ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"Sistemas temporizados implementados con microcontroladores Atmel, construcción de plataforma básica para explicar el uso detallado del temporizador Timer 2."

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES.

Presentado por:

Christian Antonio Blanco Salazar

Raúl Israel Bejarano Saritama

GUAYAQUIL – ECUADOR

AÑO 2012

AGRADECIMIENTO

A Dios.

A nuestros padres.

A todas las personas que contribuyeron en el desarrollo de este trabajo.

Al Ing. Carlos Valdivieso.

A las personas que en conjunto forman a la ESPOL por brindarnos la oportunidad de adquirir conocimientos y formarnos profesionalmente.

DEDICATORIA

A Dios, nuestro creador por su amor y protección.

A nuestros maestros, por su paciencia, bondad y generosidad por iluminar nuestra senda con la luz del conocimiento, por enseñarnos a ser útiles a los demás.

A nuestros padres y hermanos que son la fuerza que nos permite superarnos.

A nuestros amigos y compañeros, por la alegría compartida, por ser espíritu de solidaridad, por animarnos constantemente hacia la culminación de nuestra carrera.

TRIBUNAL DE SUSTENTACIÓN

Ing. Carlos Valdivieso A.

Profesor del Seminario de Graduación

Ing. Hugo Villavicencio V.

Delegado del Decano

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este trabajo, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

_____

Christian Antonio Blanco Salazar

_____

Raúl Israel Bejarano Saritama

RESUMEN

A continuación se presenta un resumen de cada capítulo.

En el Capítulo 1 se da una introducción sobre el tema de los microcontroladores, características del Timer2 del Atmega169, así como su configuración, modos de operación y diferencia respecto a otros microcontroladores.

En el Capítulo 2 se hace referencia a las Herramientas de Hardware y software que son necesarias para la implementación de este proyecto.

En el Capítulo 3 se describe los ejercicios utilizados para observar la funcionalidad del Timer2, así como los dispositivos necesarios para la implementación de los mismos.

En el Capítulo 4 se muestra el desarrollo e implementación de los ejercicios mediante el uso de diagramas de bloque y de flujo, lo cual permite la comprensión de cada ejercicio, además se muestra la funcionalidad de cada ejercicio utilizando la herramienta de simulación.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Con el presente trabajo se implementará una plataforma para el estudio y aprendizaje de las características y configuración del Temporizador Timer2, para lo cual se utilizará el AVR Butterfly de ATMEL en la implementación de los ejercicios desarrollados en el proyecto.

El objetivo del proyecto “Sistemas temporizados implementados con microcontroladores Atmel” es: la construcción de una plataforma básica para explicar el uso detallado del temporizador Timer 2, a través del desarrollo de una serie de ejercicios los cuales permiten conocer el funcionamiento, características y configuración del temporizador Timer2, para el desarrollo de cada ejercicio se han elaborado los diagramas de bloques y diagramas de flujo, permitiendo así una mayor comprensión del desarrollo e implementación de cada ejercicio.

Se ha utilizado la herramienta de software Proteus para comparar el comportamiento de los ejercicios implementados en la parte de simulación como en la parte real. Con esto los estudiantes de la materia de microcontroladores contarán con una versátil herramienta para el desarrollo de sus prácticas de laboratorio.

CAPÍTULO 1

1. Fundamento Teórico

En el presente capítulo se dará una introducción sobre las generalidades de la familia Atmel, así como la respectiva descripción del Timer2, detallando sus características, diferentes modos de operación y los registros involucrados para la configuración del mismo.

1.1. Generalidades

La implementación de sistemas electrónicos compactos y eficientes han permitido que el microcontrolador sea un dispositivo necesario para el desarrollo de los mismos, un microcontrolador es un circuito integrado que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada y salida, además cuenta con módulos internos (ADC, Temporizadores /Contadores,…).

El microcontrolador es utilizado en nuestro país para el diseño de sistemas electrónicos, debido a que nos permite manejar pantallas de cristal líquido (LCD), teclado matricial, sensores, memoria, además que ofrecen interfaces para comunicación (Serial, USB, RF…) con otros dispositivos.

Actualmente los fabricantes de microcontroladores más conocidos son: ST Microelectronics, Texas Instruments, Motorola Semiconductors, Infineon Technologies, Intel Corp, Microchip, Atmel Corp, Hitachi Corp, National Semiconductors. Debido a que el proyecto se basa en la aplicación del dispositivo AVR Butterfly, se explicará brevemente lo que significa la marca ATMEL y sus microcontroladores AVR.

Los chips Atmel se fabrican utilizando los más avanzados procesos de integración, que incluyen tecnologías BiCMOS, CMOS y Germanio de Silicio (SiGe) con lo cual la corporación ATMEL ha logrado ser líder mundial en el diseño, fabricación y comercialización de semiconductores avanzados, con enfoque en microcontroladores que incluyen lógica avanzada, memoria no volátil, circuitos integrados mezcladores de señal, componentes para radio frecuencia y sensores. Estas funciones se comercializan como productos estándar de aplicación específica (ASSP) o productos para clientes específicos (ASIC), proporcionando una respuesta rápida y flexible a las necesidades de los clientes de Atmel.

1.2. Descripción del Timer2

El Timer 2 del Atmega169 es un temporizador/contador de 8 bits. Su registro de conteo conocido como **TCNT2** es controlado por la unidad de control lógico, la cual se encarga de indicar el sentido del conteo, es decir si cambia de forma ascendente a descendente o viceversa, dependiendo del modo de operación que este configurado. Su registro de configuración principal es el **TCCR2A**, el cual se muestra a en la Figura 1.1 (1).

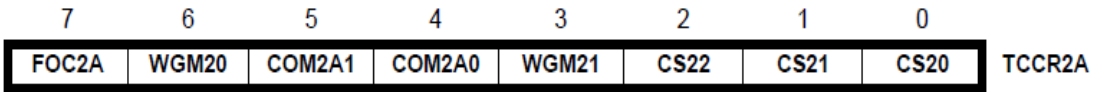


Figura 1.1 Registro TCCR2A del Timer 2.

El Timer 2 posee dos tipos de interrupciones que son: interrupción por desborde la cual se genera cuando su registro de conteo TCNT2 llega a su valor máximo (0xFF), y la interrupción por comparación la cual se genera cuando se cumple la igualdad entre el registro de comparación OCR2A con el registro de conteo TCNT2 (1).

La importancia de este Temporizador radica en su capacidad para generar señales PWM, generar una señal periódica con una frecuencia específica, realizar conteo de eventos y generar una señal de precisión usando la comparación de salida.

CS22	CS21	CS20	Descripción
0	0	0	Timer/Contador (apagado)
0	0	1	clk_{T2S}/ 1 (Sin preescalar)
0	1	0	clk_{T2S}/8
0	1	1	clk_{T2S}/32
1	0	0	clk_{T2S}/64
1	0	1	clk_{T2S}/128
1	1	0	clk_{T2S}/256
1	1	1	clk_{T2S}/1024

Figura 1.2 Bits para seteo del Pre-escalar del Timer 2

El Atmega169 está equipado con un sistema de tres temporizadores flexibles y potentes, los cuales se los conoce como Timer 0, Timer 1 y Timer2. En la Figura 1.3 se presenta la tabla comparativa con las características de los Timer.

Timer 0	Timer 1	Timer 2
- 8-bit timer/counter - 10-bit clock prescaler	- 16-bit timer/counter - 10-bit clock prescaler	- 8-bit timer/counter - 10-bit clock prescaler
Funciones – Pulse width modulation – Frequency generation – Event counter – Output compare	**Funciones** – Pulse width modulation – Frequency generation – Event counter – Output compare – 2 ch – Input capture	**Funciones** – Pulse width modulation – Frequency generation – Event counter – Output compare
Modos de Operacion – Normal – Clear timer on compare match (CTC) – Fast PWM – Phase correct PWM	**Modos de Operacion** – Normal – Clear timer on compare match (CTC) – Fast PWM – Phase correct PWM	**Modos de Operacion** – Normal – Clear timer on compare match (CTC) – Fast PWM – Phase correct PWM

Figura 1.3 Tabla comparativa con las características de los Timer.

La conducta del Timer/Contador y los pines de comparación de salida son definidos por la combinación de los bits en el modo de Generación de Forma de Onda (WGM21:0) y los bits del modo de Comparación de Salida (COM2A1:0). Los bits del modo de Comparación de Salida no afectan la secuencia de conteo, mientras que los bits del modo de Generación de la Forma de Onda sí. Los bits COM2A1:0 controlan si la salida generada PWM deberá ser invertida o no (invertida o no invertida PWM). Para el modo de PWM no invertida los bits COM2A1:0 controlan si la salida deberá ser puesta a uno, limpiada o invertida en una comparación igualada.

Mode	WGM21 (CTC2)	WGM20 (PWM2)	Timer/Counter Mode of Operation	TOP	Update of OCR2A at	TOV2 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR2A	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

Figura 1.4 Modos de operación de Timer2

Este temporizador puede ser configurado para operar en uno de los cuatro modos de funcionamiento como se lo muestra en la Figura 1.4, designados mediante los bits WGM01 y WGM00 en el registro TCCR2A (1)**.**

Los diferentes valores asignados a WGM21:0 indicaran el modo de operación como se muestra a continuación:

BOTTON: El contador alcanza el BOTTON cuando este llega a cero

MAX: El contador alcanza su MAX cuando este llega a 0xFF en decimal 255

TOP: El contador alcanza el Top dependiendo su modo de operación

1.2.1 Modo Normal

El modo más simple es el modo normal (WGM21:0 = 0). En este modo la dirección del conteo es siempre ascendente y no se limpia el contador. El contador simplemente se sobrescribe cuando pasa de su máximo valor de 8 bits (TOP = 0xFF) y entonces se reinicia desde su valor más bajo (0x00). En operación normal la Bandera de Sobreflujo del Timer/Contador (TOV2) será puesta a uno en el mismo ciclo de reloj del Timer como el TCNT2 llega a ser cero. La bandera TOV2 en este caso se comporta como el noveno bit, excepto si esta puesto en uno, y no es limpiado. Sin embargo, combinado con la interrupción de sobreflujo del Timer que automáticamente limpia la bandera TOV2, la resolución del Timer puede ser incrementada por software. No existen casos especiales que considerar en el modo normal, un nuevo valor en el contador puede ser escrito en cualquier instante.

La unidad de comparación de salida puede ser usada para generar interrupciones en algún momento dado. Usar la salida de comparación para generar formas de onda en el modo normal no es recomendado, ya que esto ocupara demasiado tiempo para el CPU (1).

1.2.2 Modo de Comparación de Limpieza del Timer (CTC)

En el modo de comparación de limpieza del Timer (WGM21:0=2), el registro OCR2A se utiliza para manipular la resolución del contador. En el modo CTC el contador se limpia a cero cuando el valor del contador TCNT2 iguala a OCR2A. El OCR2A define el valor tope para el contador, aunque también su resolución.

Este modo permite mayor control de la frecuencia de salida de comparación igualada. También simplifica la operación de conteo de eventos externos.

El diagrama de tiempos para el modo CTC se muestra en la Figura 1.5.

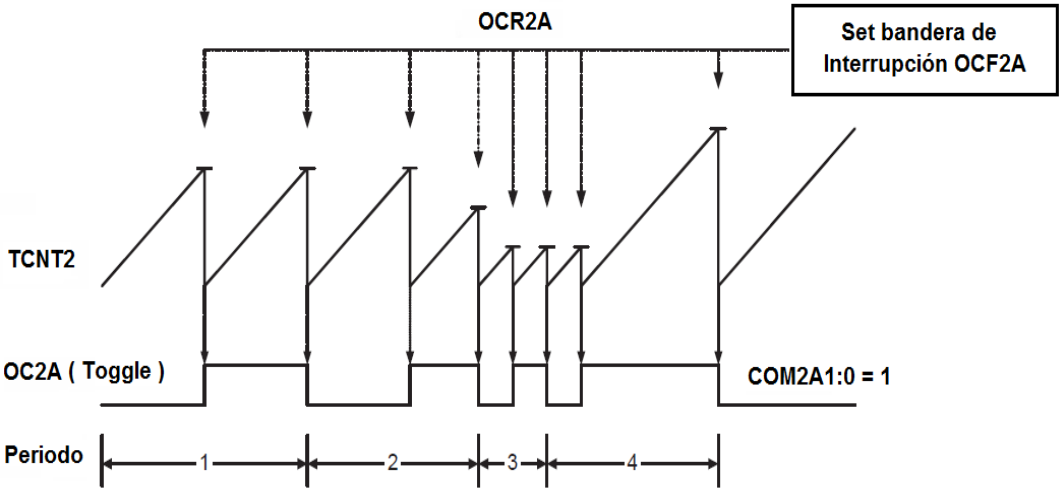


Figura 1.5 Diagrama de tiempos para el modo CTC

El valor del contador (TCNT2) se incrementa hasta la igualdad en la comparación entre TCNT2 y OCR2A, y entonces el contador TCNT2 se limpia.

Una interrupción puede generarse cada vez que el valor del contador alcanza el valor TOP usando la bandera OCF2A. Si la interrupción está habilitada, la rutina de manejo de la interrupción puede ser usada para actualizar el valor TOP. Sin embargo, cambiando TOP a un valor cercano a BOTTOM cuando el contador está corriendo sin ningún valor bajo de pre-escalador, debe realizarse con cuidado ya que en el modo CTC no se tiene la característica de doble buffer. Si el valor nuevo escrito OCR2A es tan más pequeño que el valor actual deTCNT2, el contador perderá la igualdad de comparación. El contador entonces contara a su máximo valor (0xFF) y volverá a contar desde 0x00 antes de que la igualdad de comparación ocurra.

Para generar una forma de onda de salida en el modo CTC, la salida OC2A puede ser invertida “toogle” de su nivel lógico cada vez que exista una igualdad de comparación, simplemente ajustando los bits de modo de comparación de salida a un modo “toggle” (COM2A1:0=1). El valor de OC2A no será visible en el pin del puerto a menos que la dirección de datos para el pin sea colocado como salida. La forma de onda generada tendrá una frecuencia máxima de $f_{OC2A}=f_{clk\_I/O}/2$ cuando OCR2A sea puesto en cero (0x00). La frecuencia de la forma de onda se define por la siguiente ecuación:

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2.N.(1+OCRnx)}$$ **Ecuación 1-1**

La variable N representa el factor pre-escalar (1, 8, 32, 64, 256 o 1024). En el modo normal de operación, la bandera TOV2 se coloca a uno en el mismo ciclo de reloj cuando contador pasa de MAX a 0x00 (1).

1.2.3 Modo PWM rápido

La modulación rápida PWM o el modo rápido PWM (WGM21:0=3) provee una alta generación en frecuencia de forma de onda PWM. El PWM rápido difiere de la opción anterior de PWM por su operación de una sola pendiente. El contador se incrementa de BOTTOM a MAX, entonces se reinicia desde BOTOM. En el modo no invertido de comparación de salida, el bit de comparación de salida (OC2A) se limpia cuando se iguala la comparación entre TCNT2 y OCR2A, y se ajusta a BOTTOM. En el modo invertido de comparación de salida, la salida es puesta a uno cuando se iguala la comparación y se limpia a BOTTOM. Debido a la operación de una sola pendiente, la frecuencia de operación del modo PWM rápido puede ser doblada tan alta como el modo de fase correcto de PWM que usa la operación dual de pendientes. Esta frecuencia alta hace que el modo rápido PWM se ajuste en aplicaciones de regulación de potencia, rectificación y DAC.

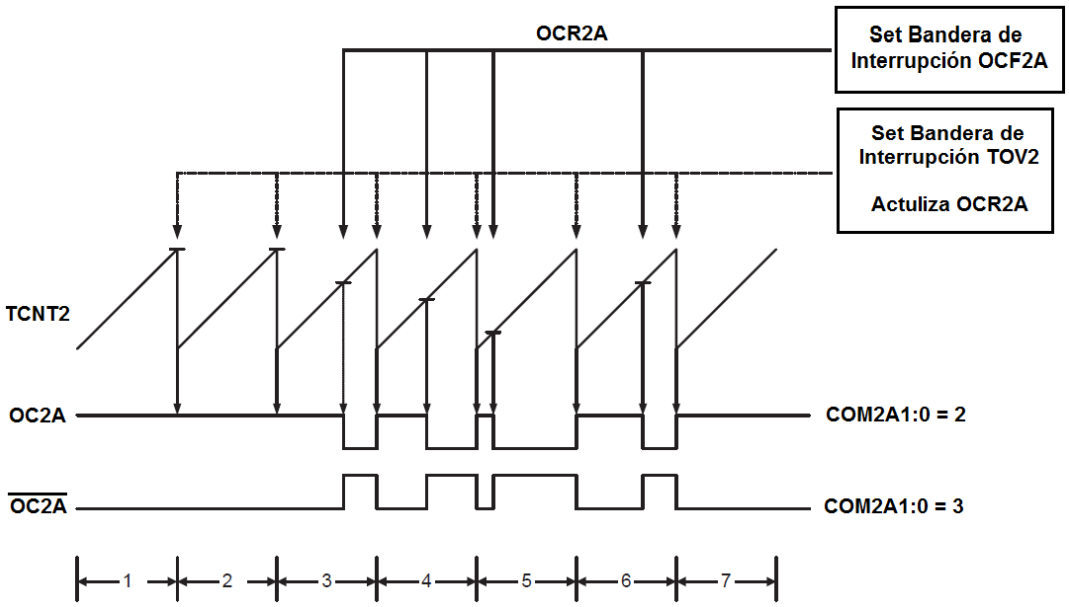


Figura 1.6 Diagrama de tiempos para el modo PWM rápido

En el modo rápido PWM, el contador se incrementa hasta el valor del contador iguala el valor MAX. El contador es entonces limpiado en el siguiente ciclo de reloj. El diagrama de tiempo para el modo PWM rápido se muestra en la Figura 1.6.

El valor de TCNT2 está en el diagrama de tiempo mostrado como un histograma para ilustrar la operación de una sola pendiente. El diagrama incluye salidas de PWM no invertida e invertida. La línea pequeña horizontal marca en donde la pendiente llega aTCNT2 y se iguala a OCR2A.La bandera de sobreflujo Timer/Contador (TOV2) se coloca a uno cada vez que el contador alcanza MAX. Si la interrupción está habilitada, la rutina que maneja la interrupción puede ser usada para actualizar el valor a comparar.

En el modo PWM rápido, la unidad de comparación permite la generación de formas de onda de PWM en el pin OC2A. Ajustando los bits COM2A1:0=2 producirá un PWM no invertido y un PWM invertido se logra ajustando los bits COM2A1:0=3 se genera un PWM invertido. El valor de OC2A actual será solamente visible en el pin del puerto si la dirección de los datos para el pin de puerto se coloca como salida. La forma de onda de PWM se genera poniendo a uno (o a cero) el registro OC2A en la comparación de igualdad entre OCR2A y TCNT2, y poniendo a cero (o a uno) el registro OC2A en el ciclo del Timer de reloj, el contador es limpiado (cambiando de MAX a BOTTOM).

La frecuencia PWM para la salida puede ser calculada por la siguiente ecuación:

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$ **Ecuación 1-2**

La variable N representa el factor pre-escalar (1, 8, 64, 256 o 1024). Los valores extremos para el registro OCR2A representan casos especiales cuando se genera una forma de onda PWM de salida en el modo rápido PWM. Si el OCR2A es igual BOTTOM, la salida será un pequeño salto por cada ciclo de reloj MAX+1. Ajustando de OCR2A igual a MAX resultara en una salida constante alta o baja (dependiendo de la polaridad del ajuste de salida por los bits COM0A1:0).

Una forma de onda con frecuencia de salida de un 50% a razón de ciclo en el modo rápido PWM puede ser alcanzado ajustando OC2A a su nivel lógico invertido en cada igualdad de comparación (COM2A1:0=1). La forma de onda generada tendrá una frecuencia máxima de $f_{OC2A} = f_{clk\_I/O}/2$ cuando OCR2A este en cero. Esta característica es similar al modo CTC de OC2A invertido, excepto por la característica del doble buffer en la unidad de comparación de salida este habilitada en el modo rápido PWM (1).

1.2.4 Modo PWM de Fase Correcta

El modo de fase correcta PWM (WGM21:0=1) provee una alta resolución de fase correcta de forma de onda en la generación de PWM. El modo de fase correcta de PWM se basa en la operación de doble pendiente. El contador se incrementa repetidamente desde BOTTOM a MAX y se decrementa desde MAX a BOTTOM. En el modo de comparación de salida no invertida, la comparación de salida (OC2A) se limpia cuando se igualan los registros entreTCNT2 y OCR2A mientras se cuenta ascendentemente y se pone a uno en la comparación igualada mientras se cuenta descendentemente.

En el modo de comparación de salida invertida, la operación se invierte. La operación de pendiente doble tiene una frecuencia de operación máxima más baja que la operación de una sola pendiente.

Sin embargo, debido a la característica simétrica de los modos de doble pendiente PWM, estos modos se prefieren para aplicaciones de control de motores. El diagrama de tiempos para el modo de fase correcta PWM se muestra en la Figura 1.7.

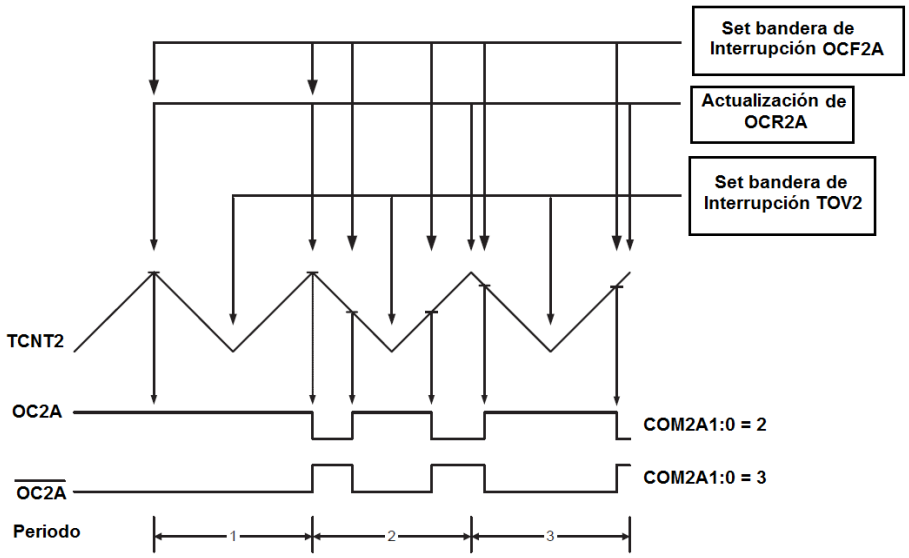


Figura 1.7 Diagrama de tiempos para el modo de fase correcta

La resolución PWM para el modo de fase correcta PWM es fija a 8 bits. En el modo de fase correcta PWM el contador se incrementa hasta que el valor del contador iguala a MAX. Cuando el contador alcanza el MAX, cambia la dirección de la cuenta. El valor de TCNT2 será igual a MAX en un ciclo de reloj del Timer.

El valor deTCNT2 se muestra como un histograma para ilustrar la operación de doble pendiente. El diagrama incluye salidas de PWM invertidas y no

invertidas. La línea horizontal en la pendiente TCNT2 representa la comparación igualada entre OCR2A y TCNT2.

La bandera de sobreflujo del Timer/Contador (TOV2) se pone a uno cada vez que el contador alcanza el BOTTOM. La bandera de interrupción se usa para generar una interrupción cada vez el contador alcanza el valor de BOTTOM.

En el modo de fase correcta PWM, la unidad de comparación permite la generación de formas de onda PWM en el pin OC0A. Ajustando los bits COM2A1:0=2 producirá una salida PWM no invertida. Una salida PWM invertida se genera colocando los bits COM2A1:0=3. El valor actual OC2A solamente será visible en el pin del puerto si la dirección de datos para el pin es puesto como salida. La forma de onda PWM es generada limpiando (o poniendo a 1) el registro OC2A en la comparación igualada entre OCR2A y TCNT2 cuando el contador se incrementa, y poniendo a uno (o limpiando) el registro OC2A en la comparación igualada entre OCR2A yTCNT2 cuando el contador se decrementa. La frecuencia del PWM para la salida cuando se usa el modo de fase correcta PWM puede ser calculada por la siguiente ecuación:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510} \qquad \textbf{Ecuación 1-3}$$

La variable N representa el factor pre-escalar (1, 8, 64, 256 o 1024). Los valores extremos para el registro OCR2A representan casos especiales cuando se genera una forma de onda de salida PWM en el modo de fase correcta PWM. El registro OCR2A es puesto igual a BOTTOM, la salida será continuamente en bajo y cuando se pone igual a MAX la salida será continuamente alta para el modo PWM no invertido. Para el PWM invertido la salida tendrá valores lógicos opuestos (1,2).

1.3. Diferencia en el Timer 2 entre microcontroladores Atmel, Microchip e Intel

Para analizar las características del Timer2 entre los tres fabricantes, se han tomado como referencia los microcontroladores Atmega169 (Atmel), Pic 16F87x (Microchip) y microcontrolador 8052(Intel).

El Timer2 del AVR es un temporizador-contador ascendente /descendente de 8 bits a diferencia del Timer2 del Pic 16F87X el cual es un temporizador/contador ascendente de 8 bits mientras que el Intel 8052 es un temporizador-contador ascendente de 16 bits.El Timer2 del AVR solo tiene un Preescalador que posee 8 posibles valores (0,1,8,16,32,64,128,256,1024) a diferencia del Timer2 del Pic el cual posee un Preescalador con 3 posibles valores (1, 4, 6) más un Post-escalador mientras que el Intel 8052 posee un Preescalador con tres posibles valores(8, 13, 16).

El Timer2 del AVR puede realizar una variación del PWM clásico implementado mediante el conteo ascendente y descendente del registro del conteo del Timer el cual es conocido como modo de operación PWM Phase Correct a diferencia del Timer2 del Pic que solo posee modo de operación PWM clásico implementado mediante el conteo ascendente del registro de conteo mientras que el Timer 2 del Intel 8052 no posee este modo de operación, sin embargo posee otro modo denominado generador de baudios.

El Timer2 del AVR tiene la cualidad de poder ser configurado de tal manera que realice su conteo hasta su desborde es decir 0xFF (Modo Normal, Modo PWM rápido y Modo PWM Phase Correct) o sino hasta coincidir la igualdad con el registro de comparación OCR2A luego de lo cual inicia un nuevo conteo (Modo CTC) a diferencia del Timer2 de PIC el cual solo lleva el conteo hasta que se cumple la igualdad con su registro de comparación PR2 (11).

El tiempo de la Temporización para activar la Bandera por desborde se calcula:

Para el Timer2 del AVR T= Tosc x Preescalar x 256

Para el PIC T= 4 x Tosc x Preescalar x valor de PR2+1 x Postescalar

Para el Intel 8052 T=12 x Tosc x Preescalar

CAPÍTULO 2

2. Fundamento teórico

El presente capítulo contiene información sobre las herramientas de Software y Hardware utilizadas. Desde el punto de vista del Software se realizará una descripción del entorno de trabajo de cada uno de ellos con la finalidad de entender su utilidad en la elaboración de cada uno de los ejercicios planteados mientras que desde el punto de vista del Hardware se detallara sus características importantes y los elementos que lo conforman con el fin de entender los recursos disponibles y limitaciones en cada uno de los dispositivos utilizados.

2.1. Herramientas de software

En la actualidad existen varias herramientas de software para realizar la simulación de los AVR Atmel y su programación tanto en lenguaje ensamblador como en lenguaje C. Para la elaboración de los ejercicios

planteados en este proyecto utilizaremos los programas AVR Studio 4 y El Proteus, los cuales serán detallados a continuación.

2.1.1 AVR Studio 4

AVR Studio es un programa distribuido gratuitamente por Atmel, trabaja con proyectos desarrollados en Ensamblador y en lenguaje C para microcontroladores AVR (8).

Descripción

AVR es un Ambiente de Desarrollo Integrado (IDE) para escribir y depurar aplicaciones AVR en Windows 2000/XP/Vista/7. Contiene herramientas para administración de proyectos, un editor de archivos fuente, un simulador y emulador In-circuit para los microcontroladores AVR de 8 bits.

AVR Studio trabaja perfectamente con el STK500 que permite la programación de los chips AVR y además es compatible con interfaces JTAG.

Características del AVR Studio 4

- Un núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemáticas y de manejo de archivos, proporcionadas por bibliotecas.
- Genera un código de maquina compatible con simuladores como Proteus.

• Punteros a direcciones de memoria, que permiten el uso de la información necesaria.

La ventana de AVR Studio se muestra en la figura 2.1 y sus partes se describen a continuación.

Menú de comandos.- El menú de comandos es similar al de otros programas de Windows con las opciones respectivas, como lo son: File, Project, Build, View, Tools, Debug y Help.

Barra de herramientas.- Contiene botones que ejecutan las las opciones mostradas en el menú de comandos.

Ventana Workspace.- En esta ventana se puede trabajar con los Registros de propósito múltiple, el Procesador, el Stack Monitor y los Puertos I/O del microcontrolador.

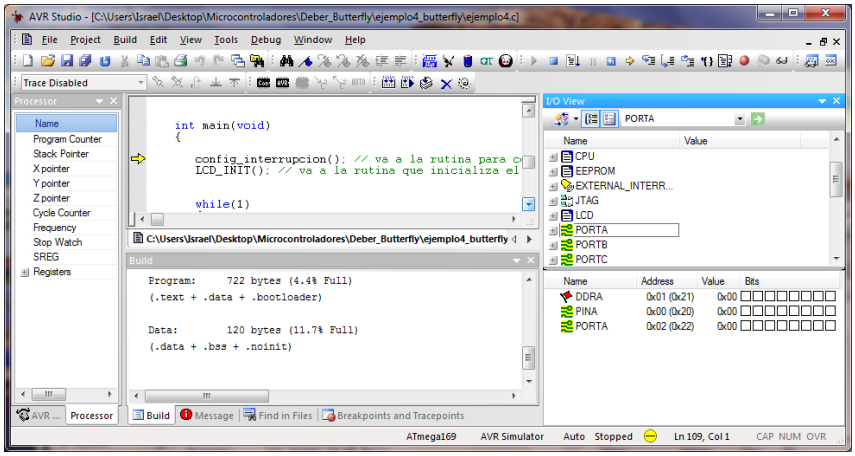


Figura 2.1 Ventana del programa AVR Studio 4

Ventana del editor del código de programa.- Esta ventana es donde se escribe el código del programa que se está desarrollando y que luego va a ser simulada.

2.1.2 Proteus

PROTEUS desarrollado por Labcenter Electronics; es un entorno integrado diseñado para la realización completa de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño, simulación, depuración y construcción. Este emulador consta de dos programas principales: Ares e Isis, y los módulos VSM y Electra.

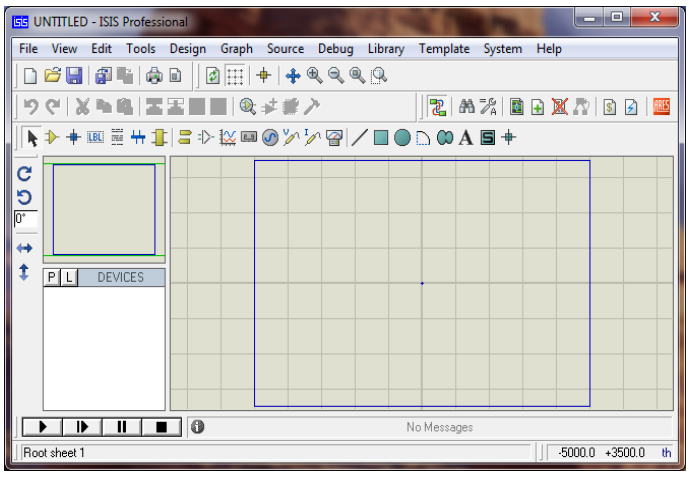


Figura 2.2 Ventana del programa Proteus

ISIS.- *Intelligent Schematic Input System* (Sistema de Enrutado de Esquemas Inteligente); es la herramienta que permite simular el esquema eléctrico del proyecto; incorporando una librería de más de 6.000 modelos

de dispositivos digitales y analógicos como son: resistencias, microcontroladores, leds, relés, microprocesadores, incluyendo fuentes de alimentación, generadores de señales, etc.

ARES.- Advanced Routing and Editing Software (Software de Edición y Ruteo Avanzado); es la herramienta de enrutado, ubicación y edición de componentes, se utiliza para la fabricación de placas de circuito impreso (PCB's). En la siguiente Figura se muestra la ventana principal Figura 2-2.

2.2. Herramientas de Hardware

Existen un sin número de microcontroladores de diferentes fabricantes que pueden ser utilizados en la elaboración de proyectos electrónicos dependiendo de las características y especificaciones técnicas requeridas. Para la implementación de los ejercicios elaborados en este proyecto la herramienta de hardware utilizada es el kit AVR Butterfly del fabricante Atmel el cual se presenta en detalle a continuación.

2.2.1 Tarjeta Butterfly

El AVR Butterfly es un Kit de desarrollo, entrenamiento y aprendizaje de la familia Atmel, que contiene el siguiente hardware: un microcontrolador ATmega169V, LCD, Joystick, altavoz, cristal de 32 KHz, DataFlash de 4 Mbit, convertidor de nivel RS-232, interfaz USART, interfaz USI, sensor de

temperatura, sensor de luz, ADC, conectores para acceso a periféricos, y Batería de 3 V.

El AVR Butterfly presenta las siguientes características:

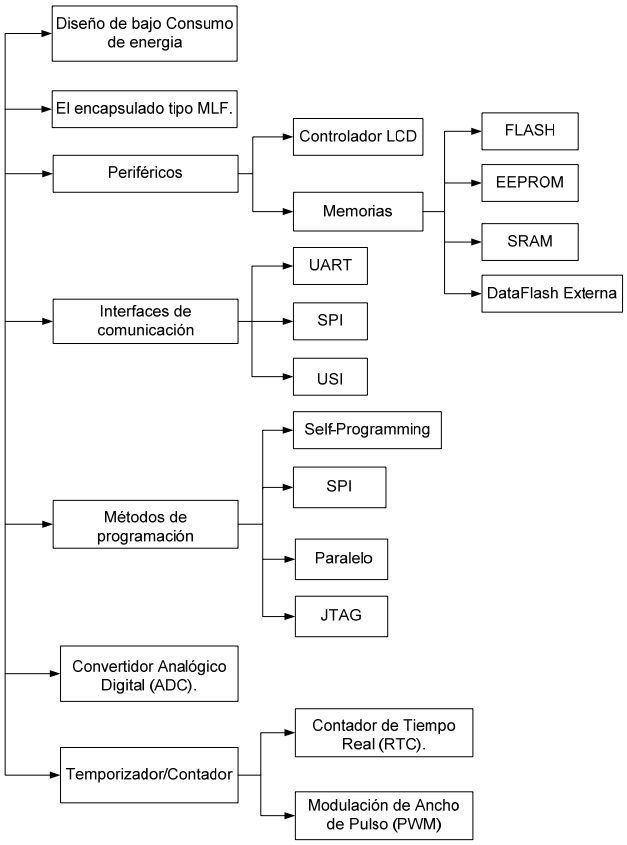


Figura 2.3 Características del AVR Butterfly

El AVR Butterfly utiliza el microcontrolador AVR ATmega169V, que combina la Tecnología Flash con un avanzado y versátil microcontrolador de 8 bits. En la Figura se puede apreciar el Kit AVR Butterfly.

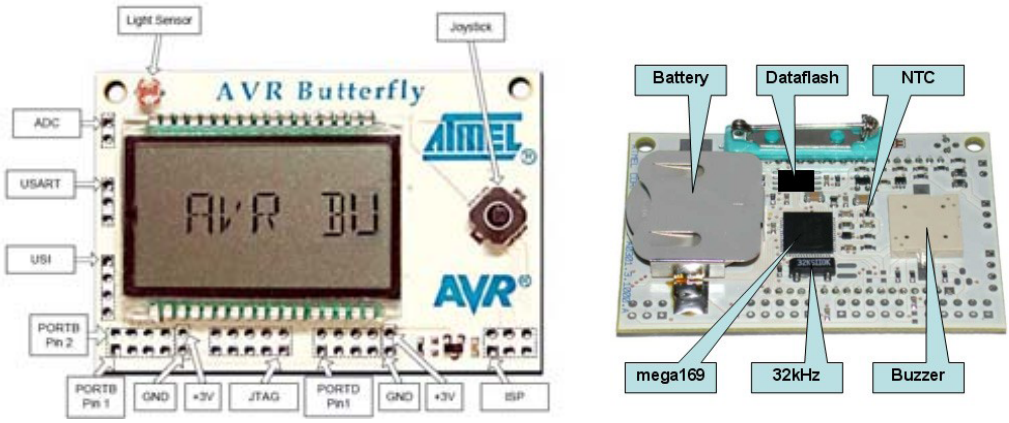


Figura 2.4 AVR Butterfly

Joystick

El AVR Butterfly tiene un joystick en miniatura para operar la entrada de usuario. Maneja en cinco direcciones, incluyendo push arriba, abajo derecha izquierda y centro. La línea común de todas las direcciones es GND. Esto significa que pull-p interna debe estar habilitado en el ATMEGA 169 a detectar a partir de la entrada de la palanca de mando, véase la figura 2.5.

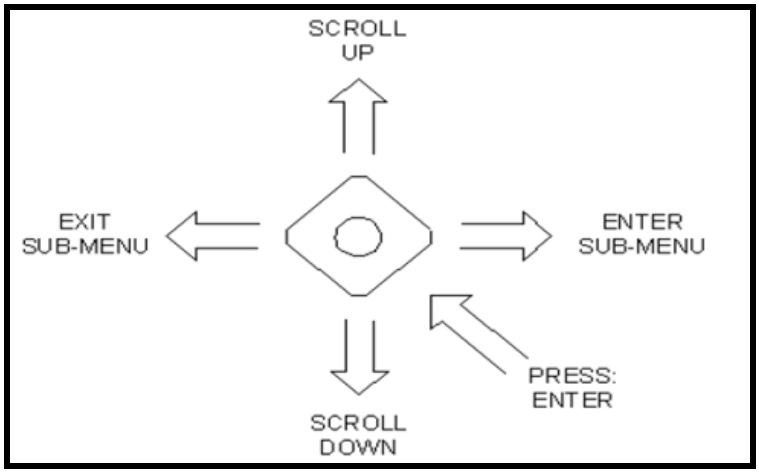


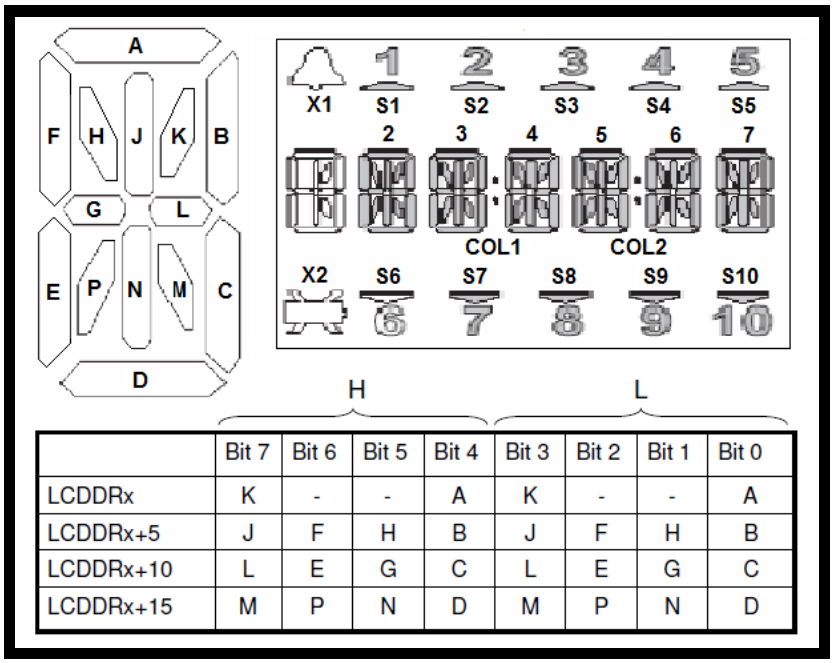
Figura 2.5 AVR Joystick

Pantalla LCD

La pantalla LCD del AVR Butterfly es la misma que la utilizada en la disposición STK502 de Atmel. Las conexiones entre el Atmega169 y la pantalla LCD también son las mismas.

STK502 es un módulo superior diseñado para añadir soporte Atmega169 a la placa de desarrollo STK500 de Atmel Corporation.

STK502 incluye una pantalla LCD. Cuenta con seis dígitos de 14 segmentos, y algunos segmentos adicionales. En general, la pantalla es compatible con 120 segmentos. La pantalla está diseñada para la tensión de funcionamiento de 3V.



	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDDRx	K	-	-	A	K	-	-	A
LCDDRx+5	J	F	H	B	J	F	H	B
LCDDRx+10	L	E	G	C	L	E	G	C
LCDDRx+15	M	P	N	D	M	P	N	D

Figura 2.6 Registros de segmentos LCD Butterfly

Atmega169

El Atmega169 es un microcontrolador de baja potencia CMOS de 8 bits basado en el AVR mejorado la arquitectura RISC. Mediante la ejecución de instrucciones de gran alcance en un solo ciclo de reloj, el Atmega 169 logra tasas de transferencia cerca de 1 MIPS por MHz que permite al diseñador del sistema optimizar el consumo de energía en comparación con la velocidad de procesamiento.

El núcleo AVR combina un amplio conjunto de instrucciones con 32 registros de propósito general de trabajo. Todos los 32 registros están conectados directamente a la unidad lógica aritmética (ALU), lo que permite dos registros independientes que se alcanzará en una sola instrucción ejecutada en un ciclo de reloj. La arquitectura resultante es un código más eficiente mientras que alcanza 24 rendimientos de hasta 10 veces más rápido que los convencionales microncontroladores CISC.

El Atmega 169 proporciona las siguientes características:

- Capacidades de 512 bytes de EEPROM y 1K bytes de SRAM.
- 54 registros de propósito general y 32 registros de propósito general.
- Controlador de LCD con la resistencia de step-up de tensión
- Sistema de interrupción

CAPÍTULO 3

3. Descripción e implementación del proyecto

Con el objetivo de facilitar el aprendizaje de la programación en lenguaje ensamblador y en lenguaje C de los microcontroladores Atmel y de manera muy particular del kit AVR Butterfly, hemos procedido a elaborar una plataforma en la cual se podrá implementar cada uno de los ejemplos propuestos en este proyecto sobre Timer2 y de esta manera visualizar y comprender los diferentes modos del operación de este Temporizador.

3.1. Implementación de la plataforma AVR Butterfly

Para la implementación del proyecto se necesitaron de siguientes componentes.

- **Protoboard** o placa de pruebas, el cual nos permite construir los prototipos de los circuitos electrónicos. Normalmente se utilizan para la realización de pruebas experimentales.

- **Cuatro pilas triple A recargables**, que se utilizan como fuente de poder del microcontrolador.
- **Tarjeta Butterfly,** la cual será programada para la implementación física de los ejercicios.
- **Socket para Pilas** utilizado para mantenerlas fijas y adicionalmente darnos la facilidad de cambiarlas en caso de que estén descargadas.
- **Bus de datos**: los cuales van conectados a los puertos B , D ,F de la tarjeta Butterfly según el requerimiento del ejemplo a desarrollar.
- **Cable USB a DB9:** se lo utiliza para poder cargar el código de cada ejemplo en el microcontrolador atmega169 del Avr Butterfly

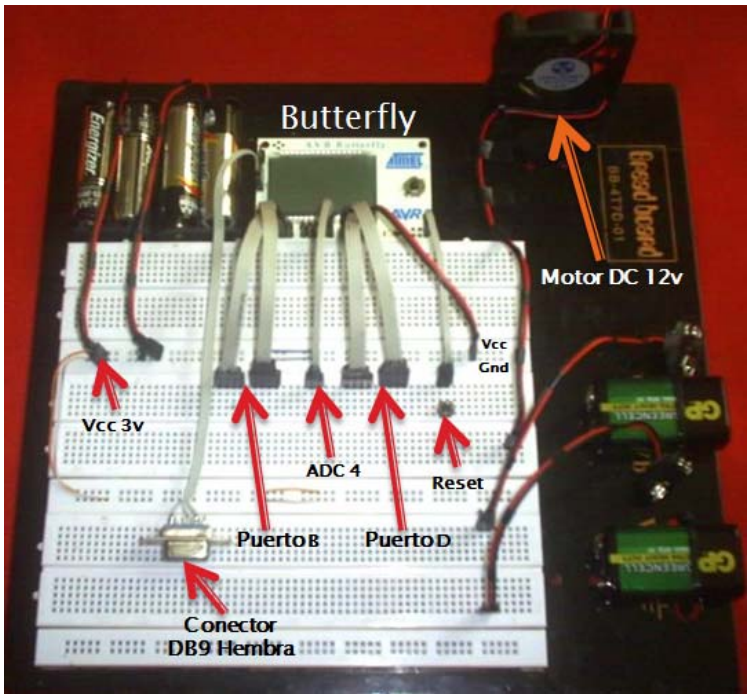


Figura 3.1 Plataforma AVR Butterfly

3.2. Descripción Teórica de los Ejemplos

El proyecto se basa en la implementación de sistemas para mostrar el uso del Timer2, por lo que se han desarrollado una serie de programas implementados tanto en lenguaje ensamblador como en lenguaje C, los cuales serán detallados a continuación.

3.2.1 Contador up/down

El siguiente programa está desarrollado en lenguaje ensamblador y consiste en un contador ascendente o descendente de dos dígitos, con una frecuencia de conteo de 1Hz, el cual se cambia el sentido del conteo mediante el uso joystick es decir, si se presiona hacia arriba el conteo es ascendente o si se presiona hacia abajo el conteo es descendente. El fundamento teórico se enfoca en la configuración del Timer2 en modo de operación Normal para realizar el conteo cada vez que ocurre la interrupción por desborde (**overflow**) del Timer2. Adicionalmente se configura el Timer2 para funcionar con fuente de reloj externa de 32,768 Hz y el respectivo preescalizador de 128 para obtener un tiempo de temporización por desborde igual a 1Hz, calculado mediante la siguiente fórmula:

$$t_{(over-flow)} = (f_{clk}/N)/256$$ **Ecuación 3-1**

Donde $\mathbf{f_{(clk)}}$ es la frecuencia de reloj en este caso externo y **N** es el valor del preescalizador. Finalmente el conteo será mostrado en el LCD de Butterfly.

3.2.2 Motor de 3 velocidades

El siguiente programa está desarrollado en lenguaje ensamblador y consiste en un motor de 3 velocidades donde su velocidad se cambia mediante el uso del joystick es decir, si se presiona hacia arriba se incrementa la velocidad o si se presiona en medio se disminuye la velocidad entre las 3 velocidades que se tiene. El fundamento teórico se enfoca en la configuración del Timer2 en modo de Operación PWM Phase Correct para generar una onda cuadrada en el pin OC2A al ocurrir la igualdad entre el valor cargado en el registro de comparación OCR2A con el valor del registro de conteo del Timer TCNT2. Para poder generar el cambio de velocidad en el motor, se debe cambiar el valor del registro de comparación OCR2A lo que da como resultado la variación del ancho del pulso de la onda cuadrada pero el periodo de la misma no cambia, esto es lo ocurre cada vez que se usa el Joystick para hacer el respectivo cambio de velocidad. Finalmente mediante los leds se indica en que velocidad se encuentra el motor es decir si en velocidad 1, velocidad 2 o velocidad 3.

Elementos:

• Plataforma AVR Butterfly y Motor DC de 12v

• 3 Leds y 4 resistencias de 330 Ohm

• Transistor 2N222 y Diodo 1N4007

• 2 pilas de 9v y regulador 7812

3.2.3 Encender/Apagar Led cada segundo utilizando interrupción por comparación.

El siguiente programa esta implementado en lenguaje C y consiste en el encendido o apagado de un Led cada 1 segundo. El fundamento Teórico se enfoca en la configuración del Timer2 en modo Normal para encender o apagar un LED cada 1segundo.

Para generar el estado de encendido del Led, se hace uso de la interrupción por Comparación es decir, cuando el valor del Registro OCR2A es igual al valor de registro de conteo del Timer2 TCNT2 entonces se procede a encender el Led y a mostrar en el LCD del Butterfly el mensaje "**ON**"**.**

Para generar el estado de apagado del Led, se hace uso de la interrupción por desborde es decir, cuando el valor del registro de conteo del Timer2 TCNT2 alcanza su valor máximo 0xFF entonces se procede a apagar el Led y a mostrar en el LCD del Butterfly el mensaje "**OFF**"**.**

Elementos:

• Plataforma AVR Butterfly

• Led

• Resistencia de 330 ohm

3.2.4 Onda cuadrada de frecuencia variable

El siguiente programa esta implementado en lenguaje C y consiste en la generación de una onda cuadrada cuya frecuencia se la hace variar mediante el uso de un potenciómetro. El fundamento Teórico se enfoca en la configuración del Timer2 en modo CTC para generar una onda cuadrada mediante el Pin OC2A al ocurrir la igualdad entre el valor cargado en el registro de comparación OCR2A con el valor del registro de conteo del Timer2 TCNT2. Para poder generar el cambio de la frecuencia en la onda cuadrada, se debe cambiar el valor del registro OCR2A lo que genera una variación del periodo de la onda cuadrada, esto se logra cargando continuamente OCR2A con el valor del registro ADCH generado de la conversión ADC del voltaje inyectado en el Pin ADC4 al variar la resistencia del potenciómetro, dicha conversión ADC está configurada con ajuste hacia la izquierda para obtener una resolución de 8 bits los cuales quedan almacenados en ADCH. Finalmente en el LCD del Butterfly se presenta el valor BCD de OCR2A para poder realizar el respectivo cálculo de la frecuencia y así comprobar lo obtenido en la simulación.

Elementos:

- Placa Prototipo
- Potenciómetro 10k
- Led
- Resistencia de 330 ohm

3.2.5 Control de velocidad motor DC

El siguiente programa esta implementado en lenguaje C y consiste en el control de la velocidad de un motor DC mediante el uso de un potenciómetro. El fundamento Teórico se enfoca en la configuración del Timer2 en modo FAST PWM para generar una onda cuadrada mediante el Pin OC2A al ocurrir la igualdad entre el valor cargado en el registro de comparación OCR2A con el valor del registro de conteo del Timer2 TCNT2. Para poder generar la variación de la velocidad del motor, se debe cambiar el valor del registro OCR2A lo que genera una variación en el ancho del pulso de la onda cuadrada pero su periodo no cambia, esto se logra cargando continuamente OCR2A con el valor del registro ADCH generado de la conversión ADC del voltaje inyectado en el Pin ADC4 al variar la resistencia del potenciómetro, dicha conversión ADC está configurada con ajuste hacia la izquierda para obtener una resolución de 8 bits los cuales quedan almacenados en ADCH. Finalmente en el LCD del Butterfly se presenta continuamente el mensaje "TIMER2 MODO FAST PWM".

Elementos:

• Plataforma AVR Butterfly

• Potenciómetro 10k y 1 resistencia de 330 Ohm

• Transistor 2N2222 y Diodo 1N4007

• Motor DC 12v

• 2 baterías de 9v y Regulador 7812

CAPÍTULO 4

4. Simulación y pruebas

En el presente capítulo se desarrolla el diagrama a de bloques el cual nos permite representar gráficamente el funcionamiento de los ejercicios implementados; el diagrama de flujo que nos permite representar gráficamente el algoritmo que utilizamos para el desarrollo de cada ejercicio; el código de cada ejemplo, estos códigos son compilados en el programa AVR Studio, y entonces procedemos a probar su funcionamiento utilizando el programa Proteus, el cual nos permite cargar los programas en el AVR Butterfly y simular el comportamiento del mismo, para finalmente ser implementado físicamente en el protoboard.

4.1. Programa: Contador UP/DOWN

El siguiente programa esta implementado en lenguaje de ensamblador y consiste en un contador up / down, el sentido del conteo se cambia mediante el joystick del butterfly, donde la dirección hacia arriba indica conteo ascendente y la dirección hacia abajo indica conteo descendente.

4.1.1 Diagrama de Bloques: Contador UP/DOWN

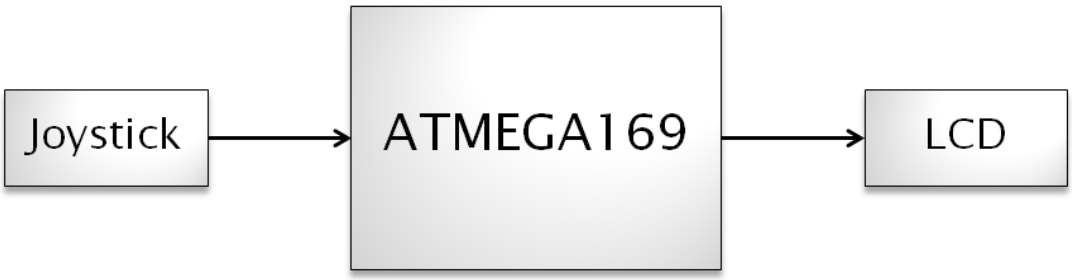
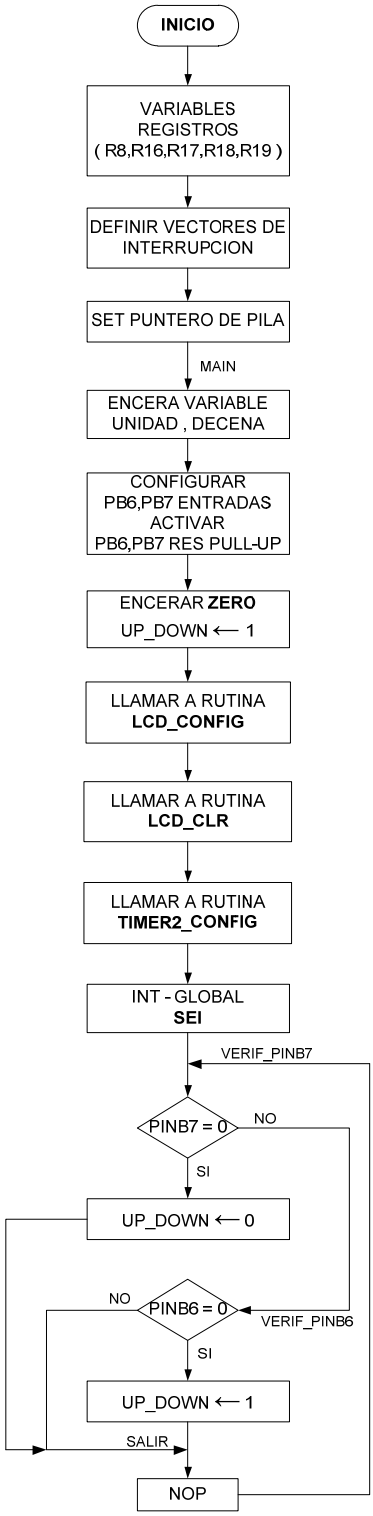
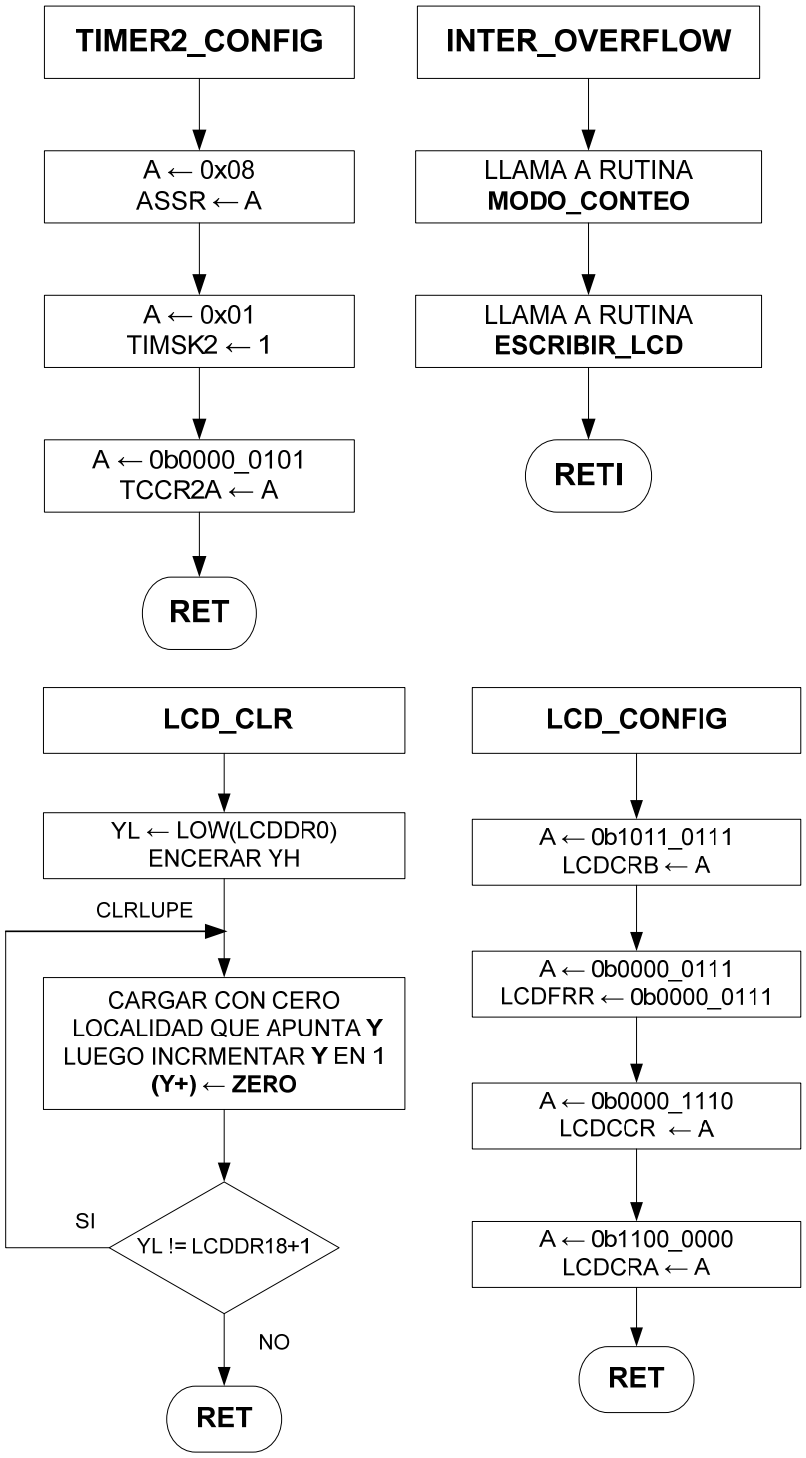


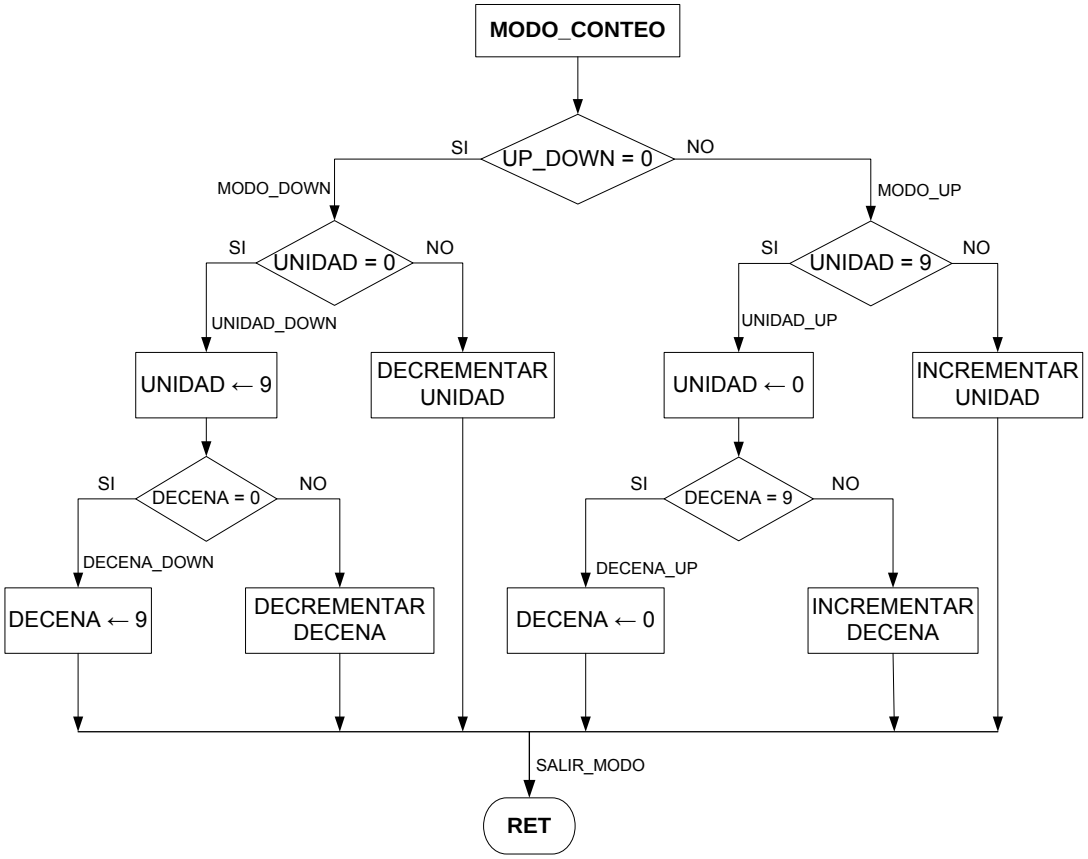
Figura 4.1 Diagrama de Bloques del contador UP/DOWN

4.1.2 Diagrama de flujo: Contador UP/DOWN



4.1.3 Diagrama de flujo: Rutinas Contador UP/DOWN




MODO_CONTEO
UP_DOWN = 0
SI
NO
MODO_DOWN
MODO_UP
UNIDAD = 0
UNIDAD = 9
UNIDAD_DOWN
UNIDAD_UP
UNIDAD ← 9
DECREMENTAR UNIDAD
UNIDAD ← 0
INCREMENTAR UNIDAD
DECENA = 0
DECENA = 9
DECENA_DOWN
DECENA_UP
DECENA ← 9
DECREMENTAR DECENA
DECENA ← 0
INCREMENTAR DECENA
SALIR_MODO
RET

4.1.4 Programa principal en lenguaje ensamblador.

```
/*****************************************************************

                    MICROCONTROLADORES AVANZADOS

INTEGRANTES:    RAUL BEJARANO
                CHRISTIAN BLANCO

GRUPO:  4

 PROGRAMA:  CONTADOR UP / DOWN

DESCRIPCION:  EL SIGUIENTE PROGRAMA ES UN CONTADOR UP / DOWN. EL SENTIDO DEL
CONTEO SE CAMBIA MEDIANTE EL JOSTICK DEL BUTTERFLY DONDE HACIA ARRIBA INDICA
CONTEO ASCENDENTE Y HACIA ABAJO INDICA CONTEO DESCENDENTE.

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

.INCLUDE "M169DEF.INC"    ;(DEFINICION DEL ATMEL A UTILIZAR)

;                       ---------------------------------------------------------------------------
============| DECLARACION DE VARIABLES Y VECTORES DE INTERRUPCION |============
;                       ---------------------------------------------------------------------------


.DEF              A=R16
.DEF         ZERO=R8      ; VARIABLE ZERO ES EL REGISTRO R8
.DEF     UNIDAD=R17     ; VARIABLE UNIDAD ES EL REGISTRO R17
.DEF    DECENA=R18     ; VARIABLE DECENA ES EL REGISTRO R18
.DEF UP_DOWN=R19     ; VARIABLE UP_DOWN ES EL REGISTRO R19


.ORG $0000          ; VECTOR DE INICIO DE PROGRAMA O RESET
  RJMP INICIO

.ORG $000A          ; VECTOR DE INTERRUPCION TIMER2 POR OVERFLOW
  RJMP INTER_OVERFLOW


;                                           ---------------------------------------------
;============================| SETEO DEL PUNTERO DE PILA |=========================
;                                            ---------------------------------------------
INICIO:

  LDI A,LOW(RAMEND)
  OUT SPL,A
  LDI A,HIGH(RAMEND)
  OUT SPH,A

;                                                   --------------------
;===================================| RUTINA MAIN |================================
;                                                    --------------------

MAIN:

    CLR   UNIDAD      ; ENCERA LA VARIABLE UNIDAD
    CLR   DECENA      ; ENCERA LA VARIABLE DECENA
    CBI   DDRB,6      ; PORTB6 COMO ENTRADA
    CBI   DDRB,7      ; PORTB7 COMO ENTRADA
    SBI   PORTB,6     ; ACTIVA RESISTENCIA PULL-UP EN PORTB6
    SBI   PORTB,7     ; ACTIVA RESISTENCIA PULL-UP EN PORTB7

    CLR   ZERO                  ; ENCERA LA VARIABLE CERO
```

```
   LDI    UP_DOWN,1              ; ARRANCA EN MODO_UP

           RCALL  LCD_CONFIG          ; CONFIGURAR EL LCD BUTTERFLY
   RCALL  LCD_CLR               ; LIMPIAR LOS SEGMETOS DEL LCD
   RCALL TIMER2_CONFIG    ; CONFIGURAR EL TIMER2

   SEI   ; ACTIVA LA INTERRUPCION GLOBAL


  VERIF_PINB7:

            SBIC     PINB,7              ; ¿ SE HA PRESIONADO BOTON DOWN PB7 ?
    RJMP   VERIF_PINB6     ;   NO
    LDI      UP_DOWN,0       ;   SI
    RJMP   SALIR

  VERIF_PINB6:

            SBIC     PINB,6          ; ¿ SE HA PRESIONADO BOTON DOWN PB6 ?
    RJMP   SALIR             ;   NO
    LDI      UP_DOWN,1    ;   SI

  SALIR:
          NOP
    RJMP   VERIF_PINB7

;=================================================================================
;                          ---------------------------------------------------------------------------------
;================| RUTINA DE INTERRUPCION CADA 32,768/128/256 = 1 SECOND |==========
;                          ---------------------------------------------------------------------------------

INTER_OVERFLOW:

   RCALL  MODO_CONTEO     ; LLAMA A LA RUTINA MODO_CONTEO

   RCALL  ESCRIBIR_LCD       ; LLAMA A LA RUTINA ESCRIBIR_LCD

   RETI

;                                      ----------------------- ---------------------------------
;======================| CONFIGURACION DEL LCD BUTTERFLY |======================
;                                      ------------------------------------------------------------

LCD_CONFIG:

       LDI A,0b1011_0111  ;SETEO DEL RELOJ, CICLO DE TRABAJO Y NUMERO DE PINES A USAR
       STS LCDCRB, A      ;HABILITA TODOS LOS SEGEMENTOS
       ;LCDCRB: [LCDCS,LCDB2,LCDMUX1,LCDMUX0,_,LCDPM2,LCDPM1,LCDPM0]

       LDI A,0b0000_0111  ;SETEO DEL FRAME RATE A 32Hz
       STS LCDFRR, A      ;LCDFRR: [_,LCDPS2,LCDPS1,LCDPS0,_,LCDCD2,LCDCD1,LCDCD0]

       LDI A,0b0000_1110  ;(LCDCC3,LCDCC2,LCDCC1)
       STS LCDCCR, A      ;SETEO DEL CONTRASTE DE LA PANTALLA

       LDI A,0b1100_0000  ; (LCDEN,LCDAB)
       STS LCDCRA, A      ; ENCIENDE EL LCD
       RET

;                                      ----------------------------------------------
;======================| LIMPIAR SEGMENTOS DEL LCD |==============================
;                                      ----------------------------------------------

  /* FUNCION MEDIANTE LA CUAL SE VA LIMPIANDO CADA UNO DE LOS SEGMENTOS DEL LCD
                                                HACIENDO
          CERO CADA UNO DE SUS RESPECTVOS REGISTROS DESDE LCDDRO - LCDDR18 */
```

```
LCD_CLR:
        LDI   YL,LOW(LCDDR0) ; SE CARGA EN YL LA DIRECCION DEL REGISTRO LCDDR0
        CLR YH
CLRLUPE:
        ST   Y+,ZERO             ; CARGA CON CERO LA DIRECCION A LA QUE APUNTA Y
        CPI  YL,LCDDR18+1    ; ¿ YL ES IGUAL A LA DIRECCION DE LCDDR18+1 ?
        BRNE CLRLUPE          ;   NO
        RET                          ;   SI

;                                        ----------------------------------------------
;======================| CONFIGURACION DEL TIMER2 |=============================
;                                         ----------------------------------------------

TIMER2_CONFIG:

       LDI    A,8                  ;  ELIGIR TIMER2 CLOCK DE 32.768KHz
       STS   ASSR,A
       LDI    A,1                  ;  1=OVF, 2=OCF0A MATCH - SET TIMER MODE
       STS  TIMSK2,A            ;  HABILTA LA INTERRUPCIOM OVF (OVERFLOW)
       LDI    A,0b0000_0101   ;  TIMER PRESCALE=128 ; MODO DE OPERACION NORMAL
       STS  TCCR2A,A
       RET

;                                          ----------------------------------------------
;========================| ESCRIBIR NUMEROS EN EL LCD |==========================
;                                          ----------------------------------------------

/* FUNCION PARA ESCRIBIR LA UNIDAD Y LA DECENA CARGANDO LOS RESPECTIVOS VALORES
EN LOS
  LOS REGISTROS QUE MANEJAN AL LCD
  PARA LA UNIDAD REGISTROS    LCDDR1, LCDDR6, LCDDR11, LCDDR16
  PARA LA DECENA REGISTROS   LCDDR0, LCDDR5, LCDDR10, LCDDR15
*/

ESCRIBIR_LCD:

   ;---- ESCRIBIR LA UNIDAD ----------

   LDI  ZL,LOW(LCD_TABLA*2)
   LDI  ZH,HIGH(LCD_TABLA*2) ;SE CARGA EN EL PUNTERO Z LA DIRECCION DE LCD_TABLA

     MOV    A,UNIDAD    ; COPIA EN A EL VALOR DE LA UNIDAD
     LSL      A               ; OFFSET PARA APUNTAR AL DATO QUE QUEREMOS DE LA TABLA
     ADD    ZL,A            ; SUMA EL OFFSET AL PUNTERO ZL
     ADC    ZH,ZERO

     LPM    A,Z             ; CARGA EN A EL DATO DE MEMORIA DE PROGRAMA APUNTADO POR Z
     ANDI   A,0x0F        ; FILTRA EL NIBBLE MENOS SIGNIFICATIVO
     STS    LCDDR1,A   ; COPIA EN LCDDR1 EL VALOR QUE CONTIENE A

     LPM    A,Z+         ; CARGA EN A EL DATO DE MEMORIA DE PROGRAMA
                                ; APUNTADO POR Z Y LUEGO INCREMENTA EN 1 Z
            SWAP   A           ; INTERCAMBIA EL NIBBLE MAS SIGNIFICATIVO CON EL NIBBLE
                                ; MEMOS SIGNIFICATIVO Y VICEVERSA
     ANDI   A,0x0F        ; FILTRA EL NIBBLE MENOS SIGNIFICATIVO
     STS    LCDDR6,A   ; COPIA EN LCDDR6 EL VALOR QUE CONTIENE A

     LPM    A,Z              ; CARGA EN A EL DATO DE MEMORIA DE PROGRAMA APUNTADO POR Z
     ANDI   A,0x0F         ; FILTRA EL NIBBLE MENOS SIGNIFICATIVO
            STS    LCDDR11,A   ; COPIA EN LCDDR11 EL VALOR QUE CONTIENE A

     LPM    A,Z        ; CARGA EN A EL DATO DE MEMORIA DE PROGRAMA APUNTADO POR Z
     SWAP   A         ; INTERCAMBIA EL NIBBLE MAS SIGNIFICATIVO CON EL NIBBLE MEMOS
                              ; SIGNIFICATIVO Y VICEVERSA
     ANDI   A,0x0F          ; FILTRA EL NIBBLE MENOS SIGNIFICATIVO
```

```
            STS    LCDDR16,A     ; COPIA EN LCDDR16 EL VALOR QUE CONTIENE A

   ;------ ESCRIBIR LA DECENA ---------

   LDI  ZL,LOW(LCD_TABLA*2)
   LDI  ZH,HIGH(LCD_TABLA*2) ;SE CARGA EN EL PUNTERO Z LA DIRECCION DE LCD_TABLA

     MOV    A,DECENA    ; COPIA EN A EL VALOR DE LA DECENA
     LSL     A                ; OFFSET PARA APUNTAR AL DATO QUE QUEREMOS DE LA TABLA
     ADD    ZL,A              ; SUMA EL OFFSET AL PUNTERO ZL
     ADC    ZH,ZERO

     LPM    A,Z        ; CARGA EN A EL DATO DE MEMORIA DE PROGRAMA APUNTADO POR Z
            SWAP   A          ; INTERCAMBIA EL NIBBLE MAS SIGNIFICATIVO
                  ; CON EL NIBBLE MEMOS SIGNIFICATIVO Y VICEVERSA
     ANDI   A,0xF0         ; FILTRA EL NIBBLE MAS SIGNIFICATIVO
     STS    LCDDR0,A    ; COPIA EN LCDDR0 EL VALOR QUE CONTIENE A

     LPM    A,Z+       ; CARGA EN A EL DATO DE MEMORIA DE PROGRAMA
                           ;  APUNTADO POR Z Y LUEGO INCREMENTA EN 1 Z
     ANDI   A,0xF0         ; FILTRA EL NIBBLE MAS SIGNIFICATIVO
     STS    LCDDR5,A    ; COPIA EN LCDDR5 EL VALOR QUE CONTIENE A

     LPM    A,Z         ; CARGA EN A EL DATO DE MEMORIA DE PROGRAMA APUNTADO POR Z
     SWAP   A          ; INTERCAMBIA EL NIBBLE MAS SIGNIFICATIVO
                           ; CON EL NIBBLE MEMOS SIGNIFICATIVO Y VICEVERSA
     ANDI   A,0xF0          ;  FILTRA EL NIBBLE MAS SIGNIFICATIVO
            STS    LCDDR10,A     ; COPIA EN LCDDR10 EL VALOR QUE CONTIENE A

     LPM    A,Z                ; CARGA EN A EL DATO DE MEMORIA DE
                                    ;  PROGRAMA APUNTADO POR Z
     ANDI   A,0xF0          ;   FILTRA EL NIBBLE MAS SIGNIFICATIVO
            STS     LCDDR15,A    ;  COPIA EN LCDDR15 EL VALOR QUE CONTIENE A
     RET

;                                        ----------------------------------------------------
;=========================| CONFIGURA EL MODO DE CONTEO |======================
;                                        ----------------------------------------------------

/* RUTINA CON LA CUAL SE VERIFICA EL SENTIDO DE CONTEO Y SE EVALUA SI LLEGO A SU
VALOR
  MAXIMO (99) O MINIMO (00) PARA INICIAR NUEVAMENTE EL CONTEO SEGUN EL SENTIDO
*/

MODO_CONTEO:

          CPI       UP_DOWN,0         ; ¿UP_DOWN ES IGUAL A 0?
   BREQ   MODO_DOWN      ;   SI
   RJMP   MODO_UP            ;   NO



;---------- MODO DE CONTEO DESCENDENTE -----------

MODO_DOWN:

   CPI      UNIDAD,0               ; ¿UNIDAD ES IGUAL A 0?
   BREQ   UNIDAD_DOWN      ;   SI
   DEC      UNIDAD                ;   NO
            RJMP    SALIR_MODO         ;   NO

UNIDAD_DOWN:

          LDI     UNIDAD,9                ; CARGA UNIDAD CON EL VALOR 9
          CPI     DECENA,0               ;  ¿DECENA ES IGUAL A 0?
          BREQ  DECENA_DOWN        ;    SI
```

```
        DEC   DECENA                ;   NO
        RJMP  SALIR_MODO            ;   NO

DECENA_DOWN:

          LDI    DECENA,9     ;CARGA DECENA CON EL VALOR 9
   RJMP SALIR_MODO

;---------- MODO DE CONTEO ASCENDENTE ----------

MODO_UP:

  CPI     UNIDAD,9          ; ¿UNIDAD ES IGUAL A 9?
  BREQ   UNIDAD_UP      ;   SI
         INC     UNIDAD           ;   NO
        RJMP   SALIR_MODO    ;   NO

UNIDAD_UP:

        LDI    UNIDAD,0          ;  CARGA UNIDAD CON EL VALOR 0
        CPI    DECENA,9          ; ¿DECENA ES IGUAL A 9?
        BREQ  DECENA_UP       ;   SI
        INC    DECENA           ;  NO
         RJMP  SALIR_MODO      ;   NO

DECENA_UP:

         LDI  DECENA,0

;---------- SALIDA DE LA RUTINA ----------

SALIR_MODO:

     RET

;                                        ----------------------------------------------
;==============================| TABLA PARA EL SEGMETO LCD |=======================
;                                         ----------------------------------------------

/* TABLA EN LA CUAL SE TIENE EL VALOR QUE SE DEBE CARGAR EN UN SOLO NIBBLE DE CADA
  UNO DE LOS 4 REGISTROS QUE CONTROLAN UN SEGMENTO DE LCD, PARA PODER OBTNER EL
  RESPECTIVO NUMERO EN LA PANTALLA
*/

LCD_TABLA:
;          mpnd  legc   jfhb   k    a -------> SEGMENTO LCD
 .DW 0b_0001_0101_0101_0001 ;0             -----a-----
 .DW 0b_0010_0000_1000_0000 ;1          | \    |   / |
 .DW 0b_0001_1110_0001_0001 ;2                f   h   j   k  b
 .DW 0b_0001_1011_0001_0001 ;3          |    \ | /   |
 .DW 0b_0000_1011_0101_0000 ;4          ---g--  ---l---
 .DW 0b_0001_1011_0100_0001 ;5          |    / | \   |
 .DW 0b_0001_1111_0100_0001 ;6          e   p  n  m  c
 .DW 0b_0000_0001_0101_0001 ;7          | /    |   \ |
 .DW 0b_0001_1111_0101_0001 ;8            -----d-----
 .DW 0b_0001_1011_0101_0001 ;9

;================================================================================
```

4.1.5 Simulación: Contador UP/DOWN

Como se indicó en el capítulo 3, el programa cargado en el AVR Butterfly cuenta de forma progresiva o regresiva de acuerdo al botón presionado, el valor es mostrado en la pantalla LCD. El conteo se realiza del 0 al 99.

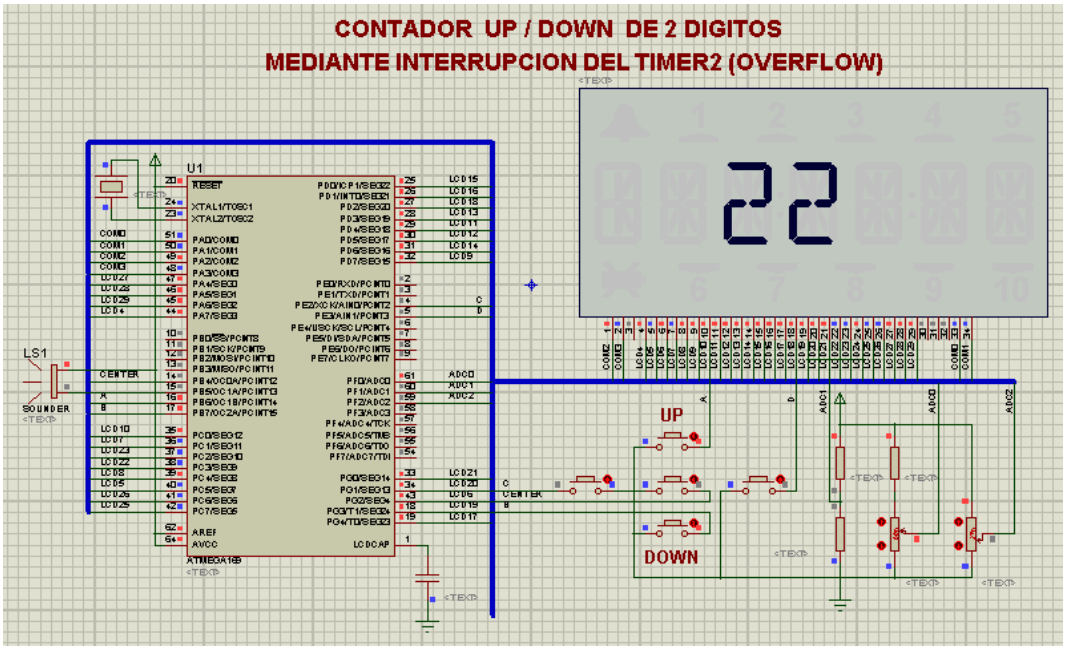


Figura 4.2 Esquemático: Contador UP/DOWN

4.2. Programa: Motor de 3 velocidades

El siguiente programa esta implementado en lenguaje de ensamblador y consiste en un motor de 3 velocidades, la velocidad del motor es controlada por el joystick, cuando se presiona hacia arriba se incrementa la velocidad y si se presiona en el medio la velocidad disminuye.

4.2.1 Diagrama de Bloques: Motor de 3velocidades

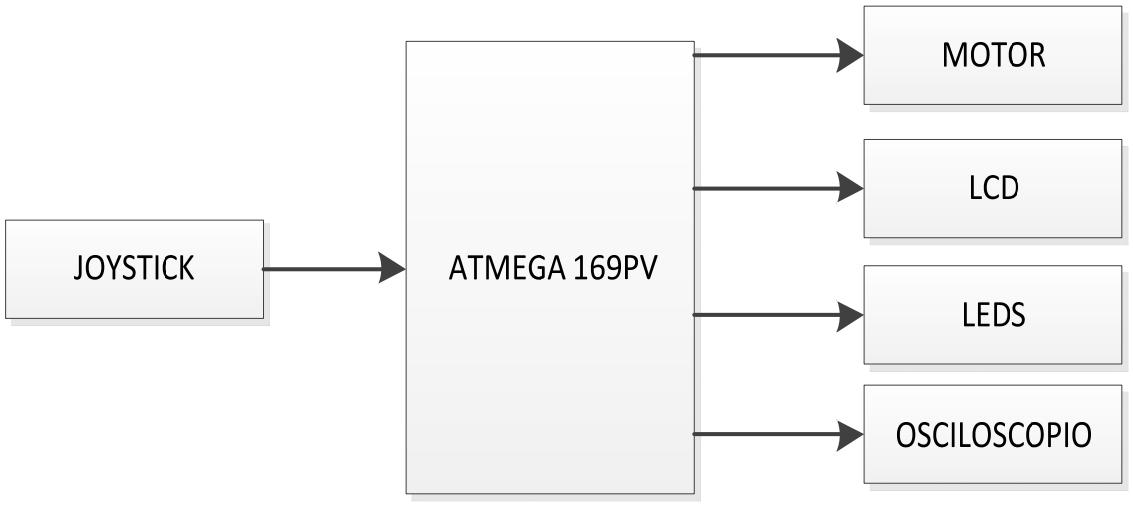
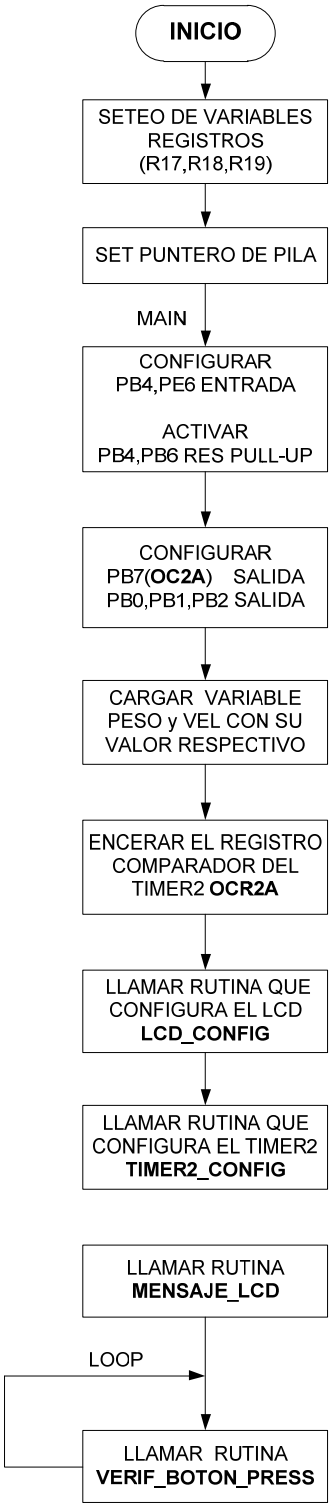
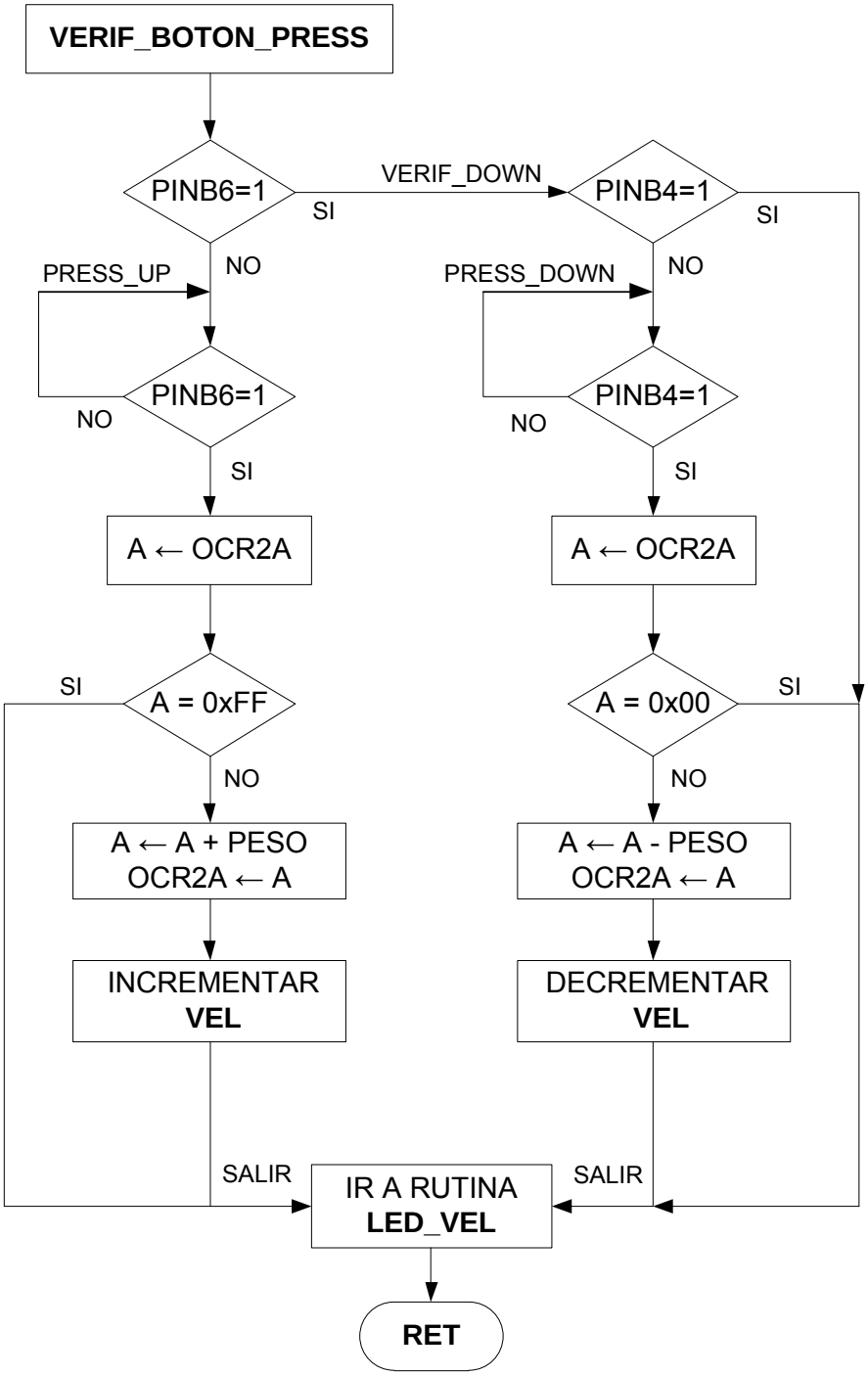


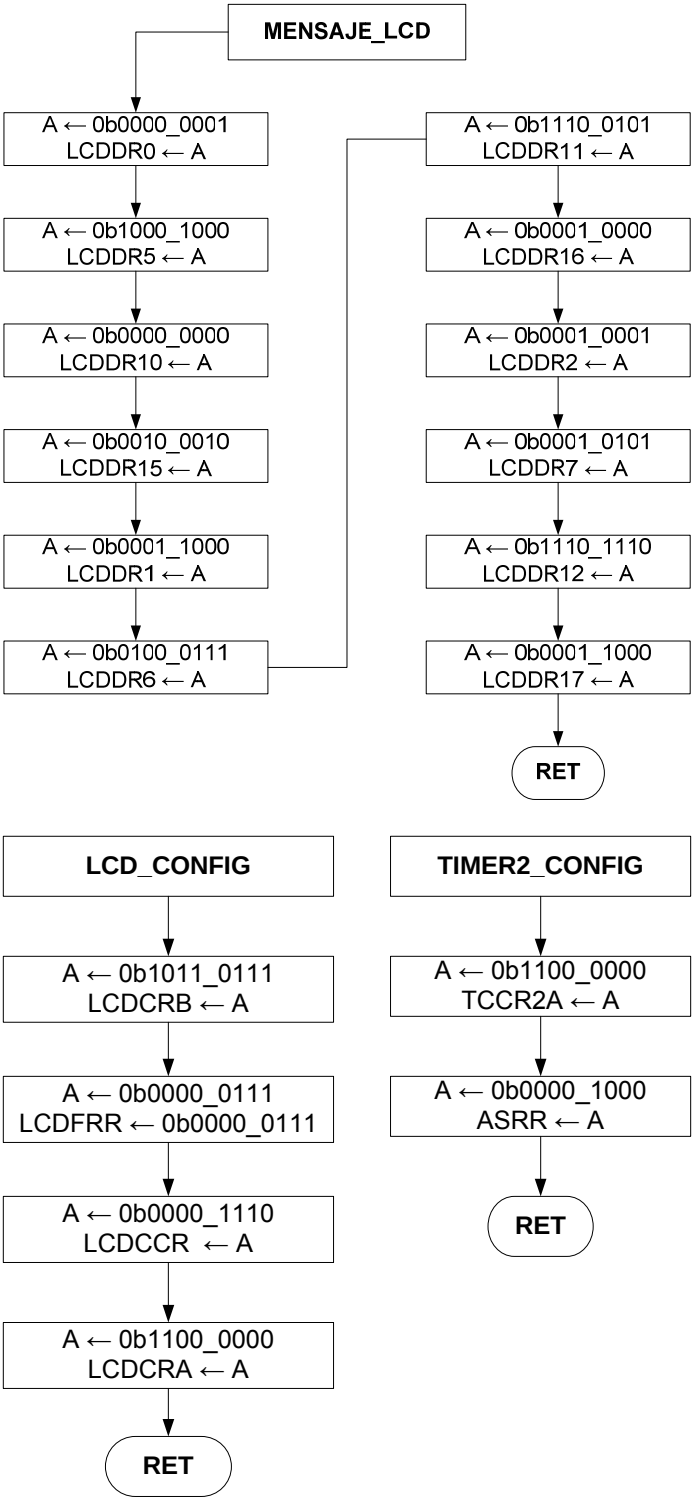
Figura 4.3 Diagrama de Bloques: Motor de 3 velocidades

4.2.2 Diagrama de flujo : Motor de 3 velocidades.



4.2.3 Diagrama de flujo : Rutinas Motor de 3 velocidades.




MENSAJE_LCD
A ← 0b0000_0001
LCDDR0 ← A
A ← 0b1000_1000
LCDDR5 ← A
A ← 0b0000_0000
LCDDR10 ← A
A ← 0b0010_0010
LCDDR15 ← A
A ← 0b0001_1000
LCDDR1 ← A
A ← 0b0100_0111
LCDDR6 ← A
A ← 0b1110_0101
LCDDR11 ← A
A ← 0b0001_0000
LCDDR16 ← A
A ← 0b0001_0001
LCDDR2 ← A
A ← 0b0001_0101
LCDDR7 ← A
A ← 0b1110_1110
LCDDR12 ← A
A ← 0b0001_1000
LCDDR17 ← A
RET
LCD_CONFIG
A ← 0b1011_0111
LCDCRB ← A
A ← 0b0000_0111
LCDFRR ← 0b0000_0111
A ← 0b0000_1110
LCDCCR ← A
A ← 0b1100_0000
LCDCRA ← A
RET
TIMER2_CONFIG
A ← 0b1100_0000
TCCR2A ← A
A ← 0b0000_1000
ASRR ← A
RET

4.2.4 Programa principal en lenguaje ensamblador.

```
/*****************************************************************************

MICROCONTROLADORES AVANZADOS

INTEGRANTES:    RAUL BEJARANO
                CHRISTIAN BLANCO
GRUPO:  4

PROGRAMA:   MOTOR DE 3 VELOCIDADES

DESCRIPCION:    EL SIGUIENTE PROGRAMA ES UN MOTOR DC  CON 3 VELOCIDADES
LAS CUALES SE LAS          VA CAMBIANDO HACIENDO USO DE JOSTICK DONDE SI SE
PRESIONA HACIA LA DERECHA SE AUMENTA LA VELOCIDAD Y SI SE PRESIONA A LA
IZQUIERDA DISMINUYE  LA VELOCIDAD

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

.INCLUDE "M169DEF.INC"    ;   DEFINICION BUTTERFLY

;                                       ------------------------------------------------
;=========================| DECLARACION DE VARIABLES |==============
;                                        ------------------------------------------------

.DEF        A = R17   ; VARIABLE A ES EL REGISTRO R17
.DEF PESO = R18   ;  VARIABLE PESO ES EL REGISTRO R18
.DEF     VEL = R19   ; VARIABLE VEL ES EL REGISTRO R19

.ORG $0000           ; STARTUP VECTOR
 RJMP INICIO
;                                   ------------------------------------------------
;======================| SETEO DEL PUNTERO DE PILA |==================
;                                    ------------------------------------------------
INICIO:

  LDI A,LOW(RAMEND)
  OUT SPL,A
  LDI A,HIGH(RAMEND)
  OUT SPH,A
;                                                 ----------------------
;==============================| RUTINA MAIN |============================
;                                                  ----------------------
MAIN:

  LDI     A,0x87
          OUT   DDRB,A                ;  PB0,PB1,PB2,PB7 COMO SALIDA
  SBI   PORTB,6
  SBI   PORTB,4                 ; HABILITA RESISTENCIA PULL-UP EN PB6 Y PB4
  LDI    PESO,85                ;  VARIABLE PARA INCREMENTAR VELOCIDAD
  LDI    VEL,0                   ;  VARIABLE PARA ENCENDIDO DE LOS LEDS
  LDI    A,0                     ;  CARGA A CON 0
  STS   OCR2A,A               ;  ENCERA EL REGISTRO COMPARADOR DE TIMER2
  RCALL LCD_CONFIG       ;  LLAMA A LA RUTINA LCD_CONFING
  RCALL TIMER2_CONFIG  ;  LLAMA  A LA RUTINA TIMER2_CONFING

    LOOP:
```

```
        RCALL VERIF_BOTON_PRESS  ;  LLAMA A LA RUTINA VERIF_BOTON_PRESS
  RCALL MENSAJE_LCD              ;  LLAMA A LA RUTINA MENSAJE_LCD

    RJMP LOOP

;=============================================================================

;                                    ---------------------------------------
;  ======================| RUTINA CONFIRAR TIMER2 |===================
;                                    ---------------------------------------
TIMER2_CONFIG:

      LDI   A,0b0110_0101
      STS TCCR2A,A     ; OC2A MODO DE OPERACION 3
                       ; TIMER PRESCALE=128
                       ; MODO DE OPERACION PWM PHASE CORRRECT
      LDI A,8
      STS ASSR,A         ;RELOJ EXTERNO XLAT 32768HZ
      RET

;                                     ---------------------------------------
;=========================| RUTINA CONFIGURAR LCD |===================
;                                     ---------------------------------------
LCD_CONFIG:

      LDI A,0b1011_0111   ; SETEO DEL RELOJ, CICLO DE TRABAJO Y NUMERO DE PINES
A USAR
      STS LCDCRB, A       ; HABILITA TODOS LOS SEGEMENTOS
      ;LCDCRB: [LCDCS,LCDB2,LCDMUX1,LCDMUX0,_,LCDPM2,LCDPM1,LCDPM0]

      LDI A,0b0000_0111  ; SETEO DEL FRAME RATE A 32Hz
      STS LCDFRR, A      ; LCDFRR:
[_,LCDPS2,LCDPS1,LCDPS0,_,LCDCD2,LCDCD1,LCDCD0]

      LDI A,0b0000_1110  ; (LCDCC3,LCDCC2,LCDCC1)
      STS LCDCCR, A      ; SETEO DEL CONTRASTE DE LA PANTALLA

      LDI A,0b1100_0000  ; (LCDEN,LCDAB)
      STS LCDCRA, A      ; ENCIENDE EL LCD
      RET
;                    ------------------------------------------------------------------------------------
;=============| RUTINA VERIFICAR BOTON DE JOSTICK PRESIONADO |====
;                    ------------------------------------------------------------------------------------
VERIF_BOTON_PRESS:

     VERIF_UP:
            SBIS    PINB,6            ;  ¿PRESIONO BOTON UP?
            RJMP  PRESS_UP       ;    SI , SALTA A LA ETIQUETA PRESS_UP
            RJMP  VERIF_DOWN   ;   NO , SALTA A LA ETIQUETA VERIF_DOWN
     PRESS_UP:
            SBIS    PINB,6          ;  ¿DEJO DE PRESIONAR EL BOTON UP?
            RJMP  PRESS_UP    ;     NO , SALTA A LA ETIQUETA PRESS_UP
            LDS     A,OCR2A       ;  CARGA EN A EL VALOR DEL REGISTRO OC2RA
      CPI     A,0xFF          ;  ¿ES A IGUAL A 0xFF?
      BREQ  SALIR           ;   SI , SALTA A LA ETIQUETA SALIR
            ADD    A,PESO         ;  A=A+PESO
      STS     OCR2A,A        ;  CARGA EN OCR2A EL VALOR DE A
            INC     VEL             ;  INCREMENTA VEL (ENCIENDE LED)
```

```
            RJMP  SALIR            ;  SALTA A LA ETIQUETA SALIR
      VERIF_DOWN:
            SBIS PINB,4              ;  ¿PRESIONO BOTON DOWN?
            RJMP PRESS_DOWN     ;  SI  ,  SALTA A LA ETIQURTA PRESS_DOWN
            RJMP SALIR               ;  NO , SALTA A LA ETIQUETA SALIR
       PRESS_DOWN:
            SBIS    PINB,4           ;  ¿DEJO DE PRESIONAR EL BOTON UP?
        RJMP  PRESS_DOWN     ;  NO , SALTA A LA ETIQUETA PRESS_DOWN
        LDS     A,OCR2A          ;  CARGA EN A EL VALOR DEL REGISTRO OC2RA
        CPI      A,0                ;  ¿ES A IGUAL A 0x00?
        BREQ  SALIR              ;   SI , SALTA A LA ETIQUETA SALIR
            SUB     A,PESO           ;   A=A-PESO
        STS     OCR2A,A          ;  CARGA EN OCR2A EL VALOR DE A
        DEC     VEL                ;   DECREMENTA VEL (ENCIENDE LED)
            RJMP   SALIR             ;  SALTA A LA ETIQUETA SALIR

        SALIR:
            RCALL LED_VEL    ; ENCENDER EL RESPECTIVO LED DE VELOCIDAD
         RET
;                                   ----------------------------------------------------
;======================| RUTINA MENSAJE EN EL LCD |===================
;                                   ----------------------------------------------------
  ; Se cargan los respectivos valores en los registros que manejan los segmentos del
                        ; LCD  para poner en la pantalla el mensaje  timer2

 MENSAJE_LCD:

   ;         T     I
  LDI  A,0b0000_0001
  STS LCDDR0,A
  LDI  A,0b1000_1000
  STS LCDDR5,A
  LDI  A,0b0000_0000
  STS LCDDR10,A
  LDI  A,0b0010_0010
  STS LCDDR15,A

  ;       M     E                 SEGMENTO DEL LCD
  LDI  A,0b0001_1000   ;           ------a------
  STS LCDDR1,A         ;          | \   |    / |
  LDI  A,0b0100_0111   ;          f   h   j   k   b
  STS LCDDR6,A         ;          |  \  |  /   |
  LDI  A,0b1110_0101   ;           --g--- ---l ---
  STS LCDDR11,A        ;          |   / |  \   |
  LDI  A,0b0001_0000   ;          e  p  n  m  c
  STS LCDDR16,A        ;          | /   |   \  |
                              ;          -----d-------

  ;       R     2
  LDI A,0b0001_0001
  STS LCDDR2,A
  LDI A,0b0001_0101
  STS LCDDR7,A
  LDI A,0b1110_1110
  STS LCDDR12,A
  LDI A,0b0001_1000
  STS LCDDR17,A
```

```
 RET
;                          ------------------------------------------------------
;====================| ENCENDER LED DE LA VELOCIDAD |================
;                          ------------------------------------------------------

 LED_VEL:

  CPI      VEL,0             ;¿ VEL IGUAL 0 ?
  BREQ   APAGAR_LEDS  ;  SI
  CPI      VEL,1             ;  NO, ¿ VEL IGUAL 1 ?
  BREQ   LED_VEL1      ;  SI
  CPI      VEL,2             ;  NO, ¿ VEL IGUAL 1 ?
  BREQ   LED_VEL2      ;  SI
  RJMP   LED_VEL3      ;  NO, SALTA A ETIQUETA

  APAGAR_LEDS:

  ; APAGA LED PB0,PB1,PB2
  CBI PORTB,0
  CBI PORTB,1
  CBI PORTB,2
  RJMP SALIDA

  LED_VEL1:

  ; ENCIENDE LED PB0 -  APAGA LED PB1;PB2
  SBI PORTB,0
  CBI PORTB,1
  CBI PORTB,2
  RJMP SALIDA

  LED_VEL2:

  ; ENCIENDE LED PB1 -  APAGA LED PB0,PB2
  SBI PORTB,1
  CBI PORTB,0
  CBI PORTB,2
  RJMP SALIDA

 LED_VEL3:

 ; ENCIENDE LED PB2 -  APAGA LED PB0,PB1
 SBI PORTB,2
 CBI PORTB,0
 CBI PORTB,1
 RJMP SALIDA

SALIDA:

      RET
;=====================================================================
```

4.2.5 Simulación: Motor 3 velocidades

El programa cargado en el AVR Butterfly controla las 3 velocidades del motor a través del joystick, cuando se presiona hacia la arriba se incrementa la velocidad y si se presiona en medio la velocidad disminuye.

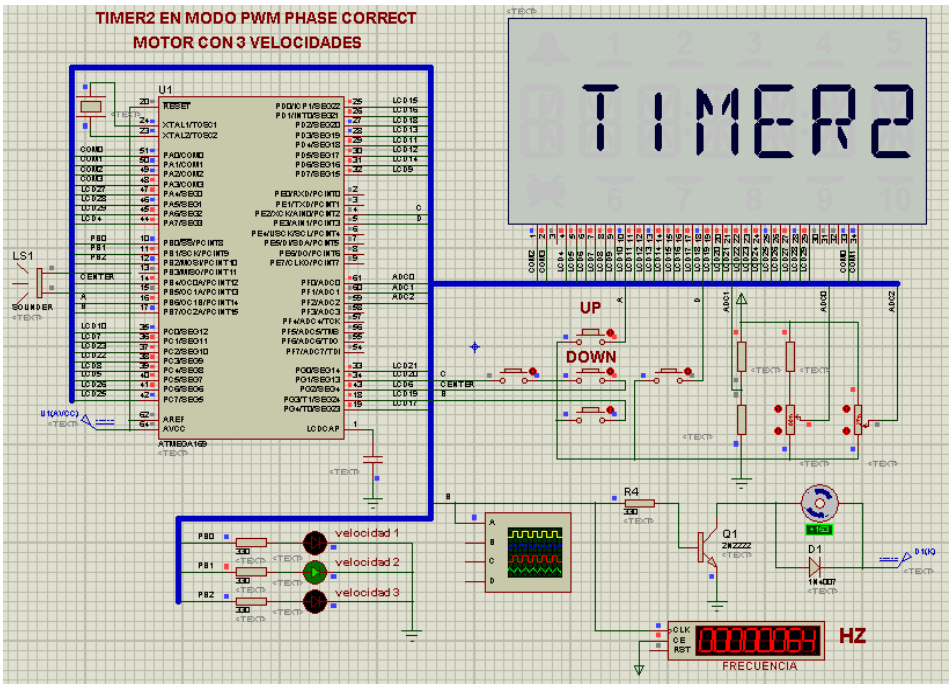


Figura 4.4 Esquemático: Motor de 3 velocidades

4.3. Programa: Encender/Apagar Led cada segundo utilizando interrupción por comparación

El siguiente programa esta implementado en lenguaje de C y consiste en encender o apagar un led cada 1 segundo haciendo uso de las interrupciones por comparación y por desborde del timer2.

4.3.1 Diagrama de Bloques: Encender/Apagar Led cada segundo utilizando interrupción por comparación

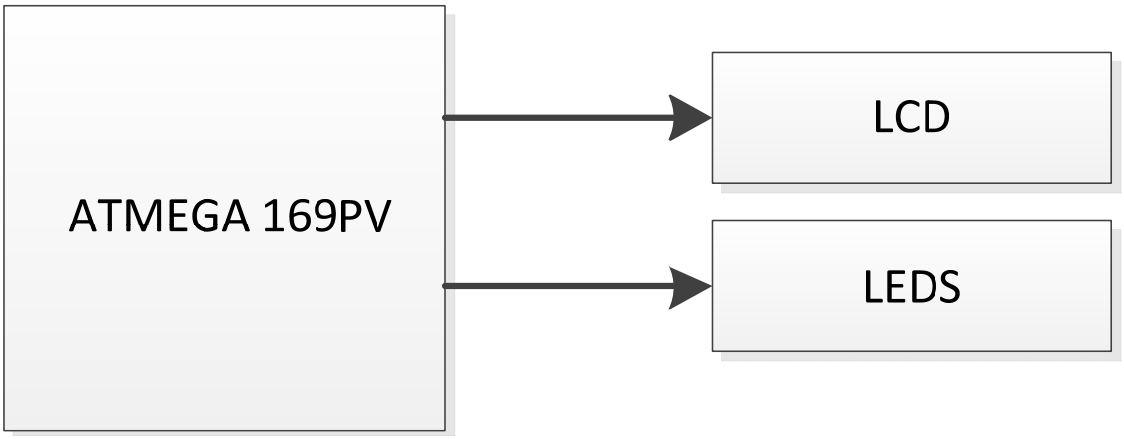
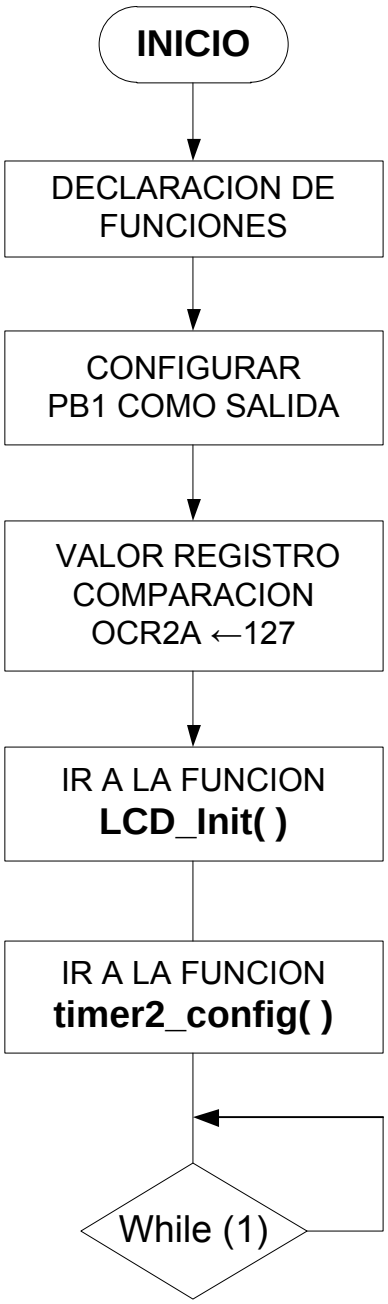
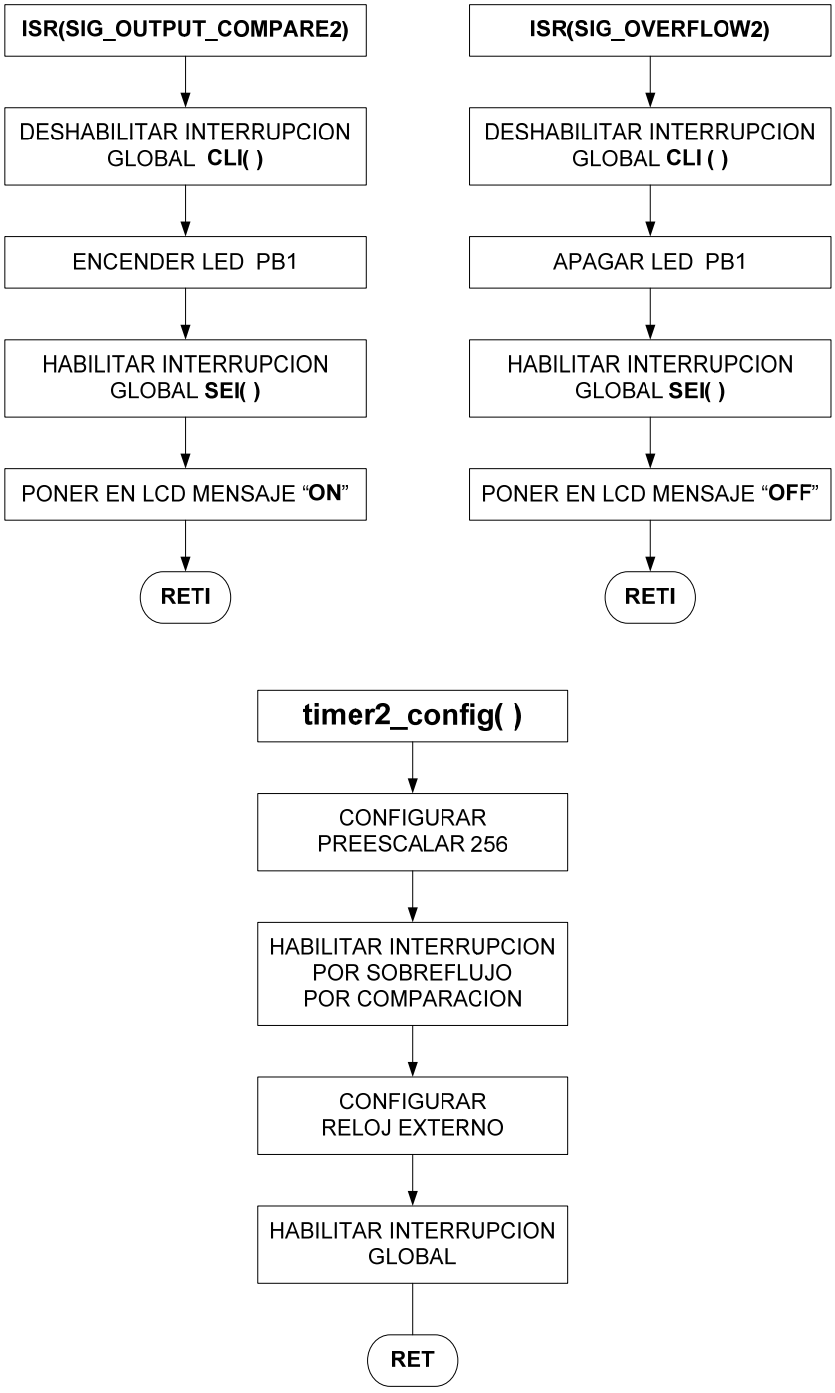


Figura 4.5 Diagrama de Bloques Encender/Apagar Led cada segundo utilizando interrupción por comparación

4.3.2 Diagrama de Flujo: Encender/Apagar Led cada segundo utilizando interrupción por comparación



4.3.3 Diagrama de flujo : Rutinas Encender/Apagar Led cada segundo utilizando interrupción por comparación



4.3.4 Programa principal en lenguaje C.

```
/*****************************************************************

                    MICROCONTROLADORES AVANZADOS

  INTEGRANTES:      RAUL BEJARANO
                    CHRISTIAN BLANCO

  GRUPO # 4

  PROGRAMA: LED ON-OFF

  DESCRIPCION: EL PRORAMA CONSISTE EL ENCENDER O APAGAR UN LED CADA 1
 SEGUNDO HACIENDO USO DE LAS INTERRUPCIONES POR COMPARACION Y POR
 DESBORDEDEL TIMER2


*****************************************************************/
 #include <avr/io.h>
 #include <avr/interrupt.h>
 #include "lcd_driver.h"

/*                              ------------------------------------------------
=======================| DECLARACION DE FUNCIONES |=================
                               ------------------------------------------------
*/
 void timer2_config();  // Función para configurar el Timer2


/*                          ---------------------------------------------------------------------
================| FUNCIONES DE VECTORES DE INTERRUPCION |========
                          ---------------------------------------------------------------------
*/

 ISR(SIG_OUTPUT_COMPARE2) // Interrupción por comparación OCR2A = TCNT2
 {
  cli();    // Deshabilita la interrupción global

   PORTB =(1<<PB1); // Enciende el Led
   sei();  // Habilita la interrupción global
   LCD_puts(" ON");    // Presentar mensaje ON en el LCD del Butterfly
 }

 ISR(SIG_OVERFLOW2)  // Interrupción por desborde(OVERFLOW) TCNT2= 0xFF
 {
  cli();    // Deshabilita la interrupción global
```

```
   PORTB &=(0<<PB1);  // Apaga el Led
   sei();   // Habilita la interrupción global

   LCD_puts(" OFF");      // Presentar mensaje OFF en el LCD del Butterfly

   sei();   // Habilita la interrupción global
 }
/*                                          ----------------------
==============================| RUTINA MAIN |=========================
                                            ----------------------
*/
 int main(void)
 {

  DDRB   = 0X02;      // PINB1 como salida
  OCR2A  = 127;       // Valor de la comparación
  LCD_Init();          //  Función Configurar LCD
  timer2_config();     //  Función Configurar del Timer2

  while(1)
   {
       // Espera por interrupción
   }
 }
//=============================================================================
=====
/*                                 ------------------------------------------------------
=======================| CONFIGURARCION DEL TIMER2 |===============
                                   ------------------------------------------------------
*/
 void timer2_config()
 {
  TCCR2A = (6<<CS20);  // Timer2 en modo NORMAL – Pre-escalar a 256

  TIMSK2 = (1<<OCIE2A |1<<TOIE2);
              // Habilita la interrupción por comparación (OCR2A = TCNT2)
              // Habilita la interrupción por desborde (overflow)

  ASSR =(1<<AS2);  // Reloj externo XLAT 32.768 Hz

  sei();     // Habilita la interrupción global

  //Tiempo para ocurre la interrupción por Comparación
  // tcomp= (32.768/N)/(OCR2A+1)  ;  N : Pre-escalar

 }
//=============================================================================
```

4.3.5 Simulación: Encender/Apagar Led cada segundo utilizando interrupción por comparación

Como se indicó en el capítulo 3, el programa cargado en el AVR Butterfly apaga y enciende el led conectado al pin PB1 cada segundo, utilizando la interrupción por comparación y por desborde del Timer 2.

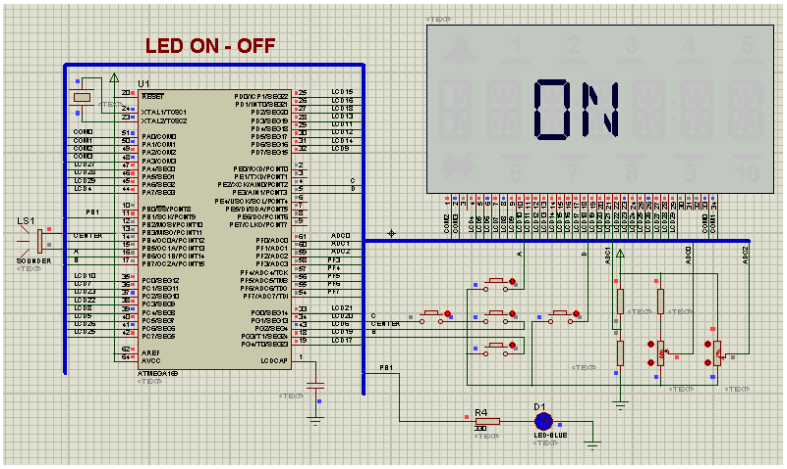


Figura 4.6 Esquemático: Encender/Apagar (Estado ON)

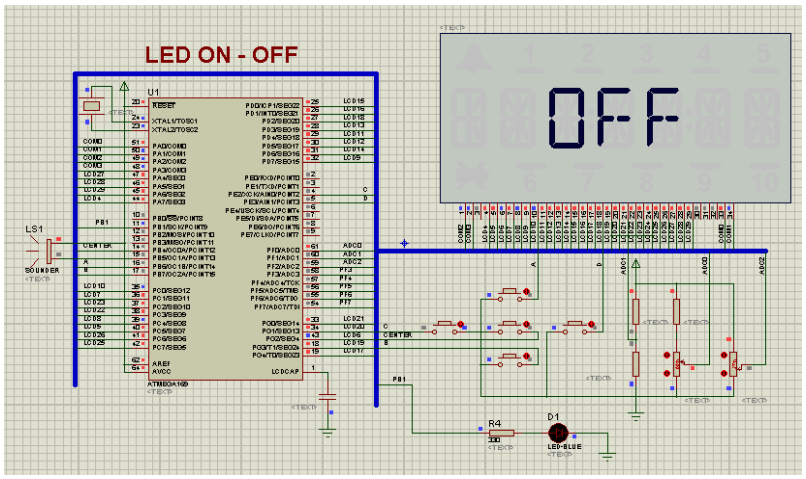


Figura 4.7 Esquemático: Encender/Apagar (Estado OFF)

4.4. Programa: Onda cuadrada de frecuencia variable

El siguiente programa esta implementado en lenguaje C y consiste en generar una onda cuadrada por el pin OC2A. Se hace variar la frecuencia en función de la conversión ADC de un voltaje controlado por un potenciómetro conectado al pin ADC4 y se muestra el valor de OCR2A en el LCD para realizar el respectivo cálculo de la frecuencia de oscilación de la onda cuadrada generada en el pin OC2A.

4.4.1 Diagrama de Bloques: Onda cuadrada de frecuencia variable

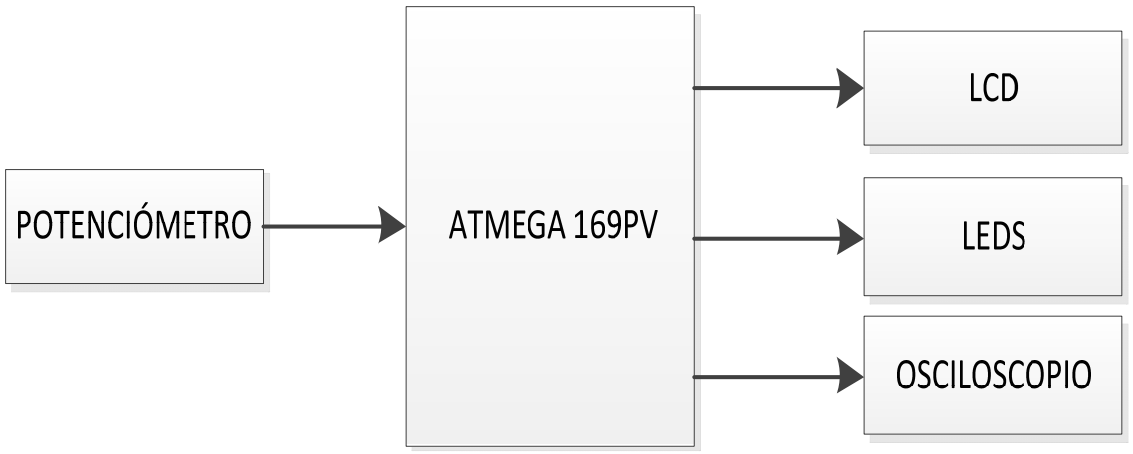
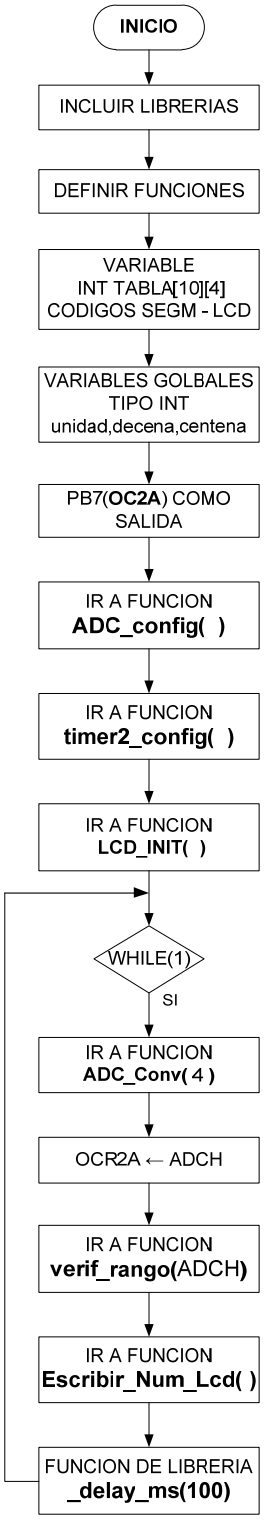
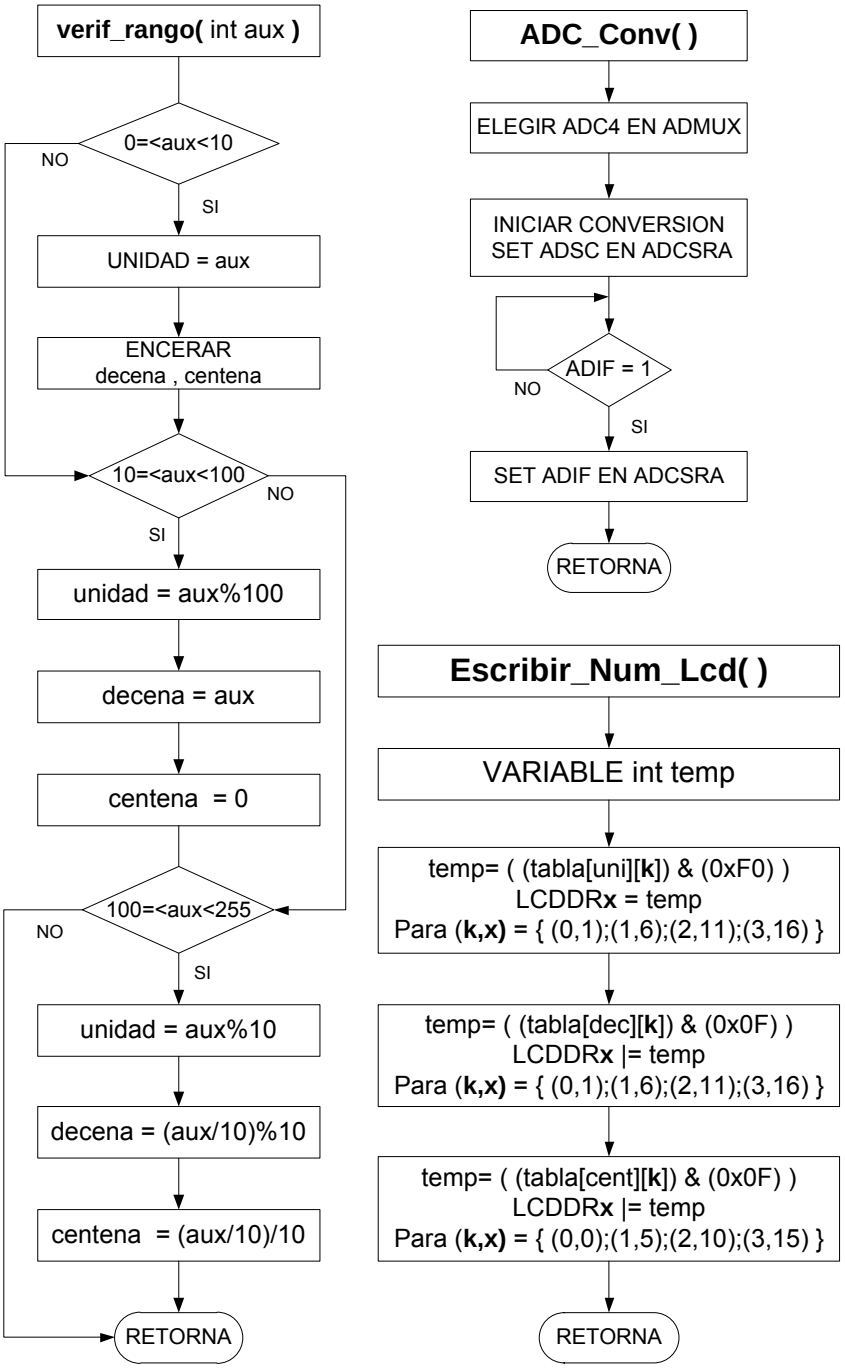


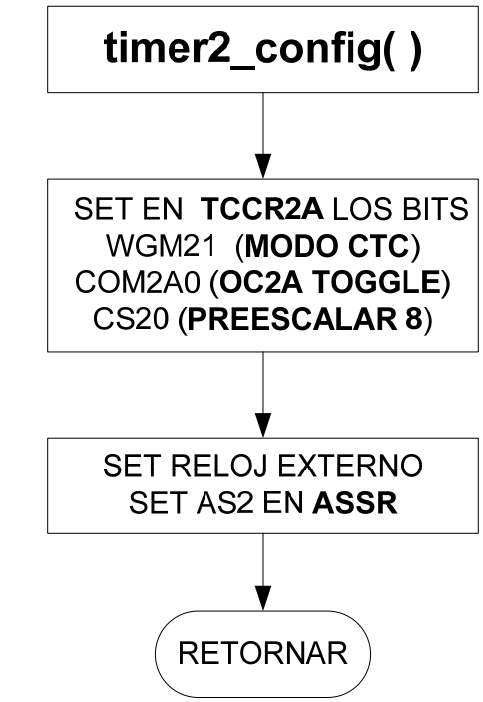
Figura 4.8 Diagrama de Bloques: Onda cuadrada de frecuencia variable

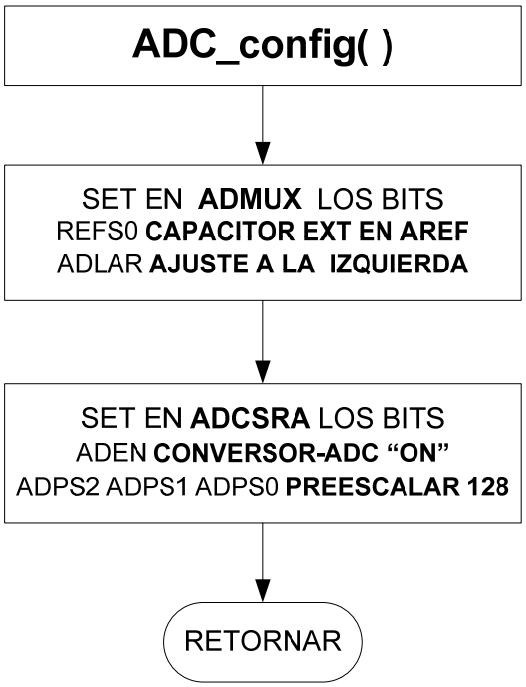
4.4.2 Diagrama de Flujo: Onda cuadrada de frecuencia variable



4.4.3 Diagrama de Flujo: Rutinas Onda cuadrada de frecuencia variable




timer2_config( )
SET EN TCCR2A LOS BITS
WGM21 (MODO CTC)
COM2A0 (OC2A TOGGLE)
CS20 (PREESCALAR 8)
SET RELOJ EXTERNO
SET AS2 EN ASSR
RETORNAR


ADC_config( )
SET EN ADMUX LOS BITS
REFS0 CAPACITOR EXT EN AREF
ADLAR AJUSTE A LA IZQUIERDA
SET EN ADCSRA LOS BITS
ADEN CONVERSOR-ADC "ON"
ADPS2 ADPS1 ADPS0 PREESCALAR 128
RETORNAR

4.4.4 Programa principal en lenguaje C: Onda cuadrada de frecuencia variable

```
/*****************************************************************
                    MICROCONTROLADORES AVANZADOS

INTEGRANTES:      RAUL BEJARANO
                  CHRISTIAN BLANCO

GRUPO # 4

PROGRAMA: ONDA CUADRADA DE FRECUENCIA VARIABLE

DESCRIPCION: SE GENERA UNA ONDA CUADRADRA POR EL PIN OC2A. SE HACE
VARIAR LA FRECUENCIA EN FUNCION DE LA  CONVERSION ADC DE UN VOLTAJE
CONTROLADO POR UN POTENCIOMETRO EN EL PIN ADC4 Y SE MUESTRA EL VALOR
DE OCR2A EN EL LDC PARA REALIZAR EL RESPECTIVO CALCULO DE LA FRECUENCIA
DE OSCILACION DE LA ONDA CUADRADA GENERADA EN EL PIN OC2A

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

#include <avr/io.h>
#include <util/delay.h>

/*                          ---------------------------------------------
==================| DECLARACION DE FUNCIONES |================
                          ---------------------------------------------
*/
 void timer2_config();                  //  Función para configurar el Timer2
 void ADC_config(void);                 //  Función para configurar el ADC
 void ADC_Conv(unsigned char ch); //  Función para realizar conversión ADC
 void LCD_INIT();                       //  Función para configurar el LCD
 void verif_rango(int aux);             //  Función para verificar rango de OCR2A
                                        //  y realiza conversión BCD

 void Escribir_Num_Lcd(int uni,int dec,int cent);  //  Función para escribir en LCD
                                                   //  el valor de OCR2A

/*                                  ------------------------
=========================| TABLA DE LCD |=======================
                                  ------------------------
*/
  int tabla[10][4]= { 0x11,0x55,0x55,0x11,   // cero
                      0x00,0x88,0x00,0x22,   // uno
                            0x11,0x11,0xEe,0x11,   // dos
                      0x11,0x11,0xBb,0x11,   // tres
                                0x00,0x55,0xBb,0x00,   // cuatro
```

```
                    0x11,0x44,0xBB,0x11,   // cinco
                    0x11,0x44,0xFF,0x11,   // seis
                    0x11,0x11,0x11,0x00,   // siete
                    0x11,0x55,0xFF,0x11,   // ocho
                    0x11,0x55,0xBB,0x00};  // nueve

/*                                  ---------------------------------
==========================| VARIABLES GOLBALES |=================
                                    ---------------------------------
            Las variables declaradas a continuación son usadas para
           guardar los resultados de la conversión de OCR2A a BCD

*/
 int unidad;
 int decena;
 int centena;

/*                                   ---------------------
============================| RUTINA MAIN |=========================
                                     ---------------------
*/
 int main(void)
 {

   ADC_config();    //  configurar el ADC
   timer2_config();         //  configurar del Timer2
   LCD_INIT();
   while(1)
   {
   ADC_Conv(4);       // realizar conversión ADC
   OCR2A=ADCH;    // valor de comparación del timer2

   verif_rango(ADCH); // va la función veri_rango
          Escribir_Num_Lcd(unidad, decena, centena);
                                  // va a la función Escribir_Num_Ldc
          _delay_ms(100);
    }
 }
//=============================================================/*
-                                 ---------------------------------------------
====================| CONFIGURARCION DEL TIMER2 |=============
                                   -----------------------------------------
*/

void timer2_config()
{
   TCCR2A = (1<<WGM21|0<<WGM20 |0<<COM2A1 | 1<<COM2A0 |
2<<CS20);
   /* Timer2 en modo CTC
```

```
    OC2A como Toggle cuando TCNT2=OCR2A
   Prescalar a 8
  */
  DDRB   = 0X80;         // OC2A/PB7  como salida
  ASSR    = (1<<AS2);   // Fuente de reloj externa XLAT 32.768 Hz
}

/*                          ----------------------------------------
==================| CONFIGURACION DEL ADC |======================
                            ----------------------------------------
*/

 void ADC_config(void)
 {
  ADMUX|=(1<<REFS0);   // AVcc con capacitor externo en AREF
  ADMUX|=(1<<ADLAR);   // ajuste a la Izquierda
  ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
                              // Enciende el convertidor ADC
                              // Configura el Preescalar a 128
 }

/*                                  ----------------------------
========================| CONVERSION ADC |======================
                                  ----------------------------
*/
 void ADC_Conv(unsigned char ch)
 {
    ch= ch & 0b00000111;                 //  Indica cual ADC elegimos
    ADMUX |= ch;                              //  Selección del ADC en este caso
ADC4
    ADCSRA|=(1<<ADSC);                   //  Inicia la conversión
    while(!(ADCSRA & (1<<ADIF)));      //  Espera por ADIF, conversión completa

    ADCSRA|=(1<<ADIF);                   //  Limpia ADFI poniendo 1 en este bit
 }

/*                              ------------------------------------------------
====================| VERIFICAR RANGO DEL OCR2A |================
                              ------------------------------------------------
 Función que se verifica en que rango esta el valor de OC2RA para luego realizar la
                                        conversión
  a BCD del valor de OCR2A y así obtener el valor de la unidad, decena y centena
                                        respectivamente
*/
void verif_rango(int aux)
 {

  // Se verifica si está entre 0 y 9
```

```
  if ((aux>=0) && (aux<10))
  {
            unidad  = aux;
      decena  = 0;
      centena = 0;
          }

   // Se verifica si está entre 10 y 99

  if ((aux>=10) && (aux<100))
          {
            unidad  = aux%10;
      decena  = aux/10;
            centena = 0;
          }

   // Se verifica si está entre 100 y 255

          if ((aux>=100) && (aux<255))
          {
            unidad  = aux%10;
            decena  = (aux/10)%10;
            centena = (aux/10)/10;
          }
 }

/*                              ------------------------------------------
====================| ESCRIBIR NUMEROS EN LCD |==================
                              ------------------------------------------
*/
 void Escribir_Num_Lcd(int uni, int dec, int cent)
 {
  int temp;
   // Se escribe la unidad en el LCD

  temp   = ( (tabla[uni][0]) & (0xF0) );
  LCDDR1 = temp; //carga en registro LCDDR1 el valor de temp
          temp   = ( (tabla[uni][1]) & (0xF0) );
  LCDDR6 = temp; //carga en registro LCDDR6 el valor de temp
          temp   = ( (tabla[uni][2]) & (0xF0) );
  LCDDR11= temp; //carga en registro LCDDR11 el valor de temp
          temp   = ( (tabla[uni][3]) & (0xF0) );
  LCDDR16= temp; //carga en registro LCDDR16 el valor de temp

    // Se escribe la decena en el LCD
          temp   = ( (tabla[dec][0]) & (0x0F) );
  LCDDR1 |= temp; //carga en registro LCDDR1 el valor de temp
          temp   = ( (tabla[dec][1]) & (0x0F) );
```

```
  LCDDR6 |= temp; //carga en registro LCDDR6 el valor de temp
         temp   = ( (tabla[dec][2]) & (0x0F) );
  LCDDR11|= temp; //carga en registro LCDDR11 el valor de temp
         temp   = ( (tabla[dec][3]) & (0x0F) );
  LCDDR16|= temp; //carga en registro LCDDR16 el valor de temp

    // Se escribe la centena en el LCD

  temp   = ( (tabla[cent][0]) & (0xF0) );
  LCDDR0 = temp; //carga en registro LCDDR0 el valor de temp
         temp   = ( (tabla[cent][1]) & (0xF0) );
  LCDDR5 = temp; //carga en registro LCDDR5 el valor de temp
         temp   = ( (tabla[cent][2]) & (0xF0) );
  LCDDR10= temp; //carga en registro LCDDR10 el valor de temp
         temp   = ( (tabla[cent][3]) & (0xF0) );
  LCDDR15= temp; //carga en registro LCDDR15 el valor de temp
 }

/*                                          ----------------------------
============================| CONFIGURAR LCD |====================
                                           ----------------------------
*/
 void LCD_INIT()

 {
  LCDCRB =(1<<LCDCS)|(3<<LCDMUX0)|(7<<LCDPM0) ;

  //  [LCDCS,LCD2B,LCDMUX1,+MUX0,-,LCDPM2,+PM1,+PM0]
  //   Reloj externo, 1/4 de ciclo de trabajo, 25 segmentos

  LCDFRR = (0<<LCDPS0)|(7<<LCDCD0);

 //   [-,LCDPS2,+PS1,+PS0,-,LCDCD2,+CD1,+CD0]
 //    Preescalar/16, clock divide to /8

  LCDCCR=(1<<LCDCC3)|(1<<LCDCC2)|(1<<LCDCC1);

 //   [LCDDC2,+DC1,+DC0,LCDMDT,LCDCC3,+CC2,+CC1,+CC0]
 //    Contraste  a 3.30 volts

  LCDCRA=(1<<LCDEN)|(1<<LCDAB);

  //   [LCDEN,LCDAB,-,LCDIF,LCDIE,LCDBD,LCDCCD,LCDBL]
  //     Enciende el LCD
 }

//=============================================================
```

4.4.5 Simulación: Onda cuadrada de frecuencia variable

El programa cargado en el AVR Butterfly genera una onda cuadrada por el pin OC2A. Se hace variar la frecuencia en función de la conversión ADC de un voltaje controlado por un potenciómetro en el pin ADC4 y se muestra el valor de OCR2A en el LCD y así realizar el respectivo cálculo de la frecuencia de oscilación de la onda cuadrada generada en el pin OC2A.

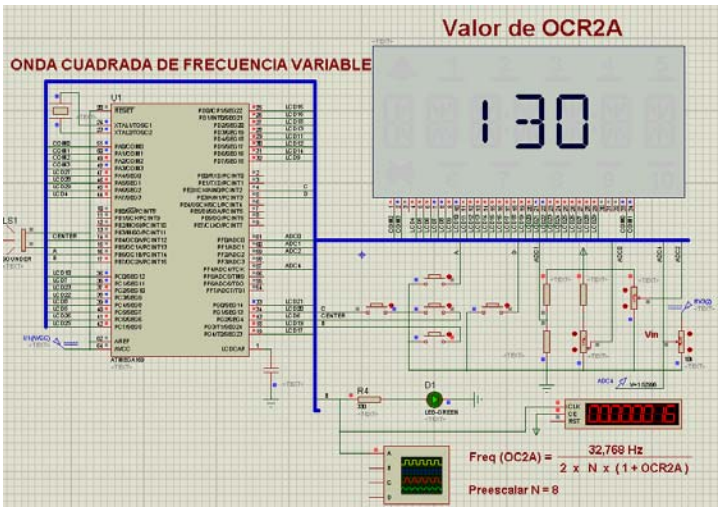


Figura 4.9 Esquemático: Onda cuadrada de frecuencia variable

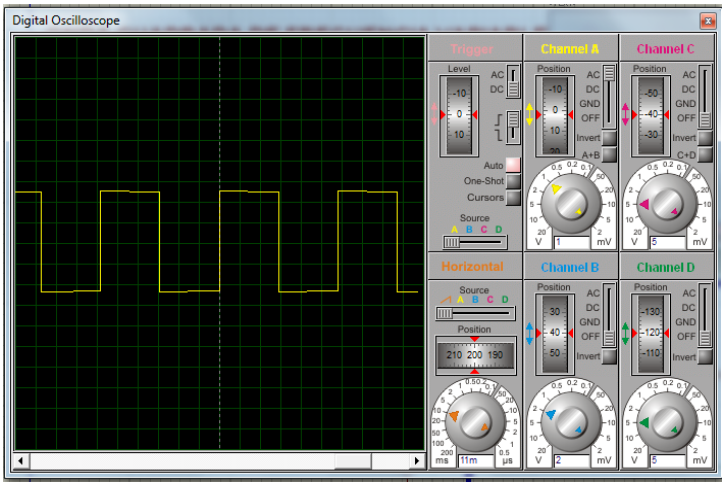


Figura 4.10 Señal cuadrada generada.

4.5. Programa: Control de velocidad motor DC

El siguiente programa esta implementado en lenguaje C y consiste en configurar el Timer 2 en modo FAST-PWM y la conversión ADC de un voltaje dado por un potenciómetro mediante el cual se va regulando la velocidad del motor DC.

4.5.1 Diagrama de Bloques: Control de velocidad motor DC

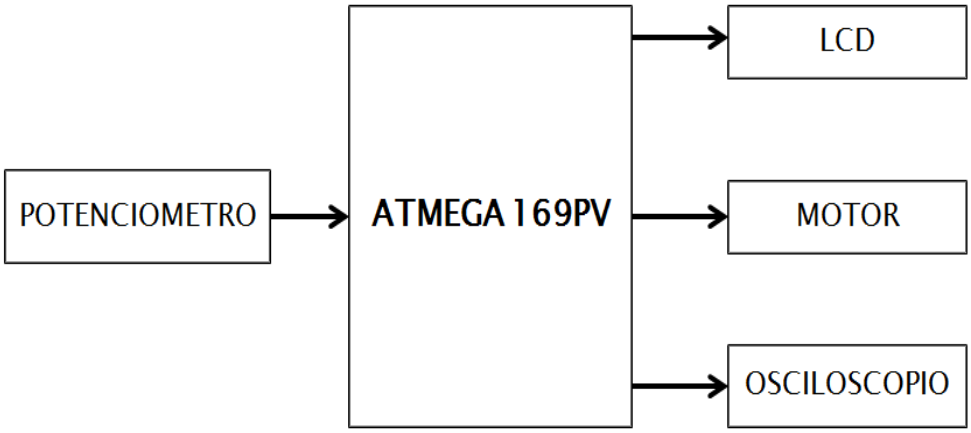
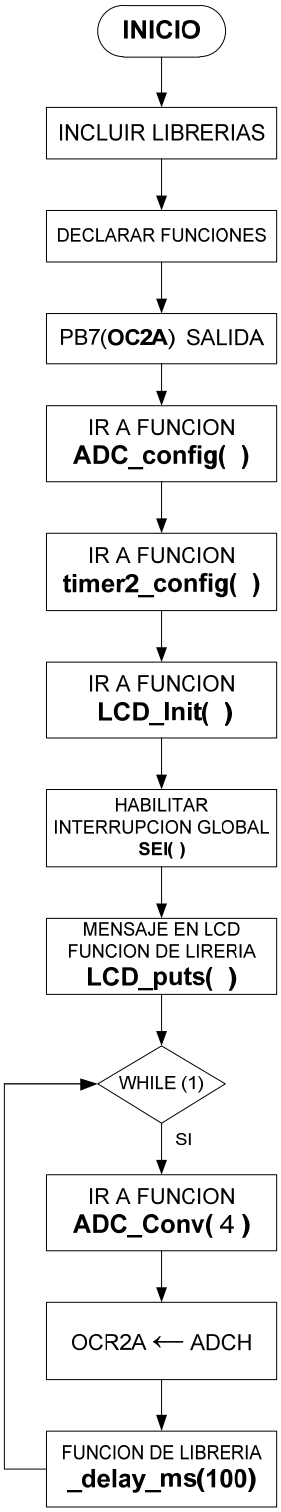
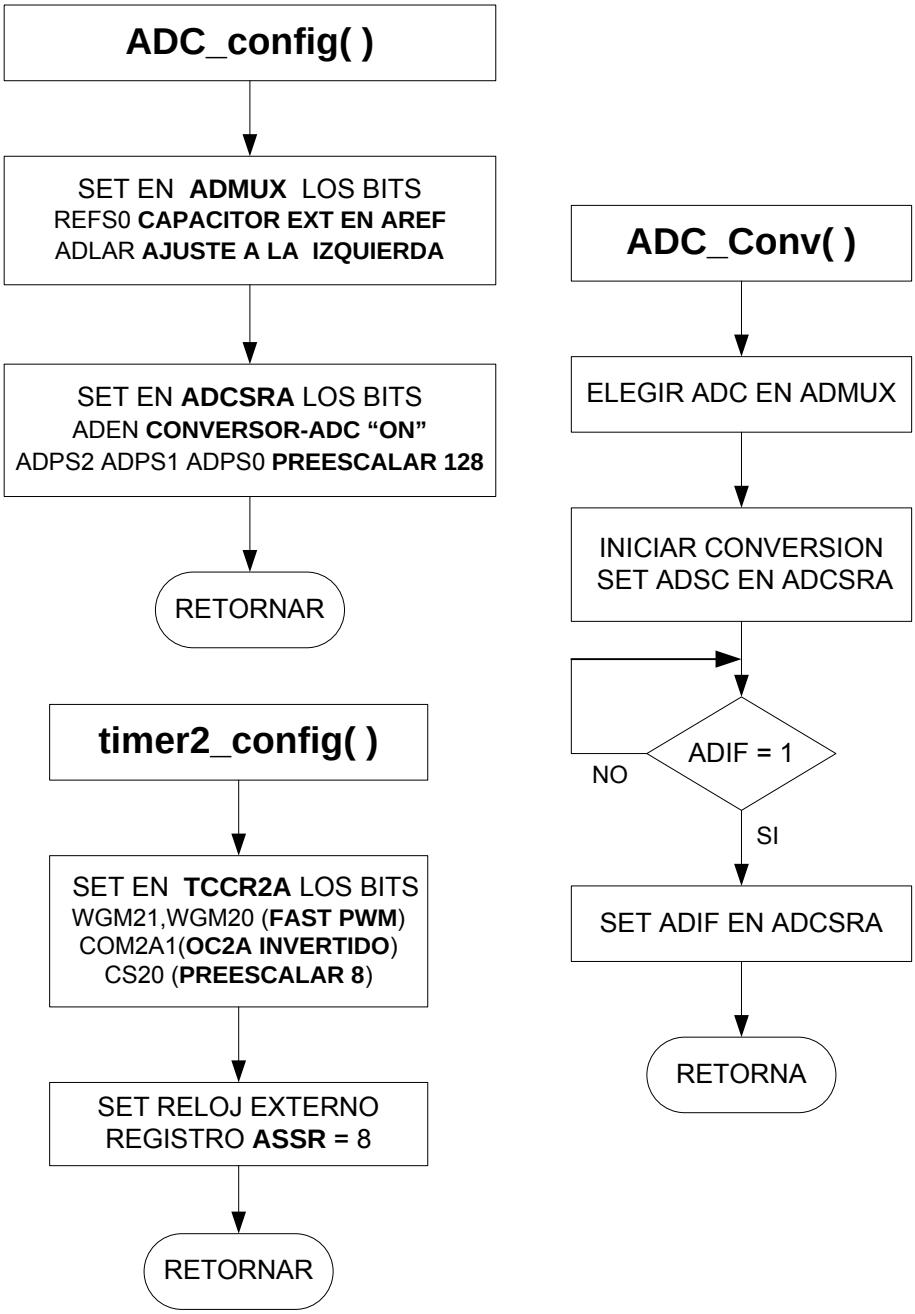


Figura 4.11 Diagrama de Bloques: Control de velocidad de motor DC.

4.5.2 Diagrama de Flujo: Control de velocidad motor DC.



4.5.3 Diagrama de Flujo: Rutinas Control de velocidad motor DC.



4.5.4 Programa principal en lenguaje C: Control de velocidad motor DC.

```
/*****************************************************************
                    MICROCONTROLADORES AVANZADOS

 GRUPO # 4

 INTEGRANTES:      RAUL BEJARANO
                   CHRISTIAN BLANCO

PROGRAMA: CONTROL DE VELOCIDAD DE MOTOR DC MEDIANTE EL TIMER 2 Y EL ADC

DESCRIPCION: EL SIGUIENTE EJEMPLO SE LLEVA A CABO LA CONFIGURACION DEL TIMER 2 EN MODO FAST PWM Y LA CONVERSION ADC DE UN VOLTAJE DADO POR UN POTENCIOMETRO MEDIANTE EL CUAL SE VA REGULANDO LA VELOCIDAD DEL MOTOR DC

*****************************************************************/
 #include <avr/io.h>
 #include <avr/interrupt.h>
 #include <util/delay.h>
 #include "lcd_driver.h"

/*                          ---------------------------------------------
==================| DECLARACION DE FUNCIONES |==================
                          ---------------------------------------------
*/
 void timer2_config();                    //  Función para configurar el Timer2
 void ADC_config(void);                   //  Función para configurar el ADC
 void ADC_Conv(unsigned char ch);  //  Función para realizar conversión ADC

/*                                  ---------------------
=======================| RUTINA MAIN |=========================
                                  ---------------------
*/

 int main(void)
 {
   ADC_config();    //  Función Configurar el ADC
   timer2_config();         //  Función Configurar del Timer2
   LCD_Init();              //  Función Configurar LCD
   sei();                   //  Habilita interrupción Global
   LCD_puts("TIMER2 MODO FAST PWM");  // Mensaje en el LCD

   while(1)
```

```
  {
     ADC_Conv(4);        // Realizar conversión ADC
     OCR2A=ADCH;      // Carga en OCR2A el valor de ADCH
          //   Donde: OCR2A: registro de comparación del TIMER2
          //   ADCH: registro de valor de conversión en 8 bits
     _delay_ms(100);
   }
 }

/*                              ----------------------------------------------
==================| CONFIGURARCION DEL TIMER2 |=================
                                ----------------------------------------------
*/

 void timer2_config()
 {
  TCCR2A = (1<<WGM21|1<<WGM20 |1<<COM2A1 | 0<<COM2A0 |
2<<CS20);
  /*              Timer2 en modo FAST PWM
                 OC2A en modo invertido
          Pre-escalar a 8
  */
  ASSR = 8;             // Reloj externo XLAT 32.768 Hz
  DDRB   = 0X80;      // OC2A/PB7  como salida
 }

/*                                 ---------------------------------------
=====================| CONFIGURACION DEL ADC | ==================
                                ---------------------------------------
*/
 void ADC_config(void)
 {
  ADMUX|=(1<<REFS0);       //   AVcc con capacitor externo en AREF
  ADMUX|=(1<<ADLAR);       //   Ajuste a la Izquierda
  ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
   // Enciende el convertidor ADC
   // Configura el Pre-escalar a 128
 }
/*                                  ---------------------------
=======================| CONVERSION ADC |=======================
                                ---------------------------
   Función para realizar la conversión ADC cargando los respectivos valores a los
                                        registros.
         Adicionalmente recibe un parámetro que indica el pin ADC a usarse
*/

 void ADC_Conv(unsigned char ch)
 {
    ch= ch & 0b00000111;             //  Indica cual ADC elegimos
```

```
    ADMUX |= ch;                    //  Selección del ADC en este caso
ADC4
    ADCSRA|=(1<<ADSC);              //  Inicia la conversión
    while(!(ADCSRA & (1<<ADIF)));    //  Espera por ADIF, conversión
completa
    ADCSRA|=(1<<ADIF);              //  Limpia ADFI poniendo 1 en este bit
 }
/*============================================================*/
```

4.5.5 Simulación: Control de velocidad motor DC

El programa cargado en el AVR Butterfly configura el Timer 2 en modo PWM-rápido y la conversión ADC de un voltaje dado por un potenciómetro mediante el cual se va regulando la velocidad del motor DC.

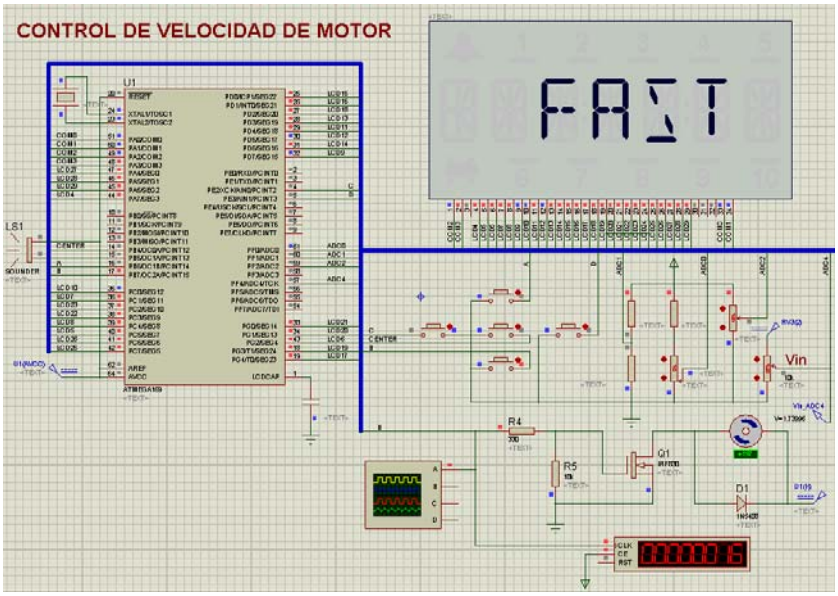


Figura 4.12 Esquemático: Control de velocidad de motor DC

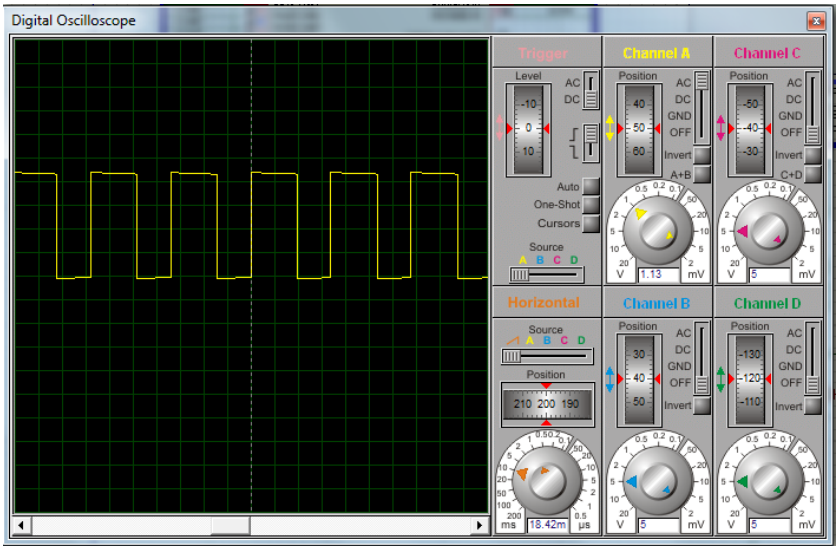


Figura 4.13 Esquemático: Señal cuadrada generada.

CONCLUSIONES

1. Al realizar el proyecto se logró implementar una plataforma entrenadora para el Kit AVR Butterfly de la familia ATMEL con lo cual los futuros estudiantes de la materia de microcontroladores contarán con esta versátil herramienta para el desarrollo de sus prácticas de laboratorio, además los ejercicios elaborados contribuirán a reforzar la comprensión de la configuración del Timer2.

2. El Kit AVR Butterfly es una poderosa herramienta de aprendizaje, es práctico, eficaz y muy amigable; que con el desarrollo del proyecto se va descubriendo progresivamente las características del microcontrolador ATmega169. La realización de los ejercicios de este proyecto nos ayudó a comprender la correcta manera de configurar el Timer2 en sus diferentes modos de operación y sus diferentes tipos de interrupciones y de igual manera la diferencia de programar usando lenguaje de ensamblador o lenguaje C.

3. El Kit AVR Butterfly con el controlador LCD y el microcontrolador ATmega169, permite abaratar costos en la implementación de aplicaciones en las cuales se necesite presentar información haciendo uso del LCD. Además como AVR Studio y WinAVR son gratuitos, se evita el uso ilegal de software con licencias adulteradas.

RECOMENDACIONES

1. Se recomienda no asentar el Kit AVR Butterfly en superficies conductivas, ya que se pueden producir cortocircuitos, los cuales provocan daños en el dispositivo.

2. Se puede utilizar una fuente de 3 voltios externa en caso de que la pila que incluye el fabricante en el Kit AVR Butterfly este descargado por el uso continuo.

3. Al momento de escribir un programa, es recomendable segmentarlo en funciones que luego serán utilizadas por el programa principal, con el fin de llevar un orden progresivo y así evitar confusiones al momento de revisar o corregir el código. Así también es necesario etiquetar cada función implementada con relación al proceso que se ejecuta con el fin de que sea de fácil entendimiento para las personas que necesiten utilizar el código en un nuevo programa.

ANEXOS

Section 1

Introduction

The AVR Butterfly evaluation kit is designed to demonstrate the benefits and key features of the AVR microcontrollers. It is a stand alone microprocessor module that can be used in numerous applications:

- The AVR architecture in general and the ATmega169 in particular
- Low power design
- The MLF package type
- Peripherals
  - LCD controller
  - Memories
    - Flash, EEPROM, SRAM, external DataFlash
  - Communication interfaces
    - UART, SPI, USI
  - Programming methods
    - Selfprogramming/ Bootloader, SPI, Parallel, JTAG
  - Analog to Digital Converter (ADC)
  - Timers/Counters
    - Real Time Clock (RTC)
    - Pulse Width Modulation (PWM)

It also serve as a development kit for the ATmega169, and can be used as a module in other products.

Figure 1-1. AVR Butterfly

1.1 Resources Available on the AVR Butterfly Kit

The following resources are available on the Butterfly kit.

- ATmega169 (MLF-package)
- LCD-on-glass display with 120 segments, for demonstrating the ATmega169 LCD controller.
- Joystick, 4-directions with centre push, as user input
- Piezo element, to play sounds
- 32kHz Xtal for the RTC
- 4 Mbit DataFlash, for data storage
- RS-232 level-converter, for communicating with off-board units
- Negative Temperature Coefficient (NTC) thermistor, to measure temperature
- Light Dependent Resistor (LDR), to measure light intenisty
- 3V button cell battery (600mAh) to provide operating power
- JTAG emulation, for debugging
- USI-interface, for additional communication interface
- Supported by AVR Studio 4.
- Pre-programmed with a demonstration application, including bootloader
- No external hardware is required to reprogram the AVR Butterfly

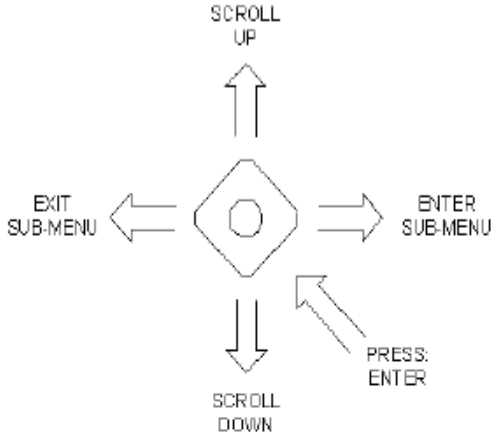
The ATmega169 in the kit controls the external peripherals, and can also be used to do voltage readings from 0 to 5 volts. The kit can be reprogrammed a number of different ways including serial programming through the JTAG port. Most users will prefer to use the preloaded bootloader with AVR Studio to download new code.

For more information about the ATmega169, see the datasheet at www.atmel.com.

2.2 Joystick Input

To operate the AVR Butterfly a joystick is used as user input. It operates in five directions, including center-push, see *Figure 2-1*.

Figure 2-1. Joystick Input



Using the joystick one can move around in the menu shown in *Figure 2-2*, and edit values, entering name, etc. Here are examples on how to enter your name.

2.2.1 Entering Your Name Using the Joystick:

1. Press the joystick up ("SCROLL UP") to wake the AVR Butterfly. If "AVR BUTTERFLY" is notscrolling over the display, press the joystick to the left ("EXIT SUB-MENU") until it does.
2. Press the joystick down ("SCROLL DOWN") three times, so the string "NAME" is displayed.
3. Press the joystick to the right ("ENTER SUB-MENU"). If this is the first time a name is entered, the string "ENTER NAME" will be displayed, otherwise the name already entered will be displayed and you have to press the joystick to the right ("ENTER SUB-MENU") once more.
4. When "ENTER NAME" is displayed press center push ("ENTER"). If this is the first time you enter a name, the character "A" should be blinking in the right side in the display, otherwise the last character of the already entered name will blink.
5. Press the joystick up ("SCROLL UP") or down ("SCROLL DOWN") to get to the wanted character. Press the joystick to the right ("ENTER SUB-MENU") to add a new character or press the joystick to the left ("EXIT SUB-MENU") to remove a character.
6. When you have got all the characters, up to maximum 25, press center push ("ENTER") to save this name. The name will now be displayed in the display. If the name is more than 6 characters long it will scroll over the display, otherwise it will be displayed static.

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-Chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  - 16K bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - 4 x 25 Segment LCD Driver
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Universal Serial Interface with Start Condition Detector
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 53 Programmable I/O Lines
  - 64-lead TQFP and 64-pad QFN/MLF
- Speed Grade:
  - ATmega169V: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 8 MHz @ 2.7 - 5.5V
  - ATmega169: 0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V
- Temperature range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode:
    1 MHz, 1.8V: 350µA
    32 kHz, 1.8V: 20µA (including Oscillator)
    32 kHz, 1.8V: 40µA (including Oscillator and LCD)
  - Power-down Mode:
    0.1µA at 1.8V

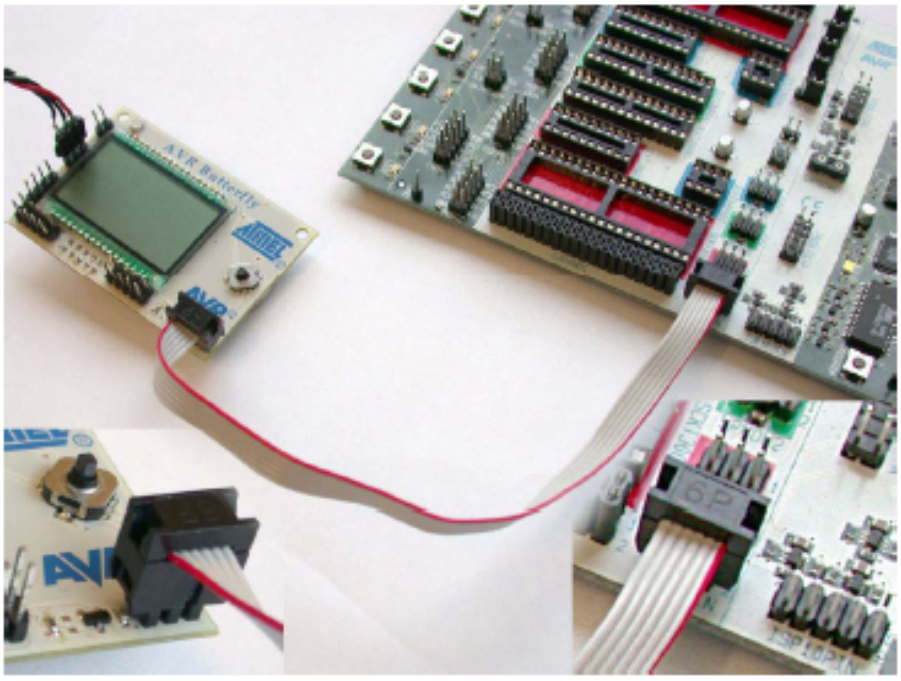
8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

ATmega169V
ATmega169

Notice:
Not recommended in new designs.

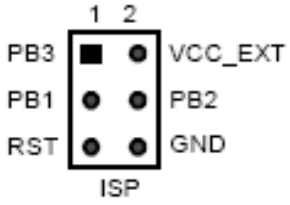
2514P–AVR–07/06

Figure 3-2. In-System Programming



To program the ATmega169 using ISP Programming mode, connect a 6-wire cable between the ISP6PIN connector on the STK500 board and J403 the ISP connector on the AVR Butterfly as shown in *Figure 3-2*. This device can be programmed using the Serial Programming mode in the AVR Studio4 STK500 software. Instead of soldering in a ISP-header, one can make contact just by pressing the header to the footprint. Make sure that pin 1 on the STK500 match with pin 1 on the AVR Butterfly. See *Figure 3-3* for the pinout of the ISP Connector.

Figure 3-3. ISP Connector, J403



BIBLIOGRAFÍA

[1] ATMEL,"8-BIT MICROCONTROLLER WITH 16K BYTES IN-SYSTEM PROGRAMMABLE FLASH", http://www.atmel.com/dyn/resources/prod_documents/doc2514.pdf
Fecha de consulta: 22 de agosto del 2011.

[2] BUTTERFLY LOGGER, "ESQUEMÁTICO DEL AVR BUTTERFLY", http://www.brokentoaster.com/butterflylogger/hw.html
Fecha de consulta: 22 de agosto del 2011.

[3] VICO DANIEL, "TIMER 2 DEL AVR ATMEL Y SUS REGISTROS", http://vicodg.blogspot.com/2011/03/timer-0.html#!/2011/03/timer-2.html
Fecha de consulta: 4 de septiembre del 2011.

[4] CALDERON JOHAN, "LENGUAJE C SOBRE COMO CONFIGURAR EL TIMER DEL AVR ATMEL", http://www.micro2c.com/categoria-Atmel-AVR-c2p0.html
Fecha de consulta: 4 de septiembre del 2011.

[5] GONZALES DANIEL, "CÓDIGOS EN ENSAMBLADOR SOBRE EL DISPLAY Y JOYSTICK", http://sites.google.com/site/avrasmintro/home/5-butterfly-lcd-joystick,
Fecha de consulta: 13 septiembre del 2011.

[6] ENGINEERSGARAGE, “CÓDIGO DE ADC”, http://www.engineersgarage.com/embedded/avr-microcontroller-projects/adc-circuit
Fecha de consulta: 13 septiembre del 2011.

[7] ATMEL NORWAY , “CODIGOS EN ENSAMBLADOR”, http://avr.15.forumer.com/a/beginners-amp-butterflies-5/
Fecha de consulta: 13 septiembre del 2011.

[8] TODO TOPIC, “CÓDIGO DE TIMER EN ASSAMBLER Y TIEMPO DE INSTRUCCIÓN”, http://www.todopic.com.ar/foros/index.php?topic=27420.100
Fecha de consulta: 13 septiembre del 2011.

[9] BOELLMANN WERNER, “AVR LIBC”, http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html
Fecha de consulta: 13 septiembre del 2011.

[10] GUERRERO RICHARD, “KIT DE DESARROLLO AVR BUTTERFLY, DESARROLLO DE GUÍA DE PRÁCTICAS DE LABORATORIO Y TUTORIALES”, www3.espe.edu.ec:8700/bitstream/21000/424/.../T-ESPE-014271.pdf
Fecha de consulta: 27 de octubre del 2011.

[11] UNIVERSIDAD DE OVIEDO,"MODULOS DE TEMPORIZACION DEL PIC", http://www.ate.uniovi.es/fernando/Doc2006/Sed_06 /Presentaciones/Timers_v2.pdf
Fecha de consulta: 27 de octubre del 2011.

[12] KUI ATTIE, "TIEMPO DE TEMPORIZACIÓN POR SOBREFLUJO IGUAL A 1HZ", http://www.sparkfun.com/products/540
Fecha de consulta: 27 de octubre del 2011.

[13] CACERES MARTIN, "ESQUEMA DE REFERENCIA PARA LA PARTE DE POTENCIA DEL MOTOR DC 12V", http://www.moddear.com.ar/showthread.php/172-Controlador-y-Medidor-de-rpm-s-MODDEAR/page3
Fecha de consulta: 8 de noviembre del 2011