ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DE MÓDULOS DIDÁCTICOS BÁSICOS BASADOS EN EL MICROCONTROLADOR ATMEGA164 PARA EL LABORATORIO DE MICROPROCESADORES DE LA ESFOT

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

JOEL SANTIAGO CARVAJAL YANNUZZELLI

joel.carvajal@epn.edu.ec

DIRECTOR: ING. VIVIANA PARRAGA VILLAMAR

viviana.parragav@epn.edu.ec

CODIRECTORA: MSC. MÓNICA VINUEZA RHOR

monica.vinueza@epn.edu.ec

Quito, Noviembre 2018

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Joel Santiago Carvajal Yannuzzelli, bajo nuestra supervisión.

_____

Ing. Viviana Parraga Villamar
DIRECTORA DEL PROYECTO

_____

Ing. Mónica Vinueza Rhor
CODIRECTORA DEL PROYECTO

DECLARACIÓN

Yo Joel Santiago Carvajal Yannuzzelli declaró bajo juramento que el trabajo aquí descrito es de mi autoría, que no ha sido previamente presentado para ningún grado o certificación profesional y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración, cedo mis derechos de propiedad intelectual correspondientes a este trabajo a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual por su Reglamento y por la normatividad Institucional vigente.

_____

JOEL SANTIAGO CARVAJAL YANNUZZELLI

DEDICATORIA

Dedico este trabajo a mis padres, ellos siempre me han estado apoyando a su manera, no dejaron que me desmorone cuando las cosas se ponían complicadas y siempre me decían que sigan intentando, que ellos estarán conmigo en todo momento.

A mi abuelita, que siempre me ha estado mandando sus ánimos a la distancia, que siempre me ha dicho que sin importar cuanto tiempo me tome siga adelante.

A mis amigos más cercanos que a su manera me acompañaron, e influyeron en mí para continuar en este difícil camino y que hicieron que disfrute cada momento.

También me lo dedico a mí mismo, como prueba de que puedo cumplir y superar un reto a pesar de todas las adversidades que se me presentaron.

AGRADECIMIENTO

Agradezco a toda mi familia, que a pesar de cualquier problema sigue manteniéndose y son por los cuales he seguido adelante, aun cuando hubo momentos en los que se pudo abandonar todo.

También agradezco a mis profesores y compañeros que compartieron sus conocimientos y lecciones de vida que me ayudaron a crecer como persona.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La materia de Microprocesadores tiene como objetivo que los estudiantes aprendan a diseñar sistemas de baja complejidad basados en un microcontrolador de estudio. El reconocer los elementos básicos y su conexión en un sistema que está basado en un microcontrolador, la arquitectura de los microcontroladores comerciales usando la documentación técnica del fabricante. Además, de conocer la estructura y funcionamiento interno de los microcontroladores.

Entonces, el presente proyecto tiene como finalidad renovar el laboratorio de Microprocesadores, mediante la implementación de placas didácticas básicas con un microcontrolador ATmega164PA como elemento principal y algunos de los elementos electrónicos más elementales al momento de empezar con el estudio de la carrera de electrónica como son: resistencias, diodos LED, *displays* de 7 segmentos, pulsadores, entre otros.

En consecuencia, se revisaron los temas y elementos electrónicos, así como la hoja de datos técnicos del microcontrolador ATmega164PA, con el objetivo de definir cómo conectar todos estos dispositivos para obtener módulos didácticos a ser utilizados por los estudiantes.

Inicialmente se realizó el diseño de un módulo, colocando elementos que brinden un manejo de las placas didácticas mucho más versátil. Definidos los diseños tanto esquemáticos como físicos, se procedió a la construcción de las placas y pruebas de funcionamiento a través de un programador USB llamado USBasp.

De acuerdo a los temas seleccionados se realizaron hojas guías de laboratorio para instructores y estudiantes que sirvan como referencia del uso que puede darse de los módulos didácticos y además, demostrar su funcionamiento.

Finalmente, se describe lo más relevante de las pruebas de funcionamiento y sus complementos. Así mismo un manual de usuario, en este se detallan el manejo del módulo didáctico y los elementos que lo conforman.

Palabras clave: placa didáctica, microcontrolador ATmega164PA, programador USBasp, hojas guías laboratorio, manual de usuario.

ABSTRACT

The subject of Microprocessors has as objective that students learn to design los complexity systems based on a study microcontroller. Recognizing the basic elements and their connections in a system that is based on a microcontroller, the architecture of a commercial microcontrollers using the manufacturer's technical documentation. In addition, to know the structure and internal functioning of the microcontrollers.

So, the present project has the purpose of renewing the microprocessor laboratory, through the implementation of basic didactic plates with an Atmega164PA microcontroller as the main element and some of the most elementary electronic elements at the moment of beginning with the study of the electronic career as they are: resistors, LEDs, 7 segment displays, pushbuttons, among others.

Consequently, the topics and electronic elements were reviewed, as well as the technical data sheet of the of the Atmega164PA microcontroller, with the objective of defining how to connect all these devices to obtain teaching modules to be used by students.

Initially the design of a module was made, placing elements that provide a much more versatile handling of the teaching boards. Defined both schematic and physical designs, proceeded to the construction of the plates and tests of operation through a USB programmer called USBasp.

According to the selected topics, laboratory guides sheets were made for instructors and students that serve as reference of the use that can be given of the didactic modules and also, demonstrate its working.

Finally, describe the most relevant aspect of the performance tests and their complements. Likewise, a user manual, this article details the use of the didactic module and the elements that make up.

Keywords: *didactic plate, Atmega164PA microcontroller, USBasp programmer, laboratory guide sheets, user manual.*

1. INTRODUCCIÓN

El proyecto integrador consiste en la implementación de 10 módulos didácticos básicos para el laboratorio de Microprocesadores de la ESFOT. Con estos módulos se quiere lograr un avance en cuanto al uso de nuevas tecnologías en microcontroladores y lenguajes de programación. Muchos proyectos en electrónica actualmente son basados en dispositivos que pueden abarcar mucha cantidad de información en tiempos muy cortos como: ATmega, Arduino, Raspberry, etc.

Para realizar estos módulos didácticos básicos, lo primero fue analizar los temas que se aplicarán, para seleccionar los elementos electrónicos que lo conforman. Lo siguiente fue realizar el diseño que se hizo utilizando el programa Proteus. Para la construcción de estos se aplicó los conocimientos en técnicas de circuitos impresos y soldadura. Finalmente se realizaron las pruebas de funcionamiento para posteriormente aplicar los ejemplos de ejercicios y diseñar las diferentes prácticas de laboratorio.

Los módulos didácticos básicos están hechos en placas baquelitas, el elemento principal es un microcontrolador ATmega164PA. Además, vienen incorporados varios elementos electrónicos básicos que se utilizan como punto de partida en el ensamblaje de circuitos electrónicos como: resistencias, leds, pulsadores, interruptores, *displays* de 7 segmentos y un teclado matricial de 3x4.

Además de los elementos electrónicos mencionados, los módulos didácticos básicos cuentan con unos elementos denominados regletas. En estas regletas se pueden conectar elementos electrónicos adicionales y también algunos de los elementos electrónicos están conectados a través de ellas como: *displays* de 7 segmentos, leds y el microcontrolador ATmega164PA, para cuando dado el caso estos elementos dejen de funcionar estos puedan ser reemplazados.

Para el funcionamiento de estos módulos didácticos básicos se cuenta con un programador USB, llamado USBasp. Este programador no solo grabará los códigos de programa en el microcontrolador ATmega, sino que además sirve como fuente de alimentación. El programador USB es compatible con cualquier sistema *Windows*, lo cual permite que los módulos puedan ser utilizados en cualquier máquina simplemente instalando el programa indicado y algún *driver* en caso de ser necesario.

En estos módulos didácticos básicos se podrán aplicar varios de los temas de estudio que se abarcan al empezar la materia de Microprocesadores, entre ellos están: manejo

de puertos, señales PWM, interrupciones y temporizaciones. Para dar un ejemplo de la aplicación de estos temas se realizaron modelos de hojas guías para prácticas de laboratorio tanto para los estudiantes e instructores.

Las hojas guías para estudiantes contienen los ejercicios que ellos deben realizar, mientras que las hojas guías para los instructores contienen: configuraciones de los módulos didácticos básicos, simulaciones, códigos de programa y alguna recomendación adicional que surgió al realizar dicha práctica.

1.1 Marco Teórico

ATmega164PA

Es un microcontrolador AVR CMOS de 8 *bits*, alto rendimiento y baja potencia. Permite ejecutar poderosas instrucciones en un solo ciclo de reloj, el microcontrolador logra rendimientos cercanos a 1 MIPS (millones de instrucciones por segundo) por MHz. De esta forma facilita al diseñador para optimizar el dispositivo de consumo de energía y la velocidad de procesamiento. En la figura 1.1 se muestra el ATmega164PA con su distribución de pines y la descripción de cada uno.

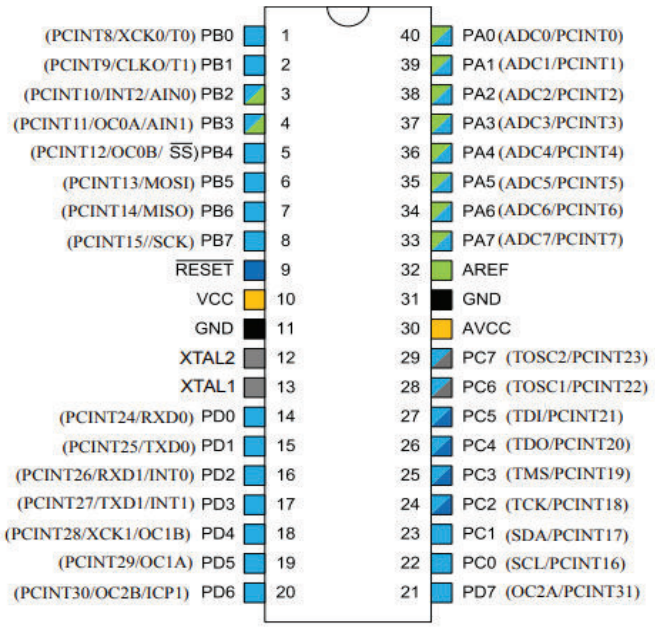


Figura 1.1 Distribución pines ATmega164PA [1]

El núcleo ATmel AVR combina un conjunto de registros de 32 instrucciones de trabajo de propósito general. Cada uno de estos registros está conectado de manera directa a la unidad aritmética lógica (ALU). Esto permite dos registros independientes para acceder a una sola instrucción ejecutada en un ciclo de reloj. Toda esta arquitectura logra que los

rendimientos sean hasta diez veces más rápidos que los microcontroladores CISC (set de instrucciones de computador complejas) convencionales.

En la siguiente tabla se muestra las características más relevantes del microcontrolador ATmega164PA de su hoja de datos técnicos [1].

Tabla 1.1 Parámetros técnicos del ATmega164PA [1]

Característica	**Valor**
Número de Pines	40
Bytes de Memoria Flash	16K
Bytes de SRAM	1K
Bytes de EEPROM	512
Velocidad de la CPU (MIPS)	20
ADC	10 - *bits* 15ksps
Canales ADC	8
Comparadores analógicos	1
Temporizadores / contadores	2 x 8 *bits*, 1 x 16 *bits*
Canales PWM	6
Rango de tensión de funcionamiento (V)	1.8 a 5.5
Rango de temperatura (ºC)	-40 a 85

El ATmega164PA es un poderoso microcontrolador que proporciona una solución altamente flexible y rentable para muchas aplicaciones de control integradas. Es compatible con un conjunto completo de herramientas de desarrollo y sistemas que incluyen: compiladores C, ensambladores de macros, depuradores / simuladores de programas, emuladores en circuito y kits de evaluación [2] [3].

Manejo de puertos E/S

Los microcontroladores cuentan con un conjunto de pines que son utilizados como entrada y salida de datos o señales digitales. Este conjunto de pines se les denomina “puerto”. Los puertos del microcontrolador son el punto de comunicación, a través de ellos se pueden efectuar procesos de control electrónico sobre dispositivos de potencia, instrumentación, etc.

Estos puertos pueden presentar sus datos a través de elementos como: leds, *displays* de 7 segmentos, etc. Además, los puertos pueden receptar datos del exterior a través de elementos como: interruptores, pulsadores, teclado, etc.

El funcionamiento del microcontrolador está controlado a través de sus registros y los puertos no son la excepción, por esta razón un puerto no puede tener más de 8 pines: 1 pin por cada bit de un registro, pero si puede tener menos de 8 pines [4]. Los registros son posiciones de memoria que pueden almacenar un dato, estos registros se identifican mediante un número, el cual se denomina dirección de memoria y generalmente ese número es expresado en formato hexadecimal.

Para utilizar un puerto, este primero debe ser configurado ya sea de entrada o salida. Así mismo cada pin de un puerto puede ser configurado de manera independiente del resto de pines del mismo puerto. El microcontrolador Atmega164PA consta de 4 puertos, cada uno de estos formado por 8 pines; estos puertos denominados: puerto A (33 – 40), puerto B (1 – 8), puerto C (22 – 29) y puerto D (14 – 21), como se observa en la figura 1.1. Cada microcontrolador puede tener varios de estos puertos dependiendo de su modelo.

Señales PWM

La modulación por ancho de pulso (PWM) de una señal es una técnica que logra producir el efecto de una señal analógica sobre una carga, a partir de la variación de la frecuencia y ciclo de trabajo de una señal digital. Estas señales son utilizadas comúnmente en el control de aplicaciones, por ejemplo: control de motores de corriente continua, sistemas hidráulicos, y algunos otros dispositivos mecánicos [5].

Las señales PWM pueden ser de tipo cuadrada o sinusoidal como se observa en la figura 1.2 en la cual el ancho relativo cambia con respecto al periodo de la misma. Para emular una señal PWM dentro del microcontrolador es necesario configurar ciertos registros. De tal modo que el valor de la señal sea en proporción al voltaje que maneja el microcontrolador, por ejemplo: si el valor del voltaje está en 0V, el tamaño de la señal PWM puede ser la mínima, caso contrario si el valor de voltaje es 5V, el tamaño de la señal puede ser el máximo ancho de pulso. Como se puede ver en la tabla 1.1, el microcontrolador ATmega164PA cuenta con 6 canales PWM, los cuales son los pines con la descripción OC#X, ejemplo: OC0A, OC1B, etc. [6].

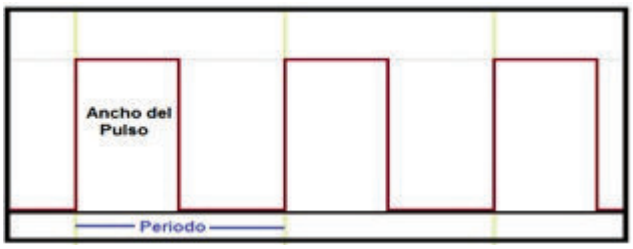


Figura 1.2 Ejemplo de una señal PWM [5]

Por tanto, PWM es una técnica que consiste en variar el ancho de pulso de una señal de voltaje, con el objetivo de controlar la cantidad de potencia administrada a los componentes electrónicos conectados.

Convertidor Analógico – Digital (ADC)

El convertidor analógico – digital o ADC, permite tomar y medir señales analógicas en forma digital. Se puede decir que realiza un proceso contrario al PWM, ya que este transforma una señal digital en una analógica. Los ADC convierten un voltaje analógico externo en un número, con el que se puede operar. En la figura 1.3 se puede observar como internamente la señal de entrada analógica es transformada en una señal digital [7].

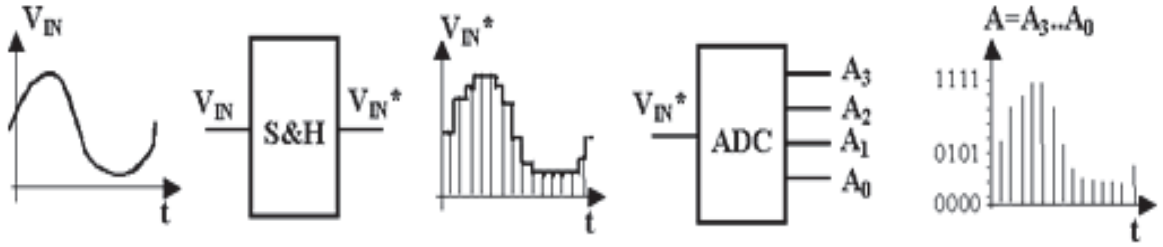


Figura 1.3 Transformación de señal analógica a señal digital [7]

Como se menciona en la tabla 1.1, el microcontrolador Atmega164PA tiene 8 ADC y como se muestra en la figura 1.1, se ubican en el puerto A de este. Cada ADC maneja como entrada analógica un número binario de 10 *bits*, es decir los valores que pueden llegar a operar se encuentra en un rango de 0 a 1023. Para utilizar estos ADC es necesario configurar los pines como entradas para que manejan estos valores.

Cuando el microcontrolador maneja un ADC, mide el voltaje que ingresa en un pin y lo convierte en un número. Pero para que éstos funcionen necesitan una tensión o voltaje de referencia para poder trabajar de manera adecuada. Este voltaje de referencia (VREF) puede ser el voltaje con el que normalmente trabaja el microcontrolador, aunque se pueden elegir otros voltajes de referencia. El voltaje de referencia debe ser aplicado a dos pines del microcontrolador y estos son: **AREF** y **AVCC**. El pin **AREF**, es la entrada analógica al ADC, mientras que el pin **AVCC**, provee energía al ADC.

La frecuencia de trabajo del microcontrolador puede ser un factor importante para el trabajo de los ADC, ya que con una frecuencia adecuada se puede obtener una conversión más confiable. Con una frecuencia de trabajo alta la conversión será más rápida pero no muy confiable, caso contrario si la frecuencia de trabajo es baja la conversión será más lenta pero mucho más confiable [8].

Temporizadores

El microcontrolador maneja los llamados *timers* los cuales pueden trabajar como temporizador o contador. Como se muestra en los datos técnicos de la tabla 1.1, el microcontrolador Atmega164PA cuenta con 3 *timers*, de los cuales dos son de 8 *bits* y uno de 16 *bits*.

Los *timers* son módulos que trabajan en paralelo con el procesador, permitiendo que las operaciones de temporización y conteo se puedan llevar a cabo de manera eficiente, mientras el procesador se ocupa de otras tareas. A pesar de las diferencias operativas que pueden tener entre sí los *timers*, la configuración y control son muy similares en todos los casos. En la figura 1.4 se puede observar un ejemplo de los registros internos del *timer1* de un microcontrolador ATmega [9].

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Figura 1.4 Registros internos del timer1 [9]

Si se configura los registros del *timer* para que trabaje como contador, el conteo que realiza es dependiendo el *timer*, si es de 8 *bits* puede contar desde 0 hasta 255, mientras que si es de 16 *bits* puede contar desde 0 hasta 65536. Para empezar a trabajar como contador el temporizador del *timer* se sincroniza con el reloj externo del microcontrolador y se le denomina contador síncrono.

Para usar los *timers* como temporizadores, se debe configurar los mismos registros, pero con diferentes valores. Al usarlo de este modo, la cuenta del temporizador irá aumentando con cada ciclo de reloj generado por el oscilador interno. La cuenta del temporizador será dependiendo qué *timer* se está utilizando y del pre-escalador utilizado (un divisor de frecuencia, puede tener valores de: 1, 8, 32, 64, 128, 256 y 1024). Si por ejemplo se usa una frecuencia de trabajo de 1MHz, la cuenta aumentará en una unidad cada microsegundo [10].

Una de las aplicaciones de los *timers* es usarlo como contador para realizar tareas simultáneas dentro del microcontrolador. Mientras el ATmega realiza un cierto proceso, el *timer* realiza un conteo configurado para que al momento que finalice interrumpa el proceso principal y realice otro diferente.

Interrupciones

Las interrupciones son extremadamente útiles ya que permiten que el microcontrolador esté atento a un evento en particular, el cual detiene su ejecución actual para ejecutar lo que viene a ser una "interrupción de la rutina de servicio". Las interrupciones son eventos que hacen que el programa principal se detenga y realice otros procesos, mientras estas interrupciones no se produzcan se realizarán únicamente las tareas del programa principal.

Las interrupciones en los microcontroladores se pueden generar cuando se configuran para este propósito. Por ejemplo: cambios de estados lógicos, un proceso en ejecución se detenga, etc. Estas interrupciones pueden ser generadas tanto de forma interna como de forma externa. Las interrupciones externas se pueden producir a través de: pulsadores, interruptores, teclados, etc. Los pines más relevantes para producir una interrupción externa son: **INT0, INT1** e **INT2**. Dependiendo el modelo del microcontrolador ATmega pueden tener más o menos pines **INT**.

Si se observa la figura 1.1, los pines **INT0** e **INT1** se encuentran en el puerto D (PD2 y PD3 respectivamente), mientras que el **INT2** se encuentra en el puerto B (PB2). Cada pin **INT** trabaja de manera independiente, pero realizan la misma tarea, generar una interrupción externa al microcontrolador. En la figura 1.5 se puede observar una simple conexión para usar uno de estos pines **INT** para generar una interrupción externa.

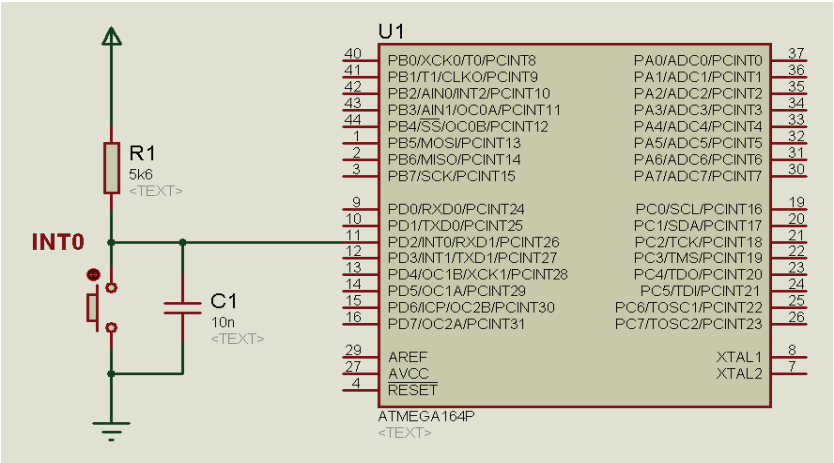


Figura1.5 Esquema de conexión para una interrupción externa [11]

No únicamente los pines **INT** se pueden usar para producir una interrupción externa, el resto de pines con la descripción **PCINT** son capaces de producirlas. La diferencia está en que los pines **INT**, son pines ya listos para producir una interrupción externa, mientras que los **PCINT**, deben ser configurados y la interrupción externa se produce cuando hay un cambio de estado lógico en sus pines [11].

Por otra parte, las interrupciones internas se pueden realizar mediante el uso de los *timers*, como ya se mencionó en el punto anterior, pueden ser usados como contadores, temporizadores o incluso como comparadores.

2. METODOLOGÍA

La selección de los temas aplicados en los módulos didácticos básicos son: manejo de puertos entrada – salida digital y analógica, señales PWM, interrupciones y temporizadores. Los elementos que son útiles para abarcar estos temas son: interruptores, pulsadores, leds, teclado matricial, *display* de 7 segmentos y resistencias. Todos estos conectados al microcontrolador ATmega164PA. Además, el programador USB facilita el desarrollo de las prácticas de laboratorio ya que una vez que el programador cargue el programa, este actúa como fuente de alimentación.

El diseño del prototipo del módulo didáctico básico se realizó primero de manera lógica, conectando los diferentes elementos y considerando las especificaciones técnicas del microcontrolador ATmega164PA. Una vez que todos los dispositivos electrónicos estuvieron conectados entre sí, el diseño lógico se exportó a la plataforma donde se realiza el diagrama de pistas, donde los elementos eran colocados en la baquelita del módulo didáctico básico, ahora considerando detalles como: dimensiones reales, separación entre pines o terminales y tamaño de los elementos.

La construcción de los módulos didácticos básicos se hizo en placas baquelita, con la mayoría de los elementos soldados a esta. Elementos como el microcontrolador ATmega164PA, programador USB, *display* de 7 segmentos, leds, se conectan al módulo a través de unas regletas para que puedan ser cambiados en caso de mal funcionamiento. Además, la arquitectura del módulo didáctico básico cuenta con más regletas, las cuales sirven para acoplar nuevas conexiones con otros elementos y además de conexiones para alimentación.

Las prácticas de laboratorio para desarrollar los diferentes temas se plantearon en hojas guías tanto para los estudiantes como para los instructores. La información que tienen

estas hojas guías para estudiantes es: cuestionario sobre los puntos más relevantes y los ejercicios a desarrollar, dependiendo el tema abarcado. Las hojas guías para los instructores contienen: configuraciones de *software* y de *hardware*, diagramas de flujo, simulaciones de los circuitos, ejemplos de los códigos de programación y alguna recomendación adicional sobre algún error producido.

En las pruebas se evaluó el correcto funcionamiento de los diferentes elementos que conforman el modulo didáctico básico. Primero se probó que exista comunicación entre el programador USB y el modulo didáctico básico, con la conexión producida el programador USB podía leer, borrar y cargar programas fuentes en el microcontrolador. Los resultados esperados fueron visualizaciones a través de los leds, *display,* ingreso de datos a través de los pulsadores, interruptores y el teclado matricial. Finalmente se pudo observar el acople de elementos electrónicos al módulo didáctico básico a través de las regletas.

3. ANÁLISIS Y RESULTADOS

3.1 Diseño lógico del módulo didáctico básico

Para el diseño de la placa de circuito impreso se utilizó el programa Proteus, específicamente su herramienta ISIS (*Intelligent Schematic Input System*). Esta plataforma es ideal para una rápida realización de complejos diseños de esquemas electrónicos destinados para pruebas de simulación y pruebas para construcción de equipos electrónicos.

Como se mencionó antes con los temas que se aplicarán en los módulos didácticos básicos se determinaron que los elementos que lo conforman son: microcontrolador ATmega164, programador USBasp, 4 pulsadores, 1 *dipswitch* de 4, 1 banco de 10 leds, 1 arreglo de 4 *displays* de 7 segmentos, 1 teclado matricial 3x4, resistencias de 330Ω, 5.6KΩ, 10KΩ y regletas tanto para conexión de elementos y conexiones de elementos extra.

Siguiendo la distribución de pines del ATmega164PA (fig. 1.1) este microcontrolador cuenta con 4 puertos A, B, C y D. En el puerto A están todos los canales analógicos. En el puerto B se encuentran algunos de los pines que sirven para programar el microcontrolador. En el puerto C la mayoría de sus pines son para procesos más avanzados. En el puerto D se encuentran algunos pines para generar ondas PWM.

En el anexo C, el cual es un manual de usuario se encuentra más detallado los elementos electrónicos y el puerto del microcontrolador al cual está cada uno conectado, pero en resumen estos se encuentran distribuidos como se muestra en la siguiente figura:

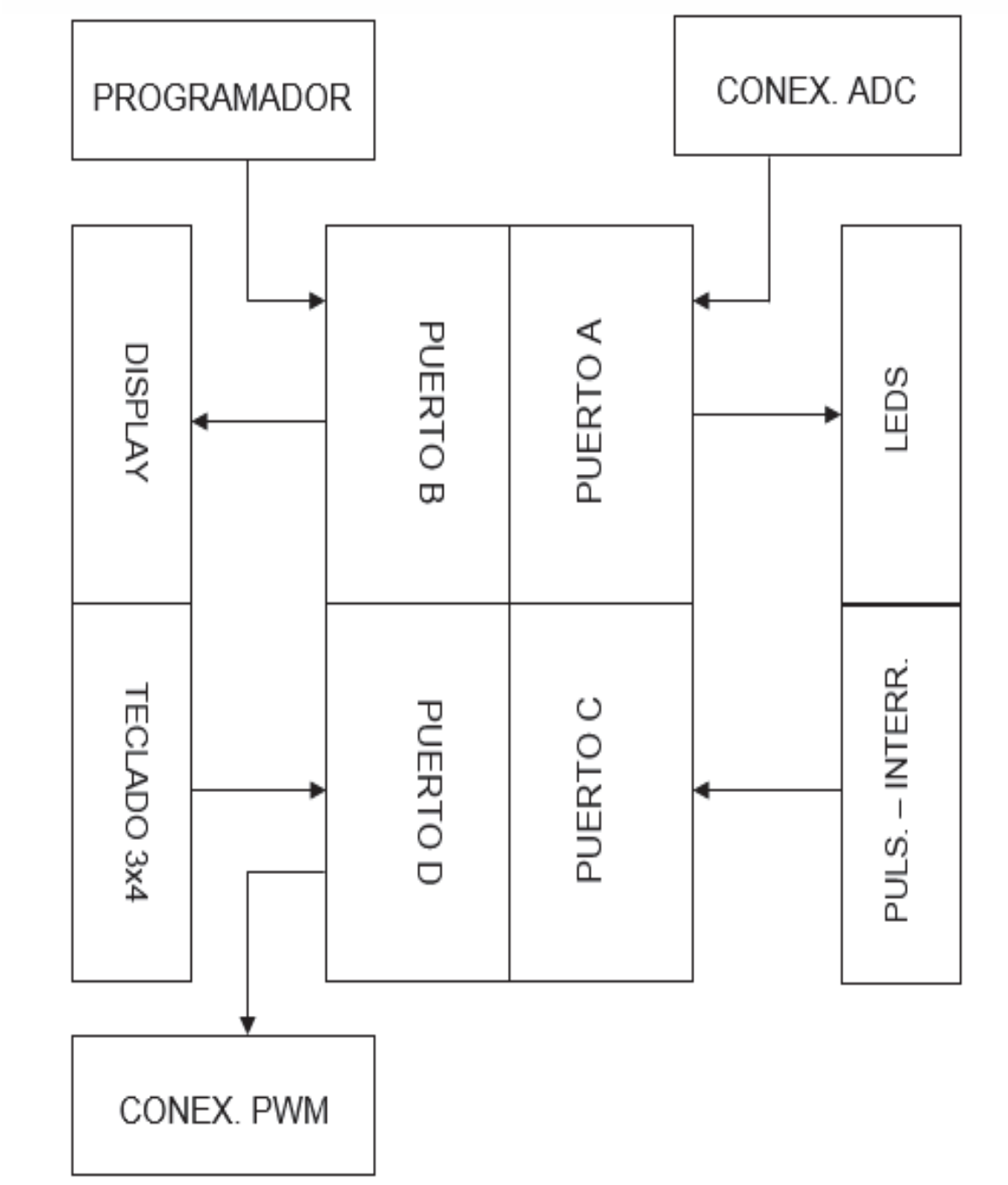


Figura 3.1 Etapas del Módulo Didáctico Básico

- **Puerto A:** banco de 10 leds y una regleta de 8 para usar los canales analógicos.
- **Puerto B:** arreglo de *display* de 7 segmentos y regleta conexión a programador.
- **Puerto C:** 4 pulsadores y 1 *dipswitch* de 4 interruptores.
- **Puerto D:** teclado matricial 3x4 y una regleta de 8 para conexión a canales PWM.

Además, se colocan otras regletas aparte 1 regleta de 8 para conexión directa a VCC, 1 regleta de 8 para conexión directa a GND, 1 regleta de 4 para pines 10 - 13 (VCC, GND, Xtal2, Xtal1) y 1 regleta de 3 para pines 30 – 32 (AVCC, GND, AREF). Finalmente se añadió un circuito adicional para resetear el módulo didáctico básico.

Creación de los elementos electrónicos en ISIS

En diferentes versiones de Proteus se puede dar el caso que el(los) componente(s) que se necesita no se dispongan, o que existan pero no tenga su encapsulado o representación real. En la versión de Proteus utilizada se dio el caso de que existía una representación lógica del ATmega164P, pero en una versión antigua, la cual contaba con alrededor de 44 pines y la utilizada en el módulo didáctico básico solo cuenta con 40 pines en total [12].

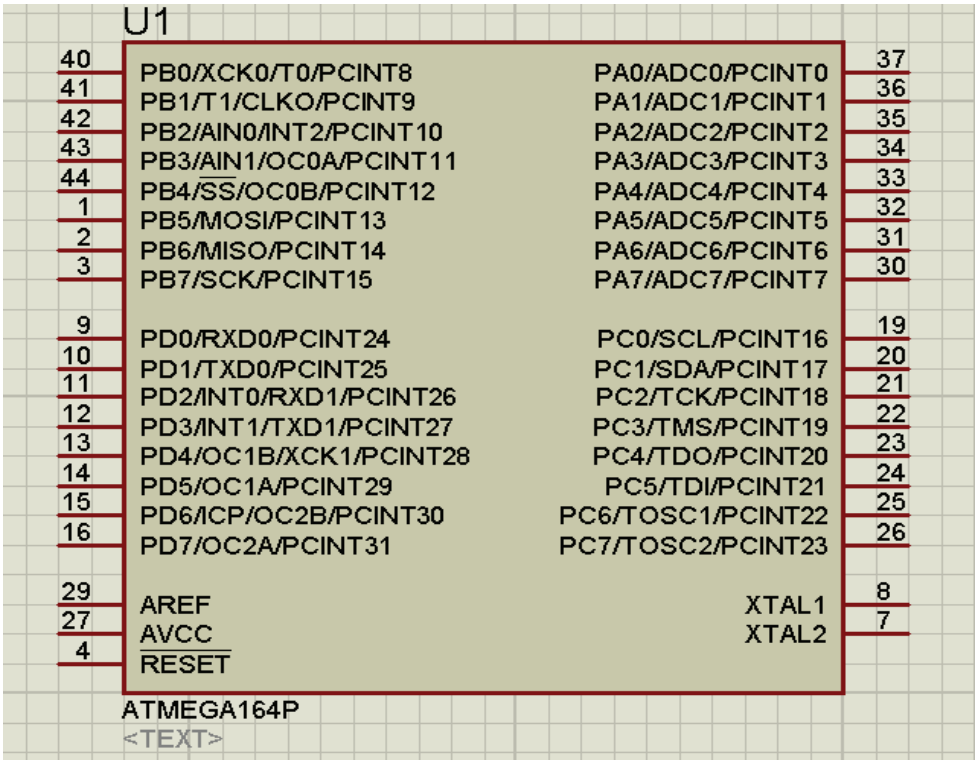


Figura 3.2 ATmega164P versión anterior [12]

La figura 3.2 es la representación lógica del ATmega164P en la versión de Proteus7, lo siguiente fue crear una nueva representación del microcontrolador para el diseño, mediante las herramientas de diseño gráfico del Proteus se realizó el nuevo modelo del ATmega164P. Además, mediante una herramienta llamada "*Device Pins Mode*" se añadió los pines del microcontrolador con sus nombres respectivos. Finalmente la nueva representación lógica del microcontrolador ATmega164PA se muestra en la siguiente figura [13].

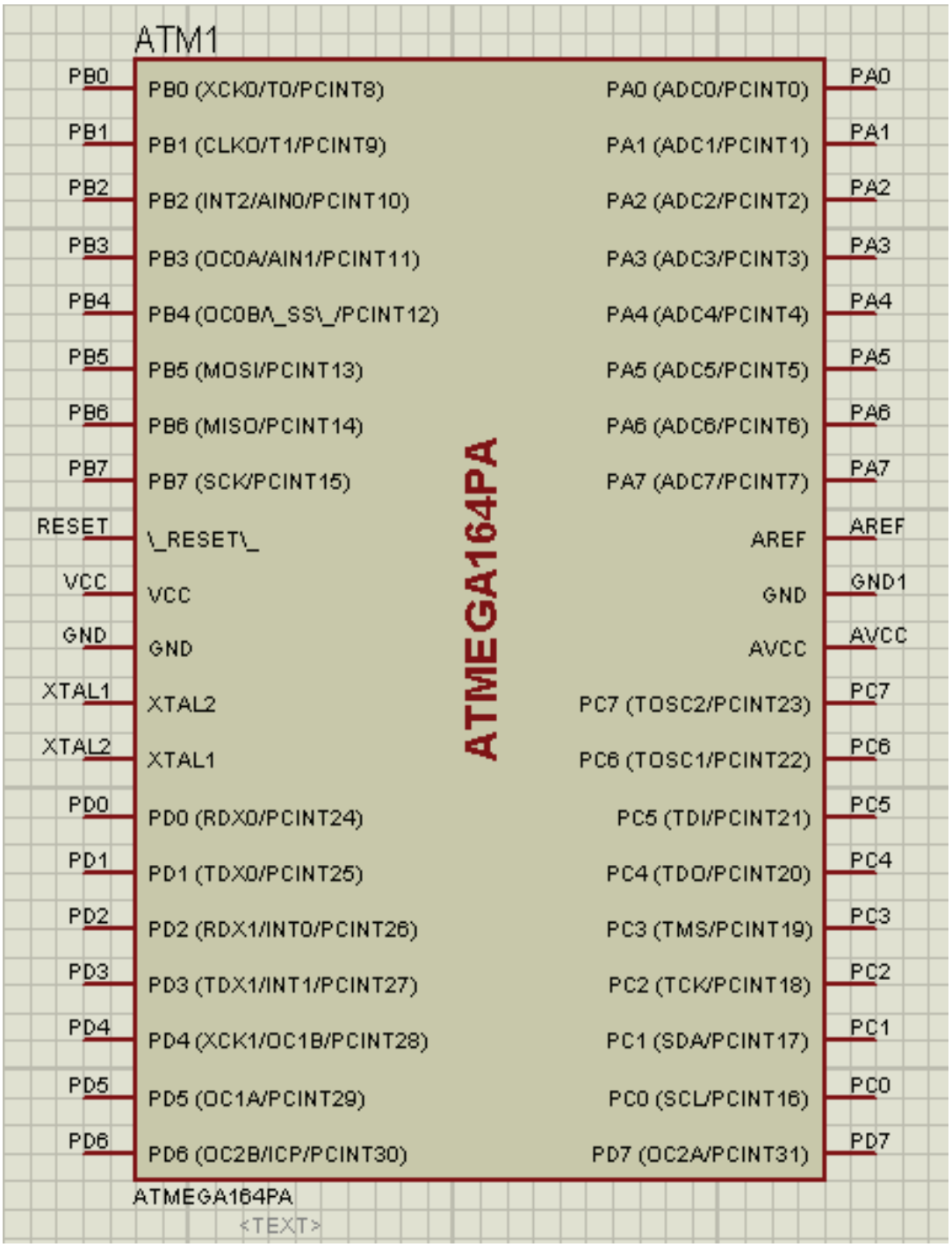


Figura 3.3 ATmega164PA símbolo lógico [1]

Otros componentes electrónicos también fueron creados de la misma manera como por ejemplo: banco de 10 leds y la regleta que sirve de conexión al programador USB. Estas nuevas representaciones no realizan ningún proceso, pero sirvieron para el desarrollo del diseño lógico y físico. En la figura 3.4 se muestran los circuitos de cada uno de los puertos, el circuito de conexión del programador y el circuito de reseteo. En el anexo A se muestra el diseño real con la unión de todos estos circuitos y las demás regletas de conexión (AREF – AVCC) y (XTAL1 – XTAL2).

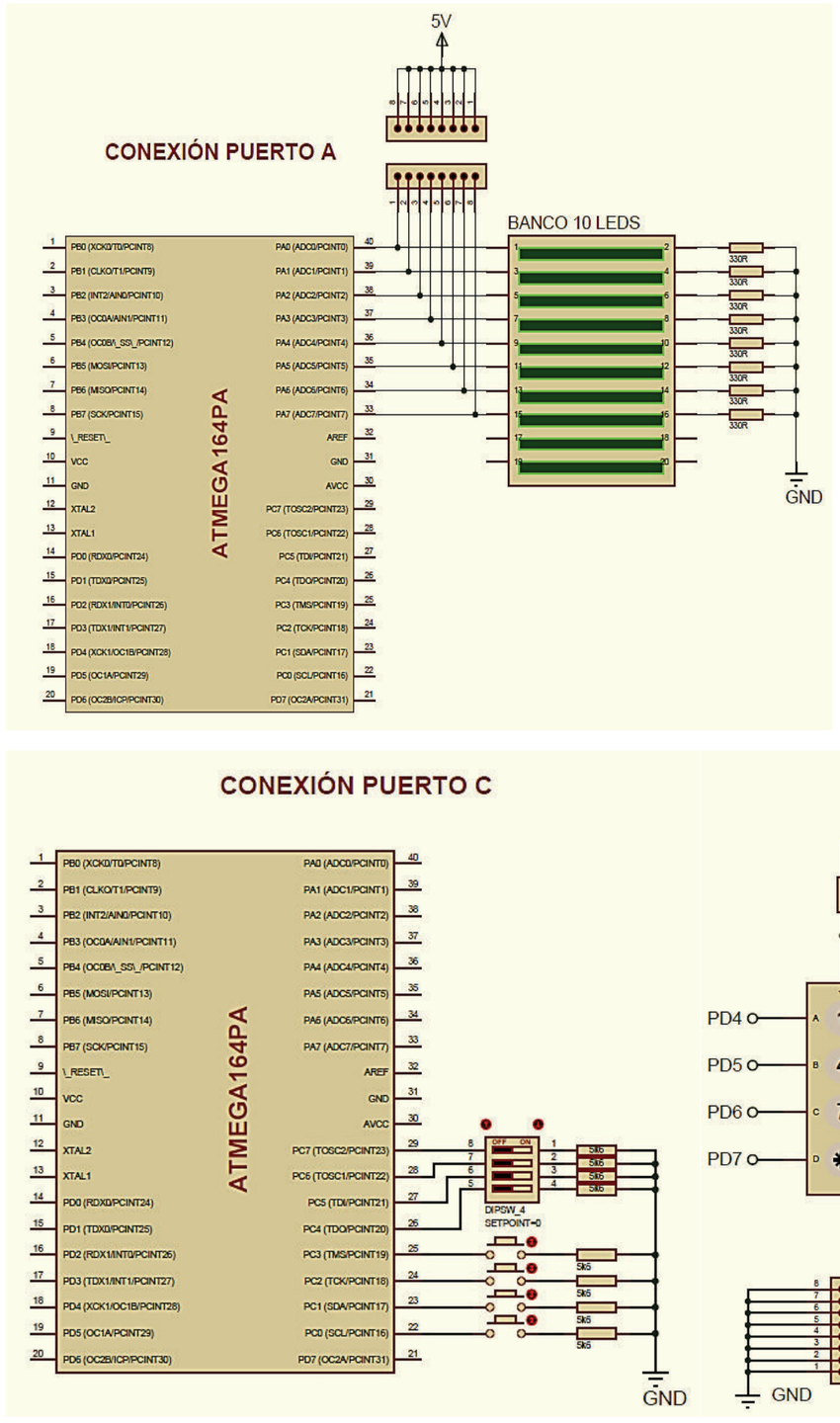


Figura 3.4 Circuitos del módulo didáctico básico

3.2 Diseño físico y construcción de los módulos didácticos básicos

Después de tener el diseño lógico completo, este es transferido a la plataforma ARES (*Advanced Routing and Editing Software*). Esta herramienta de enrutado, ubicación y edición de componentes, se utilizó para la fabricación de las placas de circuito impreso. Al igual que en la anterior se tuvieron que crear los encapsulados y finalmente completar los paquetes PCB (*Printed Circuit Board*).

Los componentes electrónicos que necesitaron de su paquete PCB fueron: microcontrolador ATmega, USBasp, teclado matricial 3x4, *display* de 7 segmentos, pulsadores, *dipswitch* y las diferentes regletas. Para la creación de estos se tuvieron en cuenta detalles como: dimensiones (ancho y largo) y separación entre pines o terminales.

Así mismo como en la plataforma ISIS, mediante las herramientas de diseño se crearon las representaciones reales de cada elemento, un ejemplo es el microcontrolador ATmega mostrado en la figura 3.5. Luego de esto se enlazaron este paquete PCB a su correspondiente representación lógica en el ISIS. Una vez que cada elemento tenga su paquete PCB las pistas de conexión surgieron para su posterior configuración.

Figura 3.5 Representación física del microcontrolador ATmega164PA [1]

Creación de los paquetes PCB

Para este proceso primero se guardaron tanto el diseño lógico y como el físico dentro de las librerías del ISIS y del ARES respectivamente. Ya en la plataforma ISIS se colocó el elemento en este caso el microcontrolador ATmega (fig. 3.3), haciendo clic derecho sobre

él y en la opción "*Make Device*", en las diferentes ventanas de opción aparece una con nombre "*Packaging*" y luego en la opción "*Add/Edit*" [14].

En esta se buscó el encapsulado del microcontrolador ATmega (fig. 3.5) y la imagen de este apareció en la parte derecha para poder ir enlazando los diferentes pines del microcontrolador del diseño lógico al físico. En la figura 3.6 se muestra cómo se asignaron los pines, además, se muestran detalles del microcontrolador como sus medidas y la distancia que existe entre sus pines.

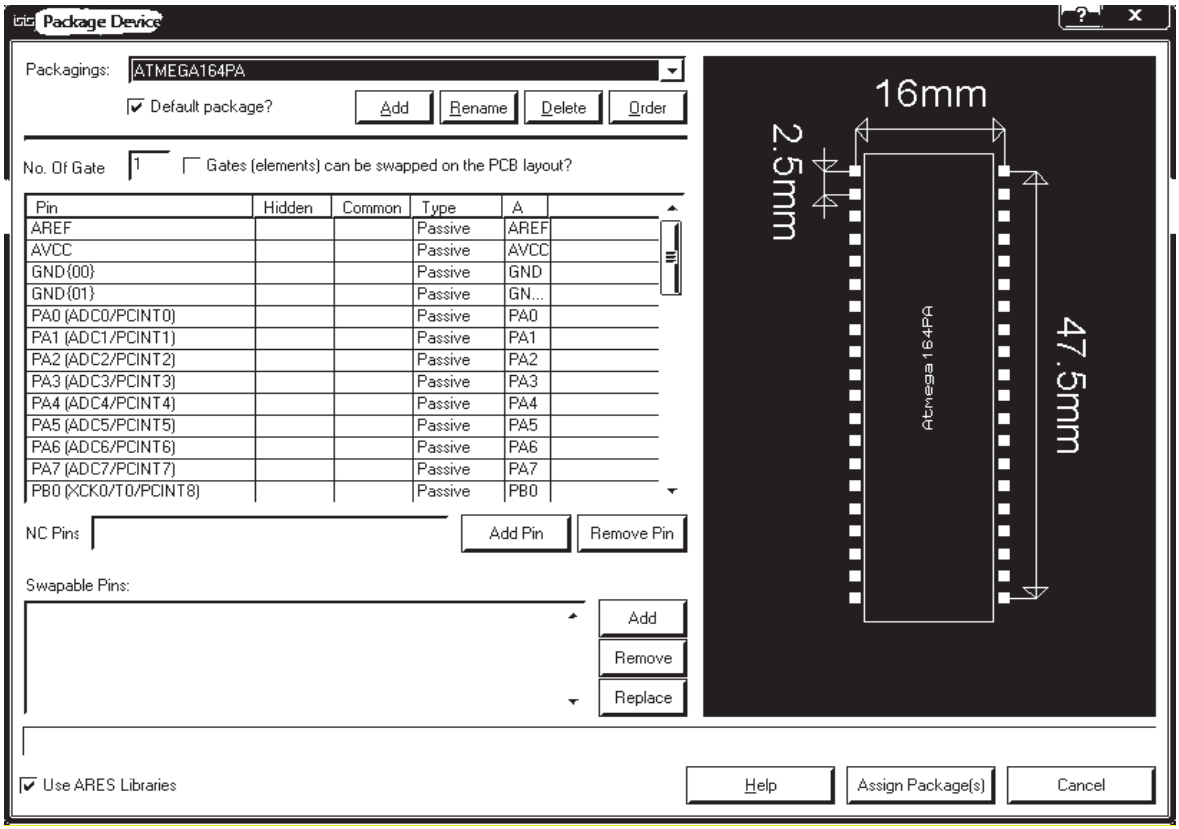


Figura 3.6 Ventana de edición del encapsulado del dispositivo [14]

Ya con todos los pines enlazados se guardó todo con el botón "*Assign Packages*", se realizó el mismo procedimiento para el resto de elementos electrónicos. En la plataforma primero se colocó un ejemplo del tamaño que tendría la baquelita, la cual al ordenar todos los elementos en la baquelita, esta quedó con un tamaño de 15 cm x 17 cm.

Una vez ordenados todos los elementos en la placa se procedió al trazado de las pistas de conexión entre los diferentes elementos. Después, se configuraron detalles como: tamaño de las pistas de forma para que no se produzcan cortos circuitos, tamaño y forma de los pines para que estos no se pierdan al momento de realizar las perforaciones [15]. Finalmente el diseño físico se muestra en la figura 3.7.

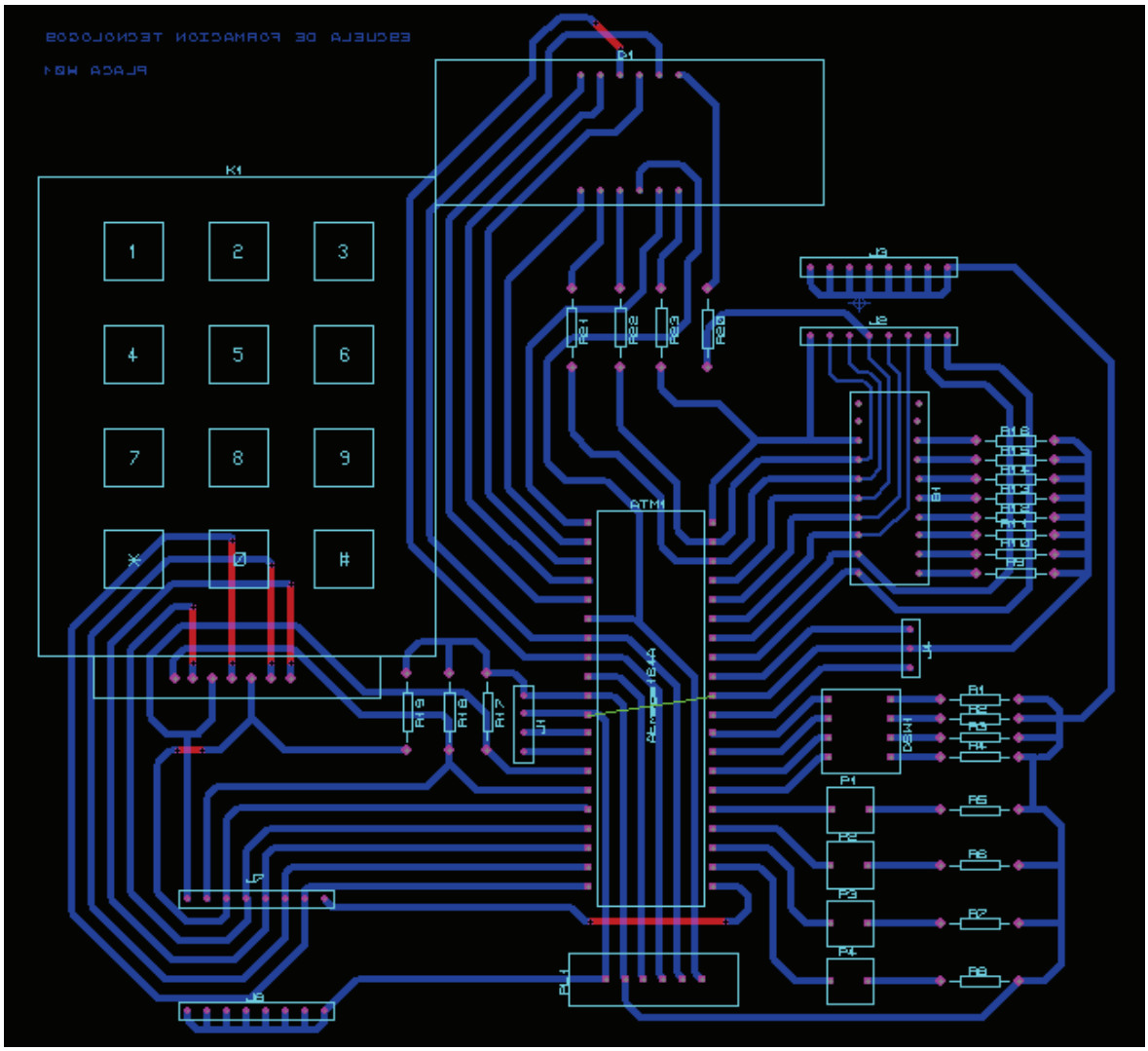


Figura 3.7 Diseño físico del módulo didáctico básico

Construcción del módulo didáctico básico

Para pasar el diseño a la placa baquelita esta debió ser impresa en una hoja termotransferible, además de imprimirla a láser para que al aplicarle calor la imagen se estampe en esta. Ya con el diseño implantado en la baquelita se procedió a sumergirla en agua caliente mezclada con ácido férrico, la cual en la industria electrónica se utiliza como grabado químico de plaquetas de circuito impreso [16].

Al sumergir la baquelita todo el cobre innecesario se fue disolviendo y quedó únicamente la forma del diseño del módulo didáctico. Luego se empezó a perforar los pines y colocar los elementos electrónicos para soldarlos. Se probó que había continuidad entre los elementos por medio del multímetro y una vez probado su funcionamiento con un ejemplo de programa se procedió a ponerle una estampa y un marco de madera para protección. En la figura 3.8 se muestra el módulo didáctico terminado con cada una de sus partes.

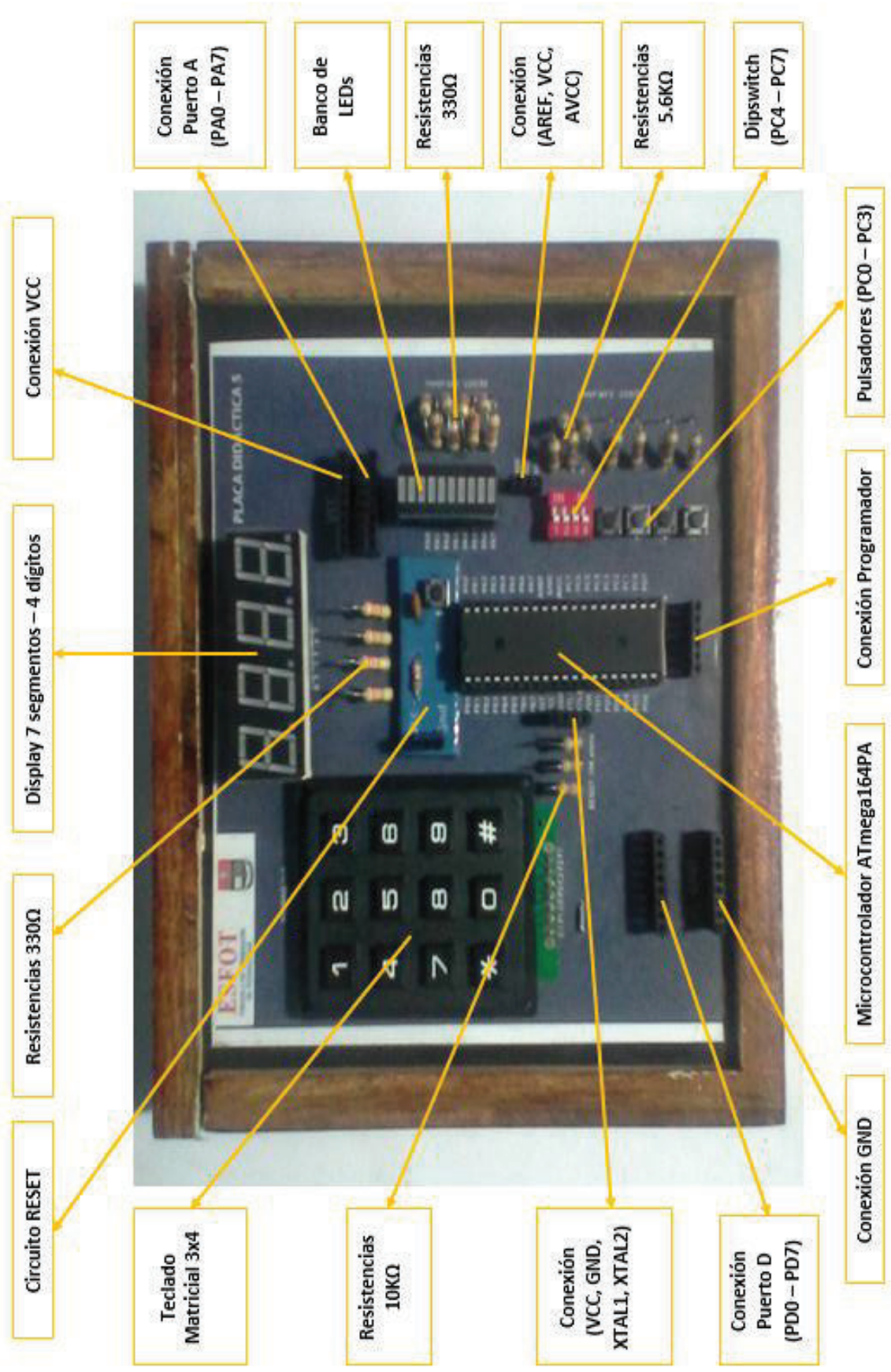


Figura 3.8 Forma final del módulo didáctico básico y sus partes

Costo de los módulos didácticos básicos

En la tabla 3.1 se muestra el costo de cada material que se utilizó para la elaboración de este proyecto, tanto el costo de solo un módulo didáctico básico como el costo total de la construcción de los 10 módulos.

Tabla 3.1 Costos de los elementos del módulo didáctico básico

Material	Precio	Cantidad 1 Módulo	Cantidad Total 10 Módulos	Costo Total
ATmega164PA	$ 6,50	1	10	$ 65,00
Teclado Matricial 3x4	$ 4,50	1	10	$ 45,00
Dipswitch 4 vías	$ 0,80	1	10	$ 8,00
Pulsadores 2P 5 mm	$ 0,15	4	40	$ 6,00
Resistencias de 330Ω	$ 0,02	12	120	$ 2,40
Resistencias de 5,6KΩ	$ 0,02	8	80	$ 1,60
Resistencias de 10KΩ	$ 0,02	3	30	$ 0,60
Display 4 dig. 7 seg. cátodo común	$ 1,00	1	10	$ 10,00
Banco 10 leds	$ 2,75	1	10	$ 27,50
Espadines Macho	$ 2,20	1	3	$ 6,60
Espadines Hembra	$ 1,80	3	40	$ 72,00
Baquelita 30x20	$ 5,60	1	5	$ 28,00
Papel termotransferible	$ 2,30	1	10	$ 23,00
Programadores USBasp	$ 10,00	1	10	$ 100,00
Kit de cables protoboard	$ 2,50		2	$ 5,00

El costo total de la creación de los 10 módulos didácticos básicos fue alrededor de $400. Cada módulo viene con su programador USBasp y además viene con 10 cables protoboard incluidos, en esta tabla no se incluyen gasto como estaño, cautín, pomada y el anaquel donde se colocan los módulos.

3.3 Modelo de hojas guías de laboratorio para estudiantes

Práctica 1. Manejo de puertos de entrada – salida.

Objetivos

- Introducir a los estudiantes en la programación del microcontrolador Atmega164PA utilizando lenguaje de alto nivel en MikroC para AVR.
- Familiarizar a los estudiantes sobre el diseño y los componentes que contienen los módulos didácticos básicos.
- Adaptar a los estudiantes con la forma de programar los microcontroladores con el programador USBASP utilizando el software PROGISP.

Trabajo Preparatorio

Cuestionario

- Investigar cómo trabaja el programa PROGISP: cómo leer memoria, borrar y cargar programas.
- Consultar la función de los pines que se utilizan para programar el microcontrolador.
- Consultar cuál es el comando y la sintaxis de los comandos *DDR, PORT, IF, ELSE, BUTTON* en MikroC.
- Consultar la manera de generar retardos en MikroC.
- Ventajas y desventajas de los lenguajes de alto y bajo nivel.

Actividades a desarrollar

1. Escribir un programa utilizando lenguaje de alto nivel en MikroC para AVR, el cual permita utilizar los pulsadores PC0 y PC1 conectados al puerto C y 8 diodos Leds conectados al puerto A, para que cumplan los siguientes requerimientos:

- OPCIÓN 1: Escribir un programa para encender el bit menos significativo (LSB) conectado en PA0 y realice el desplazamiento del encendido al bit más significativo (MSB) conectado en PA7 con un retardo de 500 ms, el mismo que se repetirá indefinidamente.

- OPCIÓN 2: Utilizando el comando *if* y el comando *button* permitir que al cambiar la posición de PC7, realice el proceso contrario al descrito en la opción 1, es decir, el encendido del bit se desplazará desde el más significativo al menos significativo, sin importar el estado que se encontraba,

así mismo con un retardo de 500 ms y de manera indefinida y que al retornar el interruptor vuelva a la función que realiza en la opción 1.

- OPCIÓN 3: Usando los comandos *if* y *button* permitir que al presionar el pulsador PC0, todos los leds conectados en el puerto A se apaguen con un retardo de 2 segundos y luego que se encienda el led en PA0 para que realice la función principal.

- OPCIÓN 4: Mediante otra condición utilizando los comandos *if* y *button* permitir que al presionar el pulsador PC1, todos los leds conectados en el puerto A se enciendan y apaguen durante 1 segundo y volver a encender el led en PA0 para que realice la función principal.

2. Incluir los diagramas de flujo de los programas creados que explique el funcionamiento de los mismos.

3. Realizar la respectiva simulación probando el funcionamiento del programa realizado.

Equipo disponible en el Laboratorio

- Módulos didácticos básicos.
- Programador USBasp.
- Software de simulación MikroC para AVR.
- Software de grabación Progisp.
- Computador

Procedimiento práctico

- Traer el código del programa hecho en MikroC y la simulación del circuito en Proteus que simule lo solicitado.
- Conectar el grabador USBasp tanto al módulo y a la computadora, y mediante el programa de grabación Progisp cargar el archivo xx.hex.
- Comprobar el funcionamiento del programa en el módulo didáctico.

Informe

- Realizar el programa diseñado en el trabajo preparatorio utilizando lenguaje de bajo nivel y comparar la programación con el realizado en el lenguaje de alto nivel.
- Desarrollar el informe correspondiente en base al formato establecido.

Práctica 2. Periféricos de entrada

Objetivos

- Implementar un circuito usando el *keypad* 3x4 en el microcontrolador Atmega164PA.
- Manejar la programación para usar la matriz de teclas del *keypad* 3x4.
- Conocer los diferentes módulos que se pueden implementar en el microcontrolador.

Trabajo Preparatorio

Cuestionario

- Consultar los comandos y función para inicializar el módulo del *keypad*.
- Consultar la función y sintaxis del comando *switch, case y break*.

Actividades a desarrollar

1.- Elaborar un programa utilizando lenguaje de alto nivel que cumpla con los siguientes requerimientos:

- OPCIÓN 1: Escribir un programa para manejar la matriz del *keypad* 3x4 conectado al puerto D del microcontrolador. Las teclas correspondientes se deberán visualizar en uno de los *displays* de 7 segmentos conectados al puerto B mientras estos sean presionados, por defecto los *displays* de 7 segmentos deberán permanecer en blanco mientras no se presione ninguna tecla. La polarización de los 4 *displays*, lo realizan los 4 primeros pines del puerto A (PA0, PA1, PA2, PA3).

2. Incluir el diagrama de flujo del programa creado que explique el funcionamiento del mismo.

3. Realizar la respectiva simulación en el Proteus.

Procedimiento práctico

- Traer el código del programa hecho en MikroC y la simulación del circuito en Proteus, realizando lo que se pide en el diseño.
- Grabar el programa xx.hex en el microcontrolador al módulo didáctico y realizar la comprobación del funcionamiento del programa.

Informe

- Desarrollar el informe correspondiente en base al formato establecido.

Práctica 3. Convertidor analógico – digital y señales PWM.

Objetivos

- Implementar un circuito usando convertidor analógico – digital (ADC) del microcontrolador Atmega164PA.
- Implementar un circuito generando una onda PWM en el microcontrolador Atmega164PA.
- Acoplar elementos electrónicos a los módulos didácticos básicos.

Trabajo Preparatorio

Cuestionario

- Consultar la sintaxis para configurar un solo pin de cualquier puerto.
- Consultar el comando para poder usar los ADC (comando inicializador y comando de lectura).
- Consultar el comando para inicializar el módulo PWM y sus componentes.
- Consultar el rango de valores que el microcontrolador Atmega164PA puede leer de un canal analógico y de la onda PWM.
- Consultar la función de los pines AREF y AVCC del microcontrolador.
- Consultar en el *data sheet* del Atmega164PA, que puertos y que pines permiten generar ondas PWM y que pines pueden ser usados como canales analógicos.

Actividades a desarrollar

1. Elaborar un programa utilizando lenguaje de alto nivel que cumpla con los siguientes requerimientos:

- OPCIÓN 1: Configurar el pin PA2 como entrada para usar su canal analógico y el pin PA0 como salida y crear las siguientes condiciones: cuando el valor de entrada entregada por el potenciómetro supere los 3 voltios el pin PA0 parpadee con un retardo de un segundo, mientras que si el valor de entrada es menor de 2 voltios el pin PA0 permanecerá en 1 lógico y si el valor de entrada esta entre los 2 y 3 voltios el pin PA0 permanecerá en 0 lógico.
- OPCIÓN 2: Configurando el pin PA6 para usar su canal analógico y uno de los *displays* conectado al puerto B, mediante las siguientes condiciones: cuando el valor de entrada entregada por el potenciómetro supere los 3 voltios en el *displays* de 7 segmentos mostrará el número 1, si el valor de entrada es menor a 2 voltios en el *display* mostrará el número 0 y si el valor de entrada esta entre los 2 y los 3 voltios el *display* este en blanco.

- OPCIÓN 3: Usando nuevamente el canal analógico 6, realizar el control de una onda PWM, la cual tendrá su salida a través de un *buzzer* conectado en el pin PD6. La amplitud de la onda deberá ser proporcional al valor de entrada del canal analógico.

2. Incluir los diagramas de flujo de los programas creados que explique el funcionamiento de los mismos.
3. Realizar las respectivas simulaciones en el Proteus.

Equipo disponible en el laboratorio

- Software de simulación MikroC para AVR.
- Módulos didácticos básicos.
- Programador USBasp.
- Software de programación Progisp.
- Cables de protoboard.
- Computador.

Equipo requerido para la práctica

- Potenciómetro de 10KΩ.
- Buzzer de 5VDC.
- Resistencia de 330Ω.

Procedimiento práctico

- Traer el código del programa hecho en MikroC y la simulación del circuito en Proteus, realizando lo que se pide en el diseño.
- Grabar el primer programa xx.hex en el microcontrolador y realizar las conexiones que se piden en la opción 1.
- Borrar el programa anterior del microcontrolador y grabar el nuevo programa xx.hex; realizar las conexiones que se piden en la opción 2 y probar el funcionamiento.
- Borrar el programa anterior del microcontrolador y grabar el nuevo programa xx.hex; realizar las conexiones que se piden en la opción 3 y probar el funcionamiento.

Informe

- Desarrollar el informe correspondiente en base al formato establecido.

Práctica 4. Manejo de temporizadores

Objetivos

- Implementar circuitos utilizando el microcontrolador Atmega164PA para el manejo de los *timers*.
- Conocer las formas de usar un temporizador.
- Comparar la programación utilizando retardos y temporizadores.

Trabajo Preparatorio

Cuestionario

- Consultar para qué sirven los *timers*.
- Consultar la función del pre - escaler.
- Consultar cuántos *timers* tiene el microcontrolador Atmega164PA.
- Consultar los comandos que se necesitan para habilitar las interrupciones mediante los *timers*.

Actividades a desarrollar

1. Elaborar un programa utilizando lenguaje de alto nivel que cumpla con los siguientes requerimientos:

    - OPCIÓN1: Escribir un programa para realizar un contador módulo 10 (0 – 9) en uno de los *displays* y cuando el contador llegue a 9 se active el *timer1* por desbordamiento, una vez activado el *timer1* los *displays* permanecerán apagados y los leds conectados al puerto A se encenderán por 1. Pasado ese segundo el *timer1* se desactivará y se repetirá el conteo.

    - OPCIÓN 2: Desarrollar un programa para generar un contador módulo 20 (0 – 19), el cual se visualizará en los *displays*. El contador deberá ir

incrementándose mediante la activación del *timer1*. La activación del *timer1* la realizará al presionarse el pulsador PC0, caso contrario si no se presiona el pulsador, el contador se detendrá.

2. Incluir los diagramas de flujo de los programas creados que explique el funcionamiento de los mismos.

3. Realizar las respectivas simulaciones probando el funcionamiento de los diseños realizados.

Equipo disponible en el laboratorio

- Software de simulación MikroC para AVR.
- Módulos didácticos básicos.
- Programador USBasp.
- Software de programación Progisp.
- Cables de protoboard.
- Computador.

Procedimiento práctico

- Traer el código del programa hecho en MikroC y la simulación del circuito en Proteus, realizando lo que se pide en el diseño.

- Grabar el primer programa xx.hex en el microcontrolador y realizar la comprobación del funcionamiento del programa.

- Borrar el programa anterior del microcontrolador y grabar el nuevo programa xx.hex y comprobar el funcionamiento del programa en el módulo didáctico.

Informe

- Desarrollar el informe correspondiente en base al formato establecido.

Práctica 5. Manejo de interrupciones.

Objetivos

- Implementar circuitos utilizando el microcontrolador Atmega164PA para el manejo de interrupciones.

- Generar interrupciones externas al programa principal que esté realizando el microcontrolador.

Trabajo Preparatorio

Cuestionario

- Consultar el concepto de interrupción para microcontroladores.

- Consultar los tipos de interrupciones que se pueden generar utilizando el microcontrolador Atmega164PA.

- Consultar cómo hacer el llamado a una función que está fuera de la función principal.

Actividades a desarrollar

1.- Elaborar un programa utilizando lenguaje de alto nivel que cumpla con los siguientes requerimientos:

- OPCIÓN 1: Escribir un programa que realice un contador módulo 60 (0 – 59) el cual se visualizará en dos de los *displays* conectados al puerto B y mediante la interrupción del pulsador PC1 se reseteará el conteo.

- OPCIÓN 2: Escribir un programa que permita presentar en el arreglo de *displays*, aplicando el criterio de barrido, la palabra “HOLA” de forma permanente. Mediante el pulsador PC0 hacer el llamado a la función interrupción que presentará la palabra “POLI” durante 2 segundos para posteriormente dejar los *displays* en blanco y vuelva a la función principal.

Además, mediante el pulsador PC1 hacer el llamado a otra función interrupción que esta vez presentará la palabra “CHAO” igualmente por 2 segundos y de la misma manera al finalizar dejará los *displays* en blanco para que retorne a la función principal.

2.- Incluir los diagramas de flujo de los programas creados que explique el funcionamiento de los mismos.

3.- Realizar las respectivas simulaciones probando el funcionamiento de los programas desarrollados.

Equipo disponible en el laboratorio

- Software de simulación MikroC para AVR.
- Módulos didácticos básicos.
- Programador USBasp.
- Software de programación Progisp
- Cables de protoboard.
- Computador.

Procedimiento práctico

- Traer el código del programa hecho en MikroC y la simulación del circuito en Proteus, realizando lo que se pide en el diseño.

- Grabar el primer programa xx.hex en el microcontrolador y realizar la comprobación del funcionamiento del programa.

- Borrar el programa anterior del microcontrolador y grabar el nuevo programa xx.hex y comprobar el funcionamiento del programa en el módulo didáctico.

Informe

- Desarrollar el informe correspondiente al formato establecido.

3.4 Modelo de hojas guías de laboratorio para instructores

Práctica 1. Manejo de puertos de entrada – salida.

Configuración de software

- **Comandos que se pueden utilizar para esta práctica:**

DDRX: configura todos los pines del puerto (A, B, C, D) ya sean de entrada (0) o de salida (1).

Ej. DDRA = 0xFF;

DDX#_Bit: configura únicamente el pin asignado (0 – 7) del puerto como entrada o salida.

Ej. DDA1_Bit = 0; DDD7_Bit = 1;

PortX: asigna valor a los puertos puede ser de forma hexadecimal (0xFF) o de forma binaria (0b11110000).

Ej. PortA = 0xFF; PortB = 0b11110000;

PortX.B#: asigna un estado lógico únicamente al pin seleccionado (0 – 1).

Ej. PortB.B0 = 1; PortA.B5 = 0;

Delay_ms, delay_us: genera un retardo en función de mili o micro segundos.

Ej. delay_ms (500); delay_us (800).

If: comando para establecer una condición.

Ej. If (portd == 0xFF); el doble igual es para establecer la operación de igual

Else: comando para que realice la operación si la condición if no se cumple pero dado el caso si se establece varias condiciones.

Ej. Else portd.B1 = 0;

Button: esta es parte de una librería que sirve para detectar cuando se presiona un botón y se produce un cambio en el estado lógico del pin al que esté conectado el botón.

Ej. Button (&PinB, 0, 1, 1)); detecta un uno lógico el pin 0 del puerto B.

0: pin del puerto, **1:** periodo de anti rebote en milisegundos, **1:** estado lógico que detectara.

- **Programación del microcontrolador a través del Progisp**

Seleccionar correctamente el modelo del microcontrolador antes de grabar, tal como se muestra en la figura 3.9.

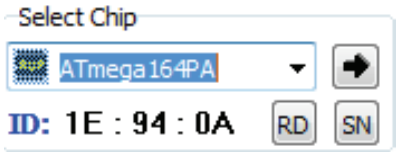


Figura 3.9 Selección del modelo del microcontrolador

De preferencia borrar la memoria *flash* del microcontrolador (*Erase*) ya que puede saltar una especie de error si se trata de grabar y ya tiene algún programa previo al grabado, en la figura 3.10 se puede observar las opciones seleccionadas más adecuadas para borrar y cargar programas [17].

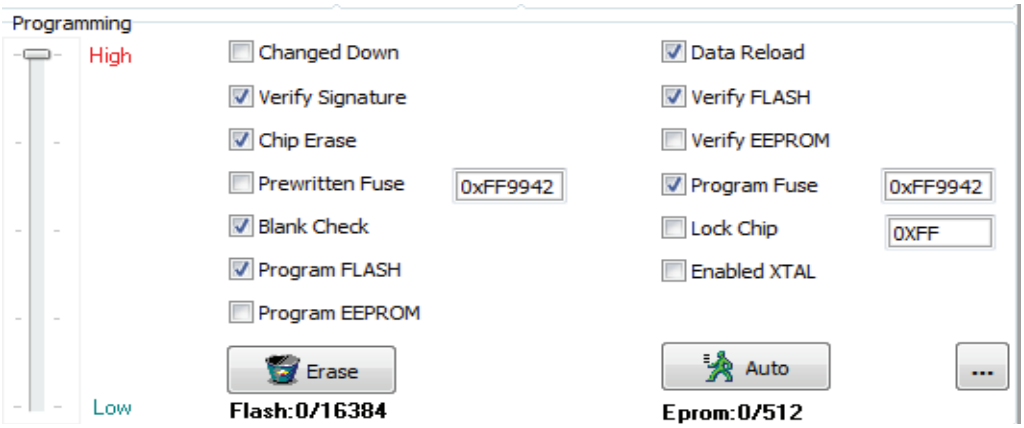


Figura 3.10 Configuraciones de programación

Si al momento de grabar, borrar o leer el microcontrolador, aparece el error mostrado en la figura 3.11, la razón puede ser que el microcontrolador no esté bien conectado o no está haciendo contacto con los pines del quemador, por tanto asegurarse que haya contacto entre ellos.

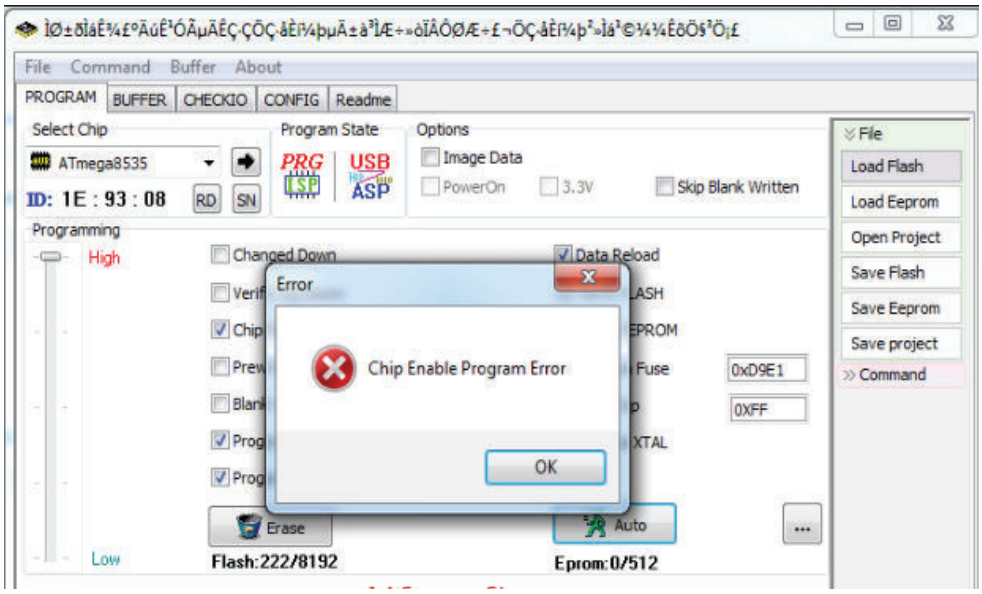


Figura 3.11 Error de habilitación de programa en chip

Configuración de hardware

Los pines de conexión para el programador USB están especificados tanto en el módulo didáctico como en el mismo programador. Se puede observar la forma de conexión o de acoplamiento del programador al módulo didáctico. El programador, la plaquita de conexión y el módulo tienen indicado los pines y formas de conectar.

Figura 3.12 Conexión del programador USB al módulo

El banco de leds está conectado al puerto A del microcontrolador y únicamente enciende 8 de los 10 leds. Los pulsadores e interruptores están conectados al puerto C, pulsadores (PC0 – PC3) e interruptores (PC4 – PC7).

Figura 3.13 Banco de leds en el puerto A

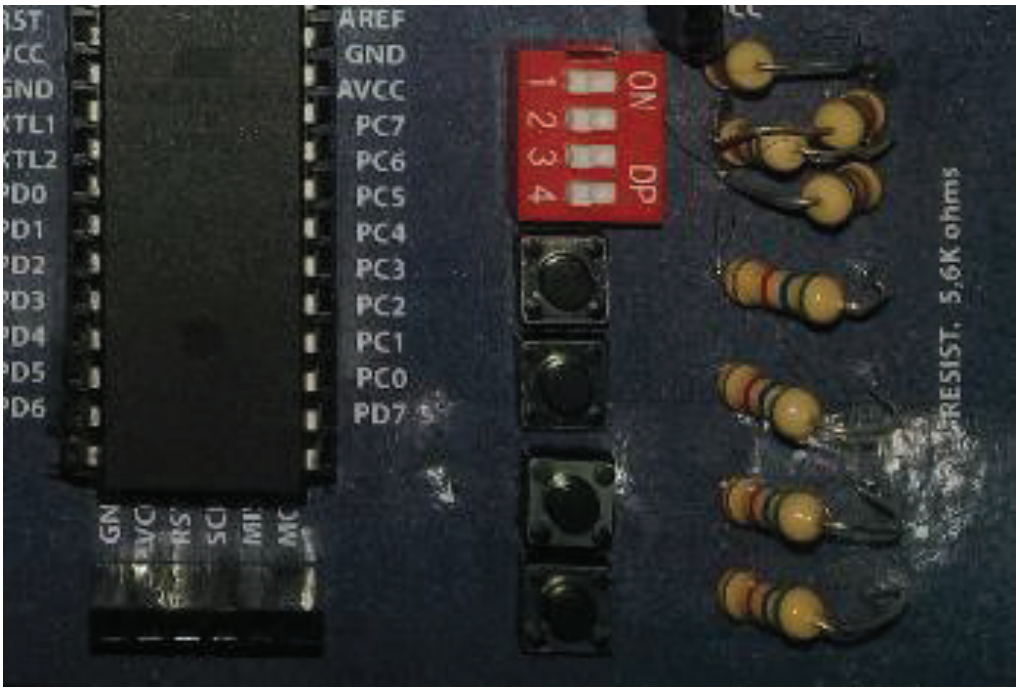


Figura 3.14 Dipswitch y pulsadores en el puerto C

Tanto los interruptores como los pulsadores al cerrarse su conexión irán directamente a tierra, es decir, detectarán un 0 lógico. La conexión de estos es una conexión *Pull – Up*, como se observa en la siguiente figura.

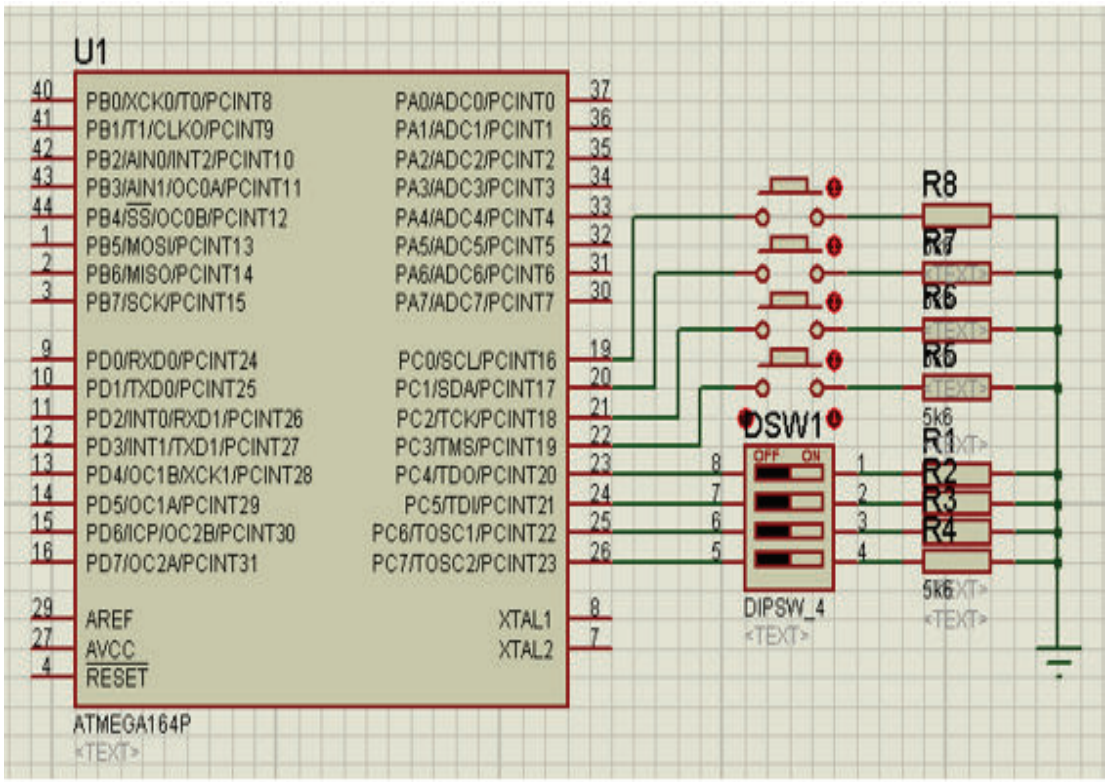
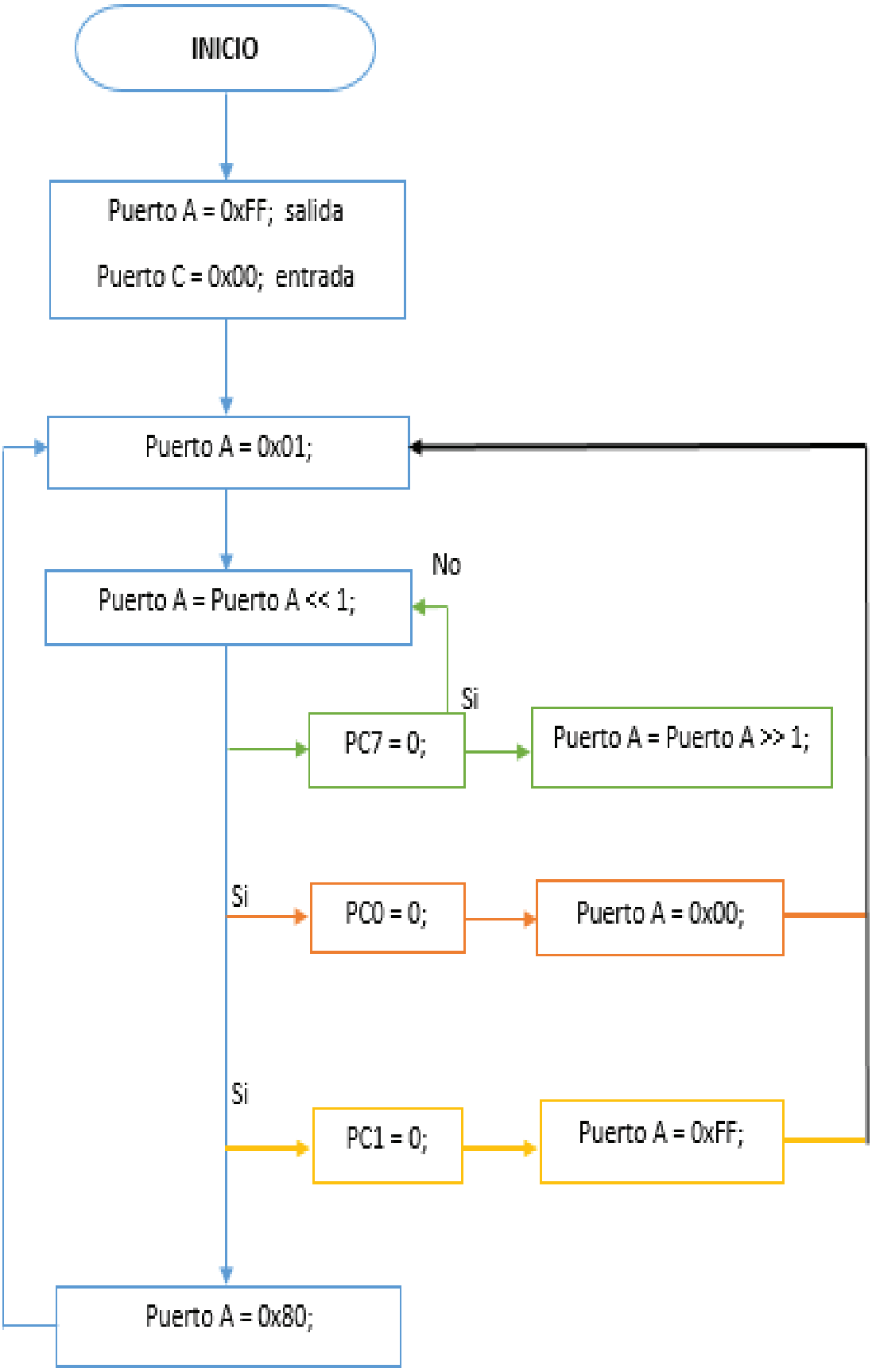


Figura 3.15 Representación lógica de la conexión en el puerto C

Diagrama de flujo



Simulaciones

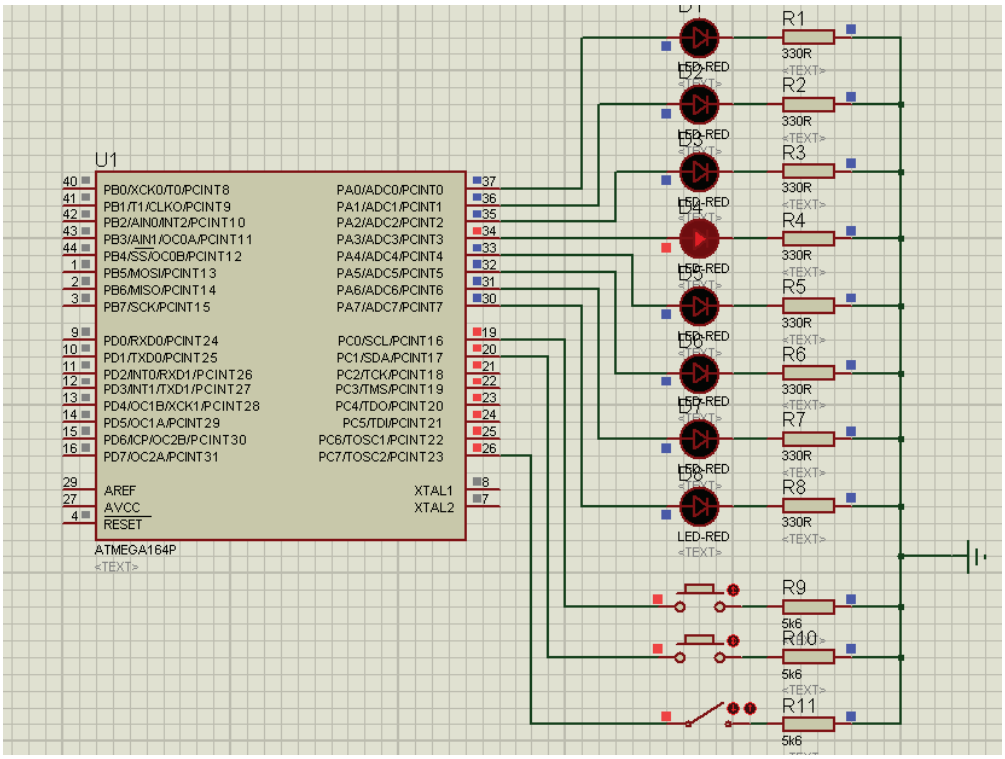


Figura 3.16 Simulación de la práctica 1

Ejemplo de programación.

Opción 1: Escribir un programa para encender el bit menos significativo (LSB) conectado en PA0 y realice el desplazamiento del encendido al bit más significativo (MSB) conectado en PA7 con un retardo de 500 ms, el mismo que se repetirá indefinidamente.

```
void main() {
     DDRA = 0xFF;        //Puerto A configurado como salidas
     DDRC = 0x00;        //Puerto C configurados como entradas
     porta = 0x01;       //Valor inicial en el puerto A
     portc = 0xFF;       //Valor inicial en el puerto C

  do{
        porta = porta << 1;        //Desplazamiento bit A0 a A7
        delay_ms(500);             //retardo 500 milisegundos
        if (porta == 0x80){        //Condicion para repetir proceso
        porta = 0x01;              //encendido del bit A0
        delay_ms(500);             //retardo de 500 milisegundos
        }

  }  while(1);                     //Lazo infinito
}
```

Figura 3.17 Código de programa de la opción 1 de práctica 1

Opción 2: Utilizando el comando *if* y el comando *button* permitir que al cambiar la posición del interruptor PC7, realice el proceso contrario al descrito en la opción 1, es decir, el encendido del bit se desplazará desde el más significativo al menos significativo, sin importar el estado que se encontraba, así mismo con un retardo de 500 ms y de manera indefinida y que al retornar el interruptor vuelva a la función que realiza en la opción 1.

```
do{
    if (Button(&pinc,7,1,0)){    //Condicion al presionar pulsador PC0
        porta = porta >> 1;       //Desplazamiento bit A7 a A0
        delay_ms(500);            //retardo de 500 milisegundos
        if (porta == 0x01){       //Condicion para repetir proceso
            porta = 0x80;         //encendido del bit A7
            delay_ms(500);        //retardo de 500 milisegundos
        }
    }
    else{                         //Proceso principal
        porta = porta << 1;       //Desplazamiento bit A0 a A7
        delay_ms(500);            //retardo 500 milisegundos
        if (porta == 0x80){       //Condicion para repetir proceso
        porta = 0x01;             //encendido del bit A0
        delay_ms(500);            //retardo de 500 milisegundos
        }
    }
```

Figura 3.18 Código de programa de la opción 2 de práctica 1

Opción 3: Mediante otra condición usando los comandos *if* y *button* permitir que al presionar el pulsador PC0, todos los leds conectados en el puerto A se apaguen con un retardo de 2 segundos y luego que se encienda el led en PA0 para que realice la función principal.

```
if (Button(&pinc,0,1,0)){     //Condicion al presionar pulsador PC0
   porta = 0x00;              //Todos bits puerto A en 0
   delay_ms(2000);            //retardo de 2 segundos
   porta = 0x01;              //valor inicial puerto A para proceso principal
   delay_ms(500);             //retardo de 500 milisegundos
}
```

Figura 3.19 Código de programa de la opción 3 de práctica 1

Opción 4: Mediante otra condición usando los comandos *if* y *button* permitir que al presionar el pulsador PC1, todos los leds conectados en el puerto A se enciendan y apaguen durante 1 segundo y volver a encender el leds en PA0 para que realice la función principal.

```
if (Button(&pinc,1,1,0)){     //Condicion al presionar pulsador PC1
   porta = 0xFF;              //Todos bits puerto A en 1
   delay_ms(1000);            //retardo de 1 segundo
   porta = 0x00;              //Todos bits puerto A en 0
   delay_ms(1000);            //retardo de 1 segundo
   porta = 0x01;              //Valor inicial puerto A para proceso principal
   delay_ms(500);             //retardo de 500 milisegundos
}
```

Figura 3.20 Código de programa de la opción 4 de práctica 1

Práctica 2. Periféricos de entrada

Configuración de software

- **Comandos que se pueden utilizar para la práctica**

Para utilizar el keypad se debe inicializar el módulo especificando el puerto al cual está conectado.

KeypadPort: Inicia la comunicación con el puerto establecido.
Ej. KeypadPort at PortB;

Keypad_PortDirection: Dirección del puerto del Keypad.
Ej. Keypad_PortDirection at DDRB;

Char: Es una variante integral para datos, en tamaño de bytes es 1 y por tanto su rango es de 0 a 255. Con esto puede ayudar para asignar los caracteres que se desea ver al presionar las teclas del keypad.
Ej. Const Char Keymap [4] = {´1´, ´2´, ´3´, ´4};

Switch: Sentencia utilizada para pasar el control a una rama de programa específica, en función de una determinada condición.
Ej. Switch (key) {
}

Case: Etiqueta de la condición, que se realizará si esta se cumple.
Ej. Case ´1´: KP = 0x06;

Break: Sentencia utilizada para evitar la ejecución de otros casos, cada caso debe finalizar con un corte o descanso.
Ej. Case ´2´: KP = 0x5B; break;

Default: Sentencia que sirve para establecer una condición si ninguno de los casos se encuentra en operación.
Ej. Default: KP = 0x00; break;

MikroC AVR tiene una herramienta para visualizar los caracteres en el *display* de 7 segmentos que es el editor de 7 segmentos, en el cual se encuentra el código del carácter, tanto si el *display* es de ánodo o cátodo común.

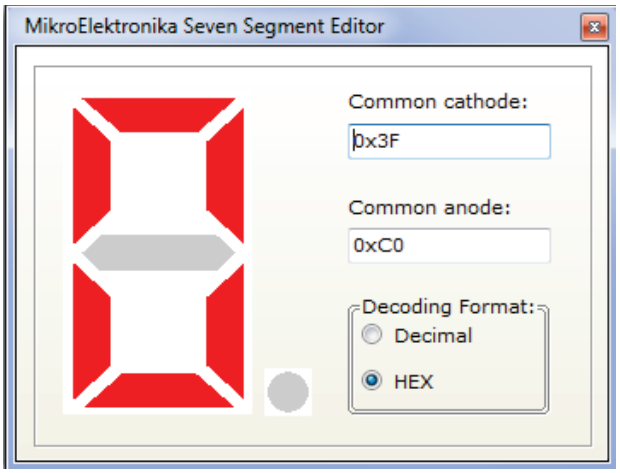


Figura 3.21 Editor de 7 segmentos

Configuración de hardware

El *keypad* 3x4 está conectado al puerto D del microcontrolador, las columnas se conectan a los pines PD0, PD1 y PD2, respectivamente la C0, C1 y C2, además de estar conectadas a sus resistencias de 10K para protección. Mientras las filas se conectan a los pines PD4, PD5, PD6 y PD7. El pin PD3 está libre sin ninguna conexión a algún elemento.

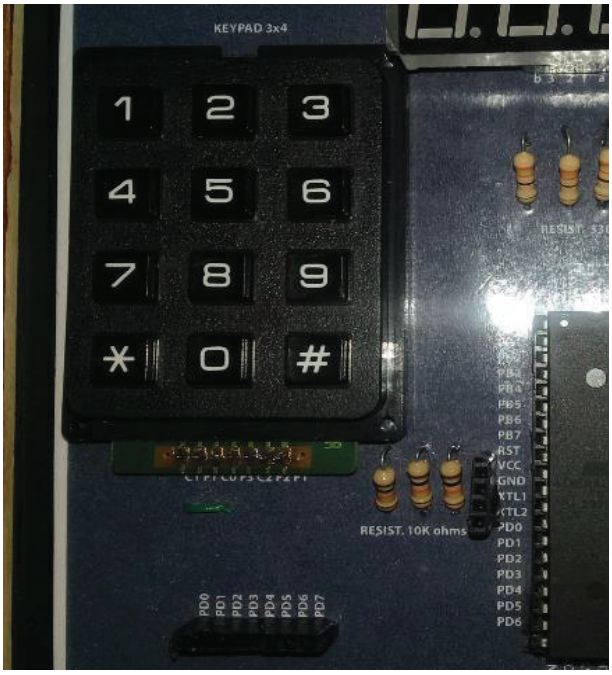


Figura 3.22 Teclado matricial 3x4 conectado al puerto D

El arreglo de *display* o el *display* de 7 segmentos de 4 dígitos se encuentra conectado al puerto B del microcontrolador. Estos *displays* en su totalidad son de cátodo común. Los controles para estos o las polarizaciones de estos se encuentran en los 4 primeros pines del puerto A (PA0, PA1, PA2, PA3). Así que para usarlos estos pines también deberán ser configurados en el programa.

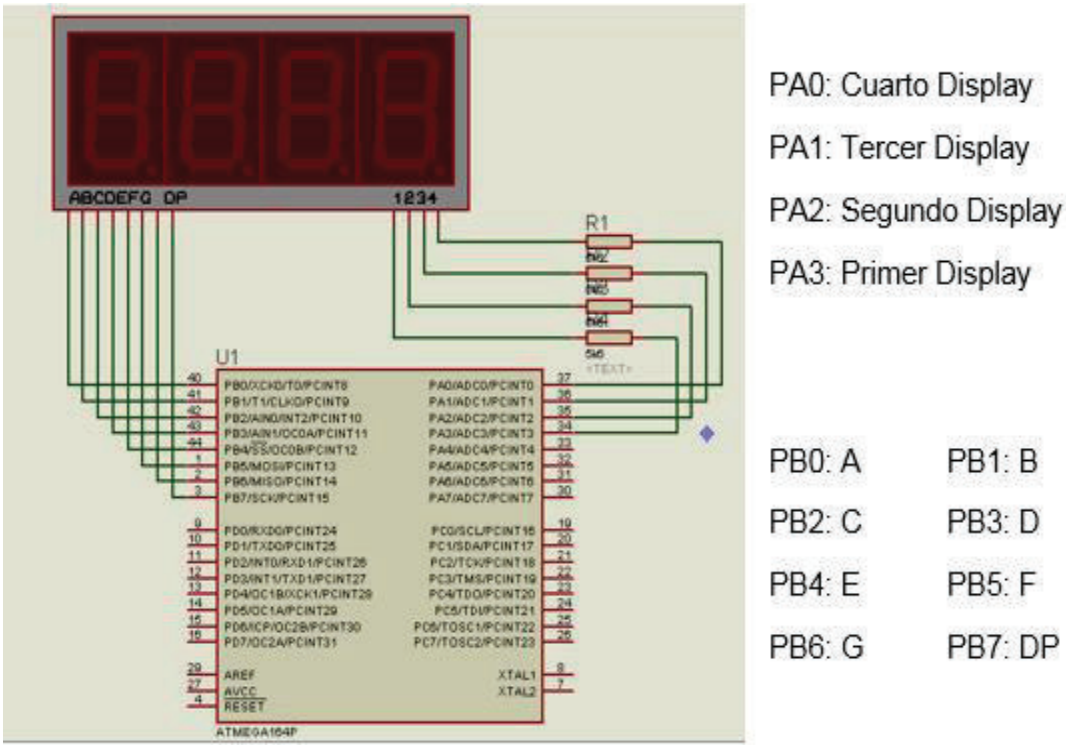
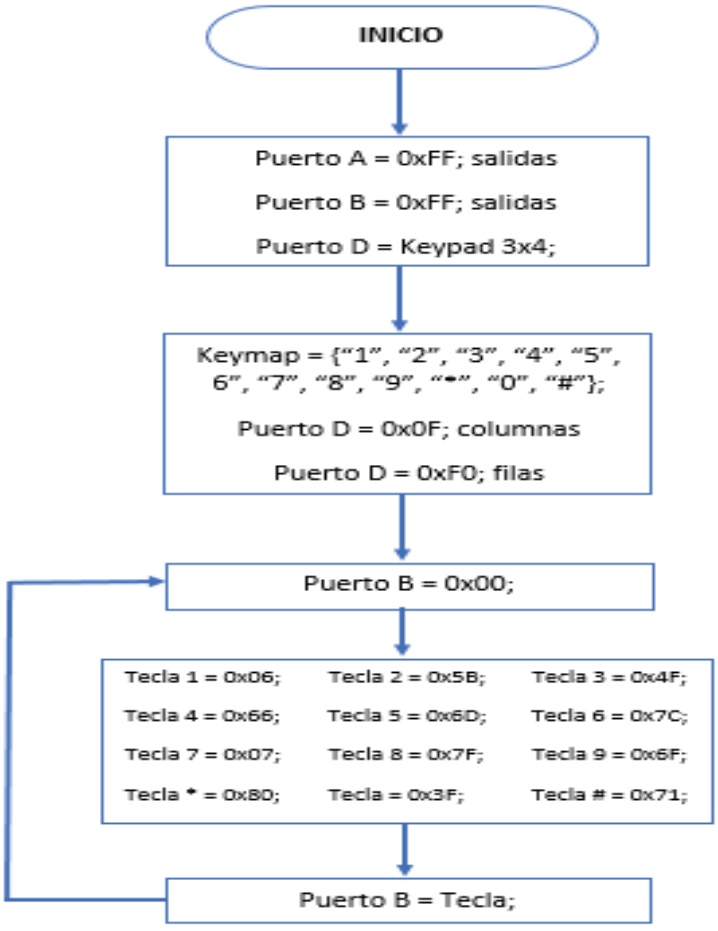


Figura 3.23 Conexión lógica del display de 7 segmentos al puerto B

Diagramas de flujo



Simulaciones

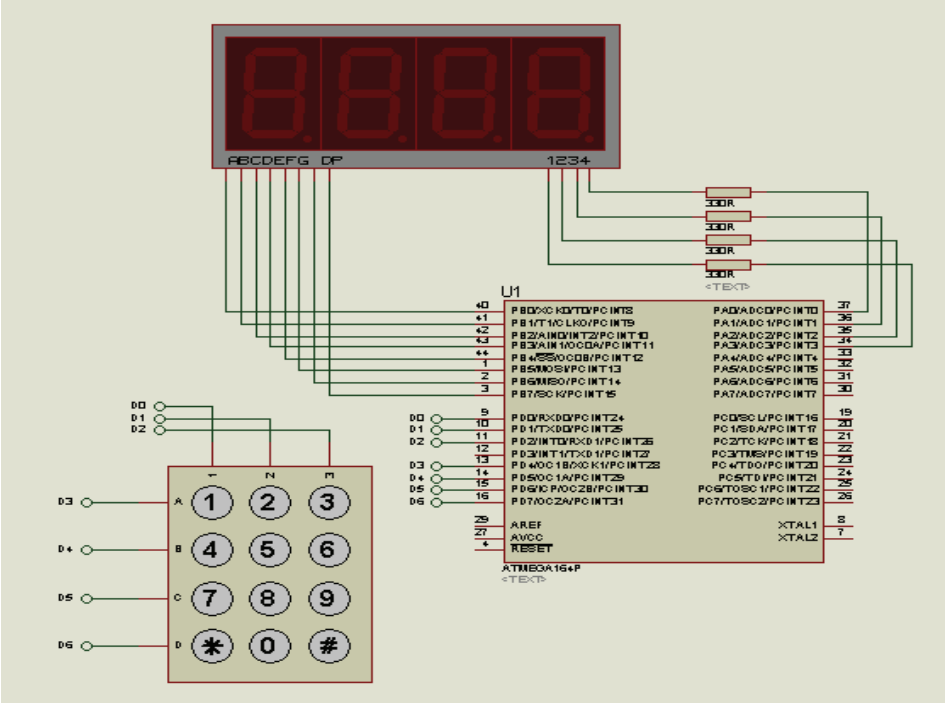


Figura 3.24 Simulación de práctica 2

Ejemplo de programación

Opción 1: Escribir un programa para manejar la matriz del *keypad* 3x4 conectado al puerto D del microcontrolador. Las teclas correspondientes se deberán visualizar en uno de los *displays* de 7 segmentos conectados al puerto B mientras estos sean presionado, por defecto los *displays* de 7 segmentos deberán permanecer en blanco mientras no se presione ninguna tecla. La polarización de los 4 *displays*, lo realizan los 4 primeros pines del puerto A (PA0, PA1, PA2, PA3).

```
const char keymap[13]={'/','1','4','7','*','2','5','8','0','3','6','9','#'};
unsigned short kp1;
short int kp;

//Keypad module conections
char keypadport at portd;
char keypadport_direction at DDRD;
char keypadPin at PinD;

//Funcion Keypad
void Keypad_Init_AVR(){
     keypadport_direction = 0xf0;
     keypadport = 0x0f;
}
```

Figura 3.25 Inicialización del módulo del keypad

```
char Keypad_Key_Click_AVR(){
     char aux = 0;
     char tecla;
     char columna = 0;
     char fila = 0;
     tecla = 0;
     columna = keypadPin&0b00001111;
     if ((columna) == 0b00001111){
     }
     else{
          aux = 0x0F;
          fila = columna;
          while(fila != 0b00001111){
          aux = (aux << 1) + 1;
          keypadPort = aux;
          delay_ms(5);
          fila = keypadPin&0b00001111;
          tecla = tecla + 1;
          }
          columna = (~columna) & 0b00001111;
          while(columna > 1){
          tecla = tecla + 4;
          columna = columna >> 1;
          }
     }
     keypadport = 0x0F;
     return tecla;
}
```

Figura 3.26 Código de programa para leer la matriz del teclado

```
void main(){
    DDRA = 0xFF;
    DDRB = 0xFF;
    Keypad_Init_AVR();
    do{
         kp1 = Keymap[Keypad_Key_Click_AVR()];
         switch(kp1){
           case '1': kp = 0x06; break;
           case '2': kp = 0x5B; break;
           case '3': kp = 0x4F; break;
           case '4': kp = 0x66; break;
           case '5': kp = 0x6D; break;
           case '6': kp = 0x7C; break;
           case '7': kp = 0x07; break;
           case '8': kp = 0x7F; break;
           case '9': kp = 0x6F; break;
           case '*': kp = 0x80; break;
           case '0': kp = 0x3F; break;
           case '#': kp = 0x71; break;
           default : kp = 0x00; break;
         }
         porta = 0x0E;
         portb = kp;

    }while(1);
}
```

Figura 3.27 Código programa para lectura del teclado

Recomendaciones

- Sugerir a los estudiantes que tengan cuidado al momento de manipular los keypad ya que pueden abrirse las pistas.
- Si hay algún problema al mostrar la tecla presionada, la razón puede ser que exista contacto entre las sueldas, normalmente se pueden acumular impurezas entre estos puntos de soldadura que pueden ocasionar un corto.

Práctica 3. Convertidor analógico – digital y señales PWM.

Configuración de software

- **Comandos para utilizar convertidores analógico – digital.**

Para el uso de los convertidores Analógicos – Digitales o ADC, los pines deben estar configurados como entradas, el comando inicializador es para todos los ADC, pero también se debe configurar que canal o canales se van a utilizar.

ADC_Init (): Inicializa el módulo ADC.
Ej. ADC_Init ();

ADC_Read (X): realiza la lectura del canal analógico que se establece (0 – 7).
Ej. PortB = ADC_Read (3);

Todos los canales analógicos tienen el mismo rango de lectura de 10 *bits*, es decir 1024 unidades (0 – 1023), esto en función de la alimentación que se esté dando al canal analógico. Si la alimentación es de 5V (1023 = 5V), lo mismo si la alimentación es de 10V (1023 = 10V).

- **Comandos para utilizar módulos PWM.**

Para producir una onda PWM, así como los ADC hay que inicializar el módulo, sin embargo, aún debe especificarse el pin por el cual la onda PWM tendrá su salida basándose en el *data sheet* del Atmega164PA. El comando para inicializar el módulo PWM es el siguiente:

PWM1_Init (_PWM1_Fast_Mode, _PWM1_Prescaler_8, _PWM1_NON_Inverted, variable);

PWMX_Init (): Inicializa el módulo PWM, puede ser PWM1 o PWM2.
Ej. PWM2_Init (_Pwm2_Fast_Mode, _PWM2_Prescaler_8, _PWM2_Non_Inverted);

1. Modo Onda: selecciona modo PWM, puede ser: Fase Correcta o PWM Rápido.
Ej. _PWM1_FAST_MODE, _PWM1_PHASE_CORRECT_MODE,

2. Prescaler: establece un divisor de frecuencia, puede tener valores de: 1, 8, 32, 64, 128, 256 y 1024.
Ej. _PWM2_Prescaler_8, _PWM2_Prescaler_128,

3: Inversor: invierte o no la onda PWM.
Ej. _PWM1_NON_INVERTED, _PWM1_INVERTED,

4. Variable: tamaño de la onda PWM, puede tomar valores de 0 a 255.

Configuración de hardware

Para el uso de los ADC se deberá utilizar un elemento que entregue una entrada analógica, en este caso un potenciómetro y para eso se deberá realizar las conexiones mediante cables tanto a la alimentación, como a los pines del módulo. Para ello se tiene entradas a VCC junto a 8 entradas al puerto A del microcontrolador que manejan los ADC.

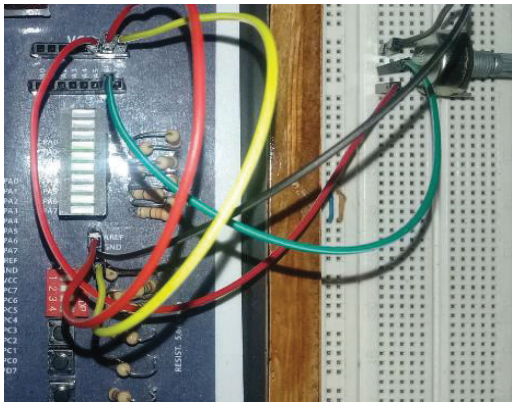


Figura 3.28 Conexión de un potenciómetro al módulo didáctico básico

Además, para poder utilizar los ADC se deben conectar o dar alimentación a estos dos pines que son muy importante para esta práctica.

AREF: entrada analógica al conversor analógico – digital.
AVCC: provee energía al conversor analógico – digital.

Para el ejercicio de las ondas PWM se deben realizar las conexiones al puerto D ya que estos pines tienen algunos de sus pines la función para producir estas ondas. De la misma manera tiene un espadín para hacer conexión a tierra.

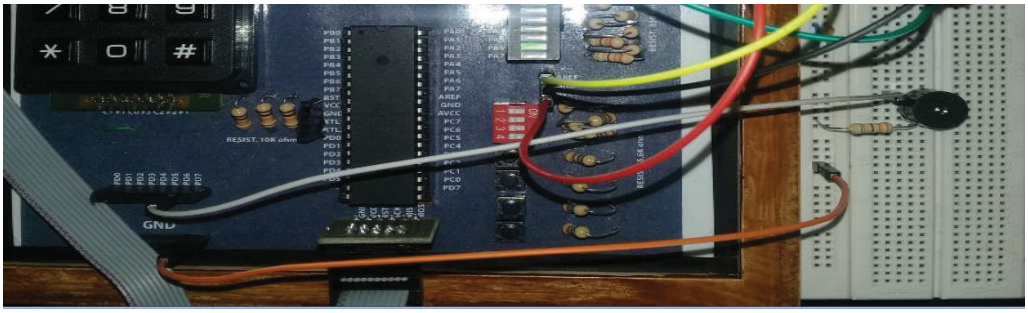
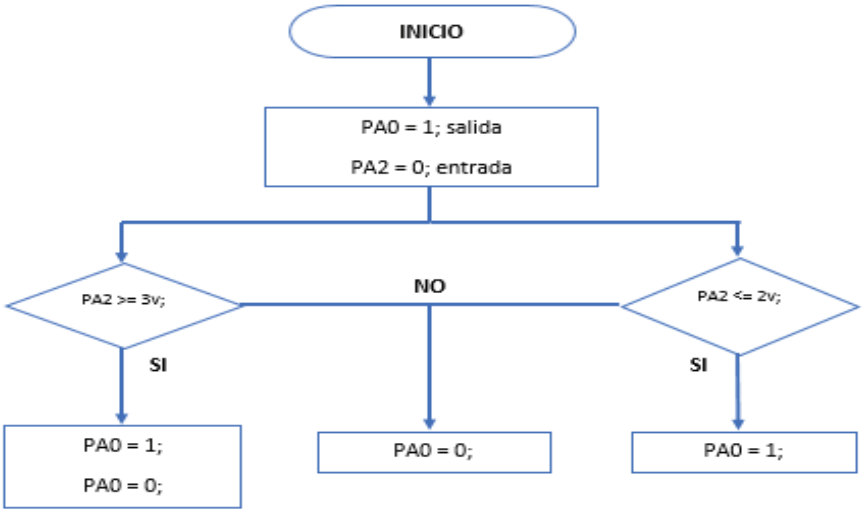


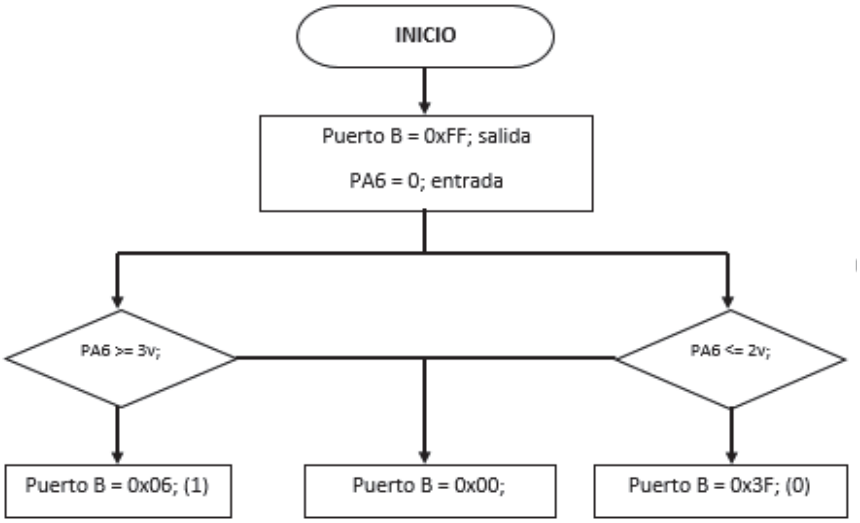
Figura 3.29 Conexión de un buzzer al módulo didáctico básico

Diagramas de flujo

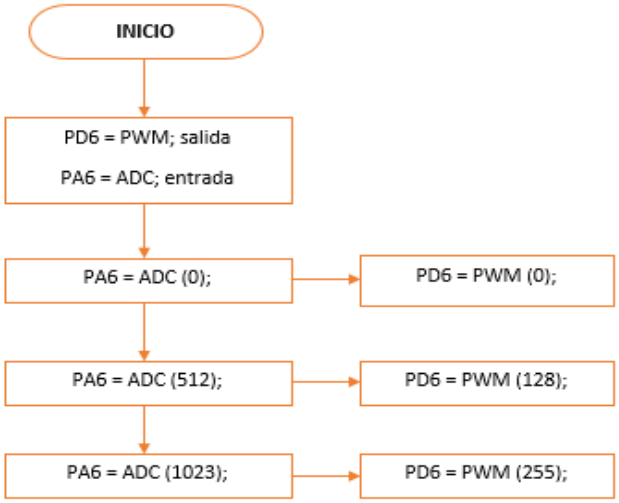
Opción 1



Opción 2



Opción 3



Simulaciones

Opción 1

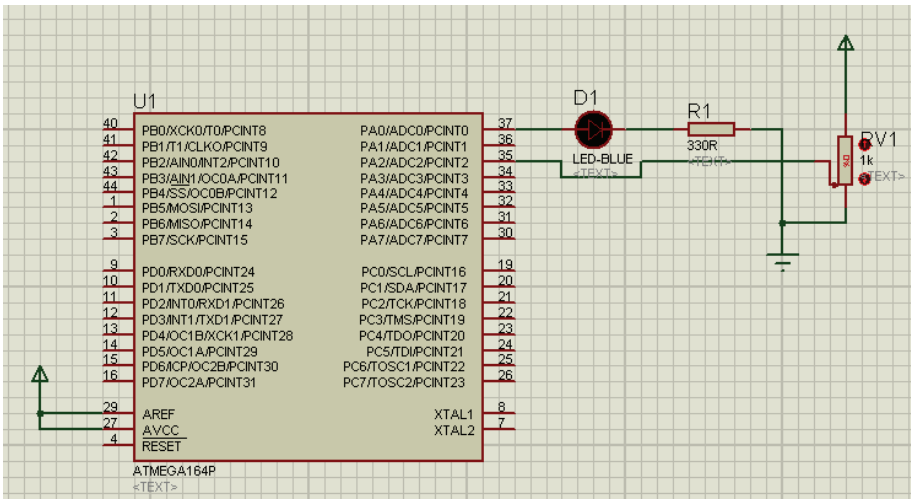


Figura 3.30 Simulación de la opción 1 de práctica 3

Opción 2

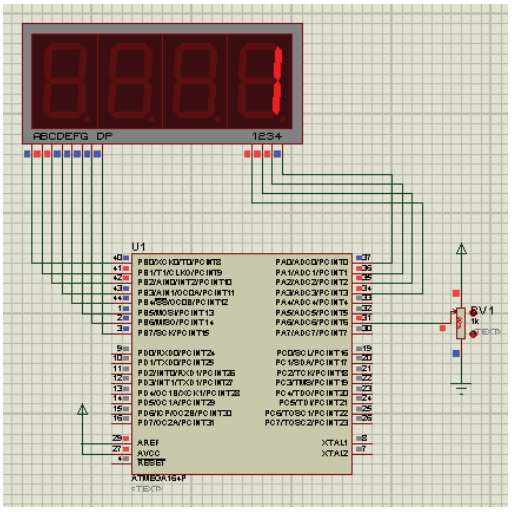


Figura 3.31 Simulación de la opción 2 de práctica 3

Opción 3

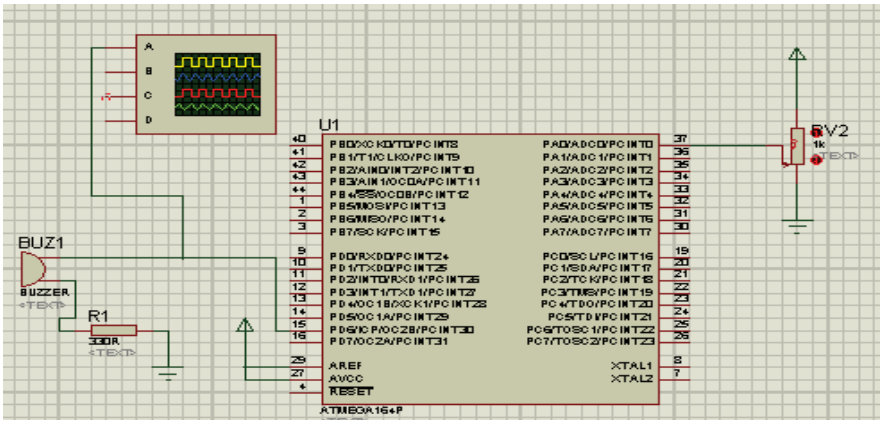


Figura 3.32 Simulación de la opción 3 de práctica 3

Ejemplo de programación.

Opción 1: Configurar el pin PA2 como entrada para usar su canal analógico y el pin PA0 como salida y crear las siguientes condiciones: cuando el valor de entrada entregada por el potenciómetro supere los 3 voltios el pin PA0 parpadee con un retardo de un segundo, mientras que si el valor de entrada es menor de 2 voltios el pin PA0 permanecerá en 1 lógico y si el valor de entrada esta entre los 2 y 3 voltios el pin PA0 permanecerá en 0 lógico.

```
int valor;                      //Variable de lectura
int a = 613;                    //Variable referencia para 3 voltios
int b = 409;                    //Variable referencia para 2 voltios
void main() {
    DDA0_bit = 1;               //Pin A0 configurado como salida
    DDA2_bit = 0;               //Pin A2 configurado como entrada
    ADC_Init();                 //Inicializar el Canal analogico
    do{
      valor = adc_read(2);      //Operacion de lectura del canal analogico 2
      if (valor > a){           //Condicion si lectura es mayor a 3 voltios
        porta.B0 = 0;           //A0 en 0 logico
        delay_ms(1000);         //retardo de 1 segundo
        porta.B0 = 1;           //A0 en 1 logico
        delay_ms(1000);         //retardo de 1 segundo
      }
      else {
        porta.B0 = 0;           //valor de salida si no cumplen las condiciones
      }
      if (valor < b){           //Condicion si lectura es menor a 2 voltios
        porta.B0 = 1;           //A0 en 1 logico
      }
    }while(1);
}
```

Figura 3.33 Código de programa de la opción 1 de práctica 3

Opción 2: Configurando el pin PA6 para usar su canal analógico y uno de los *displays* conectado al puerto B, mediante las mismas condiciones: cuando el valor de entrada entregada por el potenciómetro supere los 3 voltios en el *display* de 7 segmentos mostrará el número 1, si el valor de entrada es menor a 2 voltios en el *display* mostrará el número 0 y si el valor de entrada esta entre los 2 y los 3 voltios el *display* este en blanco.

```
int valor;                          //Variable de lectura
int a = 613;                        //Variable referencia para 3 voltios
int b = 409;                        //Variable referencia para 2 voltios
void main() {
     DDA0_bit = 1;                  //Pin A0 configurado como salida
     DDA1_bit = 1;
     DDA2_bit = 1;
     DDA3_bit = 1;
     DDA6_bit = 0;                  //Pin A6 configurado como entrada
     DDRB = 0xFF;                   //Puerto B configurado como salida
     ADC_Init();                    //Inicializar el Canal analogico
     do{
        valor = adc_read(6);        //Operacion de lectura del canal analogico 6
        if (valor > a){             //Condicion si lectura es mayor a 3 voltios
           porta.B0 = 0;            //Encendido del bit A0 para encendido de display
           porta.b1 = 1;
           porta.B2 = 1;
           porta.b3 = 1;
           portb = 0x06;            //Valor para visualizar numero 1
        }
        else {
           portb = 0x00;
        }
        if (valor < b){             //Condicion si lectura es menor a 2 voltios
           porta.b0 = 0;            //Encendido del bit A0 para encendido de display
           porta.b1 = 1;
           porta.B2 = 1;
           porta.b3 = 1;
           portb = 0x3F;            //Valor para visualizar numero 0
        }
     }while(1);
}
```

Figura 3.34 Código de programa de la opción 2 de práctica 3

Opción 3: Usando nuevamente el canal analógico 6, realizar el control de una onda PWM, la cual tendrá su salida a través de un *buzzer* conectado en el pin PD6. La amplitud de la onda deberá ser proporcional al valor de entrada del canal analógico.

```
char amplitud;                      //Variable de amplitud onda PWM

void main() {
     DDRA = 0x00;                   //Puerto A como entradas
     DDD6_bit = 1;                  //Pin D6 como salida onda PWM acorde datasheet
     Adc_Init();                    //Inicializado el ADC
     Pwm2_Init(_Pwm2_Fast_Mode,_Pwm2_Prescaler_8,_Pwm2_Non_Inverted, amplitud); //Inicializado modulo PWM

     do{
        amplitud = adc_read(0) / 4;     //Operacion transformar valores ADC a PWM
        Pwm2_Set_Duty(amplitud);        //Establece el valor de amplitud en funcion ADC
     }
     while(1);                      //Lazo infinito
}
```

Figura 3.35 Código de programa de la opción 3 de práctica 3

Práctica 4. Manejo de temporizadores

Configuración de software

- **Comando para la programación**

Para lo referente a temporizaciones se puede utilizar 3 *timers* que tiene el microcontrolador Atmega164PA: *timer0, timer1* y *timer2*. La función que se puede usar para llamar al *timer* es:

Void TimerXOverflow_ISR: Función que se realiza cuando se produce un desbordamiento del *timer* establecido.
Ej. Void Timer1_Overflow_ISR ()

SREG_I_BIT: Registro que habilita las interrupciones globales (internos y externos).
Ej. SREG_I_Bit = 1; Interrupciones Habilitadas

TOIEX_BIT: Habilita el *timer* establecido por desbordamiento.
Ej. TOIE1_Bit = 1; Habilita *Timer*1 por desbordamiento

TCCRXB: Pre-escaler que reduce una señal de alta frecuencia a una baja frecuencia.
Ej. TCCR1B = 2; Inicia el *timer*1 con un pre-escaler de 8.

Para visualizar diferentes caracteres en el arreglo de *displays*, es mejor compilar el programa en una frecuencia de 8MHz o 10MHz, ya que si se usa frecuencias más bajas la visualización no será comprensible en los *displays*, también los retardos en los que se realice la operación de presentar caracteres en los *displays* es recomendable que sean bajos casi en función de microsegundos.

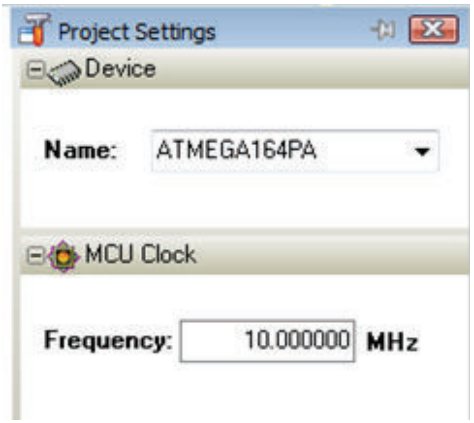


Figura 3.36 Selección de frecuencia de trabajo

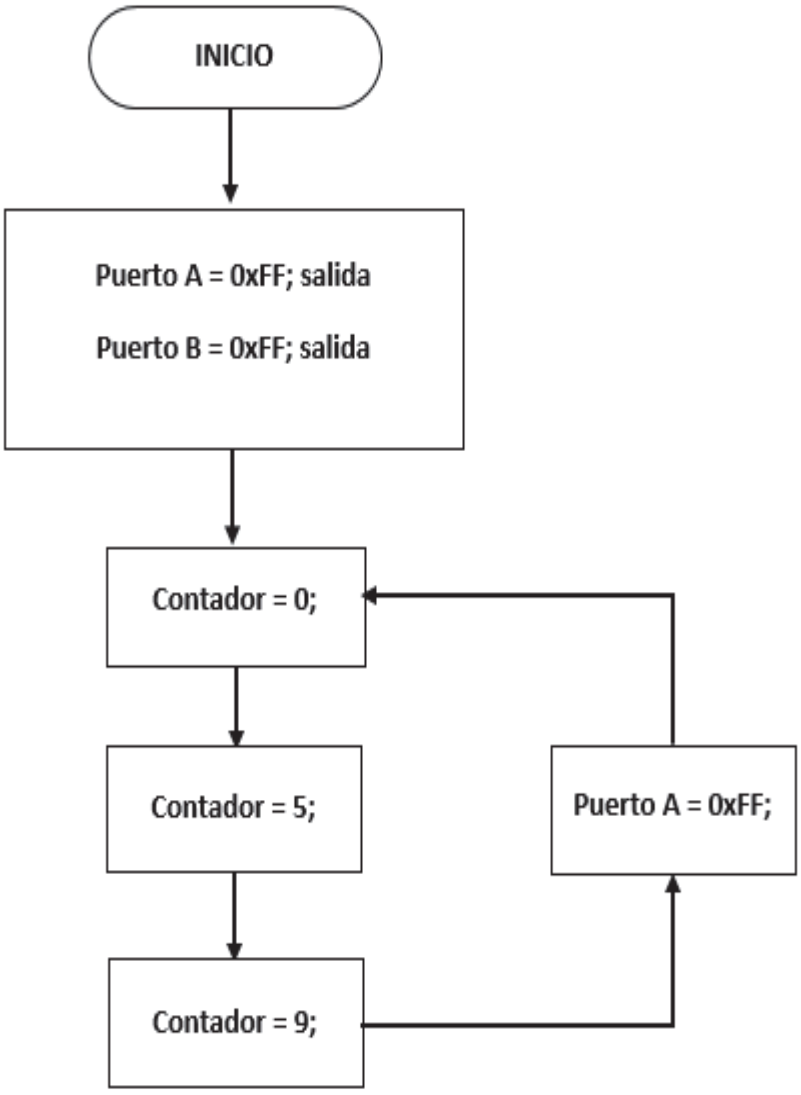
Una de las formas que se recomienda para realizar un contador es mediante la sentencia *for* y para visualizar los números también se recomienda hacer lo mismo que en la práctica del *keypad* donde se establecen los caracteres que observar.

Configuraciones de Hardware

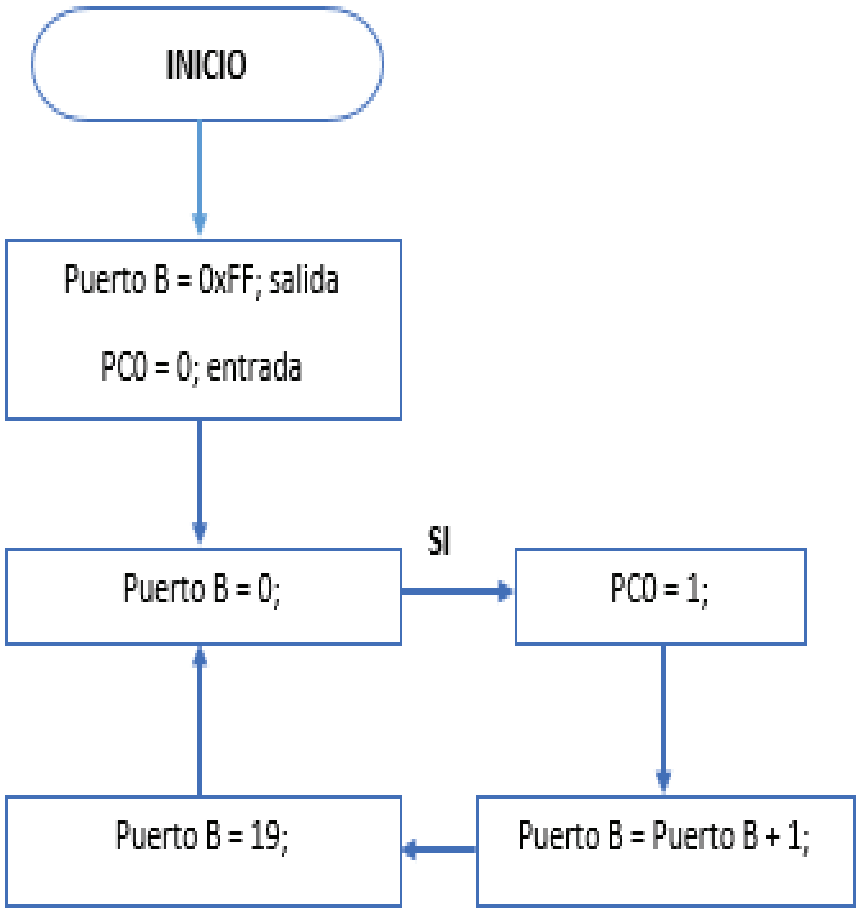
Recordar que para encender los *displays* los 4 primeros pines del puerto A deben estar configurados como salidas, como se indicó en una práctica previa de qué pin del puerto A controla el encendido de cada *display* de 7 segmentos.

Diagramas de flujo

Opción 1



Opción 2



Simulaciones

Opción 1

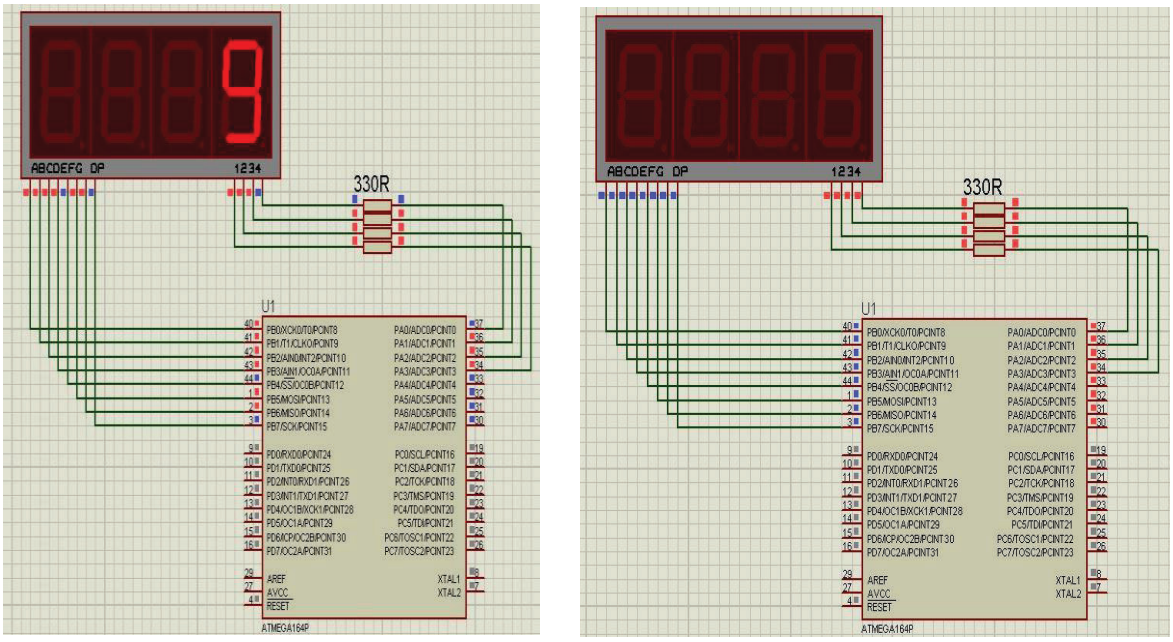


Figura 3.37 Simulación de la opción 1 de práctica 4

Opción 2

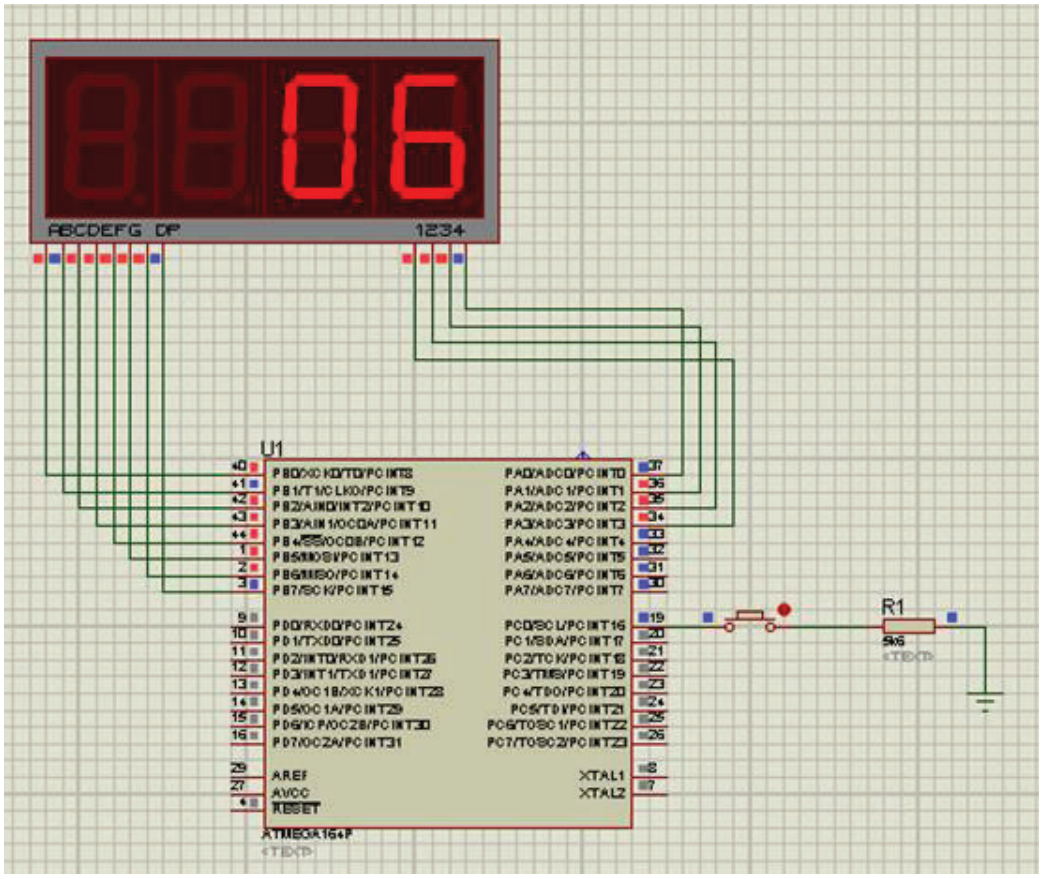


Figura 3.38 Simulación de la opción 2 de práctica 4

Ejemplo de programación

Opción 1: Escribir un programa para realizar un contador módulo 10 (0 – 9) en uno de los *displays* y cuando el contador llegue a 9 se active el *timer1* por desbordamiento, una vez activado el *timer1* los *displays* permanecerán apagados y los leds conectados al puerto A se encenderán por 1. Pasado ese segundo el *timer1* se desactivará y se repetirá el conteo.

```
int display[11] = {0x00,0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};     //# Decimales
char unidad = 0;                                                               // variable unidad

void Timer1Overflow_ISR() org IVT_ADDR_Timer1_OVF{                             // función de timer1
     if(unidad >= 10){                                                         // condicion si unidad llega a 10
       unidad = 0;                                                             // unidad vuelve a cero
       portb = 0x00;                                                           // displays apagado
       porta = 0xFF;                                                           // puerto A en 1
     }
}
```

```
void main(){
      DDRA = 0xFF;                    // Puerto A como entrada
      porta.B0 = 0;                   // Pin A0 valor inicial
      DDRB = 0xFF;                    // Puerto B como entrada
      portb = 0;                      // Puerto B valor inicial
      Sreg_I_bit = 1;                 // Habilitacion temporizador
      Toie1_bit = 1;                  // Habilitacion temporizaador por desbordamiento
      do{
      porta = 0x0E;                   // Valor encendido cuarto display
      portb = display[unidad];        // Visualiza el valor de la unidad
      delay_ms(1000);                 // Retardo de 1 segundo
      unidad++;                       // Incremento de unidad
      TCCR1B = 2;                     // Prescaler de 8 y arranca el timer1
      }
      while(1);                       // Lazo infinito
}
```

Figura 3.39 Código de programa de la opción 1 de práctica 4

Opción 2: Desarrollar un programa para generar un contador módulo 20 (0 – 19) el cual se visualizará en los *displays*. El contador deberá ir incrementándose mediante la activación del *timer1*. La activación del *timer1* la realizará al presionarse el pulsador PC0, caso contrario si no se presiona el pulsador, el contador se detendrá.

```
int display[16] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71};    //# Hexadecimales
char decena = 0;  //variable decena
char unidad = 0;  //variable unidad

void Timer1Overflow_ISR() org IVT_ADDR_Timer1_OVF{    //Funcion para usar Timer1
    unidad++;           //incremento de unidad
    TCCR1B = 0;         //apagado del timer1
}
```

```
void main(){
      DDRC = 0x00;          //puerto C como entradas
      PortC.B0 = 1;         //pin C0 pull - up
      DDRA = 0xFF;          //puerto A como salidas
      DDRB = 0xFF;          //puerto B como salidas
      Sreg_I_bit = 1;       //habilitacion de interrupcion por temporizador
      Toie1_bit = 1;        //habilitacion desbordamiento de temporizador
      do{
         porta = 0x0E;                    //configuracion para observar unidad
         portb = display[unidad];         //visualiza el valor de la unidad
         delay_ms(1);                     //retardo de 1 milisegundo
         porta = 0x0D;                    //configuracion para observar decena
         portb = display[decena];         //visualiza el valor de la decena
         delay_ms(1);                     //reatardo de 1 milisegundo
         if(unidad > 9){                  //condicion si unidad llega a 9
            unidad = 0;                   //unidad vuelve a 0
            decena++;                     //decena incrementa en 1
            if(decena > 1){               //condicion si decena mayor a 1
            decena = 0;                   //decena vuelve a 0
            }
         }
         if(Button(&PinC,0,1,0)){         //Pin C0 detecta 0 logico
         TCCR1B = 2;                      //Arranca el timer1 con prescaler de 8
         }
      }
      while(1);                           //Lazo infito
}
```

Figura 3.40 Código de programa de la opción 2 de práctica 4

Práctica 5. Manejo de interrupciones.

Configuración de software

- **Comandos para el desarrollo de la práctica**

Una de las mejores formas de producir una interrupción de manera externa es utilizando los registros INT del microcontrolador que son 3: ***INT0, INT1*** **e** ***INT2***. Estos registros interrumpen los procesos que realiza el microcontrolador, para hacer que funcionen se debe habilitar las interrupciones globales con el registro ***SREG_I_BIT*****.**

SREG_I_BIT: Habilita las interrupciones globales.
Ej. SREG_I_BIT = 1; Habilita las interrupciones globales.

INTX_BIT: Registro de interrupción.
Ej. INT1_BIT = 1; Habilita registro de interrupción INT1

Configuración de hardware

Si se quiere usar las banderas INT0, 1, se deben conectar interruptores o pulsadores en los pines donde el *data sheet* indican que están en el puerto D o por otra parte hacer que los pulsadores del puerto C accionen estas banderas.

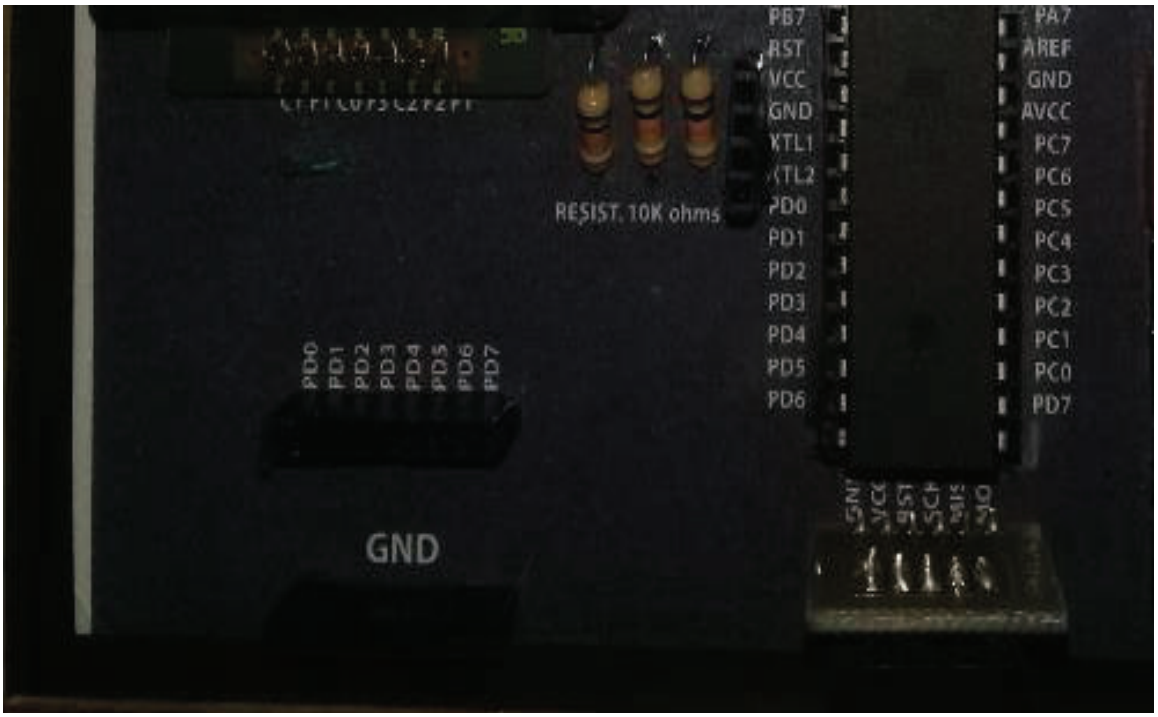
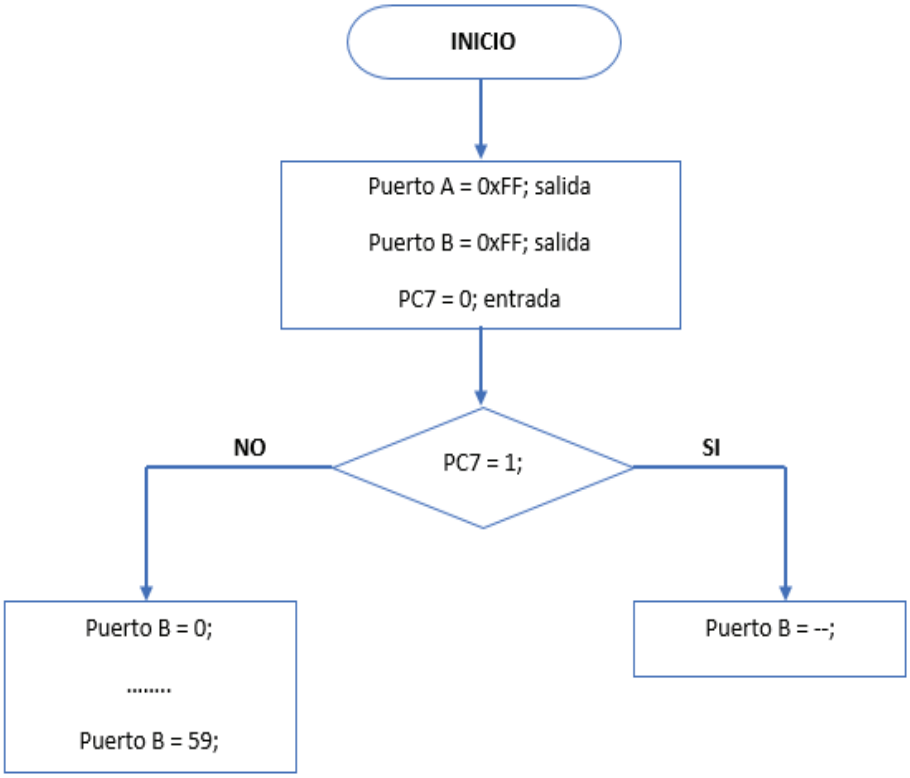


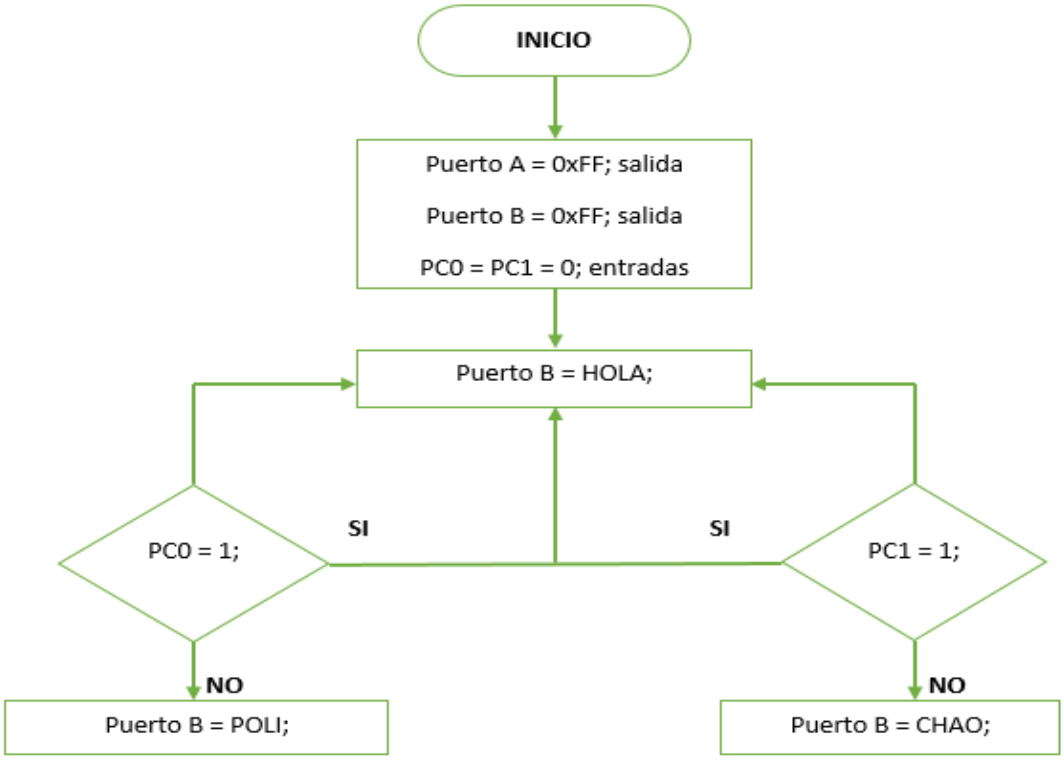
Figura 3.41 Regleta de conexión al puerto D

Diagramas de flujo

Opción 1



Opción 2



Simulaciones

Opción 1

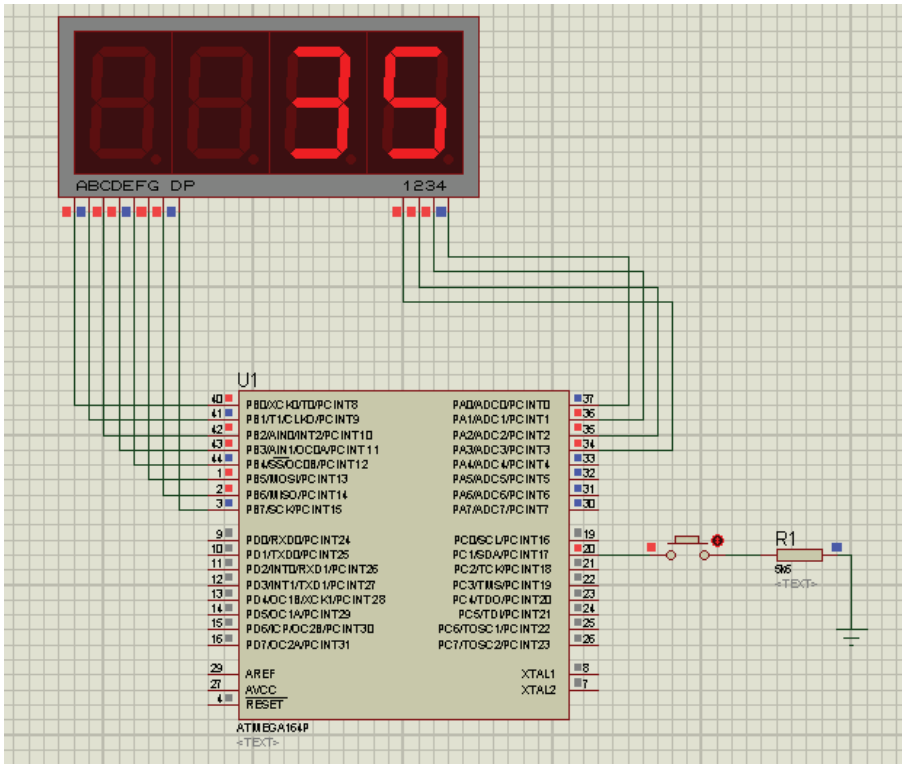


Figura 3.42 Simulación de la opción 1 de práctica 5

Opción 2

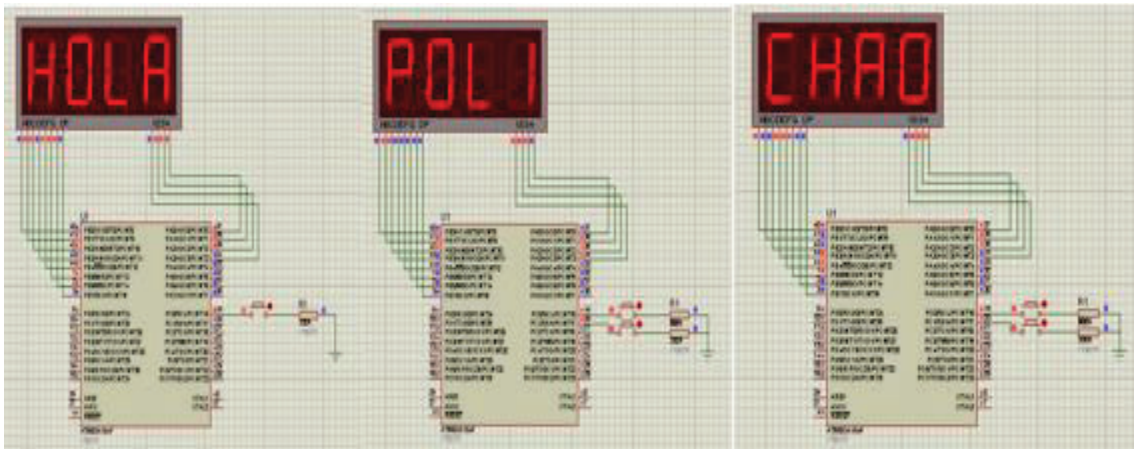


Figura 3.43 Simulación de la opción 2 de práctica 5

Ejemplo de programación

Opción 1: Escribir un programa que realice un contador módulo 60 (0 – 59) el cual se visualizará en dos de los *displays* conectados al puerto B y mediante la interrupción del pulsador PC1 se reseteará el conteo.

```
void main() {
    int display[16] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71};  //# Hexadecimales
    int cont1 = 0;              //variable unidad
    int cont2 = 0;              //variable decena
    int x;                      //variable sentencia for
    DDRA = 0xFF;                //puerto A como salida
    DDRB = 0xFF;                //puerto B como salida
    DDC7_bit = 0;               //Pin C1 puerto C como entrada
    portc.B7 = 1;               //Valor asignado al pin C1
    do{
        if (Button(&pinc,7,1,0)){  //Condicion al presionar pulsador PC1
          SREG_I_Bit = 1;          //Llamada a la funcion de Interrupcion
          INT1_Bit = 1;
          portb = 0x00;
          porta = 0x80;
          delay_ms(100);           //retardo de 100 milisegundos
        }
        else
        for (x = 0; x <= 60; x++){  //Sentencia for para contador modulo 60
        porta = 0x0E;               //configuracion visualizar la unidad
        portb = display[cont1];     //visualiza la unidad
        delay_us(900);              //retardo de 900 microsegundos
        porta = 0x0D;               //configuracion visualizar la decena
        portb = display[cont2];     //visualiza la decena
        delay_us(900);              //retardo de 900 microsegundos
        }
        cont1++;                    //incremento de la unidad en 1
        if (cont1 > 9){             //condicion si unidad llega a 9
          cont1 = 0;                //unidad vuelve a 0
          cont2++;                  //decena incrementa en 1
          if(cont2 > 5)             //condicion si decena llega a 5
          cont2 = 0;                //decena vuelve a 0
        }
        SREG_I_Bit = 0;
        INT1_Bit = 0;
    }
    while(1);                       //lazo infinito
}
```

Figura 3.44 Código de programa de la opción 1 de práctica 5

Opción 2: Escribir un programa que permita presentar en el arreglo de *displays*, aplicando el criterio de barrido la palabra “HOLA” de forma permanente. Mediante el pulsador PC0 hacer el llamado a la función interrupción que presentará la palabra “POLI” durante 2 segundos para posteriormente dejar los *displays* en blanco y vuelva a la función principal. Además, mediante el pulsador PC1 hacer el llamado a la función interrupción que presentara la palabra “CHAO” igualmente por 2 segundos y de la misma manera dejar los *displays* en blanco para que retorne a la función principal.

```
//Funcion de la interrupcion POLI
void Interrupcion(){
      while(l < 100)
      {
          porta = 0x07;
          portb = 0x73;                 // F
          delay_us(800);
          porta = 0x0b;
          portb = 0x3F;                 // O
          delay_us(800);
          porta = 0x0d;
          portb = 0x38;                 // L
          delay_us(800);
          porta = 0x0e;
          portb = 0x06;                 // I
          delay_us(800);
          l++;
      }
      if (l >= 100){                    // Condicion para terminar interrupcion
         l = 0;                         // variable vuelve a cero
         porta = 0x0F;                  // Apagado de los displays
      }
}
//Funcion de la Interrupcion CHAC
void Interrupcion1(){
      while(j < 100)
      {
          porta = 0x07;
          portb = 0x39;                 // C
          delay_us(800);
          porta = 0x0b;
          portb = 0x76;                 // H
          delay_us(800);
          porta = 0x0d;
          portb = 0x77;                 // A
          delay_us(800);
          porta = 0x0e;
          portb = 0x3F;                 // O
          delay_us(800);
          j++;
      }
      if (j >= 100){                    // Condicion para terminar interrupcion
         j = 0;                         // variable vuelve a cero
         porta = 0x0F;                  // Apagado de los displays
      }
}
void main() {
      DDC0_bit = 0;                     // Pin C0 como entrada
      DDC1_bit = 0;                     // Pin C1 como entrada
      DDRA = 0xFF;                      // Puerto A como salida
      DDRB = 0xFF;                      // Puerto B como salida
      portc.B0 = 1;                     // valor del Pin C0
      portc.B1 = 1;
      while(1){
          if(Button(&pinc,0,1,0)){      // condicion cuando presione pulsador
              Interrupcion();           // produce la funcion de interrupcion
              delay_ms(100);            // retardo para volver a funcion principal
          }
          else
          porta = 0x07;
          portb = 0x76;                 // H
          delay_us(800);
          porta = 0x0b;
          portb = 0x3F;                 // O
          delay_us(800);
          porta = 0x0d;
          portb = 0x38;                 // L
          delay_us(800);
          porta = 0x0e;
          portb = 0x77;                 // A
          delay_us(800);
          if(Button(&pinc,1,1,0)){      // condicion cuando presione pulsador
              Interrupcion1();          // produce la funcion de interrupcion
              delay_ms(100);            // retardo para volver a funcion principal
          }
      }
}
```

Figura 3.45 Código de programa de la opción 2 de práctica 5

3.5 Pruebas de funcionamiento de los módulos didácticos básicos

Práctica 1. Manejo de puertos entrada y salida

En la figura 3.47, la práctica que se muestra es el desplazamiento de un *bit* por todo el puerto A, el cual se irá visualizando en el banco de leds y mediante los pulsadores o interruptores este proceso se invertirá o realizará acciones como encender todos los leds o apagarlos.

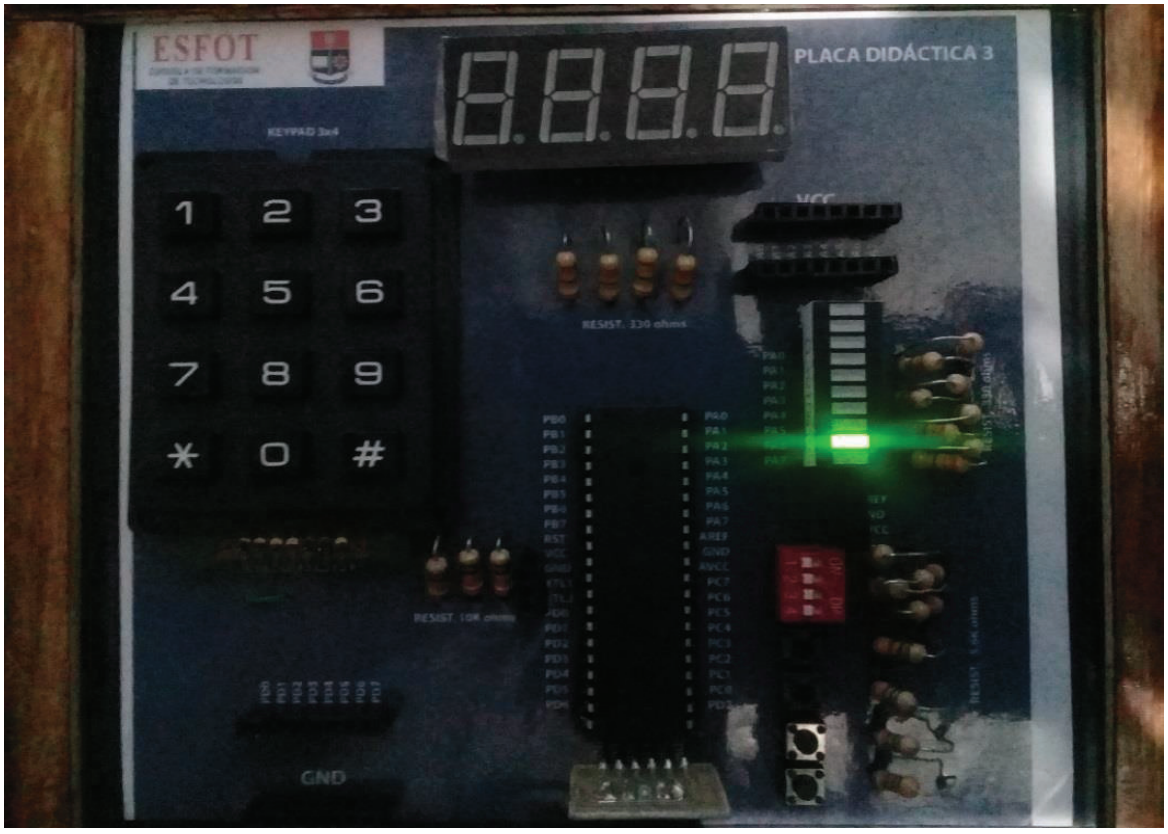


Figura 3.46 Prueba de funcionamiento práctica 1

Recomendaciones durante la práctica

- Los retardos durante el desplazamiento pueden ser altos, esto se puede solucionar colocando una frecuencia baja (1MHz) al momento de compilar el programa en el MikroC.
- El programa original desplaza el bit desde PA0 a PA7, si el proceso es invertido durante la transición de PA7 a PA0, este no hará ningún cambio ya que no hay ningún bit encendido. La solución es restablecer la condición a la original y resetear el microcontrolador.

Práctica 2. Periféricos de entrada

En la figura 3.48, la práctica consiste en que el microcontrolador lea el dato ingresado por el teclado matricial conectado al puerto D y que dicho dato sea visualizado en uno de los *displays* de 7 segmentos conectado al puerto B. Además, mientras no se esté ingresando ningún dato el *display* debe permanecer apagado.

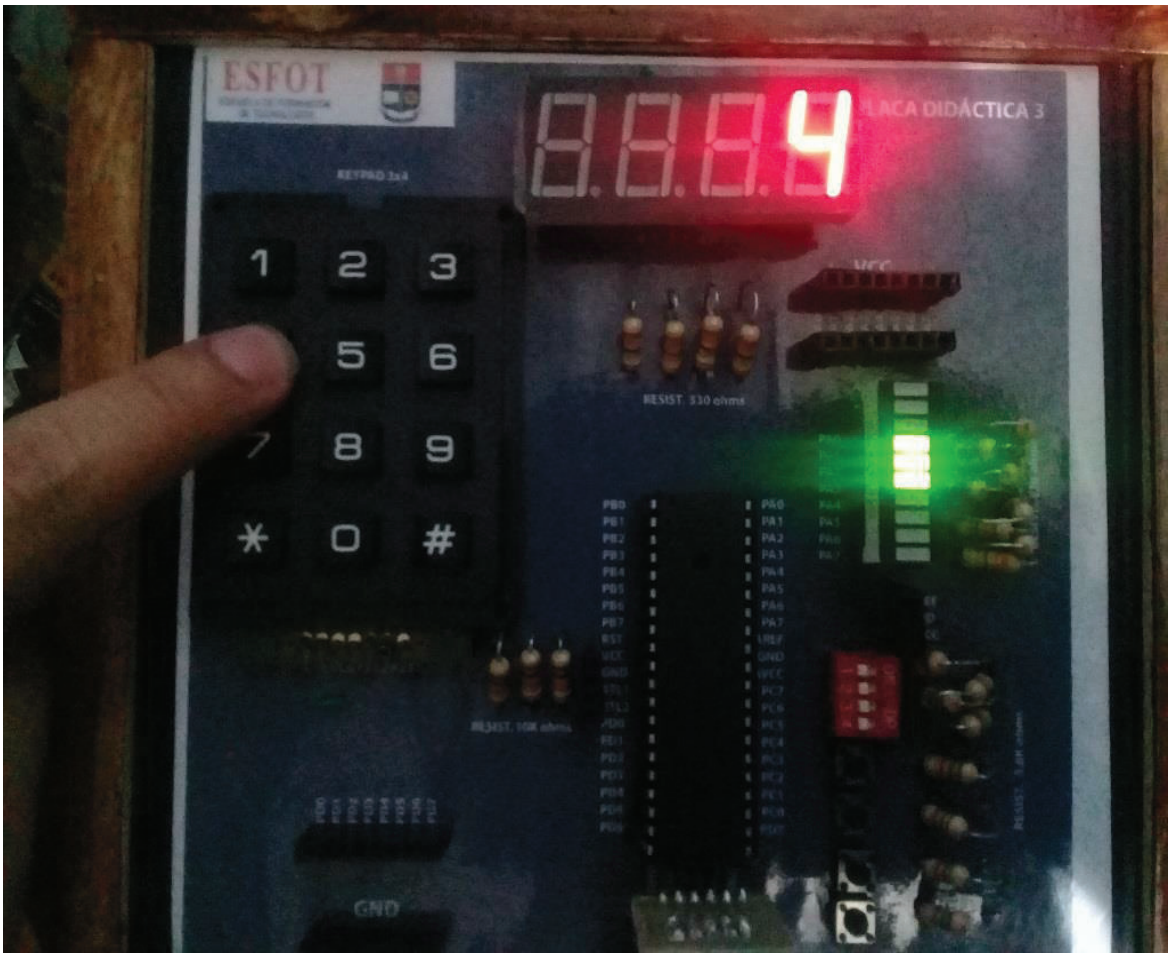


Figura 3.47 Prueba de funcionamiento práctica 2

Recomendaciones durante la práctica

- Al momento de pulsar una de las teclas podría aparecer otro número, esto puede ser debido a que los puntos de soldadura estén haciendo un corto entre ellos. La solución es simplemente revisar si hay alguna impureza que este ocasionando el corto y limpiarlo.
- Para activar los *displays* se deben configurar los cuatro primeros *bits* del puerto A (A0 – A3), ya que los *displays* instalados son de cátodo común estos se encienden configurando con 0 y se apagan con 1. Como se muestra en la figura el cuarto *display* se activa con el pin PA0.

Práctica 3. Convertidor analógico – digital y señales PWM

En la figura 3.49, este ejercicio consiste en conectar un potenciómetro y dependiendo la lectura de este en el *display* se visualizará 1 o 0. Mientras que en la figura 3.50, el ejercicio consiste en generar una onda PWM la cual su magnitud será controlada a través del potenciómetro y la intensidad de la onda será manifestada a través del *buzzer.*

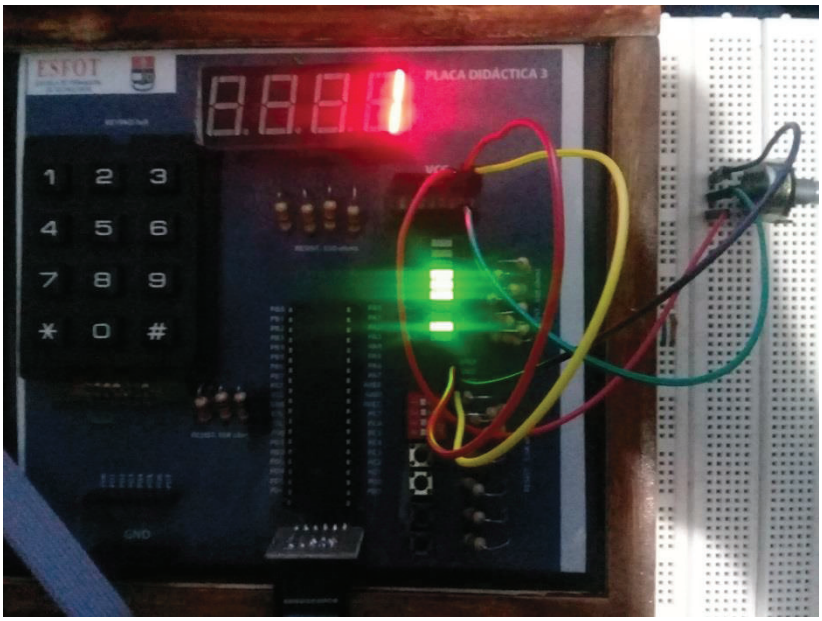


Figura 3.48 Prueba de funcionamiento práctica 3 opción 2

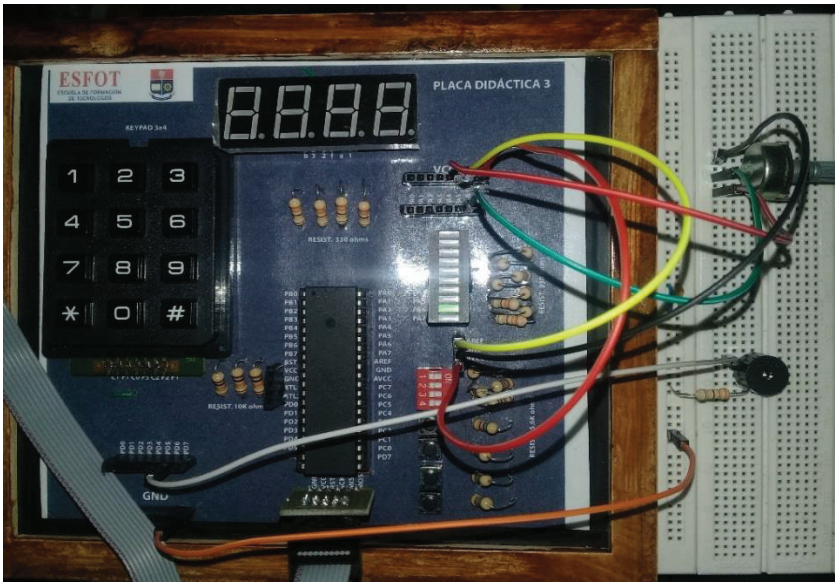


Figura 3.49 Prueba de funcionamiento práctica 3 opción 3

Recomendaciones durante la práctica

- Para usar correctamente el potenciómetro los pines AREF y AVCC deben conectarse a VCC.

Práctica 4. Manejo de Temporizadores

En la figura 3.51, la práctica que se muestra es un contador módulo 20 (0 – 19). El conteo se visualiza en los *displays*. Este contador irá incrementando activando el *timer1*, a través de un pulsador del puerto C (PC0). Caso contrario si no se usa el pulsador el contador se detendrá.

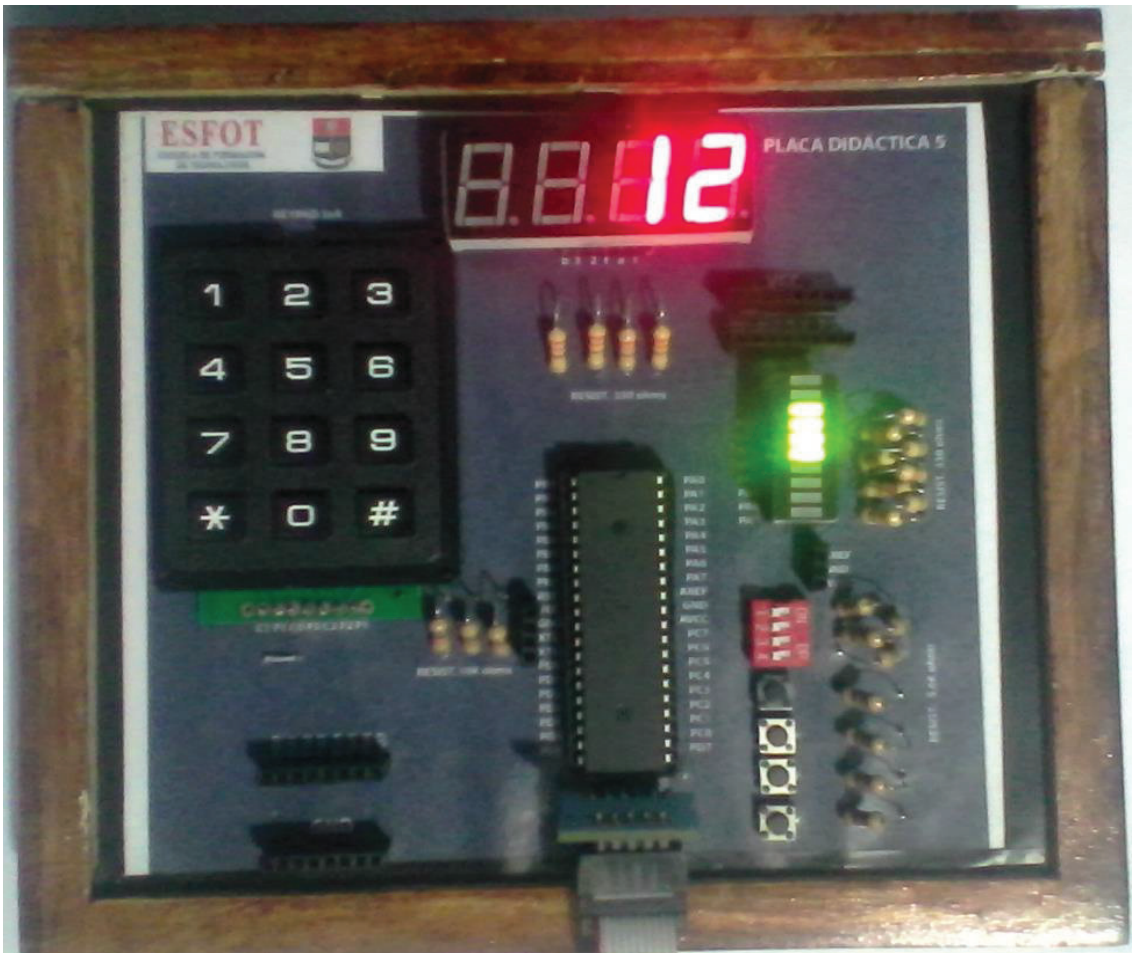


Figura 3.50 Prueba de funcionamiento práctica 4 opción 2

Recomendaciones durante la práctica

- Al momento de compilar el programa en el MikroC, se recomienda que la frecuencia sea alta como por ejemplo de 8MHz, ya que si el programa se lo compila a baja frecuencia al momento de grabarlo en el microcontrolador no se encenderán correctamente los segmentos del *display*.
- Recordar que para utilizar los *displays* se debe configurar los cuatro primeros *bits* del puerto A, en la figura 3.51 en el programa se están usando los cuatro *displays* pero por código de programa lo mostrado en el primer y segundo *display* es ningún segmento encendido.

Práctica 5. Manejo de interrupciones.

El ejercicio de esta práctica, el programa principal los *displays* de 7 segmentos muestran la palabra “HOLA” como en la figura 3.52. Al presionar un pulsador el programa principal se interrumpe y la palabra mostrada cambia por “POLI” como se ve en la figura 3.53, esta nueva palabra se mostrada solo por unos segundo para luego volver al programa principal.

Figura 3.51 Prueba de funcionamiento práctica 5 opción 2 “HOLA”

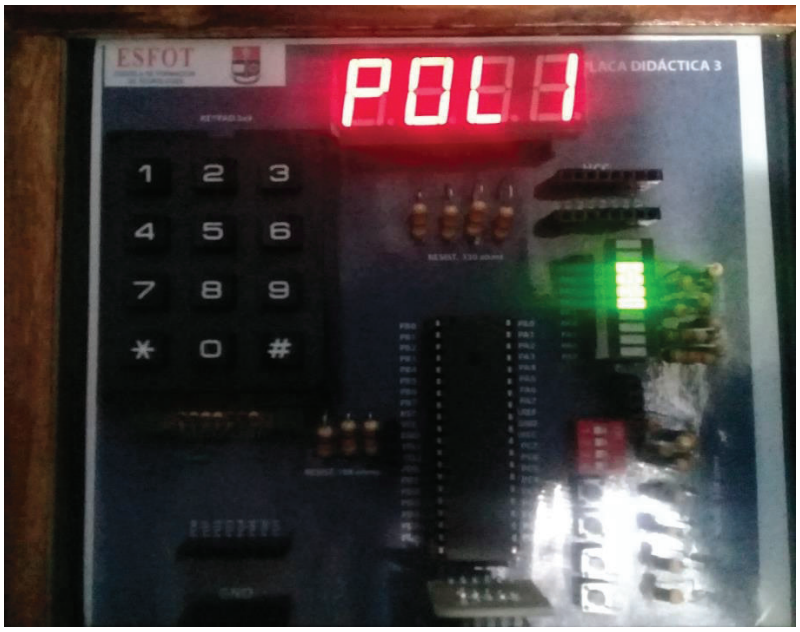


Figura 3.52 Prueba de funcionamiento práctica 5 opción 2 “POLI”

Recomendaciones durante la práctica

- Al igual que en la práctica 4 compilar el programa con una frecuencia alta (8MHz).

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Para el diseño y construcción se aplicaron las técnicas de la materia “Taller de circuitos impresos” recibidos en los primeros semestres de la carrera: como usar el programa *ARES*, plasmar un diseño en placa y la soldadura. Con este diseño propuesto se pueden aplicar los temas de estudio y además, con la arquitectura del módulo se pueden conectar más elementos a libertad del usuario. La implementación de estos módulos renueva al laboratorio de microprocesadores y con ello se pueden aplicar para proyectos de módulos didácticos más avanzados.

- Los modelos de hojas guías de estudiantes fueron hechas en referencias a hojas de prácticas que son utilizadas en diversos laboratorios que contienen los objetivos de la práctica y los ejercicios a realizar por parte de los estudiantes. Las hojas de instructores y las pruebas de funcionamiento contienen información para ayudar a los estudiantes a manejar a los módulos didácticos.

- Al momento de cargar, leer o borrar un código de programa, puede aparecer el mensaje en la ventana *“Chip Enable Program Error”*. Este mensaje quiere decir que la conexión entre el microcontrolador y el programador está fallando. La solución es simplemente revisar que ambos estén correctamente conectados al módulo; las indicaciones de los pines del microcontrolador están marcados en la placa y el programador tiene sus indicaciones en su placa de conexión.

- Por una mala configuración de los fusibles al momento de programar, el microcontrolador puede bloquearse. Una solución sería aplicar una señal de reloj al pin XTAL1, la frecuencia de dicha señal tiene que ser cuatro veces mayor a la frecuencia de programación.

- Se añadió un circuito de *reset* que no formaba parte del diseño original, debido a que al momento de aplicar algunos de los programas, por ejemplo en la práctica de manejo de puertos en momentos de transición el microcontrolador no seguía realizando su proceso.

- En la práctica de uso de periféricos, cuyo objetivo es demostrar el uso del teclado matricial o *keypad*, el programa MikroC ofrece una ayuda únicamente para inicializar el módulo *keypad* y que el microcontrolador sepa que tiene conectado un teclado. Por tanto se debe continuar escribiendo los comandos para que el microcontrolador lea la matriz del teclado y acepte los datos que se le ingresan.

- En la práctica de los canales ADC y señales PWM, para que el microcontrolador acepte los datos analógicos provenientes del potenciómetro se deben conectar los pines AREF y AVCC a la fuente, en este caso a VCC. Ya que el pin AREF sirve para la entrada analógica al conversor analógico – digital y el pin AVCC provee energía al conversor analógico – digital.

- Tanto en las prácticas de temporizaciones e interrupciones, para usar todos los *displays* simultáneamente lo mejor es que la frecuencia de trabajo del programa al compilar sea la misma o mayor a la frecuencia de programación del microcontrolador ATmega. Caso contrario, el barrido de los *displays* no se visualizará con claridad o también los segmentos de los *displays* apenas se encenderán.

4.2 Recomendaciones

- Tener precaución al momento de manipular ciertos elementos, un buen ejemplo es el teclado matricial o *keypad*, este es el elemento más frágil y puede terminarse desoldándose de la placa por una mala manipulación.

- Al momento de utilizar el *keypad* considerar revisar que en sus puntos de soldadura no existan ningún tipo de impurezas entre sus puntos de soldadura ya que puede producir un corto entre ellas y mostrar errores al momento de usarlo.

- Al momento de programar el módulo didáctico procurar no tener ningún elemento extra conectado, ya que el consumo de corriente de algún otro elemento puede impedir que el programador cargue los programas fuente en el microcontrolador.

- De preferencia siempre borrar la memoria *flash* del microcontrolador antes de grabar otro programa fuente, ya que esto puede provocar un mensaje de error en el Progisp. Tal vez no afecte al microcontrolador de manera inmediata, pero podría provocar un mal funcionamiento en un futuro.

5. Referencias Bibliográficas

[1] ATMEL, «Datasheet Atmega164P - Atmega324P - Atmega644P,» [En línea]. Available: https://www.mcselec.com/atmelpdf/atmega164p_324p_644p.pdf.

[2] Microchip, «ATMEGA164PA,» [En línea]. Available: http://www.microchip.com/wwwproducts/en/ATmega164PA.

[3] Atmel Corporation, «Atmega 164PA Datasheet Summary,» [En línea]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42713-ATmega164PA_Datasheet_Summary.pdf.

[4] N. Software, «Niple Blog,» Microchip, [En línea]. Available: http://www.niplesoft.net/blog/2016/01/15/puertos-del-microcontrolador/. [Último acceso: 15 Enero 2016].

[5] «HETPRO,» 15 Abril 2014. [En línea]. Available: https://hetpro-store.com/TUTORIALES/pwm/.

[6] RACSO, «Arduino UTFSM,» 21 Mayo 2014. [En línea]. Available: http://www.arduino.utfsm.cl/modulacion-por-ancho-de-pulso-pwm/.

[7] E. Crespo, «Aprendiendo Arduino,» [En línea]. Available: https://aprendiendoarduino.wordpress.com/tag/conversor-analogico-digital/.

[8] Microcontroladores, «Programación de Microcontroladores PIC, AVR, Arduino,» [En línea]. Available: http://microcontroladores-mrelberni.com/convertidor-analogico-digital-avr/.

[9] Microcontroladores, «Programación de Microcontroladores PIC, AVR, ARDUINO,» [En línea]. Available: http://microcontroladores-mrelberni.com/timer1-avr-temporizador-contador/.

[10] Microcontroladores, «Programación de Microcontroladores PIC, AVR, ARDUINO,» [En línea]. Available: http://microcontroladores-mrelberni.com/timer0-avr/.

[11] MrElberni, «Electricidad y Electrónica,» 7 Octubre 2015. [En línea]. Available: http://mrelberni.blogspot.com/2015/10/interrupcion-externa-avr.html.

[12] Escuela Politécnica Nacional, FACULTAD DE INGENIERIA ELÉCTRICA Y ELECTRÓNICA, 2018. [En línea]. Available: http://ciecfie.epn.edu.ec/wss/VirtualDirectories/80/CControlC/laboratorios/microprocesados/HojasGuias/Lab_Sistemas_MIcroprocesados_Practica4_2018B.pdf.

[13] J. Salas, «TodoElectrodo,» 12 Octubre 2012. [En línea]. Available: http://todoelectrodo.blogspot.com/2012/10/introducion-crear-un-componente-en-isis.html.

[14] H. M. –. B. d. Electrónica, «CIFPNº1 – Desarrollo de Productos Electrónicos,» 2012 - 2013. [En línea]. Available: https://cifpn1hectorm.wordpress.com/2013/02/07/creacion-de-componentes-en-ares/.

[15] O. Gallardo Puertas, *Fabricación de placas de circuito impreso con Proteus,* Valladolid, 2015.

[16] «Corporativo Quimico Global S.A,» 11 Abril 2011. [En línea]. Available: https://quimicoglobal.mx/cloruro-ferrico/.

[17] E. I. y. D. Electrónico, «Programador USBasp Atmel AVR 51 Atmega Attiny 3.3v / 5v,» [En línea]. Available: https://electronilab.co/tienda/programador-usbasp-atmel-avr-51-atmega-attiny-3-3v-5v/.

ANEXO A. Esquema Lógico

Creación de nuevos símbolos en ISIS

Siempre se puede acudir a la librería del programa y encontrarse que el elemento que se necesita utilizar no este, ni ningún otro elemento que realice funciones diferentes con el mismo encapsulado. En este caso el microcontrolador que se requiere para el módulo didáctico es un ATmega164PA y el programa ISIS tiene un microcontrolador ATmega164P y además la cantidad de pines es muy diferente, el uno cuenta con 40 pines y el otro con 44 pines respectivamente.

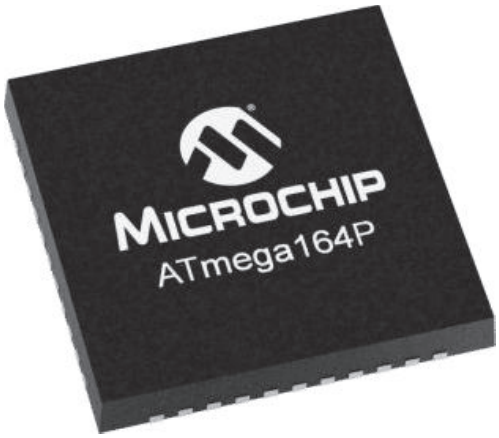


Ilustración A.1 Microcontrolador ATmega164P versión antigua

Como se observa en la ilustración A.1, la forma es totalmente diferente a como es el microcontrolador hoy en día. Como se mencionó la actual cuenta con solo 40 pines y su forma es más rectangular que cuadrática como se muestra en la ilustración A.2.

Ilustración A.2 Microcontrolador ATmega164PA versión actual

Lo primero es crear su símbolo mediante las herramientas de diseño gráfico, situadas en la parte izquierda. Luego de eso se podrá editar lo que se crea conveniente como: tipo de línea, anchura, color, etc. Lo siguiente fue colocar todos los pines que se van a utilizar. Para crear estos pines hubo que hacer el uso de la herramienta "*Device Pins Mode*" que se encuentra en la barra de la izquierda (ilustración A.3) y se colocaron en donde se creyó necesario y también se roto alguno de ellos, simplemente usando la opción "*rotate*".

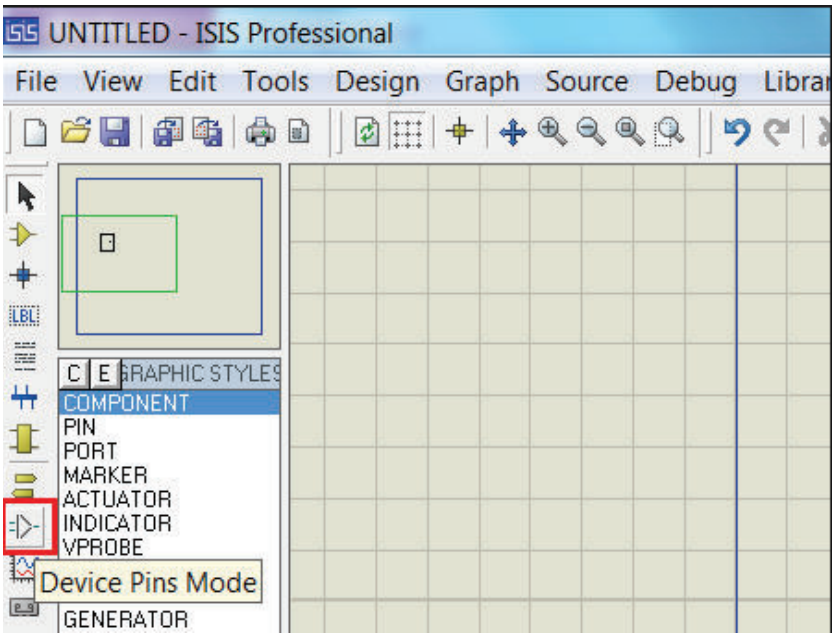


Ilustración A.3 Device pins mode - creación pines del símbolo

Una vez todos los pines colocados, para configurar sus propiedades solo se hace doble clic y en ellos se pueden: dar nombres, número de pin y si dicho pin va ser de entrada, salida, alimentación, etc. También se puede indicar si se desea ver el número del pin o su nombre, como se observa en la ilustración A.4.

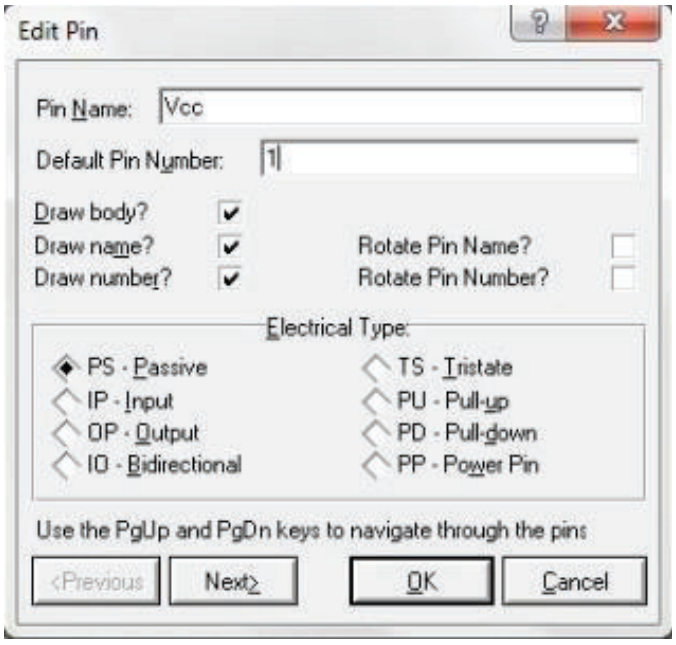


Ilustración A.4 Ventana de configuración de pines

Cuando se haya terminado de configurar la creación del nuevo símbolo, se selecciona todo el símbolo y haciendo clic derecho en la opción "*Make Device*", aparece una ventana en la cual se procede a darle un nombre al elemento (ilustración A.5). Dado que aún no

tiene encapsulado se puede omitir el paso para añadirlo. Por último, se guarda el nuevo dispositivo electrónico en una librería y se le asigna una categoría, sub-categoría, descripción, código o alguna nota.

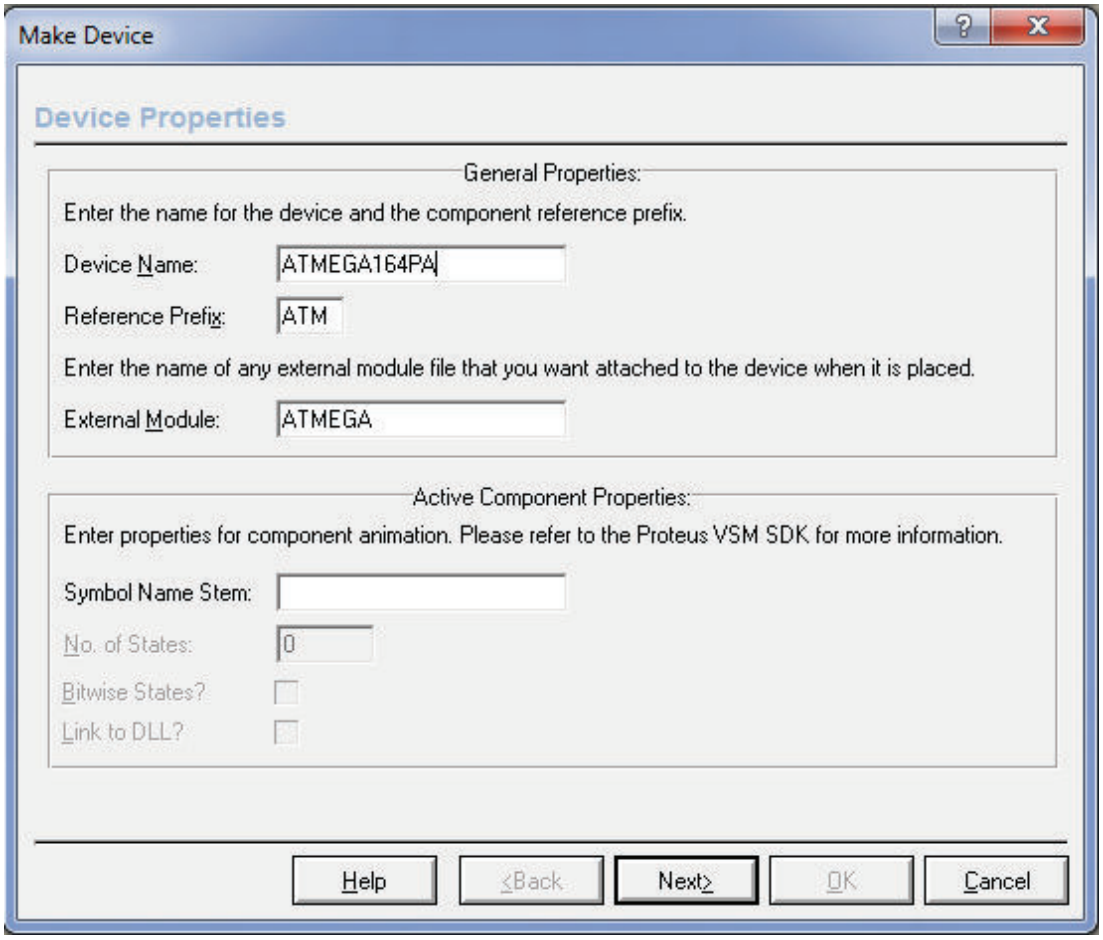


Ilustración A.5 Configuración del nuevo símbolo

Al finalizar se pulsa "*OK*" y ya estará guardado y lo siguiente será crear su encapsulado para posteriormente asignarlo. Como antes se mencionó algunos de los elementos que conforman el módulo didáctico básico tuvieron que ser creados, mientras tanto otros solo se tuvo que crear sus encapsulados, los elementos creados en la plataforma ISIS fueron:

- Microcontrolador ATmega164PA.
- Programador USBasp.
- Banco de 10 leds.

En la ilustración A.6 se puede observar el diseño esquemático, el cual fue transferido a la herramienta ARES del Proteus, para diseñar: el tamaño del módulo, localización de cada uno de los elementos que lo conforman, diagrama de pistas de conexión y los puntos de soldadura de los elementos.

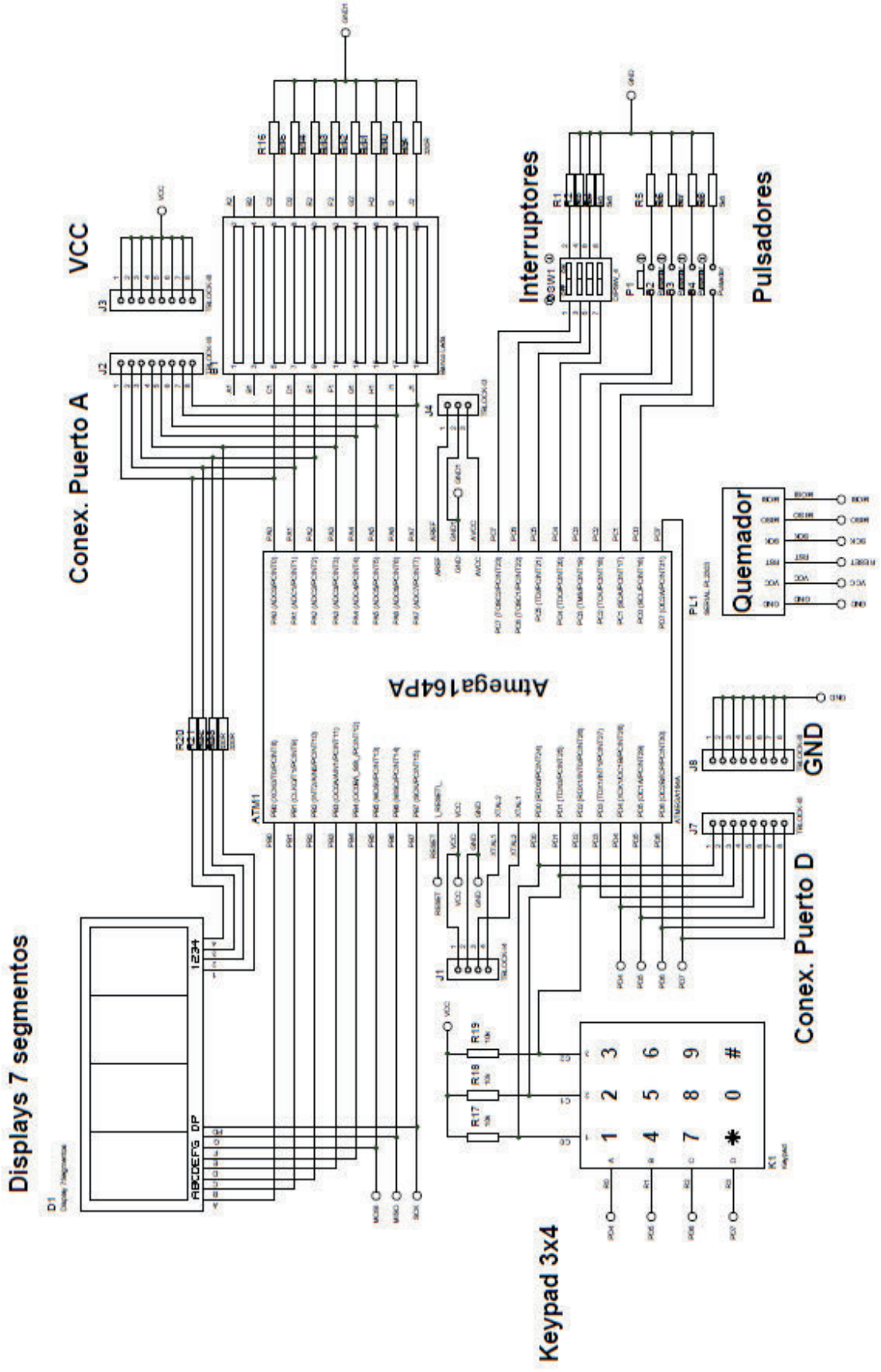


Ilustración A.6 Diseño esquemático del módulo didáctico básico

ANEXO B. Esquema Físico

Creación de encapsulados y paquetes PCB

Para la creación de los encapsulados lo importante es tener información como: dimensiones (ancho y largo), separación entre sus pines y si se desea una representación visual del elemento. Una vez diseñado el nuevo elemento con todos estos detalles, a más de eso se debe darle nombre a cada pin para facilitar las conexiones al momento de hacer el enrutado.

Al igual que en ISIS, las herramientas de diseño se ubican en la parte izquierda de la ventana. Luego de dibujar el contorno del elemento se pueden añadir los pines mediante la herramienta "*DIL Pad Mode*", los cuales serán los puntos de soldadura. Se recomienda hacerlos de un tamaño algo mayor y se colocarán de forma que cumpla con la distancia que se desea tener entre pines o terminales.

Una vez terminado el dibujo del encapsulado, lo siguiente será asignar a cada terminal el número de pin que le corresponde. Para ello se hace doble clic sobre cada uno de ellos y se introduce el número que corresponda. Además, mediante la herramienta "*2D Graphics Text Mode*" se le puede poner un texto que sería el nombre del elemento para identificarlo. En la ilustración B.1 se muestran los encapsulados tanto del microcontrolador ATmega164PA y el programador USBasp.

Ilustración B.1 Encapsulado ATmega164PA y USBasp

Ya con todas las especificaciones, lo siguiente es convertir el encapsulado en un paquete PCB. Se selecciona todo el encapsulado y se hace clic derecho en la opción "*Make Packages*". Aparecerá una ventana (ilustración B.2) donde se le darán algunas propiedades como: nombre del encapsulado, la librería a la que será asignado y la categoría donde se va a introducir. Además, se puede añadir una pequeña descripción.

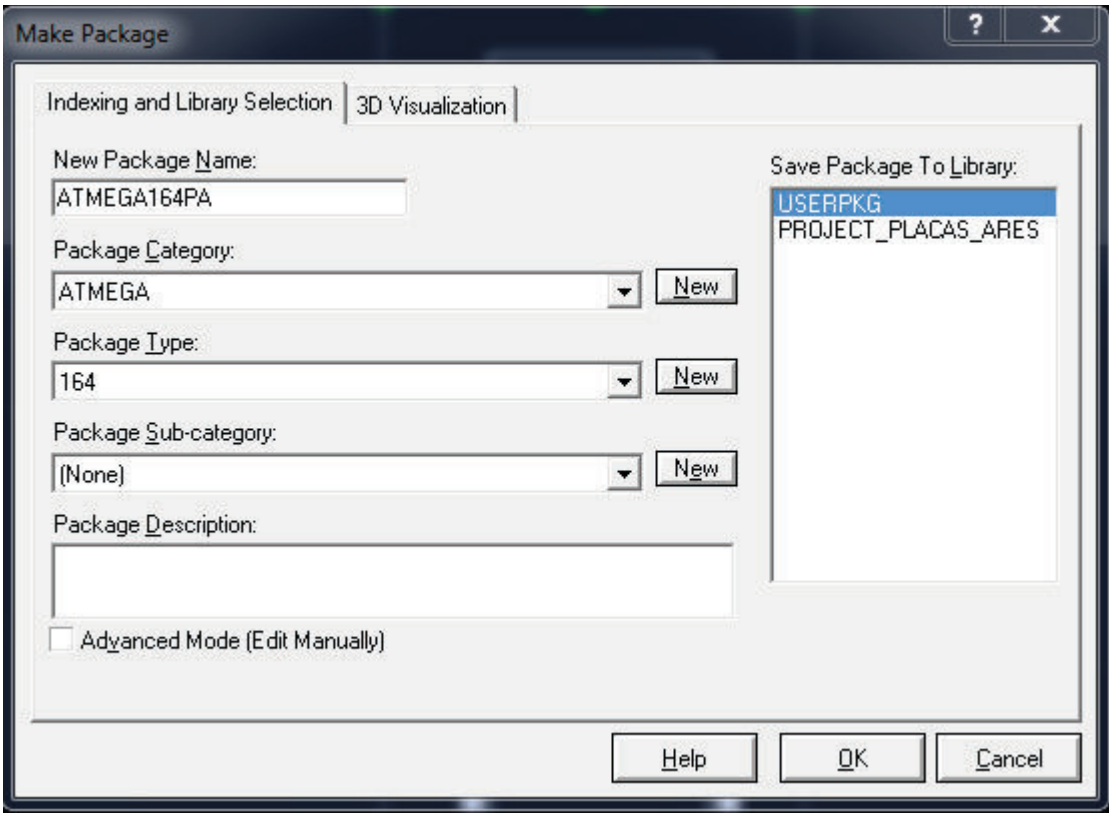


Ilustración B.2 Ventana de selección de librería y categoría

Una vez guardado, lo siguiente será enlazarlo con el diseño creado en el ISIS. Abriendo nuevamente la ventana "*Make Device*" y en la opción del encapsulado se busca el que se acabó de crear. Después hay que fijarse si los pines están situados correctamente y si no hay que arreglarlo. La imagen del enlazado se puede observar en el capítulo 3 en la sección de construcción de los módulos didácticos en la figura 3.6.

Los elementos que se tuvieron que crear encapsulados y luego su paquete PCB fueron:

- Microcontrolador ATmega164PA.
- Programador USBasp.
- Arreglo de *display* de 7 segmentos.
- Teclado matricial o *keypad.*

- *Dipswitch*.
- Pulsadores.
- Regletas.
- Banco de 10 leds.

Lo siguiente es colocar los elementos o componentes y se puede realizarlo de dos formas: automática o manual. De la forma manual se puede colocar cada elemento por separado donde se desee dentro de los límites de la placa, la cual esta dibujada con un borde amarillo. Para la forma automática hay que seleccionar la herramienta "*Auto Placer*", que se encuentra en la pestaña "*Tools*" como se observa en la ilustración B.3.

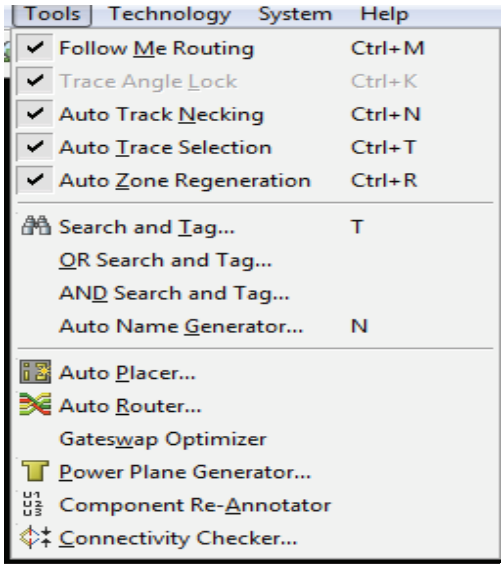


Ilustración B.3 Selección del "Auto Placer"

Los elementos se irán colocando de la forma en que el programa lo considere lo más óptimo, pero se puede modificar a gusto. El ruteo o trazado de las pistas se puede realizar de la misma forma automática que el posicionamiento de los componentes, automático o manual. De la forma manual se selecciona la herramienta "*Track Placement and Editing*" ubicado en el lado izquierdo de la ventana.

Mientras que para un trazado automático hay que seleccionar la herramienta "*Auto – router*" del menú "*Tools*". Aparecerá una ventana de dialogo donde se muestran algunas opciones por defecto como se muestra en la figura B.4, luego al pulsar en "*Begin Routing"* el programa comenzará a trazar las pistas. En la barra de estado de la parte inferior se podrá observar el progreso y los resultados se irán contemplando en la placa. Donde las pistas azules indican que la pista va debajo de la placa, mientras que las pistas rojas indican que la pista va sobre la placa.

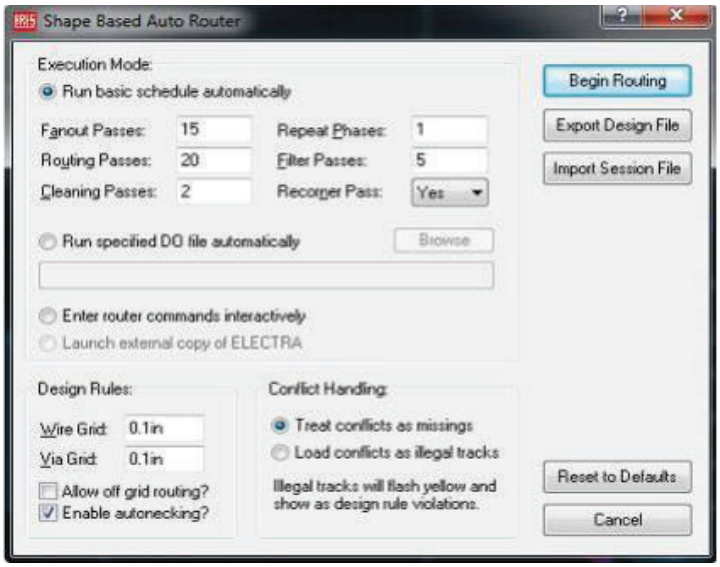


Ilustración B.4 Ventana de edición para trazado de pistas

Las pistas también se pueden editar haciendo clic derecho sobre ellas, aparecerán opciones como cambiar el tamaño de la pista, cambiar de lado superior a inferior o viceversa, etc. Finalmente ya con el diseño completado, lo siguiente fue obtener únicamente el diagrama de pistas (ilustración B.5), el cual se puede obtener el grafico en un archivo .pdf, para imprimirlo a laser en un papel termotransferible y luego sobre la placa.

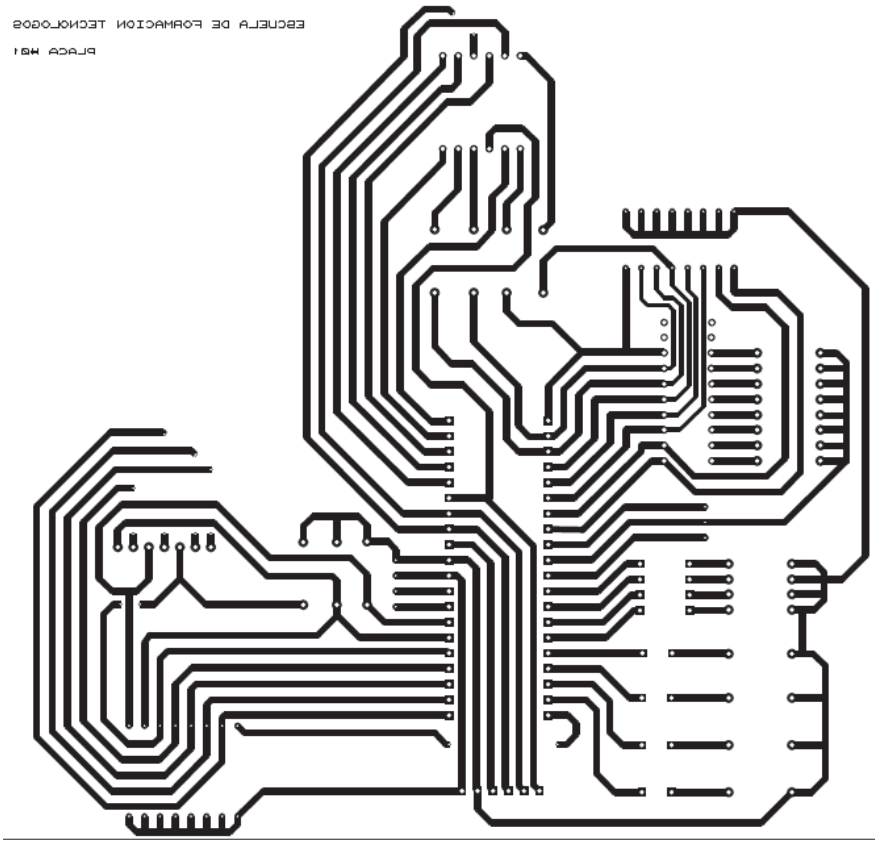


Ilustración B.5 Diagrama de pistas del módulo didáctico básico

ANEXO C. Manual de Usuario del Módulo Didáctico Básico

1. *Datasheet* – Hoja de datos del ATmega164PA

Este módulo didáctico básico está basado en un microcontrolador Atmega164PA, el cual en sus especificaciones técnicas cuenta con 40 pines, los cuales están distribuidos de la siguiente manera: 4 puertos (A, B, C, D) cada uno con 8 pines, 2 pines XTal (1 y 2) para conexiones de oscilador o un cristal de cuarzo, 3 pines para alimentación (1 VCC y 2 GND), 1 pin AREF (entrada analógica al conversor analógico - digital), 1 pin AVCC (provee energía al conversor analógico – digital) y 1 pin de Reset, tal como se observa en la ilustración C.1.

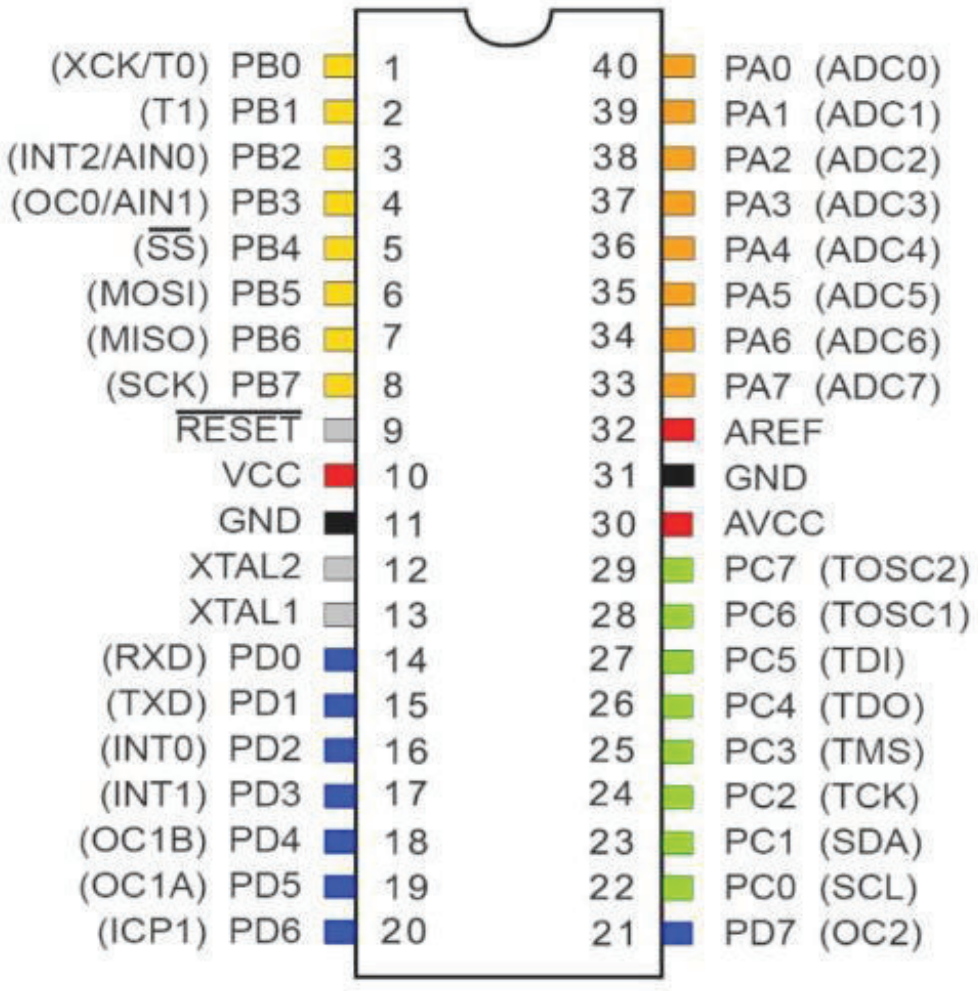


Ilustración C.1 Datasheet Atmega164PA.

2. Programador USB - USBasp

La programación de este microcontrolador se realizara a través de un programador USB para microcontroladores ATmel AVR llamado USBasp. Simplemente consiste en un

atmega88 o atmega8 y un par de componentes pasivos. El programador usa un controlador, este programador además tendrá la función de fuente de alimentación para los módulos didácticos básicos. En la ilustración C.2 se muestra el programador USBasp.

Ilustración C.2 Programador USBasp.

3. Requisitos del sistema

Para el correcto funcionamiento de estos módulos didácticos y su programador es necesario cubrir con ciertos requisitos, de software más no de *hardware.*
Los requisitos de software para poder ejecutar la programación de los módulos didácticos básicos son:

- Programa de grabación Progisp.
- *Driver* USBasp.
- Programa de escritura MikroC *for* AVR.

Los requisitos mínimos de hardware serían los siguientes:

- Placa de conexión USBasp.
- Computador personal.

4. Instalación de componentes *software*

4.1 Progisp

Para programar el microcontrolador primero se debe descargar el programa Progisp y los drivers para que el equipo reconozca al programador USBasp.

Escribir en el navegador de preferencia **Progisp**, descargarlo e instalarlo y cuando finalice esta será la ventana que permitirá leer, borrar, cargar programas, etc. (ilustración C.3).

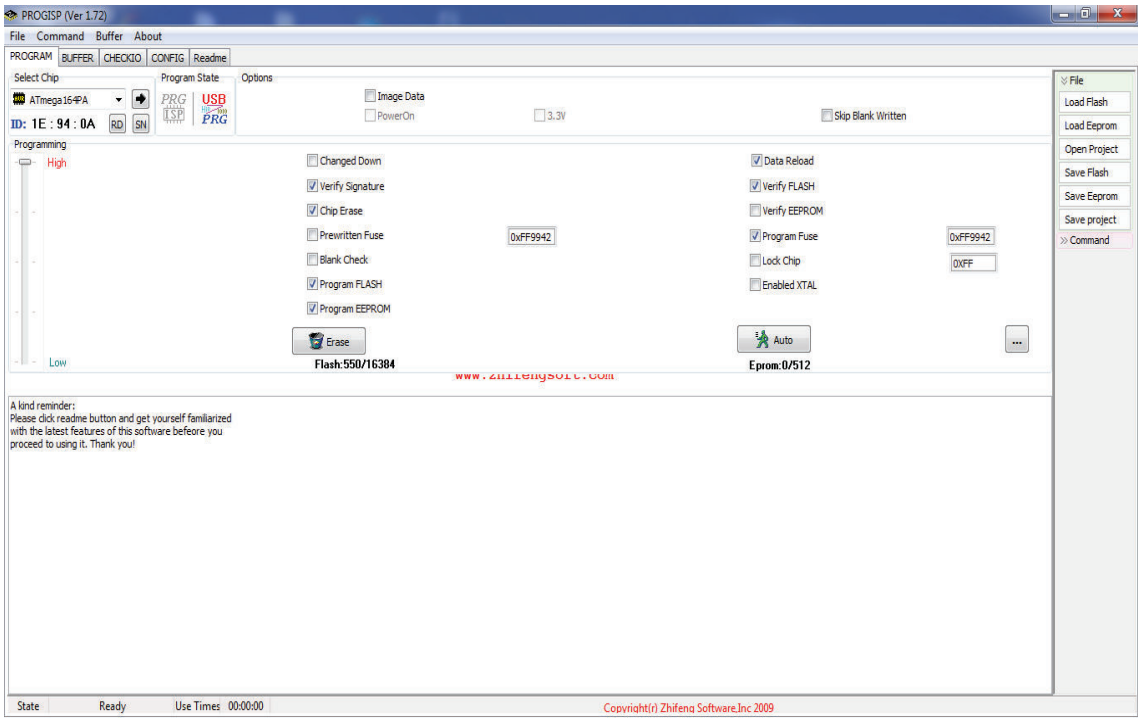


Ilustración C.3 Ventana del programa Progisp

4.2 USBasp

Al conectar el programador USBasp tanto la computadora como el Progisp no reconocerán al dispositivo, por tanto es necesario primero descargar los drives del USBasp, que puede ser descargado del enlace: http://www.fischl.de/usbasp/.

Seleccionar la primera opción la cual hace referencia a la última versión disponible, se descarga y luego se prosigue con los siguientes pasos:

- Abrir panel de control.
- Abrir la ventana de administración de dispositivos.
- En la pestaña de otros dispositivos estará el USBasp con un símbolo amarillo que indicara que no está instalado.
- Dar clic derecho en actualizar software de controlador.
- Luego en la opción buscar software de controlador en el equipo.
- Introducir la dirección donde se descargó el *driver*.

- Hacer clic en la opción de incluir subcarpetas.
- Dar clic en siguiente y comenzara la instalación.

Finalmente aparecerá la confirmación de la instalación del USBasp (ilustración C.4).

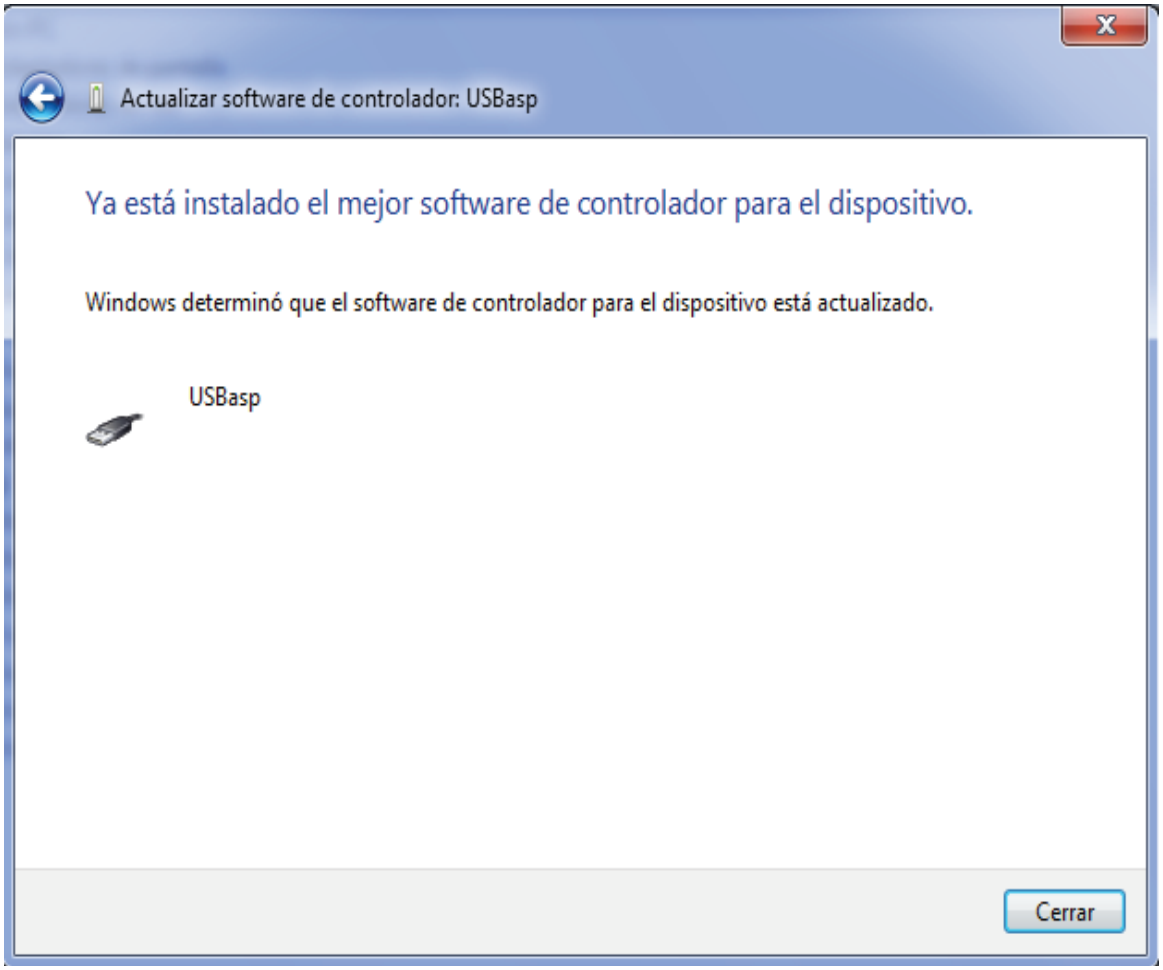


Ilustración C.4 Ventana de instalación del USBasp

5. MikroC PRO *for* AVR

EL programa para escribir en lenguaje de programación que se utilizará es el MikroC PRO *for* AVR, programa parecido al MikroC PRO *for* PIC. El cuál es un compilador C con todas las funciones para dispositivos AVR.

Proporciona un conjunto de bibliotecas que simplifican la inicialización y el uso de MCU compatibles con AVR y sus módulos. La instalación del MikroC para AVR se puede hacer mediante Internet usando el siguiente enlace: http://www.mikroe.com/mikroc/avr/

En este enlace se entrara a la página de Mikroelectronica, se descargará un archivo comprimido el cual contendrá el ejecutor del programa. Seguir los pasos para instalar,

cuando termine ya se podrá abrir y usar el programa aunque este estará en estado de programa no registrado (ilustración C.5), el cual no afecta el funcionamiento del mismo para crear programas y demás.

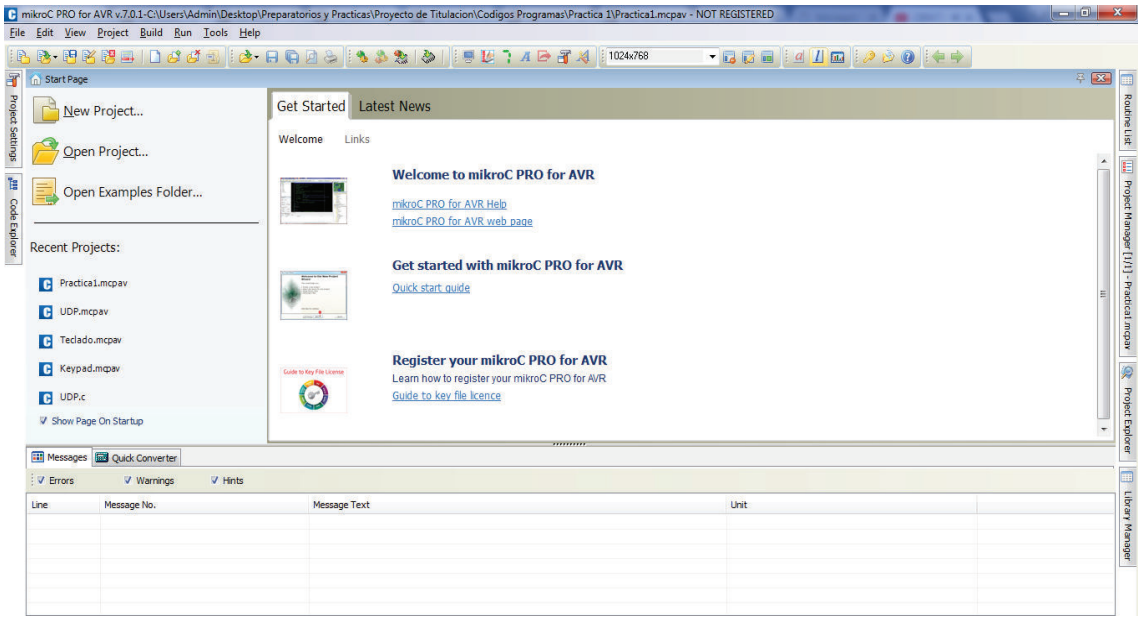


Ilustración C.5 Programa MikroC for AVR

6. Diseño del Módulo Didáctico Básico

El módulo didáctico básico está compuesto por los siguientes elementos electrónicos:

- Atmega164PA.
- Diodos LED.
- *Display* de 4 dígitos de 7 segmentos.
- Dipswitch de 4.
- 4 Pulsadores.
- Keypad 3x4.
- Programador USBasp.

El módulo tiene ciertos elementos que se pueden extraer, dado el caso que sufran alguna avería y estos están conectados con regletas o espadines (ilustración C.6) son:

- Atmega164PA.
- Diodos LED.
- *Display* de 4 dígitos de 7 segmentos.

- Programador USBasp.

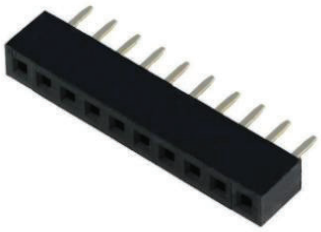


Ilustración C.6 Espadines Hembra

El módulo también cuenta con otros espadines que sirven para realizar conexiones extra, los cuales están conectados al puerto A y otro al puerto D. Hay dos espadines de 8, uno para conectar a VCC y otro para conectar a GND. Por último hay un espadín de 4 y otro de 3. El espadín de 4 tiene conexión con los pines: VCC, GND, XTAL1 y XTAL2. Y el espadín de 3 tiene conexión con los pines: AREF, GND y AVCC.

7. Distribución de elementos y conexiones

7.1 Arreglo de *display* de 7 segmentos

El Atmega164PA está compuesto de 40 pines los primeros 8 pines pertenecen al puerto B. En este puerto están conectado los *displays* de 7 segmentos. La conexión de este elemento está de la siguiente manera: el segmento A conectado al pin PB0, el segmento B al pin PB1 y así respectivamente. Este *display* cuenta con 12 pines (ilustración C.7), los cuales 8 son para los diferentes segmentos y 4 son para el encendido de los diferentes dígitos. Los pines de control están conectados al puerto A, el 4 está conectado con el pin PA0, el 3 con el pin PA1 y así respectivamente.

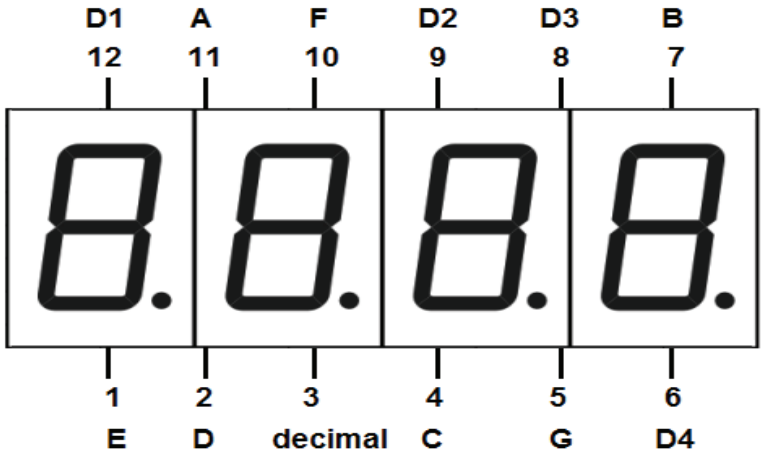


Ilustración C.7 Distribución pines de Display de 4 dígitos de 7 segmentos

7.2 Teclado matricial o *keypad* 3x4

Desde el pin 14 hasta el pin 21 del microcontrolador, lo compone el puerto D, en este puerto está conectado el Keypad o Teclado numérico. Este keypad se compone de 7 pines: 3 para columnas y 4 para filas (ilustración C.8). Los 3 pines correspondientes a las columnas están conectadas a los pines PD0, PD1 y PD2. Los 4 pines de las filas se conectan a los pines PD3, PD4, PD5 y PD6. En este puerto también está conectado un espadín hembra para conexiones adicionales y además junto a él está otro espadín hembra el cual permite la conexión a tierra.

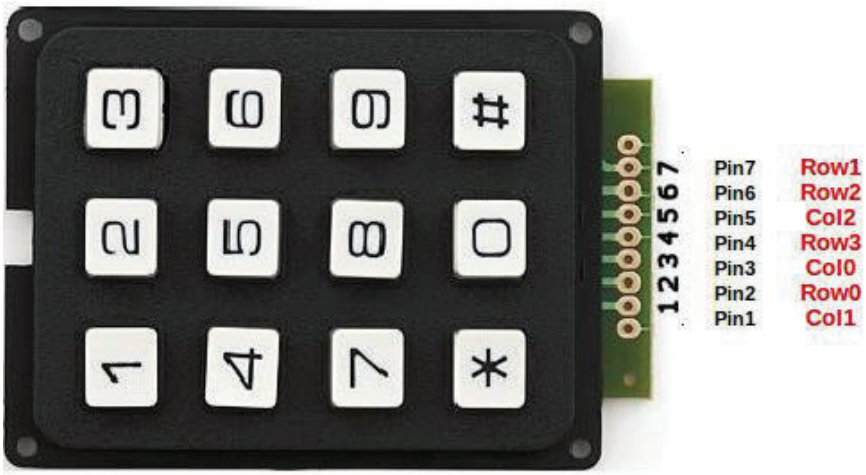


Ilustración C.8 Distribución pines Keypad 3x4.

7.3 Pulsadores y *Dipswitch* de 4 interruptores

El puerto C va desde el pin 22 hasta el pin 29 del ATmega164PA, en este puerto están los pulsadores e interruptores (ilustración C.9). Los 4 primeros pines PC0 al PC3 están conectados los pulsadores y en los otros 4 pines PC4 al PC7 está el Dipswitch de 4 y todos estos conectados con resistencias de 5.6KΩ para protección.

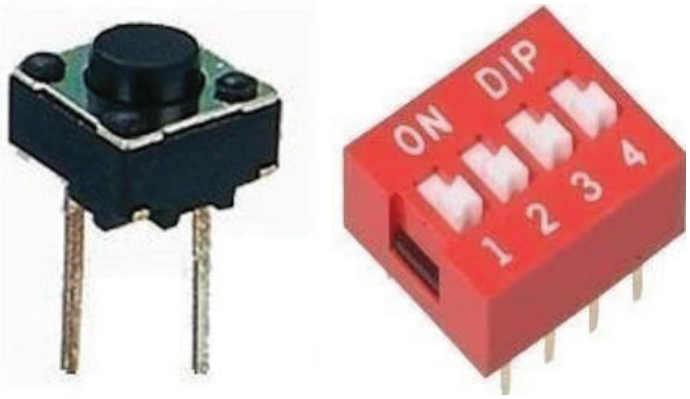


Ilustración C.9 Pulsador y dipswitch de 4

7.4 Banco de 10 leds

Finalmente desde los pines 33 al 40 conforman el puerto A en este se conecta un banco de leds (ilustración C.10) los cuales están conectados en serie con resistencias de 330Ω. También en este puerto está conectado un espadín de 8, esto sirve para el uso de los convertidores analógico – digital y junto a este otro espadín de 8 con conexión a VCC.

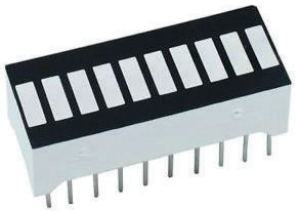
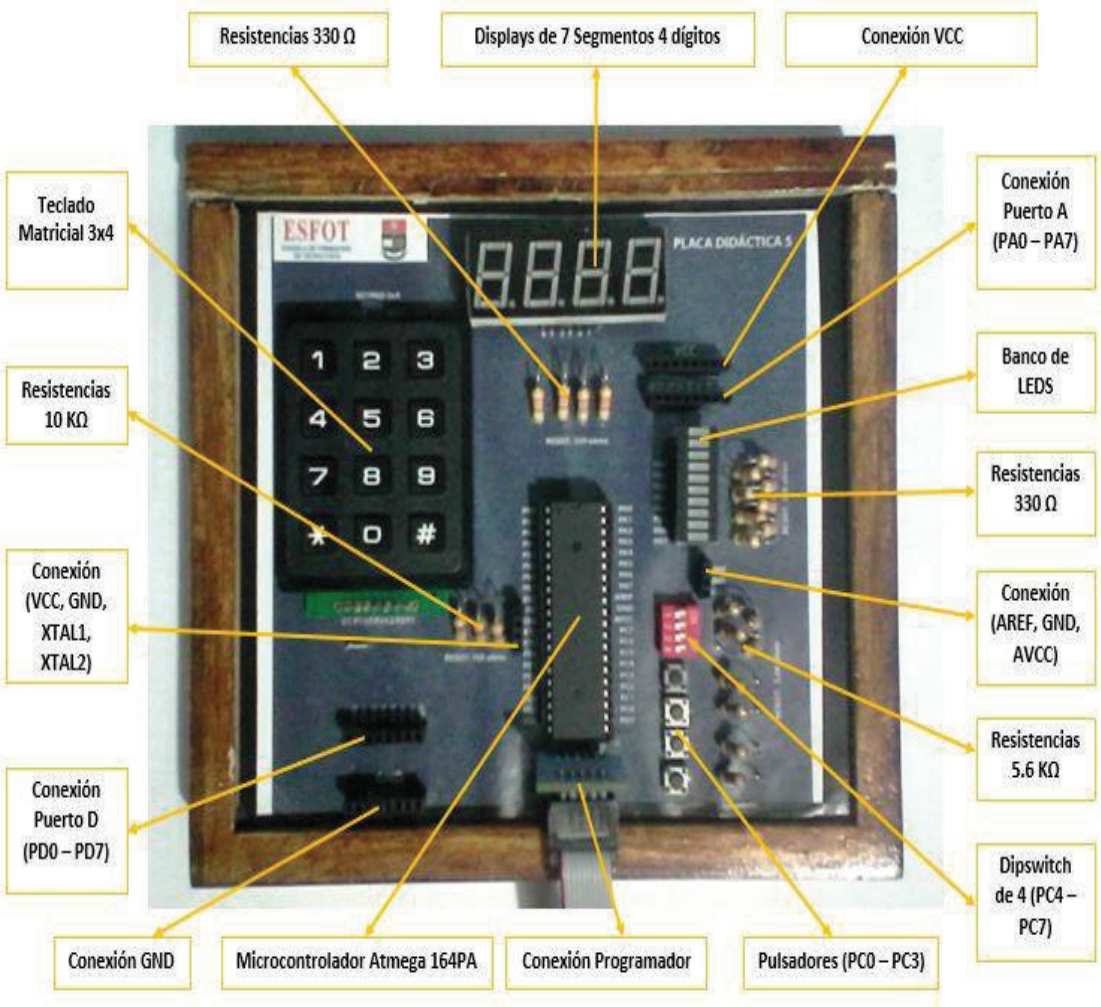


Ilustración C.10 Banco de 10 Leds

El diseño tiene marcado la información de cada pin para ayudar con las conexiones (ilustración C.11).



IlustraciónC.11 Módulo didáctico básico

8. Escritura de programas en MikroC *for* AVR

Para escribir un programa primero abrir el programa MikroC, clic en New Project y se debe seguir los siguientes pasos:

- Tipo de proyecto: estándar o visual.
- Configuración del proyecto: nombre proyecto, ubicación donde guardar el proyecto, tipo de microcontrolador y la frecuencia de trabajo (ilustración C.12).
- Añadir archivos: dado el caso si se quiere añadir archivos en formato .c.
- Librerías: se añaden todas las librerías o algunas específicas.

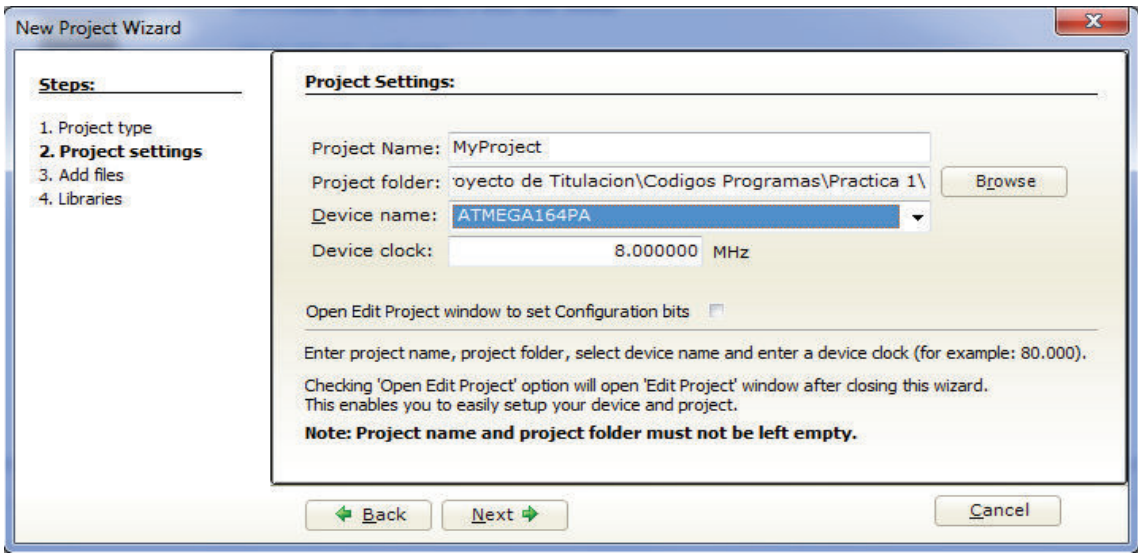


Ilustración C.12 Configuración proyecto: nombre, destino, dispositivo, frecuencia

Luego de seguir todos los pasos se creará el proyecto y la pestaña en la cual se escribe el programa a realizar en formato .c (ilustración C.13).

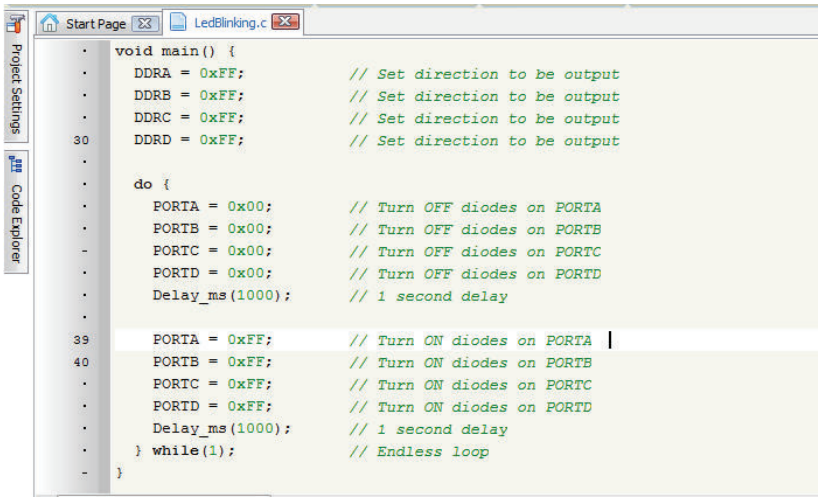


Ilustración C.13 Ejemplo de programa en MikroC

Dado el caso de que se haya elegido mal el dispositivo o la frecuencia de trabajo a la izquierda de la ventana esta la pestaña de configuración del proyecto (ilustración C.14).

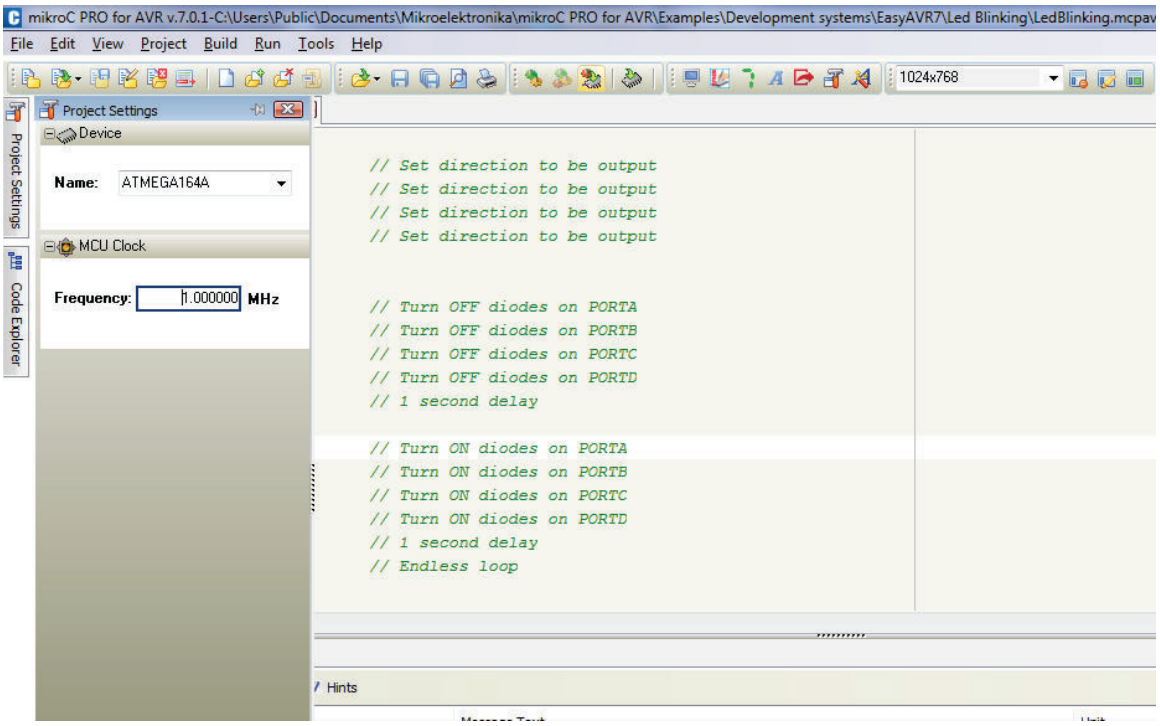


Ilustración C.141 Pestaña de configuración de proyecto

Si requiere ver las bibliotecas que cuenta el programa MikroC a la derecha de la ventana está el Library Manager (ilustración C.15).

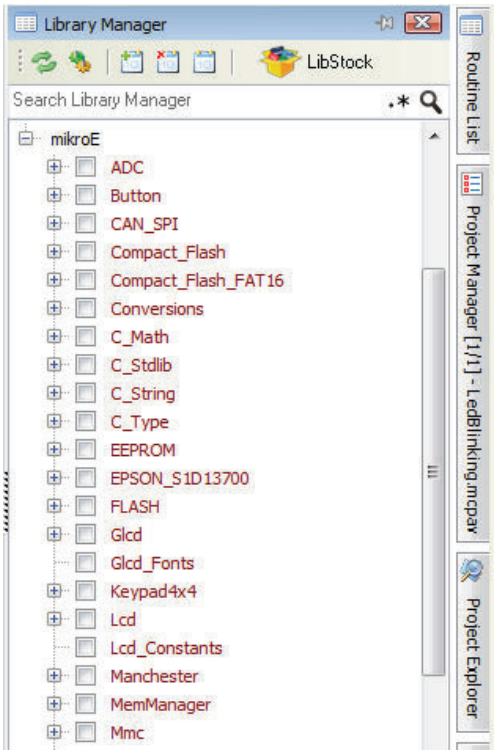


Ilustración C.15 Librerías del MikroC for AVR

Si se quiere buscar ayuda sobre como: escribir algún programa o información sobre comandos y librerías se debe presionar F1 o en la pestaña "*Help*" se puede abrir la ventana de ayuda del MikroC (ilustración C.16).

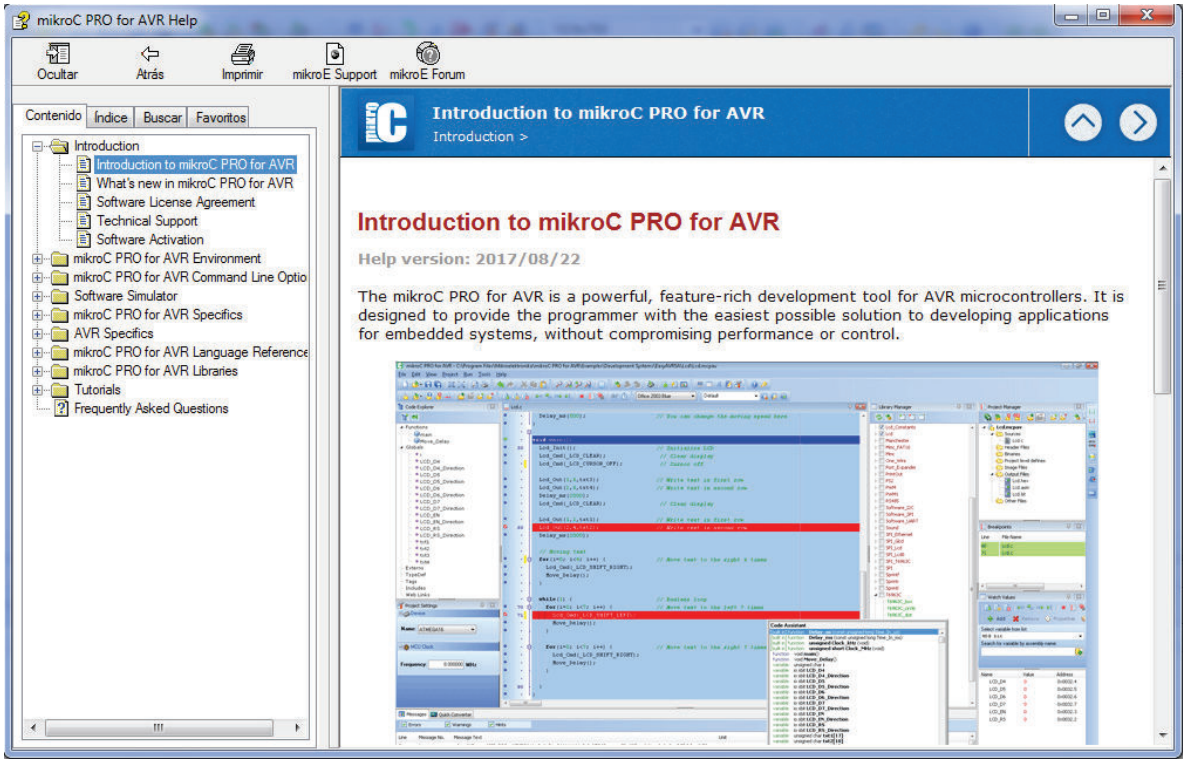


Ilustración C.16 Ventana de ayuda del MikroC

Finalmente cuando tenga el programa terminado, la compilación se la realiza tecleando Control + F9 o en la pestaña "*Build*", el cual creará un archivo .hex en donde se guardó el proyecto, dado el caso si el programa no tiene errores, si se produce la línea donde está el error se marcará de color rojo. Este archivo .hex es el que servirá para cargarlo en el microcontrolador mediante el programa Progisp.

9. Programación del módulo didáctico básico.

Con el programa Progisp se puede leer, borrar, cargar tanto la memoria *flash* como la Eeprom del microcontrolador. Antes que nada debe estar conectado el USBasp al módulo a través de su adaptador, el cual es una placa baquelita pequeña la cual conecta al microcontrolador con los siguientes pines:

- **MOSI:** (*Master Output Slave Input*) salida de datos del Master y entrada de datos al Slave.

- **MISO:** (*Master Input Slave Output*) salida de datos del Slave y entrada de datos al Master.
- **SCK:** (*Serial Clock*) señal de reloj del bus. Esta señal rige la velocidad que se transmite cada bit.
- **RESET:** el proceso del microcontrolador se resetea.
- **VCC:** conexión de alimentación a 5V.
- **GND:** conexión a tierra.

10. Pasos para programar el microcontrolador

- Selecciona el chip que en este caso será el ATmega164PA (ilustración C.17).

Ilustración C.17 Selección del chip en Progisp

- Ya seleccionado el chip ahora ya se puede leer, borrar y cargar programas. Para leer lo que está en el chip se selecciona la opción en la pestaña "*command*" y clic en "*read flash*". Y el registro de lo que este grabado en el microcontrolador se puede observar haciendo clic en la pestaña "*Buffer*" (ilustración C.18).

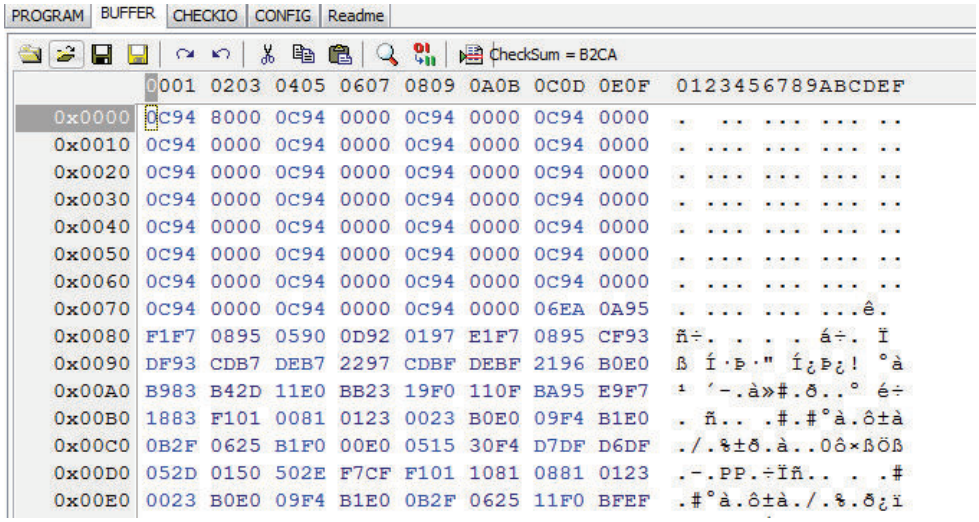


Ilustración C.18 Programa escrito previamente en el microcontrolador

- Para borrar lo que está escrito en el microcontrolador simplemente se hace clic en "*Erase"*, mientras que para grabar en la parte derecha se hace clic en la opción

"*Load Flash"* se selecciona el archivo .hex que quiere grabar y finalmente hacer clic en "*Auto*" para grabar (ilustración C.19).

Ilustración C.19 Entorno de opciones para borrar y cargar programas

- Si se da el caso que se requiera grabar el programa a una frecuencia diferente usando un reloj exterior o un cristal de cuarzo, hay que hacer clic en [...], en esta ventana (ilustración C.20) en la pestaña de "*navigation*" se encontrara todas estas opciones y para grabar estas especificaciones se debe hacer clic en "*write*", luego cerrar esta ventana y cargar el programa como se indica en el paso anterior.

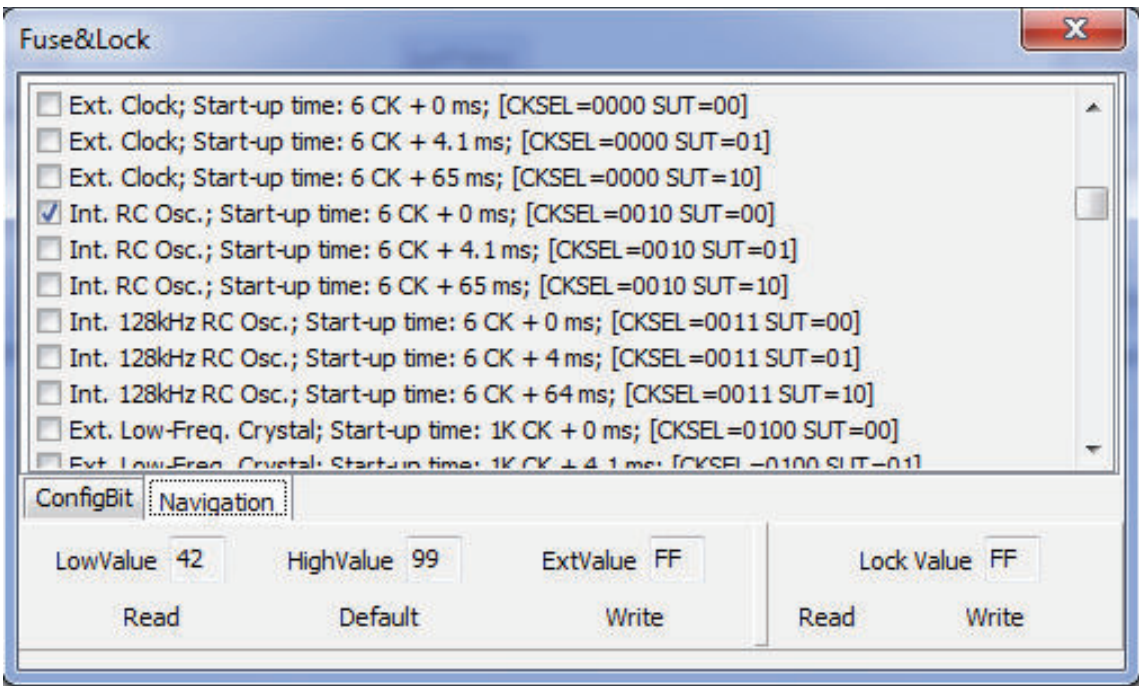


Ilustración C.202 Ventana de configuración de reloj

11. Posibles fallas y recomendaciones

La situación más frecuente que se puede presentar al momento de grabar es que aparezca un mensaje que dice: "*Chip Enable Program Error*" o error de habilitación de programa en chip (ilustración C.21).

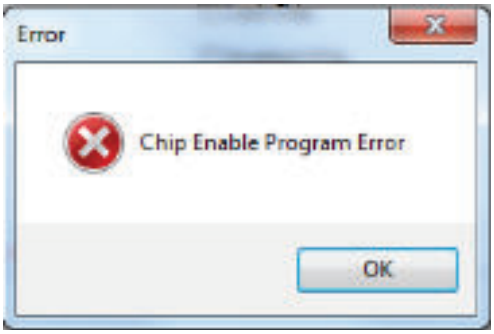


Ilustración C.21 Chip enable program error

Esto se produce debido a una falla en la conexión entre el microcontrolador y el programador USB, la solución es simplemente conectar verificar que el microcontrolador este correctamente a la placa. La forma de distribución de pines está indicada en la placa del módulo didáctico y el programador también tiene sus pines indicados en su placa de conexión.

Otros inconvenientes que pueden darse es que el *chip* seleccionado no es el correcto y el Progisp no permite que se haga la programación. Por otra parte se puede dar un mal funcionamiento debido a que el programa escrito en el MikroC se haya realizado en base a un microcontrolador diferente.

Si la frecuencia de trabajo que se estableció en el MikroC es diferente a la del microcontrolador, eso no quiere decir que no funcionará. Lo que pasará es que los procesos del programa se realizarán en diferentes tiempos, con latencia alta o viceversa.