERIE SÍNCRONA: CREAR UN PROTOCOLO PROPIO

Al crear un protocolo propio, se puede aplicar el conocimiento, e implementar las señales que se crea conveniente con los periféricos del PIC.

## 7. DISPOSITIVOS USB EN MODO BULK TRANSFER

Se puede implementar un dispositivo de transferencia bidireccional masiva de información *bulk transfer*. Esto es posible en los microcontroladores que soporten USB 2.0 por hardware, como lo es la serie 18Fxx5x.

## 8. DISPOSITIVOS USB EN MODO HID

Los dispositivos de interfaz humana, como ratones y teclados USB, pueden ser implemen*t*ados en algunos microcontroladores, aprovechando el módulo USB que tienen embebido.

### 9. DISPOSITIVOS USB EN MODO CDC

Dispositivos de comunicaciones, sirven para emular puertos seriales virtuales (COM).

### 10. PROCESAMIENTO ANIDADO CON PIC

A través de los diversos protocolos de comunicación, pueden dividirse las tareas de una implementación entre varios PICs, sincronizando y optimizando el poder de procesamiento de la aplicación.

## SECCIÓN 3: APLICACIONES DE TEORÍA DE DSP

### 1. MUESTREO DE SEÑALES CONTINUAS

Los microcontroladores son dispositivos eminentemente digitales. Sin embargo, son capaces de trabajar con señales analógicas a través de convertidores que muestrean la señal analógica y guardan en registros específicos, valores discretos que representan dicha señal.

### 2. CUANTIZACIÓN Y RECONSTRUCCIÓN DE SEÑALES ANALÓGICAS

Cuando se obtienen las muestras de una señal analógica, se definen también niveles de operación. Estos son intervalos de tiempo en los cuales la amplitud es constante, para que el número de amplitudes infinito de una señal analógica pueda ser representado en el número finito de bits que tienen los registros del convertidor del microcontrolador. A la definición de estos niveles se le llama cuantización.

### 3. CONVERSOR DE SEÑALES ANALÓGICAS A DIGITALES

Es un módulo que integran algunos microcontroladores, que cumple con las funciones de muestrear y cuantizar una señal analógica.

### 4. TRANSFORMADA RÁPIDA DE FOURIER (FFT) A UNA SEÑAL OBTENIDA DEL CONVERSOR ANALÓGICO DIGITAL

La transformada de Rápida de Fourier se usa para análisis y procesamiento de señales, especialmente para trabajar con su respuesta en frecuencia. Se puede usar el algoritmo *Split-radix,* para calcular la FFT de una secuencia guardada en los registros del conversor analógico – digital del PIC, en lugar de una secuencia dada.

### 5. ANALIZADOR DE ESPECTRO DE AUDIO APLICANDO FFT

Una señal de audio convenientemente tratada, puede ser procesada por el conversor analógico – digital del microcontrolador, y los valores resultantes pueden ser visualizados ya sea en un LCD o en una aplicación gráfica específica.

### 6. FILTROS DIGITALES CON PIC

Aplicando combinaciones de operaciones matemáticas a los valores en los registros del conversor analógico – digital, se pueden logar filtros digitales, ya sean éstos pasa bajos, pasa altos o pasa banda.

### 7. GENERADOR DE FUNCIONES USANDO LOS MÓDULOS DEL PIC

El PIC posee módulos que generan diversos tipos de impulsos, tal que debidamente amplificados, por ejemplo con los arreglos Darlington de un ULN 2003 y controlados pueden constituirse en un generador de funciones.

### 8. MICRÓFONO ELECTRET CON PIC

Con una pequeña interfaz que acondicione las señales, eliminando el ruido eléctrico y ajustando la salida a niveles ttl; se puede conectar un pequeño micrófono al PIC, lo cual constituye una magnífica forma de ingresar señales analógicas.

### 9. RECONOCIMIENTO DE VOZ USANDO CONVERTIDOR A/D

Identificando las formas de onda de palabras simples como “uno”, “no”, etc., se pueden determinar equivalentes digitales identificables por el microcontrolador.

### 10. OSCILOSCOPIO USB DE BAJA RESOLUCIÓN CON PIC

Utilizando las entradas analógicas y los módulos USB de algunos microcontroladores, como los de la serie 18Fxx5x; es factible recibir datos en una aplicación de PC que representen las señales de entrada. La resolución estará limitada por el tamaño del registro y la velocidad de conversión del hardware analógico – digital.

## SECCIÓN 4: ROBÓTICA

### 1. SENSORES ANALÓGICOS

La premisa de la robótica es la autonomía del sistema. Para lograr ello se deben recoger datos del medio, para lo cual se han desarrollado los sensores. Ellos dotan de “*sentidos*” a las implementaciones robóticas. Existen varias clases de sensores, y aquellos que sean analógicos deberán utilizar los conversores analógico – digital que poseen la mayoría de los microcontroladores.

### 2. SENSORES DIGITALES

El tipo digital de sensor envía una señal digital proporcional a la magnitud censada. Esta señal puede ser trabajada directamente por cualquier microcontrolador.

## 3. CONTROL DE SENSOR ÓPTICO

Los sensores ópticos, por ejemplo el infrarrojo CNY70; detectan las variaciones en la longitud de onda que recogen. Esto se puede utilizar para detectar cambios en la superficie, lo cual es verdaderamente útil por ejemplo en la construcción de seguidores de línea.

## 4. CONTROL DE SENSOR DE PROXIMIDAD

Existen diversos sensores de proximidad y distancia, como el formado por una pareja de infrarrojo y fototransistor colocados de manera conveniente. Otros emiten ultrasonido y calculan la distancia con el tiempo en que tarda el rebote, otros hacen lo mismo con un láser. Algunos detectan el cambio en la inductancia o emiten un campo magnético. De cualquier forma entregan una señal ya sea analógica o digital, que puede ser procesada por un microcontrolador.

## 5. PRIORIDAD DE PROCESAMIENTO

Al ocuparse de varias tareas, el microcontrolador tiene maneras de dar prioridad a las más importantes o críticas, ya sea a través de sus registros de configuración o de interrupciones u otros métodos.

## 6. CONTROL DE UN MOTOR A PASOS

Un motor a pasos, como el 24BYJ48; tiene la particularidad de poder controlar de manera muy precisa la posición de su eje. Esto se logra al activar determinados terminales que excitan la bobina en el sector deseado. Un microcontrolador solo, no puede encargarse del trabajo, sin embargo a través de un driver como el ULN2003 es ampliamente utilizado para estos menesteres.

## 7. GRADOS DE LIBERTAD

El movimiento de un robot o sistema autónomo está limitado la cantidad y características de los motores que lo muevan. Los grados de libertad se cuentan por la cantidad de articulaciones que tiene dicho sistema. En cada articulación tiene uno o más motores de diversos tipos, que pueden ser manejados con un microcontrolador.

## *8. JOCKSTICK* CON PIC Y PULSADORES

El jockstick es una palanca de mando que permite controlar la dirección de movimiento. Un grupo de pulsadores convenientemente ubicados y controlados por el PIC, pueden convertirse en uno de estos dispositivos.

## 9. CONTROL CON COMANDOS DE VOZ

A través de una entrada de audio, una vez identificadas las señales, se pueden desatar procesos dentro del microcontrolador en función de ellas.

### 10. AUTÓMATA CON PIC

Contando con los sensores, es posible implementar un sistema capaz de reaccionar de determinada manera en un entorno cambiante. A esto se le llama autómata y es perfectamente posible de implementar con un microcontrolador.

## SECCIÓN 5: SISTEMAS EMBEBIDOS

### 1. CONTROL DE TEMPERATURA POR PUERTO USB

Se puede implementar un sistema que mida datos a través de un sensor, envíe esos datos a una aplicación en PC a través del puerto USB en los microcontroladores que lo posean, y actúe en consecuencia a las órdenes por él recibidas.

### 2. CONTROL DE RELE POR PUERTO USB

Se puede implementar un sistema que controle una interface de potencia a través del puerto USB en los microcontroladores que lo posean.

### 3. SENSORES PIR (SENSOR INFRAROJO PASIVO O *PASSIVE INFRARED SENSOR*)

Un PIR es un sensor de movimiento, que escanea el lugar de la implementación con infrarrojos. Es configurable y puede ajustarse su sensibilidad, ya sea por software o por hardware, dependiendo del modelo. Generalmente trabajan con una señal analógica. Un modelo barato y ampliamente utilizado es el sensor PIR de domo Dyp-Me003.

### 4. REGULACIÓN DE LUZ AMBIENTAL CON PIR

Se puede implementar un sistema tal que, si no detecta movimiento, apague las luces del sitio en donde se encuentre implementado.

### 5. EMULACIÓN DE DIMMER CON PIC

Un dimmer sirve para controlar el paso de voltaje a una aplicación. Se puede emular este comportamiento  través de los módulos PWM del PIC, para controlar el ángulo de disparo de dispositivos como el diac, scr o triac.

### 6. CONTROL DE LUZ AMBIENTAL AUTOMÁTICO CON SENSOR PIR

Implementando un sensor PIR y un dimmer, se puede regular la luz ambiental de tal modo que se ajuste automáticamente a los requerimientos de iluminación de cada momento.

## 7. PLC CON PIC

Con arreglos de relés o implementando interfaces de potencia con semiconductores, se puede armar un arreglo lógico programable (PLC), utilizando microcontroladores.

## 8. CONTROL DE PROCESOS DE POTENCIA CON PIC Y RELÉ

Los microcontroladores, con una interfaz de potencia apropiada, es capaz de manejar procesos de potencia, ya sea a través de elementos pasivos o activos como el relé.

## 9. APLICACIONES INDUSTRIALES DE COMUNICACIÓN $I^2C$

El objetivo primario de $I^2C$ es comunicar dispositivos industriales entre sí. Cada microcontrolador puede ser uno de estos dispositivos, simulando aplicaciones industriales de manera barata y realista.

## 10. APLICACIONES INDUSTRIALES DE PROTOCOLO MODBUS

Se puede aplicar el protocolo MODBUS con microcontroladores, a través de la implementación de las direcciones y demás consideraciones que intervienen en él. Incuso con el módulo principal se puede simular un terminal que sea maestro de una serie de dispositivos industriales.

# SECCIÓN 1: PRÁCTICAS BÁSICAS

## PRÁCTICA 1.1: LED PARPADEANTE EN ENSAMBLADOR

**Introducción.**

Cada familia de microcontroladores tiene un juego de instrucciones que son compiladas directamente por un software, como el *MPASM Assembler*; el cual las traduce a lenguaje hexadecimal. Estas instrucciones en hexadecimal se ubicarán en los registros. Un programa que contiene exclusivamente este tipo de instrucciones se dice que está escrito en ensamblador.

La forma más básica de controlar un proceso con los pines de salida de un microcontrolador es poniéndolo en un estado bajo o uno alto. Esto significa tener el valor de Vdd o Vss a la salida del pin. Una referencia visual simple de este comportamiento se consigue conectando un led al pin en cuestión.

**Objetivos.**

- Compilar un programa escrito en ensamblador, que encienda y apague un led controlado por un pin del microcontrolador.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- MPLAB IDE 8.87
- Compilador MPASM assembler v5.46

**Módulos**

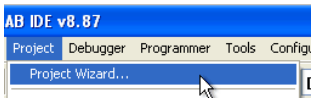
- Módulo principal.
- Módulo programador.

**Materiales.**

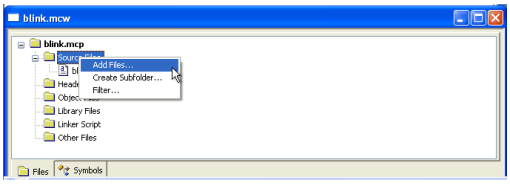
- PIC18F2550 (Reemplazable modificando el programa)

**Procedimiento.**

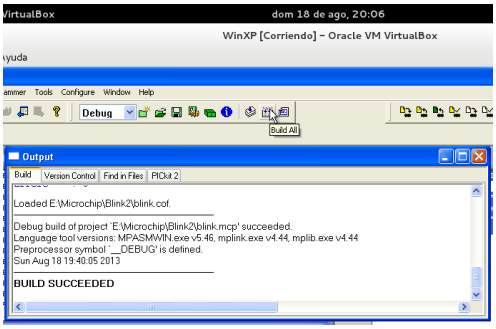
1. Abrir MPLAB IDE.
2. Crear en nuevo proyecto: *Project -> Project Wizard*.



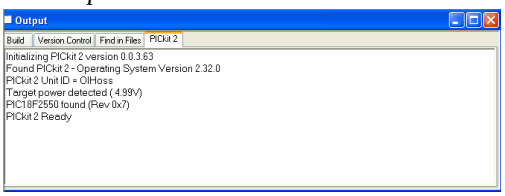

3. Elegir el PIC18F2550.
4. Elegir *MPASM Assembler* como herramienta de compilación.
5. Guardar el proyecto en una locación cercana a la raíz del sistema (para evitar inconvenientes).
6. Poner un nombre adecuado como *blink.*
7. No agregar ningún archivo existente y finalizar.
8. Crear un nuevo archivo.
9. Pegar el contenido del programa.
10. Grabar con el mismo nombre del proyecto, agregando la extensión *.asm;* en el ejemplo quedaría como *blink.asm*
11. Hacer clic derecho en *Source Files* del panel de navegación y agregar *blink.asm*



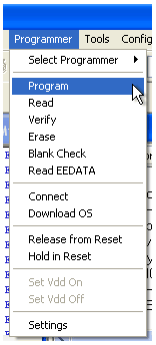

12. Presionar el botón *build all.* Se debe obtener como resultado *BUILD SUCCEEDED*, en el panel *OUTPUT.*




13. Elegir *Absolute* como opción de generación de código.
14. Cablear desde RA0 hasta cualquier led.
15. Poner el jumper LED en posición ON.
16. Conectar el módulo programador al principal.
17. Conectar un cable USB al módulo programador.
18. Seleccionar en el menú *Programmer -> Select programmer -> PICkit2*. Se debe obtener la siguiente salida en *Output:*




19. Pulsar *Programmer -> Program*




20. El led conectado al Puerto A0 debe comenzar a parpadear.

**Programa.**

```
;*******************************************************************************
*
; Bits de configuración
;*******************************************************************************
*
CONFIG FOSC = INTOSCIO_EC          ; Seleccionar el oscilador interno.
CONFIG MCLRE = OFF                 ; Desactivar Master Clear.
;*******************************************************************************
*
; Declaración de direcciones de constantes
;*******************************************************************************
*
STATUS      equ     0fd8h   ; Asignar la dirección 0fd8h del registro STATUS.
TRISA       equ     0f92h   ; Asignar la dirección 0f92h del registro TRISA.
PORTA       equ     0f80h   ; Asignar la dirección 0f80h del registro PORTA.
CON1        equ     0f08h   ; Asignar la dirección 0f08h para primer contador de retardo.
CON2        equ     0f09h   ; Asignar la dirección 0f08h para segundo contador de retardo.
;*******************************************************************************
*
```

```
; Configuración de puertos
;*******************************************************************************
*
movlw   00h             ; Poner 0 en registro w.
movwf   TRISA   ; Mover el valor de w al registro de TRISA para poner puerto A como salida.
;*******************************************************************************
*
; Programa principal
;*******************************************************************************
*
Inicio  movlw   01h     ; Poner un valor diferente de cero en el registro w.
movwf   PORTA           ; Mover el valor de w a PORTA. La salida representará el binario
correspondiente.
;*******************************************************************************
*
; Retardo por programación
;*******************************************************************************
*
LAZO1   decfsz  CON1,1          ; Restar 1 al valor por defecto en CON1 (255).
Goto    LAZO1                   ; Si CON1 es diferente de cero, continuar restando.
decfsz  CON2,1                  ; Restar 1 al valor por defecto en CON2 (255).
goto    LAZO1                   ; Volver al inicio del lazo.
;*******************************************************************************
*
; Apagar LED
;*******************************************************************************
*
movlw   00h             ; Poner el valor 00h en el registro w.
movwf   PORTA           ; Mover el valor en W a PORTA (dirección 0f80h).
;*******************************************************************************
*
; Segundo retardo por programación
;*******************************************************************************
*
LAZO2   decfsz  CON1,1          ; Restar 1 al valor por defecto de CON1 (255).
Goto    LAZO2                   ; Si CONT1 es diferente de cero, continuar.
decfsz  CON2,1                  ; Restar 1 al valor por defecto de CON2 (255).
goto    LAZO2                   ; Volver al inicio del lazo.
;*******************************************************************************
*
; Volver al inicio del programa
;*******************************************************************************
*
        goto    Inicio          ; Volver a etiqueta Inicio.
;*******************************************************************************
*
; Finalizar el programa
;*******************************************************************************
*
        end
```

**Conclusiones.**

_____
_____
_____
_____
_____
_____

**Recomendaciones.**

**Reto.**

- Modificar el programa y la implementación, de tal forma que el Pin RA3 sea el que controle el led.

**Bibliografía.**


- MPLAB® IDE User's Guide.
- MPASM™ Assembler, MPLINK™ Object Linker, MPLIB™ Object Librarian User's Guide.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.


**PRÁCTICA 1.2: LED PARPADEANTE EN C**

**Introducción.**

Los programas escritos en ensamblador tienen la innegable utilidad de aprovechar al máximo los recursos del microcontrolador. Sin embargo resultan difíciles de manejar cuando se trata de aplicaciones complejas. Debido al hecho de que los PICs actuales tienen una buena cantidad de recursos a un precio reducido, para la mayoría de programas actuales, existe una infinidad de lenguajes de alto nivel, como C, Basic o jalv2. Estos compiladores agrupan ciertos procesos en ensamblador, en una sola palabra reservada. Con esto, la curva de aprendizaje se reduce, así como el tiempo de implementación. Además, casi todos aceptan instrucciones en ensamblador, con lo cual se puede lograr que un programa responda de mejor manera ante ciertos eventos críticos.

**Objetivos.**

- Compilar un programa escrito en C, que encienda y apague un led controlado por un pin del microcontrolador.
- Grabar en el PIC los archivos hexadecimales generados.

**Recursos.**

- MPLAB X 1.60
- Compilador XC18 v1.12

**Módulos**

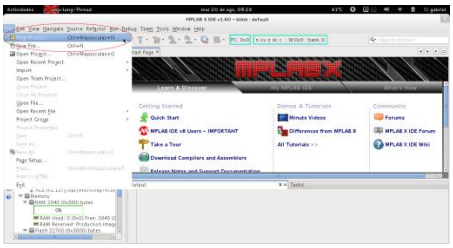
- Módulo principal.
- Módulo programador.

**Materiales.**

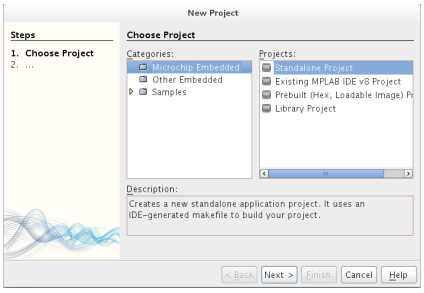
- PIC18F2550 (Reemplazable modificando el programa)

**Procedimiento.**

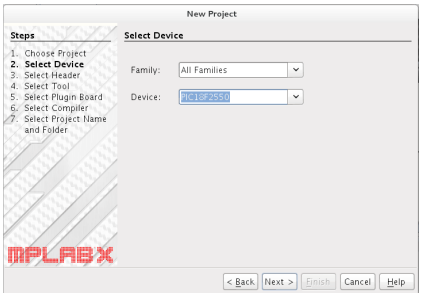
1. Abrir MPLAB X.
2. Crear en nuevo proyecto: *File -> New Project*.



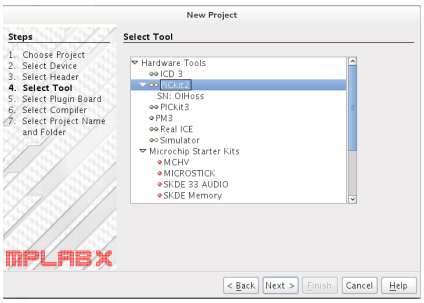
3. Escoger proyecto *Stand Alone.*



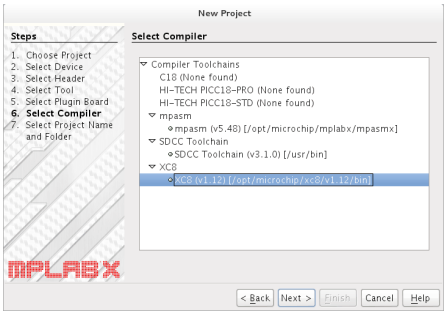

4. Elegir el PIC18F2550.



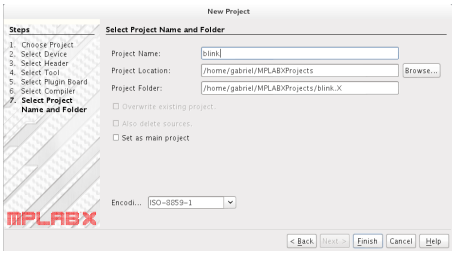

5. Escoger PICkit2 como herramienta de programación.



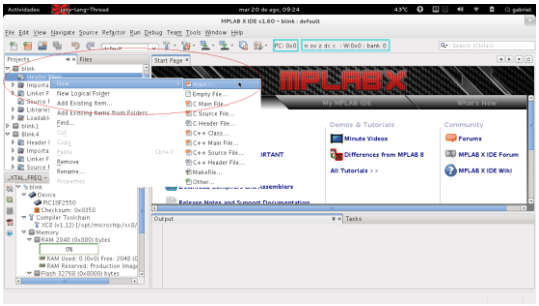

6. Elegir XC8 como herramienta de compilación.



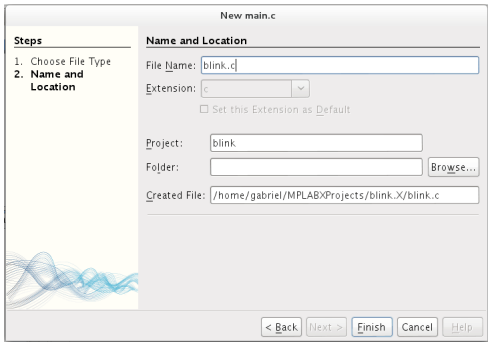

7. Guardar el proyecto en una locación cercana a la raíz del sistema (para evitar inconvenientes).



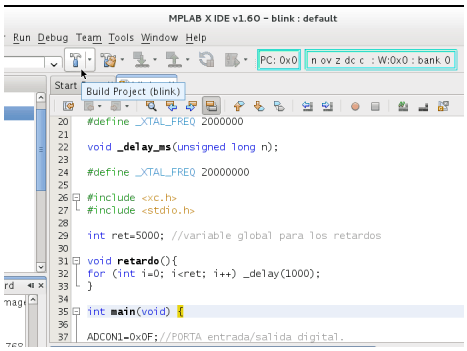

8. Hacer clic derecho en *Header Files -> New -> main.c*



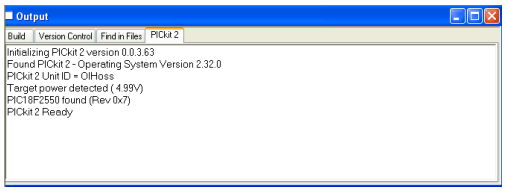

9. Nombrar de acuerdo al proyecto.



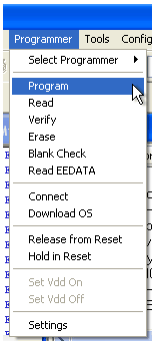

10. Pegar el contenido del programa.
11. Presionar el botón *Build Project.* Se debe obtener como resultado *BUILD SUCCEEDED*, en el panel *OUTPUT*.



12. Cablear desde RB0 hasta cualquier led.
13. Poner el jumper LED en posición ON.
14. Conectar el módulo programador al principal.
15. Conectar un cable USB al módulo programador.
16. Seleccionar en el menú *Programmer -> Select programmer -> PICkit2*. Se debe obtener la siguiente salida en *Output:*




17. Pulsar *Programmer -> Program*




18. El led conectado al Puerto B0 debe comenzar a parpadear.

**Programa en C.**

```
/*
 * File:   blink.c
 * Author: gabriel
 *
 * Created on 20 de agosto de 2013, 09:58 AM
 */
#include "xc.h"         // Cabecera genérica de dispositivos.
#include <stdio.h>            //Cabecera para funciones de entrada/salida.

#pragma config FOSC = HS    // Oscilador HS

#define _XTAL_FREQ 4800000  // Define frecuencia trabajo del PIC.

void retardo()      // Retardo por instrucciones
{
   int i, j;
   for(j=0x001; j<=0x050; j++)
   {
      for(i=0x001; i<=0x1388; i++);
   }
}

int main(void)                          // Rutina principal
{
   TRISBbits.TRISB0 = 0x00;    // Declarar PORTB.0 como salida

   while(1)                             // Bucle infinito
      {
      PORTBbits.RB0=1;        // Poner PORTB.0 a estado alto
      retardo();                    // Llamar a subrutina retardo()
      PORTBbits.RB0=0;        // Poner PORTB.0 a estado bajo
```

```
        retardo();                      // Llamar a subrutina retardo()
            }
    return 0;   // Regresa valor cero si se ha completado la función main.
}
```

**Conclusiones.**

**Recomendaciones.**

**Reto.**

- Modificar el programa y la implementación, de tal forma que el Pin RA2 sea el que controle el led.

**Bibliografía.**


- MPLAB® X IDE User's Guide.
- MPLAB® XC8 C Compiler User's Guide.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.


**PRÁCTICA 1.3: LED CON PULSADOR**

**Introducción.**

Los microcontroladores tienen pines que pueden trabajar tanto como de entrada como de salida de información digital. Esto se puede aprovechar para controlar manualmente un proceso. Para ello se utilizará un pulsador, que lleva a estado bajo una entrada que se mantiene en alto debido a la tensión que cae en una resistencia. Esto se utilizará para notificar al PIC que un cambio se ha producido a la entrada

**Objetivos.**

- Compilar un programa escrito en C, que encienda y apague un led al toque de un pulsador conectado al microcontrolador.
- Grabar en el PIC los archivos hexadecimales generados.

**Recursos.**

- MPLAB
- Compilador XC8 v1.12

**Módulos**

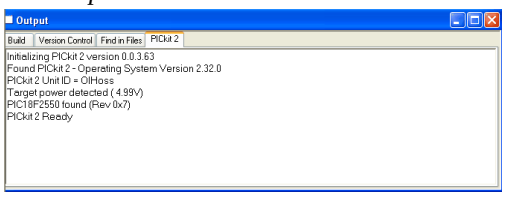
- Módulo principal.
- Módulo programador.

**Materiales.**

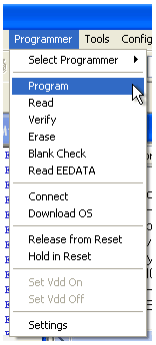
- PIC18F2550 (Reemplazable modificando el programa)

**Procedimiento.**

1. Abrir MPLAB X.
2. Seguir el procedimiento detallado previamente para crear un proyecto en el IDE de su elección.
3. Pegar el contenido del programa.
4. Presionar el botón *Build Project.* Se debe obtener como resultado *BUILD SUCCEEDED*, en el panel *OUTPUT*.
5. Cablear desde RA0 hasta cualquier led.
6. Cablear desde RB0 hasta cualquier pulsador.
7. Poner el jumper LED en posición ON.
8. Conectar el módulo programador al principal.
9. Conectar un cable USB al módulo programador.
10. Seleccionar en el menú *Programmer -> Select programmer -> PICkit2*. Se debe obtener la siguiente salida en *Output:*




11. Pulsar *Programmer -> Program*




12. El led conectado al Puerto A0 debe apagarse y encenderse mientras esté presionado el pulsador.

**Programa en C.**

```
/*
 * File:   blink.c
 * Author: gabriel
 *
 * Created on 20 de agosto de 2013, 09:58 AM
 */
#include "xc.h"         // Cabecera genérica de dispositivos
#include <stdio.h>          //Cabecera para funciones de entrada/salida.
```

```
#pragma config FOSC = HS    // Oscilador HS

#define _XTAL_FREQ 4800000  // Define frecuencia trabajo del PIC.

void retardo()      // Retardo por instrucciones
{
   int i, j;
   for(j=0x001; j<=0x01; j++)
   {
      for(i=0x001; i<=0x1388; i++);
   }
}

int main(void)         // Rutina principal
{
TRISA=0x00;                  //Todo PORTA como salida.
TRISB=0XFF;                  //Todo PORTB como entrada.
   while(1)                 // Bucle infinito
   {
      if(PORTBbits.RB0==0)    // Si PORTB.0 está en estado bajo
      {
      PORTAbits.RA0=0;       // Poner PORTA.0 a estado bajo
      //retardo();            // Llamar a subrutina retardo()
      }
      else                  // Si PORTB.0 está en estado alto
      {
      PORTAbits.RA0=1;       // Poner PORTA.0 a estado alto
      retardo();             // Llamar a subrutina retardo()
           }
   return 0;   // Regresa valor cero si se ha completado la función main.
   }
}
```

**Conclusiones.**

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

**Recomendaciones.**

_____
_____
_____
_____
_____

**Reto.**

- Modificar el programa y la implementación, de tal forma que el Pin RA3 sea el que controle el led.

**Bibliografía.**


- MPLAB® X IDE User's Guide.
- MPLAB® XC8 C Compiler User's Guide.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.


**PRÁCTICA 1.4: INTERRUPCIONES**

**Introducción.**

Los microcontroladores poseen métodos para desencadenar ciertos bloques de instrucciones, cuando ocurren diferentes eventos. Estos métodos son llamados interrupciones y pueden ser disparados por cambios en el hardware o en el software.

**Objetivos.**

- Compilar un programa escrito en Basic, que muestre algunas de las diversas interrupciones del PIC.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE 1.0.4.6

**Módulos**

- Módulo principal.
- Módulo programador.
- Módulo LCD.

**Materiales.**

- PIC18F2550 (Reemplazable modificando el programa)
- Lm35.

**Procedimiento.**

1. Abrir Proton IDE.
2. Pegar el contenido del programa.
3. Compilar el programa.
4. Conectar un LM35 al puerto RA0.
5. Cablear un pulsador a RB0.
6. Cablear un pulsador a RB1.
7. Cablear desde RC6 hasta Tx_Serie.
8. Cablear desde RC7 hasta Rx_Serie.
9. Poner el jumper Serie en posición "ON"
10. Grabar el programa al microcontrolador.
11. Conectar el módulo LCD.
12. Alimentar la placa principal.
13. Se visualizará la temparetura medida por el LM35 en el LCD.

14. Al presionar cualquiera de los botones, una terminal serie configurada con 8N1 a 9600 baudios, recibirá la temperatura y el tipo de interrupción.
15. Al enviar un carácter “A” desde una terminal serie configurada con 8N1 a 9600 baudios, se obtendrá como respuesta la temperatura y el tipo de interrupción.
16. El tipo de interrupción se verá también en la segunda línea del LCD.

**Programa.**

```
'****************************************************************
'*  Name    : UNTITLED.BAS                                      *
'*  Author  : Gabriel Santamaría                                *
'*  Notice  : Copyright (c) 2010 GabSan                         *
'*          : All Rights Reserved                               *
'*  Date    : 05/12/2010                                        *
'*  Version : 1.0                                               *
'*  Notes   :                                                   *
'*          :                                                   *
'****************************************************************
Declare PROTON_START_ADDRESS = $1000     "Línea para el Bootloader
Device = 18F2550
XTAL = 48                        ' Declara un cristal de 48Mhz, el bootloader lo deja configurado a 48Mhz.
Declare HSERIAL_BAUD = 9600
'****************************************************************
'llenado de registros de acuerdo a las características deseadas
'pueden llenarse en una sola línea, igualándolos a su equivalencia
'se han llenado individualmente para efectos de estudio
'****************************************************************

TRISC.0=1 'declaro C.1 como entrada

TRISC.7=1 'declaro c.7 como entrada (para USART)
TRISC.6=0 'declaro c.6 como salida  (para USART)

"Para dispoditivos compatibles:
"OPTION_REG.7=0 'RBPU, resistencias de pull up 0=habiliadas ,1=deshabilitadas
"OPTION_REG.6=0 'INTEDG flanco de interrupciones 0=bajada, 1=subida
"OPTION_REG.5=0 'T0CS seleccion del reloj de timer0, 0=interno,1=externo
"OPTION_REG.4=0 'T0SE selección del flanco 0=incremento en bajada ,1=incremento en subida
"OPTION_REG.3=0 'PSA asignacion del prescaler 0=timer0, 1=WDT
"OPTION_REG.2=1 'PS2:PS0 dimensionamiento del prescaler TMR0/WDT
"OPTION_REG.1=1 '
"OPTION_REG.0=1 '

ADCON1.7=1 'justificacion derecha
'adcon 6-4 no implementados
ADCON1.3=1 'bits de configuración
ADCON1.2=1 'para entradas A/D
ADCON1.1=1 'solo a.0 analógica
ADCON1.0=0 '

INTCON.7=1 'GIE interrupciones globales 0=deshablitadas, 1=habilitadas
INTCON.6=1 'PEIE interrupciones de periféricos 0=deshabilitadas, 1=habilitadas
INTCON.5=0 'T0IE interrupcion de timer0 0=deshabilitada, 1=habilitada
INTCON.4=1 'INTE interrupcion por b.0 0=deshabilitada ,1=habilitada
INTCON.3=0 'RBIE interrupción por cambio de estado en puerto b 0=dshabilitada ,1=habilitada
INTCON.2=0 'T0IF bandera de desbordamiento de timer0 (debemos encerar despues de cada interrupcion) 0=libre,
1=desbordada
INTCON.1=0 'INTF bandera de desbordamiento de rb.0 (debemos encerar despues de cada interrupcion) 0=libre,
1=desbordada
INTCON.0=0 'RBIF bandera de desbordamiento de rb cambiada (debemos encerar) 0=libre, 1=al menos una entrada de
rb7:rb4 ha cambiado de estado

INTCON2.7=1 'Resistencias de PullUp habilitadas
INTCON2.6=1 'Bit de seleccion de flanco de Interrupción externa 0. 1=Flanco alto; 0=Flanco bajo

PIE1.7=0 'PSPIE habilitación del puerto paralelo esclavo 0=deshabilitado, 1=habilitado
PIE1.6=0 'ADIE interrupcion del convertidor A/D 0=deshabilitado, 1=habilitado
PIE1.5=1 'RCIE interrupción recepción USART 0=deshabilitado, 1=habilitado
PIE1.4=0 'TXIE interrupción transmisión USART 0=deshabilitado, 1=habilitado
PIE1.3=0 'SSPIE interrupción puerto serial síncrono 0=deshabilitado, 1=habilitado
PIE1.2=0 'CCP1IE interrupción CCP1IE 0=deshabilitado, 1=habilitado
PIE1.1=0 'TMR2IE interrupción cuando coincide TMR2 a PR2 0=deshabilitado, 1=habilitado
```

```
PIE1.0=0 'TMR1IE interrupción cuando se desborda TMR1 0=deshabilitado, 1=habilitado

PIR1.7=0 'PSPIF bandera de la lectura/escritura paralela esclava (debe ser encerada) 0=ha ocurrido, 1=no ha ocurrido
PIR1.6=0 'ADIF bandera de la cnversión A/D 0=no ha terminado, 1=ha terminado
PIR1.5=0 'RCIF bandera de interrupción recepción USART 0=el búfer está vacío, 1=el búfer está lleno
PIR1.4=0 'TXIF bandera de interrupción transmisión USART 0=el búfer está vacío, 1=el búfer está lleno
PIR1.3=0 'SSPI bandera de interrupcion puerto serial síncrono
PIR1.2=0 'CCP1IF bandera interrupción (debe ser encerada)
PIR1.1=0 'TMR2IF bandera cuando coincide TMR2 a PR2(debe ser encerada) 0=libre, 1=ha coincidido
PIR1.0=0 'TMR1IF bandera cuando se ha desbordado TMR1 (debe ser encerada) 0=libre, 1=se ha desbordado

't1con 7-6 no implementados
T1CON.5=1 'T1CKPS1 configurar valores del prescaler
T1CON.4=1 'T1CKPS0 configurar valores del prescaler
T1CON.3=0 'T1OSCEN habilitacion del oscilador 1 = Oscillator habilitado, 0 = Oscillator apagado
T1CON.2=0 'T1SYNC sincronización del reloj externo 1 = sincroniza 0 = NO Sincroniza
T1CON.1=0 'TMR1CS selección de fuente de reloj 1 = reloj externo 0 = reloj interno(FOSC/4)
T1CON.0=1 'TMR1ON habilita timer2 1=habilita Timer1 0=detiene Timer1

TRISA.0 = 1
TRISB = 1


Print At 1,1,"Iniciando"
DelayMS 500
Cls     '' Limpiar LCD
HRSOut "Iniciando",13
On Interrupt GoTo int

'*****************************************************************
'declaración de variables y constantes
'*****************************************************************
Symbol tiempo=50    '' Variable  para retardo
Dim Temp As Float   '' Operar con decimales
Dim i As Byte
Dim j As Byte
Dim Dato As Byte
Dim Tmr1 As Byte

'*****************************************************************
'programa principal
'*****************************************************************

Temp = 0        '' Encerar variables
Dato = 0
Tmr1 = 0
i = 0
j = 0

main:
   GoSub Sensor       '' Ir a subrutina Sensor
   GoSub Retardo      '' Ir a subrutina Retardo
   GoSub Imprimir     '' Ir a subrutina Imprimir

   For j=0 To 10      '' Retardo para medición
      GoSub Retardo
   Next

   Tmr1=PORTB.1
   If Tmr1=1 Then 'si presiono el pulsador en b.1
      PIE1.0=1   'habilito la interrupción de timer1
   Else          'si no está presionado
      PIE1.0=0   'deshabilito la interrupción de timer1
   EndIf
GoTo main

'*****************************************************************
'subrutinas
'*****************************************************************


Sensor:
```

```
   Temp = ADIn 0      ;Asigno a una variable el valor de los registros ADRESH(bits+significativos) y ADRESL(bits-
significativos) de la conversión
   Temp = Temp - 1
   Temp = Temp + 1
   Temp = Temp/128
Return

Imprimir:
   Print At 1,1,DEC2 Temp," C" 'muestro temperatura en el lcd
   ''Print At 2,1,"H ",@ADRESH   'muestro registro A/D alto en el lcd
   ''Print At 2,10,"L ",@ADRESL  'muestro registro A/D bajo en el lcd
   GoSub Retardo              'retardo para visualizar el dato
Return

Retardo:    'retardo por instrucciones para q no haga esperar a la interrupción
   For i=0 To tiempo
      DelayMS 1
   Next
Return


'*****************************************************************
'rutina de interrupción
'*****************************************************************

Disable 'deshabilita la verificación de interrupciones entre instrucciones
int:
   'discrimino interrupciones por banderas:

   If Tmr1=1 Then
    If PIR1.0=1 Then 'si se dsborda timer1
      HRSOut "int. por timer1"
      HRSOut 13,DEC2 Temp,13
      Print At 2,1,"Int. por timer1"
      PORTB = 0
      DelayMS 10 'Retardo para lectura en display, no se detectan interrupciones.
    EndIf
   EndIf

   If INTCON.1=1 Then 'si la interrupción es por b.0
      HRSOut "int. por RB.0"
      HRSOut 13,DEC2 Temp,13
      Print At 2,1,"Int. por RB.0"
      DelayMS 10 'Retardo para lectura en display, no se detectan interrupciones.
   EndIf

   If PIR1.5=1 Then 'si la interrupción es serial ejecuto
   HRSin Dato 'ingreso dato (¿¿traba a interrupción por b.0??)
     If Dato="A" Then          'si dato es "A"
      HRSOut "int. Serial"
      HRSOut 13,DEC2 Temp,13 'envia bajar fila, dato Temp con dos decimales y bajar columna (enter: 13 ASCII)
      Print At 2,1,"Int. Serial"
      DelayMS 10 'Retardo para lectura en display, no se detectan interrupciones.
     EndIf
   EndIf

  If INTCON.2=1 Then   '' Si ha habido alguna interrupción
      Dato=dato+1     '' Aumento variable Dato
      If Dato=25 Then
         HRSOut 13,DEC2 Temp,13
         Print At 2,15,"  "
         Dato=0
      EndIf
   EndIf


   INTCON.4=1 '' Habilita la interrupción por b.0
   PIE1.5=1   '' Habilita interrupción por USART
   PIE1.0=0   '' Deshabilita interrupción Timer1
   INTCON.2=0 'encera la bandera de interrupción timer0
   INTCON.1=0 'encera la bandera de interrupción b.0
   PIR1.5=0   'encera la bandera de interrupción recepción USART
   PIR1.0=0   'encera la bandera de timer1
```

```
    ''Dato=0     'encero la variable dato para nuevo conteo


    Print At 1,16,"I" '' Muestra que estoy en la interrupción
    DelayMS 50 'visualizar indicador I en el lcd
    Print At 1,16," " '' Borra el indicador de interrupción
    Print At 2,1,"                " '' Limpia segunda línea
Resume 'vuelve a la parte del programa en donde detectó la interrupción
Enable 'habilita la verificación de interrupciones entre instrucciones
```

**Conclusiones.**

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

**Recomendaciones.**

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

**Reto.**

- Modificar el programa, de tal forma que el tipo de interrupción se muestre en el LCD.

**Bibliografía.**


- Proton IDE 1.0.4.6 Help Topics.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.

# SECCIÓN 2: TRANSMISIÓN DE DATOS

## PRÁCTICA 2.1: COMUNICACIÓN SERIE

**Introducción.**

La comunicación de datos se puede realizar de diversas maneras. Una común es enviarlos a través de uno o varios canales. Cuando se envía la información de manera ordenada un bit tras otro por un solo canal, se dice que es comunicación en serie. Existen diferentes tipos de comunicación serie, uno de ellos es el estándar RS232.

Los PIC poseen un módulo USART (Universal Serial Asynchronous Receiver Transmiter) capaz de transmitir datos en serie. Estos datos se generarán en niveles de voltaje TTL (Transistor – Transistor Logic), por lo cual, se utilizará el MAX232 incluido en la placa para generar las señales invertidad en el voltaje requerido por el estándar.

**Objetivos.**

- Compilar un programa escrito en Basic, que transmita información serie en niveles RS232.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE 1.0.4.6

**Módulos**

- Módulo principal.
- Módulo programador.

**Materiales.**

- PIC18F2550 (Reemplazable modificando el programa)

**Procedimiento.**

1. Abrir Proton IDE.
2. Pegar el contenido del programa.
3. Compilar el programa.
4. Cablear desde RC6 hasta Tx_Serie.
5. Poner el jumper SERIE en posición ON.
6. Grabar el programa al microcontrolador.
7. Alimentar la placa principal.
8. En un terminal serie con configuración 8N1 a 9600; se podrá visualizar el texto “UTA - FISEI”
9. Se visualizará la misma leyenda en el LCD si el módulo está conectado.
10. Se puede utilizar el conector IDC10 para visualizar la trama en un osciloscopio.

**Programa.**

```
Declare PROTON_START_ADDRESS = $1000     "Línea para el Bootloader
Device = 18F2550    " Declarar dispositivo
Xtal = 48          " Cristal de acuerdo al bootloader
Declare Hserial_Baud = 9600 " Declarar velocidad de transmisión

Inicio:

   High PORTB.0             " Indica que ha entrado a lazo principal
   HSerOut ["Uta - FISEI",13]  " Envía datos
   Print At 1,1, "Uta - FISEI" " Imprime datos en LCD 16x2
   DelayMS 500              " Retardo
   PORTB.0 = 0              " Apaga led al final del proceso
   DelayMS 500              " Retardo
GoTo Inicio                 " Volver a etiqueta Inicio
```

**Conclusiones.**

**Recomendaciones.**

**Reto.**

- Modificar el programa para que la transmisión sea a 2400 baudios.

**Bibliografía.**


- Proton IDE 1.0.4.6 Help Topics.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.


**PRÁCTICA 2.2: USB HID**

**Introducción.**

La comunicación USB (Universal Serial Bus) es un estándar para la comunicación de datos a altas velocidades. Requiere de software tanto en el dispositivo de almacenamiento, como en la máquina que hace de anfitrión. El software de la máquina anfitrión es llamado *driver* o controlador. La mayoría d sistemas operativos poseen un controlador genérico para el modo HID (Human Interface Device); que es un dispositivo externo que se añade al hardware de la máquina.

Los PIC de la serie 18Fxx5x poseen un módulo de comunicación USB por hardware. La estructura interna se detalla en la hoja de especificaciones.

**Objetivos.**

- Compilar un programa escrito en Basic, que trabaje con el módulo USB del PIC18Fxx5x, en modo HID.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE 1.0.4.6

**Módulos**

- Módulo principal.
- Módulo programador.

**Materiales.**

- PIC18F2550 (Reemplazable modificando el programa)

**Procedimiento.**

1. Abrir Proton IDE.
2. Pegar el contenido del programa.
3. Compilar el programa.
4. Conectar un cristal de 20 Mhz.
5. Cablear desde RC4 hasta D-.
6. Cablear desde RC5 hasta D+.
7. Poner el jumper USB en posición ON.
8. Grabar el programa al microcontrolador.
9. Alimentar la placa principal.
10. Al conectar al USB all computador, se podrá visualizar el dispositivo “Entrenadora” de “UTA - FISEI”.

**Programa.**

```
Declare PROTON_START_ADDRESS = $1000     "Línea para el Bootloader
Device = 18F2550
XTAL = 48                          ' Declara un cristal de 48Mhz, el bootloader lo deja configurado a 48Mhz.

' descriptor file, located in \inc\usb_18 - a copy
' is located in the same folder as this file
USB_DESCRIPTOR = "USBProjectDESC.inc"
' we are going to use an interrupt to keep the USB
' connection alive, so disable auto polling from within
' the USBIn and USBOut commands
USBIN_AUTO_POLL  = OFF
USBOUT_AUTO_POLL = OFF

' USB buffer...
Symbol USBBufferSizeMax = 8
Symbol USBBufferSizeTX  = 8
Symbol USBBufferSizeRX  = 8
Dim    USBBuffer[USBBufferSizeMax] As Byte

' some useful flags...
Dim PP0 As Byte SYSTEM       ' USBPOLL status return
Symbol CARRY_FLAG = STATUS.0  ' high if microcontroller does not have control over the buffer
Symbol ATTACHED_STATE = 6     ' is USB attached
Symbol TRNIF = UIR.3                    ' low if USB Busy
Dim TIMER1 As TMR1L.Word      ' access 16 bits of TMR1

' use an interrupt to keep USB connection alive...
Symbol TimerPreload = $A23F   ' approx 2ms
ON_HARDWARE_INTERRUPT GoTo USBServiceInterrupt
GoTo ProgramStart

' *****************************************************************
' * the USB interrupt service routine maintains the         *
' * connection to the bus - without this routine, a         *
' * call to USBPoll, USBIn or USBOut must be made           *
' * every couple of milliseconds or so                      *
' *****************************************************************
USBServiceInterrupt:
   ' don't poll USB if it's busy...
   btfss TRNIF
   bra ExitInterrupt

   ' poll the USB interface...
   Call (Check@BusStatus)
   Call (Driver@Service)

   ' clear interrupt flag and exit...
ExitInterrupt:
   bcf PIR1,0
   TIMER1 = TimerPreload
   retfie fast

' *****************************************************************
' * program starts here...                                  *
```

```
' ************************************************************
ProgramStart:
  GoSub AttachToUSB

  ' turn on interrupt timer...
  TIMER1 = TimerPreload
  T1CON = %10000001
  PIE1 = %00000001   ' enable TMR1 overflow interrupt
  INTCON = %11000000 ' enable global and peripheral interrupts

' ************************************************************
' * main program loop                                        *
' ************************************************************
ProgramLoop:
  ' your code in here - to send data, load USBBuffer
  ' and then call DoUSBOut. To receive data, call DoUSBIn...


  GoTo ProgramLoop

' ************************************************************
' * receive data from the USB bus                            *
' ************************************************************
DoUSBIn:
  USBIn 1, USBBuffer, USBBufferSizeRX, DoUSBIn
  Return

' ************************************************************
' * transmit data                                            *
' ************************************************************
DoUSBOut:
  USBOut 1, USBBuffer, USBBufferSizeTX, DoUSBOut
  Return

' ************************************************************
' * wait for USB interface to attach                         *
' ************************************************************
AttachToUSB:
  Repeat
    USBPoll
  Until PP0 = ATTACHED_STATE
  Return
```

## Conclusiones.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

## Recomendaciones.

_____
_____
_____
_____

**Reto.**

- Usar el Manual de Proton IDE y el EasyHID USB Wizard para generar un programa que identifique al PIC como "UTA - FISEI".

**Bibliografía.**


- Proton IDE 1.0.4.6 Help Topics.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.


**PRÁCTICA 2.3: USB CDC**

**Introducción.**

El modo CDC (*Communications Devices Class*), permite crear interfaces de comunicación a través del puerto usb.

**Objetivos.**

- Compilar un programa escrito en Basic, que provea un puerto COM virtual utilizando el módulo USB del PIC18Fxx5x.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE 1.0.4.6

**Módulos**

- Módulo principal.
- Módulo programador.

**Materiales.**

- PIC18F2550 (Reemplazable modificando el programa)

**Procedimiento.**

1. Abrir Proton IDE.
2. Pegar el contenido del programa.
3. Compilar el programa.
4. Conectar un cristal de 20 Mhz.
5. Cablear desde RC4 hasta D-.
6. Cablear desde RC5 hasta D+.
7. Poner el jumper USB en posición ON.
8. Cablear todo el puerto B hacia los leds.
9. Grabar el programa al microcontrolador.
10. Alimentar la placa principal.
11. En una terminal serie con configuración 8N1 a 9600; conectar el puerto COM virtual que haya sido asignado.
12. Enviar un número del 1 al 8 para encender o apagar el led conectado al puerto B relacionado.

**Programa.**

```
Declare PROTON_START_ADDRESS = $1000
Device = 18F2550
Declare XTAL = 48
```

```
Declare OPTIMISER_LEVEL = 3
Declare REMINDERS = Off

Declare USB_DESCRIPTOR = "CDCDESC.Inc"
Dim PP0 As Byte SYSTEM
Dim Var1 As Word
Dim Array1[20] As Byte
Dim Out_Buffer As String * 20
Dim In_Buffer As String * 20
Dim Entrada As Byte

Symbol Carry_Flag = STATUS.0
Symbol TRNIF = UIR.3
TRISB=0
PORTB=0
ALL_DIGITAL = True

Repeat
USBPoll
Until PP0 = %00000110
Inicio:
Repeat
USBIn 3, In_Buffer, Auto
Until Carry_Flag = 0
DelayMS 1
USBPoll
Entrada= Val (In_Buffer,Dec)
Clear In_Buffer

Select Case Entrada
Case 1
Toggle PORTB.0
USBPoll
Clear In_Buffer
Case 2
Toggle PORTB.1
USBPoll
Clear In_Buffer
Case 3
Toggle PORTB.2
USBPoll
Clear In_Buffer
Case 4
Toggle PORTB.3
USBPoll
Clear In_Buffer
Case 5
Toggle PORTB.4
USBPoll
Clear In_Buffer
Case 6
Toggle PORTB.5
USBPoll
Clear In_Buffer
Case 7
Toggle PORTB.6
USBPoll
Clear In_Buffer
Case 8
Toggle PORTB.7
USBPoll
Clear In_Buffer
End Select
GoTo Inicio
```

## Conclusiones.

**Recomendaciones.**

**Reto.**

- Modificar el programa para usar otro puerto del PIC.

**Bibliografía.**


- Proton IDE 1.0.4.6 Help Topics.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.

# SECCIÓN 3: APLICACIÓN DE DSP

## PRÁCTICA 3.1: ENTRADAS ANALÓGICAS

**Introducción.**

DSP significa Procesamiento Digital de Señales. Hay varias formas de digitalizar señales para su tratamiento. Algunos modelos de microcontroladores tienen entradas analógicas que sondean una señal analógica; y la convierten a una señal discreta con diferentes grados de resolución. Poseen registros específicos que almacenan el valor medido en tiempo casi real.

**Objetivos.**

- Compilar un programa escrito en Basic, que lea una entrada analógica y muestre un valor legible en función de ella.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE 1.0.4.6

**Módulos**

- Módulo principal.
- Módulo programador.
- Módulo LCD.

**Materiales.**

- PIC18F2550 (Reemplazable modificando el programa)
- LM35.

**Procedimiento.**

17. Abrir Proton IDE.
18. Pegar el contenido del programa.
19. Compilar el programa.
20. Conectar un LM35 al puerto A0.
21. Grabar el programa al microcontrolador.
22. Conectar el módulo LCD.
23. Alimentar la placa principal.
24. Se debe poder leer la temperatura sensada por el LM35.

**Programa.**

```
'*****************************************************************
'* Name    : UNTITLED.BAS                                        *
'* Author  : GabSan                                              *
'* Notice  : Copyright (c) 2013 GabSan                           *
'*         : All Rights Reserved                                 *
'* Date    : 16/10/2013                                          *
'* Version : 1.0                                                 *
'* Notes   :                                                     *
'*         :                                                     *
'*****************************************************************
Declare PROTON_START_ADDRESS = $1000     ''Línea para el Bootloader
Device = 18F2550
XTAL = 48                        ' Declara un cristal de 48Mhz, el bootloader lo deja configurado a 48Mhz.
Declare HSERIAL_BAUD = 9600

ADCON1.7=1 'justificacion derecha
'adcon 6-4 no implementados
ADCON1.3=1 'bits de configuración
ADCON1.2=1 'para entradas A/D
ADCON1.1=1 'solo a.0 analógica
ADCON1.0=0 '

'*****************************************************************
'declaración de variables y constantes
'*****************************************************************
Dim Temp As Float
Dim i As Byte
Dim j As Byte
Symbol Tiempo=50    '' Constante para retardo
```

```
Print At 1,9,"OK"
DelayMS 500
Cls ''Limpiar LCD

'*****************************************************************
'programa principal
'*****************************************************************
Temp = 0        '' Encerar variables
j = 0

medir:                  '' Lazo para medir valores analógcos (cambian en función del tiempo)
   Temp = ADIn 0        '' Asigno a una variable el valor de los registros ADRESH(bits+significativos)
                        '' y ADRESL(bits-significativos) de la conversión
   Temp = Temp - 1      '' Rutina se asegura que
   Temp = Temp + 1      '' valor convertido se pase a Varable Temp
   Temp = Temp/128      '' COnversión de valos binario a decimal
   DelayMS 50
   Print At 1,1,DEC2 Temp," C" '' Imprimir temperatura en LCD
   Print At 2,1,@ADRESH    '' Imprimir registro alto de conversión A/D
   Print At 2,8,@ADRESL    '' Imprimir registro BAJO de conversión A/D

   For j=0 To 10        '' Retardo para medición
      GoSub Retardo
   Next
GoTo medir              '' Vuelve a leer la conversión

'*****************************************************************
'subrutinas
'*****************************************************************
Retardo:    'retardo por instrucciones para q no haga esperar a la interrupción
   For i=0 To Tiempo
      DelayMS 1
   Next
Return
```

## Conclusiones.

## Recomendaciones.

**Reto.**

- Modificar el programa y la implementación para utilizar cualquiera de las otras entradas analógicas del PIC.

**Bibliografía.**


- Proton IDE 1.0.4.6 Help Topics.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.


## PRÁCTICA 3.3: MODULACIÓN POR ANCHO DE PULSO (PWM)

**Introducción.**

Existen microcontroladores que son capaces de generar un ancho de pulso variable, en niveles TTL, lo cual implica que la señal generada variará tanto en frecuencia como en valor eficaz. Estas transformaciones se hacen de acuerdo a la hoja de especificaciones.

**Objetivos.**

- Compilar un programa escrito en Basic, que encienda y apague un led lentamente, utilizando el módulo PWM.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE 1.0.4.6

**Módulos**

- Módulo principal.
- Módulo programador.

**Materiales.**

- PIC18F2550A (Reemplazable modificando el programa)

**Procedimiento.**

1. Abrir Proton IDE.
2. Pegar el contenido del programa.
3. Compilar el programa.
4. Cablear desde RB0 a un led.
5. Grabar el programa al microcontrolador.
6. Conectar el módulo LCD.
7. Alimentar la placa principal.
8. El led conectado a RB0 debe encenderse y apagarse lentamente.

**Programa.**

```
Declare PROTON_START_ADDRESS = $1000     "Línea para el Bootloader
Device = 18F2550
XTAL = 48                    ' Declara un cristal de 48Mhz, el bootloader lo deja configurado a 48Mhz.

ADCON1=15 " Todo portA digital
TRISB = 1 " PortB como entrada
TRISA = 0 " PortA como salida

Dim Loop As Byte

Inf:      Loop = 0           ' Iniciar contador en 0.
          Repeat                        ' Principio del lazo.
          PWM PORTB.0,Loop,30           ' Enviar el valor en Loop por 20ms.
    Inc Loop                 ' Incrementar uno al contador.
    Until Loop = 255 ' Repetir hasta que contador sea 255.

          Repeat                        ' Loop todavía contiene 255.
          PWM PORTB.0,Loop,30           ' Enviar el valor en Loop por 20ms.
```

```
    Dec Loop                ' Restar uno al contador.
    Until Loop = 0          ' Hasta que sea cero.
GoTo Inf                    ' Repetir indefinidamente.
```

**Conclusiones.**

**Recomendaciones.**

**Reto.**

- Modificar el programa, de tal forma que el led aumente la velocidad de encendido/apagado.

**Bibliografía.**


- Proton IDE 1.0.4.6 Help Topics.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.


**PRÁCTICA 3.4: TRASFORMADA RÁPIDA DE FOURIER**

**Introducción.**

A pesar de que existen varios microcontroladores específicos para DSP, que ejecutan cientos de millones de instrucciones por segundo, y ejecutan operaciones complejas con su set nativo de instrucciones, vale la pena comprobar la teoría en microcontroladores de gama media. No serán tan rápidos como los mencionados anteriormente, sin embargo, sirven a la didáctica, proporcionando aplicaciones simples y resultados observables.

Una manera de calcular la DFT (Discrete Fourier Transform), es aplicando la FFT (Fast Fourier Transform) que es un método que en divide la transformada y la realiza aplicando operaciones aritméticas simples.

**Objetivos.**

- Compilar un programa escrito en Basic, que calcule la FFT de una serie dada de valores.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE 1.0.4.6

**Módulos**

- Módulo principal.
- Módulo LCD
- Módulo programador.

**Materiales.**

- PIC18F2550 (Reemplazable modificando el programa)

**Procedimiento.**

1. Abrir Proton IDE.
2. Pegar el contenido del programa principal.
3. Crear un nuevo archivo nombrado como “FFT.inc”
4. Pegar el contenido de FFT.inc
5. Compilar el programa principal.
6. Grabar el programa al microcontrolador.
7. Conectar una señal analógica que varía de 0 a 5v en PORTA.0
8. Conectar el módulo LCD.
9. Alimentar la placa principal.
10. La señal analógica recogida por PORTA.0 debe ser representada en frecuencia en el LCD

**Programa principal.**

```
'
' Demonstrate a 32 element fixed point FFT routine
' Yielding a 16 element result. i.e. FFT_Elements / 2
'
' The spectrum is displayed on a 16x2 alphanumeric LCD
' The actual FFT routines takes approx 1.2 to complete when using a 64MHz oscillator
' Making it almost realtime
'
' Written for the Crownhill Proton compiler version 3.5.3.0 onwards using an 18F device
'
' Author Les Johnson 02-02-2012
'
' You are free to use this code as you see fit, however, a mention of the author and compiler would be appreciated.
' The code is offered as-is and Crownhill nor the author take any responsibility for its use.
'
''    Include "Amicus18.inc"              ' Configure for the Amicus18 device (18F25K20) running at 64MHz
''    Include "Amicus18_ADC.inc"          ' Load the Amicus18 ADC routines into the program
'' NOTA: El programa ha sido utilizado y modificado para que corra en el Hardware de la placa entrenadora UTA - FISEI.
''       Se ha utilizado de acuerdo a los permisos dados por el autor en el presente encabezado.

Declare PROTON_START_ADDRESS = $1000     "Línea para el Bootloader
Device = 18F2550
XTAL = 48

''   $define FFT_Use_Sqr                ' Instruct the FFT routine to use more refined real and imaginary combining
   $define FFT_Elements 32              ' The amount of elements for the FFT routine to work on
   Include "FFT.inc"                 ' Load the 8-bit FFT routines into the program
'
' Setup the Alphanumeric LCD
'
   Declare LCD_DTPIN = PORTB.4          ' LCD's Data lines (D4 to D7)
   Declare LCD_ENPIN = PORTB.2          ' LCD's EN line
   Declare LCD_RSPIN = PORTB.3          ' LCD's RS line
```

```
   Declare LCD_INTERFACE = 4            ' 4-bit interface to LCD
   Declare LCD_LINES = 2                ' LCD contains 2 lines
   Declare LCD_TYPE = ALPHANUMERIC      ' LCD type is alphanumeric
'
' Create variables used within the scale subroutine
'
   Dim Map_bTemp1 As Byte
   Dim Map_bTemp2 As Byte
   Dim Map_wTemp As Word                ' Holds the result of the scale calculation

   Dim Map_bXin As Map_bTemp1           ' Value is the value to Scale
   Dim Map_bInMin As Byte               ' Lower bound of the value's current range
   Dim Map_bInMax As Byte               ' Upper bound of the value's current range
   Dim Map_bOutMin As Byte              ' Lower bound of the value's target range
   Dim Map_bOutMax As Map_bTemp2        ' Upper bound of the value's target range
'
' Create variables used for display of the spectrum
'
   Dim bIndex As Byte                   ' General purpose index variable
   Dim bLCD_Xpos As Byte                ' Holds the x position on the LCD
   Dim wCharData As Word                ' Holds the two individual LCD bar patterns
   Dim bBarValue As Byte                ' Holds the value for the bar display on the LCD

'-------------------------------------------------------------------------------
   GoTo Main                            ' Jump over the subroutines
'-------------------------------------------------------------------------
' Table of LCD bars required for a given value
'
BarTable:
   CData 32, 32,_
         32, 08, 32, 09, 32, 10, 32, 11, 32, 12, 32, 13, 32, 14, 32, 15,_
         08, 15, 09, 15, 10, 15, 11, 15, 12, 15, 13, 15, 14, 15, 15, 15

'-------------------------------------------------------------------------
' Table of bit patterns for the bars on the LCD
'
CharTable:
   CData $FE, $40,_
         %00000, %00000, %00000, %00000, %00000, %00000, %00000, %11111,_
         %00000, %00000, %00000, %00000, %00000, %00000, %11111, %11111,_
         %00000, %00000, %00000, %00000, %00000, %11111, %11111, %11111,_
         %00000, %00000, %00000, %00000, %11111, %11111, %11111, %11111,_
         %00000, %00000, %00000, %11111, %11111, %11111, %11111, %11111,_
         %00000, %00000, %11111, %11111, %11111, %11111, %11111, %11111,_
         %00000, %11111, %11111, %11111, %11111, %11111, %11111, %11111,_
         %11111, %11111, %11111, %11111, %11111, %11111, %11111, %11111
'-------------------------------------------------------------------------------
' Scale one 8-bit value to another 8-bit value
' Input     : Map_bXin holds the value to scale
'           : Map_bInMin is the lower bound of the value's current range
'           : Map_bInMax is the upper bound of the value's current range
'           : Map_bOutMin is the lower bound of the value's target range
'           : Map_bOutMax is the upper bound of the value's target range
' Output    : pResult holds the result
' Notes     : Traps a zero value for extra speed
'
$define Map8(pX, pInMin, pInMax, pOutMin, pOutMax, pResult) '
   Map_bXin = pX        '
   Map_bInMin = pInMin  '
   Map_bInMax = pInMax  '
   Map_bOutMin = pOutMin '
   Map_bOutMax = pOutMax '
   _Map8                '
   pResult = Map_wTemp

_Map8 MACRO-
   GoSub __Map8
ENDM

#ifMacro- _Map8
__Map8:
   If Map_bXin = 0 Then                 ' Is the value to scale zero?
      Map_wTemp = 0                     ' Yes. So return a zero
```

```
   Else                                ' Otherwise...
      Map_bTemp1 = Map_bXin - Map_bInMin
      Map_bTemp2 = Map_bOutMax - Map_bOutMin
      Map_wTemp = Map_bTemp1 * Map_bTemp2
      Map_bTemp1 = Map_bInMax - Map_bInMin
      Map_bTemp2 = Map_wTemp / Map_bTemp1
      Map_wTemp = Map_bTemp2 + Map_bOutMin
   EndIf
   Return
#endIfMacro-
'-----------------------------------------------------------------------
' Create the bit patterns that make up the bars in the LCD's CGRAM.
' Input     : Code memory table CharTable holds the bit patterns
' Output    : None
' Notes     : None
'
$define CreateBarPatterns() GoSub _CreateBarPatterns

_CreateBarPatterns:
   bIndex = 0
   Repeat                              ' Create loop to read from the data table
      Print CRead8 CharTable[bIndex]           ' Transfer the table values to the LCD's CGRAM
      Inc bIndex                       ' Move up a byte
   Until bIndex >= 66                  ' Until all GCGRAM is filled
   Return
'-----------------------------------------------------------------------
' Display a vertical bar on the alphanumeric LCD
' Input     : bBarValue holds the value to display as a vertical bar on the LCD
'           : bLCD_Xpos is the x position on the LCD to display the bar
' Output    : Prints a bar on the LCD
' Notes     : None
'
$define DisplayBar() GoSub _DisplayBar

_DisplayBar:
   If bBarValue > 0 Then               ' Is there a bar to display?
      If bBarValue >= 32 Then bBarValue = 32     ' Yes. So make sure a maximum is reached
      Map8(bBarValue, 0,31, 0, 16, bBarValue)    ' Scale the value from (0 to 31) to (0 to 16)
      wCharData = CRead16 BarTable[bBarValue]    ' Read top and bottom bar pattern from the table
      Print At 1, bLCD_Xpos, wCharData.BYTE0, At 2, bLCD_Xpos, wCharData.BYTE1 ' Display the bar on the LCD
   Else                                ' Otherwise...
      Print At 1, bLCD_Xpos, " ", At 2, bLCD_Xpos, " " ' Clear the bar
   EndIf
   Return
'-----------------------------------------------------------------------
' The main program loop starts here
'
Main:
'
' Open the ADC for samples taken on AN0
' Left justify for 8-bit result
'
''    OpenADC(ADC_FOSC_64 & ADC_LEFT_JUST & ADC_2_TAD, ADC_REF_VDD_VSS, ADC_1ANA)
TRISA = %00000001 ' Configure AN0 (PortA.0) as an input
ADCON0 = %10000001
ADCON1 = %11001101 ' ADC_REF_VDD_VSS Set analogue input on PortA.0
ADCON2 = %01001110  ''  ADC_LEFT_JUST  ADC_2_TAD  ADC_FOSC_64

   Cls                                 ' Clear the LCD
   CreateBarPatterns()                 ' Create the bit patterns that make up the LCD's bars

   While 1 = 1                         ' Create an infinite loop
      '
      ' Fill the real array with 8-bit ADC values
      '
      bIndex = 0
      Repeat
         ''FFT_bRealData[bIndex] = ReadADC(ADC_CH0)       ' Read the 8-bit ADC value into the array
         FFT_bRealData[bIndex] = ADIn 0
         FFT_bRealData[bIndex] = FFT_bRealData[bIndex]-1
         FFT_bRealData[bIndex] = FFT_bRealData[bIndex]+1
         DelayUS 52                    ' Delay for a sample rate of approx 16KHz (FFT range 0 to 8KHz)
         'DelayUS 250                  ' Delay for a sample rate of approx 4KHz (FFT range 0 to 2KHz)
```

```
      Inc bIndex
    Until bIndex >= cFFT_NumberOfSamples          ' Repeat for n samples

    Fix_FFT()                                     ' Perform the FFT (Result in array FFT_bRealData)
'
' Display the spectrum on a 16x2 alphanumeric LCD
' As columns of vertical bars
'
    bLCD_Xpos = 1                                 ' Start at column 1 of the LCD
    bIndex = 0                                    ' Create a loop for all the values in the real array
    Repeat
      '
      ' We have 16 columns on the LCD, and 16 pieces of data from the FFT, So...
      ' Extract a value from the real array and use this as the bar value on the LCD
      '
      bBarValue = FFT_bRealData[bIndex]           ' Extract the value
      DisplayBar()                                ' Display a bar on the LCD
      Inc bLCD_Xpos                               ' Next column on the LCD
      If bLCD_Xpos > 16 Then bLCD_Xpos = 1        ' Back to column 1 when the end of the LCD is reached
      Inc bIndex
    Until bIndex >= cFFT_NumberOfRealElements     ' Until all the elements of the FFT are read
  Wend
```

## Programa “FFT.inc”

```
'
' Crownhill Proton compiler 8-bit fixed point FFT routine
' For use with version 3.5.3.0 onwards of the compiler and 18F devices.
'
' Written by Les Johnson 02-02-2012
'
' You are free to use this code as you see fit, however, a mention of the author and compiler would be appreciated.
' The code is offered as-is and Crownhill nor the author take any responsibility for its use.
''  NOTA: El programa ha sido utilizado y modificado para que corra en el Hardware de la placa entrenadora UTA - FISEI.
''       Se ha utilizado de acuerdo a los permisos dados por el autor en el presente encabezado.
'' Al descomentar la línea 391, se envían los datos de forma serial pero se ralentiza la transformada.

$ifndef FFT_Elements
$define FFT_Elements 64                   ' Default elements if no define is used in the main program
$endif
'
' Trap the element value
'
$If (FFT_Elements <> 128) And (FFT_Elements <> 64) And (FFT_Elements <> 32) And (FFT_Elements <> 16)
   $error "Incorret amount of elements. 128, 64, 32 or 16 required"
$endif

   $define FFT_NWave $eVal FFT_Elements * 2       ' Calculate the size of the Sin and Cos waves required
'
' Create variables used for the merging of imaginary and real data
'
   Dim FFT_wReal As Word SYSTEM
   Dim FFT_wImaginary As Word SYSTEM
   Dim FFT_bImaginary As FFT_wImaginary.BYTE0
   Dim FFT_bReal As FFT_wReal.BYTE0
'
' Create variables used by Fix_FFT
' Made mostly system types so they stay is bankless Access RAM for speed
'
   Dim FFT_bTemp1 As Byte SYSTEM
   Dim FFT_bTemp2 As Byte SYSTEM
   Dim FFT_bTemp3 As Byte SYSTEM
   Dim FFT_bTemp4 As Byte SYSTEM

   Dim FFT_bM As Byte SYSTEM
   Dim FFT_bMR As Byte SYSTEM
   Dim FFT_bJ As Byte SYSTEM
   Dim FFT_bL As Byte SYSTEM
   Dim FFT_bK As Byte SYSTEM

   Dim FFT_bIstep As Byte SYSTEM
   Dim FFT_bIndex As Byte SYSTEM

   Dim FFT_bQR As Byte SYSTEM
   Dim FFT_bQI As Byte SYSTEM
   Dim FFT_bTempReal As Byte SYSTEM
   Dim FFT_bTempImaginary As Byte SYSTEM
   Dim FFT_bWR As Byte SYSTEM
   Dim FFT_bWI As Byte SYSTEM
```

```
   Dim FFT_bRealData[FFT_Elements] As Byte Heap       ' Holds the real data
   Dim FFT_bImaginaryData[FFT_Elements] As Byte Heap  ' Holds the imaginary data

   Dim FXPM_wPROD As PRODL.Word                       ' Make a 16-bit variable from PRODL\H

   Symbol cFFT_NumberOfSamples = FFT_Elements         ' The amount of elements in the FFT
   Symbol cFFT_NumberOfRealElements = FFT_Elements / 2 ' The amount of real elements after the FFT has performed its task
   Symbol cNWave_Div4 = FFT_NWave / 4                 ' Length of SineWave divided by 4

$If FFT_NWave = 256
   $define cLog2_Nwave 8                          ' Log2(256)
$elseif FFT_NWave = 128
   $define cLog2_Nwave 7                          ' Log2(128)
$elseif FFT_NWave = 64
   $define cLog2_Nwave 6                          ' Log2(64)
$elseif FFT_NWave = 32
   $define cLog2_Nwave 5                          ' Log2(32)
$elseif FFT_NWave = 16
   $define cLog2_Nwave 4                          ' Log2(16)
$endif
'-------------------------------------------------------------------------------
   GoTo _FFT_Main_                               ' Jump over the subroutines
'-------------------------------------------------------------------------------
' 8-bit Signed Fixed-Point multiplication & scaling.
' Input     : pMultiplier holds the 8-bit value to multiply
'           : pMultiplicand holds the 8-Bit value to multiply with
' Output    ' pResult holds the resulting product of the multiplication
' Notes     : Scaling ensures that result remains within 16-bits.
'           : The code is rolled out for speed instead of using loops
'
$define Fixed_Multiply(pMultiplier, pMultiplicand, pResult) '
   If pMultiplicand = 0 Then       '
      pResult = 0                  '
   ElseIf pMultiplier = 0 Then     '
      pResult = 0                  '
   Else                            '
      movf pMultiplier,w,0         '
      Mulwf pMultiplicand,0        '
      btfsc pMultiplicand,7,0      '
      subwf PRODH,f,0              '
      btfss pMultiplier,7,0        '
      Bra $ + 6                    '
      movf pMultiplicand,w,0       '
      subwf  PRODH,f,0             '
      Rrcf   PRODH,f,0             '
      Rrcf   PRODL,f,0             '
      Rrcf   PRODH,f,0             '
      Rrcf   PRODL,f,0             '
      Rrcf   PRODH,f,0             '
      Rrcf   PRODL,f,0             '
      Rrcf   PRODH,f,0             '
      Rrcf   PRODL,f,0             '
      Rrcf   PRODH,f,0             '
      Rrcf   PRODL,f,0             '
      Rrcf   PRODH,f,0             '
      Rrcf   PRODL,f,0             '
      movlw  3                     '
      andwf  PRODH,f,0             '
      WREG = PRODL & 1             '
      bcf    STATUS,C,0            '
      Rrcf   PRODH,f,0             '
      Rrcf   PRODL,f,0             '
      btfsc  PRODH,6,0             '
      bsf    PRODH,7,0             '
      pResult = FXPM_wPROD + WREG  '
   EndIf

'-------------------------------------------------------------------------
' Create 8-bit sine wave data
'
SineWave:
$If FFT_NWave = 256
   CData As Byte 000, -002, -003, -005, -006, -008, -009, -011,_
                 -012, -014, -016, -017, -019, -020, -022, -023,_
                 -024, -026, -027, -029, -030, -032, -033, -034,_
                 -036, -037, -038, -039, -041, -042, -043, -044,_
                 -045, -046, -047, -048, -049, -050, -051, -052,_
                 -053, -054, -055, -056, -056, -057, -058, -059,_
                 -059, -060, -060, -061, -061, -062, -062, -062,_
                 -063, -063, -063, -064, -064, -064, -064, -064,_
                 -064, -064, -064, -064, -064, -064, -063, -063,_
                 -063, -062, -062, -062, -061, -061, -060, -060,_
                 -059, -059, -058, -057, -056, -056, -055, -054,_
```

```
            -053, -052, -051, -050, -049, -048, -047, -046,_
            -045, -044, -043, -042, -041, -039, -038, -037,_
            -036, -034, -033, -032, -030, -029, -027, -026,_
            -024, -023, -022, -020, -019, -017, -016, -014,_
            -012, -011, -009, -008, -006, -005, -003, -002,_
            000, 002, 003, 005, 006, 008, 009, 011,_
            012, 014, 016, 017, 019, 020, 022, 023,_
            024, 026, 027, 029, 030, 032, 033, 034,_
            036, 037, 038, 039, 041, 042, 043, 044,_
            045, 046, 047, 048, 049, 050, 051, 052,_
            053, 054, 055, 056, 056, 057, 058, 059,_
            059, 060, 060, 061, 061, 062, 062, 062,_
            063, 063, 063, 064, 064, 064, 064, 064,_
            064, 064, 064, 064, 064, 064, 063, 063,_
            063, 062, 062, 062, 061, 061, 060, 060,_
            059, 059, 058, 057, 056, 056, 055, 054,_
            053, 052, 051, 050, 049, 048, 047, 046,_
            045, 044, 043, 042, 041, 039, 038, 037,_
            036, 034, 033, 032, 030, 029, 027, 026,_
            025, 023, 022, 020, 019, 017, 016, 014,_
            012, 011, 009, 008, 006, 005, 003, 002
$elseif FFT_NWave = 128
   CData As Byte 000, -003, -006, -009, -012, -016, -019, -022,_
            -024, -027, -030, -033, -036, -038, -041, -043,_
            -045, -047, -049, -051, -053, -055, -056, -058,_
            -059, -060, -061, -062, -063, -063, -064, -064,_
            -064, -064, -064, -063, -063, -062, -061, -060,_
            -059, -058, -056, -055, -053, -051, -049, -047,_
            -045, -043, -041, -038, -036, -033, -030, -027,_
            -024, -022, -019, -016, -012, -009, -006, -003,_
            000, 003, 006, 009, 012, 016, 019, 022,_
            024, 027, 030, 033, 036, 038, 041, 043,_
            045, 047, 049, 051, 053, 055, 056, 058,_
            059, 060, 061, 062, 063, 063, 064, 064,_
            064, 064, 064, 063, 063, 062, 061, 060,_
            059, 058, 056, 055, 053, 051, 049, 047,_
            045, 043, 041, 038, 036, 033, 030, 027,_
            025, 022, 019, 016, 012, 009, 006, 003
$elseif FFT_NWave = 64
   CData As Byte 000, -006, -012, -019, -024, -030, -036, -041,_
            -045, -049, -053, -056, -059, -061, -063, -064,_
            -064, -064, -063, -061, -059, -056, -053, -049,_
            -045, -041, -036, -030, -024, -019, -012, -006,_
            000, 006, 012, 019, 024, 030, 036, 041,_
            045, 049, 053, 056, 059, 061, 063, 064,_
            064, 064, 063, 061, 059, 056, 053, 049,_
            045, 041, 036, 030, 025, 019, 012, 006
$elseif FFT_NWave = 32
   CData As Byte 000, -012, -024, -036, -045, -053, -059, -063,_
            -064, -063, -059, -053, -045, -036, -024, -012,_
            000, 012, 024, 036, 045, 053, 059, 063,_
            064, 063, 059, 053, 045, 036, 025, 012
$elseif FFT_NWave = 16
   CData As Byte 000, -024, -045, -059, -064, -059, -045, -024,_
            000, 024, 045, 059, 064, 059, 045, 025
$else
   $error "Unknown FFT_NWave length"
$endif
'------------------------------------------------------------------------
' Create 8-bit cosine wave data
'
CosineWave:
$If FFT_NWave = 256
   CData As Byte 064, 064, 064, 064, 064, 064, 063, 063,_
            063, 062, 062, 062, 061, 061, 060, 060,_
            059, 059, 058, 057, 056, 056, 055, 054,_
            053, 052, 051, 050, 049, 048, 047, 046,_
            045, 044, 043, 042, 041, 039, 038, 037,_
            036, 034, 033, 032, 030, 029, 027, 026,_
            024, 023, 022, 020, 019, 017, 016, 014,_
            012, 011, 009, 008, 006, 005, 003, 002,_
            000, -002, -003, -005, -006, -008, -009, -011,_
            -012, -014, -016, -017, -019, -020, -022, -023,_
            -024, -026, -027, -029, -030, -032, -033, -034,_
            -036, -037, -038, -039, -041, -042, -043, -044,_
            -045, -046, -047, -048, -049, -050, -051, -052,_
            -053, -054, -055, -056, -056, -057, -058, -059,_
            -059, -060, -060, -061, -061, -062, -062, -062,_
            -063, -063, -063, -064, -064, -064, -064, -064,_
            -064, -064, -064, -064, -064, -064, -063, -063,_
            -063, -062, -062, -062, -061, -061, -060, -060,_
            -059, -059, -058, -057, -056, -056, -055, -054,_
            -053, -052, -051, -050, -049, -048, -047, -046,_
```

```
            -045, -044, -043, -042, -041, -039, -038, -037,_
            -036, -034, -033, -032, -030, -029, -027, -026,_
            -024, -023, -022, -020, -019, -017, -016, -014,_
            -012, -011, -009, -008, -006, -005, -003, -002,_
            000, 002, 003, 005, 006, 008, 009, 011,_
            012, 014, 016, 017, 019, 020, 022, 023,_
            024, 026, 027, 029, 030, 032, 033, 034,_
            036, 037, 038, 039, 041, 042, 043, 044,_
            045, 046, 047, 048, 049, 050, 051, 052,_
            053, 054, 055, 056, 056, 057, 058, 059,_
            059, 060, 060, 061, 061, 062, 062, 062,_
            063, 063, 063, 064, 064, 064, 064, 064
$elseif FFT_NWave = 128
   CData As Byte 064, 064, 064, 063, 063, 062, 061, 060,_
            059, 058, 056, 055, 053, 051, 049, 047,_
            045, 043, 041, 038, 036, 033, 030, 027,_
            024, 022, 019, 016, 012, 009, 006, 003,_
            000, -003, -006, -009, -012, -016, -019, -022,_
            -024, -027, -030, -033, -036, -038, -041, -043,_
            -045, -047, -049, -051, -053, -055, -056, -058,_
            -059, -060, -061, -062, -063, -063, -064, -064,_
            -064, -064, -064, -063, -063, -062, -061, -060,_
            -059, -058, -056, -055, -053, -051, -049, -047,_
            -045, -043, -041, -038, -036, -033, -030, -027,_
            -024, -022, -019, -016, -012, -009, -006, -003,_
            000, 003, 006, 009, 012, 016, 019, 022,_
            024, 027, 030, 033, 036, 038, 041, 043,_
            045, 047, 049, 051, 053, 055, 056, 058,_
            059, 060, 061, 062, 063, 063, 064, 064
$elseif FFT_NWave = 64
   CData As Byte 064, 064, 063, 061, 059, 056, 053, 049,_
            045, 041, 036, 030, 024, 019, 012, 006,_
            000, -006, -012, -019, -024, -030, -036, -041,_
            -045, -049, -053, -056, -059, -061, -063, -064,_
            -064, -064, -063, -061, -059, -056, -053, -049,_
            -045, -041, -036, -030, -024, -019, -012, -006,_
            000, 006, 012, 019, 024, 030, 036, 041,_
            045, 049, 053, 056, 059, 061, 063, 064
$elseif FFT_NWave = 32
   CData As Byte 064, 063, 059, 053, 045, 036, 024, 012,_
            000, -012, -024, -036, -045, -053, -059, -063,_
            -064, -063, -059, -053, -045, -036, -024, -012,_
            000, 012, 024, 036, 045, 053, 059, 063
$elseif FFT_NWave = 16
   CData As Byte 064, 059, 045, 024, 000, -024, -045, -059,_
            -064, -059, -045, -024, 000, 024, 045, 059
$else
   $error "Unknown FFT_NWave length"
$endif

'------------------------------------------------------------------------
' Perform a forward Fast Fourier Transform
' Input     : Array FFT_bRealData holds the 8-bit ADC samples to perform the FFT upon
' Output    : Array FFT_bRealData holds real data from the FFT
'           : Array FFT_bImaginaryData holds imaginary data from the FFT
' Notes     : None
'
$define Fix_FFT() GoSub _Fix_FFT

_Fix_FFT:
   Clear FFT_bImaginaryData                ' Clear the imaginary data array
'
' Decimation in time - re-order data
'
   FFT_bMR = 0
   FFT_bM = 1
   Repeat
      FFT_bL = cFFT_NumberOfSamples
      Repeat
         FFT_bL = FFT_bL >> 1
         FFT_bTemp1 = FFT_bMR + FFT_bL
      Until FFT_bTemp1 < cFFT_NumberOfSamples
      FFT_bTemp1 = FFT_bL - 1
      FFT_bMR = FFT_bMR & FFT_bTemp1
      FFT_bMR = FFT_bMR + FFT_bL
      If FFT_bMR > FFT_bM Then
         FFT_bTempReal = FFT_bRealData[FFT_bM]
         FFT_bRealData[FFT_bM] = FFT_bRealData[FFT_bMR]
         FFT_bRealData[FFT_bMR] = FFT_bTempReal
      EndIf
      Inc FFT_bM
   Until FFT_bM >= cFFT_NumberOfSamples
'
```

```
' Perform the FFT
'
   FFT_bL = 1
   FFT_bK = cLog2_Nwave - 1
   While FFT_bL < cFFT_NumberOfSamples
      FFT_bIstep = FFT_bL << 1
      FFT_bM = 0
      Repeat
         FFT_bJ = FFT_bM << FFT_bK
         FFT_bWR = CRead8 CosineWave[FFT_bJ]                ' Cosine
         FFT_bWI = CRead8 SineWave[FFT_bJ]                  ' Sine
         FFT_bIndex = FFT_bM
         Repeat
            FFT_bJ = FFT_bIndex + FFT_bL

            FFT_bTemp3 = FFT_bRealData[FFT_bJ]
            FFT_bTemp4 = FFT_bImaginaryData[FFT_bJ]

            Fixed_Multiply(FFT_bWR, FFT_bTemp3 , FFT_bTemp1)    ' 8-bit Fixed point multiply into FFT_bTemp1
            Fixed_Multiply(FFT_bWI, FFT_bTemp4 , FFT_bTemp2)    ' 8-bit Fixed point multiply  into FFT_bTemp2
            FFT_bTempReal = FFT_bTemp1 - FFT_bTemp2

            Fixed_Multiply(FFT_bWR, FFT_bTemp4 , FFT_bTemp1)    ' 8-bit Fixed point multiply into FFT_bTemp1
            Fixed_Multiply(FFT_bWI, FFT_bTemp3 , FFT_bTemp2)    ' 8-bit Fixed point multiply into FFT_bTemp2
            FFT_bTempImaginary = FFT_bTemp1 + FFT_bTemp2

            FFT_bQR = FFT_bRealData[FFT_bIndex]
            FFT_bQR = FFT_bQR.SByte / 2                 ' Reduce the chance of overflow
            FFT_bQI = FFT_bImaginaryData[FFT_bIndex]
            FFT_bQI = FFT_bQI.SByte / 2                 ' Reduce the chance of overflow
            FFT_bRealData[FFT_bJ] = FFT_bQR - FFT_bTempReal
            FFT_bImaginaryData[FFT_bJ] = FFT_bQI - FFT_bTempImaginary
            FFT_bRealData[FFT_bIndex] = FFT_bQR + FFT_bTempReal
            FFT_bImaginaryData[FFT_bIndex] = FFT_bQI + FFT_bTempImaginary
            FFT_bIndex = FFT_bIndex + FFT_bIstep
         Until FFT_bIndex >= cFFT_NumberOfSamples
         Inc FFT_bM
      Until FFT_bM >= FFT_bL
      FFT_bK = FFT_bK - 1
      FFT_bL = FFT_bIstep
   Wend
' Fall through to "_ConvertToReal"
'-------------------------------------------------------------
' Convert the real and imaginary arrays to a single real array
' Input     : Array FFT_bRealData holds the real data from the FFT
'           : Array FFT_bImaginaryData holds the imaginary data from the FFT
' Output    : Results are placed back into the 'FFT_bRealData' array
' Notes     : There are two ways to compute the real part:
'           : 1). Find the real value by the square root of the squares of real and imaginary values
'           : 2). Find the real value by summing the real and imaginary parts (quicker, but dirty)
'           : Also removed the first element which is the DC component
'
_ConvertToReal:
   FFT_bIndex = 0
   Repeat
      FFT_bReal = FFT_bRealData[FFT_bIndex]             ' Extract a value from the real array
      FFT_bReal = Abs FFT_bReal                         ' Make sure it's positive
      FFT_bImaginary = FFT_bImaginaryData[FFT_bIndex]   ' Extract a value from the imaginary array
      FFT_bImaginary = Abs FFT_bImaginary               ' Make sure it's positive
      '
      ' Quick and dirty method
      '
$ifndef FFT_Use_Sqr
      FFT_bRealData[FFT_bIndex] = FFT_bReal + FFT_bImaginary    ' Place the summed value back into the RealData array (quick and
dirty method)
$else
      '
      ' Slower but more refined method (also traps zero values for extra speed)
      '
      If FFT_bImaginary = 0 Then                        ' Is FFT_bImaginary zero?
         If FFT_bReal = 0 Then                          ' Yes. So is FFT_bReal zero?
            FFT_bRealData[FFT_bIndex] = 0               ' Yes. So place a zero into the array
         Else                                           ' Otherwise...
            FFT_wReal = FFT_bReal * FFT_bReal           ' Square the real value
            FFT_bRealData[FFT_bIndex] = ISqr FFT_wReal  ' Find the square root of real
         EndIf
      ElseIf FFT_bReal = 0 Then                         ' Otherwise. Is FFT_bReal zero?
         If FFT_bImaginary = 0 Then                     ' Yes. So is FFT_bImaginary zero?
            FFT_bRealData[FFT_bIndex] = 0               ' Yes. So place a zero into the array
         Else                                           ' Otherwise...
            FFT_wImaginary = FFT_bImaginary * FFT_bImaginary    ' Square the imaginary value
            FFT_bRealData[FFT_bIndex] = ISqr FFT_wImaginary     ' Find the square root of imaginary
         EndIf
```

```
      Else                                        ' Otherwise. None are zero
        FFT_wImaginary = FFT_bImaginary * FFT_bImaginary      ' Square the imaginary value
        FFT_wReal = FFT_bReal * FFT_bReal                     ' Square the real value
        FFT_bRealData[FFT_bIndex] = ISqr FFT_wImaginary + FFT_wReal ' Find the square root of real and imaginary
      EndIf
$endif
      ''HSerOut [@FFT_bRealData[FFT_bIndex],13]               ''ENVÍA LOS VALORES POR PUERTO SERIE
      Inc FFT_bIndex
   Until FFT_bIndex >= cNWave_Div4
   FFT_bRealData#0 = 0                                ' Remove the DC content from the first element
   Return
'-------------------------------------------------------------------------------
_FFT_Main_:
```

# SECCIÓN 4: ROBÓTICA

## PRÁCTICA 4.1: CONTROL DE SERVO MOTORES CON PIC

**Introducción.**

Los microcontroladores, por su bajo costo y simplicidad, son de los más usados para la robótica. Los servo motores son dispositivos controlados por PWM que tienen una gran relación potencia – tracción.

**Objetivos.**

- Compilar un programa escrito en Basic, que controle un servo motor.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE

**Módulos**

- Módulo principal.
- Módulo programador.
- Módulo Relevadores.

**Materiales.**

- PIC18F2550 (Reemplazable modificando el programa)

**Procedimiento.**

1. Abrir Proton IDE.
2. Pegar el contenido del programa.
3. Compilar el programa.
4. Conectar el módulo relevadores.
5. Cablear desde BORNERA SALIDA RA.0 hasta el control PWM del servo.
6. Grabar el programa al microcontrolador.
7. Alimentar la placa principal.
8. El servo deberá girar alternativamente de derecha a izquierda.

**Programa.**

```
Declare PROTON_START_ADDRESS = $1000     "Línea para el Bootloader
Device = 18F2550
XTAL = 48                          ' Declara un cristal de 48Mhz, el bootloader lo deja configurado a 48Mhz.

'****************************************************************
'declaración de variables
'****************************************************************
Dim Posicion As Word    "Variable para posición
ADCON1=%11111111        "Todo PORTA digital

DelayMS 150
Cls

Posicion = 1500

'****************************************************************
'programa principal
'****************************************************************
Inicio:

 Repeat                    " Inicio del lazo
   Servo PORTA.0, Posicion     " Enviar el valor en Posición
   DelayMS 20                " Esperar 20 ms
   Inc Posicion              " Incrementar uno al contador.
   Print At 1, 1, "Servo = ", DEC4 Posicion    " Imprimir en LCD para debug
 Until Posicion = 2000         " Repetir hasta que contador sea 2000.

 Repeat                    " Inicio del lazo
    Servo PORTA.0, Posicion    " Enviar el valor en Posición
    DelayMS 20               " Esperar 20 ms
    Dec Posicion             " Restar uno al contador.
    Print At 1, 1, "Servo = ", DEC4 Posicion   " Imprimir en LCD para debug
 Until Posicion = 1000         " Repetir hasta que contador sea 1000.
```

**GoTo** Inicio

**Conclusiones.**

**Recomendaciones.**

**Reto.**

- Modificar el programa, para utilizar un puerto diferente en la implementación.

**Bibliografía.**


- Proton IDE 1.0.4.6 Help Topics.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.


**PRÁCTICA 4.2: CONTROL DE GIRO DE MOTORES DC CON PIC**

**Introducción.**

Existe gran variedad de motores en el mercado, de los cuales los más asequibles son los de tipo DC. Incluso se pueden conseguir sin costo alguno a partir de chatarra electrónica. Esta circunstancia los hace ideales para la experimentación. Cuando se trabaja con Puente H es importante no alimentar las dos entradas a la vez, ya que se podría destruir el equipo.

**Objetivos.**

- Compilar un programa escrito en C, que gire alternativamente un motor DC.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- MPLAB X 1.60
- Compilador XC18 v1.12

**Módulos**

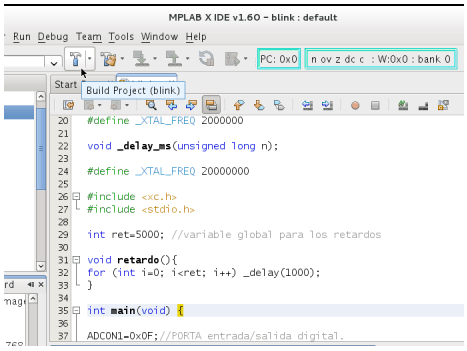
- Módulo principal.
- Módulo programador.
- Módulo Relevadores.

**Materiales.**

- PIC18F2550 (Reemplazable modificando el programa)

**Procedimiento.**

1. Abrir MPLAB X.
2. Crear en nuevo proyecto de acuerdo al proceso descrito en Práctica 1.2.
3. Pegar el contenido del programa.
4. Presionar el botón *Build Project.* Se debe obtener como resultado *BUILD SUCCEEDED*, en el panel *OUTPUT*.




5. Grabar el .hex generado al PIC.
6. Cablear de PORTA.0 y PORTA.1 a pulsadores.
7. Conectar el módulo Relevadores.
8. Conectar las terminales del motor dc a la bornera “MOTOR”
9. Asegurarse de que el jumper de selección de alimentación esté en la posición de la fuente adecuada. Es importante no sobrecargar el puerto USB del ordenador.
10. Alimentar la placa principal.
11. Al pulsar PORTA.0 el motor girará en sentido antihorario.
12. Al pulsar PORTA.1 el motor girará en sentido horario.

**Programa.**

```
/*
 * File:   motor_uno.c
 * Author: gabriel
 *
 * Created on 20 de agosto de 2013, 09:58 AM
 */
#include "xc.h"         // Cabecera genérica de dispositivos
#include <stdio.h>  //Cabecera para funciones de entrada/salida.

#pragma config FOSC = HS    // Oscilador HS

#define _XTAL_FREQ 4800000  // Define frecuencia trabajo del PIC.

void retardo()      // Retardo por instrucciones
{
   int i, j;
   for(j=0x001; j<=0x01; j++)
   {
      for(i=0x001; i<=0x1388; i++);
   }
}
```

```
int main(void)          // Rutina principal
{
ADCON1=0xFF;            // PORTA digital
TRISA=0XFF;                     //Todo PORTA como entrada.
TRISB=0x00;                     //Todo PORTB como salida.
PORTAbits.RA0 = 0;      // Inicializo variables.
PORTAbits.RA1 = 0;
PORTBbits.RB0 = 0;
PORTBbits.RB1 = 0;

while(1)                // Bucle infinito
   {
   if(PORTAbits.RA0&&PORTAbits.RA1==1) // Si ambos botones están pulsados.
   {
   PORTBbits.RB0=0;    // Poner PORTB.0 a estado bajo
   PORTBbits.RB1=0;    // Poner PORTB.1 a estado bajo
   }
   else             // Si alguno o ninguno de los pulsadores está accionado
   {
   if(PORTAbits.RA0==1)    // Si pulsador en PORTA.0 está accionado
      {
      PORTBbits.RB0=1;       // Poner PORTB.0 a estado alto y,
      PORTBbits.RB1=0;       // Poner PORTB.1 a estado bajo
      }
      else                 // Si pulsador en PORTA.0 NO está accionado
      {
         if(PORTAbits.RA1==1)    // Y pulsador en PORTA.1 está accionado
         {
         PORTBbits.RB0=0;       // Poner PORTB.0 a estado bajo
         PORTBbits.RB1=1;       // Poner PORTB.0 a estado alto
         }
         PORTBbits.RB0=0;       // Poner PORTB.0 a estado bajo
         PORTBbits.RB1=0;       // Poner PORTB.0 a estado bajo
      }
   }
   }
return 0;   // Regresa valor cero si se ha completado la función main.
}
```

**Conclusiones.**

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

**Recomendaciones.**

_____
_____
_____
_____

**Reto.**

- Modificar el programa, para utilizar PWM y controlar la velocidad y el sentido de giro.

**Bibliografía.**


- MPLAB® X IDE User's Guide.
- MPLAB® XC8 C Compiler User's Guide.
- PIC18 Peripheral Library Help Document.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.


**PRÁCTICA 4.3: AUTO ROBOT SIMPLE CONTROLADO CON PULSADORES**

**Introducción.**

Se pueden construir robots que obedezcan comandos, aprovechando las entradas y salidas que poseen los microcontroladores. La manera básica es controlar un par de motores para lograr que el robot se mueva según se le ordene.

**Objetivos.**

- Compilar un programa escrito en Basic, que controle un auto robot.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE

**Módulos**

- Módulo principal.
- Módulo programador.

**Materiales.**

- PIC18F2550 (Reemplazable modificando el programa)

**Procedimiento.**

1. Abrir Proton IDE.
2. Pegar el contenido del programa.
3. Compilar el programa.
4. Cablear desde RA.0 hasta el control del servo 0.
5. Cablear desde RA.1 hasta el control del servo 1.
6. Grabar el programa al microcontrolador.
7. Alimentar la placa principal.
8. Al pulsar los botones del módulo, los motores deben moverse acorde a las instrucciones.

**Programa.**

```
Device = 18F2550
XTAL = 48

'****************************************************************
'declaración de variables y constantes
'****************************************************************
Dim Posicion As Word    " Variable para posición
Dim Dir As Byte         " Variable para sentido de desplazamiento
```

```
ADCON1=%11111111        " Todo PORTA digital
TRISA = 0               " PORTA como salida
TRISB = 1               " PORTB como entrada
DelayMS 150
Cls

Posicion = 1500

'*****************************************************************
'programa principal
'*****************************************************************
Inicio:                 "lazo para determinar cuál de los botones está conectado.
 If PORTB.0 == 0 Then
  GoSub Adelante
 ElseIf PORTB.1 == 0 Then
  GoSub Izquierda
 ElseIf PORTB.2 == 0 Then
  GoSub Derecha
 ElseIf PORTB.3 == 0 Then
  GoSub Atras
 EndIf
GoTo Inicio

'*****************************************************************
'Subrutinas
'*****************************************************************
Derecha:
 Repeat                 " Inicio del lazo
    Servo PORTA.0, Posicion     " Enviar el valor en Posición a servo 0
    DelayMS 20                  " Esperar 20 ms
    Inc Posicion                " Incrementar uno al contador.
 Until Posicion = 1750          " Repetir hasta que contador sea 1750.
Return

Izquierda:
 Repeat                 " Inicio del lazo
    Servo PORTA.1, Posicion     " Enviar el valor en Posición a servo 1
    DelayMS 20                  " Esperar 20 ms
    Inc Posicion                " Incrementar uno al contador.
 Until Posicion = 1750          " Repetir hasta que contador sea 1750.
Return

Adelante:
 Repeat                 " Inicio del lazo
    Servo PORTA.0, Posicion     " Enviar el valor en Posición a servo 0
    Servo PORTA.1, Posicion     " Enviar el valor en Posición a servo 1
    DelayMS 20                  " Esperar 20 ms
    Inc Posicion                " Incrementar uno al contador.
 Until Posicion = 1750          " Repetir hasta que contador sea 1750.
Return

Atras:
 Repeat                 " Inicio del lazo
     Servo PORTA.1, Posicion    " Enviar el valor en Posición a Servo 0
     Servo PORTA.1, Posicion    " Enviar el valor en Posición a Servo 1
     DelayMS 20                 " Esperar 20 ms
     Dec Posicion               " Restar uno al contador.
 Until Posicion = 1250          " Repetir hasta que contador sea 1250.
```

Return

**Conclusiones.**

**Recomendaciones.**

**Reto.**

- Modificar el programa para usarlo con un PIC de la serie 16F.

**Bibliografía.**


- Proton IDE 1.0.4.6 Help Topics.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.

# SECCIÓN 5: SISTEMAS EMBEBIDOS

## PRÁCTICA 5.1: ALARMA BÁSICA CON CLAVE

**Introducción.**

Existe una amplia gama de aplicaciones reales con microcontroladores en sistemas embebidos. Se puede dotar de poder de procesamiento a una alarma electrónica.

**Objetivos.**

- Compilar un programa escrito en Basic, que se desactive con una secuencia específica de caracteres.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE 1.0.4.6

**Módulos**

- Módulo principal.
- Módulo programador.
- Módulo LCD.

**Materiales.**

- PIC16F877A (Reemplazable modificando el programa)
- Teclado matricial.

**Procedimiento.**

1. Abrir Proton IDE.
2. Pegar el contenido del programa.
3. Compilar el programa.
4. Grabar el programa al microcontrolador.
5. Conectar el módulo LCD.
6. Conectar el teclado matricial al puerto B.
7. Alimentar la placa principal.
8. Al digitar una clave incorrecta debe leerse “Incorrecto” en el LCD.
9. Al digitar la clave “1234” se debe leer “Correcto” en el LCD.

**Programa.**

```
'*****************************************************************
'* Name    : UNTITLED.BAS                                        *
'* Author  : Gabriel Santamaría                                  *
'* Notice  : Copyright (c) 2010 GabSan                           *
'*         : All Rights Reserved                                 *
'* Date    : 04/10/2010                                          *
'* Version : 1.0                                                 *
'* Notes   :                                                     *
'*         :                                                     *
'*****************************************************************
Device=16F877A
Xtal=4
ADCON1=7
LCD_Type hitachi " Especificar lcd de 16x2 líneas
Declare Keypad_Port PORTC " Puerto al cual está conectado el Teclado Matricial

DelayMS 50
Cls
TRISA=0 " Configuar puerto A como salida para control de puerta y alarma
TRISB=%00001111
TRISC=%00001111 " Configurar el puerto de tal forma q queden entradas y salidas para manejar el teclado

Dim tecla As Byte
Dim digitar[4] As Byte
Dim clave[4] As Byte
Dim i As Byte
Dim j As Byte
```

```
Dim mantener As Byte
Dim mal As Byte
Dim bien As Byte

clave[0]=1
clave[1]=2
clave[2]=3
clave[3]=4

inicio:
While mantener <> 42
   While i<11
      tecla=InKey

      If tecla==0 Then
         digitar[i]=1
         i=i+1
         Print At 1,i,"*"
      EndIf

      If tecla==0 Then
         digitar[i]=1
         i=i+1
         Print At 1,i,"*"
      EndIf

      If tecla==0 Then
         digitar[i]=1
         i=i+1
         Print At 1,i,"*"
      EndIf

      If tecla==0 Then
         digitar[i]=1
         i=i+1
         Print At 1,i,"*"
      EndIf

      If tecla==0 Then
         digitar[i]=1
         i=i+1
         Print At 1,i,"*"
      EndIf

   Wend

   While 1>=0
      If mantener==35 Then
         i=i-1
         Print At 1,i," "
      EndIf
   Wend
Wend

If i<>4  Then
   Print At 2,1,"incorrecta"
EndIf

For i=0 To 3
   If clave[i]==digitar[i] Then
      GoTo salto
   Else
      GoTo salir
   EndIf
   salto:
Next

Print At 2,1,"correcta"
GoTo inicio

salir:
   j=j+1
   Print 2,1,"incorrecto"
```

```
DelayMS 1000
GoTo inicio
```

**Conclusiones.**

**Recomendaciones.**

**Reto.**

- Modificar el programa para introducir la clave de forma manual al principio del mismo.

**Bibliografía.**


- Proton IDE 1.0.4.6 Help Topics.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.


## PRÁCTICA 5.2: SENSOR DE TEMPERATURA CON RELÉ

**Introducción.**

Las salidas de un microcontrolador, con una interface apropiada de salida; puede utilizarse para controlar virtualmente cualquier proceso, independientemente de la potencia del mismo.

Algo común es poner un relé controlado por el pin. Sin embargo éste debe protegerse contra corrientes inversas que podrían destruir el PIC. Con un transistor y un diodo es suficiente en la mayoría de los casos, ya que la entrada controlaría al transistor y éste al relé. Existen integrados que tienen este circuito en su interior y pueden manejar grandes potencias.

**Objetivos.**

- Compilar un programa escrito en Basic, que active o desactive un relé al llegar a límites de temperatura establecidos por el usuario.
- Grabar en el PIC el archivo hexadecimal generado.

**Recursos.**

- Proton IDE 1.0.4.6

**Módulos**

- Módulo principal.
- Módulo programador.
- Módulo LCD.

**Materiales.**

- PIC16F877A (Reemplazable modificando el programa)
- DS1820.
- Teclado matricial.

**Procedimiento.**

1. Abrir Proton IDE.
2. Pegar el contenido del programa.
3. Compilar el programa.
4. Conectar un DS1820 al puerto A0.
5. Cablear un pulsador a RB0.
6. Cablear un pulsador a RB1.
7. Cablear desde RC6 hasta Tx_Serie.
8. Cablear desde RC7 hasta Rx_Serie.
9. Conectar un teclado matricial al puerto C de la placa.
10. Poner el jumper Serie en posición "ON"
11. Grabar el programa al microcontrolador.
12. Conectar el módulo LCD.
13. Alimentar la placa principal.
14. Se debe visualizar el texto "Inicializando" en el LCD.
15. Introducir temperatura máxima.
16. Introducir temperatura mínima.
17. Conectar módulo relevadores.
18. Calentar el DS1820 hasta la temperatura máxima. El relé debe abrirse.
19. Cuando la temperatura alcance el mínimo el relé debe cerrarse.

**Programa.**

```
'*****************************************************************
'*  Name    : UNTITLED.BAS                                       *
'*  Author  : Gabriel Santamaría                                 *
'*  Notice  : Copyright (c) 2010 GabSan                            *
'*          : All Rights Reserved                                *
'*  Date    : 23/10/2010                                         *
'*  Version : 1.0                                                *
'*  Notes   :                                                    *
'*          :                                                    *
'*****************************************************************
;Org 0500

Device=16F877A
Xtal=4
ADCON1=7
option reg.7=1

TRISB=%00001111

DelayUS 50
LCD_DTPin PORTD.4 ;empiezo las líneas de datos de d.4 a d.7
LCD_ENPin PORTA.5 ;señal enable en c.7 q es la siguiente en el pic
LCD_RWPin PORTA.4
```

```
LCD_RSPin PORTA.3
Cls

Dim tecla As Byte           'se llena con el valor q intruducimos en el teclado
Dim tempmax As Byte
Dim tempmin As Byte
Dim aceptar As Byte
Dim i As Byte               'posición del vector para unidades decenas centenas
Dim llenado [3] As Byte     'vector de llenado
Symbol DQ = PORTA.0          ' Place the DS1820 on bit-1 of PORTA
Dim Temp As Float           ' Holds the temperature value
Dim Templcd As Float              ' Holds the counts remaining value

'*****************************************************************************
'Presentación inicial
'*****************************************************************************
'$FE, 1         Clear display
'$FE, 2         Return home (beginning of first line)
'$FE, $0C       Cursor off
'$FE, $0E       Underline cursor on
'$FE, $0F       Blinking cursor on
'$FE, $10       Move cursor left one position
'$FE, $14       Move cursor right one position
'$FE, $C0       Move cursor to beginning of second line
'$FE, $94       Move cursor to beginning of third line
'$FE, $D4       Move cursor to beginning of fourth line


DelayMS 50      'retatdo para que se energize lcd
Print At 1,1,"Iniciando"
DelayMS 500
Cls
DelayMS 250


'*****************************************************************************
'Rutina principal
'*****************************************************************************

inicio:

   tecla=16              'inicializa tecla a su estado pasivo
   Print At 1,1,"Temp Max:"
   Print $FE, $C0  'mover cursor a la segunda linea
   GoSub llenar          'subrutina "llenar" llena un vector de tres posiciones (temp max sensor=125ºC)
   tempmax=aceptar       'pone en tempmax el valor que va a comparar
   DelayMS 250

   Cls
   Print At 1,1,"Temp Min:"
   Print $FE, $C0  'mover cursor a la segunda linea
   GoSub llenar          'llena vector de "tres" (puedo terminarlo antes)  posiciones y devuelve la variable correspondiente
   tempmin=aceptar

   DelayMS 250

   GoSub bug1         'controla q las variables introducidas no sean iguales
   GoSub bug2         'controla q tempmax no sea menor q tempmin

   Cls                'limpiar el lcd para empezxar a mostrar valores medidos
   GoTo medir            'con los valores correspondientes en las variables, mido temperatura


'*****************************************************************************
'Subrutinas de llenado:
'*****************************************************************************

llenar:          'pone en vector llenado[i] el valor leído del teclado
   While i<3      'mientras tenga espacio, sigo llenando el vector
      GoSub leer  'me devuelve el valor pulsado, almacenado en la variable tecla
      If tecla=="a" Then   'estoy aceptendo el valor
         GoSub transformar
         GoSub encerar    'encero para que no haya conflictos en el nuevo proceso
```

```
        Return
      Else If tecla=="c" Then 'invoco la subrutina de borrar si oprimo "c"
        GoSub borrar_caracter
      Else
        llenado[i]=tecla    'lleno el vector
        tecla=16            'pongo tecla a su estado pasivo
        Print At 2,i+1,@llenado[i]
        i=i+1
      EndIf
    Wend

    Print At 2,10,"Ok?" 'mensaje preguntando si va a introducir el valor
    GoSub leer
    If tecla=="a" Then
      GoSub encerar
      Return
    ElseIf tecla=="b" Then
      GoSub reiniciar
    EndIf
GoSub encerar
Return              'devuelvo el vector lleno a la rutina principal


leer:   ;escanea el teclado hasta que se presione una tecla (si no se presiona devuelve 16)
  While tecla==16 'mientras no presione la tecla me quedo en el lazo
    tecla=InKey
    DelayMS 170 'antirrebote
    tecla=LookUpL tecla,[4,3,2,1,8,7,6,5,"b","a",0,9,"f","e","d","c",16];para adquirir datos de acuerdo al tkmatricial
  Wend
Return


borrar_caracter:            'si recibe una "b" en el teclado, borra
    Print At 2,i, " " 'imprimir espacio vacío
    Print $FE, $10      'regresar una posicion
    DelayMS 150
    i=i-1
    tecla=16
    GoSub leer
Return


encerar:    'encera las variables para nuevo proceso
    tecla=16
    i=0
Return


transformar:                'transforma las variables en cada posición de vector llenado en el valor correspondiente decimal
    If i=1 Then
      aceptar=llenado[0]
    ElseIf i=2 Then
      aceptar=10*llenado[0]+llenado[1]
    Else
      aceptar=100*llenado[0]+10*llenado[1]+llenado[2]
    EndIf
Return

'*****************************************************************************
'Subrutinas de control de datos:
'*****************************************************************************

bug1:   'tempmax=tempmin
    If tempmax==tempmin Then
      Cls
      Print "ERROR! tmax=tmin!"
      Print $FE, $C0  'mover cursor a la segunda linea
      Print "presione tecla"
      GoSub leer
      GoSub reiniciar
    EndIf
Return
```

```
bug2:   'tempmax<tempmin
   If tempmax<tempmin Then
      Cls
      Print "ERROR! tmin>tmax!"
      Print $FE, $C0  'mover cursor a la segunda linea
      Print "presione tecla"
      GoSub leer
      GoSub reiniciar
   EndIf
Return

'*************************************************************************
'Subrutinas de control de temperatura:
'*************************************************************************

medir:
   OWrite DQ,1,[$CC,$44] 'Ordeno al sensor empezar convertir temperatura
   High DQ               ' pongo a DQ en alto mientras escribo datos (termino conversión)
   DelayMS 750           'tiempo para q el sensor convierta y se estabilice
   OWrite DQ,1,[$CC,$BE] 'requiero información en memiria scratchpad
   ORead DQ,0,[Temp.LowByte,Temp.HighByte] 'divido la info binaria entregada por el sensor
   ;Templcd=Temp/16
   GoSub complemento2
   Print At 1,1,"Tmax:",@tempmax
   Print At 1,10,"Tmin:",@tempmin
   Print At 2,1,"T actual: ",Dec2 Templcd," C"
   GoSub comunicar
   GoSub control
GoTo medir

complemento2:
   Cls
   Templcd=Temp.HighByte
   Print @Templcd.HighByte
   Stop
Return

Symbol calefactor PORTC.0 'controla relé calefactor
Symbol sobre PORTC.1 'controla sobre temperatura deseada ST
Symbol bien PORTC.2 'controla temperatura en rango
Symbol baja PORTC.3 'controla baja temparatura BT

control:

   If Templcd>tempmin And Templcd<tempmax Then
      GoSub ok
   EndIf

   If Templcd>tempmax Then
      GoSub st
   EndIf

   If Templcd<tempmin Then
      GoSub bt
   EndIf
Return

comunicar:
    HRSOut Dec2 Templcd,13
    DelayUS 50
Return

ok:
   Low calefactor
   Low sobre
   High bien
   Low baja
Return

bt:
   High calefactor
   Low sobre
   Low bien
```

```
   High baja
Return

st:
   Low calefactor
   High sobre
   Low bien
   Low baja
Return

parar:
Print "DETENIDO"
Print $FE, $C0  'mover a la segunda línea
Print "Pulse A para reiniciar"

Return

reiniciar:  'reinicia todo el proceso, encerando las variables
   i=0
   tempmax=0
   tempmin=0
   aceptar=0
   Cls
GoTo inicio
```

## Conclusiones.

## Recomendaciones.

## Reto.

- Modificar el programa, usando la interrupción de cambio de estado en RB0 para realizar una parada de emergencia de la implementación.

**Bibliografía.**


- Proton IDE 1.0.4.6 Help Topics.
- PIC16F87xA Datasheet.
- PIC18F2455/2550/4455/4550 Data sheet.

# GLOSARIO

**EDA** *Electronic Design Automation*, Diseño Electrónico Automatizado. Es un conjunto de programas informáticos que asisten en el diseño de circuitos electrónicos.

**GNU** *Gnu is not Unix*; es un acrónimo recursivo (Las iniciales corresponden al significado) de Gnu no es Unix, un proyecto de software libre, que intenta poner a disposición del público en general todo el desarrollo de un Sistema Operativo totalmente abierto.

**GPL** *GNU General Public License*, Licencia Pública General GNU, es un tipo de licencia que especifica que todo desarrollo bajo ella será libre, redistribuible y el autor será citado en todo trabajo que se base en dicho trabajo.

**Linux** Es un núcleo de Sistema Operativo Libre. Un núcleo asigna los recursos físicos de la máquina a los programas que lo requieran.

**GNU/Linux** Es la combinación de los desarrollos GNU y Linux para crear Sistemas estables.

***Software* Libre** Es aquel *Software* cuyo código fuente está disponible para modificación.

***Software* Gratuito** Es aquel *Software* que está disponible gratuitamente. No necesariamente es libre.

***mil*** Una milésima de pulgada.

**PCB** *Printed Circuit Board*, Placa de Circuito Impreso. Es una placa que tiene pistas conductoras y elementos electrónicos interconectados de tal manera que cumplen una determinada función.