ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"UTILIZACION DE MICROCONTROLADORES ATMEL PARA LA REALIZACION DE HARDWARE DEMOSTRATIVO DE DIVERSOS TIPOS DE INTERRUPCIONES INTERNAS Y EXTERNAS DE LOS MICROCONTROLADORES ATMEL"

TESINA DE SEMINARIO

Previo la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentada por:

Giovanni Bismark Granados Ortiz

Jorge Antonio Vega Rosero

GUAYAQUIL – ECUADOR

2012

AGRADECIMIENTO

En primer lugar a Dios, ya que sin la ayuda de Él, nada de lo que hemos logrado hubiera sido posible.

A la ESPOL por habernos permitido ser parte de ella y de esa manera recibir nuestra formación académica.

Al Ing.Carlos Valdivieso, por su ayuda indispensable y desinteresada durante el desarrollo de este trabajo de investigación.

Jorge Antonio Vega Rosero

DEDICATORIA

El trabajo y lo que representa la elaboración de este documento no sería posible sin el soporte y respaldo de mis Padres, sin el apoyo incondicional de mi novia y amigos y sin la oportunidad que me dio la Escuela. Mi dedicatoria va para todos ellos.

Jorge Antonio Vega Rosero

AGRADECIMIENTO

A todas las personas que desde el inicio hasta el presente han contribuido con mi desarrollo.

En especial a mi Familia.

Giovanni Bismark Granados Ortiz

DEDICATORIA

El presente Trabajo se lo Dedico a mis Padres por su apoyo en todo momento.

Giovanni Bismark Granados Ortiz

TRIBUNAL DE SUSTENTACION

_____

Ing. Carlos Valdivieso A.

Profesor del Seminario de Graduación

_____

Ing.Hugo Villavicencio V.

Delegado del Decano

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina, nos corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

(Reglamento de Graduación de la ESPOL)

Jorge Vega Rosero. **Giovanni Granados Ortiz**

RESUMEN

El objetivo de este proyecto es el de entregar una guía práctica y material indispensable para todo estudiante que desee conocer sobre la Tecnología de los Microcontroladores Atmel.

Este proyecto se centra en la Utilización de las Interrupciones del Microcontrolador ATMEGA169 y su flexibilidad para ser implementadas en kit de evaluación AVR BUTTERFLY.

Aquí se tratan algunas de las interrupciones importantes y su implementación en 5 problemas resueltos que serán de guía para futuros proyectos y mejores desarrollos.

Los problemas implementados aquí se realizaron en el Programa AVR 4, los primeros dos están en Lenguaje ASEMBLER y los tres siguientes en lenguaje C. Su demostración y simulación están para ser probados en el Programa PROTEUS, y su implementación física se presenta en un Protoboard para hacerlo muy didáctico.

INDICE GENERAL

Contenido **Pag**

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION.

El Objetivo de este trabajo es desarrollar proyectos didácticos con metodología gradual de dificultad para el aprendizaje de los microcontroladores Atmel, se ha concentrado el trabajo en particular con el microcontrolador Atmega169, siendo un peldaño para los futuros compañeros de ingeniería electrónica, ya que existe muy poca bibliografía en lengua hispana sobre este tipo de microcontroladores, colaborando con nuestro aporte, esperemos que sea un complemento guía para que sigan en desarrollo de esta tecnología.

En este proyecto se concentra en el tema de las interrupciones, desarrollando cinco aplicaciones, dos de los cuales son desarrollados en el lenguaje asembler y tres restantes en lenguaje C.

Este trabajo se divide en una gama de cinco capítulos los cuales serán descritos en base a la teoría y aplicación en el tema de las interrupciones, considerando las especificaciones del microcontrolador.

CAPITULO 1

1.- INTERRUPCIONES, COMPARACION DEL AVR CON OTROS MICROCONTROLADORES

1.1. INTERRUPCIONES.

Las Interrupciones son un método del que disponen los Microcontroladores, para atender alguna circunstancia que requiera su inmediata atención. Al presentarse un pedido de interrupción el Microcontrolador da por terminado cualquier instrucción en curso, y hará un salto a una subrutina de Interrupción, una vez terminada esta subrutina, volverá a su labor anterior.

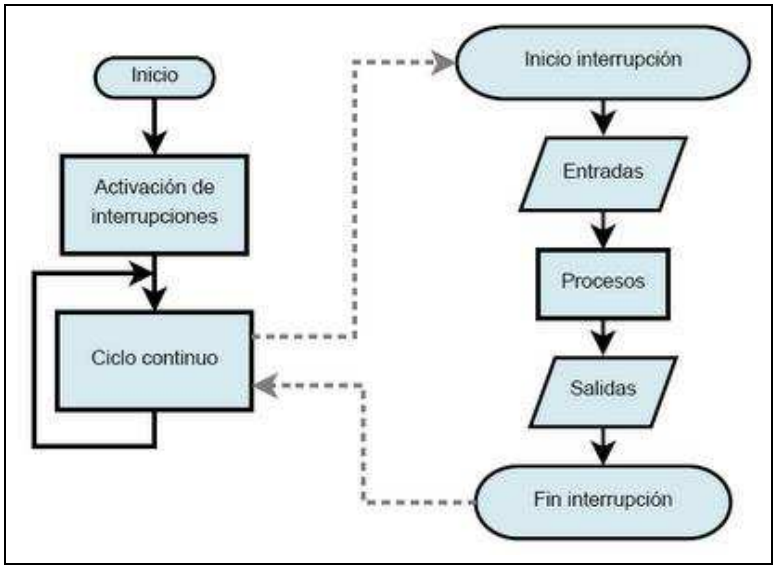

Figura 1-1: Diagrama de bloque básico de un programa con subrutinas de Interrupción

Si no existieran las Interrupciones, el Microcontrolador tendría que continuamente comprobando el estado de los dispositivos Externos e Internos.

Una vez terminado el tratamiento de la interrupción, es importante que el Microcontrolador siga con lo que estaba haciendo. Por eso se debe antes de tratar la Interrupción se guarde de alguna forma el estado del Micro y al terminar la rutina del Micro se restaure su estado antes del ingreso a la Subrutina.

En especial antes de ingresar a una interrupción se debe guardar el estado del REGISTRO STATUS, una buena manera es hacerlo en la PILA o STACK, aplicando sentencias PUSH y POP.

Una pila es un bloque consecutivo de datos de la memoria asignada por el programador. Este bloque de memoria se puede utilizar tanto por el control de microcontrolador interno, así como el programador para almacenar datos temporalmente. La pila funciona de modo (LIFO), es decir, el último dato que ingresa es el primero que debe ser recuperada.

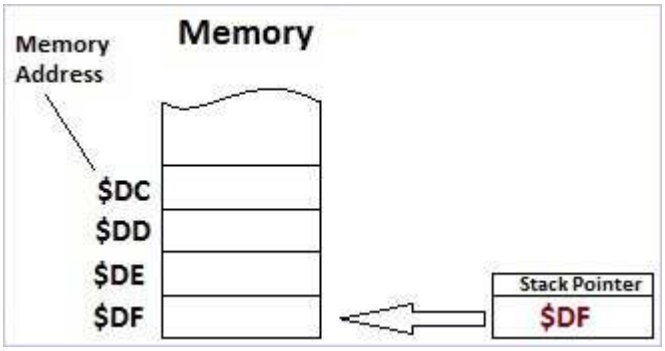

Figura 1-2a: Inicial Stack.

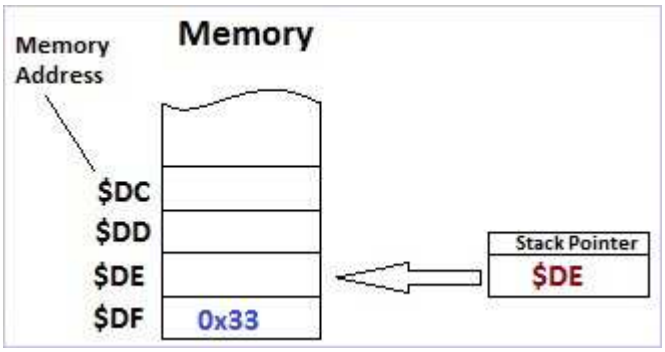

Figura 1-2b: Stack después del Primer PUSH

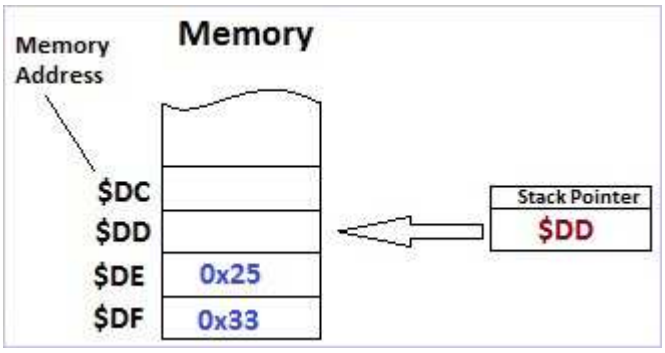

Figura 1.-c: Stack después del segundo PUSH

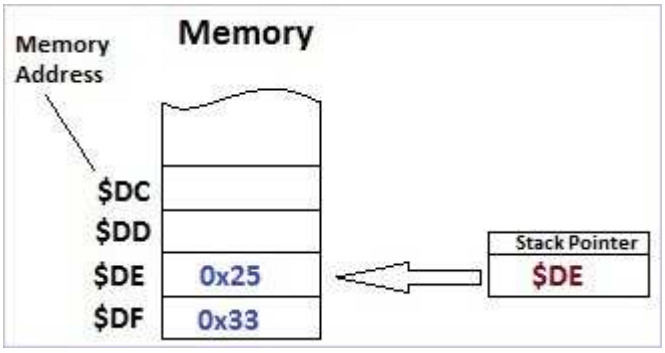

Figura 1-2d: Stack después de instrucción POP

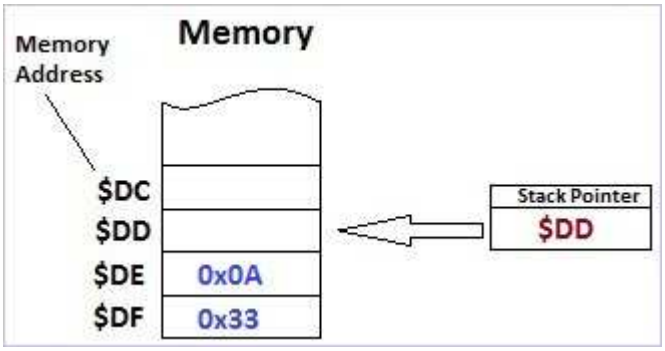

Figura 1-2e: Stack después de tercer PUSH

Se pueden distinguir dos tipos de Interrupciones, Interrupciones por Software e Interrupciones por Hardware.

En el AVR tenemos diferentes fuentes de interrupciones, cada una teniendo su espacio de memoria de programa para el vector de interrupción, en esta dirección se indicará cual es la dirección de salto donde se encuentra el código que se debe ejecutar para esa interrupción. El contador de programa se carga con la dirección del vector de interrupción una vez que ésta sucede y ahí tendremos una instrucción de salto a la dirección de memoria donde está el código que atiende a la interrupción.

En la parte más baja de la memoria de programa están estos vectores de interrupción, si dos interrupciones ocurren al mismo tiempo la interrupción con una posición de memoria más baja se ejecuta primero.

RESET Y VECTORES DE INTERRUPCION ATMEGA169			
NUMERO DE VECTOR	**DIRECCION DE PROGRAMA**	**FUENTE**	**DEFINICION DE INTERRUPCION**
1	0X0000	RESET	PIN EXTERNO RESET
2	0X0002	INT0	INTERRUPCION EXTERNA INT0
3	0X0004	PCINT0	INTERRUPCION EXTERNA POR CAMBIO EN PIN
4	0X0006	PCINT1	INTERRUPCION EXTERNA POR CAMBIO EN PIN
5	0X0008	TIMER2 COMP	TIMER/COUNTER2 COMPARACION CON VALOR DEFINIDO
6	0X000A	TIMER2 OVF	TIMER/COUNTER2 POR DESBORDE
7	0X000C	TIMER1 COMP	TIMER/COUNTER1 COMPARACION CON VALOR DEFINIDO
8	0X000E	TIMER1 COMPA	TIMER/COUNTER1 COMPARACION CON VALOR DEFINIDO EN A
9	0X0010	TIMER1 COMPB	TIMER/COUNTER1 COMPARACION CON VALOR DEFINIDO EN B
10	0X0012	TIMER1 OVF	TIMER/COUNTER1 POR DESBORDE
11	0X0014	TIMER0 COMP	TIMER/COUNTER0 COMPARACION CON VALOR DEFINIDO
12	0X0016	TIMER0 OVF	TIMER/COUNTER0 POR DESBORDE
13	0X0018	SPI, STC	SPI TRANSMISION SERIAL COMPLETA
14	0X001A	USART,RX	USART, RX COMPLETA
15	0X001C	USART, UDRE	USART, REGISTRO DE DATO VACIA
16	0X001E	USART, TX	USART, TX COMPLETA
17	0X0020	USI START	USI CONDICION DE INICIO
18	0X0022	USI OVERFLOW	USI DESBORDE
19	0X0024	ANALOG COMP	COMPARADOR ANALOGICO
20	0X0026	ADC	ADC, CONVERSION COMPLETA
21	0X0028	EE READY	EEPROM LISTA
22	0X002A	SPM READY	ALMACENAMIENTO EN MEMORIA DE PORGRAMA LISTA
23	0X002C	LCD	LCD INICIO DE FRAME.

Tabla 1-1: distribución de los Vectores de Interrupción del Atmega169.

En nuestro estudio nos basaremos en las configuraciones del Atmega169, ya que es el Microcontrolador integrado en el AVRBUTTERFLY.

VECTOR PARA LENJUAGE C	NUEMERO DE VECTOR
SIG_INTERRUPT0	_VECTOR(1)
SIG_INTERRUPT1	_VECTOR(2)
SIG_INTERRUPT2	_VECTOR(3)
SIG_INTERRUPT3	_VECTOR(4)
SIG_OUTPUT_COMPARE2	_VECTOR(5)
SIG_OVERFLOW2	_VECTOR(6)
SIG_INPUT_CAPTURE1	_VECTOR(7)
SIG_OUTPUT_COMPARE1A	_VECTOR(8)
SIG_OUTPUT_COMPARE1B	_VECTOR(9)
SIG_OVERFLOW1	_VECTOR(10)
SIG_OUTPUT_COMPARE0	_VECTOR(11)
SIG_OVERFLOW0	_VECTOR(12)
SIG_SPI	_VECTOR(13)
SIG_UART0_RECV	_VECTOR(14)
SIG_UART0_DATA	_VECTOR(15)
SIG_UART0_TRANS	_VECTOR(16)
SIG_USI_START	_VECTOR(17)
SIG_USI_OVERFLOW	_VECTOR(38)
SIG_COMPERATOR	_VECTOR(19)
SIG_ADC	_VECTOR(20)
SIG_EEPROM_READY	_VECTOR(21)
SIG_SPM_READY	_VECTOR(22)
SIG_LCD	_VECTOR(23)

Tabla 1-2: Vectores de Interrupción para ser utilizados en Lenguaje C.

En la Tabla 1-1 se muestran los Vectores de Interrupción del Atmega169, estos sirven para Programar en Asembler. Sin embargo en Lenguaje C, cambia la forma de Aplicarlos. Se utiliza la Librería IOM169.H en esta librería está definido los Vectores.

En C no se necesita declarar en la Cabecera del Programa los Vectores, solo Basta con Incluir la Función:

```
SINGNAL(VECTOR){
        SENTENCIAS…..
}
```

1.2. EL MERCADO DE LOS MICROCONTROLADORES.

Existe una gran diversidad de Microcontroladores. Quizá la clasificación más importante sea entre Microcontroladores de 4, 8, 16 ó 32 bits. Aunque las prestaciones de los Microcontroladores de 16 y 32 bits son superiores a los de 4 y 8 bits, la realidad es que los Microcontroladores de 8 bits dominan el mercado y los de 4 bits se resisten a desaparecer. La razón de esta tendencia es que los Microcontroladores de 4 y 8 bits son apropiados para la gran mayoría de las aplicaciones, lo que hace absurdo emplear micros más potentes y consecuentemente más caros.

Uno de los sectores que más tira del mercado del Microcontrolador es el mercado automovilístico. De hecho, algunas de las familias de Microcontroladores actuales se desarrollaron pensando en este sector, siendo modificadas posteriormente para adaptarse a sistemas más genéricos. El mercado del automóvil es además uno de los más exigentes: los componentes electrónicos deben operar bajo condiciones extremas de vibraciones, choques, ruido, etc. Y seguir siendo fiables.

En cuanto a las técnicas de fabricación, cabe decir que prácticamente la totalidad de los Microcontroladores actuales se fabrican con tecnología CMOS 4 (Complementary Metal Oxide Semiconductor). Esta tecnología supera a las técnicas anteriores por su bajo consumo y alta inmunidad al ruido.

1.3. SELECCIÓN DE MICROCONTROLADOR.

A la hora de escoger el Microcontrolador a emplear hay que tener en cuenta multitud de factores, como la documentación y herramientas de desarrollo disponibles y su precio, la cantidad de fabricantes que lo producen y por supuesto las características del Microcontrolador (tipo de memoria de programa, número de temporizadores, interrupciones, etc.):

Costes. Para el fabricante que usa el Microcontrolador en su producto una diferencia de precio en el Microcontrolador de algunos céntimos es importante (el consumidor deberá pagar además el coste del empaquetado, el de los otros componentes, el diseño del hardware y el desarrollo del software). Si el fabricante desea reducir costes debe tener en cuenta las herramientas de apoyo con que va a

contar: emuladores, simuladores, ensambladores, compiladores, etc. Es habitual que muchos de ellos siempre se decanten por Microcontroladores pertenecientes a una única familia.

Aplicación. Antes de seleccionar un Microcontrolador es imprescindible analizar los requisitos de la aplicación:

• ***Procesamiento de datos:*** puede ser necesario que el Microcontrolador realice cálculos críticos en un tiempo limitado. En ese caso debemos asegurarnos de seleccionar un dispositivo suficientemente rápido para ello. Por otro lado, habrá que tener en cuenta la precisión de los datos a manejar: si no es suficiente con un Microcontrolador de 8 bits, puede ser necesario acudir a Microcontroladores de 16 ó 32 bits, o incluso a hardware de coma flotante.

• ***Entrada Salida:*** para determinar las necesidades de Entrada/Salida del sistema es conveniente conocer el diagrama de bloques del mismo, de tal forma que sea sencillo identificar la cantidad y tipo de señales a controlar. Una vez realizado este análisis puede ser necesario añadir periféricos externos o cambiar a otro Microcontrolador más adecuado a ese sistema.

• ***Consumo:*** algunos productos que incorporan Microcontroladores están alimentados con baterías. Lo más conveniente en un caso como éste puede ser que el Microcontrolador esté en estado de bajo consumo pero que despierte ante la activación de una señal (una interrupción) y ejecute el programa adecuado para procesarla.

• ***Memoria:*** El tipo de memoria a emplear vendrá determinado por el volumen de ventas previsto del producto: de menor a mayor volumen será conveniente emplear EPROM, OTP y ROM.

En cuanto a la cantidad de memoria necesaria deberemos hacer una estimación de cuánta memoria volátil y no volátil es necesaria y si es conveniente disponer de memoria no volátil modificable.

• ***Ancho de palabra:*** el criterio de diseño debe ser seleccionar el Microcontrolador de menor ancho de palabra que satisface los requerimientos de la aplicación. Usar un Microcontrolador de 4 bits supondrá una reducción en los costes importante, mientras que uno de 8 bits puede ser el más adecuado si el ancho de los datos es de un byte. Los Microcontroladores de 16 y 32 bits, debido a su elevado coste, deben reservarse

para aplicaciones que requieran altas prestaciones (Entrada/Salida potente o espacio de direccionamiento muy elevado).

• ***Diseño de la placa:*** la selección de un Microcontrolador concreto condicionará el diseño de la placa de circuitos.

1.4. INTERRUPCIONES EN 8051/8052

El Intel 8051 es un Microcontrolador (µC) desarrollado por Intel en 1980 para uso en productos embebidos. Es un Microcontrolador muy popular.

Los núcleos 8051 se usan en más de 100 Microcontroladores de más de 20 fabricantes independientes como Atmel, Dallas Semiconductor, Philips, Winbond, entre otros.

La denominación oficial de Intel para familia de µCs 8051 es MCS 51.

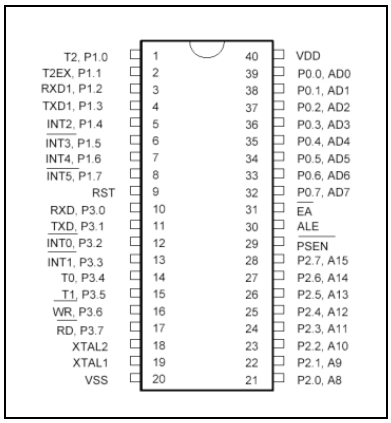

Figura 1-3: Pines del Microcontrolador Intel 8052

El Microcontrolador 8052 tiene seis interrupciones, mientras que el 8051 tienen cinco. La Figura 1-4 muestra los distintos tipos de Interrupciones, señalando los indicadores que activan TF2 y EXF2 en el 8051 por no tener implementado el Timer2.

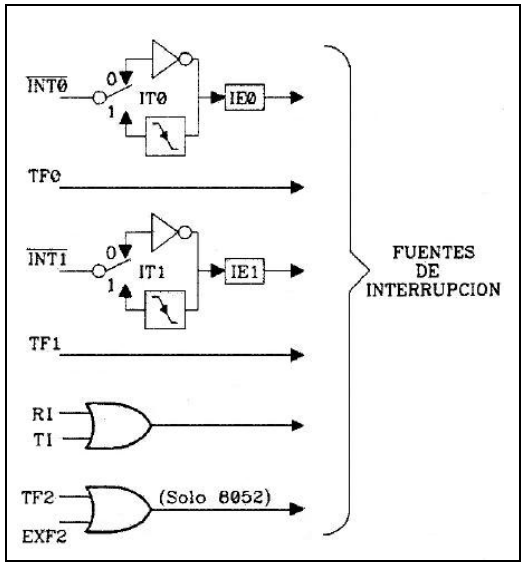

Figura 1-4: Interrupciones del Intel 8052

Los bits de bandera que generan las interrupciones pueden ser cancelados. En algunas interrupciones por hardware cuando estas son vectorizadas. No obstante todos los bits pueden cancelarse por software escribiendo ceros en el registro correspondiente.

Cada una de estas fuentes de interrupción pueden ser individualmente habilitadas o inhabilitadas poniendo a <<uno>> o a <<cero>> el bit correspondiente del registro IE (interrup Enable Register) perteneciente a SFR (Special Funcion Register).

RESET Y VECTORES DE INTERRUPCION INTEL 8052			
NUMERO DE VECTOR	**DIRECCION DE PROGRAMA**	**FUENTE**	**DEFINICION DE INTERRUPCION**
1	0X0003	INT0	EXTERNA INT0
2	0X000B	TIMERO	TIMER0
3	0X0013	INT1	EXTERNA INT1
4	0X001B	TIMER1	TIMER1
5	0X0023	RI + TI	PUERTO SERIE
6	0X002B	TIMER2 + T2EX	T2EX

Tabla 1-3: distribución de los Vectores de Interrupción del Intel 8052.

1.5. INTERRUPCIONES EN PIC 16F887

El 16f887 es un Microcontrolador de la familia 16 a 8bit, que cuenta con unas características especiales como son:

-oscilador interno configurable desde 32khz hasta 8mhz con frecuencias distintas seleccionables por software dentro de este rango incluidas las mencionadas.

-2 comparadores (amplificador operacional) independientes del resto del integrado, bueno con la particularidad de que C2 comparador 2 puede ser trabajado con TMR1.
-Una de las ventajas es que el voltaje de referencia para estos comparadores lo proporciona el circuito integrado internamente o también puede ser externo. Con la particularidad de que si se apaga el Vref para el comparador este no consume

corriente pero sin embargo proporciona un detector de cruce por cero muy útil para controlar procesos liados a la corriente AC.

-El CAD es de 10bit y tiene una frecuencia de trabajo independiente o un preescaler por llamarlo así, para poder tener una mejor conversión y tiempos requeridos mínimos.

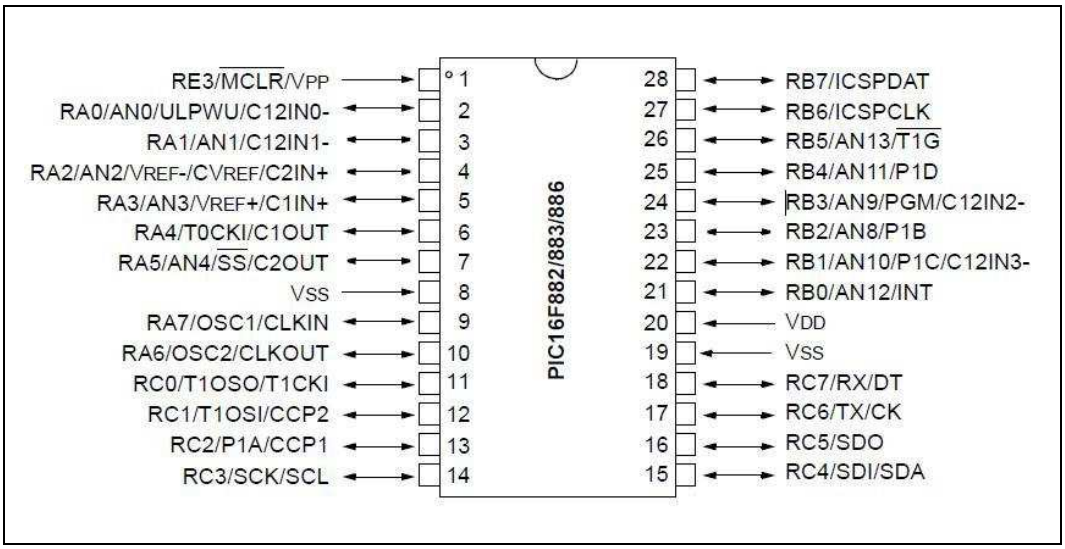

Figura 1-5: Distribución de pines de Microcontrolador 16f887

NUMERO DE VECTOR	DEFINICION DE INTERRUPCION
1	• External Interrupt RB0/INT
2	• Timer0 Overflow Interrupt
3	• PORTB Change Interrupts
4	• 2 Comparator Interrupts
5	• A/D Interrupt
6	• Timer1 Overflow Interrupt
7	• Timer2 Match Interrupt
8	• EEPROM Data Write Interrupt
9	• Fail-Safe Clock Monitor Interrupt
10	• Enhanced CCP Interrupt
11	• EUSART Receive and Transmit Interrupts
12	• Ultra Low-Power Wake-up Interrupt
13	• MSSP Interrupt

Tabla 1-4: Distribución de los Vectores de Interrupción del PIC16F887.

1.6. INTERRUPCIONES EN ATMEGA169, Y SU APLICACIÓN EN EL AVR BUTTERFLY

1.6.1 MICROCONTROLADOR ATMEGA169

El AVRBUTTERFLY tiene integrado el Microcontrolador ATEMEGA169, la tabla 1-1 muestra la distribución de todos los vectores de Interrupción Disponibles para el ATMEGA169.

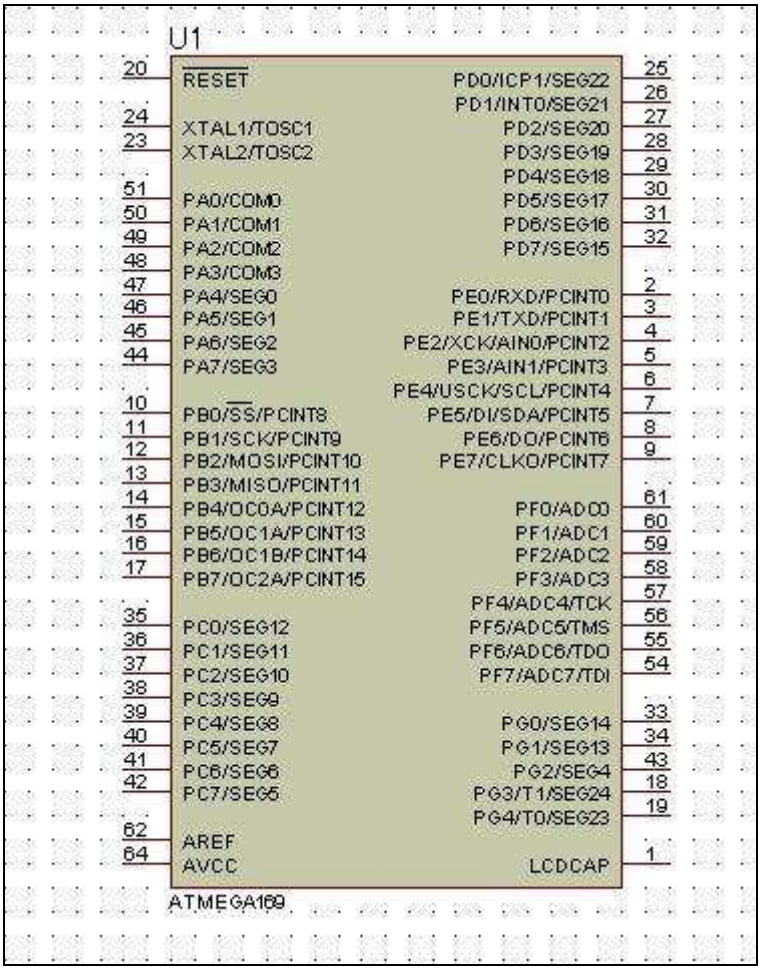

Figura 1-6: Distribución de pines de Microcontrolador ATMEGA169

Debido a la circuitería interna del Butterfly no todos los pines son accesibles, es mas no todas las interrupciones de la tabla 1-1 pueden ser implementadas en el AVRBUTTERFLY, un ejemplo claro es el de la INT0, que está en el Puerto D, y todo el Puerto D, es utilizado para controlar al LCD.

1.6.2 HABILITACION DE INTERRUPCION GLOBAL

Para que cualquier interrupción sea habilitada se deberá colocar un 1 lógico en El bit 7 del registro SREG, este bit se denomina (I) Interrupción Global, sin este bit no se habilita, el Microcontrolador no dará paso a ninguna Interrupción ni externa ni interna.

Bits	7	6	5	4	3	2	1	0
SREG	I	T	H	S	V	N	Z	C

Figura 1-7: distribución de bits del Registro STATUS

1.6.3 PEDIDO DE INTERRUPCION EXTERNA INT0

1.6.3.1 INTERRUPCION INT0

El pin #6 del Microcontrolador Attiny2313, es el pin dispuesto por Fábrica para Utilizarlo con la Interrupción Externa INT0. El Butterfly tiene integrado como Microcontrolador el Atmel169, pero el Butterfly no tiene a disponibilidad la utilización del la Interrupción INT0, debido a que utiliza este pin para el LCD interno del Butterfly, así que se utilizara para describir el uso de la Interrupción Externa INT0 el Microcontrolador Attiny2313. El diagrama de la figura 1-8. ES un circuito básico para mostrar el funcionamiento de la interrupción INT0.

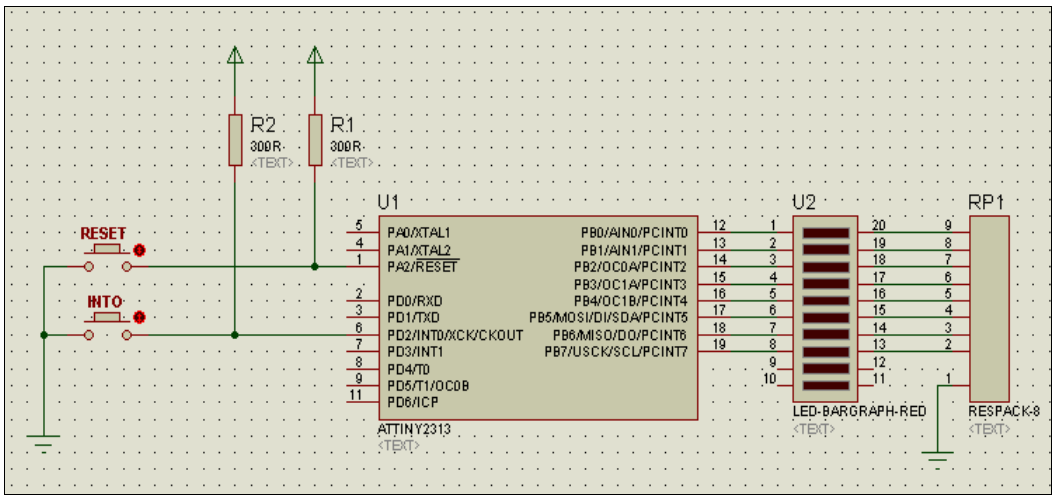

Figura 1-8: Diagrama propuesto para prueba de Interrupción Externa INT0

1.6.3.1 Registros involucrados: SREG, GIMSK, MCUCR, GIFR

Para habilitar la Interrupción INT0 se debe poner a 1 el bit GIMSK.INT0 (BIT #6), con esto y junto a la activación de las Interrupciones Globales, SREG.I (BIT #7) tenemos habilitada la interrupción. Con esto ya se puede trabajar con la Interrupción INT0.

Bits	7	6	5	4	3	2	1	0
GIMSK	INT1	INT0	PCIE	-	-	-	-	-

Figura 1-9: distribución de bits del Registro GIMSK

El Registro GIMSK corresponde al Attyni2313, en el ATMEL169, este registro cambia de nombre a **EIMSK.**

Hay que tener en cuenta que el PIN INTO del Microcontrolador Attiny2313, es una entrada digital, por lo tanto pueden darse ciertos escenarios en el cambio de estado, estos escenarios de cambio son:

- Cambio de estado bajo al estado alto.
- Cambio de estado alto al estado bajo.
- Cambio de cualquier estado.

Para que la interrupción se dispare por cualquiera de estos cambios definidos, hay que configurar el Registro MCUCR, por defecto al estar el Registro MCUCR lleno de 0 lógico, estamos en la opción “a” de la tabla 1-4.

Bits	7	6	5	4	3	2	1	0
MCUCR	PUD	SM1	SE	SM0	ISC1	ISC1	ISC0	ISC0

Figura 1-10. Distribución de bits del Registro de Control MCU

Los bits asociados a la Interrupción INT0 son el MCUCR.ISC01 (Bit #1) y MCUCR.ISC00 (Bit #0), la tabla 1-4 muestra las diferentes combinaciones con sus respectivos detalles que disparan la interrupción INT0.

CONFIGURACION DE BITS DEL REGISTRO MCUCR PARA EL USO DE LA INTERRUPCION INT0			
Opción	**ISC001**	**ISC00**	**EVENTOS PARA DISPARAR LA INTERRUPCION INT0**
a	0	0	El Nivel Lógico 0 en la entrada INT0 genera un pedido de Interrupción
b	0	1	Cualquier cambio en la entrada INT0 genera un pedido de Interrupción
c	1	0	La transición de bajada en la entrada INT0 genera un pedido de Interrupción.
d	1	1	La transición de subida en la entrada INT0 genera un pedido de Interrupción.

Tabla 1-5: Configuración de bits del registro MCUCR

Como observación, al solo modificar el Registro **GIMSK** por defecto los valores de ISC01 Y ISC00 son cero. Por lo que la Interrupción INT0 se activaría con el cambio de estado lógico alto a estado lógico bajo.

1.6.3.2 EFECTOS SOBRE OTROS REGISTROS:

Cuando ocurre un cambio lógico en el pin INT0, la bandera INTF0 (bits #6 del Registro EIFR) se pondrá en estado lógico alto, siempre y cuando se hayan hecho las habilitaciones para la Interrupción Externa INT0; estas son poner en estado alto al bit GIMSK.INT0 (BIT #6), poner en estado alto al bit SREG.I (BIT #7) y además de seleccionar el tipo de evento en el cambio de estado lógico del Pin INT0, el tipo de eventos están descritos en la tabla 1-4; el MCU ejecutara un salto al respectivo vector de Interrupción correspondiente a la Interrupción INT0, este Vector es el 0X0001 en el Microcontrolador Attyni2313.

Bits	7	6	5	4	3	2	1	0
EIFR	INTF	INTF	PCIF	-	-	-	-	-

Figura 1-11. Distribución de bits del Registro EIFR,

(External Interrupt Flag Register)

1.6.3.3 PASOS PARA CONFIGURACION DE INTERRUPCION INT0

Vector de interrupción, corresponde al 0x0001

ASEMBLER:

```
; -----------------------------------------------------
; VECTORES DE INTERRUPCION
; -----------------------------------------------------
.ORG 0x0000 ; INICIO DEL PROGRAMA
RJMP RESET
.ORG 0x0001 ; VECTOR INTERRUPCION INT0
RJMP INT_INT0
;-------------------------------------------------------------
```

Habilitar los bits en los registros correspondientes.

ASEMBLER:

```
;-----------------------------------------------------------------------
; HABILITACION DE REGISTROS PARA UTILIZAR LA INTERRUPCION INT0
;-----------------------------------------------------------------------
HABILITAR_INT0:
  IN    REG_TEMP,GIMSK          ; GUARDAMOS EL ESTADO DEL REGISTRO GIMSK,
        SBR REG_TEMP,(1<<INT0)  ; COLOCAMOS UN ALTO EN EL BIT6, SIN ALTERAR EL
                                ; RESTO DEL CONTENIDO DEL REGISTRO.
        OUT GIMSK,REG_TEMP      ; ACTUALIZAMOS EL REGSITRO GIMSK CON EL NUEVO
                                ; VALOR
        SEI                     ; HABILITACION DE INTERRUPCION GLOBAL
  RET
;-----------------------------------------------------------------------
```

1.6.4 PEDIDO DE INTERRUPCION POR CAMBIO EN LOS TERMINALES

1.6.4.1 INTERRUPCIONES PCINT0 Y PCINT1

El ATMEGA169 Tiene 16 interrupciones por cambio en los Terminales, 8 están ubicadas en el puerto D y otras 8 en el puerto B, las del puerto D tienen su propio vector de interrupción este es el 0x0004 y las del puerto B tienen el vector 0x0006, la tabla 1-5 muestra esta distribución.

DISTRIBUCION DE PINES PARA UTILIZACION DE INTERRUPCIONES POR CAMBIO DE PORTICO EN EL BUTTERFLY				
PUERTO	**VECTOR ASOCIADO**	**INTERRUPCION**	**PINes ATMEL169**	**CONFIGURACION DE JOYST**
PUERTO E	0X0004	PCINT0	#2	LIBRE
		PCINT1	#3	LIBRE
		PCINT2	#4	***BOTON IZQUIERDA***
		PCINT3	#5	***BOTON DERECHA***
		PCINT4	#6	LIBRE
		PCINT5	#7	LIBRE
		PCINT6	#8	LIBRE
		PCINT7	#9	LIBRE
PUERTO B	0X0006	PCINT8	#10	LIBRE
		PCINT9	#11	LIBRE
		PCINT10	#12	LIBRE
		PCINT11	#13	LIBRE
		PCINT12	#14	***BOTON CENTRAL***
		PCINT13	#15	RESERVADO PARLANTE
		PCINT14	#16	***BOTON ARRIBA***
		PCINT15	#17	***BOTON ABAJO***

Tabla 1-6: Distribución de pines en el dispositivo Butterfly

Este tipo de interrupción se dispara al haber un cambio en cualquiera de los pines habilitados para una Interrupción por cambio en los pórticos. En el atmel169 se pueden activar hasta 16 pines, sin embargo en el Butterfly, no todos estos pines están disponibles,

Las interrupciones por cambio en los pórticos se configuran pin a pin, es decir individualmente. De tal manera que si queremos que solo haya una interrupción de este tipo en un solo pin del puerto E, hay que configurar solo el pin d ese puerto. Agrupando el puerto E y el puerto B por separado, por lo que realmente tenemos dos tipos de interrupción con las mismas características.

En el puerto E se configuran hasta 8 de ellas, que van desde PCINT0 hasta PCINT7. Para estos pines se utilizara el vector de Interrupción 0x0004. Es decir que al configurar Interrupción por cambio en los pórticos en el puerto E, inmediatamente nos direccionaremos al vector de Interrupción 0x0004

En el puerto B se configuran hasta 8 de ellas, que van desde PCINT8 hasta PCINT15. Para estos pines se utilizara el vector de Interrupción 0x0006. Es decir que al configurar Interrupción por cambio en los pórticos en el puerto B, inmediatamente nos direccionaremos al vector de Interrupción 0x0006.

Nuestro estudio se centra en el Butterfly, por lo que se observa que de las 16 fuentes de interrupción por cambio en los pórticos, tan solo tenernos disponibles 10 para nuestro libre uso. Puesto que 5 son ocupadas para el joystin integrado. Y el PIN #15 es utilizado por el Butterfly como salida para la conexión del parlante.

Estas interrupciones funcionan en forma similar a las INT0, difieren en que no tenemos los controles para que tipo de transición, tan solo se activara esta interrupción cuando haya cambios lógicos en las entradas digitales de los pines previamente configurados. y para su funcionamiento debemos del mismo modo configurar ciertos registros para su habilitación.

1.6.4.2 Registros involucrados: SREG, EIMSK

Para activar estas interrupciones debemos habilitar 3 condiciones sucesivas, si una de ellas no es activada no se habrá habilitado la interrupción

CONDICION 1

Antes que cualquier cosa, hay que tener en cuenta que tenemos 16 interrupciones distribuidas 8 en el puerto E y 8 en el Puerto B EN EL MICROCONTROLADOR ATMEGA169, para seleccionar la activación de un pin determinado, basta poner en uno lógico el bit correspondiente.

Por ejemplo, si queremos que el pin #9 del Atmel 169 (ver tabla 1-5) sea nuestra entrada activa para una interrupción, debemos colocar un 1 lógico en PCMSK0.PCINT7 (BIT #7)

Bits	7	6	5	4	3	2	1	0
PCMSK0	PCIN	PCIN	PCIN	PCIN	PCIN	PCIN	PCIN	PCIN

Figura 1-12. Distribución de bits del Registro PCMSK0

(Pin Change Mask Register 1)

Bits	7	6	5	4	3	2	1	0
PCMSK1	PCIN	PCIN	PCIN	PCIN	PCIN	PCIN	PCIN	PCIN

Figura 1-13. Distribución de bits del Registro PCMSK1

(Pin Change Mask Register 1)

Para hacer una pre-activación en el puerto E (PCINT8 hasta la PCINT15) debemos colocar un 1 lógico en el bit EIMSK.PCIE1 (BIT #7).

Así mismo para habilitar las interrupciones por cambio en los pórticos, del puerto B (PCINT0 hasta la PCINT7) debemos colocar 1 lógico en el EIMSK.PCIE0 (BIT #6).

CONDICION 2

Habilitación en EIMSK.PCIE0, este bit al ponerse a uno habilita las interrupciones por cambio en el pórtico, pero solo del puerto E.

Para habilitar las interrupciones por cambio en el pórtico del puerto B debe poner a 1 lógico el bit EIMSK.PCIE1

Bits	7	6	5	4	3	2	1	0
EIMSK	PCIE	PCIE	-	-	-	-	-	INT0

Register)

Figura 1-14. Distribución de bits del Registro EIMSK,

(External Interrupt Mask)

CONDICION 3

La condición final es habilitar la interrupción global, este es poner en 1 lógico SREG.I (bit #7 del Registro SREG)

1.6.4.3 PASOS PARA CONFIGURACION DE INTERRUPCION INT0

Vector de interrupción, corresponde al 0x0001

```
;VECTORES DE INTERRUPCION----------------------
.ORG 0x0000     ;INICIO DEL PROGRAMA
RJMP RESET
.ORG 0x0004     ;VECTOR PARA INTERRUPCION PCINT0
```

```
RJMP INT_PCINT0
.ORG 0x0006     ;VECTOR PARA INTERRUPCION PCINT1
RJMP INT_PCINT1

        ;-----------------------------------
```

1) Habilitar los bits en los registros correspondientes.

```
;HABILITACION DE REGISTROS PARA UTILIZAR LA INTERRUPCION PCINT0
HABILITAR_PCINT0:
        IN  REG_TEMP,EIMSK      ;GUARDAMOS EL ESTADO ANTERIOR DE GIMSK,
        SBR REG_TEMP,(1<<6)         ;COLOCAMOS UN ALTO EN EL BIT6, NO SE
                                    ALTERA EL RESTO DE BITS, EIMSK.PCIE0
        OUT EIMSK,REG_TEMP              ;ACTUALIZAMOS EL REGSITRO GIMSK
CON EL                                  NUEVOVALOR

        LDS     REG_TEMP,PCMSK0     ;GUARDAMOS EL ESTADO ANTERIOR DE PCMSK0,
        SBR REG_TEMP,(1<<PCINT2)    ;COLOCAMOS UN ALTO EN EL BIT6, NO SE
                                    ALTERA EL RESTO DE BITS
        STS PCMSK0,REG_TEMP             ;ACTUALIZAMOS EL REGSITRO PCMSK0
CON EL                                  NUEVO       VALOR
        SEI                         ;HABILITACION DE INTERRUPCION GLOBAL
RET
```

Una observación importante. En la línea N. 4 debió usarse “SBR REG_TEMP,(1<<PCIE0)”, pero no se utilizo, presento problemas, al parecer en las declaraciones propias del compilador Del valor de PCIE0.

CAPITULO 2

2- FUNDAMENTO TEORICO

Se describe los fundamentos teóricos básicos de las herramientas de software y hardware utilizados para la implementación de este proyecto.

2.1 HERRAMIENTAS DE SOFTWARE PARA EL DESARROLLO DEL PROYECTO.

Para el desarrollo del Proyecto se utilizara las siguientes herramientas de Software:

AVR Studio 4 de Atmel, Entorno de programación para Microcontroladores Atmel.

Proteus 7.7 SP2, Software para la simulación del proyecto para interactuar el hardware y software de manera virtual.

2.1.1 AVRstudio 4

Existen herramientas que son comunes a todas las familias de los AVRs, entre ellas, se puede mencionar el ambiente de Desarrollo denominado AVR STUDIO, que cuenta con la posibilidad de, no solo programar en assembler, sino de integrar un compilador para C (GNU/GCC).

Ambos son gratuitos y se pueden obtener accediendo al website de ATMEL. La característica principal del **AVR STUDIO** es su fácil manejo, permite visualizar rápidamente que ocurre con los registros, como también así se pueden modificar. En este ámbito se pueden realizar simulaciones por software como así también un debug utilizando.

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

AVR Studio 4 consiste de muchas ventanas y sub-módulos.

2.1.1.1 Descripción general del IDE en AVRstudio 4

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 2.2 se puede apreciar las ventanas principales del IDE.

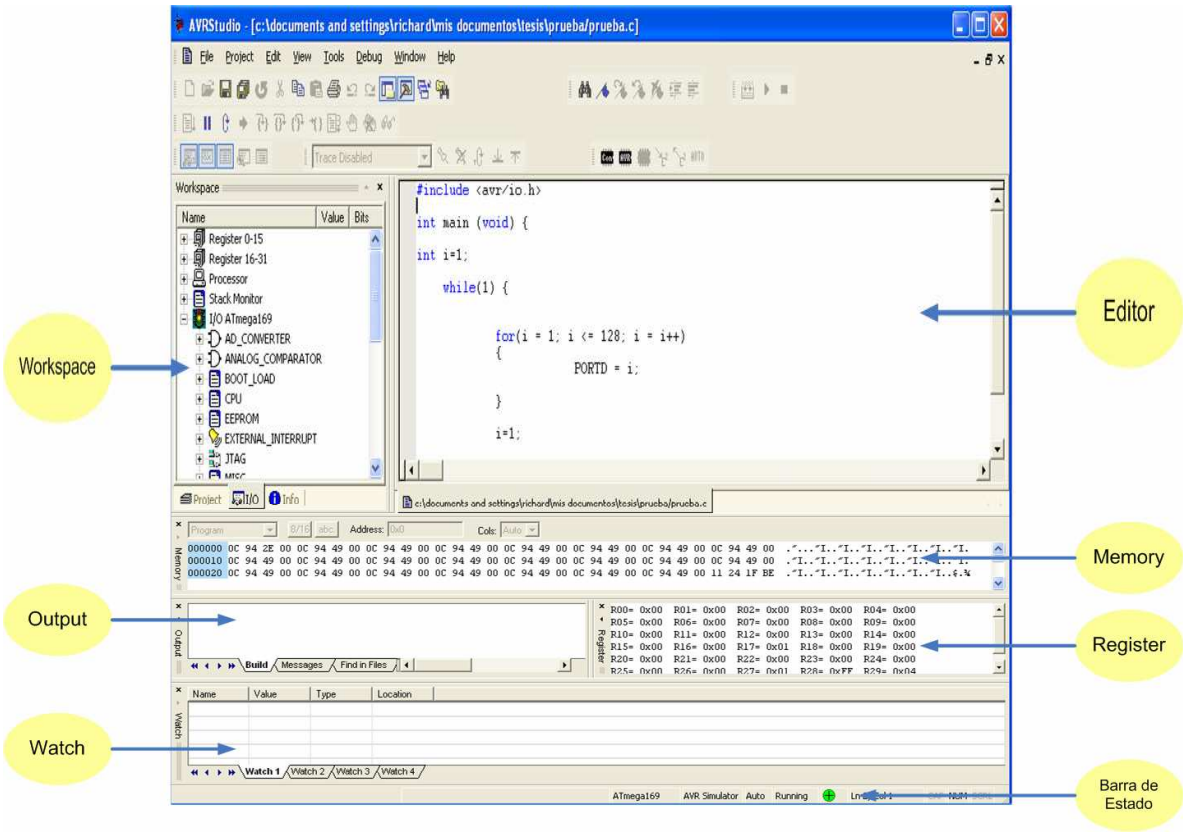

Figura 2-1: entorno de desarrollo de AVRstudio 4

2.1.1.2 Generación de proyectos en AVR Studio 4.

Al iniciar el AVR Studio 4 desde el menú [Inicio] [Programas] [Atmel AVR Tools].

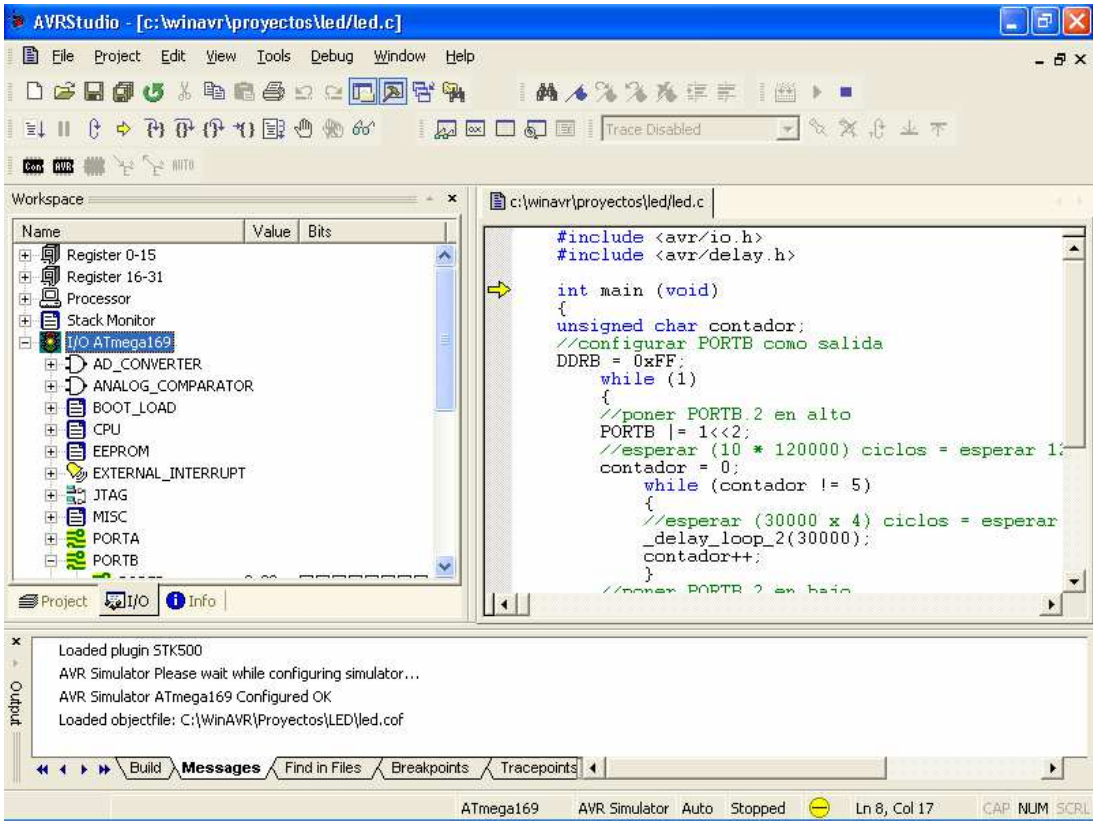

Figura 2-2 AVR Studio, interfaz del usuario en AVR Simulator

El AVR Studio 4 soporta un amplio rango de herramientas para emulación y depuración. Por conveniencia el usuario debe seleccionar la funcionalidad de simulación incluida, el AVR Simulator; ya que, el resto de opciones requieren de H/W especial. Se debe seleccionar el Microcontrolador que utilizará en su aplicación, en este caso ATmega169. El AVR Studio iniciará y abrirá un archivo de extensión ".asm".

2.1.1.3 Simulador en AVR Studio 4.

El AVR Simulator es un simulador para la arquitectura y dispositivos AVR. Este simula la CPU, incluyendo todas las instrucciones, interrupciones y la mayoría de los módulos de I/O del chip.

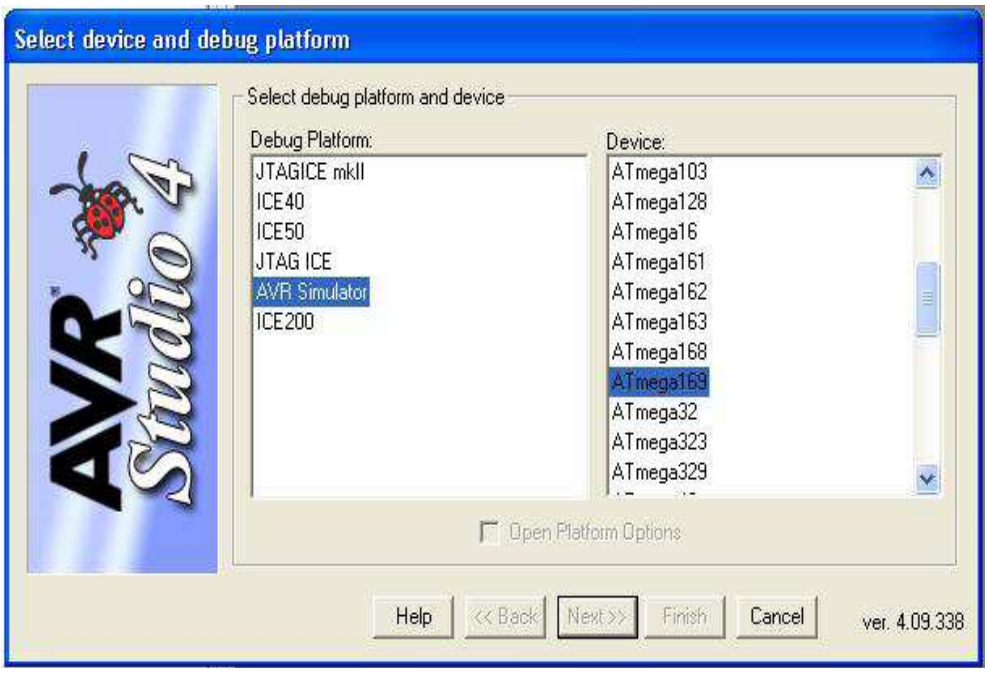

Figura 2-3: AVR Studio, selección de Microcontrolador

El AVR Simulator permite al usuario usar los comandos normales de depuración tal como Run, Break, Reset, Single step, set breakpoints y watch variables. Las vistas I/O, Register y Memory son totalmente funcionales usando el AVR Simulator.

2.1.2 PROTEUS VERSIÓN 7.4

PROTEUS es un programa para simular circuitos electrónicos complejos integrando inclusive desarrollos realizados con Microcontroladores de varios tipos, en una herramienta de alto desempeño con unas capacidades graficas impresionantes.

Presenta una filosofía de trabajo semejante al SPICE, arrastrando componentes de una barra e incrustándolos en la aplicación.

El Programa PROTEUS es una aplicación CAD que se compone de tres módulos básicos:

ISIS ("Esquema Inteligente de Sistema de Entrada") que es el módulo de captura de esquemas.

VSM ("Modelamiento de Sistema Virtual") es el módulo de simulación, incluyendo PROSPICE.

ARES ("Modelamiento Avanzado de Enrutamiento") es el módulo para realización de circuitos impresos (PCB).

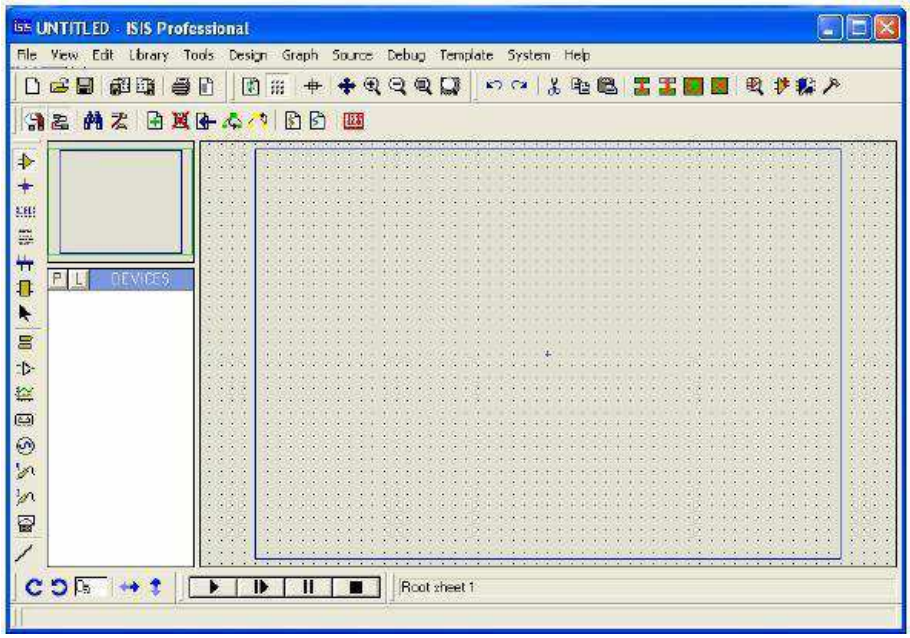

Figura 2-4: Interfaz gráfica de Proteus

2.2 HERRAMIENTAS DE HARDWARE PARA LA IMPLEMENTACIÓN DEL PROYECTO.

KIT DE DESARROLLO AVR BUTTERFLY, es el Hardware para el desarrollo del proyecto, se describe a continuación.

2.2.1 KIT DE DESARROLLO AVRBUTTERFLY

El Kit AVR Butterfly se diseñó para demostrar los beneficios y las características 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 2.1 se puede apreciar el Kit AVR Butterfly.

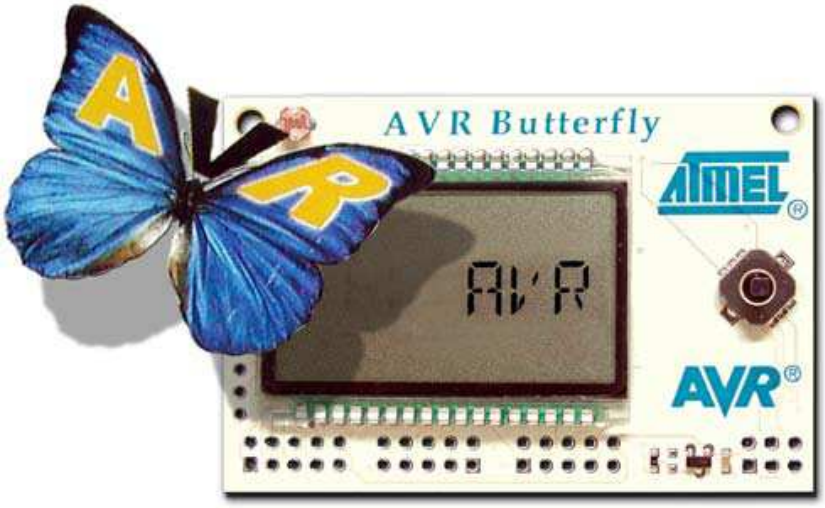

Figura 2-5 Kit de Desarrollo AVR Butterfly

Características del Kit de Desarrollo AVRButterfly:

- 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.
- Data Flash 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).

2.2.1.1 Hardware Disponible En El Kit AvrButterfly

Los siguientes recursos están disponibles en el Kit AVR Butterfly:

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), conectado en PF0.

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 pre programada.

Compatibilidad con el Entorno de Desarrollo AVR Studio 4.

En las Figuras 2.7y 2.8 se observa el Hardware disponible en el AVR Butterfly.

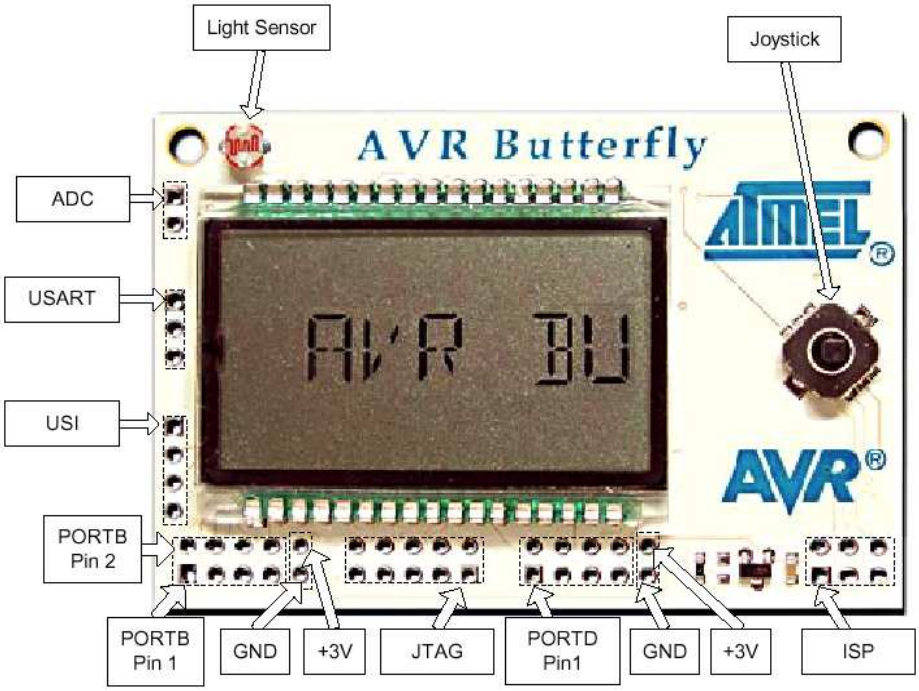

Figura 2-6: Hardware Disponible (Parte Frontal)

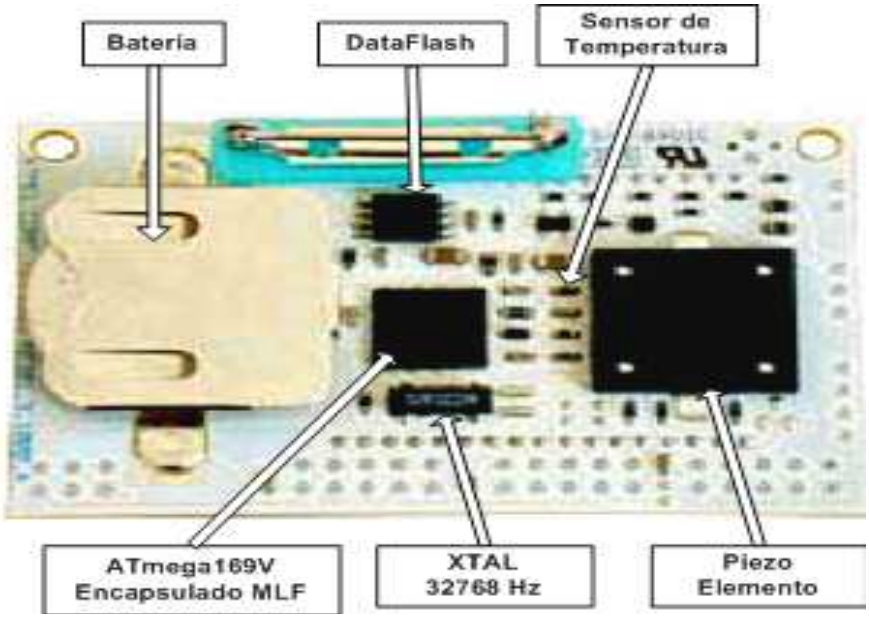

Figura 2-7: Hardware Disponible (Parte Posterior)

2.2.1.2 FIRMWARE INCLUIDO en el Kit AVR Butterfly

El AVR Butterfly viene con una aplicación pre programado. Esta sección presentará una revisión de los elementos de esta aplicación.

Los siguientes bloques vienen pre programados en el AVR Butterfly:

- Código Cargador de Arranque (BootloaderCode).
- Código de la Aplicación.
- Máquina de Estados.
- Funciones incluidas:
- Nombre-etiqueta.

- Reloj (fecha).
- Mediciones de temperatura.
- Mediciones de luz.
- Lecturas de voltaje.
- Reproducción de tonadas/melodías.
- Ahorro de energía automático.
- Ajuste de contraste del LCD.
- Más funciones podrán ser agregadas después, como por ejemplo:
- Calculadora.
- Función de recordatorio.
- Alarma (alarmas diarias, temporizadores para la cocina, etc.).
- Reproducción de melodías y visualización del texto (función de Karaoke).
- Con la DataFlash de 4 Mb el usuario podrá almacenar una cantidad grande de datos.

La Figura 2.8 muestra el menú de la aplicación que viene con el AVR Butterfly. La columna a la izquierda muestra el menú principal: “AVR Butterfly”, “Tiempo”, “Música”, etc.

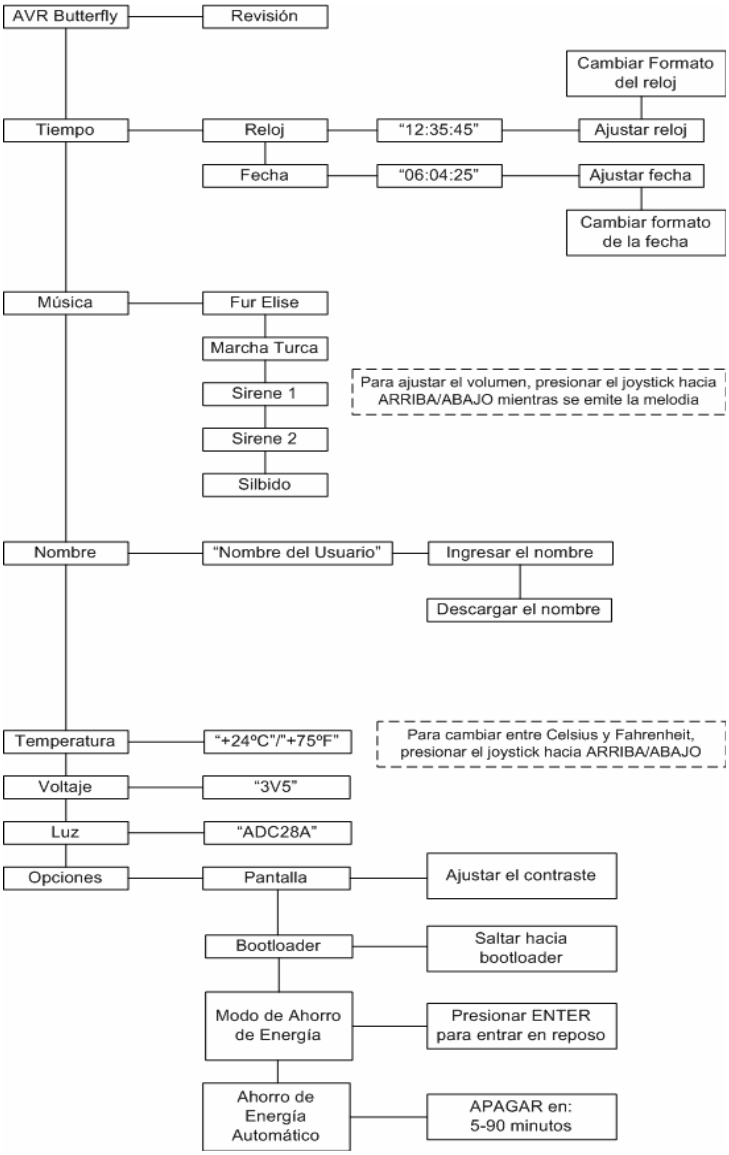

Figura. 2-8 Firmware Incluido en el AVR Butterfly (en español)

2.2.1.3 JOYSTICK

Para operar el AVR Butterfly se emplea el joystick como una entrada para el usuario. Este opera en cinco direcciones, incluyendo presión en el centro; tal como se puede ver en la Figura 2.5

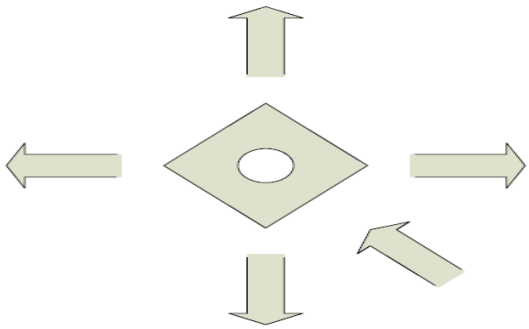

Figura 2-9: Entrada tipo Joystick

2.2.1.4 ACTUALIZACIÓN. EL BOOTLOADER

El AVR Butterfly viene con un Bootloader que usa la característica self-programming del microcontrolador ATmega169. El Bootloader combinado con el circuito convertidor de nivel RS-232, integrado en el AVR Butterfly, hace posible actualizar la aplicación sin ningún hardware externo adicional. El AVR Prog, que es una herramienta incluida en el AVR Studio 4, es usado como interfaz entre la PC y el usuario. Ver Figura 2.11.

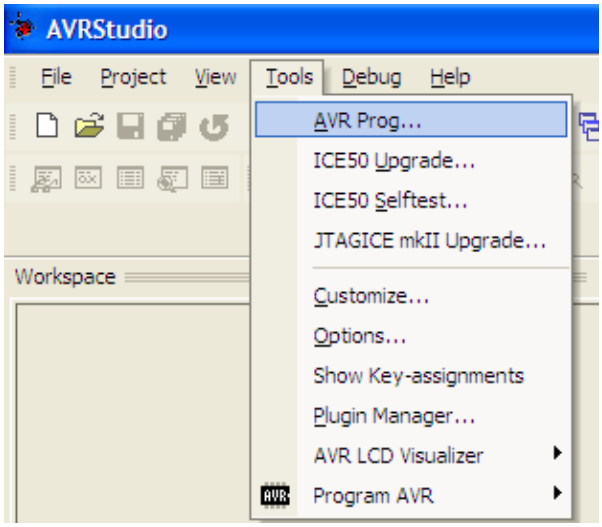

Figura. 2-10 AVR Prog en el AVR Studio4

Los datos son transmitidos hacia el Microcontrolador a través de la interfaz RS-232, para lo cual se debe conectar un cable serial desde la PC hacia el AVR Butterfly.

2.2.1.5 Actualización del AVR ATmega169 en el Kit AVR Butterfly

Desde la aplicación del AVR Butterfly se puede saltar hacia la sección de Arranque (bootsection), desplazándose a través del menú: "Opciones>Bootloader>Saltar hacia Bootloader", simplemente reseteando el microcontrolador ATmega169 al cortocircuitar los pines 5 y 6 en el conector J403, conector ISP. Ver Figura 2.12

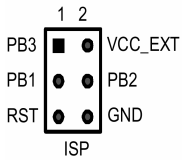

Figura. 2-11: Conector J403 para ISP

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, como se ve en la Figura 2.13, 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 “ErasingDevice”, “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.

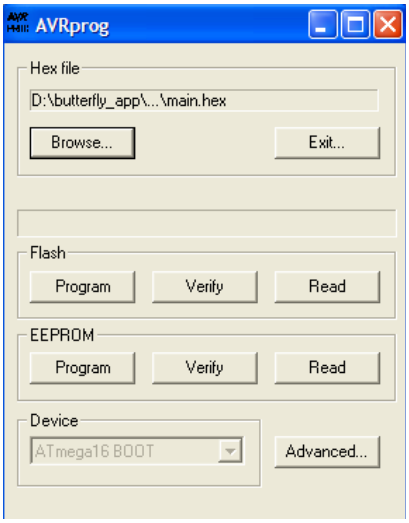

Figura. 2-12: AVR Prog

2.2.1.6 Saltar hacia el Sector de la Aplicación.

Para saltar de la Sección de Arranque hacia la de Aplicación presione el joystick hacia ARRIBA.

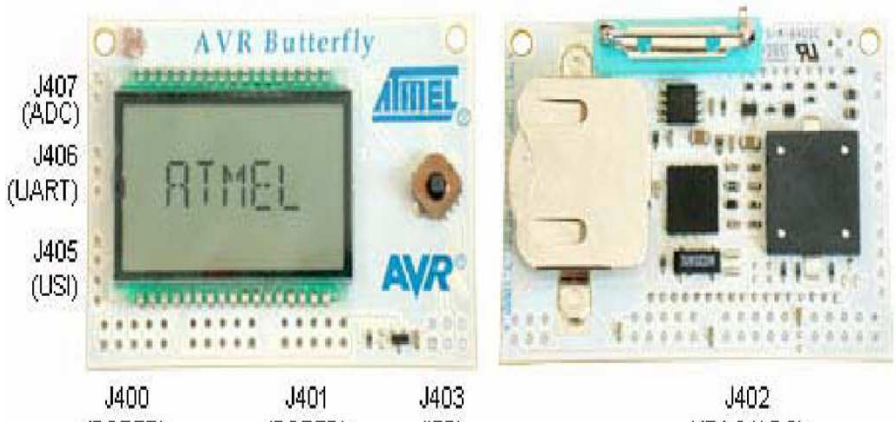

Figura. 2-13: Conectores del AVR Butterfly para acceso a periféricos

2.2.1.7 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.

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.

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

Tabla. 2-1 Distribución de pines, AVR Butterfly Vs. PC

En la Figura 2.14 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 seaprecia un conector DB9 hembra soldado a los cables que se conectan a la interfaz USART del AVR Butterfly (derecha).

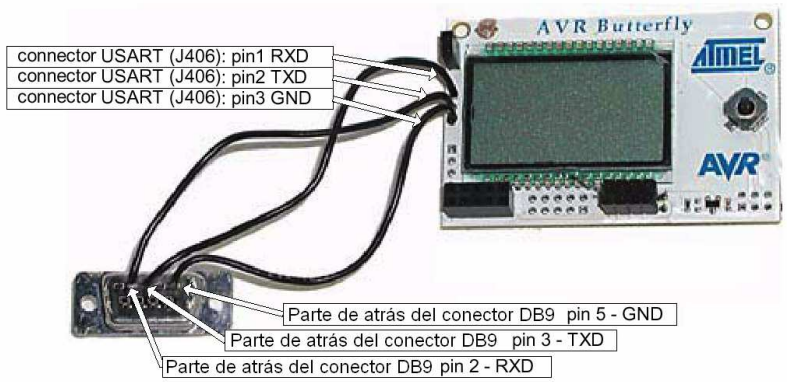

Figura. 2.14 Conexiones para interfaz USART del AVR Butterfly

2.2.1.8 El LCD delKit AVR Butterfly

Interfaz muy simple para mostrar información podría ser el estado de unos LEDs. 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.

2.2.1.9 Conexiones entre el LCD y el microcontrolador ATmega169 en el Kit AVR Butterfly.

Los pines para el LCD en el AVR se sitúan en PORTA, PORTC, PORTD y PORTG. El vidrio LCD incluido en el AVR Butterfly tiene un total de 120 segmentos controlados a través de cuatro terminales comunes y 30 líneas de segmento. Puesto que el Atmega169 es capaz de manejar 100 segmentos, algunos de los segmentos del vidrio LCD no están conectados en el AVR Butterfly.

En la Figura 2.17 se muestra el vidrio LCD montado en el AVR Butterfly. Este consiste de siete símbolos alfanuméricos y varios símbolos fijos: números de cero a nueve, una campana, un indicador de batería descargada (low-battery) y flechas de navegación

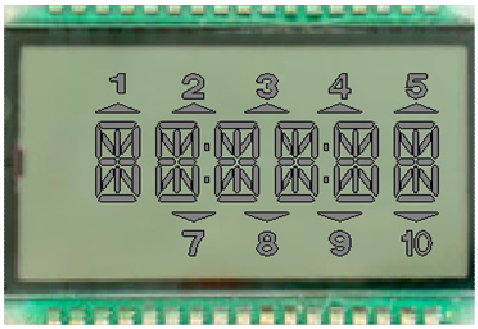

Figura. 2-15 Vidrio LCD

El AVR Butterfly tiene seis grupos similares de segmentos, donde cada grupo de segmentos es capaz de desplegar un carácter alfanumérico. Un cierto grupo de segmentos capaz de desplegar un carácter alfanumérico es llamado un dígito LCD. Este consiste de 14 segmentos separados. La Figura 2.18 muestra un dígito LCD y la letra usada para referirse a cada uno de los segmentos dentro del dígito LCD.

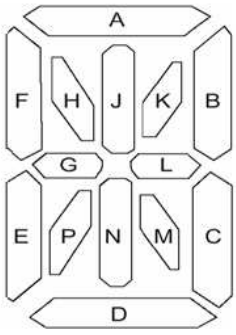

Figura. 2-16 Segmentos y Letras de Referencia de los Dígitos LCD

El LCD debe funcionar a ¼ Duty y 1/3 Bias (ver Capítulo 2, Controlador de LCD) para poder controlar los cuatro terminales comunes. Es muy importante notar que la

energía para el LCD es suministrada desde el microcontrolador ATmega169 y no tienen líneas de alimentación separadas.

2.2.2 Fuente de alimentación externa.

El avrbutterfly puede ser alimentado/energizado por una fuente de voltaje externa de 3 v como se muestra en la figura 2.19 a través de los pines vcc_ext y gnd de los conectores externos portb y portd.

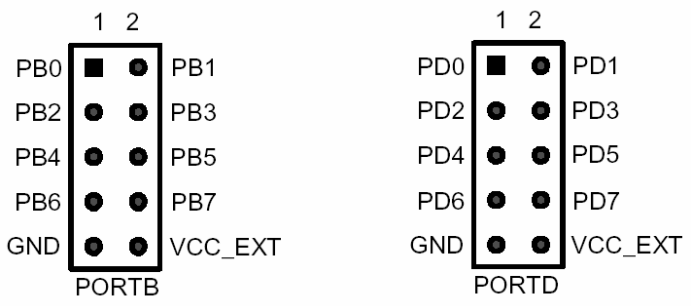

Figura. 2-17 AVR Butterfly, PORTB (a) y PORTD (b)

Se recomienda utilizar dos baterías tipo AA de 1.5 V y un porta-baterías para las mismas,

CAPITULO 3

3.-DESCRIPCION DE LOS PROYECTOS

A continuación se describen 5 proyectos ilustrativos que tiene como objetivo mostrar las funcionalidades y desarrollos básicos realizados en el BUTTERFLY, dos de estos serán desarrollados en Asembler, y tres en lenguaje C.

3.1. PROYECTO1: MANEJO DE JOYSTICK POR MEDIO DE INTERRUPCIONES PCINT0 Y PCINT1.

3.1.1 OBJETIVO.-

Manejo de sentencias en Asembler, configuración de Joystick del AVRBUTTERFLY, por medio de interrupciones, Utilización del Modulo Usart, para enviar dato referente a la Opción escogida por medio del Joystick.

3.1.2 ENUNCIADO:

Elaborar un Programa en lenguaje C que habilite el Joystick del ArvButterfly, al presionar cualquier opción del Joystick este deberá Transmitir el nombre de la Opción escogida por medio de transmisión Usart. Por lo tanto si se presiona el Joystick hacia arriba deberá transmitir la palabra ARRIBA. y de esta manera para las otras 4 opciones restantes.

3.1.3 ESQUEMA DEL PROYECTO.

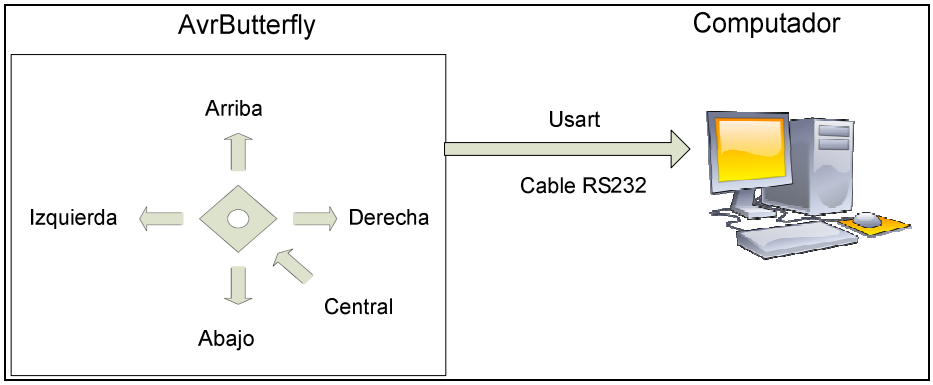

Figura 3-1 Diagrama Esquemático de PROYECTO 1

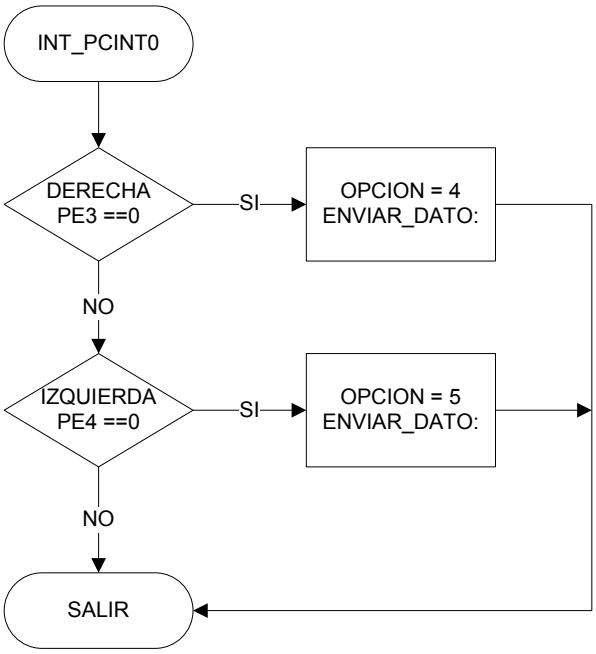

Figura 3-2 Diagrama de Flujo de Subrutina INT_PCINT0

3.1.4 Diagrama de flujo del Proyecto.

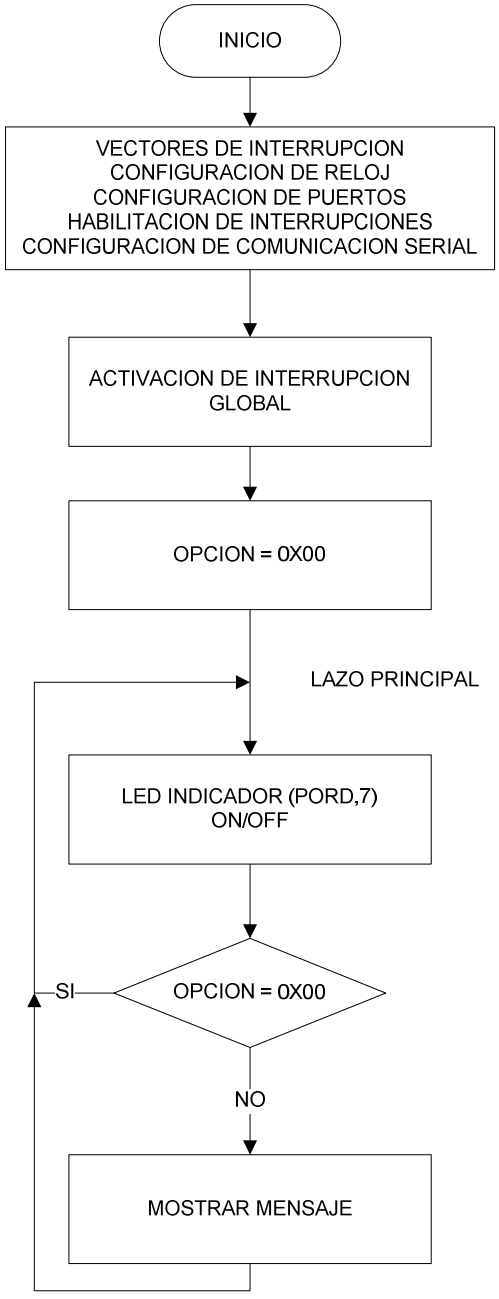

Figura 3-3 Diagrama de Flujo de PROYECTO 1

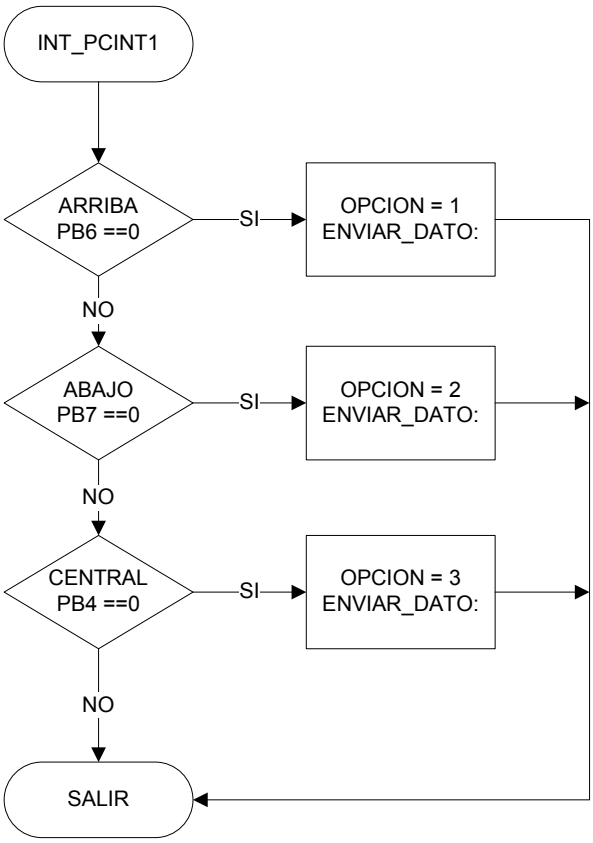

Figura 3-4 Diagrama de Flujo de Subrutina INT_PCINT1

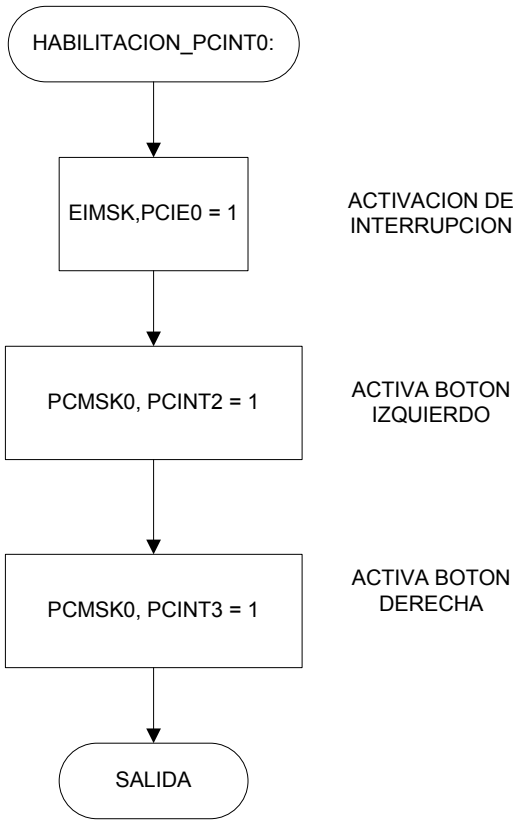

Figura 3-5 Diagrama de Flujo de Rutina HABILITACION _PCINT0

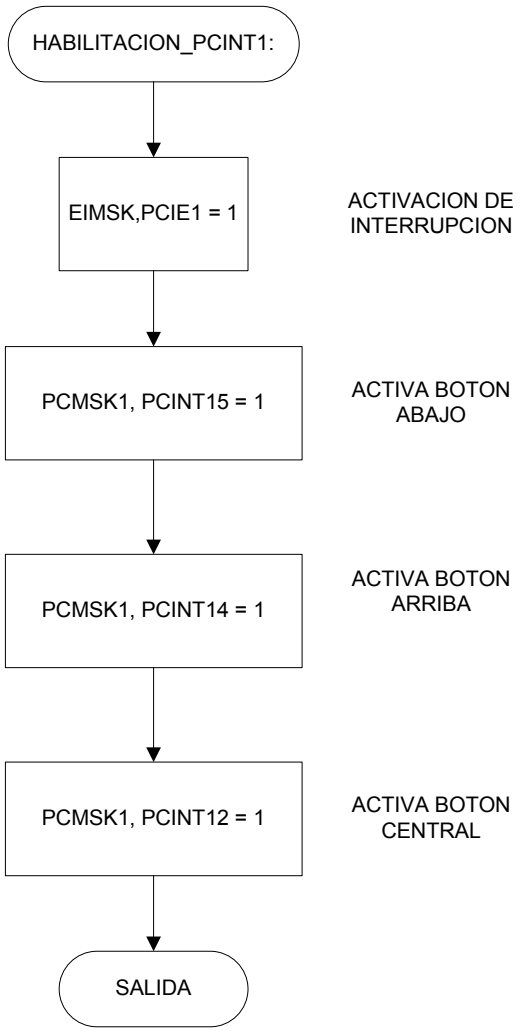

Figura 3-6 Diagrama de Flujo de Rutina HABILITACION _PCINT0

3.1.5 CÓDIGO FUENTE.

```
/*
 * PRIMERO.asm
 *
 *  Created:            27/10/2011 10:22:59
 *  Author:             Giovanni Granados & Jorge Vega.
 *  Tema:               MANEJO DE JOYSTICK POR MEDIO DE INTERRUPCIONES PCINT0
 *                      Y PCINT1
 */



;INICIO DE PROGRAMA
.INCLUDE "M169DEF.INC"  ;BUTTERFLY/ATMEGA169 DEFS


.def REG_TEMP=r16     ;ASIGNA UN NOMBRE A REGISTRO
.def Temp     =r17    ;USO TEMPORAL.
.def ROJO     =R18
.def AZUL     =R19

.def Delay    =r20    ;Delay variable 1 PARA USO DE RETARDO
.def Delay2   =r21    ;Delay variable 2 PARA USO DE RETARDO
.def OPCION   =R22    ;OPCION DE JOYSTICK
.def COLUMNA =R23
.def LINEA    =R24

.def SELECCION=R25

.def VERDE    =R26
.def CENTRAL  =R27



//VECTORES DE INTERRUPCION-----------------------------------
.ORG 0x0000           ;INICIO DEL PROGRAMA
RJMP RESET
.ORG 0x0004                 ;VECTOR PARA INTERRUPCION PCINT0(PUERTO E)
RJMP INT_PCINT0
.ORG 0x0006                 ;VECTOR PARA INTERRUPCION PCINT1(PUERTO B)
RJMP INT_PCINT1
//-----------------------------------------------------------
```

```
RESET:
        //CONFIGURACIN DE PILA Y MEMORIA
   LDI COLUMNA,HIGH(RAMEND)
   OUT SPH,COLUMNA
   LDI COLUMNA,LOW(RAMEND)
   OUT SPL,COLUMNA


        //CONFIGURACION DE RELOJ
        //8MHZ
        LDI REG_TEMP, 0B10000000
        STS CLKPR,REG_TEMP
        LDI REG_TEMP, 0B00000000
        STS CLKPR,REG_TEMP

        //CONFUGURA LOS PUERTOS I/O
        RCALL CONFIGURACION_PUERTOS


//HABILITACIONES DE INTERRUPCIONES

        RCALL HABILITAR_PCINT0
        RCALL HABILITAR_PCINT1

//HABILITACION DE COMUNICACION SERIAL
        RCALL HABILITAR_COMUNICACION_SERIAL

        SEI
        LDI OPCION,0X00

LAZO_PRINCIPAL:

        RCALL DLY
        SBI PORTD,7
        RCALL DLY
        CBI PORTD,7
        NOP

        //COMPARO OPCION
        CPI             OPCION,0X00
        BREQ   LAZO_PRINCIPAL
        RCALL MENSAJES
        LDI             OPCION,0X00
```

```
RJMP LAZO_PRINCIPAL


//-----------------------------------------------------
//CONFIGURACION DE PUERTOS
//PORTB COMO ENTRADA
//PORTE COMO ENTREDA EXPTO TX QUE ES SALIDA.
CONFIGURACION_PUERTOS:


        LDI REG_TEMP,0B11110000      //CONFIGURACION DE PUERTO B
        LDI TEMP,        0B00101111
        OUT PORTB,REG_TEMP
        OUT DDRB,TEMP
        NOP

        LDI REG_TEMP,0B11111111 //CONFIGURACION PUERTO E
        LDI TEMP,        0B11110011
        OUT PORTE,REG_TEMP
        OUT DDRE,TEMP
        NOP



        LDI REG_TEMP,0B00000000 //CONFIGURACION PUERTO D
        LDI TEMP,        0B11111111 //PORTD,7 INDICADOR DE TRABAJO
        OUT PORTD,REG_TEMP
        OUT DDRD,TEMP
        NOP

RET //FIN CONFIGURACION DE PUERTOS
//=====================================================




//-----------------------------------------------------
//HABILITAR_COMUNICACION_SERIAL:
//

HABILITAR_COMUNICACION_SERIAL:
        //PRUSART0 = 0   EN REGISTRO PRR (POWER REDUCCION REGISTER)
        LDS REG_TEMP,PRR                    ;GUARDAMOS ESTADO ANTERIOR DE
REGISTRO PRR
```

```
        CBR REG_TEMP,(1<<PRUSART0)        ;BIT#2 EN CERO, SE MANTIENEN LOS
OTROS VALORES.
        STS PRR,REG_TEMP                  ;SE CARGA VALOR A REGISTRO PRR.

        //CONFIGURACION DE UCSRC (USART CONTROL AND ESTATUS REGISCTRER C)
        //UMSEL         =         0               0 ->ASINCRONO.
        //                                        1 ->SINCRONO.

        //UPM1 =        0               0 0 DESABILITADA PARIDAD
        //UPM0 =        0               0 1 RESERVADO
        //                                      1 0 PARIDAD PAR
        //                                      1 0 PARIDAD IMPAR

        //USBS  =       0               0 ->1 BIT DE PARADA.
        //                                      1 ->2 BITS DE PARADA.

        //UCSZ1 =       1               0 0 ->5 BITS
        //UCSZ1 =       1               0 1 ->6 BITS
        //                                      1 0 ->7 BITS
        //                                      1 1 ->8 BITS
        //OBS: UCZ2=0 EN EL REGISTRO UCSRB

        //UCPOPL =      0               POLARIDAD DE RELOJ MODO SINCRONO.
        //                                      NO USADO AQUI.

        LDI REG_TEMP,0B00000110
        STS     UCSRC,REG_TEMP            ;CONFIGURACION DE UCSRC

        //HABILITACION DE TXEN EN EL REGISTRO UCSRC

        //LDI REG_TEMP,0B00000000// prueba de cambios...
        //STS UCSRC,REG_TEMP

        LDS REG_TEMP,UCSRB                ;GUARDAMOS ESTADO ANTERIOR DE
REGISTRO UCSRB
        SBR REG_TEMP,(1<<TXEN)+(1<<RXEN)        ;BIT#3 EN 1, SE MANTIENEN LOS
OTROS VALORES.
        STS UCSRB,REG_TEMP                ;SE CARGA VALOR A REGISTRO UCSRB.


        //CONFIGURACION DE VELOCIDAD DE TRANSMISION
        //DEPENDE DE LA CONFIGURACION DE RELOJ PRINCIPAL
        //BAUDIO = 2f/(16(UBRR+1)
        LDI REG_TEMP, 0B00000000
        STS UBRRH,REG_TEMP
        // EL RELOJ ESTA TRABAJANDO A 8 MHZ
```

```
        LDI REG_TEMP, 51 //VER TABLA DE VALORES
        STS UBRRL,REG_TEMP


RET //HABILITAR_COMUNICACION_SERIAL
//=====================================================




//TRANSMITIR_DATO_USART:
//DATO A ENVIAR ESTA EN REG_TEMP
TRANSMITIR_DATO_USART:
        LDS             Temp,UCSRA
        SBRS    Temp,UDRE
        RJMP    TRANSMITIR_DATO_USART
        STS             UDR,REG_TEMP
RET //FIN TRANSMITIR DATO
//=====================================================




;HABILITACION DE REGISTROS PARA UTILIZAR LA INTERRUPCION PCINT0
HABILITAR_PCINT0:
        //habilitamos interrupcion PCIE0
        IN      REG_TEMP,EIMSK          ;GUARDAMOS EL ESTADO ANTERIOR DE
GIMSK,
        SBR REG_TEMP,(1<<6)         ;COLOCAMOS UN ALTO EN EL BIT6(PCIE0), NO SE
ALTERA EL RESTO DE BITS
        OUT EIMSK,REG_TEMP          ;ACTUALIZAMOS EL REGSITRO GIMSK
CON EL NUEVO VALOR

        LDS     REG_TEMP,PCMSK0         ;GUARDAMOS EL ESTADO ANTERIOR DE
PCMSK0,
        SBR REG_TEMP,(1<<PCINT2) + (1<<PCINT3);COLOCAMOS UN ALTO EN EL BIT6,
NO SE ALTERA EL RESTO DE BITS
        STS PCMSK0,REG_TEMP         ;ACTUALIZAMOS EL REGSITRO PCMSK0
CON EL NUEVO VALOR


RET
```

```
;HABILITACION DE REGISTROS PARA UTILIZAR LA INTERRUPCION PCINT1
HABILITAR_PCINT1:
        //habilitamos interrupcion PCIE1
        IN      REG_TEMP,EIMSK          ;GUARDAMOS EL ESTADO ANTERIOR DE
GIMSK,
        SBR REG_TEMP,(1<<7) ;COLOCAMOS UN ALTO EN EL BIT7 (PCIE1), NO SE
ALTERA EL RESTO DE BITS
        OUT EIMSK,REG_TEMP              ;ACTUALIZAMOS EL REGSITRO GIMSK
CON EL NUEVO VALOR

        LDS     REG_TEMP,PCMSK1         ;GUARDAMOS EL ESTADO ANTERIOR DE
PCMSK0,
        SBR REG_TEMP,(1<<PCINT12) + (1<<PCINT14) + (1<<PCINT15); + (
1<<PCINT15);COLOCAMOS UN ALTO EN EL BIT6, NO SE ALTERA EL RESTO DE BITS
BOTON B NO FUNCIONARA SERA PWM
        STS PCMSK1,REG_TEMP             ;ACTUALIZAMOS EL REGSITRO PCMSK0
CON EL NUEVO VALOR


RET

        ;RUTINA DELAY

        DLY:
        dec             Delay
        brne    DLY
        dec             Delay2

        brne    DLY
        ret



//-------------------------------------------------------
INT_PCINT1:
CLI

        //MOV REG_TEMP,OPCION
        NOP

        SBIS PINB,6
                RJMP ARRIBA //ARRIBA
        SBIS PINB,7
                RJMP ABAJO  //ABAJO
        SBIS PINB,4
                RJMP CENTRO_BNT //CENTRO
```

```
            NOP
        RJMP SALIDA_INT_PCINT1
        ;;;;;;;;;;;;;;;;;;;;;;;;
        ARRIBA:

            LDI             OPCION,'A'
            RJMP    SALIDA_INT_PCINT1

        ABAJO:
            LDI OPCION,'B'
            RJMP    SALIDA_INT_PCINT1


        CENTRO_BNT:
            LDI OPCION,'C'
            RJMP    SALIDA_INT_PCINT1

        SALIDA_INT_PCINT1:

SEI
RETI
//////////////////////////////////////////////////////////

//-------------------------------------------------------
INT_PCINT0:
CLI
        SBIS    PINE,2
            RJMP    IZQUIERDO
        SBIS    PINE,3
            RJMP    DERECHA
            nop
        RJMP    SALIDA_INT_PCINT0

        IZQUIERDO:
            LDI             OPCION,'I'
            RJMP    SALIDA_INT_PCINT0

        DERECHA:
            LDI             OPCION,'D'
            RJMP    SALIDA_INT_PCINT0

SALIDA_INT_PCINT0:

SEI
RETI
```

```
////////////////////////////////////////////////////

MENSAJES:
CPI OPCION,'A'
BRNE   COMPARAR_CON_B
//CASO CONTRARIO ARRIBA
        LDI REG_TEMP,'A'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'R'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'R'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'I'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'B'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'A'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'-' //SEPARADOR
        RCALL TRANSMITIR_DATO_USART
RJMP SALIR_DE_MENSAJE


COMPARAR_CON_B:
CPI             OPCION,'B'
BRNE   COMPARAR_CON_C

//CASO CONTRARIO ABAJO
        LDI REG_TEMP,'A'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'B'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'A'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'J'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'O'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'-' //SEPARADOR
        RCALL TRANSMITIR_DATO_USART
RJMP SALIR_DE_MENSAJE

COMPARAR_CON_C:
CPI             OPCION,'C'
BRNE   COMPARAR_CON_I
```

```
//CASO CONTRARIO CENTRO
        LDI REG_TEMP,'C'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'E'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'N'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'T'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'R'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'0'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'-' //SEPARADOR
        RCALL TRANSMITIR_DATO_USART
RJMP SALIR_DE_MENSAJE


COMPARAR_CON_I:
CPI             OPCION,'I'
BRNE   COMPARAR_CON_D


//CASO CONTRARIO IZQUIERDA
        LDI REG_TEMP,'I'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'Z'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'Q'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'U'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'I'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'E'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'R'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'D'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'A'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'-' //SEPARADOR
        RCALL TRANSMITIR_DATO_USART
RJMP SALIR_DE_MENSAJE
```

```
COMPARAR_CON_D:
CPI                 OPCION,0X00
BREQ   SALIR_DE_MENSAJE

LDI REG_TEMP,'D'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'E'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'R'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'E'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'C'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'H'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'A'
        RCALL TRANSMITIR_DATO_USART
        LDI REG_TEMP,'-' //SEPARADOR
        RCALL TRANSMITIR_DATO_USART
RJMP SALIR_DE_MENSAJE

SALIR_DE_MENSAJE:
RET
```

3.2. PROYECTO2: VARIACIONES DE COLOR DE LED RGB, POR MEDIO DE PWM

3.2.1 OBJETIVO.-

Mostrar el uso de las Interrupciones para generar PWM, y así poder generar una gama de colores a partir de un LED RGB.

3.2.2 ENUNCIADO:

Elaborar un controlador continuo para Led Tricolor, por medio de PWM, este debe cambiar de color en forma tenue y aleatoria.

3.2.3 ESQUEMA DEL PROYECTO.

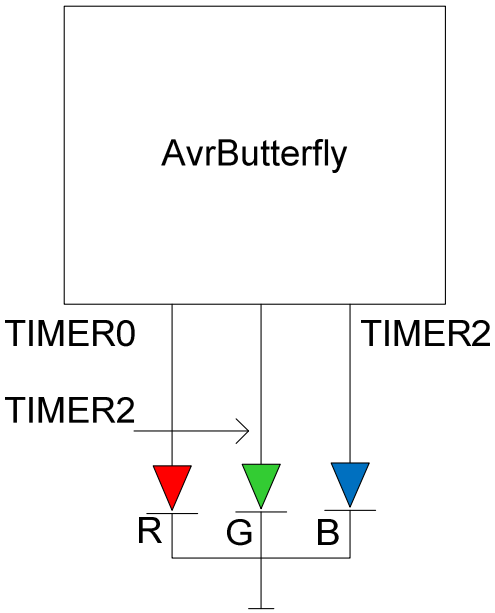

Figura 3-7 Diagrama de Bloque Proyecto 2

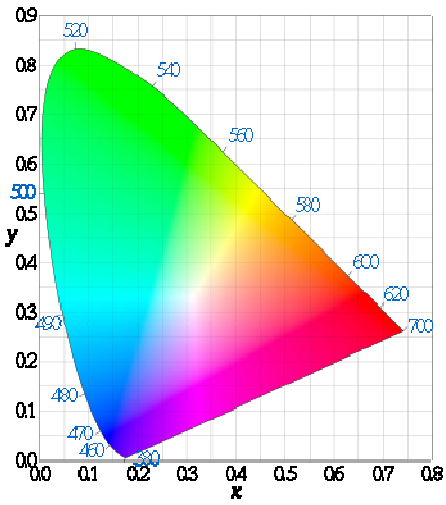

Figura 3-8 Gamas de Colores

3.2.4 DIAGRAMA DE FLUJO DEL PROYECTO.

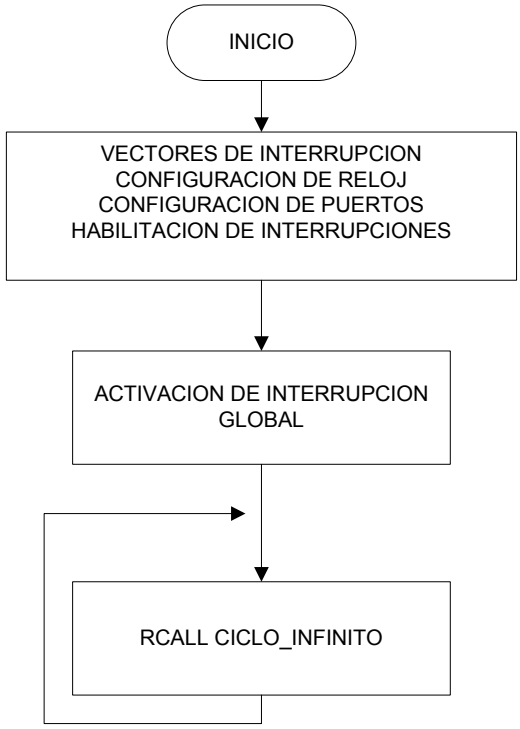

Figura 3-9 Diagrama de Flujo Principal Proyecto 2

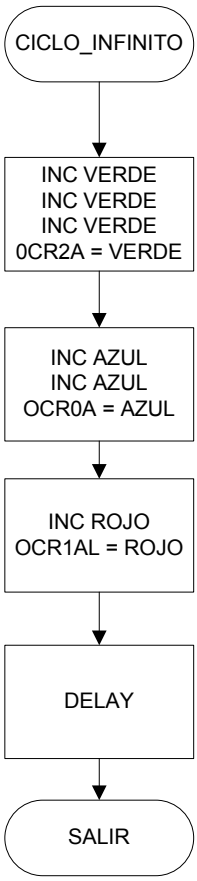

Figura 3-10 Diagrama de Flujo Rutina Ciclo Infinito

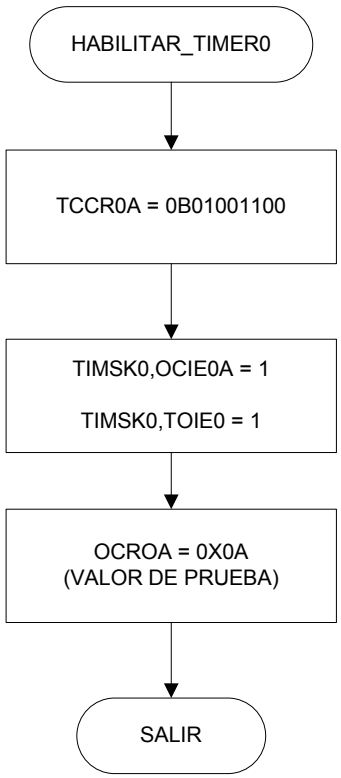

Figura 3-11 Diagrama de Flujo Habilitación de Timer0

3.2.5 CÓDIGO FUENTE.

```
/*
 * PINTX.asm
 *
 *  Created:        27/10/2011 10:22:59
 *  Author:                 Giovanni Granados & Jorge Vega.
 *          Tema:           VARIACIONES  DE COLOR DE LED RGB , POR MEDIO DE PWM
 */



;INICIO DE PROGRAMA
.INCLUDE "M169DEF.INC"  ;BUTTERFLY/ATMEGA169 DEFS


.def REG_TEMP=r16       ;ASIGNA UN NOMBRE AL REGISTRO R16 PARA HACERLO MAS
COMPRENSBLE...!!
.def Temp               =r17    ;USO TEMPORAL.
.def ROJO               =R18
.def AZUL               =R19

.def Delay              =r20    ;Delay variable 1 PARA USO DE RETARDO

.def Delay2             =r21    ;Delay variable 2 PARA USO DE RETARDO

.def OPCION_LED                 =R22    ;VARIABLE QUE CAMBIARA A CADA MOMENTO.
.def COLUMNA            =R23
.def LINEA              =R24

.def OPCION     =R25

.def VERDE              =R26
.def CENTRAL =R27



;VECTORES DE INTERRUPCION-------------------------
.ORG 0x0000             ;INICIO DEL PROGRAMA
RJMP RESET
//.ORG 0x0004                   ;VECTOR PARA INTERRUPCION PCINT0(PUERTO E)
//RJMP INT_PCINT0
.ORG 0x0008                     ;VECTOR PARA INTERRUPCION TIMER2 COMP
RJMP INT_TIMER2_COMP
```

```
.ORG 0x000A                ;VECTOR PARA INTERRUPCION TIMER0 COMP
RJMP INT_TIMER2_OVF
.ORG 0x0010                ;VECTOR PARA INTERRUPCION TIMER0 COMP
RJMP INT_TIMER1_COMPB
.ORG 0x0012                ;VECTOR PARA INTERRUPCION TIMER0 COMP
RJMP INT_TIMER1_OVFA
.ORG 0x000E                ;VECTOR PARA INTERRUPCION TIMER0 COMP
RJMP INT_TIMER1_COMPA
.ORG 0x0014                ;VECTOR PARA INTERRUPCION TIMER0 COMP
RJMP INT_TIMER0_COMP
.ORG 0x0016                ;VECTOR PARA INTERRUPCION TIMER0 COMP
RJMP INT_TIMER0_OVF



RESET:

LDI COLUMNA,HIGH(RAMEND)  ;SETUP THE STACK POINTER
OUT SPH,COLUMNA           ;AT TOP OF MEMORY AND
LDI COLUMNA,LOW(RAMEND)   ;LET STACK GROW DOWNWARDS
OUT SPL,COLUMNA


//CONFIGURACION DE RELOJ A 8MHZ
LDI REG_TEMP, 0B10000000
STS CLKPR,REG_TEMP
LDI REG_TEMP, 0B00000000
STS CLKPR,REG_TEMP

RCALL CONFIGURACION_PUERTOS



//HABILITACIONES

        RCALL HABILITAR_TIMER0_COMP
        RCALL DLY
        RCALL HABILITAR_TIMER2_COMP
        RCALL DLY
        RCALL HABILITAR_TIMER1_COMP

        SEI
LAZO_PRINCIPAL:

        RCALL CICLO_INFINITO
```

```
RJMP LAZO_PRINCIPAL



//-----------------------------------------------------
//CONFIGURACION DE PUERTOS
//PORTB COMO ENTRADA
//PORTE COMO ENTREDA EXPTO TX QUE ES SALIDA.
CONFIGURACION_PUERTOS:


        LDI REG_TEMP,0B01110000     //CONFIGURACION DE PUERTO B
        LDI TEMP,       0B10101111
        OUT PORTB,REG_TEMP
        OUT DDRB,TEMP
        NOP

        LDI REG_TEMP,0B00001111 //CONFIGURACION PUERTO E
        LDI TEMP,       0B11110011
        OUT PORTE,REG_TEMP
        OUT DDRE,TEMP
        NOP



        LDI REG_TEMP,0B00000000 //CONFIGURACION PUERTO D
        LDI TEMP,       0B11111111 //PORTD,7 INDICADOR DE TRABAJO
        OUT PORTD,REG_TEMP
        OUT DDRD,TEMP
        NOP

RET //FIN CONFIGURACION DE PUERTOS
//=======================================================




;HABILITACION DE REGISTROS PARA FUNCIONAMIENTO DEL TIMERO
HABILITAR_TIMER0_COMP:
        //TCCR0A - TIMER COUNTER CONTROL REGISTER A
        //FOC0A                                     0                   NO USADO
        //WGM00         WGM01                       1  1                FAST PWM
        //COM01A COM0A0                     0  0                DESCONECTADO OC0A
        //CS02   CS01    CS00   0  1  0             CLK/8
```

```
        //
        //ORDEN(FOC0A, WGM00, COM01A, COM0A0, WGM01, CS02,CS01,CS00)
        //
        LDI REG_TEMP,0B01001100     ;0B01001010 ;GUARDAMOS CONFIGURACION DE
MODO. Y PREESCALADOR
        OUT TCCR0A,REG_TEMP

        LDS REG_TEMP,TIMSK0                     ;GUARDAMOS EL ESTADO
ANTERIOR DE TIMSK0
        SBR REG_TEMP,(1<<OCIE0A) + (1<<TOIE0);HABILITAMOS LA INTERRUPCION
0X00014

        STS TIMSK0,REG_TEMP         ;CARGAMOS EL VALOR CONFIGURADO EN
TIMSK0

        LDI REG_TEMP,0x0A    ;CARGAMOS EL VALOR MAXIMO AL QUE LLEGARA EL
TIMER2
        STS OCR0A,REG_TEMP

RET


//HABILITACION DEL TIMER1 MODO FAST PWM
HABILITAR_TIMER1_COMP:
//PARA FAST PWM
LDI REG_TEMP,0B00000001
STS TCCR1A,REG_TEMP

LDI REG_TEMP,0B00001100
STS TCCR1B,REG_TEMP

//HABILITACION DE INTERRUPCIONES COMP Y OVF
LDS REG_TEMP,TIMSK1                 ;GUARDAMOS EL ESTADO ANTERIOR DE TIMSK0
SBR REG_TEMP,(1<<OCIE1A) + (1<<TOIE1) + (1<<OCIE1B);HABILITAMOS LA
INTERRUPCION 0X00014
STS TIMSK1,REG_TEMP                 ;CARGAMOS EL VALOR CONFIGURADO EN
TIMSK2



LDI REG_TEMP,0X00
STS OCR1AH,REG_TEMP

LDI REG_TEMP,0x0A
STS OCR1AL,REG_TEMP
```

```
LDI REG_TEMP,0X00
STS OCR1BH,REG_TEMP

LDI REG_TEMP,0xA
STS OCR1BL,REG_TEMP

RET


;HABILITACION DE REGISTROS PARA UTILIZAR LA INTERRUPCION TIMER2 COMPARACION
HABILITAR_TIMER2_COMP:
        //TCCR2A - TIMER COUNTER CONTROL REGISTER A
        //FOC2A                                 0                   NO USADO
        //WGM20         WGM21                   1  1                FAST PWM
        //COM21A COM2A0                 0  0                DESCONECTADO OC0A
        //CS22   CS21    CS20   0  1  0         CLK/8
        //
        //ORDEN(FOC2A, WGM20, COM21A, COM2A0, WGM21, CS22,CS21,CS20)
        //

        LDI REG_TEMP,0B01101001 ;GUARDAMOS CONFIGURACION DE MODO. Y PREESCALADOR
        STS TCCR2A,REG_TEMP

        LDS REG_TEMP,TIMSK2             ;GUARDAMOS EL ESTADO ANTERIOR DE TIMSK2
        SBR REG_TEMP,(1<<OCIE2A) + (1<<TOIE2);HABILITAMOS LA INTERRUPCION 0X0008
        STS TIMSK2,REG_TEMP             ;CARGAMOS EL VALOR CONFIGURADO EN TIMSK2

        LDI REG_TEMP,0x0A ;CARGAMOS EL VALOR MAXIMO AL QUE LLEGARA EL TIMER2
        STS OCR2A,REG_TEMP

RET




;HABILITACION DE REGISTROS PARA UTILIZAR LA INTERRUPCION PCINT0
HABILITAR_PCINT0:
```

```
        //habilitamos interrupcion PCIE0
        IN      REG_TEMP,EIMSK          ;GUARDAMOS EL ESTADO ANTERIOR DE
GIMSK,
        SBR REG_TEMP,(1<<6)         ;COLOCAMOS UN ALTO EN EL BIT6(PCIE0), NO SE
ALTERA EL RESTO DE BITS
        OUT EIMSK,REG_TEMP          ;ACTUALIZAMOS EL REGSITRO GIMSK
CON EL NUEVO VALOR

        LDS     REG_TEMP,PCMSK0         ;GUARDAMOS EL ESTADO ANTERIOR DE
PCMSK0,
        SBR REG_TEMP,(1<<PCINT2) + (1<<PCINT3);COLOCAMOS UN ALTO EN EL BIT6,
NO SE ALTERA EL RESTO DE BITS
        STS PCMSK0,REG_TEMP         ;ACTUALIZAMOS EL REGSITRO PCMSK0
CON EL NUEVO VALOR


RET



;HABILITACION DE REGISTROS PARA UTILIZAR LA INTERRUPCION PCINT1
HABILITAR_PCINT1:
        //habilitamos interrupcion PCIE1
        IN      REG_TEMP,EIMSK          ;GUARDAMOS EL ESTADO ANTERIOR DE
GIMSK,
        SBR REG_TEMP,(1<<7) ;COLOCAMOS UN ALTO EN EL BIT7 (PCIE1), NO SE
ALTERA EL RESTO DE BITS
        OUT EIMSK,REG_TEMP          ;ACTUALIZAMOS EL REGSITRO GIMSK
CON EL NUEVO VALOR

        LDS     REG_TEMP,PCMSK1         ;GUARDAMOS EL ESTADO ANTERIOR DE
PCMSK0,
        SBR REG_TEMP,(1<<PCINT12) + (1<<PCINT14) + (1<<PCINT15); + (
1<<PCINT15);COLOCAMOS UN ALTO EN EL BIT6, NO SE ALTERA EL RESTO DE BITS
BOTON B NO FUNCIONARA SERA PWM
        STS PCMSK1,REG_TEMP         ;ACTUALIZAMOS EL REGSITRO PCMSK0
CON EL NUEVO VALOR


RET



//SE ENCANGARA DE INCREMENTAR LOS VALORES
//PARA MODIFICAR EL PWM DE CADA TIMER
CICLO_INFINITO:
```

```
        INC             VERDE
        INC             VERDE
        INC     VERDE
        STS             OCR2A,VERDE

        INC             AZUL
        INC     AZUL
        OUT             OCR0A,AZUL

        INC             ROJO
        STS             OCR1AL,ROJO

        RCALL DLY

RET
//-----------------------------------


        ;RUTINA DELAY

        DLY:
        dec             Delay
        brne    DLY
        dec             Delay2

        brne    DLY
        ret


//-------------------------------------------------------
//COLOR AZUL
INT_TIMER0_COMP:
        CLI
                CBI PORTB,0
        SEI
RETI


INT_TIMER0_OVF:
CLI
```

```
                SBI     PORTB,0
SEI

RETI

////////////////////////////////////////////////////

//-----------------------------------------------
//COLOR ROJO
INT_TIMER1_COMPA:
        CLI
                CBI PORTB,1

        SEI
RETI


INT_TIMER1_OVFA:
CLI
                SBI     PORTB,1

SEI

RETI

////////////////////////////////////////////////////

//-----------------------------------------------
//COLOR ROJO
INT_TIMER1_COMPB:
        CLI


        SEI
RETI
////////////////////////////////////////////////////

//-----------------------------------------------
//COLOR VERDE
```

```
INT_TIMER2_COMP:
        //CLI
                //CBI PORTB,2

        //SEI
RETI


INT_TIMER2_OVF:
//CLI
                //SBI     PORTB,2

//SEI

RETI
```

3.3. PROYECTO3: CERRADURA ELECTRONICA

3.3.1 OBJETIVO.

Utilización de interrupción por cambio en los Pórticos para el ingreso de números.

Ingreso de datos en Memoria SRAM.

Se debe ingresar 5 valores por medio del teclado numérico, al coincidir con la clave guardada previamente en la Memoria EEPROM del AvrButterfly deberá activarse el Led que representa la apertura de la cerradura Electrónica.

3.3.2 ENUNCIADO

Elaborar en lenguaje C un programa que sea para apertura de una puerta electrónica, al ingresar la clave correcta se activara un led que simula la apertura de la puerta. Se debe implementar tres botones, correspondientes a los números 1,2,3.

3.3.3 ESQUEMA DEL PROYECTO.

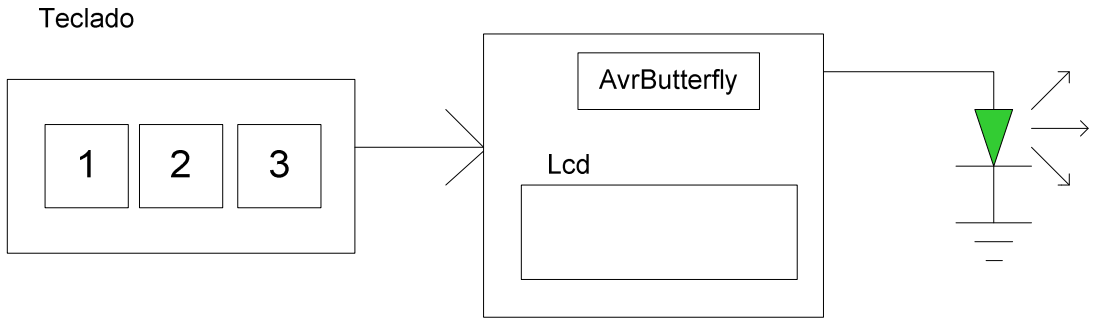

Figura 3-12: diagrama de bloque de Cerradura Electrónica.

3.3.4 DIAGRAMA DE FLUJO DEL PROYECTO.

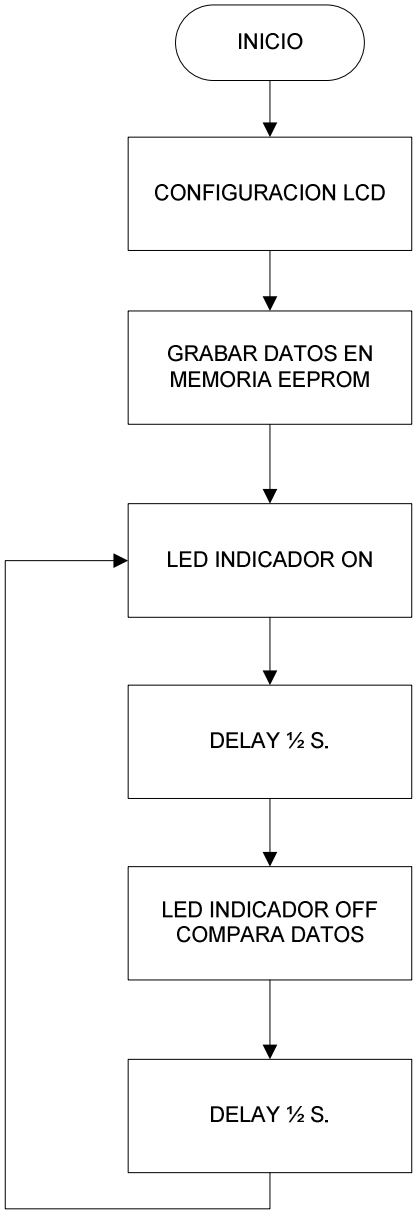

Figura 3-13: diagrama de flujo principal

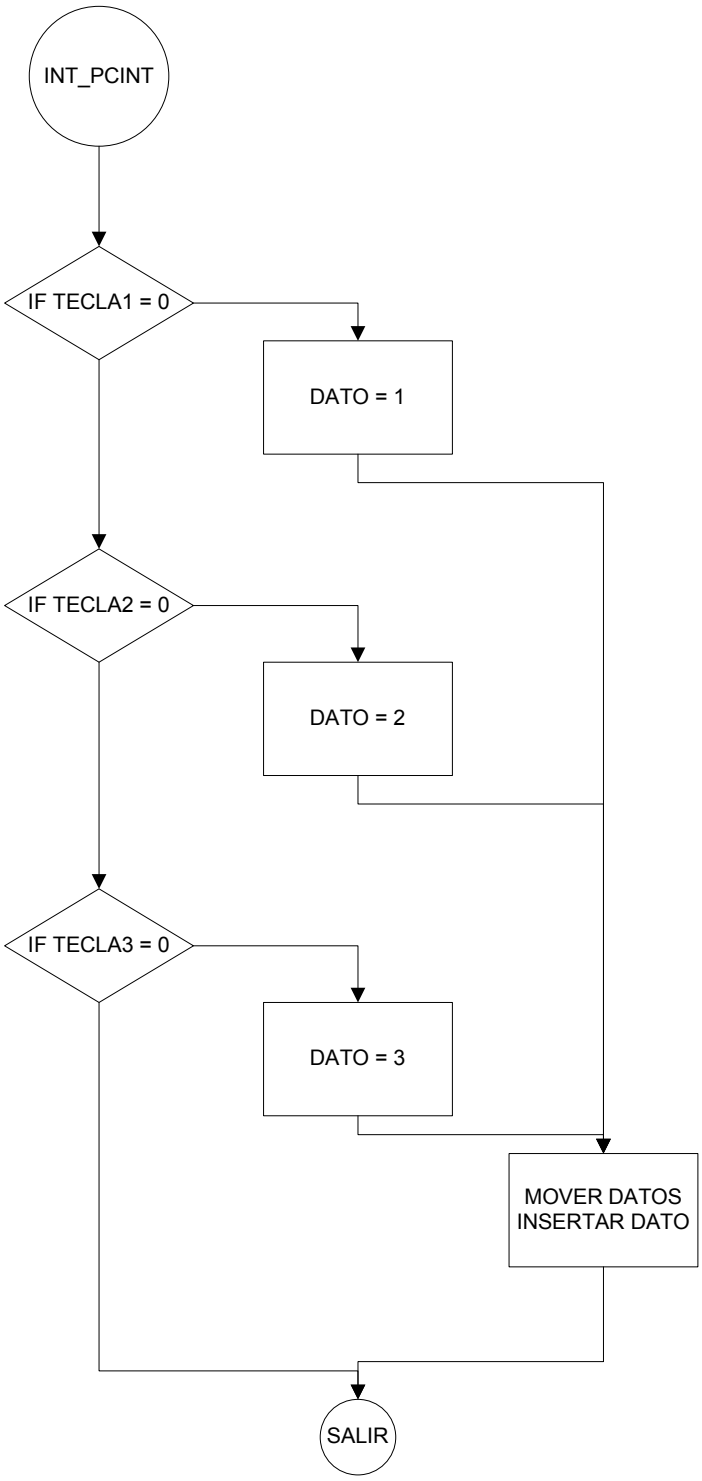

Figura 3-14: diagrama de flujo de Interrupción.

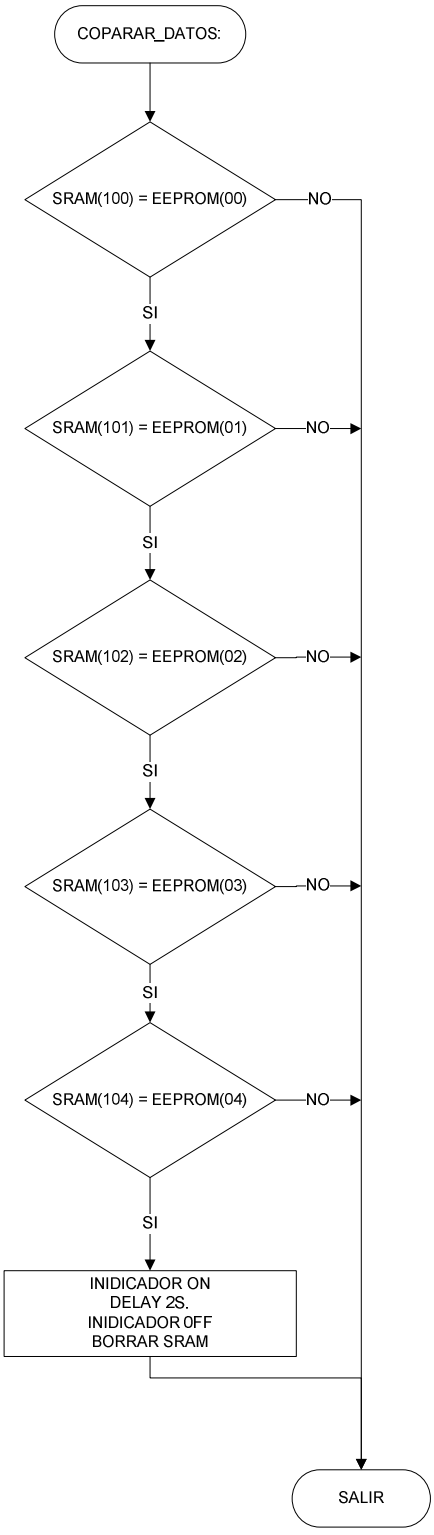

Figura 3-15: diagrama de rutina Comparar datos.

3.3.5 CÓDIGO FUENTE.

```
/*
 * TERCERO.c
 *
 * Created: 04/01/2012 15:08:57
 *  Author: Micro
 */

#include <avr/io.h>

#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <avr/sleep.h>
#include <inttypes.h>
#include <avr/eeprom.h>
#include <string.h>

#define BOOL    char

#define FALSE   0
#define TRUE    (!FALSE)
#define NULL    0

#define AUTO    3

#define INDICADOR           3

#include "lcd_functions.h"
#include "lcd_driver.h"

#define pLCDREG_test (*(char *)(0xEC))

void GRABAR_DATOS(void);
void CONFIGURAR_PUERTOS(void);

void GUARDAR_DATOS_EN_EEPROM(void);
void HABILITAR_PCINT(void);
int COMPARA_CLAVE(void);
void INGRESAR_DATO_EN_ARRAY(char);
void LED_INDICADOR_FLASH(void);


//ALMACENAMOS CLAVE
char CLAVE_ORIGINAL[] = "12312";
//ARRAY PARA ALMACENAR CLAVE A INGRESAR
char CLAVE_INGRESADA[] = "     ";
char REG_TEMP;
uint8_t INDICE;

//MENSAJE INICIAL
PGM_P MENSAJE1;
PGM_P MENSAJE2;
PGM_P MENSAJE3;

int main(void)
{
```

```
		CONFIGURAR_PUERTOS();
		LCD_Init();
		HABILITAR_PCINT();
	 sei();

	LCD_puts_f(PSTR("INICIO"),1);


	INDICE = 0X00;
	for (;;) {
				if(COMPARA_CLAVE()){
					//CLAVE INGRESADA ES CORRECTA
					LCD_puts_f(PSTR("OK"),1);
					 PORTE = (1<<7); //APERTURA DE PUERTA
					 _delay_ms(800);
					 _delay_ms(800);
					 PORTE = (0<<7); //SE CIERRA EL CIRCUITO.
					 //BORRAMOS CALVE INGRESADA
					 CLAVE_INGRESADA[0] = ' ';
					 CLAVE_INGRESADA[1] = ' ';
					 CLAVE_INGRESADA[2] = ' ';
					 CLAVE_INGRESADA[3] = ' ';
					 CLAVE_INGRESADA[4] = ' ';
					  LCD_puts_f(PSTR("INICIO"),1);

					 }

			//PRENDE Y APAGA EL LED INDICADOR DE FUNCIONAMIENTO ESTABLE
			 LED_INDICADOR_FLASH();


	}
	return 0;
}

void LED_INDICADOR_FLASH(void)
{
		PORTD = PINB;
		  PORTB = (1<<INDICADOR);
		  _delay_ms(23);
		  PORTB = (0<<INDICADOR);
		  _delay_ms(23);
}


void GRABAR_DATOS(void)
{
		//COMPROBAMOS QUE TECLA SE PRESIONO.

		char SELECCION;
		//VERIFICA SI SE PRECIONO TECLA 1
		SELECCION = (~PINB & 0X01);
		if(SELECCION == 0x01)
			{
				PORTE = (1<<4);
				INGRESAR_DATO_EN_ARRAY('1');
```

```
                }

        //VERIFICA SI SE PRESIONO TECLA 2
        SELECCION = (~PINB & 0X02);
        if(SELECCION == 0x02)
                {
                        PORTE = (1<<5);
                        INGRESAR_DATO_EN_ARRAY('2');


                }

        //VERIFICA SI SE PRESIONO TECLA 3
        SELECCION = (~PINB & 0X04);
        if(SELECCION == 0x04)
                {
                        PORTE = (1<<6);
                        INGRESAR_DATO_EN_ARRAY('3');

                }


        //VERIFICA SI SE PRESIONO TECLA CENTRAL
        SELECCION = (~PINB & 0X10);
        if(SELECCION == 0x10)
                {
                        //NINGUNA ACCION
                  }
}

//PUERTOS B0,B1,B2 COMO ENTRADA PARA TECLAS.
void CONFIGURAR_PUERTOS(void)
{
        DDRD  = 0XFF;
        DDRE = 0XF3;
        DDRB = 0X08;
        PORTB = 0XF7;

}

//ALMACENA LOS DATOS EN LA MEMORIA EEPROM PARA TENERLOS
//DE RESPALDO
void GUARDAR_DATOS_EN_EEPROM()
{

eeprom_write_byte (0X00, CLAVE_ORIGINAL[0]);
eeprom_write_byte (0X01, CLAVE_ORIGINAL[1]);
eeprom_write_byte (0X02, CLAVE_ORIGINAL[2]);
eeprom_write_byte (0X03, CLAVE_ORIGINAL[3]);
eeprom_write_byte (0X04, CLAVE_ORIGINAL[4]);

}

//COMPARA CLAVE ALMACENADA CON CLAVE INGRESADA
int COMPARA_CLAVE(void)
{
                if(strcmp(CLAVE_ORIGINAL,CLAVE_INGRESADA)==0)
                return 1;
```

```
                return 0;

}

//CONFIGURACION DE INTERRUPCIONES, TAMBIEN SE CONFIGURAN LAS BOTONERAS
//DE PORTB0, PORB1, PORB2
void HABILITAR_PCINT(void){
        //HABILITACION DE INTERRUPCONES POR CAMBIO EN PORTICOS
        EIMSK |= (1<<PCIE0) | (1<<PCIE1);

        //HABILITACION INDIVIDUAL DE INTERRUPCION EN CADA BITS.
        PCMSK0 |= (1<<PCINT2) | (1<<PCINT3);        //IZQUIERDA, DERECHA

        //TECLA1, TECLA2, TECLA3, CENTRO, ARRIBA, ABAJO
        PCMSK1  |=  (1<<PCINT8)  |  (1<<PCINT9)  |  (1<<PCINT10)  |  (1<<PCINT12)  |  (1<<PCINT14)  |
(1<<PCINT15);

}

//INTERRUPCION POR CAMBIO EN LOS PORTICOS PB0
SIGNAL(SIG_PIN_CHANGE0)
{

        PORTE = 0x01;//INDICADOR
        //REDIRECIONAMOS
        GRABAR_DATOS();

}

//INTERRUPCION POR CAMBIO EN LOS PORTICOS PB0
SIGNAL(SIG_PIN_CHANGE1)
{

        PORTE = 0x02;  //INDICADOR
        GRABAR_DATOS();

}


//INGRESA DATO EN EL ARREGLO DE CLAVES
//ANTES MUEVE LOS DATOS UNA POSISION
void INGRESAR_DATO_EN_ARRAY(char temp_i)
{
                if(INDICE == 0X00)LCD_puts_f(PSTR("X    "),1);
                if(INDICE == 0X01)LCD_puts_f(PSTR("XX   "),1);
                if(INDICE == 0X02)LCD_puts_f(PSTR("XXX  "),1);
                if(INDICE == 0X03)LCD_puts_f(PSTR("XXXX "),1);
                if(INDICE == 0X04)LCD_puts_f(PSTR("XXXXX"),1);

                CLAVE_INGRESADA[0]=CLAVE_INGRESADA[1];
                CLAVE_INGRESADA[1]=CLAVE_INGRESADA[2];
                CLAVE_INGRESADA[2]=CLAVE_INGRESADA[3];
                CLAVE_INGRESADA[3]=CLAVE_INGRESADA[4];
                CLAVE_INGRESADA[4]=temp_i;
                INDICE++;

                if(INDICE == 0X05) INDICE = 0X00;
}
```

3.4. PROYECTO4: CONVERTIDOR ANALOGICO DIGITAL

3.4.1 OBJETIVO.-

Configurar el convertidor analógico digital.

Realizar mediciones mediante el sensor TNC intergrado del AvrButterfly.

Activar un led, que representa el Encendido de un Ventilador.

3.4.2 ENUNCIADO

Elaborar un programa en lenguaje C, que active un ventilador cuando la lectura del ADC0 ha alcanzado el valor Máximo decimal de 200. Y apagar el ventilador cuando el ADC0 ha alcanzado el valor mínimo decimal de 100, utilizar el NTC integrado del AvrButterfly. Ver grafica 3-19.

3.4.3 ESQUEMA DEL PROYECTO.

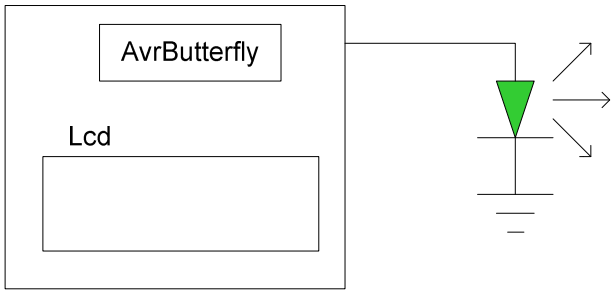

Figura 3-16: Esquema de Control de temperatura.

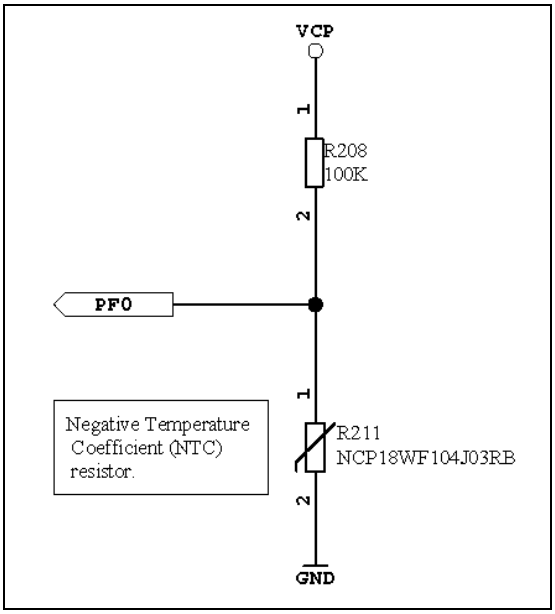

Figura 3-17: Diagrama Esquemático de sensor de temperatura interno del AvrButterfly.

El AvrButterfly tienen integrado un sensor NTC, en serie con una resistencia de 100k, esta está conectado al Puerto F, pin 0. El comportamiento de este sensor es de comportamiento logarítmico, lo implica un largo proceso de cálculo para obtener valores de temperatura exactos. Sin embargo mediantes métodos de cálculo se puede llegar a una aproximación muy aceptable. Sn embargo el proceso de señales no está dentro de este estudio, aquí se obtiene una conversión analógica a digital y las trabajaremos en forma binaria, es decir no se hará la conversión del valor binario al medidas de temperatura. Existe un valor Máximo, en el cual se prendera nuestro ventilador, y un valor mínimo en que se apagara.

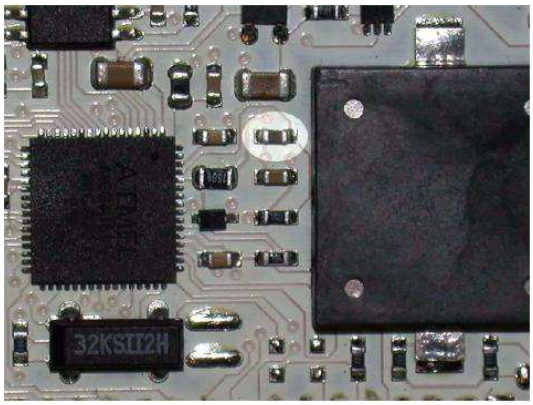

Figura 3-18: ubicación del sensor NTC en el AvrButterfly.

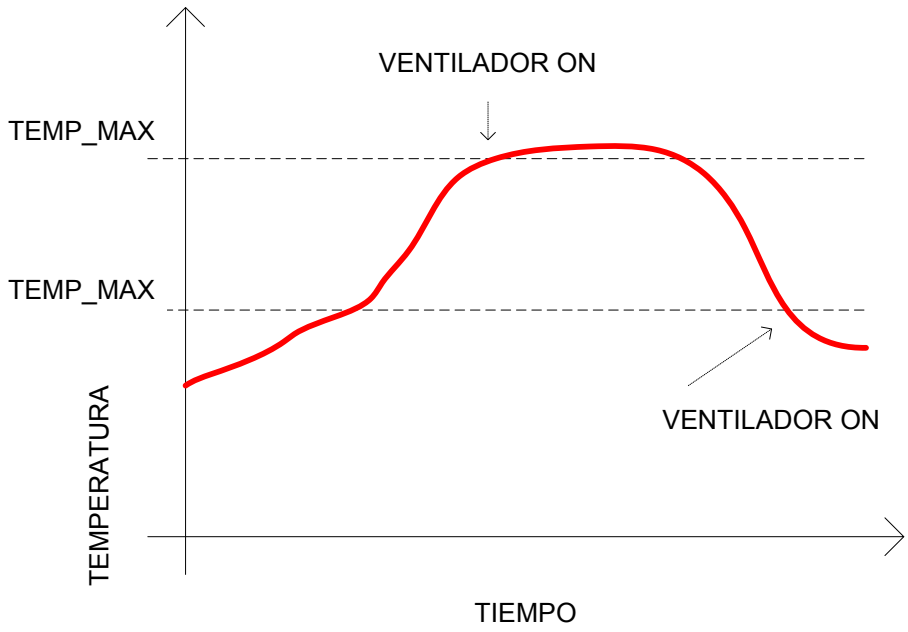

Figura 3-19: Esquema de Encendido- apagado del Ventilador con relación al cambio de Temperatura.

3.4.4 DIAGRAMA DE FLUJO DEL PROYECTO.

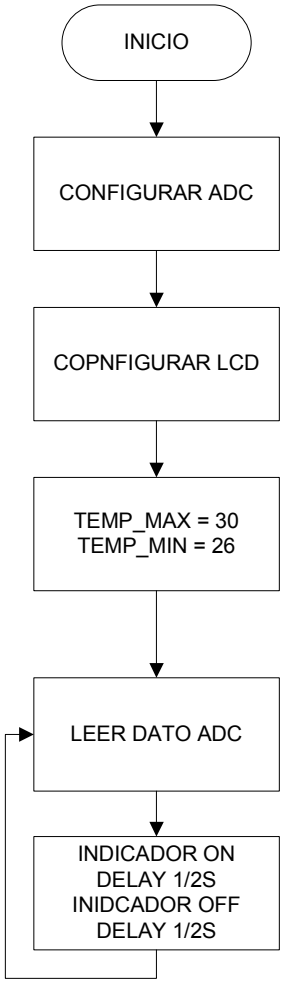

Figura 3-20: diagrama de flujo principal de Control de Temperatura

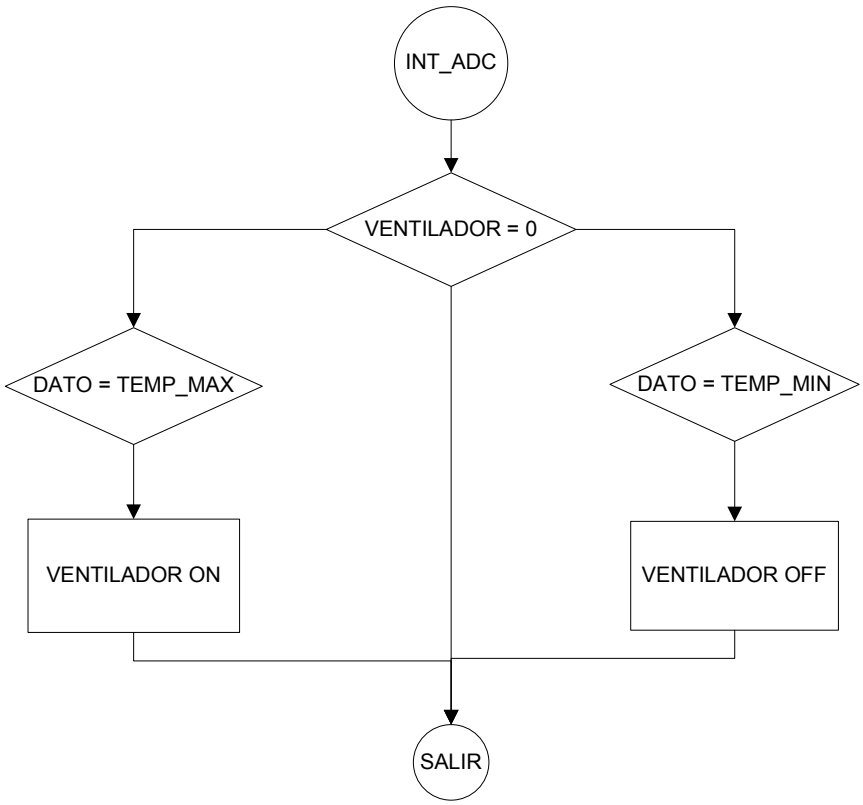

Figura 3-21: Diagrama de Flujo de Interrupción ADC

3.4.5 CÓDIGO FUENTE.

```
//*****************************************************************************
//
// File Name  : 'CUARTO.c'
// Title            : CONTROL DE VENTILADOR POR MEDIO DE LECTURA DEL ADC0
//
//*****************************************************************************

//                  4K-400
// +5---/\/\/\----+---\/\/\/\/\---gnd
//      rref      |      NTC
//
```

```
#include <stdio.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/sleep.h>
#include <util/delay.h>
#include <string.h>

#define BOOL    char
#define FALSE   0
#define TRUE    (!FALSE)
#define NULL    0
#define AUTO    3
#define INDICADOR   3

#include "lcd_functions.h"
#include "lcd_driver.h"

#define pLCDREG_test (*(char *)(0xEC))

//-----------------------

#define MAXIMO_VALOR 200
#define MINIMO_VALOR 100

void delay1s(void);
void adc_init(void);
void adc_start_conversion(uint8_t);
void init(void);

uint8_t ch;

char DATO;
char VENTILADOR;


//*************************************************************************
//
//  iNICIALIZACION DEL CONVERTIDOR
//
//*************************************************************************
void adc_init(void)
{
//SELECION DEL VOLTAJE DE REFERENCIA
//AVCC CON CAPACITOR EXTERNO AREF pin
ADMUX|=(0<<REFS1)|(1<<REFS0);
//set prescaller  ADC
ADCSRA|=(1<<ADEN)|(1<<ADIE);//enable ADC with dummy conversion
//set sleep mode  ADC noise reduction conversion
//set_sleep_mode(SLEEP_MODE_ADC);
}
//*************************************************************************
//
//  RUTINA PARA UNA SIMPLE CONVERSION
```

```
//
//*****************************************************************************
void adc_start_conversion(uint8_t channel)
{
//CANAL DE CONVERSION REALMENTE SOLO USAMOS EL CANAL 0 CORRESPONDIENTE A PF0;
ch=channel;
//set ADC channel
ADMUX=(ADMUX&0xF0)|channel;
//START CONVERSION CON INTERRUPCION HABILITADA AL FINALIZAR CONVERSION
//HABILITA INTERRUPCION GLOBAL
sei();
ADCSRA |= (1<<ADSC)|(1<<ADIE);
}

//*****************************************************************************
//
//  RETARDO DE 1s
//
//*****************************************************************************

//RETARDO 1s
void delay1s(void)
{
	uint8_t i;
	for(i=0;i<100;i++)
	{
		_delay_ms(10);
	}
}

//*****************************************************************************
//
//  init AVR
//
//*****************************************************************************

void init(void)
{

	adc_init();
}

//*****************************************************************************
//
//  ADC conversion complete service routine
//
//*****************************************************************************
ISR(ADC_vect)
{
	// SE HA FINALIZADO LA LECTURA DEL DATO
	uint16_t adc_value;
	adc_value = ADCW;
	//ALMACENA EL DATO DE LA CONVERSION

	if(VENTILADOR == 0)
```

```
	{
			if(adc_value > MAXIMO_VALOR)
			{
				VENTILADOR = 1;
				PORTE = PINE | (1<<7);
				LCD_puts_f(PSTR("ON    "),1);
			}

	}
	else
	{
			if(adc_value < MINIMO_VALOR)
			{
				VENTILADOR = 0;
				PORTE = PINE & (0<<7);
				LCD_puts_f(PSTR("OFF   "),1);
			}

	}


}

//*****************************************************************************
//
//  PROGRAMA PRINCIPAL.
//
//*****************************************************************************
int main(void)
{
	init();
	LCD_Init();
	VENTILADOR = 0X00;

	LCD_puts_f(PSTR("TEMPERATURA"),1);

	DDRE = 0XFF;

	delay1s();
	while(1)//LAZO INFINITO
	{
	//read LDR
	//adc_start_conversion(0);

	//_delay_ms(30);
	//read potentiometer
	adc_start_conversion(0);
	_delay_ms(30);
	//continue infinitely
	}
	return 0;
}
```

3.5. PROYECTO5: JUEGO DE DADOS SIMULADO EN LCD

3.5.1 OBJETIVO.-

Utilizar las interrupciones por cambio en los pórticos para habilitación de botoneras externas.

Utilización de librería LCD, para mostrar valores aleatorios.

3.5.2 ENUNCIADO

El circuito debe tener dos botoneras externas, para cada jugador, una tercera botonera que será la que da inicio al juego, además, el circuito constara de dos Led, estos indican el turno de cada jugador.

Cuando el jugador uno presiona el botón en su turno se genera un valor aleatorio y este es mostrado en el lcd.

Luego de este se dará paso para que se presione el botón del jugador 2.

Luego de esto será nuevamente el turno del jugador 1, y luego del jugador 2.

Ganara el que al sumar los valores de los dos dados obtenga el mayor puntaje.

3.5.3 ESQUEMA DEL PROYECTO.

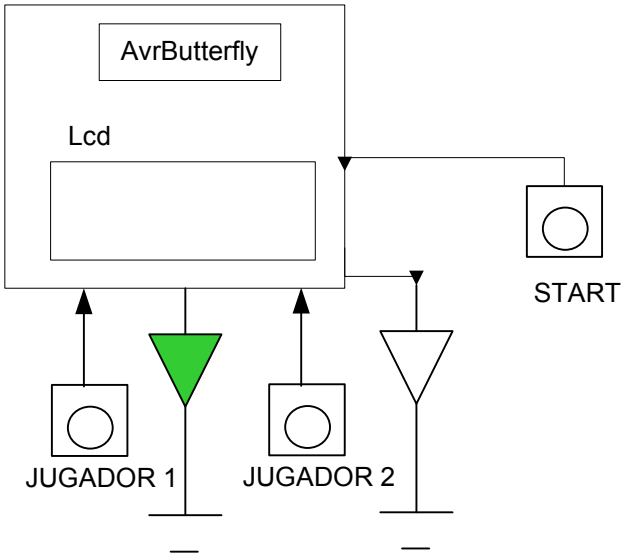

Figura 3-22 Esquema del Proyecto

3.5.4 DIAGRAMA DE FLUJO DEL PROYECTO.

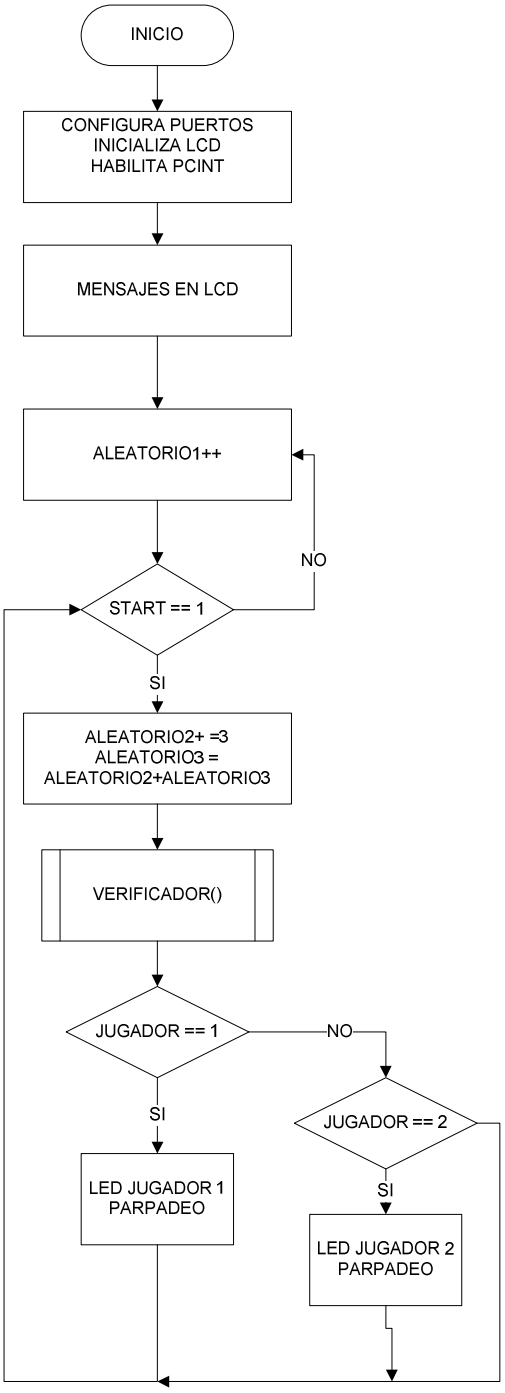

Figura 3.23: diagrama de flujo principal proyecto 5

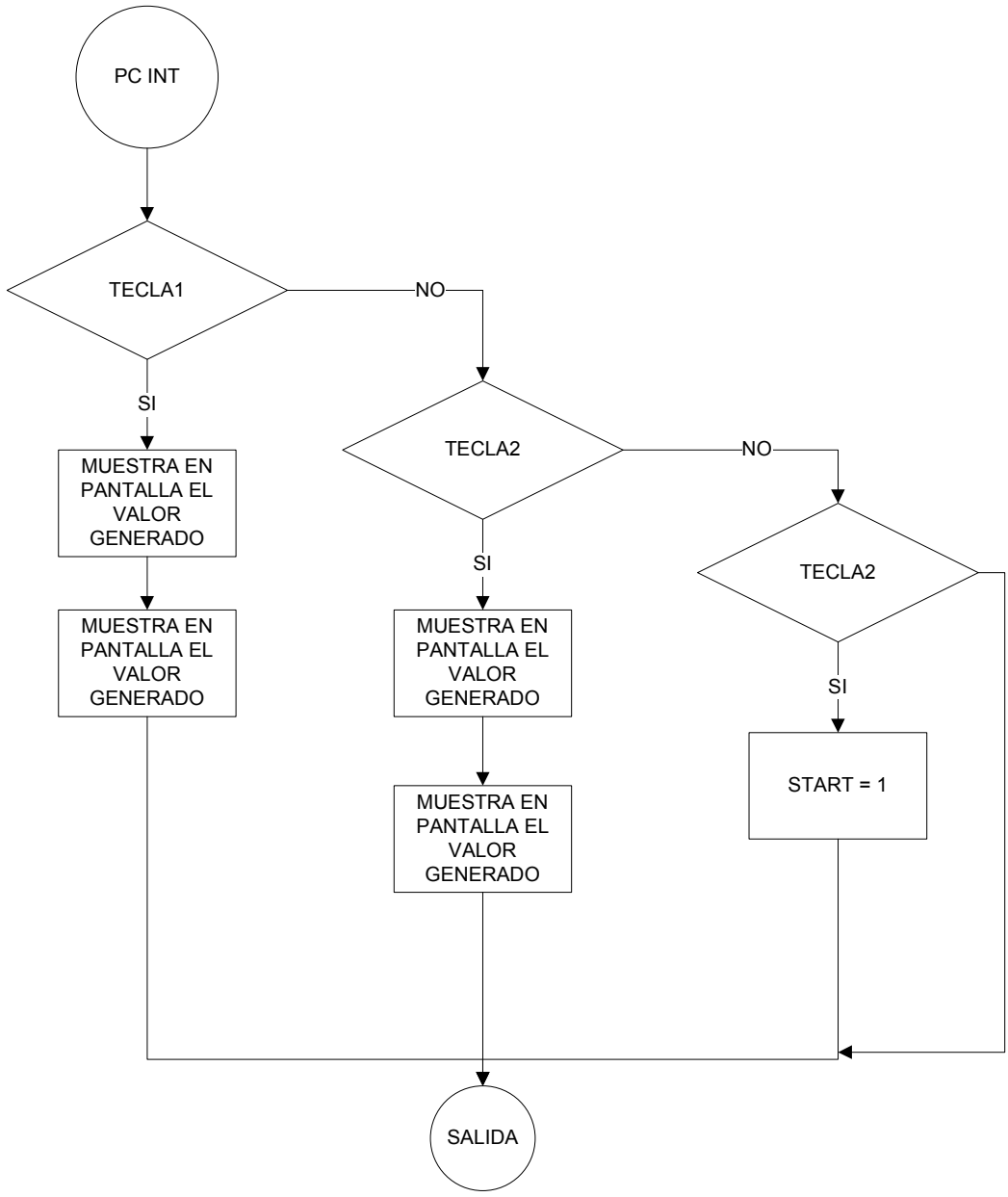

Figura 3.24: Diagrama de flujo de Interrupción PCINT DEL PROYECTO 5

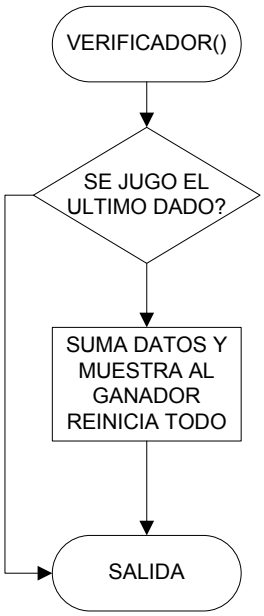

Figura 3.25: diagrama de flujo de subrutina VERIFICADOR() proyecto 5.

3.5.5 CÓDIGO FUENTE.

```
/*
 * QUINTO.c
 *
 * Created: 04/01/2012 15:08:57
 *  Author: Micro
 */

#include <avr/io.h>

#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <avr/sleep.h>
#include <inttypes.h>
#include <avr/eeprom.h>
#include <string.h>


#define BOOL    char

#define FALSE   0
#define TRUE    (!FALSE)
#define NULL    0
```

```
#define AUTO    3

#define INDICADOR    3
#define CERRADURA    7

#include "lcd_functions.h"
#include "lcd_driver.h"

#define pLCDREG_test (*(char *)(0xEC))

void GRABAR_DATOS(void);
void CONFIGURAR_PUERTOS(void);

void HABILITAR_PCINT(void);


void LED_JUGADOR1(void);
void LED_JUGADOR2(void);

void VERIFICADOR(void);


char REG_TEMP;
uint8_t INDICE;

char JUG1_DADO1=' ';
char JUG1_DADO2=' ';

char JUG2_DADO1=' ';
char JUG2_DADO2=' ';

char ALEATORIO1;
char ALEATORIO2;
char ALEATORIO3;

char JUGADOR=1; //SI ES JUGADOR 1 ES UNO..
char START;


int main(void)
{


       CONFIGURAR_PUERTOS();
       LCD_Init();
       HABILITAR_PCINT();
    sei();

   LCD_puts_f(PSTR("INICIO"),1);
   _delay_ms(800);
   LCD_puts_f(PSTR("JUEGO "),1);
      _delay_ms(800);
   LCD_puts_f(PSTR("  DE  "),1);
      _delay_ms(800);
```

```
   LCD_puts_f(PSTR("DADOS "),1);
              _delay_ms(800);

  JUGADOR = 1;
   INDICE = 0X00;
   START = 0;

   //LAZO INFINITO
   for (;;) {


       ALEATORIO1++;

       while(START == 1)
       {


       ALEATORIO2 = ALEATORIO2 + 3;
       ALEATORIO3 = ALEATORIO2 + ALEATORIO1;


       VERIFICADOR();


       if(JUGADOR == 1)
       {
              //TURNO DEL JUGADOR UNO
              //PRENDER LED DEL JUGADOR UNO
              LED_JUGADOR1();

       }
       if(JUGADOR == 2)
       {
              //TURNO DEL JUGADOR DOS
              //PRENSER LED DEL JUGADOR 2
              LED_JUGADOR2();

       }

       }

       LCD_puts_f(PSTR("START "),1);
       LCD_FlashReset();

        //PRENDE Y APAGA EL LED INDICADOR DE FUNCIONAMIENTO ESTABLE
       //  LED_INDICADOR_FLASH();


   }


   return 0;
}
```

```
void VERIFICADOR(void){

	if (JUG2_DADO2 != ' ')
	{
		_delay_ms(800);


		if( (JUG1_DADO1 + JUG1_DADO2 ) == (JUG2_DADO1 + JUG2_DADO2 ) )
		LCD_puts_f(PSTR("EMPATE"),1);
		else if((JUG1_DADO1 + JUG1_DADO2 ) > (JUG2_DADO1 + JUG2_DADO2 )
)
					LCD_puts_f(PSTR("GANA 1"),1);
					else LCD_puts_f(PSTR("GANA 2"),1);

		LCD_FlashReset();

			JUG1_DADO1=' ';
			JUG1_DADO2=' ';

			JUG2_DADO1=' ';
			JUG2_DADO2=' ';
			START = 0;
		_delay_ms(800);
		_delay_ms(800);
		_delay_ms(800);

	}

}


void LED_JUGADOR1(void)
{
	PORTD = PINB;
	   PORTB = (1<<INDICADOR);
	   _delay_ms(23);
	   PORTB = (0<<INDICADOR);
	   _delay_ms(23);
}

void LED_JUGADOR2(void)
{
	PORTD = PINB;
	   PORTE = (1<<CERRADURA);
	   _delay_ms(23);
	   PORTE = (0<<CERRADURA);
	   _delay_ms(23);
}




void GRABAR_DATOS(void)
{
```

```
//COMPROBAMOS QUE TECLA SE PRESIONO.

char SELECCION;
ALEATORIO3 = (ALEATORIO3 & 0X07);
if(ALEATORIO3 == 0 ) ALEATORIO3 = 1;
if(ALEATORIO3 == 7 ) ALEATORIO3 = 6;


//VERIFICA SI SE PRECIONO TECLA 1
SELECCION = (~PINB & 0X01);
if(SELECCION == 0x01 && START == 1)
	{
		if(JUGADOR == 1)
		{
			//JUGADOR UNO PRESIONO EN SU TURNO
			if(JUG1_DADO1==' '){

				LCD_puts_f(PSTR("	"),1);
				LCD_FlashReset();

				JUG1_DADO1 = ALEATORIO3;
				LCD_putc(0, JUG1_DADO1 + 49);
			}
			else
			if(JUG1_DADO2==' '){
				JUG1_DADO2 = ALEATORIO3;
				LCD_putc(1, JUG1_DADO2 + 49);

			}



			LCD_FlashReset();

			JUGADOR = 2;


		}
		//PORTE = (1<<4);
		//INGRESAR_DATO_EN_ARRAY('1');

	}


//VERIFICA SI SE PRESIONO TECLA 2
SELECCION = (~PINB & 0X02);
if(SELECCION == 0x02 && START == 1)
	{
		if(JUGADOR == 2)
		{
			//JUGADOR DOS PRESIONO EN SU TURNO
			if(JUG2_DADO1==' '){
				JUG2_DADO1 = ALEATORIO3;
				LCD_putc(4, JUG2_DADO1 + 49);
			}
```

```
                        else
                                if(JUG2_DADO2==' '){
                                        JUG2_DADO2 = ALEATORIO3;
                                        LCD_putc(5, JUG2_DADO2 + 49);
                                        LCD_FlashReset();}
                }
                                JUGADOR = 1;


                }
        //VERIFICA SI SE PRESIONO START ( TECLA3)
        SELECCION = (~PINB & 0X04);
        if(SELECCION == 0x04)
                {

                        START = 0X01;

                }


        //VERIFICA SI SE PRESIONO TECLA CENTRAL
        SELECCION = (~PINB & 0X10);
        if(SELECCION == 0x10)
                {
                        //NINGUNA ACCION

           }
}

//PUERTOS B0,B1,B2 COMO ENTRADA PARA TECLAS.
void CONFIGURAR_PUERTOS(void)
{
        DDRD  = 0XFF;
        DDRE = 0XF3;
        DDRB = 0X08;
        PORTB = 0XF7;

}




//CONFIGURACION DE INTERRUPCIONES, TAMBIEN SE CONFIGURAN LAS BOTONERAS
//DE PORTB0, PORB1, PORB2
void HABILITAR_PCINT(void){
        //HABILITACION DE INTERRUPCONES POR CAMBIO EN PORTICOS
        EIMSK |= (1<<PCIE0) | (1<<PCIE1);

        //HABILITACION INDIVIDUAL DE INTERRUPCION EN CADA BITS.
        PCMSK0 |= (1<<PCINT2) | (1<<PCINT3);     //IZQUIERDA, DERECHA

        //TECLA1, TECLA2, TECLA3, CENTRO, ARRIBA, ABAJO
        PCMSK1 |= (1<<PCINT8) | (1<<PCINT9) | (1<<PCINT10) | (1<<PCINT12) |
(1<<PCINT14) | (1<<PCINT15);
```

```
}

//INTERRUPCION POR CAMBIO EN LOS PORTICOS PB0
SIGNAL(SIG_PIN_CHANGE0)
{

	PORTE = 0x01;//INDICADOR
	//REDIRECIONAMOS
	GRABAR_DATOS();

}

//INTERRUPCION POR CAMBIO EN LOS PORTICOS PB0
SIGNAL(SIG_PIN_CHANGE1)
{

	PORTE = 0x02;  //INDICADOR
	GRABAR_DATOS();

}
```

CAPITULO 4

4.-SIMULACION E IMPLEMENTACION

4.1. SIMULACION PROYECTO1: MANIPULACION DE JOYSTICK

En la figura siguiente se observa la circuitería, cableado de los elementos involucrados.

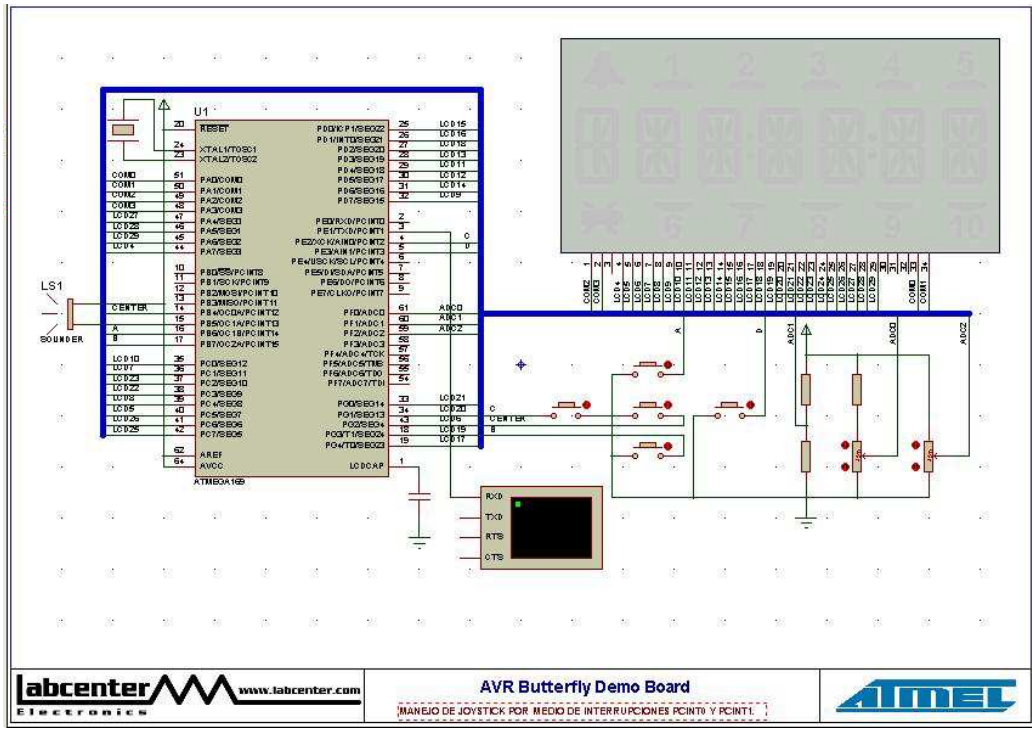

FIGURA 4-1: CIRCUITO TERMINADO DEL PROYECTO1

A continuación la figura nos muestra la pantalla al ejecutar una simulación en nuestro programa Proteus

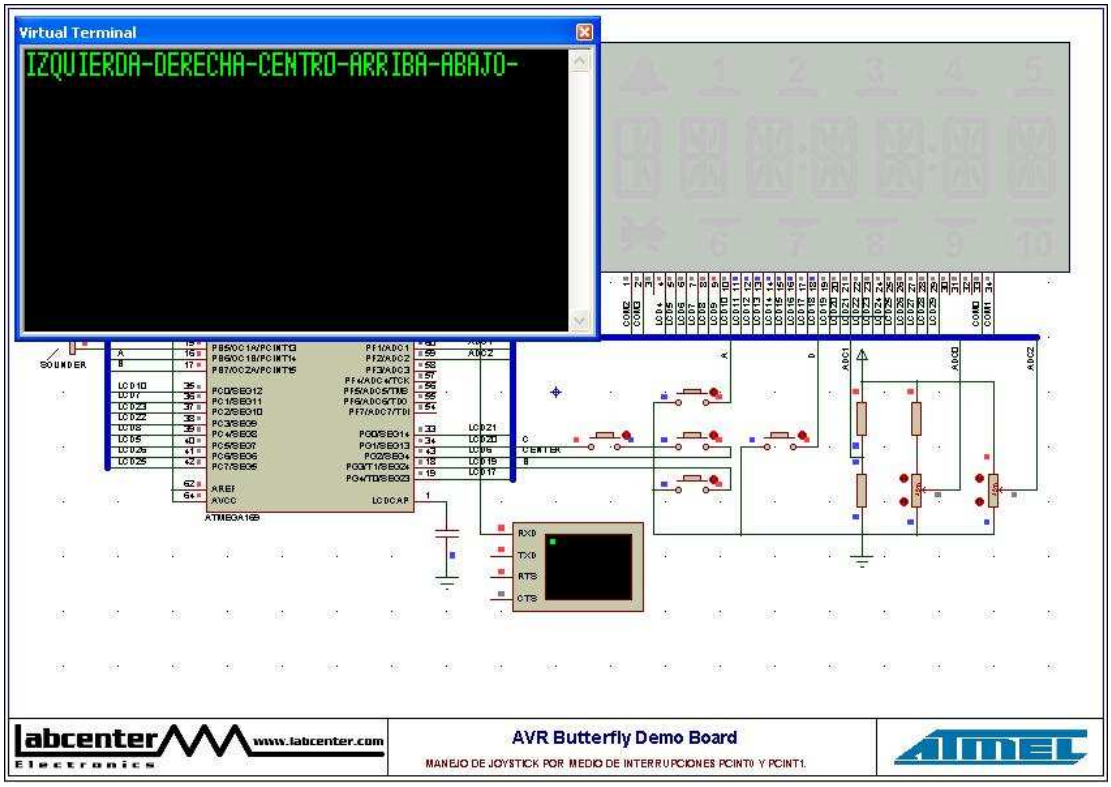

Figura 4-2: simulación del circuito en el programa Proteus.

La siguiente ilustración o figura es la pantalla de edición de componentes donde nos muestra el componente ATMEGA169, y el archivo hexadecimal que se genera al cargar el código assembler o en lenguaje C.

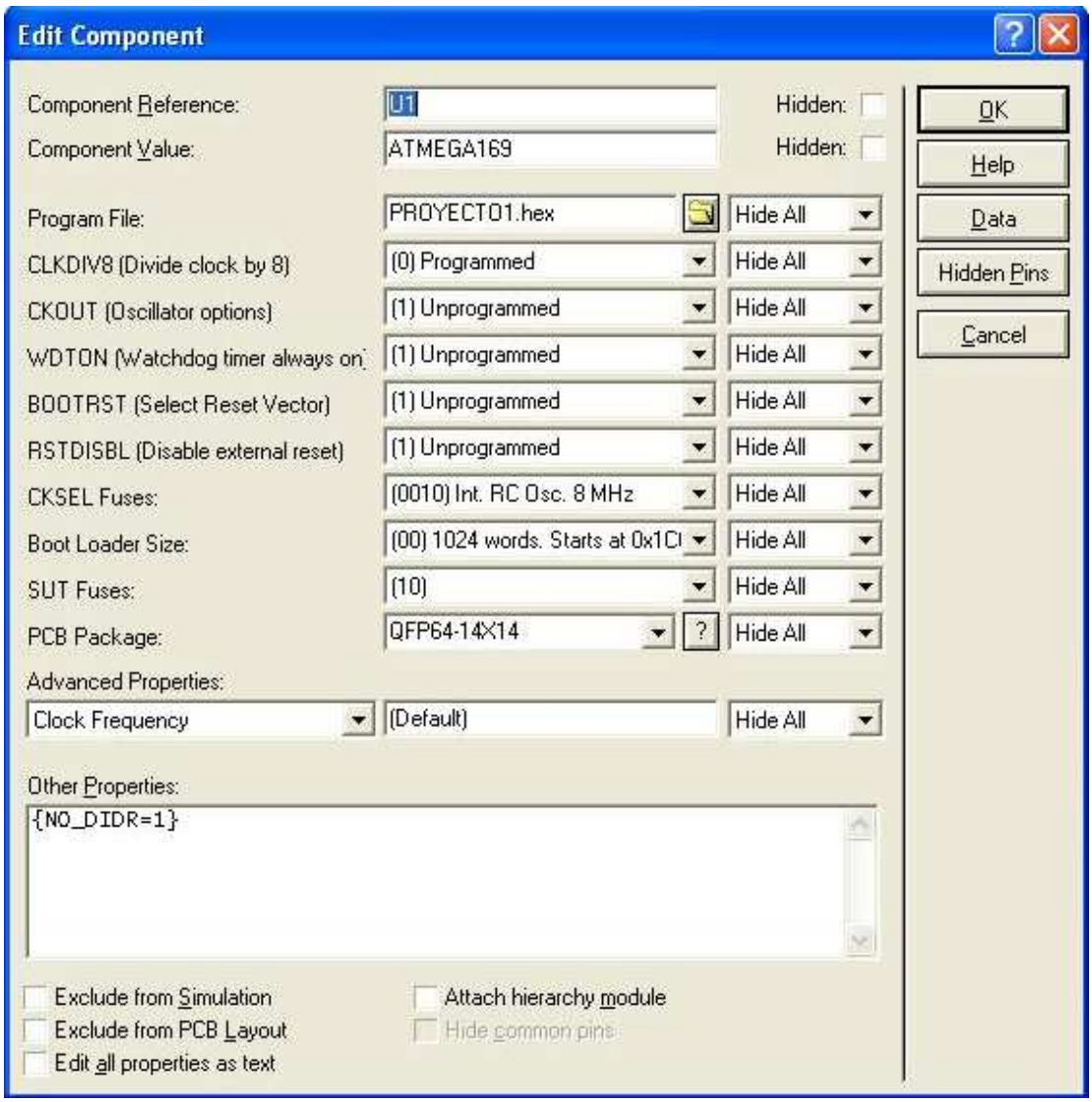

FIGURA 4-3: EDICION DE COMPONENTE ATMEGA169

En esta otra figura, se muestra otras características de los componentes tales como la velocidad en baudios, la capacidad de bits de los datos a operar.

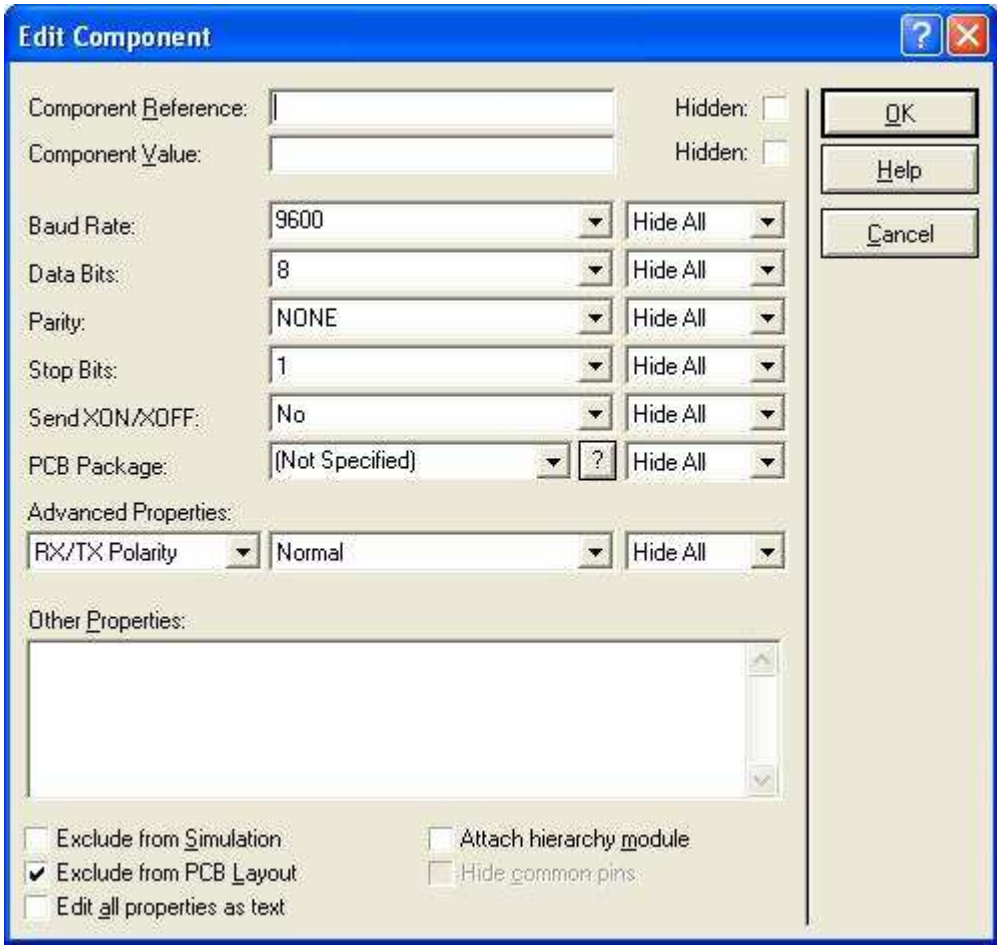

Figura 4-4: Edicion de componentes características de procesamiento

4.2 SIMULACION PROYECTO2: VARIACIONES DE COLOR DE LED GRB POR MEDIO DE MODULACION PWM

La figura muestra los componentes involucrados en la realización de este proyecto.

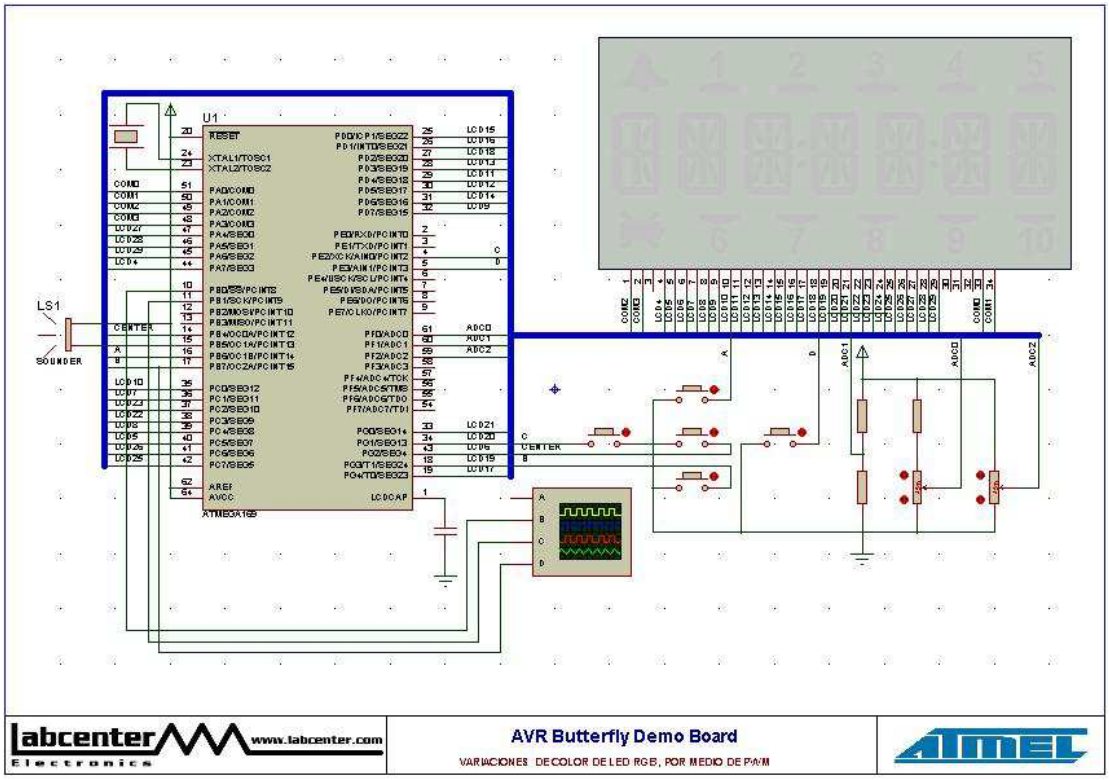

Figura 4-5: proyecto dos en Proteus vista esquemática

La figura a continuación se muestra la simulación de los pulsos que se generan utilizando la modulación PWM, cada color es a determinado rango de frecuencia.

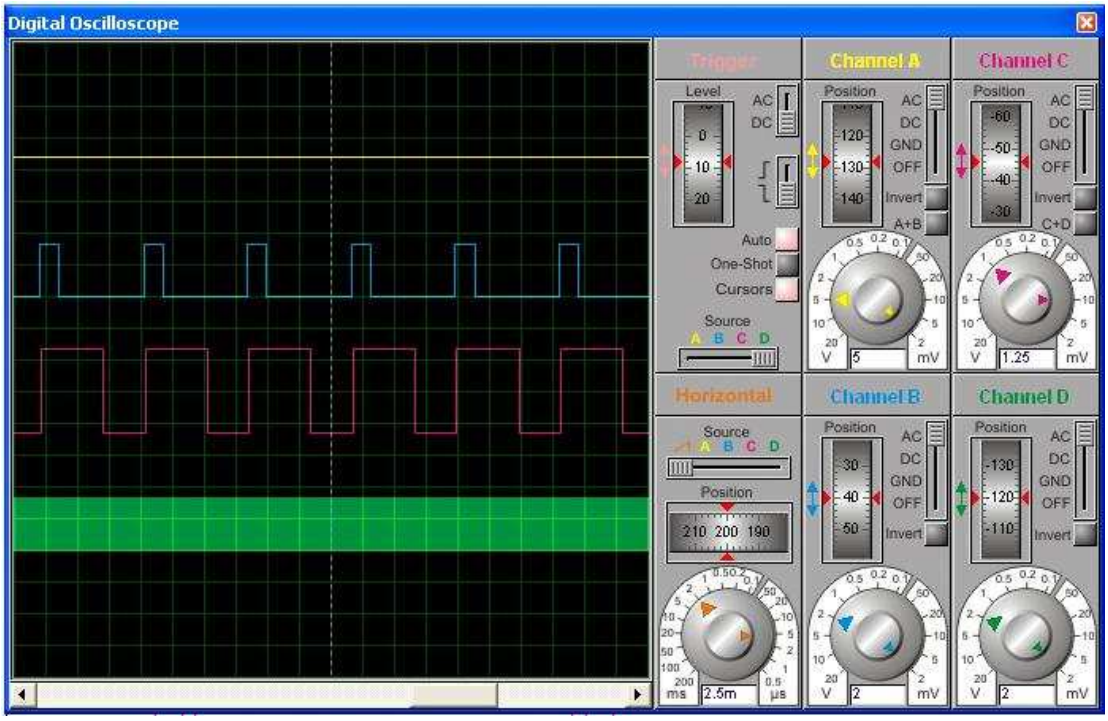

Figura 4-6: Simulación del proyecto dos en Proteus

4.3. SIMULACION PROYECTO3: CERRADURA ELECTRONICA

Se muestra en la figura los componentes utilizados para la realización de este proyecto.

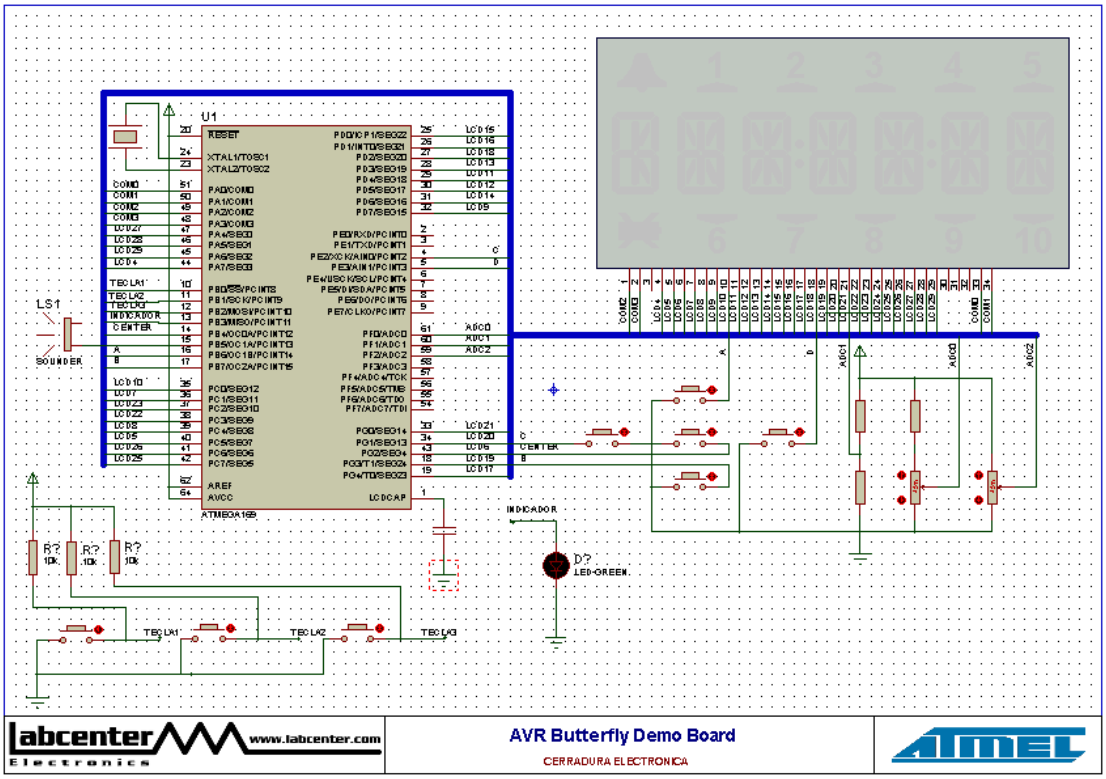

Figura 4-7: Vista esquemática en Proteus del proyecto tres

En la siguiente figura se detallan las librerías externas utilizadas para el adecuado funcionamiento del código.

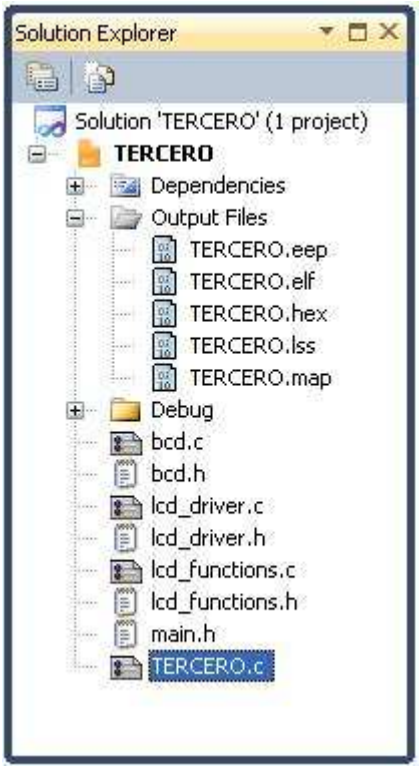

FIGURA 4-8: Librerías Externas agregadas.

En la siguiente figura se muestra la ejecución de la simulación del proyecto realizado.

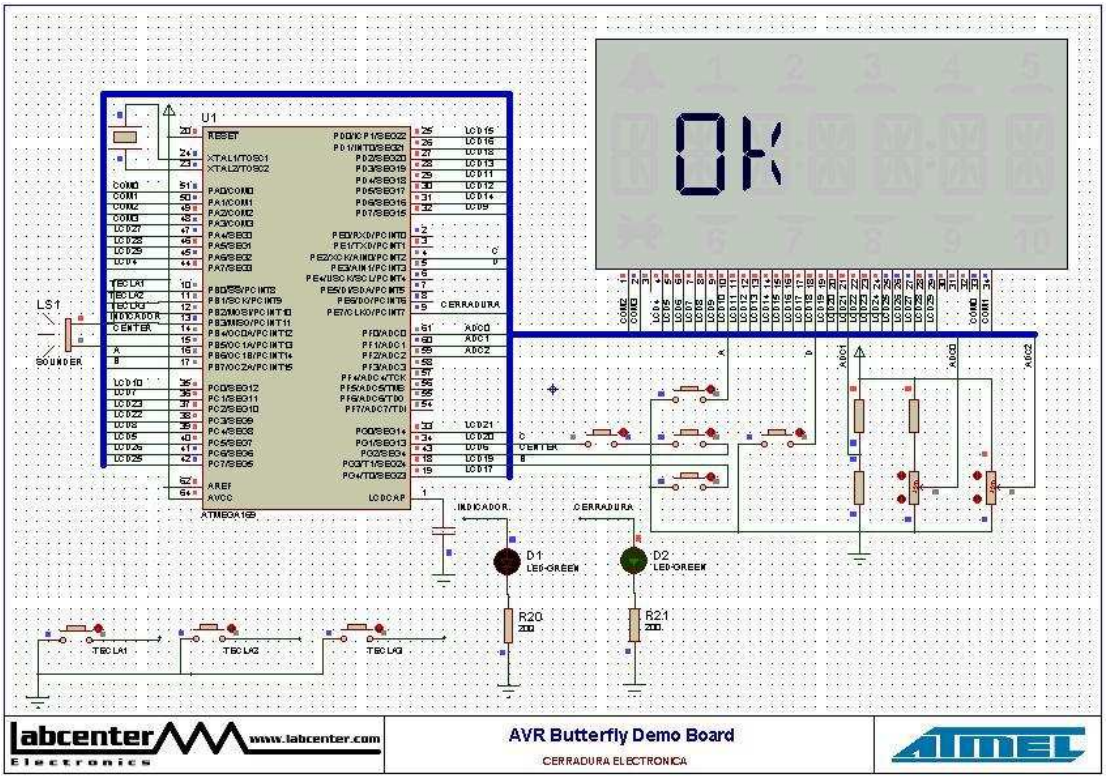

FIGURA 4-9: Diagrama de Cerradura Electrónica, Clave Correcta.

4.4. SIMULACION PROYECTO4: CONVERTIDOR ANALOGICO DIGITAL

A continuación la figura del proyecto 4 con sus respectivos componentes utilizados, simulado en el programa Proteus.

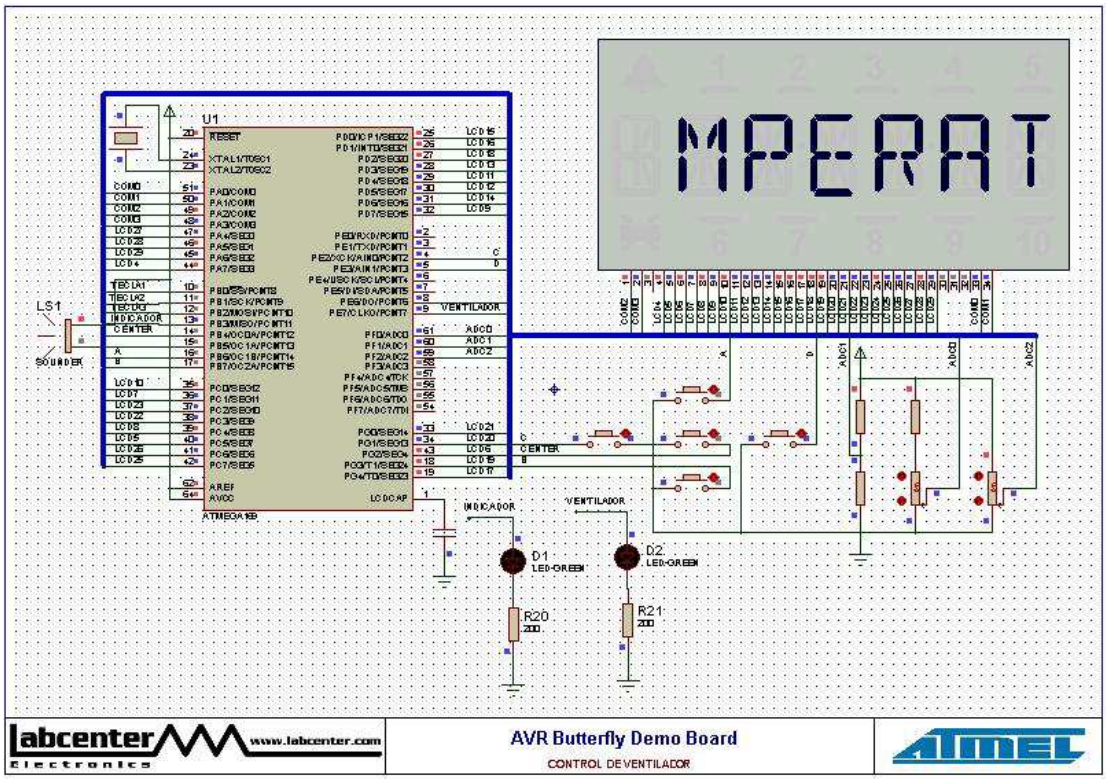

figura 4-10: diagrama del controlador de temperatura.

A continuación las librerías necesarias agregadas para la ejecución exitosa del código del programa realizado.

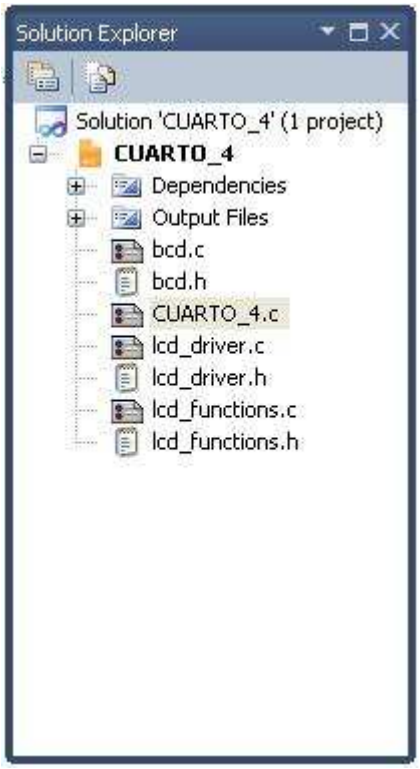

Figura 4-11: Simulación, librerías utilizadas

En esta ilustración se muestra la ejecución de la validación de exito del control de temperatura.

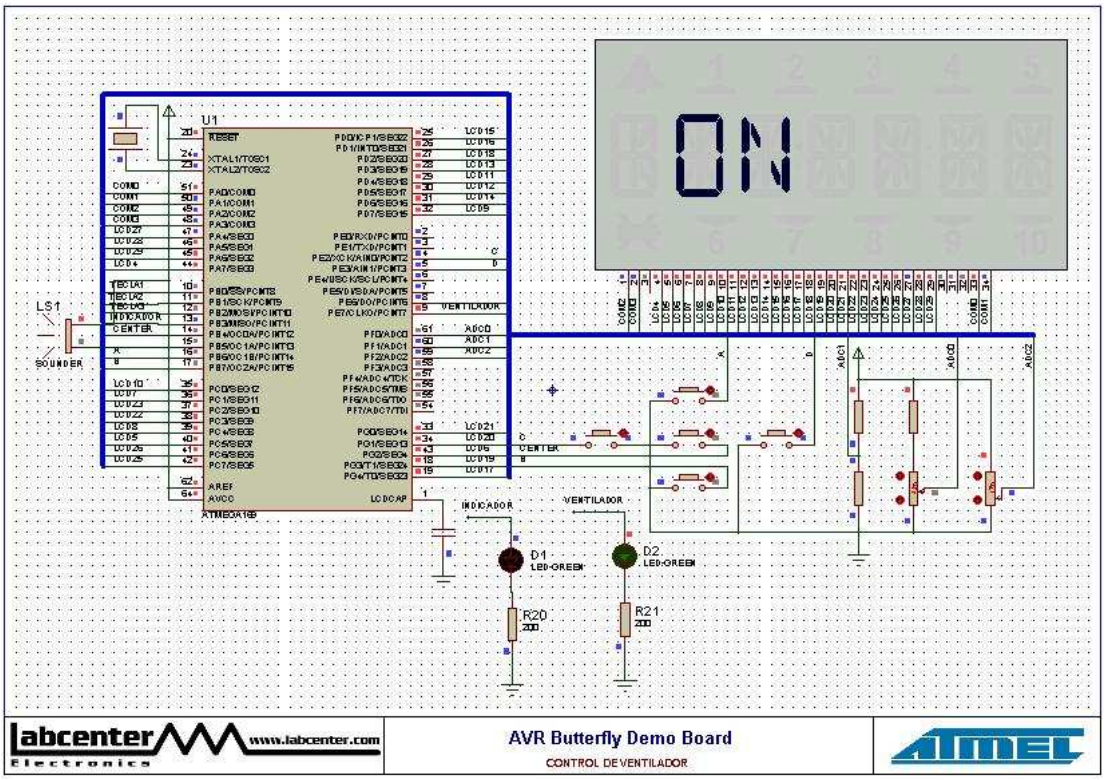

Figura 4-12: Simulación del circuito.

4.5. SIMULACION PROYECTO5: SIMULACION DE JUEGO DE DADOS

A continuación la figura del proyecto 5 con sus respectivos componentes utilizados, simulado en el programa Proteus.

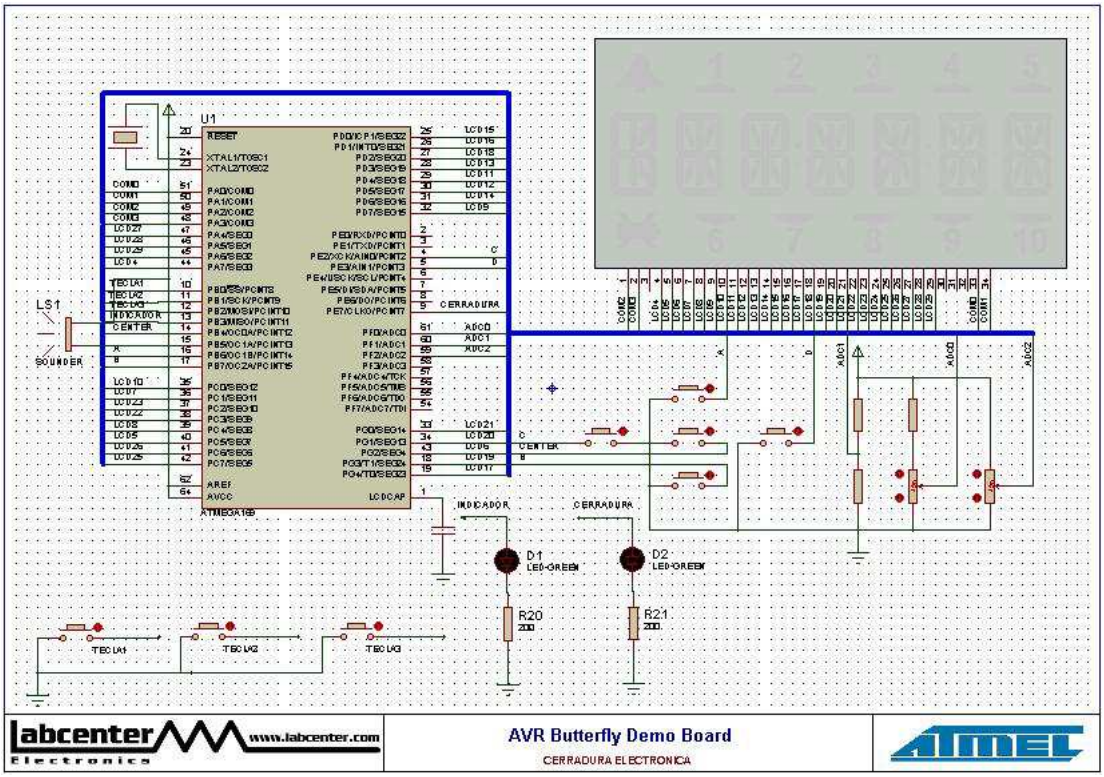

Figura 4-13: circuito del Proyecto 5.

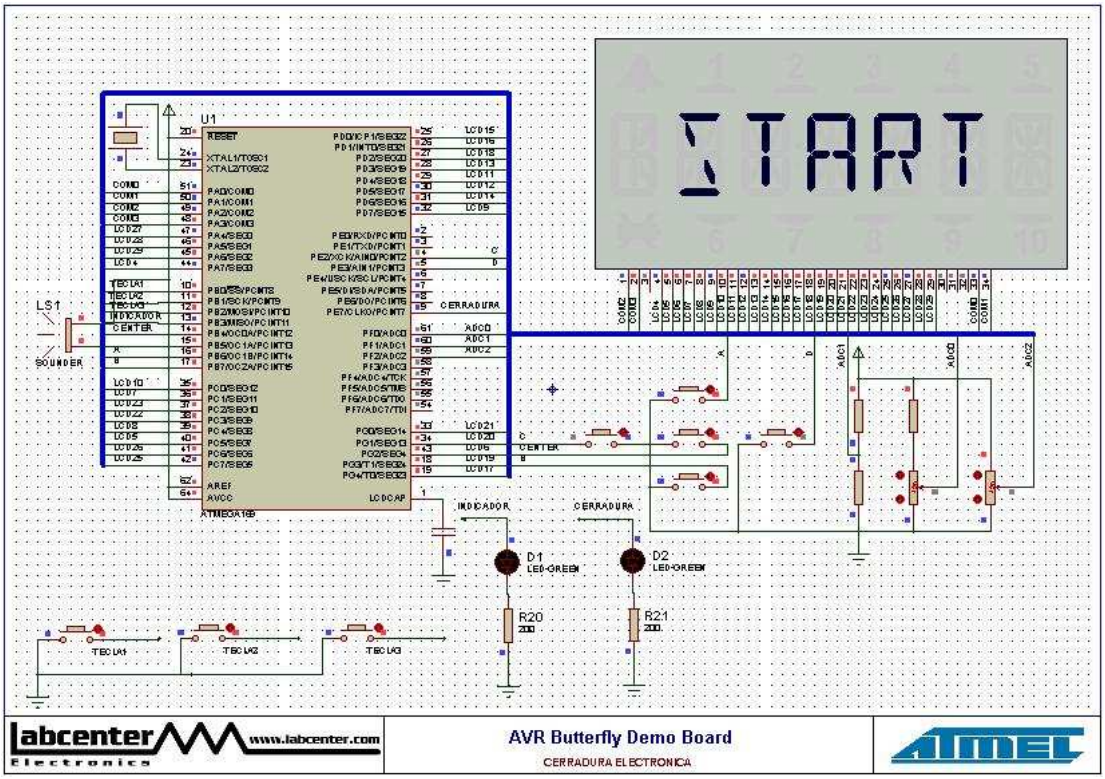

Figura 4-14: A la espera de que se presione botón start.

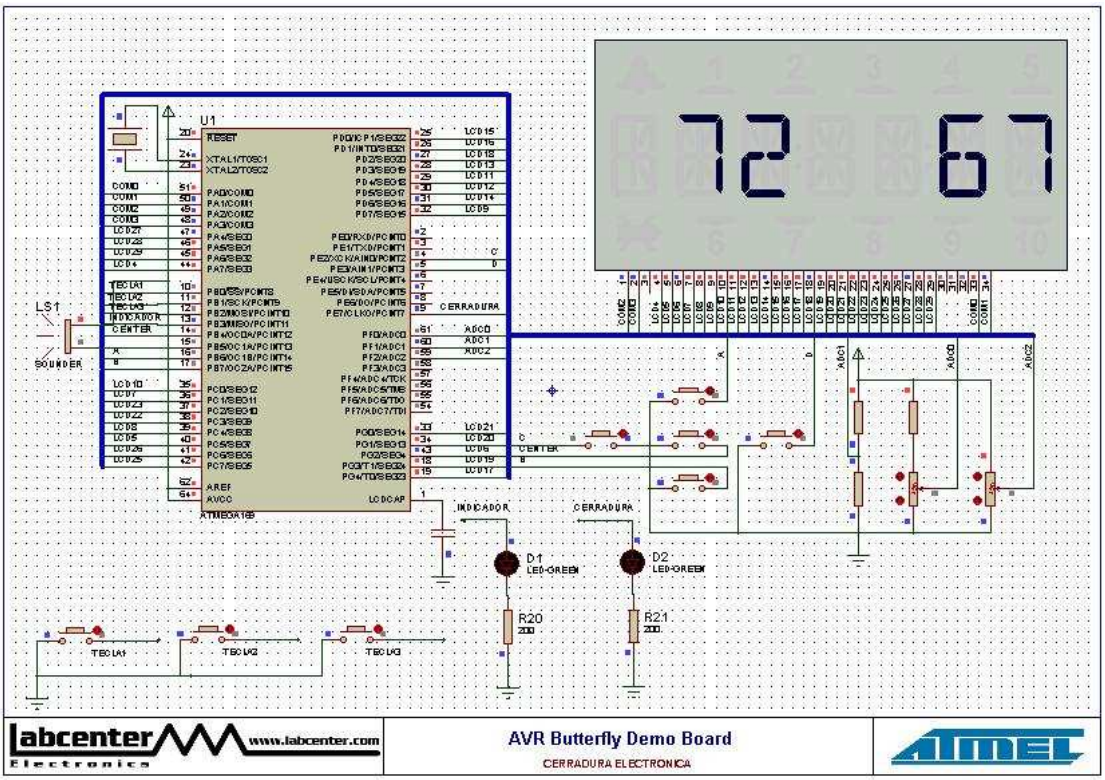

Figura 4-15: dados generados por los dos jugadores. Gana jugador 2.

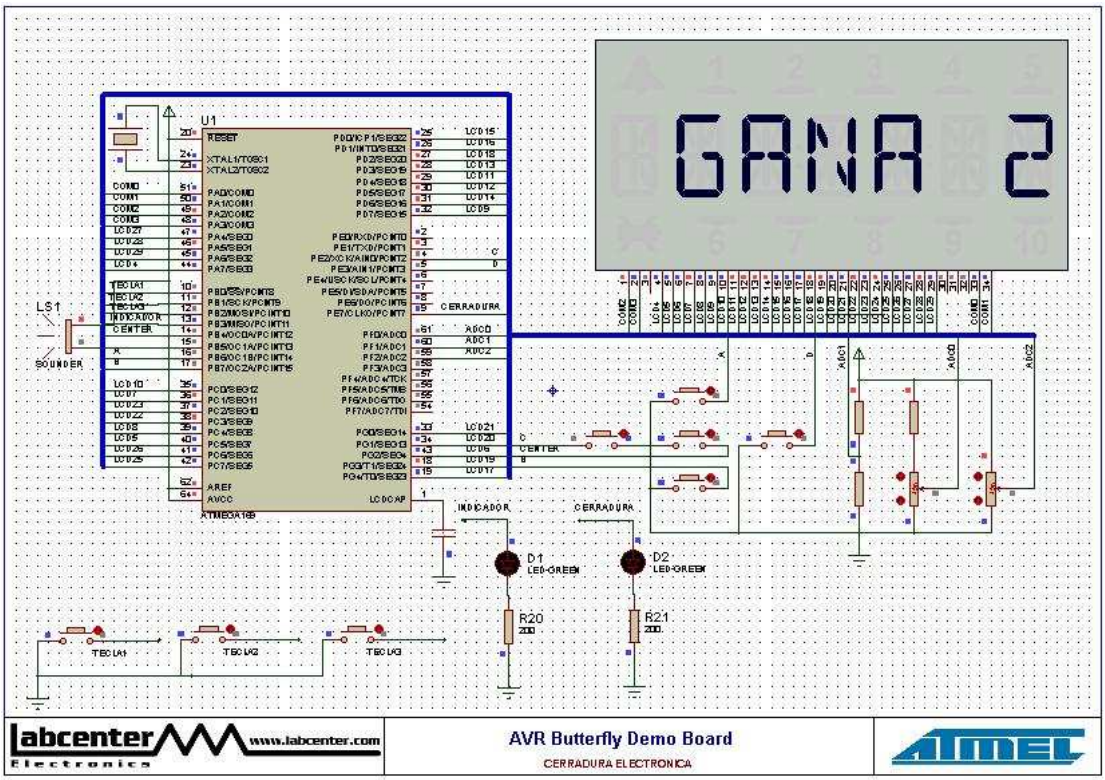

Figura 4-16: Gana jugador 2.

CONCLUSIONES

1. Se implementaron 5 proyectos Didácticos mostrando el potencial del AvrBUtterfly, estos proyectos son orientados a mostrar la versatilidad de las Interrupciones Generales de las Microcontroladores ATMEL.

2. En las Líneas de comandos tanto en lenguaje Asembler como en el Leguaje C se describen la forma en que se habilitan las interrupciones, no se utilizaron todas de las 23 interrupciones que posee el Atmega169. Sin embargo con las aplicadas se deja abierto un camino para expandirse a las demás Interrupciones.

3. Con el uso del Microcontroladores AVR como nueva tecnología, se expanden las posibilidades de desarrollos más elaborados, y junto con la tecnología de Microchip se tiene más opciones a escoger, para futuros desarrollos. En donde el precio, y más que nada la disponibilidad de los elementos es fundamental para terminar cualquier proyecto.

RECOMENDACIONES

1. Al inicio se hizo complicado, pues se relaciona la tecnología de Microchip con la ATMEL, cosa que se hizo algo confuso, el Atmega169 tiene muchas más set de Instrucciones, que al inicio las instrucciones diversas del Atmega pueden parecer las mismas, incluso pueden llegar hacer pensar al programador que es un hecho que su código está bien. Sin embargo, el aplicar las Instrucciones en assembler, hay que tener muy en cuenta la dirección de memoria de los Registros a Utilizar, como conclusión en esto les aseguro que primero hay que ver que registro se va a trabajar antes de aplicar cualquier instrucción y no al revés.

2. Complementando la recomendación descrita anterior, puedo decir que incluso el propio código expuesto en el datasheet del Atmega, NO FUNCIONA, pues al tratar de aplicarlo el compilador muestra errores. Como recomendación hay que darse cuenta que los registros del Atmega169 están en dos bancos, un banco va desde la dirección (0xFF) hasta la dirección (0x60). Por ejemplo en este banco se deben utilizar las Instrucción STS para escribir datos. Sin embargo esta misma instrucción no funciona para el Banco que va desde la Dirección 0x3F (0x5F) al 0x00 (0x20), aquí se debe utilizar la Instrucción OUT.

3. En los Programas en lenguaje C, para las Instrucciones se utiliza SIGNAL(Vector) sin embargo modernamente se utiliza ISR(vector). igualmente Funciona.

4. Se utiliza el AvrStuido 4 para el desarrollo y para grabar el AvrButterfly, sin embargo este mismo Proceso de Grabado por medio de RS232, no Funciona en la Versión 5 del AvrStudio, esto se debe a que le falta un complemento el AvrProg, este se baja de internet y debe ser incluido en las Herramientas. Con esto se consigue trabajar bien en el AvrStudio 5.

ANEXOS

BIBLIOGRAFÍA

[1] Quiceno Manrique, Andrés Felipe, Estructura general de un programa en C para sistemas embebidos, http://trucoselectronicayprogramacion.blogspot.com/2010/11/estructura-general-de-un-programa-en-c.html, Fecha de consulta noviembre 2011.

[2] AVR Tutorials, AVR Microcontroller Stack and Stack Pointer, http://www.avr-tutorials.com/general/avr-microcontroller-stack-operation-and-stack-pointer, fecha de consulta noviembre 2011.

[3] Academic, Gama de color, http://www.esacademic.com/dic.nsf/eswiki/516331 , fecha de consulta noviembre 2011

[4] DISQUS, Interrupciones, http://jmnlab.com/interrupciones/interrupciones.html, fecha de consulta noviembre del 2011

[5] Escuela Superior Politécnica de ALCOY, COMPARATIVA DE MICROCONTROLADORES ACTUALES, http://server-die.alc.upv.es/asignaturas/LSED/2002-03/Micros/downloads/trabajo.pdf, fecha de consulta noviembre del 2011

[6] Domínguez Arellano, Rufino, Interrupciones, http://ael.110mb.com/informatica/Int8051.pdf , fecha de consulta diciembre del 2011.

[7] Fundación Wikimedia, Inc, Intel 8051, http://es.wikipedia.org/wiki/Intel_8051 , fecha de consulta diciembre 2011

[8] Frino, Luis, ASPECTOS BASICOS DEL PROGRAMA PROTEUS, http://www.frino.com.ar/proteus.htm , fecha de consulta diciembre 2011

[9] Landen, AVR Butterfly Quick Start User Guide, http://elmicro.com/files/atmel/atavrbfly_quickstart.pdf , fecha de consulta diciembre 2011

[10] Doxygen, Global manipulation of the interrupt flag, http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html , fecha de consulta diciembre 2011

[11] Mikroontroller, ATMEGA128: AD-Wandlung - Interrupt, http://www.mikrocontroller.net/topic/38318 , fecha de consulta diciembre 2011

[12] Pudn, Downloads Codes Google Pudn, http://search.pudn.com/search_read_en.asp?keyword=AVR+adc , consulta diciembre 2011

[13] Pudn, ADC.c, http://read.pudn.com/downloads90/sourcecode/embed/343367/task/task1/ADC.c__.htm , fecha de consulta diciembre 2011

[14] DISQUS, ADC, http://jmnlab.com/adc/adc.html , fecha de consulta diciembre 2011

[15] Pardue, Joe, Butterfly Alternate Pin Uses, http://www.smileymicros.com/download/Butterfly%20Alternate%20Pin%20Uses.pdf?&MMN_position=62:62 , fecha de consulta diciembre 2011

[16] AVR Freaks, Temperature sensor, http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&p=819824 , fecha de consulta diciembre 2011

[17] Schmitt, Günter, Mikrocomputertechnik mit Controllern der Atmel, http://books.google.com.ec/books?id=AI_HVUE0xYEC&pg=PA413&lpg=PA413&dq=ADCO+AVR+LEER&source=bl&ots=Q-NNdjCX5U&sig=f5nsn-LNrhlUrccddH_VSZsJUGQ&hl=es&sa=X&ei=QjEKT5z5FM35ggeEiPGJAg&ved=0CCgQ6AEwAQ#v=onepage&q=ADCO%20AVR%20LEER&f=false , fecha de consulta diciembre 2011