



**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 TIMER1.”**

**TESINA DE SEMINARIO**

**Previa a la obtención del Título de:**

**INGENIERO EN ELECTRICIDAD**

**ESPECIALIZACIÓN**

**ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL.**

**Presentado por**

**ROMMEL FELIPE CHANG SUÁREZ**

**JEFFERSON EFRÉN MORENO BRIONES**

**Guayaquil – Ecuador**

**2012**

# **AGRADECIMIENTO**

A nuestros padres por el apoyo incondicional que nos han brindado a lo largo de nuestros estudios.

A nuestros profesores por sus conocimientos impartidos a lo largo de nuestra formación.

# AGRADECIMIENTO

A mis Padres sin duda, y en especial a mi Abuelita Electra ya que sin sus sabios consejos basados en sus experiencias de vida, que me los ha transmitido han sido de gran valor para estar en donde estoy ahora, cerca de culminar una gran etapa en mi vida.

*Jefferson Efrén Moreno Briones.*

# DEDICATORIA

A Dios que sin su bendición nada se hubiese podido realizar.

A mis padres, que con su apoyo incondicional y la muestra del gran amor que tengo hacia ellos se pudo realizar este importante paso en mi vida.

*Rommel Felipe Chang Suárez.*

# DEDICATORIA

A Dios por permitirme alcanzar este logro, porque sin duda, todo lo que logramos es gracias a su bendición.

A mis padres que me han brindado su incondicional amor y apoyo a lo largo de mi vida, a mis hermanos por su cariño y apoyo emocional que jamás me hizo falta, quiero retribuirles con éxitos su inmenso esfuerzo por mi formación personal y profesional.

*Jefferson Efrén Moreno Briones*

# TRIBUNAL DE SUSTENTACIÓN

---

Ing. Carlos Valdivieso A.

Profesor del Seminario de Graduación

---

Ing. Hugo Villavicencio V.

Profesor Delegado del Decano

# DECLARACIÓN EXPRESA

“La responsabilidad del contenido de esta Tesis de Grado, nos corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”

(Reglamento de Graduación de la ESPOL).

---

Rommel Felipe Chang Suárez.

---

Jefferson Efrén Moreno Briones.

# RESUMEN

La finalidad del presente proyecto es el desarrollo de una plataforma interactiva para la demostración del TIMER/COUNTER1 utilizado con los microcontroladores de Atmel para ello se utilizarán varias herramientas de Software tales como un programador para microcontroladores Atmel como lo es el AVR Studio 4 y un simulador de circuitos electrónicos como lo es el Proteus y para la implementación física se utilizará el **Kit de desarrollo AVR butterfly**, 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.

En el capítulo uno se describe una breve comparación de los microcontroladores de atmel con los microcontroladores de microchip e Intel basado en el TIMER/COUNTER1 de cada microcontrolador.

El capítulo dos trata sobre las Herramientas software y hardware que afianzan la implementación y optimización para desarrollo del Proyecto.

En el capítulo tres se describe cada aplicación desarrollada en lenguaje assembler o en C donde se bosqueja cada aplicación a nuestra plataforma interactiva en el cual demostraremos las características del TIMER/COUNTER1 de los microcontroladores de la familia ATMEL.

En el capítulo 4 se mostrará los respectivos diagramas de bloques, diagramas de flujo código de programación junto con las simulaciones y pruebas del proyecto.



## INDICE GENERAL

### Contenido

AGRADECIMIENTO.....	I
DEDICATORIA.....	III
DECLARACIÓN EXPRESA.....	VI
RESUMEN.....	VII
INDICE GENERAL.....	VIII
INDICE DE FIGURAS.....	XI
INDICE DE TABLAS.....	XII

### CAPITULO 1

#### 1.0 Generalidades

1.1.0 Descripción General.....	1
1.2.0 Los Registros.....	2
1.3.0 Fuentes de Reloj para el TIMER/COUNTER.....	3
1.4.0 La Unidad de Conteo.....	3
1.5.0 Descripción de los registros del TIMER/COUNTER1.....	5
1.5.1 Registro A para el control del TIMER/COUNTER1.....	5
1.5.2 Registro B para el control del TIMER/COUNTER1.....	7
1.5.3 Registro C para el control del TIMER/COUNTER1.....	8
1.5.4 Registro para capturar datos del timer1 – ICR1H e ICR1L.....	9
1.5.5 Registro máscara de Interrupción del Timer/Counter.....	9
1.6.0 Comparación.....	11

### CAPITULO 2

#### 2.0 Generalidades

2.1.0 Fundamento Teórico	
2.2.0 Kit de Desarrollo AVR Butterfly.....	13
2.3.0 Hardware Disponible.....	14
2.4.0 Firmware Incluido.....	16
2.4.1 Joystick.....	16
2.4.2 Menú.....	16

2.5.0	Conectores.....	17
2.6.0	Programación mediante comunicación serial con la PC.....	17
2.6.1	Distribución de pines para la comunicación serial.....	17
2.6.2	Programación del AVR.....	18
2.7.0	El LCD.....	20
2.8.0	El Joystick.....	21
2.9.0	AVR Studio Software.....	22
2.9.1	Introducción.....	22
2.10.0	Descripción General del IDE.....	23
2.11.0	Las ventanas principales del IDE AVR Studio.....	25
2.11.1	La ventana del Editor.....	25
2.11.2	La ventana Output.....	25
2.11.3	La ventana Workspace.....	26
2.11.4	La ventana Watch.....	38
2.11.5	La ventana Memory.....	29
2.11.6	La barra de estado.....	29

## CAPITULO 3

### 3.0 Generalidades

3.1.0	Componentes	
3.2.0	Implementación.....	30
3.3.0	Implementación de Proyectos.....	31
3.3.1	Contador de segundos Up/Down usando el timer1.....	31
3.3.2	Uso del timer1 CTC doble comparación.....	32
3.3.3	Reloj a diferentes frecuencias.....	33
3.3.4	Control PWM.....	34
3.3.5	Control variable (PWM y CTC).....	35

## CAPITULO 4

### 4.0 Generalidades

4.1.0	Contador de segundos Up/Down usando el timer1	
4.1.1	Fundamento teórico.....	36

4.1.2	Diagrama de flujo y Algoritmo.....	37
4.1.3	Código de Programación.....	39
4.2.0	Uso del timer1 CTC doble comparación.....	43
4.2.1	Fundamento teórico.....	43
4.2.2	Diagrama de flujo y Algoritmo .....	45
4.2.3	Código de Programación.....	47
4.3.0	Reloj a diferentes frecuencias.....	49
4.3.1	Fundamento teórico.....	49
4.3.2	Diagrama de flujo y Algoritmo .....	50
4.3.3	Código de Programación.....	52
4.4.0	Control PWM.....	54
4.4.1	Fundamento teórico.....	54
4.4.2	Diagrama de flujo y Algoritmo .....	55
4.4.3	Código de Programación.....	57
4.5.0	Control variable (PWM y CTC ).....	59
4.5.1	Fundamento teórico.....	59
4.5.2	Diagrama de flujo y Algoritmo .....	60
4.5.3	Código de Programación.....	62

Conclusiones

Recomendaciones

Bibliografía

## INDICE DE FIGURAS.

Figura. 1.1 Diagrama de Bloques del Timer/Counter1 de 16 bits.....	1
Figura. 1.2. Diagrama de Bloques de la Unidad Contadora.....	3
Figura. 2.2.0. AVR Butterfly.....	13
Figura. 2.3.0 Hardware Disponible (Parte Posterior).....	16
Figura. 2.5.0. Conectores del AVR Butterfly para acceso a periféricos.....	17
Figura. 2.6.1 Conexiones para interfaz USART del AVR Butterfly.....	18
Figura. 2.6.2.0. AVR Studio, selección del archivo COF para depuración.....	19
Figura. 2.6.2.1. Selección del AVR Simulator y Dispositivo ATmega169.....	19
Figura. 2.7.0 Vidrio LCD.....	21
Figura. 2.8.0. Joystick.....	22
Figura 2.10 Entorno de Desarrollo Integrado AVR Studio4.....	24
Figura. 2.11.1 Ventana del Editor.....	25
Figura. 2.11.2. Ventana Output.....	26
Figura. 2.11.3.0 Ventana Project.....	27
Figura. 2.11.3.1. Ventana I/O.....	27
Figura. 2.11.3.2. Ventana Info.....	28
Figura. 2.11.4. Vista Watch.....	28
Figura. 2.11.5. Vista Memory.....	29
Figura. 2.11.6. Barra de Estado.....	29
Figura. 3.2.0 Maqueta.....	30
Figura. 3.3.0. Esquema de conexiones.....	31
Figura. 3.3.1. Esquema de conexiones.....	32

Figura. 3.3.2. Esquema de conexiones.....	33
Figura. 3.3.3. Esquema de conexiones.....	34
Figura. 3.3.4. Esquema de conexiones.....	35
Figura. 4.1.0. Contador de segundos Up/Down usando el timer1.....	
Figura. 4.1.1. Uso del timer1 modo CTC doble comparación 1A y 1B.....	43
Figura. 4.1.2. Ciclo de trabajo .....	44
Figura. 4.1.3. Reloj a diferentes frecuencias.....	49
Figura. 4.1.4. Control PWM e intensidad de luz.....	54
Figura. 4.1.5. Control Variable (CTC y PWM).....	59

## INDICE DE TABLAS.

Tabla 1.1. Modo de Comparación CTC.....	5
Tabla 1.2. Modo de Comparación PWM rápido.....	6
Tabla 1.3. Modo de Comparación PWM y Fase Correcta.....	6
Tabla 1.4. Descripción de los Bits para Modo Generador de Onda.....	7
Tabla 1.5. Descripción del Selector de Reloj.....	8
Tabla 2.1. Distribución de los pines. ACR Butterfly vs. PC.....	18

# CAPITULO 1

## RELEVANCIA DEL TIMER/COUNTER1 PARA EL DESARROLL DE ESTE PROYECTO

### 1.0. GENERALIDADES

El TIMER/COUNTER1 en este proyecto se lo usa en combinación con una plataforma desarrollada para hacer énfasis en el estudio de sus características.

En este capítulo se describe las características más relevantes y funcionamiento del timer/counter1. Los registros que actúan sobre el para su correcta calibración y las comparaciones más destacadas sobre el timer1 entre la familias Intel y Microchip.

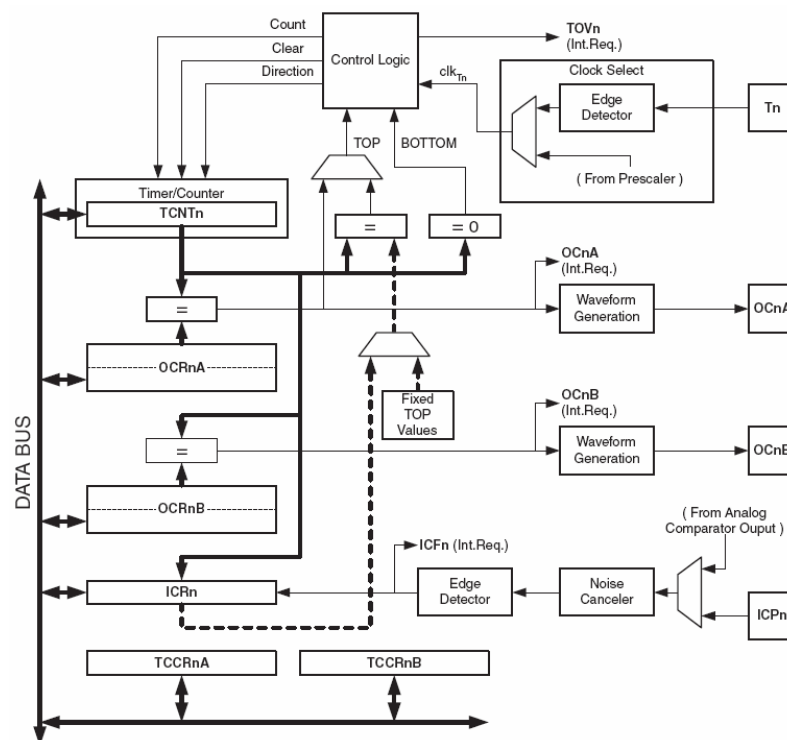
#### 1.1.0. DESCRIPCION GENERAL

La unidad del TIMER/COUNTER1 de 16 bits permite la correcta temporización (sincronización) para la ejecución del programa (administración de evento), generación de onda y medición temporizada de la señal. Las características principales del TIMER/COUNTER1 se listan a continuación. [1]

- Diseño verdadero de 16 bit (p.ej. permite PWM de 16 bit).
- Dos Unidades Comparadoras independientes.
- Registros Comparadores de doble buffer.
- Una Unidad para Captura de Datos (Input Capture).
- Anulador de Ruido para la Captura de Datos.
- Timer Encerado por Coincidencia en Comparación (Auto Recarga).

- Fase Correcta del Modulador de Ancho de Pulso (PWM), libre de fallos.
- PWM de Período Variable.
- Generador de Frecuencia.
- Contador de Evento Externo.
- Cuatro Fuentes de Interrupción Independientes (TOV1, OCF1A, OCF1B e ICF1).

En la Figura 1.1 se muestra un diagrama de bloque simplificado del Timer/Counter1



**Figura. 1.1 Diagrama de Bloques del Timer/Counter1 de 16 bits**



### 1.2.0. Los Registros.

El Timer/Counter (TCNT1), los Registros para Comparación (Output Compare Registers OCR1A/B) y el Registro de Captura de Datos (Input Capture Register ICR1) son todos los registros de 16 bits. Los Registros para el Control del Timer/Counter (Timer/Counter Control Registers TCCR1A/B) son registros de 8 bits y no tienen restricciones de acceso para la CPU.

Las señales de petición de Interrupción son todas visibles en el Registro para Indicadores de Interrupción del Timer (Timer Interrupt Flag Register TIFR1). Todas las interrupciones son enmascaradas individualmente con el Registro Enmascarador de Interrupciones del Timer (Timer Interrupt Mask Register TIMSK1.) [1]

El Timer/Counter puede ser provisto de señal de reloj internamente, vía preajustador (prescaler) o por una fuente externa de reloj en el pin T1. El bloque lógico Clock Select controla cual fuente de reloj y flanco usará el Timer/Counter para incrementar (o decrementar) su valor. El Timer/Counter permanece inactivo cuando ninguna fuente de reloj está seleccionada. La salida de la lógica del Clock Select es llamada reloj del timer ( $clk_{T1}$ ).

Los Registros de Comparación con doble buffer (OCR1A/B) son comparados en todo momento con el valor del Timer/Counter (TCNT1). El resultado de la comparación puede ser usado por el Generador de Onda para generar un PWM o una salida de frecuencia variable en el pin Output Compare (OC1A/B). El evento de Coincidencia en la Comparación también seteará al Indicador de Coincidencia en la Comparación (Compare Match Flag OCF1A/B), el cual podrá ser usado para generar una petición de interrupción por la Comparación.

La unidad para Captura de Datos (Input Capture Unit) incluye una unidad de filtraje digital (Anulador de Ruido) para reducir la probabilidad de capturar picos de ruido.

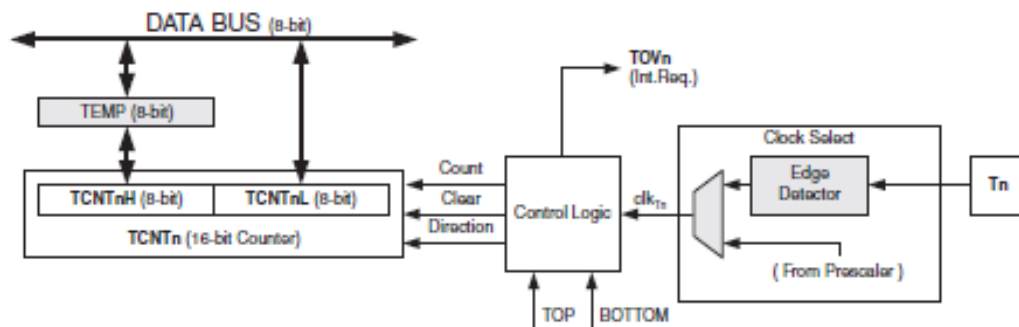
El bit PRTIM1 de “Power Reduction Register - PRR” debe ser escrito con cero para habilitar el módulo Timer/Counter1. [1]

### 1.3.0 Fuentes de Reloj para el Timer/Counter

El Timer/Counter puede ser provisto de señal de reloj por una fuente de reloj interna o externa. La fuente de reloj se selecciona por la lógica del Clock Select, la cual es controlada por los bits Clock Select (CS12:0) localizados en el Registro B para Control del Timer/Counter0 (Timer/Counter Control Register B – TCCR1B).

### 1.4.0 La Unidad de Conteo (Counter Unit)

La parte principal del Timer/Counter de 16 bit es la unidad Contadora bi-direccional programable de 16 bits. La Figura 1.2 muestra un diagrama de bloques del Contador y su entorno.



**Figura. 1.2. Diagrama de Bloques de la Unidad Contadora.**

Descripción de las señales (señales internas):

**Count** Incrementar o decrementar por uno al TCNT1.

**Direction** Seleccionar entre incremento y decremento.

**Clear** Borrar contenido del TCNT1 (poner todos los bits en cero).

**clk<sub>T1</sub>** Reloj del Timer/Counter

**TOP** Significa que TCNT1 ha alcanzado el valor máximo.

**BOTTOM** Significa que TCNT1 ha alcanzado el valor mínimo (cero).

Dependiendo del modo de operación usado, el Contador es borrado, incrementado o decrementado por el reloj del timer (clk<sub>T1</sub>). clk<sub>T1</sub> puede ser generado desde una fuente reloj externa o interna, seleccionado por los bits Clock Select (CS12:0). Cuando no se haya seleccionado ninguna fuente de reloj (CS12:0=0) el timer permanecerá detenido. Sin embargo, el valor del TCNT1 puede ser accedido por la CPU, indiferentemente de si está o no presente clk<sub>T1</sub>. [2]

La secuencia de conteo está determinada por la configuración de los bits (WGM13:0) localizados en los Timer/Counter Control Registers A y B (TCCR1A y TCCR1B).

El indicador de Desbordamiento del Timer/Counter (TOV1) es seteado de acuerdo al modo de operación seleccionado por los bits WGM13:0. TOV1 puede usarse para generar una interrupción.

## 1.5.0 Descripción de los Registros del Timer/Counter1 de 16 bits

### 1.5.1 Registro A para Control del Timer/Counter1 – TCCR1A

**Bit 7:6 – COM1A1:0. Modo de Comparación para la Unidad A (Output Compara for Unit A).**

Bit	7	6	5	4	3	2	1	0	TCCR1A
	COM1A1	COM1A0	COM1B1	COM1B0	—	—	WGM11	WGM10	
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

**Bit 5:4 – COM1B1:0. Modo de Comparación para la Unidad B (Output Compara for Unit B).**

Los bits COM1A1:0 y COM1B1:0 controlan el comportamiento de los pines Output Compare (OC1A y OC1B respectivamente). Los bits de los Registros de Dirección de Datos correspondientes a los pines OC1A y OC1B deben ser seteados para habilitarlos como salidas. [2]

La Tabla 1.1 muestra la funcionalidad de los pines COM1x1:0 cuando los bits WGM13:0 están seteados para un modo Normal o un modo CTC (sin PWM).

COM1A1/COM1B1	COM1A0/COM1B0	DESCRIPTION
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/oc1B on Compare Match
1	0	Clear OC1A/OC1B on Compare Match (Set output to low level)
1	1	Set OC1A/OC1B on Compare Match (Set output to high level)

La tabla 1.2 muestra la funcionalidad de los pines COM1x1:0 cuando los bits WGM13:0 están seteados para el modo PWM rápido.

COM1A1/COM1B1	COM1A0/COM1B0	DESCRIPTION
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 14 Or 15. Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at BOTTOM (inverting mode)

**Tabla 1.2. Modo de Comparación, PWM Rápido**

La Tabla 1.3 muestra la funcionalidad de los pines COM1x1:0 cuando los bits WGM13:0 están seteados para el modo PWM de fase correcta o de Fase y Frecuencia Correcta.

COM1A1/COM1B1	COM1A0/COM1B0	DESCRIPTION
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 11. Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match when up-counting. Set OC1A/OC1B on Compare Match when downcounting.
1	1	Set OC1A/OC1B on Compare Match when up-counting. Clear OC1A/OC1B on Compare Match when downcounting.

**Tabla. 1.3. Modo de Comparación, PWM de Fase Correcta y de Fase y Frecuencia Correcta**



**Bit 7 – ICNC1.** Anulador de Ruido para la Captura de Datos (Input Capture).

**Bit 6 – ICES1.** Selector de Flanco para la Captura de Datos.

**Bit 5 – Bit.** Reservado.

**Bit 4:3 – WGM13:2.** Modo de Generación de Onda.

**Bit 2:0 – CS12:0. Selectores de Reloj.** Los tres bits Selectores de Reloj seleccionan la fuente de reloj que será usada por el Timer/Counter1. Ver Tabla 1.5.

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk / 1 (No prescaling)
0	1	0	clk / 8 (From prescaler)
0	1	1	clk / 64 (From prescaler)
1	0	0	clk / 256 (From prescaler)
1	0	1	clk / 1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge
1	1	1	External clock source on T1 pin. Clock on rising edge

**Tabla. 1.5. Descripción del Selector de Reloj**

### 1.5.3 Registro C para Control del Timer/Counter1 – TCCR1C

Bit	7	6	5	4	3	2	1	0	TCCR1C
	ICNC1	ICES1	—	—	—	—	—	—	
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Los bits FOC1A/FOC1B están activos únicamente cuando los bits GM13:0 especifican un modo sin – PWM. Sin embargo, para asegurar compatibilidad con futuros dispositivos, estos bits deben ser puestos a cero cuando TCCR1A se escriba al operar en un modo de PWM. Al escribir un uno en los bits FOC1A/FOC1B, se fuerza a una coincidencia de comparación inmediata en la Unidad Generadora de Onda, las salidas OC1A/OC1B cambian según la configuración de sus bits COM1x1:0.

### Timer/Counter1 – TCNT1H y TCNT1L

Las dos localidades de I/O del Timer/Counter (TCNT1H y TCNT1L, combinadas TCNT1) dan acceso directo, para las operaciones de lectura y escritura, al contador de

16 bits de la Unidad Timer/Counter. Para asegurarse que ambos bytes, alto y bajo, sean leídos y escritos simultáneamente cuando la CPU accede a estos registros, el acceso es realizado usando un Registro temporal de 8 bit para Byte Alto (TEMP). Este registro temporal es compartido por todos los otros registros de 16 bit.

Registro A para Comparación con TCNT1 – OCR1AH y OCR1AL

Registro B para Comparación con TCNT1– OCR1BH y OCR1BL

Los Registros para comparación contienen un valor de 16 bits que es comparado continuamente con el valor del contador (TCNT1). Una coincidencia puede usarse para generar una interrupción por comparación o para generar una onda en el pin OC1x.

Los registros para comparación tienen un tamaño de 16 bits. Para asegurarse que ambos bytes, el alto y el bajo, sean escritos simultáneamente cuando la CPU escribe en estos registros, el acceso se lo realiza usando un registro temporal (TEMP) para el byte alto de 8 bit. Este registro temporal es compartido por todos los otros registros de 16 bits.

#### **1.5.4 Registro para capturar de datos del timer 1 – ICR1H e ICR1L**

Este registro se actualiza con el valor de contador (TCNT1) cada vez que ocurre un evento en el pin ICP. El registro ICR1 (ICR1H e ICR1L) puede ser usado para definir el valor TOP del contador.

#### **1.5.5 Registro Máscara de Interrupción del Timer/Counter1**

**Bit 5 – ICIE1.** Habilitador de la Interrupción por Captura de Datos del Timer/Counter1.

**Bit 2 – OCIE1B.** Habilitador de la Interrupción por Coincidencia en la comparación entre OCR1B y TCNT1.



**Bit 1 \_ OCIE1A.** Habilitador de la Interrupción por coincidencia de la comparación entre OCR1A y TCNT1.

**Bit 0 – TOIE1.** Habilitador de la Interrupción por Desbordamiento del Timer/Counter1-

### Registro Indicador de Interrupción del Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
	—	—	ICIE	—	—	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

**Bit 5 – ICF1: Indicador de Interrupción por Captura de Datos del Timer/Counter1.**

Este indicador se setea cuando ocurre un evento en el pin ICP1. Cuando los bits WGM13:0 están configurados para utilizar el registro ICR1 como valor TOP, el indicador ICF1 se setea cuando el contador alcanza el valor TOP.

Este indicador es puesto a cero por hardware luego de que se ejecuta el vector de interrupción correspondiente. Alternativamente, puede ser borrado escribiendo un uno lógico en su posición de bit.

**Bit 2 – OCF1B: Indicador de Interrupción por Coincidencia en la Comparación entre el OCR1B y el TCNT1 del Timer/Counter1.**

Este indicador se setea en el ciclo de reloj siguiente al ciclo en el que el valor del contador (TCNT1) coincide con el registro OCR1B (TCNT1 es igual que OCR1B).

Este indicador es puesto a cero por hardware luego de que se ejecuta el vector de interrupción correspondiente. Alternativamente, puede ser borrado escribiendo un uno lógico en su posición de bit.

**Bit 1 – OCF1A: Indicador de Interrupción por Coincidencia en la Comparación entre el OCR1A y el TCNT1 del Timer/Counter1.**

Flanco de reloj siguiente al ciclo en el que el valor del contador (TCNT1) coincide con el registro OCR1A en la comparación.

Este indicador es puesto a cero por hardware luego de que se ejecuta el vector de interrupción correspondiente. Alternativamente, puede ser borrado escribiendo un uno lógico en su posición de bit.

**Bit 0 – TOV1: Indicador de Desborde del Timer/Counter1.**

Este indicador se setea dependiendo de la configuración de los bits WGM13:0.

Este indicador es puesto a cero por hardware luego de que se ejecuta el vector de interrupción correspondiente. Alternativamente, puede ser borrado escribiendo un uno lógico en su posición de bit.

### **1.6.0 Comparaciones**

Vamos a comparar el módulo TIMER/COUNTER1 del atmega189 de atmel y el PIC 16f877 que son de similares características y luego haremos una comparación con el microcontrolador 8051 de la familia INTEL.

**Atmel vs Microchip**

Modo comparador/temporizador	SI	SI
Preescalador	clk/1 to clk/1024	1:1 to 1:8
Unidades de comparación	2	2
Cancelador de ruido en la entrada de captura	SI	NO
Entrada en la unidad de captura	SI	NO
Periodo PWM variable	SI	SI
Modulador de ancho de pulso fase corregida	SI	NO
Interrupciones individuales por timer-1	4 (OVF,OCF1A,OCF1B,OCF1C)	3 (capture,compare,overflow)
Entrada de Reset interna	NO	SI
Generador de frecuencia	SI	NO
Contador de eventos externos	SI	SI

**Atmel vs Intel**

Preescalador	clk/1 to clk/1024	1:1 to 1:64
Periodo PWM variable	SI	SI
Módulos de captura	2	1
Generador de baudios	NO	SI
Modulador de ancho de pulso fase corregida	SI	NO
Generador de frecuencia	SI	NO
Contador de eventos externos	SI	SI

# CAPITULO 2

## HARDWARE Y SOFTWARE

### 2.0. GENERALIDADES

En este capítulo vamos a describir el software y hardware utilizados para la programación

y ensamble de los proyectos que se describen en los capítulos posteriores. El uso del software de programación AVR studio 4.0 y una descripción detallada del Kit Butterfly del AVR.

#### 2.1.0 FUNDAMENTO TEORICO

La mariposa es un regulador autónomo que funciona con pilas usando placa de demostración que ejecuta el microcontrolador Atmel AVR ATMEGA169PV. El tablero incluye una pantalla de cristal líquido (LCD), joystick, altavoz, puerto serie, reloj en tiempo real (RTC), la memoria flash interna, y los sensores de temperatura y voltaje. La junta es del tamaño de una etiqueta con su nombre y con el pasador en la parte posterior puede ser usado como tal después de que el usuario introduce su nombre en la pantalla LCD.

La mariposa AVR viene precargado con el software que muestra muchas de las capacidades de los microcontroladores AVR. Firmware de fábrica puede desplazarse a su nombre, visualización de las lecturas del sensor, y mostrar la hora. La mariposa AVR también tiene un altavoz que puede reproducir todos los sonidos y la música.

### 2.2.0. KIT DE DESARROLLO AVR BUTTERFLY (HARDWARE)

El Kit AVR Butterfly se diseñó para demostrar los beneficios y las características más importantes de los microcontroladores ATMEL.

El AVR Butterfly utiliza el microcontrolador AVR ATmega169V, que combina la Tecnología Flash con el más avanzado y versátil microcontrolador de 8 bits disponible. En la Figura 2.2.0 se puede apreciar el Kit AVR Butterfly. [5]



**Figura. 2.2.0. AVR Butterfly**

El Kit AVR Butterfly expone las siguientes características principales:

- La arquitectura AVR en general y la ATmega169 en particular.
- Diseño de bajo consumo de energía.
- El encapsulado tipo MLF.
- Periféricos:
  - ✓ Controlador LCD.
  - ✓ Memorias:

- ✓ Flash, EEPROM, SRAM.
  - ✓ DataFlash externa.
- Interfaces de comunicación:
  - ✓ UART, SPI, USI.
- Métodos de programación
  - ✓ Self-Programming/Bootloader, SPI, Paralelo, JTAG.
- Convertidor Analógico Digital (ADC).
- Timers/Counters:
  - ✓ Contador de Tiempo Real (RTC).
  - ✓ Modulación de Ancho de Pulso (PWM).

El AVR Butterfly está proyectado para el desarrollo de aplicaciones con el ATmega169 y además puede usarse como un módulo dentro de otros productos.

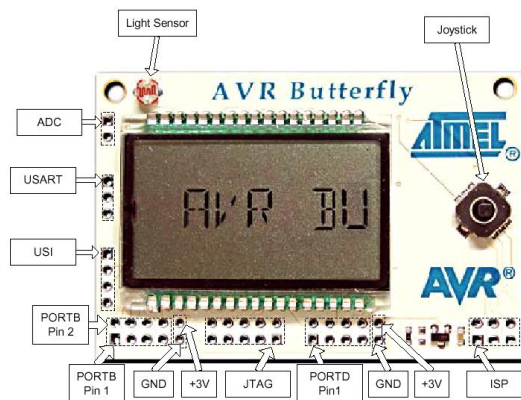
### **2.3.0. HARDWARE DISPONIBLE**

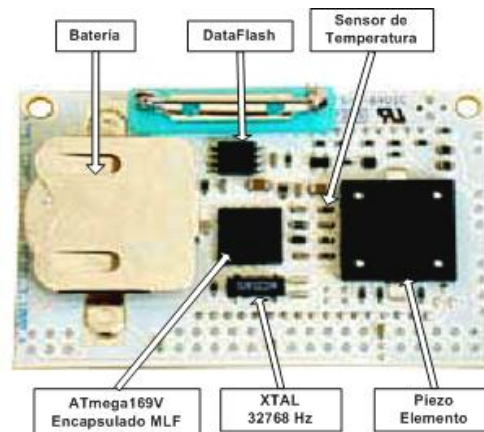
Los siguientes recursos están disponibles en el Kit AVR Butterfly: [5]

- Microcontrolador ATmega169V (en encapsulado tipo MLF).
- Pantalla tipo vidrio LCD de 120 segmentos, para demostrar las capacidades del controlador de LCD incluido dentro del ATmega169.
- Joystick de cinco direcciones, incluida la presión en el centro.
- Altavoz piezoeléctrico, para reproducir sonidos.
- Cristal de 32 KHz para el RTC.
- Memoria DataFlash de 4 Mbit, para el almacenar datos.
- Convertidor de nivel RS-232 e interfaz USART, para comunicarse con unidades fuera del Kit sin la necesidad de hardware adicional.
- Termistor de Coeficiente de Temperatura Negativo (NTC), para sensar y medir temperatura.

- Resistencia Dependiente de Luz (LDR), para sensar y medir intensidad luminosa.
- Acceso externo al canal 1 del ADC del ATmega169, para lectura de voltaje en el rango de 0 a 5 V.
- Emulación JTAG, para depuración.
- Interfaz USI, para una interfaz adicional de comunicación.
- Terminales externas para conectores tipo Header, para el acceso a periféricos.
- Batería de 3 V tipo botón (600mAh), para proveer de energía y permitir el funcionamiento del AVR Butterfly.
- Bootloader, para programación mediante la PC sin hardware especial.
- Aplicación demostrativa preprogramada.
- Copatibilidad con el Entorno de Desarrollo AVR Studio4

En las Figuras 2.3.0 y 2.3.1 se observa el Hardware disponible en el AVR Butterfly.





**Figura. 2.3.0 Hardware Disponible (Parte Posterior)**

Este Kit puede reprogramarse de varias formas diferentes, incluyendo programación serial a través del puerto JTAG; pero, se preferirá el uso del Bootloader precargado junto con el AVR Studio, para descargar nuevo código sin la necesidad de hardware especial.

#### **2.4.0. FIRMWARE INCLUIDO**

El AVR Butterfly viene con una aplicación preprogramada. Esta sección presentará una revisión de los elementos de esta aplicación. [6]

**2.4.1. Joystick** Para operar el AVR Butterfly se emplea el joystick como una entrada para él. Este opera en cinco direcciones, incluyendo presión en el centro.

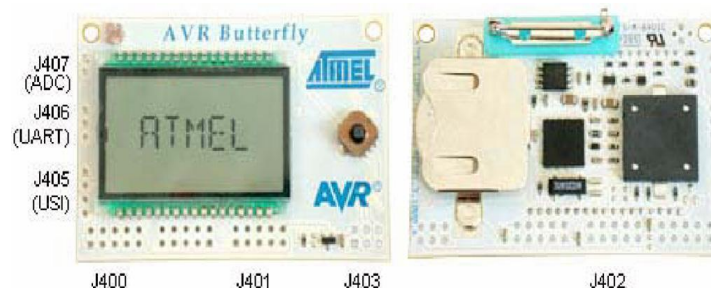
Utilizando el joystick el usuario puede moverse y editar valores, ingresar nombres, etc.

**2.4.2. Menú** El sistema de menú está creado para poder desplazarse, de manera eficiente, entre los diferentes módulos de la aplicación.



### 2.5.0. Conectores

Algunos de los pines de I/O del microcontrolador ATmega169 están disponibles en los conectores del AVR Butterfly. Estos conectores son para comunicación, programación y entrada al ADC del ATmega169. En la Figura 2.5.0 se puede apreciar los conectores del AVR Butterfly.



**Figura. 2.5.0. Conectores del AVR Butterfly para acceso a periféricos**

### 2.6.0 PROGRAMACIÓN MEDIANTE CONEXIÓN SERIAL (UART) CON LA PC

El AVR Butterfly tiene incluido un convertidor de nivel para la interfaz RS-232. Esto significa que no se necesita de hardware especial para reprogramar al AVR Butterfly utilizando la característica self-programming del ATmega169. A continuación se explica brevemente la distribución de los pines y como se debe realizar el cableado para la comunicación serial entre el AVR Butterfly y la PC. [7]

#### 2.6.1. Distribución de pines para la comunicación serial entre el AVR y la PC

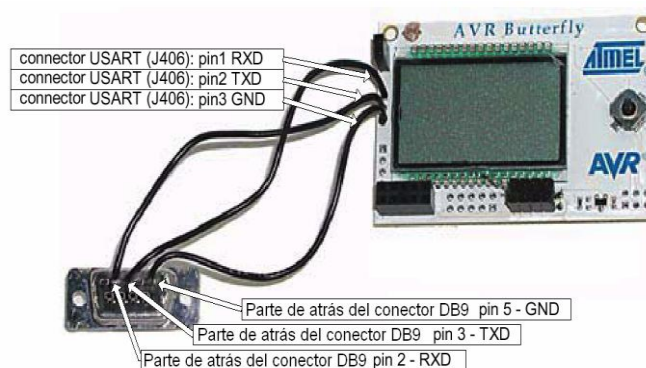
La comunicación con la PC requiere de tres líneas: TXD, RXD y GND. TXD es la línea para transmitir datos desde la PC hacia el AVR Butterfly, RXD es la línea para recepción de datos enviados desde el AVR Butterfly hacia la PC y GND es la tierra común. En la Tabla 2.1 se observa la distribución de los pines para la comunicación

serial, a la izquierda los pines del AVR Butterfly y a la derecha los pines del conector DB9 de la PC.

**Tabla. 2.1. Distribución de pines, AVR Butterfly Vs. PC**

AVR Butterfly UART	COM2
Pin 1 (RXD)	Pin 3
Pin 2 (TXD)	Pin 2
Pin 3 (GND)	Pin 5

En la Figura 2.6.1 se observa cómo se debe hacer el cableado para la comunicación, a través de la interfaz serial RS-232, entre el AVR Butterfly y la PC. A la izquierda se aprecia un conector DB9 hembra soldado a los cables que se conectan a la interfaz USART del AVR Butterfly (derecha).



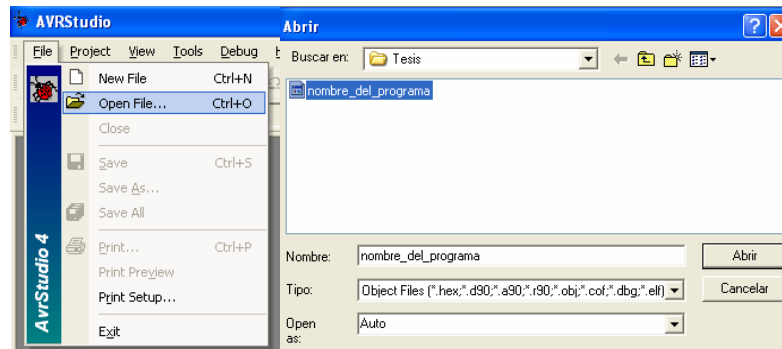
**Figura. 2.6.1 Conexiones para interfaz USART del AVR Butterfly**

## 2.6.2. Programación del AVR Butterfly

Los pasos necesarios para la Programación del Kit AVR Butterfly son los siguientes:

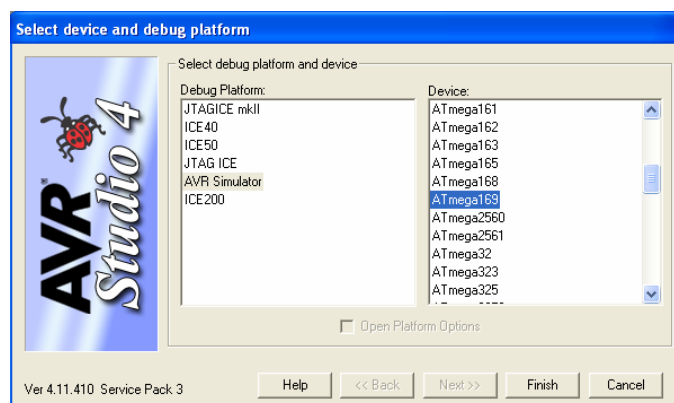
- En la PC, localizar y ejecutar el AVR Studio.

- En el menú “File” del AVR Studio seleccionar “Open File“, luego seleccionar el archivo con el que se desea programar al AVR Butterfly, tal como indica la Figura 2.6.2.0; por ejemplo: ...\`nombre_del_programa.cof`.



**Figura. 2.6.2.0. AVR Studio, selección del archivo COF para depuración**

- Seleccionar el AVR Simulator y luego el ATmega169 como en la Figura 2.6.2.1.



**Figura. 2.6.2.1. Selección del AVR Simulator y Dispositivo ATmega169**

- Presionar “Finish”.
- Conectar el cable serial entre la PC y el AVR Butterfly como se indica en la Figura 2.6.1.

- Resetear el AVR Butterfly cortocircuitando los pines 5 y 6 en el conector J403, conector ISP, o quitando y aplicando nuevamente la fuente de alimentación. Ver Figura 2.3.0.
- Luego de un reset, el microcontrolador ATmega169 comenzará desde la sección de Arranque. Nada se desplegará en el LCD mientras esté en la sección de Arranque. Entonces se deberá presionar ENTRAR en el joystick y mantener esa posición; mientras tanto desde la PC en el AVR Studio, iniciar el AVR Prog.
- Una vez que se haya iniciado el AVR Prog, soltar el joystick del AVR Butterfly. Desde el AVR Prog, utilizar el botón “Browse” para buscar el archivo con la extensión \*.hex con el que desea actualizar al AVR Butterfly. Una vez localizado el archivo \*.hex, presionar el botón “Program”. Se notará que “Erasing Device”, “Programming” y “Verifying” se ponen en “OK”, de manera automática. Luego de actualizar la aplicación, presionar el botón “Exit” en el AVR Prog para salir del modo de programación del ATmega169.
- Para que empiece a ejecutarse la nueva aplicación, resetear el AVR Butterfly cortocircuitando los pines 5 y 6 en el conector J403 y presionar el joystick hacia ARRIBA. [7]

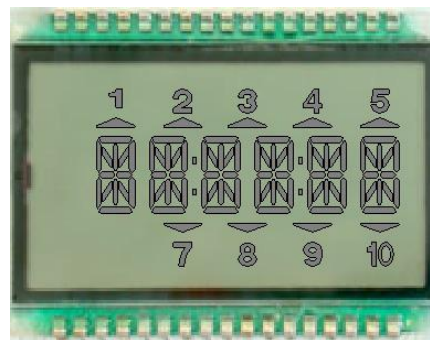
### **2.7.0 EL LCD**

En las aplicaciones donde es necesaria la interacción con el usuario es muy útil poder mostrar información para el usuario. Una interfaz muy simple para mostrar información podría ser el estado de unos LEDs; mientras que la interacción más compleja puede beneficiarse de una pantalla capaz de desplegar letras, números, palabras o incluso oraciones. Las Pantallas de Cristal Líquido (LCD) son frecuentemente usadas para desplegar mensajes. Los módulos LCD pueden ser gráficos y se los puede usar para desplegar gráficos y texto, o pueden ser alfanuméricos capaces de visualizar entre 10 y 80 caracteres. Los módulos LCD

alfanuméricos estándar son fáciles de conectar, pero son bastante costosos debido a que tienen incorporado drivers/controladores que se ocupan de la generación de los caracteres/gráficos sobre el vidrio LCD.

El vidrio LCD es la placa de vidrio en la cual el cristal líquido está contenido. Para reducir el costo de una aplicación donde se requiere una pantalla se puede elegir usar una MCU que tenga un driver incorporado para LCD. La MCU puede entonces manejar directamente el vidrio LCD, eliminando la necesidad del driver integrado en el módulo LCD. El costo de la pantalla puede reducirse tanto como para un factor de 10, puesto que un vidrio LCD (LCD sin Driver) tiene un costo mucho más bajo que un módulo LCD (LCD con Driver).

El microcontrolador ATmega169 tiene un controlador LCD (LCD Driver) integrado capaz de controlar hasta 100 segmentos. El núcleo altamente eficiente y el consumo de corriente muy bajo de este dispositivo lo hace ideal para aplicaciones energizadas por batería que requieren de una interfaz humana.

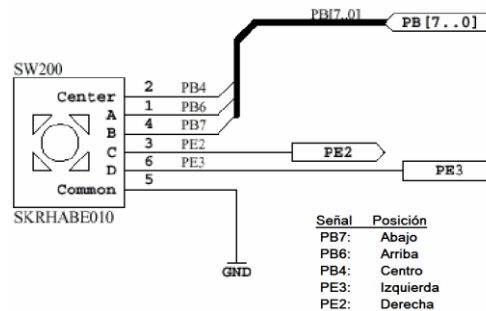


**Figura. 2.7.0 Vidrio LCD**

### **2.8.0 EL JOYSTICK**

El AVR Butterfly tiene un joystick en miniatura como entrada para el usuario. Este opera en cinco direcciones, incluyendo presión en el centro. La línea común de todas

las direcciones es GND. Esto significa que se deberá activar las pull-ups internas del microcontrolador ATmega169V para detectar la entrada desde el joystick. En la Figura 2.8.0 se puede apreciar al joystick y la referencia de los puertos del ATmega169 a los que se debe conectar.



**Figura. 2.8.0. Joystick**

## 2.9.0 AVR STUDIO 4 (SOFTWARE)

### 2.9.1 INTRODUCCIÓN

AVR Studio es un Entorno de Desarrollo Integrado (IDE) para escribir y depurar aplicaciones AVR en el entorno de Windows 9x/Me/NT/2000/XP.

AVR Studio 4 soporta varias de las fases por las cuales se atraviesa al crear un nuevo producto basado en un microcontrolador AVR. Las fases típicas son:

- La definición del producto. El producto que debe crearse se define basándose en el conocimiento de la tarea que se quiere resolver y la entrada que tendrá en el mercado.
- La especificación formal. Se define una especificación formal para el producto.
- Asignación de la tarea a un equipo. A un equipo del proyecto, que consiste de una o más personas, se le asigna la tarea de crear el producto basándose en la especificación formal.

- El equipo del proyecto pasa por la secuencia normal de diseño, desarrollo, depuración, comprobación, planificación de producción, producción, prueba y embarque.

AVR Studio apoya al diseñador en el diseño, desarrollo, depuración y parte de la comprobación del proceso.

AVR Studio es actualizado continuamente y está disponible para descargarlo desde [www.atmel.com](http://www.atmel.com)

A continuación se lista brevemente los requerimientos mínimos del sistema, necesarios para poder utilizar el AVR Studio 4 en una PC:

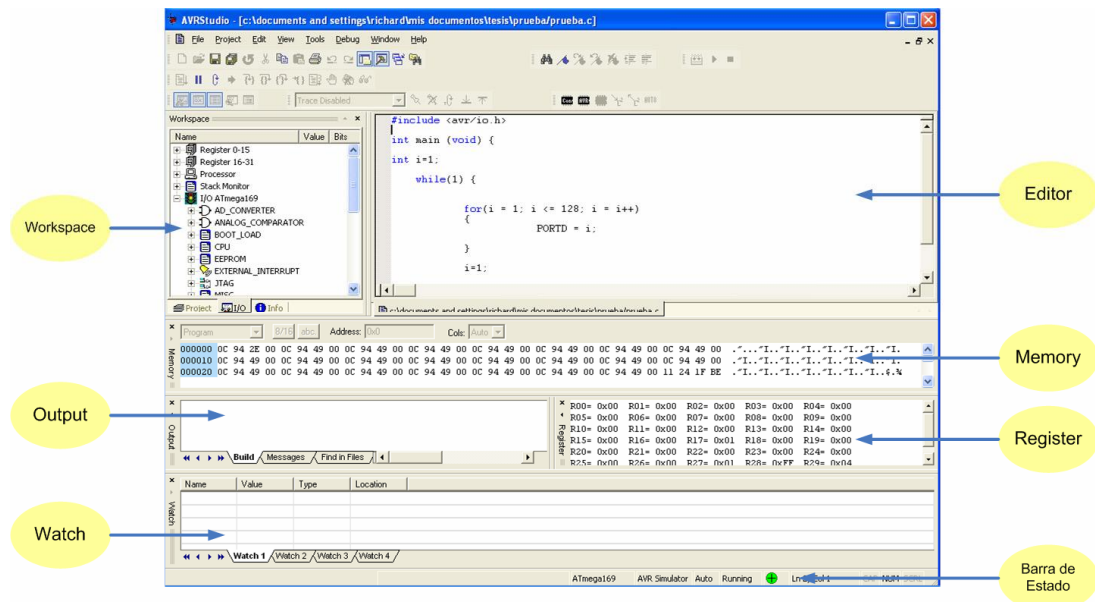
- Windows 2000/XP (o Windows NT 4.0 con Internet Explorer 5.0 o posterior).
- Windows 95/98/Me (con Internet Explorer 5.0 o posterior).
- Hardware Recomendado:
  - Procesador Intel Pentium de 200MHz o equivalente.
  - Resolución de Pantalla de 1024x768 (Resolución mínima de 800x600).
  - Memoria RAM de 64Mb.
  - Disco Duro con 50 Mb de espacio disponible.

### **2.10.0 DESCRIPCIÓN GENERAL DEL IDE**

Como se dijo anteriormente, el AVR Studio es un Entorno de Desarrollo Integrado (IDE). Éste tiene una arquitectura modular completamente nueva, que incluso permite interactuar con software de otros fabricantes.

AVR Studio 4 proporciona herramientas para la administración de proyectos, edición de archivo fuente, simulación del chip e interfaz para emulación In-circuit para la poderosa familia RISC de microcontroladores AVR de 8 bits.

AVR Studio 4 consiste de muchas ventanas y sub-módulos. Cada ventana apoya a las partes del trabajo que se intenta emprender. En la Figura 2.10 se puede apreciar



las ventanas principales del IDE.

**Figura 2.10 Entorno de Desarrollo Integrado AVR Studio4**

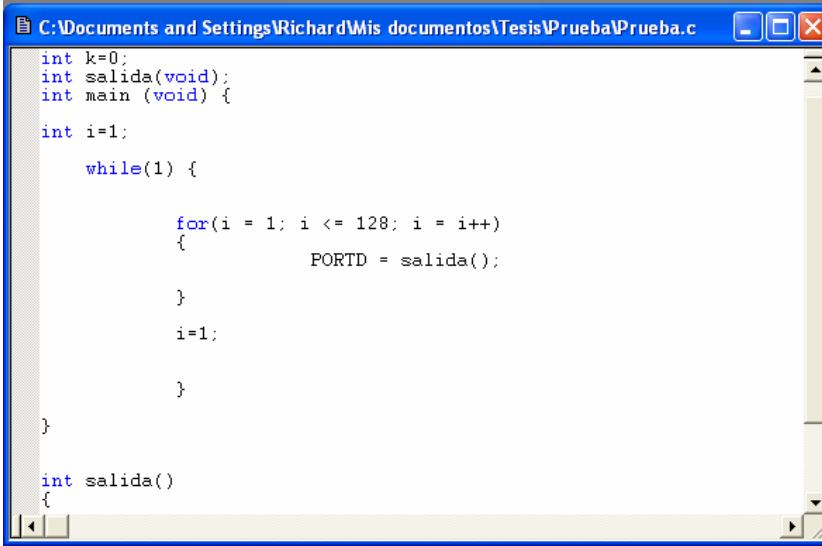
**Administración de Proyectos.** Toda creación de código dentro del AVR Studio se realiza como proyectos de programación. Todos los proyectos de código tienen un archivo project que mantiene la información acerca del proyecto, de qué archivos consta, la estructuración del ensamblador, vistas personalizadas y así sucesivamente. El archivo project asegura que el proyecto sea el mismo cada vez que regrese a él, que todo esté adecuadamente organizado y que ensamble / compile.



## 2.11.0 LAS VENTANAS PRINCIPALES DEL IDE AVR STUDIO 4

### 2.11.1. La Ventana Editor

Este es el lugar donde se realiza el trabajo de codificar. El AVR Studio usa el editor Stringray de la corporación Bsquare. Este es un editor completo para programación con toda la funcionalidad que el usuario empleará, incluso codificación de color para la sintaxis (que puede ser alterada y extendida por el usuario). La ventana del editor también se usa cuando al depurar código. En la Figura 2.11.1 se aprecia la ventana del Editor.

The image shows a screenshot of the AVR Studio Editor window. The title bar reads "C:\Documents and Settings\Richard\My documents\Tesis\Prueba\Prueba.c". The code is written in C and uses syntax highlighting. The code includes a main function with a while loop containing a for loop, and a separate salida() function.

```
int k=0;
int salida(void);
int main (void) {

int i=1;
    while(1) {

        for(i = 1; i <= 128; i = i++)
        {
            PORTD = salida();
        }

        i=1;
    }

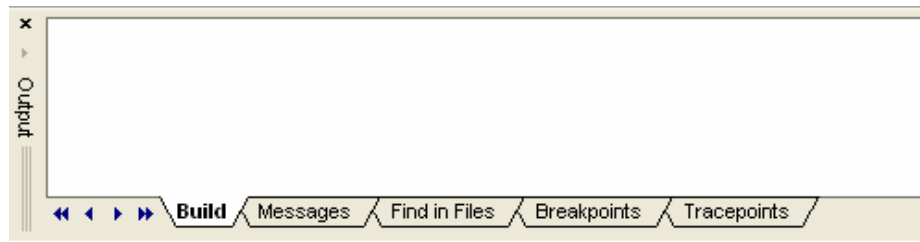
}

int salida()
{
```

Figura. 2.11.1 Ventana del Editor

### 2.11.2. La Ventana Output (Ventana de Resultados)

Es una colección de varias ventanas (Build, Messages,...) integradas en un marco etiquetado. Con las etiquetas sobre el marco se selecciona la ventana que se desea observar. En la Figura 2.11.2 se observa la ventana de Resultados (Output) y en la misma se aprecia las distintas etiquetas correspondientes a las ventanas que la conforman.



**Figura. 2.11.2. Ventana Output**

Las opciones que se presentan en esta ventana son las siguientes:

Build , messages , breakpoint y find in files. El resultado del compilador/ensamblador se dirige hacia esta ventana. Aquí puede leerse el resultado de la compilación / ensamblaje.

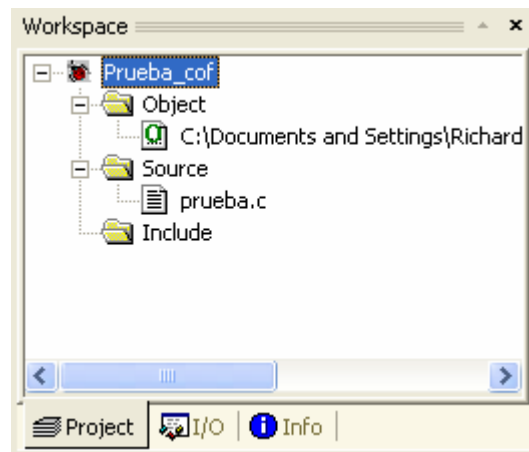
### **2.11.3. La Ventana Workspace (Área de Trabajo)**

El área de trabajo del AVR Studio 4 consiste de varias ventanas proyectadas para ayudar al desarrollador en la depuración del código que él ha escrito de forma sistemática.

Las ventanas disponibles dentro de la ventana Workspace son las siguientes:

#### **a) La ventana Project**

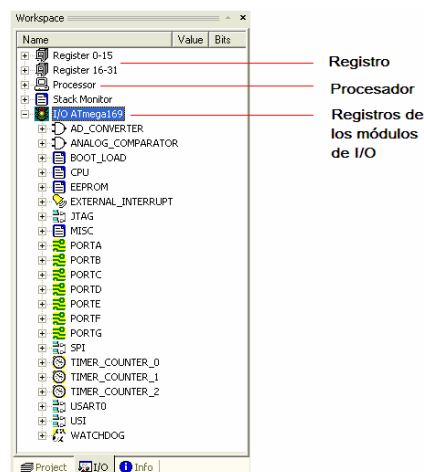
Si se ha creado un proyecto de código, la ventana Project lista los archivos que están incluidos en dicho proyecto. Si se tiene abierto un archivo objeto para depuración, la ventana Project muestra el nombre del archivo objeto actualmente cargado así como también los archivos fuente a los cuales el archivo objeto se refiere. En la Figura 2.11.3 se puede apreciar la vista Project.



**Figura. 2.11.3.0 Ventana Project**

### b) La ventana I/O

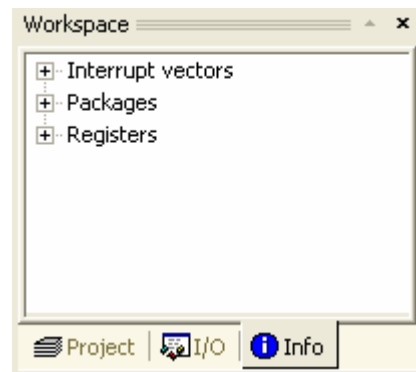
Esta ventana despliega información de los registros de I/O así como también del procesador y de los registros.



**Figura. 2.11.3.1. Ventana I/O**

### c) La ventana Info

Esta ventana es estática y muestra todas las interrupciones, configuración de pin y direcciones disponibles de I/O para el dispositivo seleccionado.

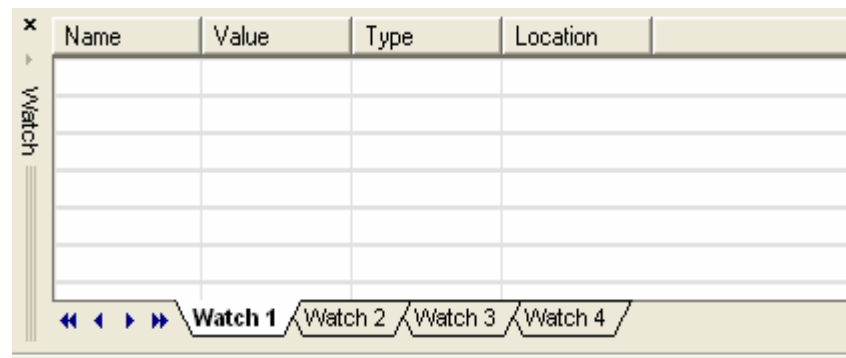


**Figura. 2.11.3.2. Ventana Info**

Como se aprecia en la Figura 2.11.3.2, la ventana Info se subdivide en los siguientes campos: Interrupt vectors ,packages ,register .

#### 2.11.4. La Ventana Watch

En la Figura 2.11.4 se aprecia la ventana para vigilancia de variables, esta es la ventana Watch.



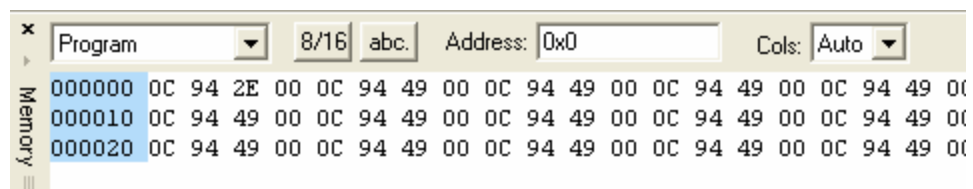
**Figura. 2.11.4. Vista Watch**

Al depurar lenguajes de alto nivel como C/C++, se necesita vigilar las variables. Con herramientas como AVR Studio 4 esto es una tarea fácil, se debe hacer clic sobre la variable que se quiera vigilar, arrastrarla y soltarla dentro de la ventana para

vigilancia (Watch). Si la variable seleccionada es una estructura o una matriz, un símbolo [+] aparecerá en frente de la variable, indicando que ésta puede ser expandida dentro de la vista. Cuando la ejecución se interrumpe temporalmente (break), la variable se actualizará automáticamente si está dentro del alcance. Diferentes lenguajes definen diferentemente su alcance, pero estar dentro del alcance significa que la variable actualmente existe en la memoria en la localidad donde el programa detiene su ejecución.

### 2.11.5. La Ventana Memory

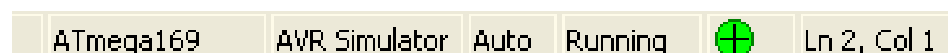
Un microcontrolador como el AVR no puede hacer nada sin memoria. El código ejecutado está en la memoria de programa, las variables se ubican en la SRAM (principalmente), los registros de I/O se distribuyen dentro del área de memoria de I/O y la EEPROM es otra área de memoria. La ventana Memory está disponible para visualizar todos los diferentes tipos de memoria asociados a los dispositivos AVR. Y en relación a la ventana Watch y la I/O, el área expuesta en la pantalla se actualiza automáticamente cuando ocurre una detención temporal (break) de la ejecución. En la Figura 2.11.5 se aprecia la ventana Memory.



**Figura. 2.11.5. Vista Memory**

### 2.11.6. La Barra de Estado

La barra de estado siempre indica el nombre del dispositivo actual, la plataforma de depuración, el estado de ejecución y la ubicación del cursor dentro de la ventana Editor; esto se puede verificar en la Figura 2.11.6.



**Figura. 2.11.6. Barra de Estado**

# CAPITULO 3

## IMPLEMENTACION DEL PROYECTO

### 3.0.0. GENERALIDADES

En este capítulo se describe la plataforma en la que se va a implementar nuestros proyectos.

Una descripción de los componentes y un análisis de las conexiones utilizadas entre el kit AVR butterfly y los periféricos utilizados.

Describiremos como trabaja la plataforma en cada uno de los proyectos que se va a implementar y su forma de uso.

### 3.1.0. Componentes

- Kit de desarrollo AVR butterfly
- Protoboard(BREAD BOARD)
- Porta pilas AA
- Pilas AA(1.5V)
- Cable UTP
- Resistencias
- Diodo led
- Conector DB9 hembra

### 3.2.0 Implementación

Vamos a describir las conexiones entre el Kit de desarrollo AVR butterfly, protoboard y la comunicación serial con el computador.

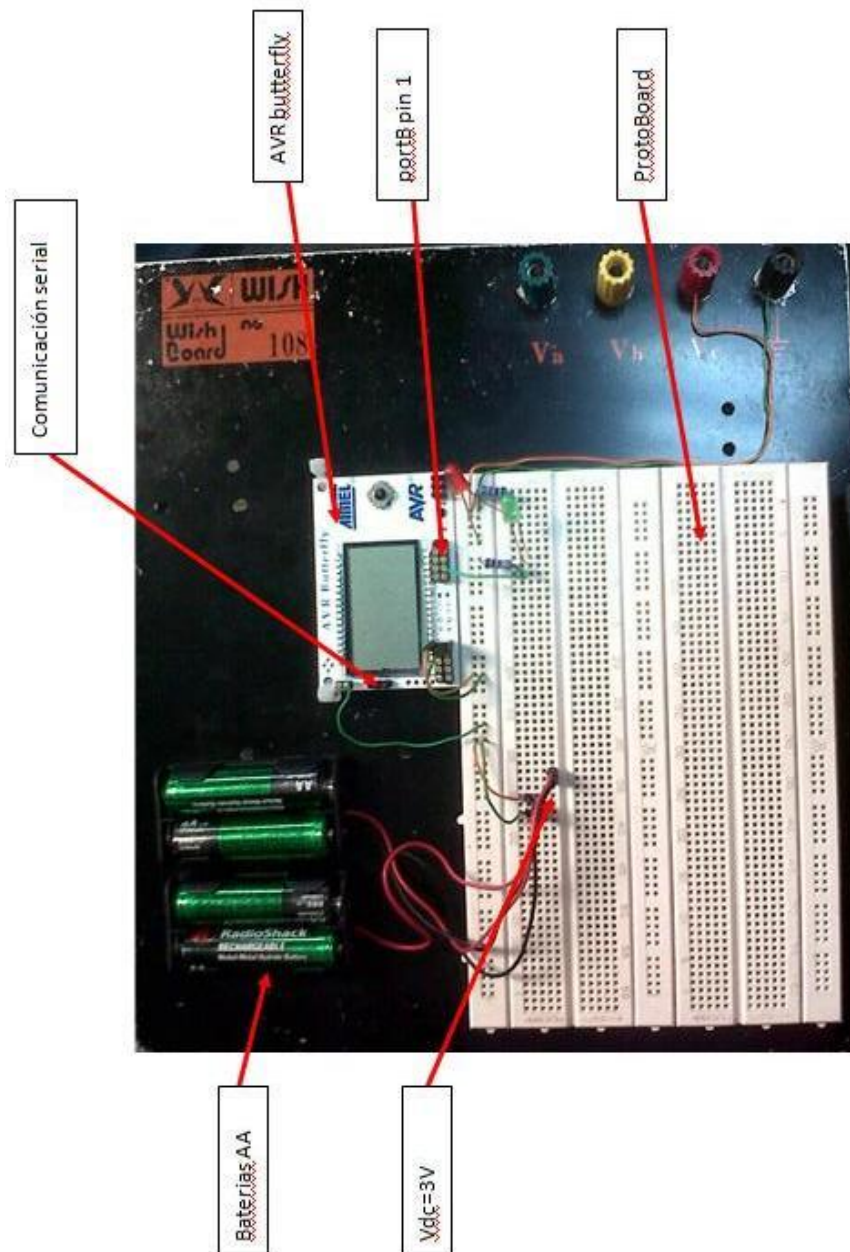


Figura. 3.2.0 Maqueta

### 3.3.0 Implementación de proyectos

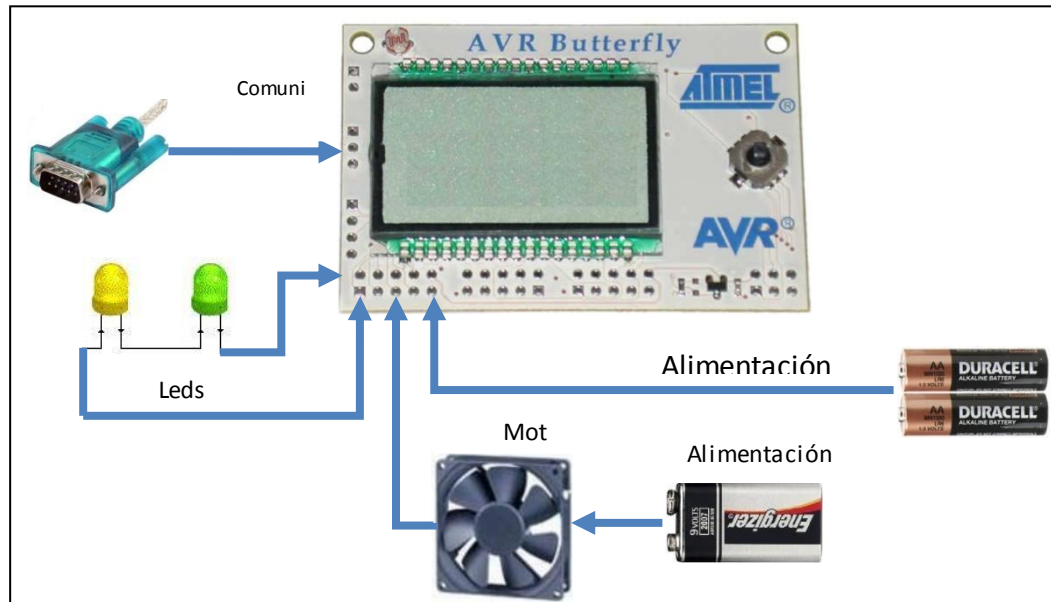


Figura. 3.3.0. Esquema de conexiones

#### 3.3.1. Contador de segundos Up/Down usando el timer1

En este proyecto se usará el timer1 en el modo libre. Se configurará a una frecuencia del preescalador  $f_{cu}/64$  se usará la interrupción por cambio de pin en el puerto E donde están conectados los botones del joystick derecha e izquierda para indicar si cuenta ascendente o descendente.

Se da uso de la LCD para mostrar el conteo de los segundos y se muestra la rutina para inicializar la pantalla LCD incorporada en el Kit Avr Butterfly.

El programa inicia y necesita de los botones derecha del joystick para contador up y el botón izquierda para el contador down.



### 3.3.2. Uso del timer1 modo CTC doble comparación 1A y 1B

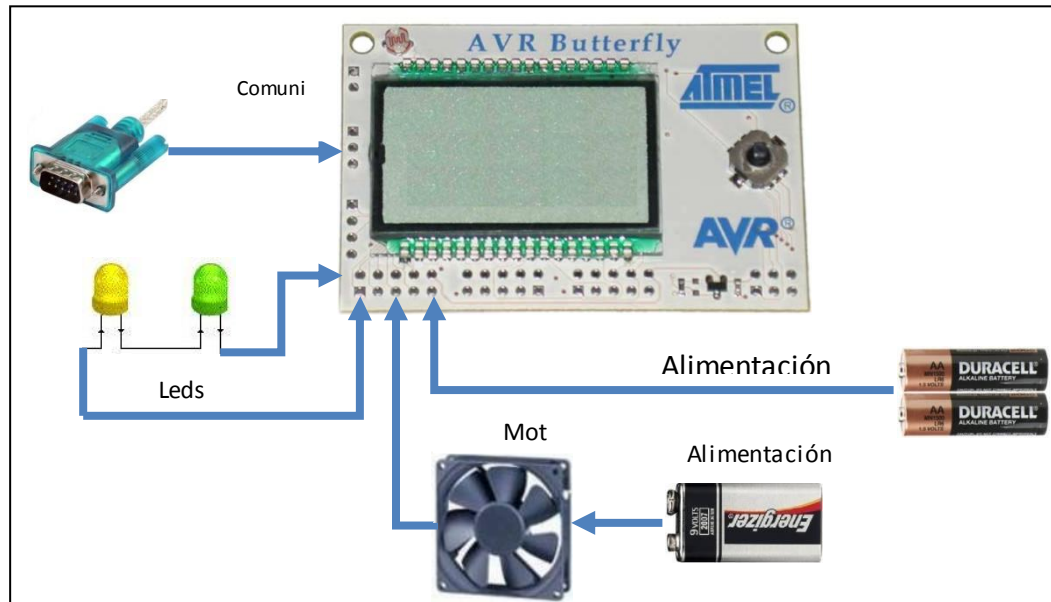


Figura. 3.3.1. Esquema de conexiones

En el siguiente proyecto se estudiará el timer1 en el modo de comparación doble usando los registros de comparación OCR1A y OCR1B.

El proyecto al encender el timer1 empieza la cuenta ascendente y cuando llega al primer de valor puesto de comparación enciende el led ubicado en el portC bit cero y cuando alcanza el siguiente valor se apaga el led.

### 3.3.3. Reloj a diferentes frecuencias

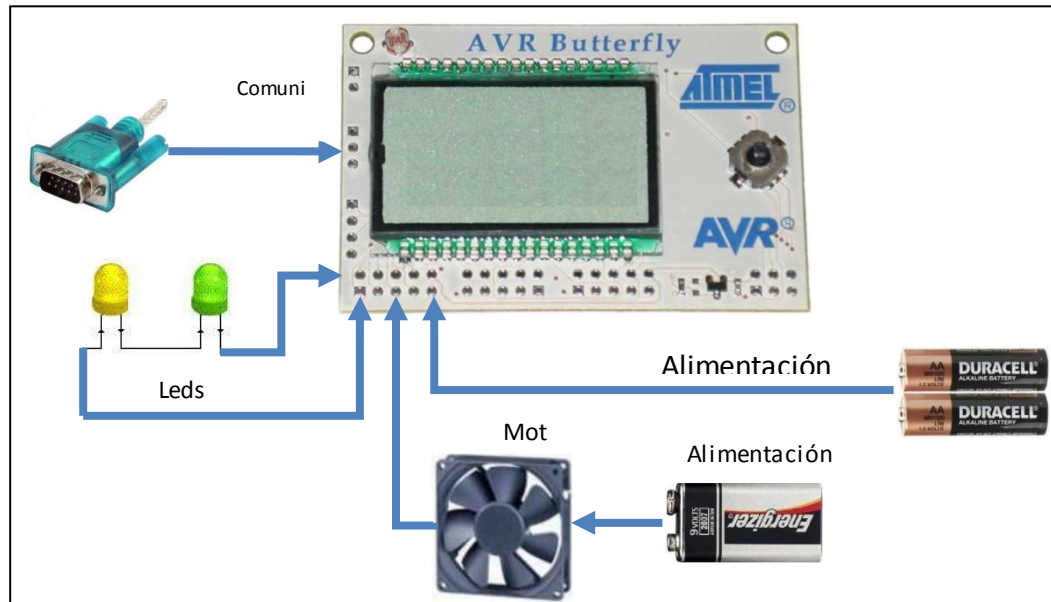


Figura. 3.3.2. Esquema de conexiones

En el siguiente proyecto vamos a generar una onda de periodo constante dependiendo del botón presionado en el joystick:

Botón izquierda 4 ms

Botón derecha 3 ms

Botón arriba 1 ms

Botón abajo 0.1 ms

La onda de salida la observamos en un led conectado en el pin 6 del puerto C

### 3.3.4. Control PWM de intensidad de luz

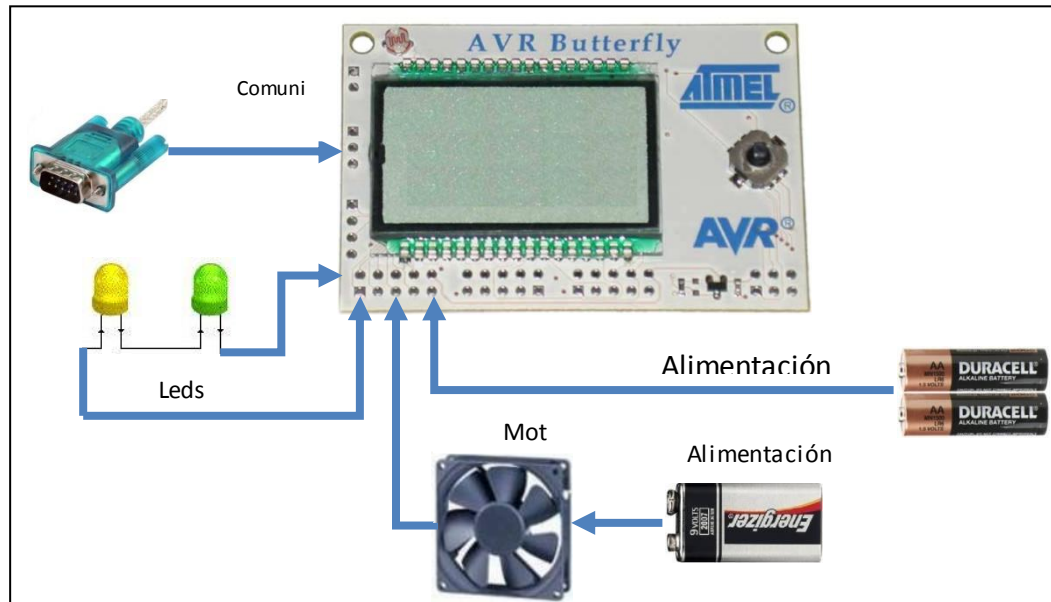


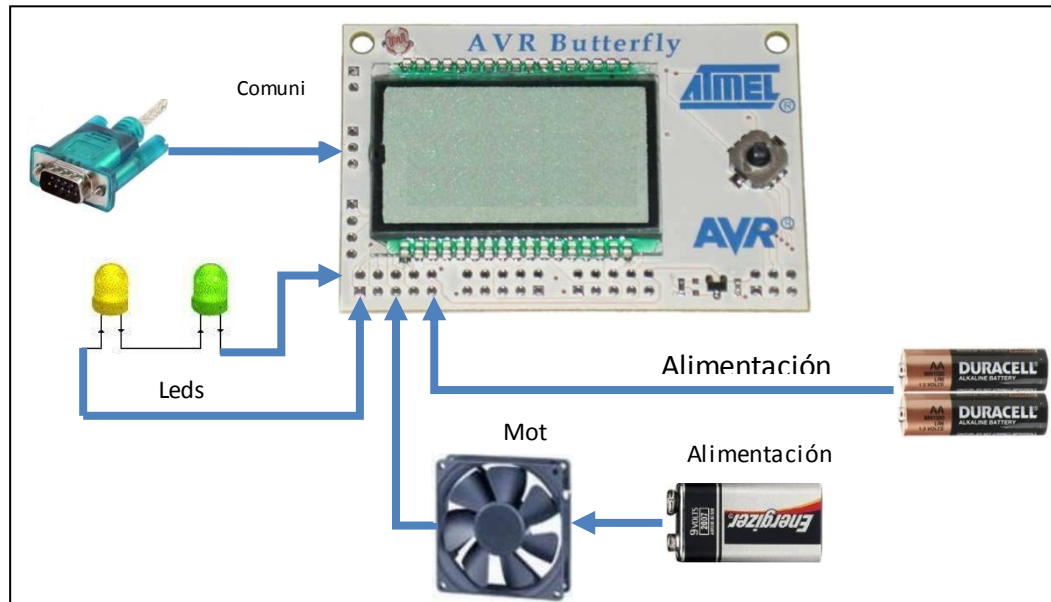
Figura. 3.3.3. Esquema de conexiones

Este proyecto se realizó para aprender a calibrar el timer1 en el modo correcto pwm. Variando el valor promedio de salida en el pin 5 del puerto B, donde estará conectado un led para mostrarnos la variación en la intensidad de luz.

Cuando encendemos el Kit AVR buterffly la intensidad de luz por default es puesta en un ciclo de trabajo del 10 %, cuando se presiona el botón derecho del joystick la frecuencia del PWM cambia a un ciclo de trabajo del 50%.

Usamos la interrupción por cambio de pin para cambiar la frecuencia de salida al 90% cuando presionamos el botón izquierdo del joystick.

### 3.3.5 Control variable (CTC y PWM)



**Figura. 3.3.4. Esquema de conexiones**

En el siguiente proyecto se estudiara el timer 1 con la programación en lenguaje assembler y analizaremos el modo de comparación usando el registro OCR1A en modo CTC y a la vez podemos cambiar al modo de control PWM.

Cuando los registros del timer 1 lleguen a la comparación A entonces habilitamos la salida del motor y en el otro periodo no se habilita el motor. En el otro modo se usa el control PWM del timer1 para manejar el motor y cambiamos de modo con los botones derecha (PWM) y el botón izquierda(CTC)

# CAPITULO 4

## PROYECTOS

### 4.0 GENERALIDADES

En este capítulo vamos a estudiar los proyectos realizados con el Kit Avr Butterffy con respecto al tema del módulo TIMER/COUNTER1 una descripción del código de los proyectos junto con un diagrama de flujo de cada proyecto para entendimiento del lector

El Timer 1 es un módulo temporizador/contador de 16 bits, que consiste en dos registros de 8 bits (TMR1H y TMR1L) que son de lectura y escritura en los proyectos usaremos el timer 1 en modo libre ,módulo de captura , modo PWM y usaremos el sistema de interrupciones correspondientes al timer1, usando el lenguaje C y assembler.

#### 4.1.0. Contador de segundos Up/Down usando el timer1

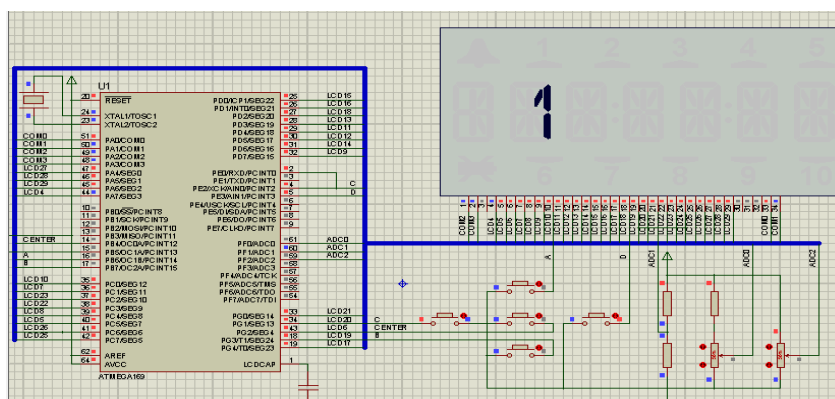


Figura. 4.1.0. Contador de segundos Up/Down usando el timer1

En este proyecto se usará el timer1 en el modo libre. Se configurará a una frecuencia del preescalador  $f_{cu}/64$ , se usará la interrupciones por cambio de pin en el puertoE donde está conectados los botones del joystick derecha e izquierda para indicar si cuenta ascendente o descendente.

Se da uso de la LCD para mostrar el conteo de los segundos y se muestra la rutina para inicializar la pantalla LCD incorporada en el Kit Avr Buterffly

El programa inicia y necesita de los botones derecha del joystick para contador up y el botón izquierda para el contador down.

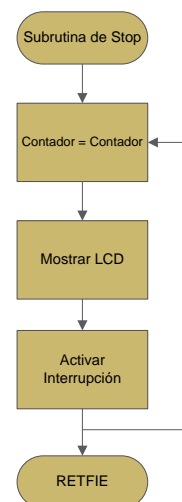
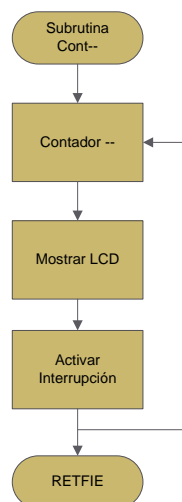
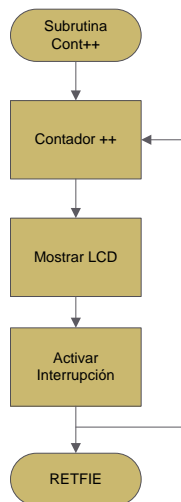
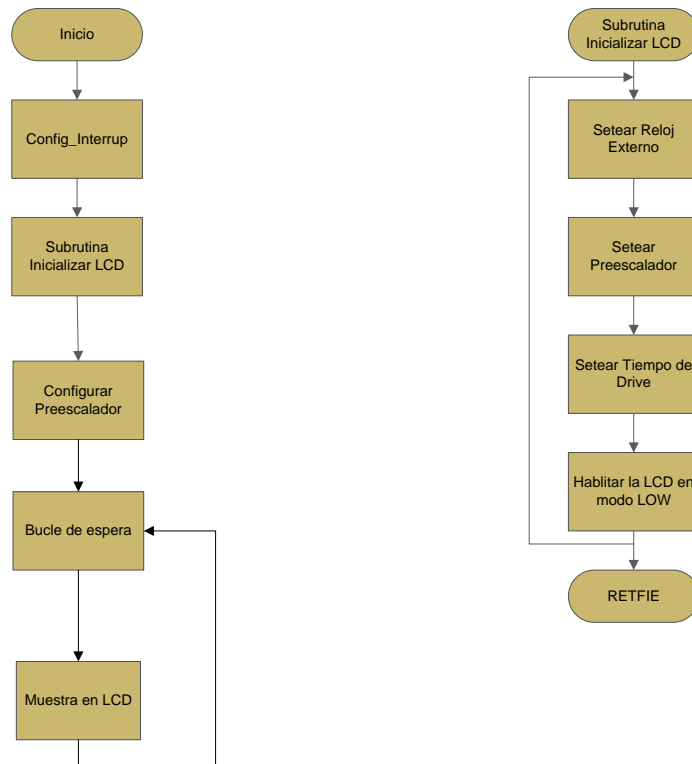
#### **4.1.1. Fundamento teórico**

Se realizó este proyecto para analizar la operación en modo libre del timer 1 que es lo más básico para empezar a conocer el TIMER/COUNTER1. Usamos el reloj interno del atmega169 con una frecuencia del preescalador a  $f_{cu}/64$  y se resetea el módulo TCNT1 cuando alcance el valor de 15624 que es el tiempo de 1 segundo con un preescalador de 64.

$$\begin{aligned} \text{Tiempo de conteo} &= (\text{Frecuencia de entrada/Prescaler}) / \text{Frecuencia objetivo} - 1 \\ &= (1\text{Mhz}/64) / 1\text{Hz} - 1 \\ &= 15624 \end{aligned}$$

Luego contamos nueve segundos y lo mostramos con una rutina en la pantalla LCD para aprender a mandar datos a la misma.

#### 4.1.2. Diagrama de flujo



### **Algoritmo**

1. Inicio.
2. Configuramos la Interrupción.
3. Llamamos a la Subrutina Inicializar LCD, la cual consiste en setear el reloj externo, posteriormente configuramos el preescalador para setear el tiempo de drive, y por último habilita la LCD en modo low.
4. Si se presiona el pin E2 se ejecutará la subrutina UP, que consiste en aumentar la variable contador en incrementos unitarios, se activa la interrupción y posteriormente el valor de la variable contador se muestra en la LCD.
5. Si se presiona el pin E3 se ejecutará la subrutina DOWN, esta consiste en decrementar la variable contador en decrementos unitarios, se activa la interrupción y posteriormente el valor de la variable contador se muestra en la LCD.
6. Fin.



### 4.1.3. Código en lenguaje C

```

/* ****
Programa 1
Contador de segundos usando el timer1 y demostración de interrupciones
Rommel Chang-Jefferson Moreno
7/09/2011
*****/

#include <avr/io.h> // io.h contiene la definición de los puertos y permite acceder a
ellos por dirección
#include <util/delay.h>
#include <avr/interrupt.h>

void LCD_INIT(); // Función para inicializar el LCD
void Escribir_Lcd(int j); // Funcion para Escribir en la LCD

int main(void)
{

    EIMSK = 0xC1; // [PCIE1,PCIE0,-,-,-,-,INT0]
    EIFR = 0xC0; // [PCIF1,PCIF0,-,-,-,-,INTF0]
    PCMSK0=(1<<PCINT3|1<<PCINT2); //activo las
interrupciones0..7[PCINT3,PCINT2]
    EICRA=0x02; // [ISC01,ISC00] tipo de flanco
    PORTE|=0x0C; // habilita resistencia de pull-up
    sei(); //setea en SREG la Bandera I
    LCD_INIT(); // va a la rutina que inicializa el
LCD
    TCCR1B |= ((1 << CS10) | (1 << CS11)); //Configura el timer con un preescalador
Fcu/64

    while(1) // Lazo infinito

    {

    }

}
ISR(SIG_PIN_CHANGE0) //Rutina de interrupción

```

```

{ cli();
  if(!(PINE & (1<<PINE2)))          //cuando se presiona hacia la izquierda
      {
        unsigned char diezsegundos = 1; // Creando una variable e igualando a cero
for (;;)                          //Lazo for infinito
    {
      if (TCNT1 >= 15624)          //Preguntamos si el timer1 llego a 1 seg
        {
          TCNT1 = 0;              // Reseteamos el timer1
          Escribir_Lcd(diezsegundos); //muestre LCD
                                   sei();
                                   diezsegundos++;

          if(diezsegundos == 11) // Preguntamos si ya paso el minuto
            {
              diezsegundos = 1; // Reseteamos la variable de conteo
            }
          }
        }
      }
    }

  if(!(PINE & (1<<PINE3)))          // cuando se presiona a la derecha
      {
        unsigned char diezsegundos =10; // Creando una variable e igualando a cero
for (;;)                          //Lazo for infinito
    {
      if (TCNT1 >= 15624)          //Preguntamos si el timer1 llego a 1 seg
        {
          TCNT1 = 0;              // Reseteamos el timer1
          Escribir_Lcd(diezsegundos); //muestre LCD
          sei();
          diezsegundos--;
          if(diezsegundos ==0) // Preguntamos si ya paso el minuto

```

```

        {
            diezsegundos = 10; // Reseteamos la variable de conteo
        }
    }
}

int tabla[10][4]= { 0x01,0x05,0x05,0x01,          //Tabla para escribir en la LCD los
números
                    0x08,0x01,0x01,0x00,
                    0x01,0x01,0x0E,0x01,
                    0x01,0x01,0x0B,0x01,
                    0x00,0x05,0x0B,0x00,
                    0x01,0x04,0x0B,0x01,
                    0x01,0x04,0x0F,0x01,
                    0x01,0x01,0x01,0x00,
                    0x01,0x05,0x0F,0x01,
                    0x01,0x05,0x0B,0x00};

void LCD_INIT()          //Rutina para inicializar la LCD

{

// Seteamos el reloj externo a 1/4 del ciclo de trabajo, 25 segmentos
LCDCRB =(1<<LCDCS)|(3<<LCDMUX0)|(7<<LCDPM0)
;//[LCDCS,LCD2B,LCDMUX1,+MUX0,-,LCDPM2,+PM1,+PM0]
//Setamos el preescalador a /16, Divisor del clock a /8
LCDFRR = (0<<LCDPS0)|(7<<LCDCD0);//[-,LCDPS2,+PS1,+PS0,-,LCDCD2,+CD1,+CD0]
//Seteamos el tiempo del drive a 330uS, Contraste a 3.30 VOLTS
LCDCCR=(1<<LCDCC3)|(1<<LCDCC2)|(1<<LCDCC1);//[LCDDC2,+DC1,+DC0,LCDMDT,LCDCC3,+CC2,+CC1,+CC0]
//Habilitamos la LCD en modo de poder bajo
LCDCRA=(1<<LCDEN)|(1<<LCDAB);//[LCDEN,LCDAB,-,LCDIF,LCDIE,LCDBD,LCDCCD,LCDBL]
}

//=====
//=====

// Rutina que escribe la palabra en el LCD segun la Tabla y el valor de j
void Escribir_Lcd(int j)
{
    unsigned char p = 0;
    p=j-1;

```

```
LCDDR0 = tabla[p][0];  
LCDDR5 = tabla[p][1];  
LCDDR10 = tabla[p][2];  
LCDDR15 = tabla[p][3];  
}
```

## 4.2.0 Uso del timer1 modo CTC doble comparación 1A y 1B

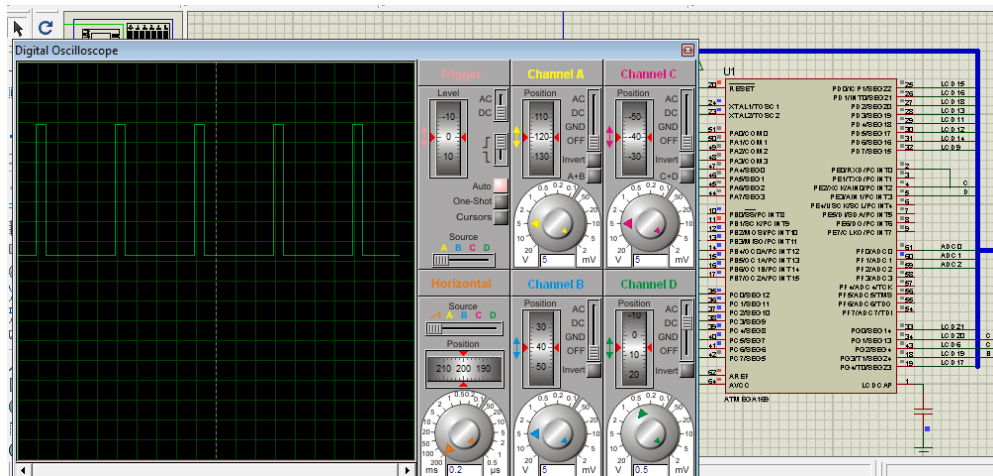


Figura. 4.1.1 Uso del timer1 modo CTC doble comparación 1A y 1B

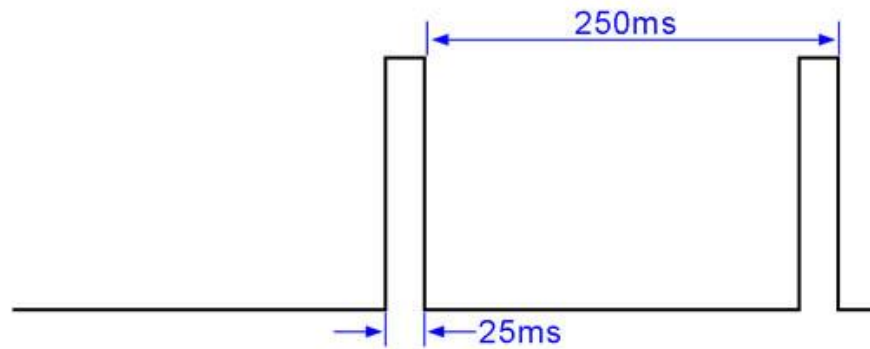
En el siguiente proyecto se estudiará el timer 1 en el modo de comparación doble usando los registros de comparación OCR1A y OCR1B.

Cuando los registros del timer 1 lleguen a la comparación B entonces apagamos un led en el puerto C y en la comparación del registro A lo encendemos, usaremos unos tiempos de comparación para producir la siguiente onda con un clock de  $f_{cu}/1024$

### 4.2.1. Fundamento teórico

Se realizó el siguiente proyecto para aprender a calibrar y utilizar el módulo de comparación del timer 1, usamos los registros OCR1A y OCR1B para hacer una doble comparación con respecto al conteo de TNCT1. Cuando el registro del timer1 alcanza el primer valor de comparación encendemos un led y cuando llegue al segundo punto de comparación apagamos. Dependiendo de los valores que se carguen en los registros de comparación se puede conseguir una señal como la de la figura, con un preescalador igual a  $f_{cu}/1024$ .

El uso de interrupciones por el modulo CTC es otro de los objetivos del proyecto para analizar y usar en diferentes tipos de proyectos



**Figura. 4.1.2** Ciclo de trabajo.

Registros de comparación

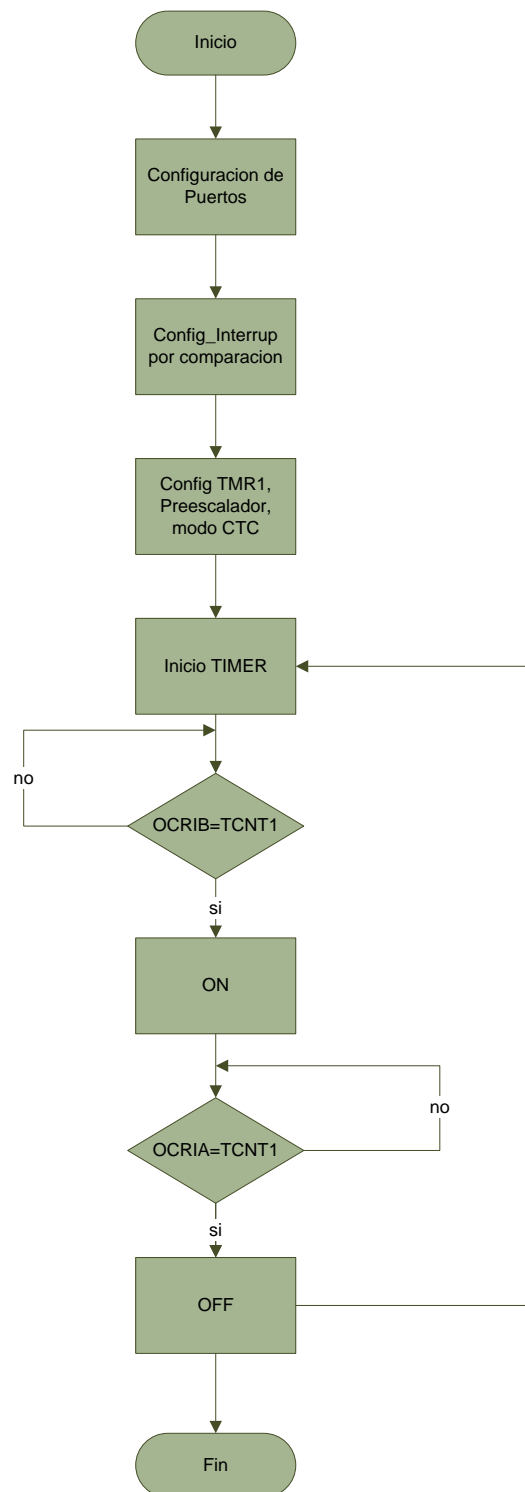
OCR1A = 1954;

Equivale a  $0.001024 * 1954 = 2\text{ms}$

OCR1B = 1929;

$0.001024 * 1929 = 1.975\text{ms}$

#### 4.2.2. Diagrama de flujo



### **Algoritmo**

1. Inicio.
2. Configuración de puertos
3. Configuración de la interrupción por comparación.
4. Configuramos el TIMER1 en el modo CTC.
5. Iniciamos el timer.
6. Si el registro de comparación B (OCR1B) es igual al Timer (TCNT1) encendemos el led indicador de que se está realizando la comparación B, si el registro de comparación no es igual al timer, entonces esperamos a que los valores tanto del registro de comparación con el del timer sean iguales.
7. Luego de la realizar la comparación entre el registro de comparación B con el timer, procedemos a realizar la otra comparación con el registro de comparación A (OCR1A), cuando este registro coincide con el valor del timer se produce la otra comparación enciendo el otro led indicador, previamente habiendo deshabilitado la otra comparación.
8. Fin.



### 4.2.3. Código

```

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

//Ejercicio 2
//Uso del timer1 modo CTC doble comparacion 1A y 1B
//Rommel Chang Suarez
//Jefferson Moreno Briones
//*****
*****

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#define green_led_on() PORTC |= _BV(0) // Indicador ON en portC
pin cero
#define green_led_off() PORTC &= ~_BV(0) // Indicador OFF en
portC pin cero
int main (void)
{
    DDRB = 0b11111111; // Puerto B como salida
    DDRC = 0b01111111; // Puerto C como salida
    TIMSK1 = _BV(OCIE1A) | _BV(OCIE1B); // Habilito la interrupcion
or timer1, Salida de comparacion A & B
    TCCR1B = _BV(CS12) | _BV(CS10) | _BV(WGM12); // Reloj /1024, 0.001024
segundos y habilito modo CTC
    OCR1A = 1954; // 0.001024*1954 ~= 2
SIG_OUTPUT_COMPARE1A se activa cada 2 segundos (led off)
    OCR1B = 1464; // 0.001024*1464 ~= 1.5
SIG_OUTPUT_COMPARE1B se activa 500ms antes de
SIG_OUTPUT_COMPARE1A (led on)
    sei(); //Habilito todas las interrupciones
    while(1)
    {
        sweep();
    }
}
void sweep() //rutina para espera la
interrupcion por comparacion
{

```

```
PORTB = 0b10000000;
for (int i=0;i<8;i++)
{
    _delay_ms(100);
    PORTB >>= 1;
}
}
ISR(SIG_OUTPUT_COMPARE1A) // interrupcion modulo de
comparación 1A
{
    green_led_off();
}
ISR(SIG_OUTPUT_COMPARE1B) //interrupcion modulo de
comparación 1B
{
    green_led_on();
}
```

### 4.3.0. Reloj a diferentes frecuencias

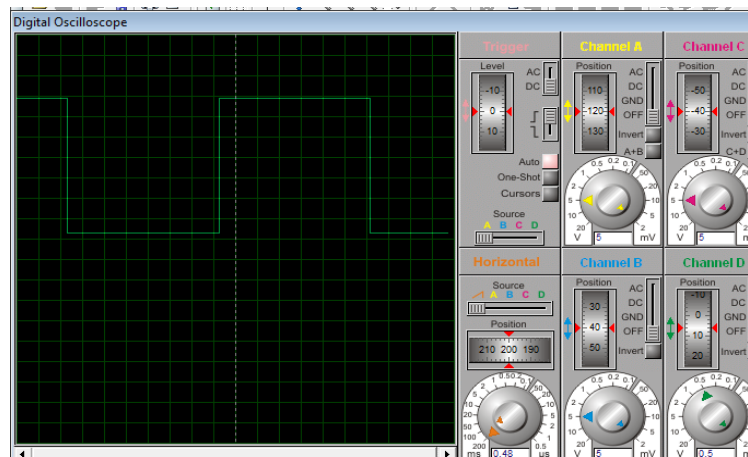


Figura. 4.1.3 Reloj a diferentes frecuencias.

En este proyecto vamos a usar el timer1 para crear un reloj y tenemos la opción de poder variar su periodo.

Habilitamos la interrupción por sobrepaso y variamos el valor del registro TCNT1 para lograr el periodo deseado

La oscilación o nuestro clock lo obtenemos en el pin 5 del portC

#### 4.3.1. Fundamento teórico

Este proyecto se desarrolló para crear una fuente de reloj basado en el timer/counter1 en modo libre (ciclo de trabajo 50%). Usaremos la interrupción por sobrepaso del timer1 para generar nuestra onda cuadrada.

Se utiliza la interrupción por cambio de pin para cambiar la frecuencia del reloj para esto usamos los botones del joystick y la señal de salida la activamos en el pin 6 del puertoC .

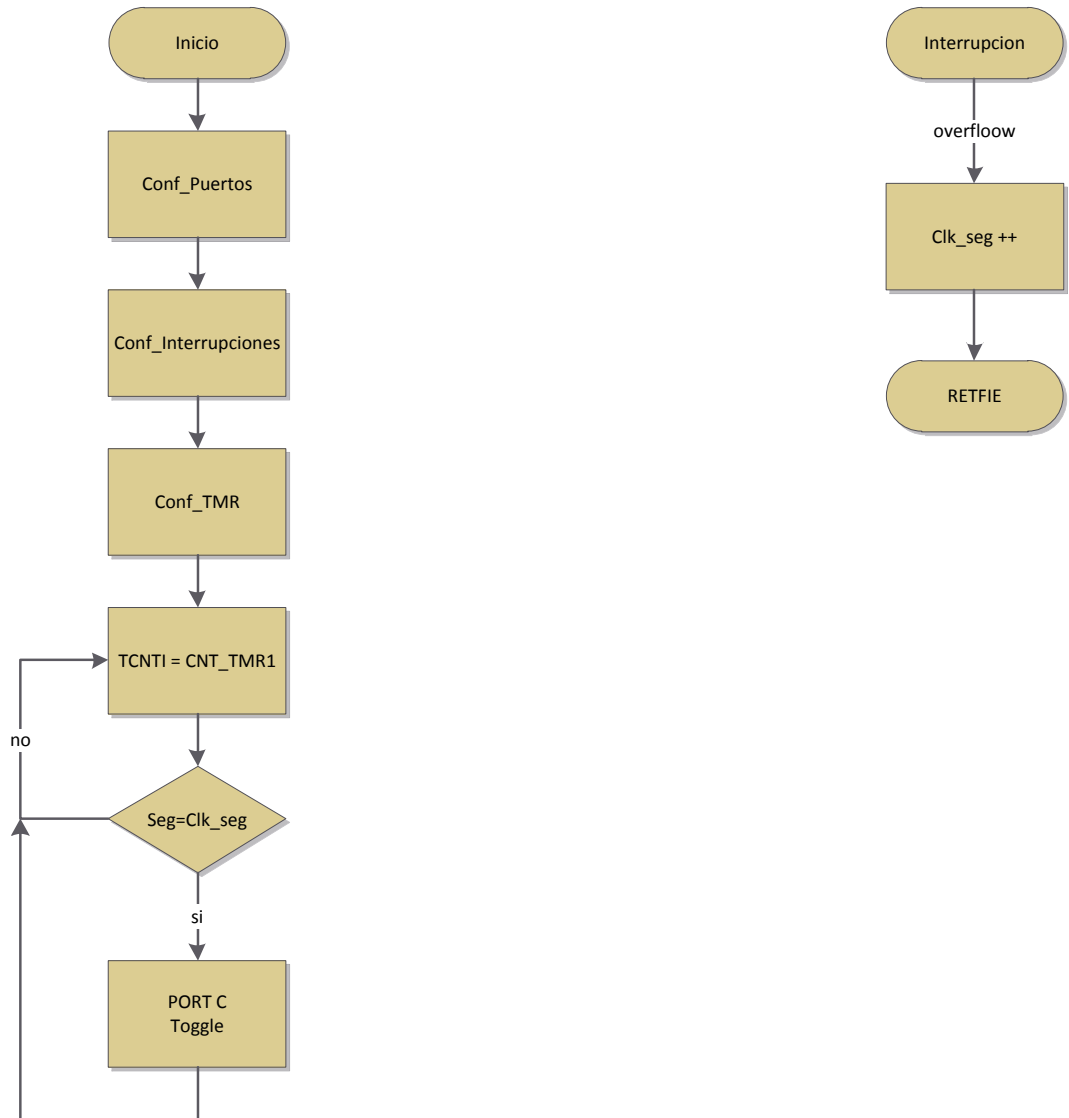
Seteamos el valor del registro TCNT1 para empezar en el momento justo para el periodo de reloj

$$\text{CNT\_timer1} = 0\text{xFFFF} - \text{CLK\_ms} * k; \quad //k= 16 = 1\text{ms}, k=160 = 10\text{ms}$$

donde CLK\_ms=10

$$\text{TCNT1} = \text{CNT\_timer1};$$

### 4.3.2. Diagrama de flujo



**Algoritmo**

1. Inicio.
2. Configuración de puertos.
3. Configuración de interrupciones, y cada vez que sea ejecutada esta interrupción la variable Clk\_seg se incrementará de uno en uno.
4. Configuración del timer según la frecuencia de oscilación deseada.
5. Al timer (TCNT1) se le asigna el valor de la variable CNT\_TMR1.
6. Se compara la variable Seg con la variable Clk\_seg, y si son iguales se puede apreciar la oscilación en el puerto C
7. Fin.

### 4.3.3. Código

```

//*****
*****
//Ejercicio 3
//Uso del timer1 interrupcion de sobrepaso y reloj
//Rommel Chang Suarez
//Jefferson Moreno Briones
//*****
*****
#include <avr/io.h>
#include <avr/interrupt.h>
#define      CLK_ms 10      // seteamos la constante a 10
unsigned int  CNT_timer1;    // El tiempo de retardo
volatile unsigned int CLK_ticks = 0;    // El numero corriente de ms
volatile unsigned int CLK_seconds = 0;    // El numero corriente de segundos
SIGNAL(SIG_OVERFLOW1)
{
    CLK_ticks += CLK_ms;        // Rutina de interrupcion
    if(CLK_ticks >= 200)
        {
            // Número de interrupciones entre cambio de
la salida
            CLK_ticks = CLK_ticks - 200;
            CLK_seconds++;
        }
    TCNT1 = CNT_timer1;
}
void CLK_setup(){
    TCCR1A = (0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) |
(0<<WGM11) | (0<<WGM10) | (0<<FOC1A) | (0<<FOC1B);    // Deshabilitamos
el modo PWM y el módulo de captura
    TCCR1B = (0<<WGM12) | (0<<WGM13) | (0<<ICNC1) | (0<<ICES1) | (1<<CS12) |
(0<<CS11) | (1<<CS10); // seteamos clk/1024
    CNT_timer1 = 0xFFFF - CLK_ms * 16;    // 16 = 1 ms, 160 = 10ms
    TCNT1 = CNT_timer1;    // empezando en el punto correpto
    TIFR1&=~(1<<TOV1);    // Seteamos para usar la interrupcion
por sobrepaso
    TIMSK1 = (1<< TOIE1 );    // Habilitamos las interrupcion por
sobrepaso
    sei();    // Habilitamos todas las interrupciones

```

```
}  
int main(void)  
{  
    int seconds = 0;  
    DDRC = 0x40;           // PortC bit 6 como salida  
    CLK_setup();  
    for (;;)              {  
        if(seconds < CLK_seconds)  
            {  
                PORTC = (PORTC & 0xBF) | ~(PORTC & _BV(6));  
                seconds = CLK_seconds;  
            }  
    }  
    return 0;  
}
```

#### 4.4.0. Control PWM de intensidad de luz

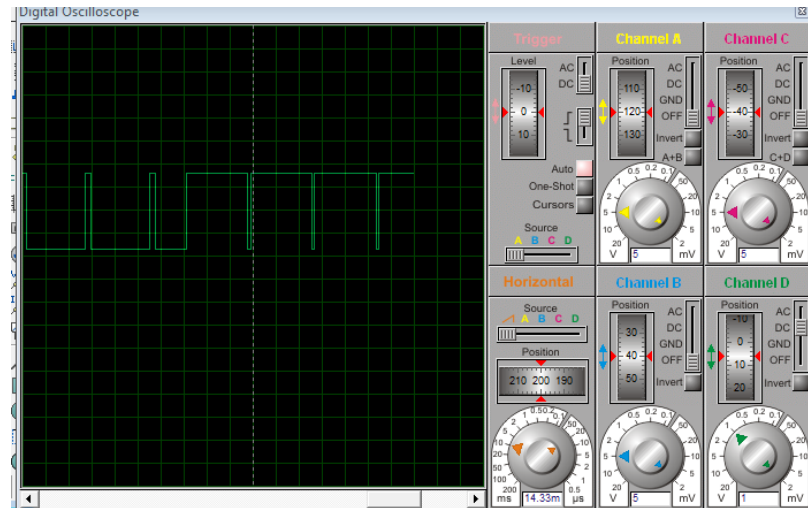
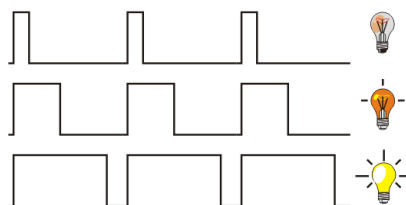


Figura. 4.1.4 Control PWM de intensidad de luz.

En este proyecto aprendemos a configurar el timer1 en modo pwm y dependiendo del registro de control OCR1A variamos el valor promedio de la salida. Un cambio de frecuencia se lo realiza por medio de la interrupción por cambio de pin y el otro cambio por la variación del pin 2 del puerto E y se muestra en la pantalla LCD el porcentaje de la salida.

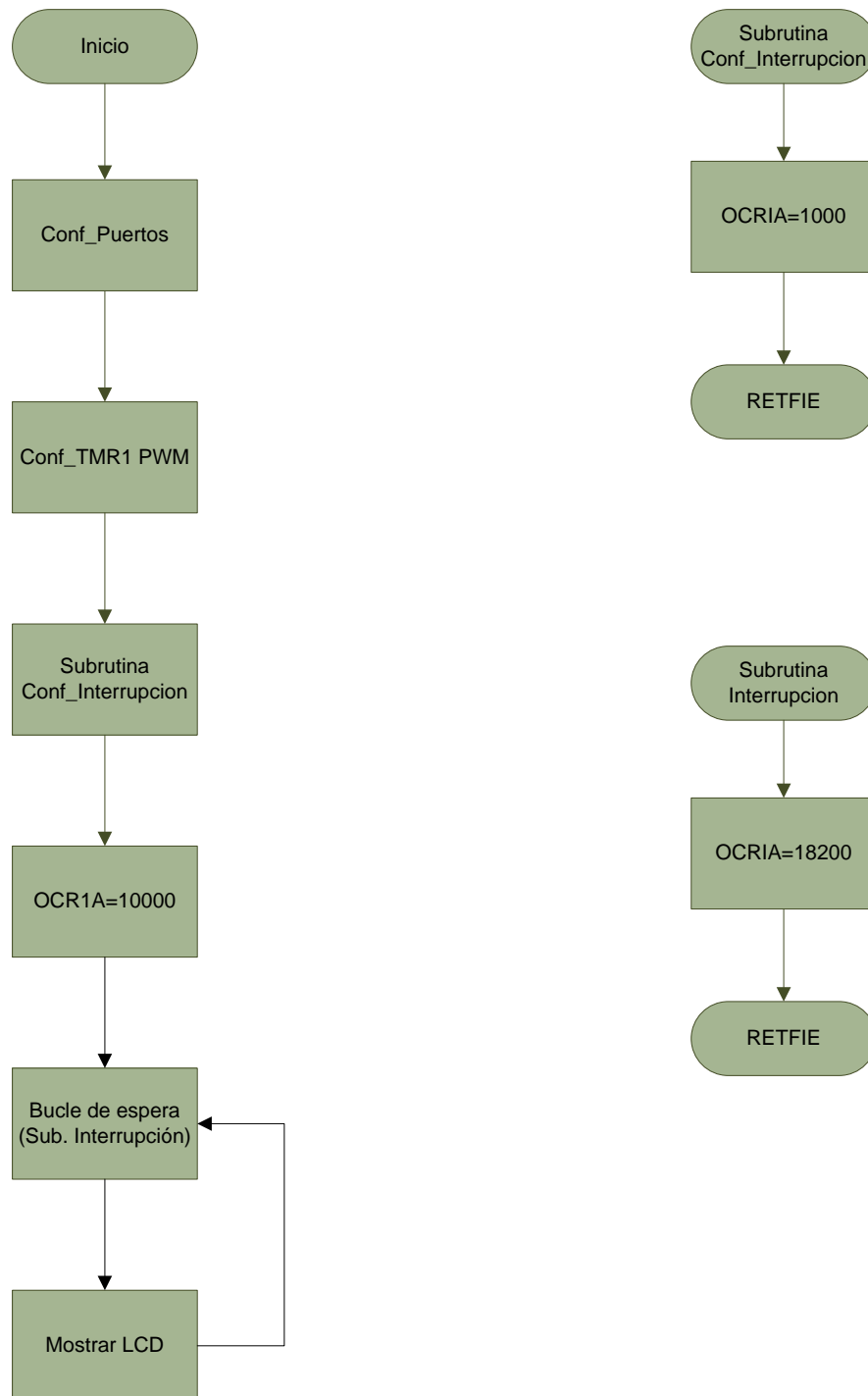
#### 4.4.1. Fundamento teórico

Las señales de frecuencia y de ciclo de trabajo variados tienen una amplia gama de aplicaciones en automatización. Un ejemplo típico es un circuito de control de potencia. Refiérase a la siguiente figura. Si un cero lógico (0) indica un interruptor abierto y un uno lógico (1) indica un interruptor cerrado, la potencia eléctrica que se transmite a los consumidores será directamente proporcional a la duración del pulso. Esta relación se le denomina Ciclo de Trabajo.





#### 4.4.2. Diagrama de flujo



### **Algoritmo**

1. Inicio.
2. Configuración de puertos.
3. Configuración del TIMER1 en modo PWM.
4. Configuramos la interrupción, en la cual al registro de comparación OCR1A se le asigna el valor de 1000, para que la modulación se produzca en un porcentaje inicial del 90%.
5. Si se habilita el pin E2 el registro de comparación OCR1A se le asigna el valor de 18200, lo que producirá un cambio en la modulación PWM.
6. Si habilitamos el pin E3 el registro de comparación OCR1A vuelve a tomar el valor de 1000, y de esa manera podemos apreciar la modulación PWM, tanto en nuestro motor o por medio de un osciloscopio.
7. Fin.

### 4.4.3. Código

```

//*****
//*****
//Ejercicio 4
//Uso del timer1 modo PWM
//Rommel Chang Suarez
//Jefferson Moreno Briones
//*****
//*****

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
void Escribir_Lcd(int j);
void LCD_INIT();
int main (void)
{

    DDRE = 0b00000000;
    PORTE = 0b00001100;
    LCD_INIT();
    DDRB = 255;                // Puerto D como salida
    TCCR1A = ((1 << COM1A1) | (1 << COM1A0)); // Set on match, Borra sobre el
TOP
    TCCR1B = ((1 << CS10) | (1 << WGM13)); // Fase + Frecuencia Correcta del
PWM, Fcpu speed
    OCR1A = 18200;             // where 0xA is the desired brightness
    ICR1 = 20000;              // Valor top
    EIMSK = 0xC1;             //[PCIE1,PCIE0,-,-,-,-,INT0]
    EIFR = 0xC0;              //[PCIF1,PCIF0,-,-,-,-,INTF0]
    PCMSK0=(1<<PCINT3|1<<PCINT2);
//[PCINT7,PCINT6,PCINT5,PCINT4,PCINT3,PCINT2,PCINT1,PCINT0]
    PCMSK1=(1<<PCINT15|1<<PCINT14|1<<PCINT12);
//[PCINT15,PCINT14,PCINT13,PCINT12,PCINT11,PCINT10,PCINT9,PCINT8]
    EICRA=0x02;                //[-,-,-,-,-,ISC01,ISC00] ; tipo de flanco
sei();                        //setea en SREG la Bandera I

    while(1)
    {
        if((PINE & 0b00000100) == 0b00000000) //si el pinE 2 es presionado
cambiamos la frecuencia pwm
        { OCR1A = 18200;
          Escribir_Lcd(0);
          _delay_ms(800);
        }
    }
}

```

```

        Escribir_Lcd(3);           //espera por interrupcion
        _delay_ms(800);
        Escribir_Lcd(4);
        _delay_ms(800);           //caso contrario OCR1A=10000
    }
    if
        ((PINE & 0b00001000) == 0b00000000) //si el pinE 2 es presionado
cambiamos la frecuencia pwm
    { OCR1A = 10000;
        Escribir_Lcd(1);
        _delay_ms(800);
        Escribir_Lcd(3);           //espera por interrupcion
        _delay_ms(800);
        Escribir_Lcd(4);
        _delay_ms(800);
        //caso contrario OCR1A=10000
    }
    if
        ((PINB & 0b00010000) == 0b00000000)
    {OCR1A = 1000;
        Escribir_Lcd(2);
        _delay_ms(800);
        Escribir_Lcd(3);           //espera por interrupcion
        _delay_ms(800);
        Escribir_Lcd(4);
        _delay_ms(800);
        //caso contrario OCR1A=10000
    }
}
}
}

```

#### 4.5.0. Control variable (CTC y PWM)

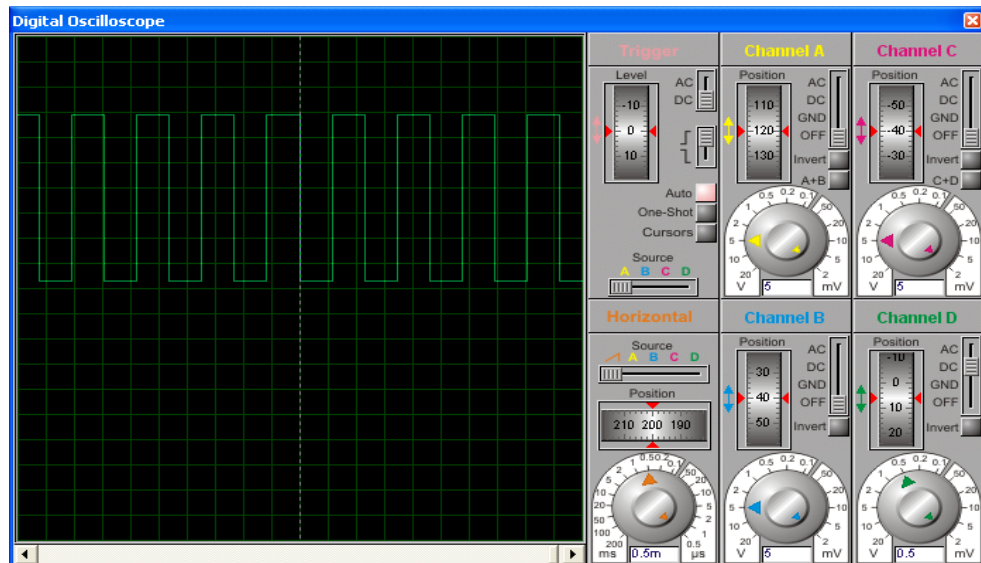


Figura. 4.1.5 Control Variable (CTC y PWM).

En el siguiente proyecto se estudiará el timer 1 con la programación en lenguaje assembler y analizaremos el modo de comparación usando el registro OCR1A en modo CTC y a la vez podemos cambiar al modo de control PWM.

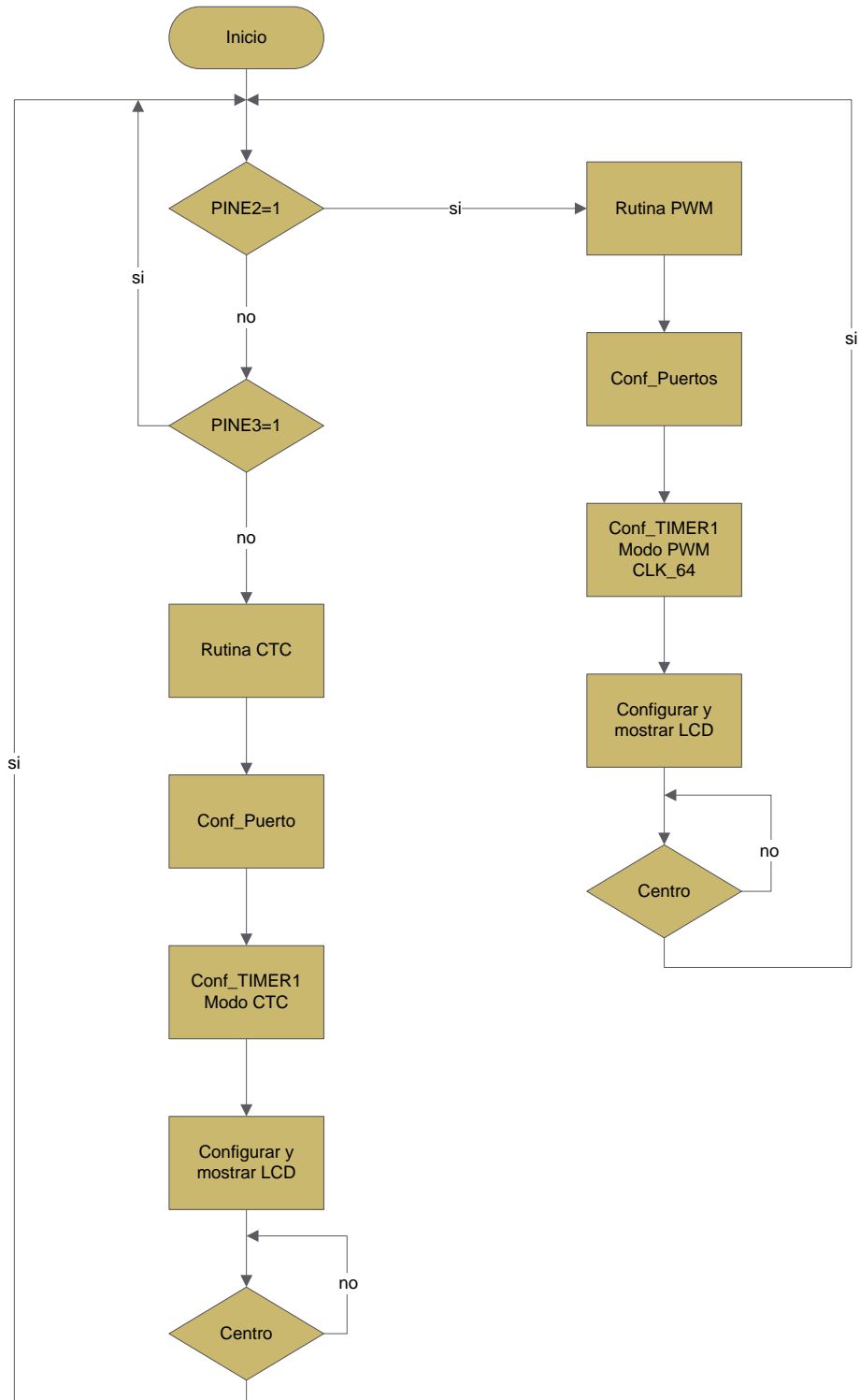
Cuando los registros del timer 1 lleguen a la comparación A entonces habilitamos la salida del motor y en el otro periodo no se habilita el motor. En el otro modo se usa el control PWM del timer1 para manejar el motor y cambiamos de modo con los botones derecha (PWM) y el botón izquierda (CTC)

#### 4.5.1. Fundamento teórico

Se realizó el siguiente proyecto para aprender a calibrar y utilizar el módulo de comparación del timer1 en el lenguaje assembler. Cuando el registro del timer1 alcanza el valor de comparación encendemos el voltaje hacia el motor. Dependiendo de los valores que se carguen en el registro de comparación se envía porcentualmente el voltaje al motor.

Se estudia en el otro modo la calibración del modo PWM en lenguaje assembler y como se calibra la pantalla en el mismo lenguaje.

#### 4.5.2. Diagrama de flujo



### **Algoritmo**

1. Inicio.
2. Si habilitamos el pin E2, se ejecutará la subrutina PWM, en la cual lo primero que haremos será configurar los puertos, para luego configurar el TIMER1 en modo PWM con CLK\_64, el siguiente paso es configurar y mostrar en la LCD.

Esta subrutina se ejecuta siempre que se mantenga habilitado el pin E2.

3. Si por otro lado habilitamos el pin E3, se ejecutará la subrutina CTC, lo que se hará en esta subrutina es configurar los puertos, y luego se configurará el TIMER1 en modo CTC, posteriormente se configura y se muestra la LCD.

Esta subrutina se ejecuta siempre que se mantenga habilitado el pin E3.

4. Fin.

### 4.5.3. Código

```

//*****
//Control variable (CTC y PWM)
//Ejercicio 5
//*****
//Rommel Chang Suarez
//Jefferson Moreno Briones
//*****
*****

.INCLUDE "m169def.inc"
.cseg
.org 0
rjmp RESET
RESET:
ldi r16, 0b00101111 ;configurar el PORT B entradas y salidas
out DDRb, r16
INICIO:
rjmp lcd
C1:
    sbic PINE,2 ;presiona el Boton derecha
    rjmp cargar
    rjmp D1

CARGAR:
    sbic PINE,2 ;suelta el Boton derecha
    rjmp CARGAR
    rjmp PWM
D1:
    sbic PINE,3 ;presiona el Boton Izquierda
    rjmp D2
    rjmp C1

D2:
    sbic PINE,3 ;suelta el Boton Izquierda
    rjmp D2
    rjmp CTC

PWM:
rjmp mostrarp
h: ldi r16,0b00010011 ; frecuencia de preescalador fcpu/64+modo ctc

```



```

sts TCCR1B,r16      ;timer1 activado
ldi r16,0b11000000 ;fcpu/64+modo ctc
sts TCCR1A,r16
ldi r16, 0x0F
sts ocr1al ,r16
ldi r16, 0x7F
sts icr1l,r16

                rjmp C1

CTC:
rjmp mostrarc
i:      ldi r16,0b00001011      ;fcpu/64+modo ctc
sts TCCR1B,r16      ;timer1 activado
ldi r16, 0b11111111      ;configura PORT D
out DDRD, r16
ldi r16, 0b11111111      ;configura PORT D
out DDRb, r16
ldi r16,0b01000000      ;fcpu/64+modo ctc
sts TCCR1A,r16
ldi r16, 0b00001000
sts ocr1al, r16
ldi r16, 0b00000000
sts ocr1ah, r16

/*      center2:
        sbic PINB,4 ;presiona el Boton (PIND6=0)
        rjmp center2

center12:
sbis PINB,4 ;suelta el Boton (PIND6=1)
rjmp center12
rjmp INICIO*/

                rjmp C1
lcd: ldi r16,0b10110111 ;Rutina para inicializar la pantalla
sts LCDCRB,r16
        ldi r16,0b00000111
sts LCDFRR,r16
ldi r16,0b00001110
sts LCDCCR,r16
ldi r16,0b11000000
sts LCDCRA,r16

```

```

    rjmp C1
mostrarp: ldi r16,0x01    ;Rutina para mostrar el dato en la pantalla
          sts LCDDR0,r16
          ldi r16,0x55
          sts LCDDR5,r16
            ldi r16,0x5E
          sts LCDDR10,r16
            ldi r16,0x30
          sts LCDDR15,r16
            ldi r16,0x08
          sts LCDDR1,r16
          ldi r16,0x07
          sts LCDDR6,r16
            ldi r16,0x05
          sts LCDDR11,r16
            ldi r16,0x00
          sts LCDDR16,r16
          tjmp h
mostrarc: ldi r16,0x11
          sts LCDDR0,r16
          ldi r16,0x84
          sts LCDDR5,r16
            ldi r16,0x04
          sts LCDDR10,r16
            ldi r16,0x21
          sts LCDDR15,r16
            ldi r16,0x01
          sts LCDDR1,r16
          ldi r16,0x04
          sts LCDDR6,r16
            ldi r16,0x04
          sts LCDDR11,r16
            ldi r16,0x01
          sts LCDDR16,r16
          tjmp i

```

```
ret
```

## **Conclusiones.**

1. Mediante el desarrollo de nuestro proyecto pudimos destacar una de las principales características del Timer1 que actúa temporizador/contador ascendente parecido al TMR0, pero con algunas peculiaridades que lo hacen muy interesante a la hora de incluir temporizaciones en nuestros programas. La primera de ellas, es que se trata de un contador de 16 bits cuyo valor se almacena en dos registros de 8 bits, en ambos registros se pueden leer y escribir su valor durante la ejecución del programa.
2. En nuestros proyectos se usó los dos modos de operación que tiene el Timer1: como temporizador y como contador. El modo de funcionamiento está determinado por el tipo de reloj seleccionado (interno -->temporizador, externo -->contador). Cuando está en modo contador su valor se incrementa en cada flanco de subida de la señal de reloj externa.
3. Cuando el Timer1 está habilitado, el valor de esos registros se incrementan desde 0000h a FFFFh y una vez que llega a su máximo valor empieza otra vez desde 0 avisándonos de ello por medio de una bandera. Si está activa la interrupción por desbordamiento del Timer 1 al desbordarse el contador, el programa entra en la función de tratamiento a la interrupción por desbordamiento del Timer1.

## **Recomendaciones.**

1. No conectar cables directamente en los espacios para conexiones externas del Kit, ya que podrían causar cortocircuito; en su lugar, colocar Headers fijos, en ocasiones hay que añadir una fuente externa de 3V para poder visualizar los datos en la LCD cuando se gasta la batería interna del KI AVR Butterfly.
2. Al momento de codificar software en lenguaje C, es recomendable segmentar el código fuente en funciones especializadas, esto quiere decir que cada función realice una sola tarea específica; de este modo se podrán utilizar las mismas funciones en otras aplicaciones.
3. Es preciso y necesario recomendar el uso del Kit AVR Butterfly, simultáneamente con la Guía de Prácticas de Laboratorio, en la cátedra de Microcontroladores.

## Bibliografía.

- [1] PARDUE, Joe, *C Programming for Microcontrollers*, tomo 1, 1<sup>ra</sup> Edición, Editorial Smiley Micros, Knoxville-Tennessee Octubre del 2005.
  
- [2] MANN, Richard, How to Program an 8-bit Microcontroller Using C Language, disponible en: [www.atmel.com](http://www.atmel.com), 23 de junio del 2004.
  
- [3] RODLAND, Arild, Novice's Guide to AVR Development, disponible en: [www.atmel.com](http://www.atmel.com), 2 de febrero del 2004.
  
- [4] AVRProg User Guide, disponible en: [www.atmel.com](http://www.atmel.com). Fecha de consulta 11/03/2011
  
- [5] 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash ATmega169V Atmega169 Rev A to E, disponible en: [www.atmel.com](http://www.atmel.com), Fecha de consulta 10/23/2011.
  
- [6] Introduction to the Atmel AVR Butterfly, disponible en: [www.atmel.com](http://www.atmel.com), Fecha de consulta 11/11/201.
  
- [7] AVR Butterfly Evaluation Kit User Guide, disponible en: [www.atmel.com/products/AVR/butterfly](http://www.atmel.com/products/AVR/butterfly), Fecha de consulta 09/02/201.