ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"COMUNICACIÓN UART DE LA PC CON LA PLATAFORMA INTERACTIVA CREADA, BASADA EN LA TARJETA DE DESARROLLO AVR BUTTERFLY A TRAVES DEL PUERTO SERIE RS232"

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRICIDAD
ESPECIALIDAD EN ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL
INGENIERO EN TELECOMUNICACIONES

Presentado por:

Jorge Gerardo Arévalo Suárez
Frank Alexander Benalcázar Cano

GUAYAQUIL – ECUADOR

AÑO 2011

AGRADECIMIENTO

A Dios primeramente por guiar nuestro camino y mostrarnos esa luz de esperanza y dotarnos siempre de inteligencia y sabiduría para saber tomar nuestras decisiones.

A nuestros padres por saber llevarnos siempre por el camino del bien y brindarnos todo su apoyo en nuestras vidas y estudios.

Al Ing. Carlos Valdivieso por saber guiarnos en el desarrollo de esta tesis y su apoyo incondicional en la ejecución de nuestro proyecto.

DEDICATORIA

A nuestros padres por ser ejemplo de nuestras vidas, por habernos brindado la mejor educación posible y ser apoyo incondicional en toda nuestra etapa de aprendizaje.

TRIBUNAL DE SUSTENTACIÓN

_____

Ing. Carlos Valdivieso A.

PROFESOR DEL SEMINARIO DE GRADUACIÓN

_____

Ing. Hugo Villavicencio V.

PROFESOR DELEGADO DEL DECANO

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta 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 Gerardo Arévalo Suárez

Frank Alexander Benalcázar Cano

ÍNDICE GENERAL

Capítulo 3

INDICE DE GRAFICOS Y TABLAS

RESUMEN

Este artículo tiene como objetivo de mostrar el funcionamiento de la comunicación UART, a través del puerto serie del computador hacia la plataforma interactiva creada, que está basada en el kit de desarrollo Avr Butterfly, la cual va permitir verificar los recursos disponibles en la comunicación UART y desarrollar muchos proyectos, el Kit de desarrollo consta de un micro-controlador integrado el Atmega169, de una pantalla LCD, de un Joystick que nos permite la interacción con la tarjeta y de otra funciones muy optimas para la realización de proyectos.

Para la realización del proyecto se utiliza el Kit AVR Butterfly, que posee un ATmega 169, que fue programado en el entorno de AVR Studio 4 con compilador GCC, para la utilización del joystick y la transmisión correspondiente de la instrucción.

El capitulo 1 de este trabajo trata sobre la importancia de las comunicaciones Uart, además de varias comparaciones con micro-controladores de otros fabricantes, en el capítulo 2 se destaca las herramientas de Hardware y Software necesarias para la ejecución del trabajo, el capítulo 3 se refiere a las plataformas necesarias para la ejecución de este proyecto, y el capitulo 4 trata sobre el enunciado y desarrollo de los proyectos que se presentaran.

INTRODUCCIÓN

El siguiente proyecto tiene como finalidad demostrar el funcionamiento de la comunicación UART, a través del puerto serie del computador hacia la plataforma interactiva creada, que está basada en el kit de desarrollo Avr Butterfly, la cual va permitir verificar los recursos disponibles en la comunicación UART y desarrollar muchos proyectos, el Kit de desarrollo consta de un microcontrolador integrado el Atmega169, de una pantalla LCD, de un Joystick que nos permite la interacción con la tarjeta y de otra funciones muy optimas para la realización de proyectos.

Para la ejecución de este proyecto hicimos uso de dos software muy funcionales, los cuales son el Avr Studio 4 de la misma fabricante del Avr Butterfly la Cía. Atmel este software lo utilizamos para la programación en lenguaje C y ensamblador, además de el Proteus 7.0 el cual nos permite simular nuestro proyecto de una forma animada y verificar su funcionamiento.

CAPITULO 1

1.0 GENERALIDADES

Para la realización de este proyecto se ha creado una plataforma interactiva que nos permite apreciar el funcionamiento de los recursos disponibles en la comunicación UART, en este capítulo vamos a tratar todo lo relacionado a la comunicación UART, su importancia, sus principales característica, el modo de establecer la comunicación y los principales fabricantes de Microcontroladores.

1.1 IMPORTANCIA DE LAS COMUNICACIONES UART

El corazón del sistema de comunicaciones serie es la UART, acrónimo de Universal Asynchronous Receiver-Transmitter. Es un chip cuya misión principal es convertir los datos recibidos del bus del PC en formato paralelo, a un formato serie que será utilizado en la transmisión hacia el exterior. También realiza el proceso contrario: transformar los datos serie recibidos del exterior en un formato paralelo entendible por el bus. [8]

La UART es un dispositivo programable en el que pueden establecerse las condiciones que se utilizarán para la transmisión (velocidad, paridad, longitud y bits de parada). En los primeros PCs, eran circuitos integrados 8250 de National Semiconductor (un chip de 40 patillas DIP -Dual-In-Line-), pero se han ido utilizando otros a lo largo de la evolución del PC.

Las UART o USART se diseñaron para convertir las señales que maneja la CPU y transmitirlas al exterior. Las UART deben resolver problemas tales como la conversión de voltajes internos del DCE (Data Communication Equipment) con respecto al DTE (Data Terminal Equipment), gobernar las señales de control, y realizar la transformación desde el bus de datos de señales en paralelo a serie y viceversa. Debe ser robusta y deberá tolerar circuitos abiertos, cortocircuitos y escritura simultánea sobre un mismo pin, entre otras consideraciones.

Este periférico está presente en casi cualquier microcontrolador, normalmente en forma de UART (Universal Asynchronous Receiver Transmitter) o USART (Universal Síncronos Asynchronous Receiver Transmitter) dependiendo de si permiten o no el modo sincrónico de comunicación.

El destino común de este periférico es la comunicación con otro microcontrolador o con una PC y en la mayoría de los casos hay que agregar circuitos externos para completar la interfaz de comunicación. La forma más común de completar el puerto serie es para comunicarlo con una PC mediante la interfaz EIA-232 (más conocida como RS-232), es por ello que muchas personas se refieren a la UART o USART como puerto serie RS-232, pero esto constituye un error, puesto que este periférico se puede utilizar para interconectar dispositivos mediante otros estándares de comunicación. En aplicaciones industriales se utiliza preferiblemente RS-485 por sus superiores alcances en distancia, velocidad y resistencia al ruido.

Cuando se requiere conectar un micro-controlador (con señales típicamente entre 3.3 y 5 V) con un puerto RS-232 estándar se utiliza un driver de línea, típicamente un MAX232 o compatible, el cual mediante dobladores de voltaje positivos y negativos

permite obtener la señal bipolar (típicamente alrededor de +/- 6V) requerida por el estándar.

El correcto funcionamiento y velocidad de la UART es fundamental para las comunicaciones serie; con independencia de la velocidad del módem, la comunicación no podrá ser más rápida que la que permita la UART.

1.2 CARACTERISTICAS DE LA USART DEL ATMEGA169

La USART o Universal Synchronous and Asynchronous serial Receiver and Transmitter (Figura 1.2), es un dispositivo de comunicación serial altamente flexible, sus principales características son:

- ✓ Operación Full Dulpex
- ✓ Registros de transmisión y recepción independientes
- ✓ Operación síncrona o asíncrona
- ✓ Generador de Baud Rate de alta resolución
- ✓ Detección de error
- ✓ Filtro de ruido
- ✓ Modo de comunicación multiproceso
- ✓ Doble velocidad en modo de comunicación asíncrono

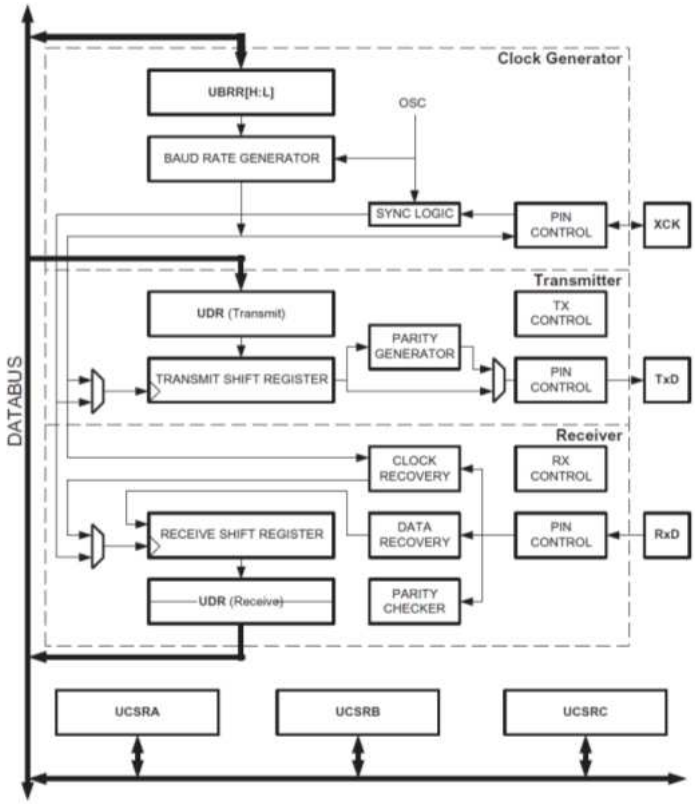


Figura 1.2: Diagrama de bloques de la Uart.

El manejo de la comunicación serial presenta muchos beneficios, entre los que destacan, el control de sistemas a través de la computadora realizando cálculos complejos, visualizando y graficando datos, entre otros. Es importante destacar que también existen muchos programas aparte de la Hyperterminal los cuales pueden entablar comunicación serial con el microcontrolador, programas como MatLab, LabVIEW, TeraTerm entre otros.

1.3 CONECTORES DEL AVR BUTTERFLY

Algunos de los pines de I/O del micro-controlador ATmega169 están disponibles en los conectores del AVR Butterfly. Estos conectores son para comunicación, programación y entrada al ADC del ATmega169. En la Figura 1.3 se puede apreciar los conectores del AVR Butterfly.

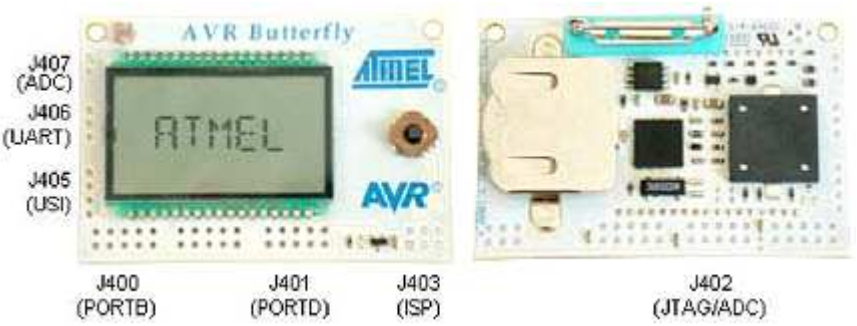


Fig. 1.3 Conectores del AVR Butterfly para acceso a periféricos

1.4 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.

1.4.1 DISTRIBUCION DE PINES PARA LA COMUNICACIÓN SERIAL ENTRE EL AVR Y LA PC

La comunicación con la PC requiere de tres líneas: TXD, RXD y GND. TXD es la línea para transmitir datos desde la PC hacia el AVR Butterfly, RXD es la línea para recepción de datos enviados desde el AVR Butterfly hacia la PC y GND es la tierra común. En la Tabla 1.4.2.1 se observa la distribución de los pines para la comunicación serial, a la izquierda los pines del AVR Butterfly y a la derecha los pines del conector DB9 de la PC.

Tabla. 1.4.1.1. Distribución de pines, AVR Butterfly Vs. PC

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

En la Figura 1.4.2.2 se observa cómo se debe hacer el cableado para la comunicación, a través de la interfaz serial RS-232, entre el AVR Butterfly y la PC. A la izquierda se aprecia un conector DB9 hembra soldado a los cables que se conectan a la interfaz USART del AVR Butterfly (derecha).

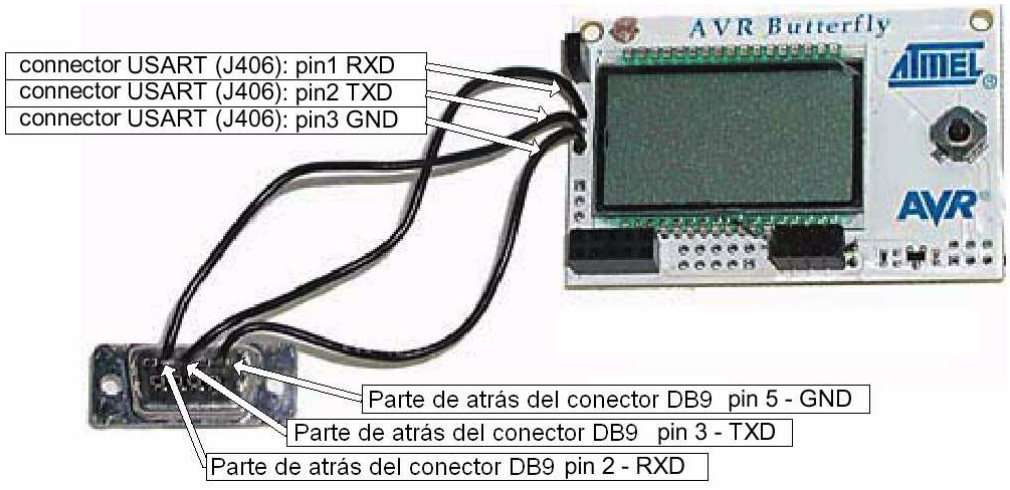


Figura. 1.4.2.2. Conexiones para interfaz USART del AVR Butterfly

1.5 LOS MICROCONTROLADORES AVR

La empresa Atmel ha desarrollado una gran cantidad de micro-controladores en diferentes gamas, de forma similar a lo que ha hecho la empresa Microchip con nuestros viejos amigos: “los PICs”. [1]

Quizá, el más popular es el ATMEL AT90S1200, que es algo así como el 16F84 de Microchip (en cuanto a popularidad se refiere). A continuación se realiza algunos datos comparativos en la Tabla 1.5 entre el AT90S1200 y el PIC16F84:

Tabla. 1.5 Comparación entre Microcontroladores de Atmel y Microchip

Nº de instrucciones:	**AVR - 89,**	**PIC – 35**
Registros RAM:	AVR - 32,	PIC – 68
Velocidad:	AVR - 12MHz,	PIC: 20MHz
Memoria de Programa:	AVR - 1kByte FLASH (512 líneas de programa, 16 bits por inst.)	PIC: 1kx14 (1024 líneas de programa de 14 bit cada una).
Memoria EEPROM libre:	AVR - 64Bytes	PIC - 64Bytes
Salidas:	AVR - 15	PIC – 13
TIMER:	AVR - 1 de 8bit (con prescaler desde CK hasta CK/1024)	PIC - 1 de 8 bit (con prescaler desde 1:2 hasta 1:256)
Comparador Analógico (NO ADC):	AVR - 1	PIC - NO POSEE
Watchdog:	AVR - Si Posee	PIC - Si Posee
Oscilador interno:	Ambos poseen, en el AVR sólo es habilitado con programación paralela.	
Niveles de pila (STACK):	AVR - 3	PIC – 8
Interrupciones:	AVR - reset, interna, externa, timer y por comparador analógico	PIC - 5 interrupciones.

Básicamente, los AVR tienen 3 registros para cada puerto de salida a saber:

- ✓ DDRB - Sirve para decir qué patas son de entrada o salida, “0” es entrada, “1” es salida (es inverso a los PIC).
- ✓ PINB - Registro que sirve para entradas solamente.
- ✓ PORTB - Registro que sirve para salidas solamente.

Esto significa que para leer una entrada se debe usar el registro PINB mientras que para escribir datos en una salida se debe emplear el registro PORTB (obviamente si hacemos referencia a las patas del puerto B).

En el ATMEL AT90S1200 el PORTB tiene 8 bits de datos, a diferencia del PORTD que tiene sólo 7. El bit 7 del PORTD no se emplea; PORTD también consta de 3 registros: DDRD, PORTD y PIND.

1.5.1 PRINCIPALES CARACTERISTICAS

Las principales características del dispositivo son las siguientes:

- Alto desempeño, baja potencia.
- Arquitectura RISC avanzada:

  -120 instrucciones poderosas, la mayoría con ejecución de un solo ciclo de reloj.

  -32x8 registros de trabajo de propósito general.

  -operación totalmente estática.
- Programa y Memoria de Datos no volátiles:

  -2/4/8 kbytes de Memoria Flash Programable en el sistema, con duración: 10000 ciclos de escritura/borrado.

  -128/256/512 bytes de EEPROM programable en el sistema, con duración: 100000 ciclos de escritura/borrado.

  -128/256/512 bytes de SRAM interna.

  -Cerrojo de programación para autoprogramar la Memoria Flash y Seguridad de Datos de EEPROM.
- Características Periféricas:

  -Contador/Temporizador de 8 bits con Prescaler y dos canales PWM.

  -Contador/Temporizador de Alta Velocidad de 8 bits con Prescaler separado:

  · Dos Salidas PWM de Alta Frecuencia con Registros de Comparación de Salida

separados.

·Generador Programable de Tiempo Muerto.

-Interfaz Serie Universal con Detector de Condición de Comienzo.

-ADC de 10 bits:

·Cuatro Canales de Una Sola Salida.

·Dos Pares de Canales ADC Diferenciales con Ganancia Programable (1x, 20x).

-Temporizador Programable de Vigilancia con Oscilador separado dentro del integrado.

-Comparador Analógico dentro del integrado.

❖ Características Especiales del Microcontrolador:

-Sistema de Depuración debug WIRE dentro del integrado.

-Programable dentro del Sistema a través del Puerto SPI.

-Fuentes de Interrupción Externas e Internas.

-Modos de Descanso en Baja Potencia, de Reducción de Ruido de ADC, y de Reducción de Potencia.

-Circuito Mejorado de Re-inicialización de Encendido.

-Circuito Programable de Detección de Brown-out (estado en que la tensión es entre un 8 y un 12% inferior al valor típico).

-Oscilador Calibrado interno.

1.6 FABRICANTES DE MICROCONTROLADORES

Por lo general los fabricantes de microprocesadores son los mismos que de microcontroladores. Los fabricantes de microcontroladores son más de 50, podemos mencionar a:

- Atmel
- Motorola
- Intel
- Microchip
- NEC
- Hitachi
- Mitsubishi
- Philips
- Matsushita
- Toshiba
- AT&T
- Zilog
- Siemens
- National Semiconductor, etc.

1.6.1 PRINCIPALES MARCAS

Según volumen de ventas y diversidad de modelos podemos establecer como principales Marcas a los siguientes fabricantes:

- Microchip Technology Corp.
- ·STMicroelectronics
- Atmel Corp.
- Intel Corp.

Como se puede apreciar en las siguientes gráficas basadas en datos referentes a ventas (Figura 1.6.1), crecimientos de empresa anuales (Figura 1.6.2), cuotas de mercado (Figura 1.6.3) y capitalización bursátil (Figura 1.6.4) referentes al mercado de los

circuitos integrados, compañías como Microchip, Intel y Atmel son susceptibles de mención y estudio debido a su especialización en el área de los microcontroladores.

Figura 1.6.1 Ventas comerciales

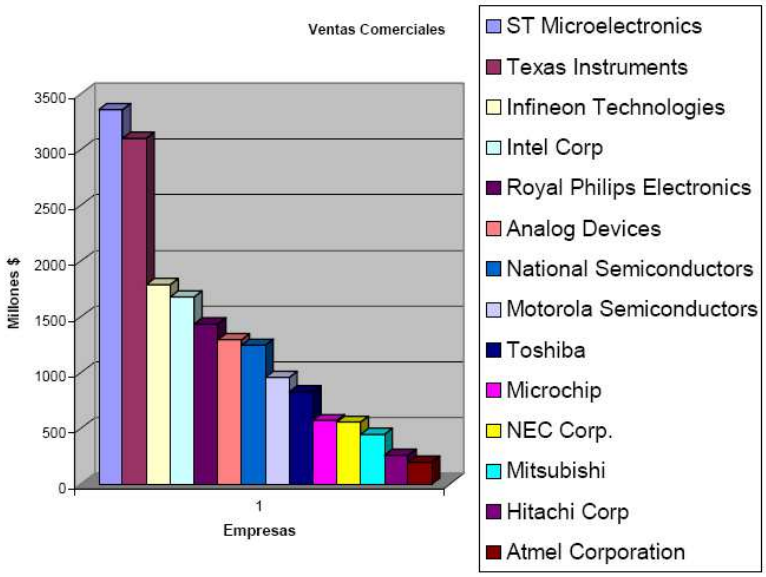


Figura 1.6.2 Crecimiento Económico

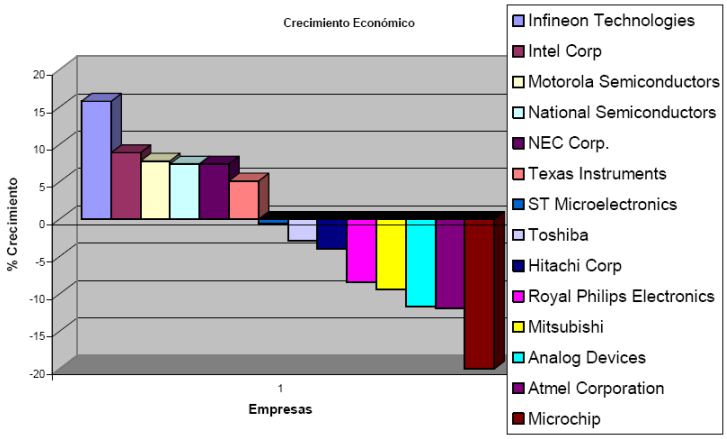


Figura 1.6.3. Cuota de mercado

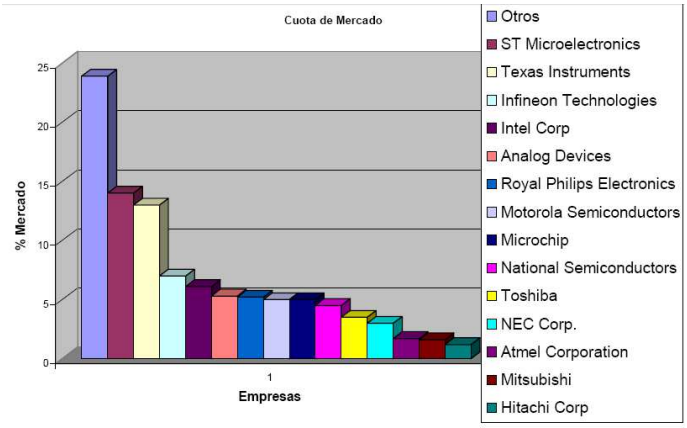
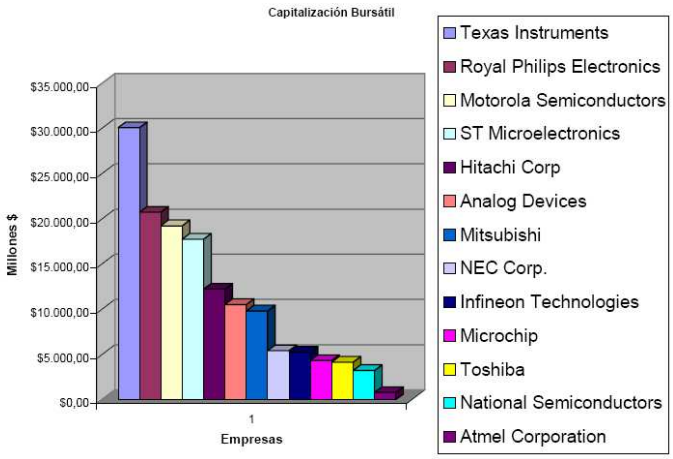


Figura 1.6.4. Capitalización



Bursátil

1.6.2 COMPARACION ENTRE FAMILIAS DE MICROCONTROLADORES

En base a la realización este trabajo se ha querido comparar entre tres familias de microcontroladores que pertenecen a tres de las mayores empresas del medio, estas son Microchip, Motorola y Atmel.

A continuación se muestran cuadros comparativos de cada una de las principales familias de microcontroladores de cada empresa y sus recursos disponibles:

MOTOROLA:

Las Familias de Microcontroladores de Motorola (Tabla 1.6.2.1.) presentan una gran cantidad de aplicaciones, por la amplia gama de microcontroladores que posee y su disponibilidad. Cabe señalar que la familia más utilizada de los microcontroladores de 8 bits de Motorola por velocidad, memoria y precio es la M68HC11.

M68HC11 Family

Devices	Internal RAM	Mask ROM	EEPROM	EPROM	OTPROM	Max Bus Frequency
68HC11F1	1024	-	512	-	-	5, 4, 3, 2
68HC711E9	512	-	512	-	12288	4, 3, 2, 1
68HC11KS1	1024	-	640	-	-	4, 3, 2
68HC711E20	768	-	512	-	20480	4, 3, 2
68HC11K0	768	-	-	-	-	4, 3, 2
68HC11K1	768	-	640	-	-	4, 3, 2
68HC711KS2	1024	-	640	32768	32768	4, 3
68HC11P1	1024	-	640	-	-	4, 3
68HC11P2	1024	32768	640	-	-	4, 3
68HC11E9	512	12288	512	-	-	3, 2, 1
68HC11E0	512	-	-	-	-	3, 2
68HC11E1	512	-	512	-	-	3, 2
68HC11D0	192	-	-	-	-	3, 2
68HC711D3	192	-	-	-	4096	3, 2

Tabla 1.6.2.1. Familia de Microcontroladores de Motorola

ATMEL:

Las familia de Microcontroladores de Atmel están surgiendo debido a las magníficas prestaciones que ofrecen con respecto a sus competidores como se pueden apreciar en las tablas, pero lo que en parte les perjudica es la falta de información técnica necesaria sobre todo en español, lo que si se encuentra con los otros fabricantes, y lo cual permite que muchas personas escojas otros microcontroladores para sus proyectos.

La familia de microcontroladores de 8 bits más utilizada de Atmel es la AVR. (Tabla 1.6.2.2).

Atmel AVR 8 bits				
Device	Flash (Kbytes)	EEPROM (Kbytes)	SRAM (bytes)	F.max (MHz)
ATmega128	128	4	4096	16
ATmega64	64	2	4096	16
ATmega32	32	1	2048	16
ATmega16	16	0,5	1024	16
ATmega162	16	0,5	1024	16
ATmega169	16	0,5	1024	16
ATmega8	8	0,5	1024	16
ATmega8515	8	0,5	512	16
ATmega8535	8	0,5	512	16

Tabla 1.6.2.2. Familia de Microcontroladores de Atmel

MICROCHIP:

Quizás de todos los fabricantes expuestos, Microchip es el que más variedad e información técnica posee, cuenta actualmente con alrededor de 159 microcontroladores diferentes.

La Familia de microcontroladores insignia de esta empresa es la PIC16 (Tabla 1.6.2.3.), y el microcontrolador más utilizado el PIC16F84.

PIC16 Microcontroller Family				
Device	Data RAM	ADC	Words	Speed
PIC16C66	368	-	8192	20
PIC16C67	368	-	8192	20
PIC16C76	368	5	8192	20
PIC16F76	368	5	8192	20
PIC16F77	368	8	8192	20
PIC16F876	368	8	8192	20
PIC16F876A	368	5	8192	20
PIC16F87	368	-	4096	20
PIC16F88	368	7	4096	20
PIC16F737	368	11	4096	20
PIC16F747	368	14	8192	20
PIC16F767	368	11	8192	20
PIC16F777	368	14	8192	20
PIC16F84A	68	-	1024	20

Tabla 1.6.2.3. Familia de Microcontroladores de Microchip

CAPITULO 2

RECURSOS DE HARDWARE Y SOFTWARE DEL MICROCONTROLADOR ATMEL AVR

2.1 INTRODUCCIÓN

ATMEL fabrica los microcontroladores de la familia AVR, esta nueva tecnología proporciona todos los beneficios habituales de arquitectura RISC (Reduced instruction set computer) y memoria flash reprogramable eléctricamente. La característica que los identifica a estos microcontroladores de ATMEL es la memoria flash y Eprom que incorpora. AVR compite con varias familias de microcontroladores bien establecidas en el mercado, tales como 8051 de Intel, 68HC11 de Motorola y la familia PIC de Microchip. La firma también produce y vende varios subproductos de la popular familia 8051 con la diferencia de que están basados en la memoria flash.

El diseño AVR de ATMEL difiere de los demás microcontroladores de 8 bits por tener mayor cantidad de registros (32) y un conjunto ortogonal de instrucciones. AVR es mucho más moderna que su competencia. Por ejemplo, los 8051, 6805 y los PIC, se los

arreglan con un único acumulador, los 658HC11 y 68HC12 tienen simplemente 2. Esto hace que la arquitectura AVR sea más fácil de programar a nivel de lenguaje ensamblador y que sea fácil de optimizar con un compilador. El gran conjunto de registros disminuye la dependencia respecto a la memoria, lo cual mejora la velocidad y disminuye las necesidades de almacenamiento de datos. Además casi todas las instrucciones se ejecutan en 1 ó 2 ciclos de reloj versus 5-10 ciclos de reloj para los chips 8051, 6805, 68HC11 y PIC.

Adicionalmente, ATMEL también proporciona en línea el entorno software (AVR estudio) que permite editar, ensamblar y simular el código fuente. Una vez ensamblado y depurado el código fuente del programa, se transferirá el código máquina a la memoria flash del microcontrolador para esto se debe disponer de otro entorno de desarrollo para programar en forma serial o paralelo la memoria flash.

A continuación se va a tratar sobre los recursos de hardware y software disponibles en la familia de microcontroladores de ATMEL, tales como Avr Studio 4, WinAvr, Avr Butterfly, las principales características del hardware y el firmware, además del Proteus que es una herramienta de software muy necesaria para la simulación de los proyectos.

2.2 AVR STUDIO 4

AVR Studio es un Entorno de Desarrollo Integrado (IDE) para escribir y depurar aplicaciones AVR en el entorno de Windows 9x/Me/NT/2000/XP/7.

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

- ✓ La definición del producto. El producto que debe crearse se define basándose en el conocimiento de la tarea que se quiere resolver y la entrada que tendrá en el mercado.
- ✓ La especificación formal. Se define una especificación formal para el producto.
- ✓ Asignación de la tarea a un equipo. A un equipo del proyecto, que consiste de una o más personas, se le asigna la tarea de crear el producto basándose en la especificación formal.
- ✓ El equipo del proyecto pasa por la secuencia normal de diseño, desarrollo, depuración, comprobación, planificación de producción, producción, prueba y embarque.

AVR Studio apoya al diseñador en el diseño, desarrollo, depuración y parte de la comprobación del proceso. AVR Studio es actualizado continuamente y está disponible para descargarlo desde www.atmel.com.

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

- ✓ Windows 7
- ✓ Windows 2000/XP (o Windows NT 4.0 con Internet Explorer 5.0 o posterior).
- ✓ Windows 95/98/Me (con Internet Explorer 5.0 o posterior).
- ✓ Hardware Recomendado:
  - ❖ Procesador Intel Pentium de 200MHz o equivalente.
  - ❖ Resolución de Pantalla de 1024x768 (Resolución mínima de 800x600).

- Memoria RAM de 64Mb.
- Disco Duro con 50 Mb de espacio disponible.

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.

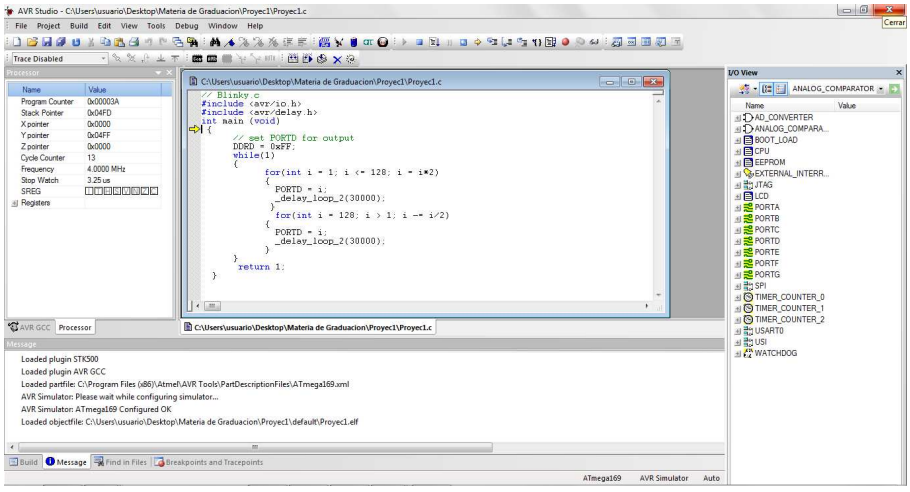


Figura 2.2: Ventana Principal IDE

2.3 WIN AVR

La distribución WinAvr, que es una recopilación de programas de software libre diseñados para facilitar las tareas de programación y desarrollo de los microcontroladores Avr. Dicha distribución WinAvr incorpora además del compilador gcc de consola, un editor de texto especialmente diseñado para ayudar al programador y hacer el código más legible mediante su resaltado con colores.

El programador vía puerto serie mprog.exe, que permite transferir el programa compilado, que se encuentra en un archivo llamado main.hex, a la memoria flash del microcontrolador utilizando únicamente un cable de tres líneas.

En pocas palabras WinAVR es un conjunto de herramientas de desarrollo para microcontroladores RISC AVR de Atmel, basado en software de código abierto y compilado para funcionar en la plataforma Microsoft Windows. WinAVR incluye las siguientes herramientas:

- ✓ avr-gcc, el compilador de línea de comandos para C y C++.
- ✓ avr-libc, la librería del compilador que es indispensable para avr-gcc.
- ✓ −avr-as, el ensamblador.
- ✓ avrdude, la interfaz para programación.
- ✓ avarice, la interfaz para JTAG ICE.
- ✓ avr-gdb, el depurador.
- ✓ Programmers Notepad, el editor.

- ✓ MFile, generador de archivo makefile.

La distribución WinAvr surge como una iniciativa para impulsar el desarrollo de software libre en el campo del desarrollo de los microcontroladores Avr de Atmel, a la vez de facilitar a sus potenciales usuarios el conocer la existencia de tales herramientas, y no sólo las más famosas, y también facilitar su actualización. Es por ello que dicho paquete de software incorpora no sólo dichas herramientas, sino además unos ficheros de ayuda y de descripción de cada una de las utilidades. Por supuesto, el soporte técnico es mayor según la herramienta tenga más importancia.

2.4 AVR BUTTERFLY

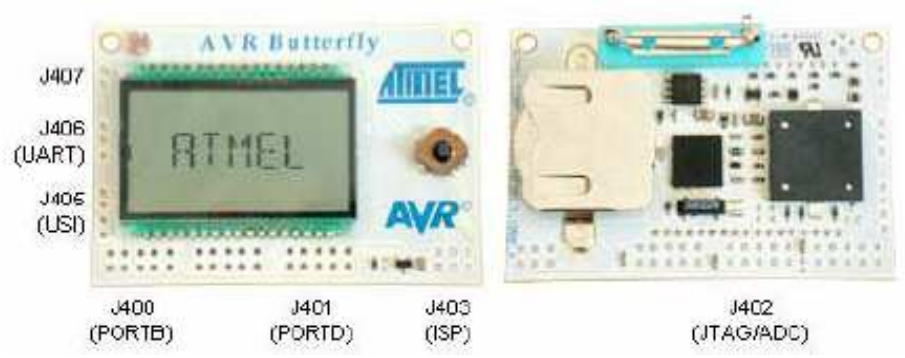


Figura 2.4: Avr Butterfly

El Kit AVR Butterfly (Figura 2.4) 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.

El Kit AVR Butterfly expone las siguientes características principales:

- ✓ La arquitectura AVR en general y la ATmega169 en particular.
- ✓ Diseño de bajo consumo de energía.
- ✓ El encapsulado tipo MLF.
- ✓ Periféricos:
  - Controlador LCD.
  - Memorias:
  - Flash, EEPROM, SRAM.
  - DataFlash externa.
- ✓ Interfaces de comunicación:
  - UART, SPI, USI.
- ✓ Métodos de programación
  - Self-Programming/Bootloader, SPI, Paralelo, JTAG.
- ✓ Convertidor Analógico Digital (ADC).
- ✓ Timers/Counters:
  - Contador de Tiempo Real (RTC).
  - Modulación de Ancho de Pulso (PWM).

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

2.4.1 HARDWARE DISPONIBLE

Los siguientes recursos están disponibles en el Kit AVR Butterfly, tal como se muestra en el diagrama de bloques de la Figura 2.4.1:

- ✓ Microcontrolador ATmega169V (en encapsulado tipo MLF).
- ✓ Pantalla tipo vidrio LCD de 120 segmentos, para demostrar las capacidades del controlador de LCD incluido dentro del ATmega169.
- ✓ Joystick de cinco direcciones, incluida la presión en el centro.
- ✓ Altavoz piezoeléctrico, para reproducir sonidos.
- ✓ Cristal de 32 KHz para el RTC.
- ✓ Memoria DataFlash de 4 Mbit, para el almacenar datos.
- ✓ Convertidor de nivel RS-232 e interfaz USART, para comunicarse con unidades fuera del Kit sin la necesidad de hardware adicional.
- ✓ Termistor de Coeficiente de Temperatura Negativo (NTC), para sensar y medir temperatura.
- ✓ Resistencia Dependiente de Luz (LDR), para sensar y medir intensidad luminosa.
- ✓ Acceso externo al canal 1 del ADC del ATmega169, para lectura de voltaje en el rango de 0 a 5 V.
- ✓ Emulación JTAG, para depuración.
- ✓ Interfaz USI, para una interfaz adicional de comunicación.
- ✓ Terminales externas para conectores tipo Header, para el acceso a periféricos.
- ✓ Batería de 3 V tipo botón (600mAh), para proveer de energía y permitir el funcionamiento del AVR Butterfly.
- ✓ Bootloader, para programación mediante la PC sin hardware especial.
- ✓ Aplicación demostrativa preprogramada.

- ✓ Compatibilidad con el Entorno de Desarrollo AVR Studio 4.

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

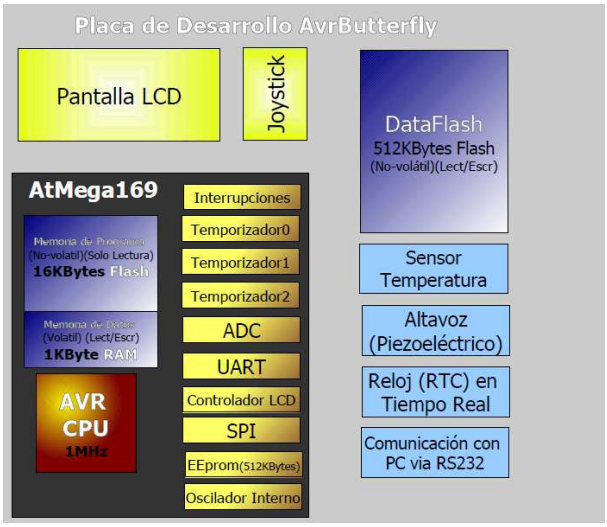


Figura 2.4.1: Diagrama de bloques Avr Butterfly

2.4.2 FIRMWARE INCLUIDO

El AVR Butterfly viene con una aplicación pre programada. 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 (Bootloader Code).
- ✓ 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.

2.5 CARACTERÍSTICAS DEL MICROCONTROLADOR ATMEGA169

El ATmega169 es un Microcontrolador de 8 bits con arquitectura AVR RISC, este posee las siguientes características:

- ✓ Arquitectura RISC avanzada.
- ✓ Conjunto de 130 instrucciones ejecutables en un solo ciclo de reloj.
- ✓ 32 x 8 registros de trabajo de propósito general.
- ✓ Rendimiento de hasta 16 MIPS a 16 MHz.
- ✓ Memoria no volátil para Programa y Datos.
  - Flash de 16 K bytes, auto-programable en el sistema.

    Resistencia: 10 000 ciclos de Escritura/Borrado.
  - Sección Opcional para Código de Arranque con Lock Bits Independientes.

    Programación en el Sistema a través del Programa de Arranque residente en el chip.

    Operación Real de Lectura Durante la Escritura.
  - EEPROM de 512 bytes.

    Resistencia: 100 000 ciclos de Escritura/Borrado.
  - SRAM Interna de 1 Kbyte.
  - Bloqueo de Programación para Seguridad del Software.
- ✓ Interfaz JTAG (conforme el Standard IEEE 1149.1)
  - Capacidad de Boundary-Scan Acorde al Standard JTAG.
  - Soporta Depuración On-chip.
  - Programación de la FLASH, EEPROM, Fusibles y Lock Bits a través de la Interfaz JTAG.
- ✓ Características de los Periféricos.
  - 6 puertos de I/O de 8-bits y 1 de 5-bits.
  - Controlador de LCD de 4 x 25 segmentos.

- Dos Temporizadores/Contadores de 8 bits con Preajustador (Prescaler) Separado y Modo de Comparación.
- Un Temporizador/Contador de 16 bits con Preajustador (Prescaler) Separado, Modo de Comparación y Modo de Captura.
- Contador de Tiempo Real con Oscilador Separado.
- Cuatro canales PWM.
- Ocho canales ADC de 8 bits cada uno.
- Interfaz Serial USART Programable.
- Interfaz Serial SPI Maestro/Esclavo.
- Interfaz Serial Universal con Detector de Condición de Inicio.
- WatchDog Timer Programable con Oscilador Separado incluido en el chip.
- Comparador Analógico.
- Interrupción y Salida del Modo de Sleep por Cambio en Pin.

✓ Características especiales del microcontrolador.

- Reset al Encendido y Detección Brown-Out Programable.
- Oscilador Interno Calibrable.
- Fuentes de Interrupción Internas y Externas.
- Cinco Modos de Sleep: Idle, ADC Noise Reduction, Power Save, Power-Down y Standby.

✓ Entradas/Salidas y Tipo de Encapsulado

- 53 Líneas de I/O Programables.
- 64 patillas en el encapsulado TQFP y 64 pads (para montaje superficial) en el encapsulado QFN/MLF.

✓ Rangos de Velocidad

- ATmega169V: 0 – 4 MHz a 1.8 – 5.5 V, 0 – 8 MHz a 2.7 – 5.5 V.
- ATmega169: 0 – 8 MHz a 2.7 – 5.5 V, 0 – 16 MHz a 4.5 – 5.5 V.

✓ Rango de Temperatura

- Desde -40 º C a 85 ºC.

✓ Consumo de Energía

- En el Modo Activo: 1 MHz, 1.8 V: 350 uA.

  32 KHz, 1.8 V: 20 uA (incluyendo Oscilador).

  32 KHz, 1.8 V: 40 uA (incluyendo Oscilador y LCD).

- En el Modo Power-Down:

  0.1 uA a 1.8 V.

El AVR ATmega169 es compatible con un completo conjunto de programas y Herramientas de Desarrollo que incluye: Compiladores C, Ensambladores de Macro, Depurador/Simuladores de Programa, Emuladores de Circuito, Kits de Iniciación y Kits Evaluación.

2.6 PROTEUS

Proteus es una aplicación CAD, compuesta de tres módulos:

✓ ISIS (Intelligent schematic input sistem): Es el módulo de captura de esquemas.

✓ VSM (Virtual sistem modelling): Es el módulo de simulación incluyendo Prospice.

- ✓ ARES (Advanced routing modelling): Es el modulo para la realización de circuitos impresos (PCB).

El modulo ISIS es un programa que nos permite dibujar sobre un área de trabajo un circuito que posteriormente podremos simular.[6]

En la manipulación del software casi siempre existirán varias opciones para un mismo fin, normalmente podremos optar por seguir un menú, acceder a un icono o trabajar en el teclado.

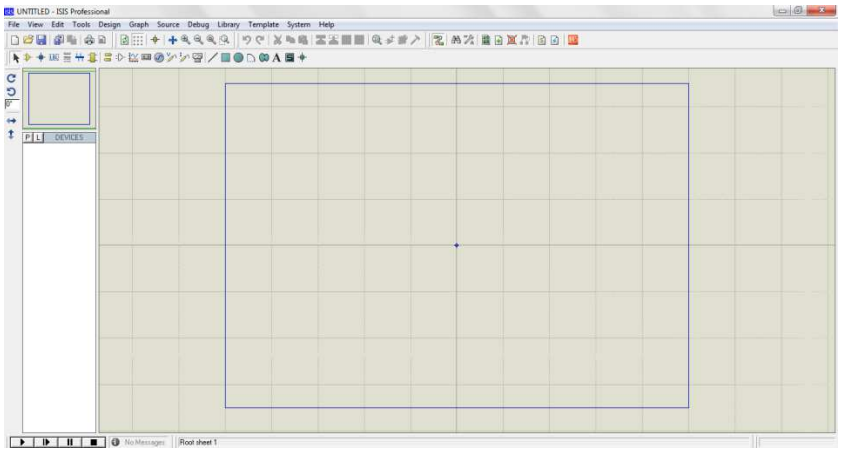


Figura 2.6 Ventana de Trabajo de Proteus

El menú permite acceder a la mayor parte de opciones del programa, sin embargo algunas solo están disponibles en los iconos de las barras de herramientas. Las barras de herramientas son varias y se pueden colocar en cualquier parte de la pantalla. El área de trabajo es donde realizaremos nuestros circuitos, ver Figura 2.6. [7]

Al abrir Proteus, lo primero que necesitamos es extraer los componentes que se van a utilizar en el circuito, para lo que debemos usar la barra de herramientas de componentes. Para acceder a una librería hay que presionar P que se encuentra en el extremo izquierdo, en las librerías son donde se encuentran los componentes y a partir de aquí se empieza a seleccionar los que necesitamos. Luego de armado nuestro circuito se lo guarda y se lo puede simular.

CAPITULO 3

PLATAFORMAS UTILIZADAS PARA DESARROLLAR EL PROYECTO

3.1 INTRODUCCIÓN

Para desarrollar este proyecto se utilizó dos plataformas que en conjunto nos permitió obtener una plataforma interactiva para realizar la demostración de los principales atributos de la comunicación UART, entre las plataformas que se utilizaron tenemos la estación de trabajo educativa y el conector serial DB9 , esto en conjunto con la tarjeta de desarrollo Avr Butterfly forman parte de la plataforma interactiva educativa, además se muestran los enunciados y listados de componentes de los ejercicios que se van a probar en la plataforma interactiva.

3.1.1 ESTACION DE TRABAJO EDUCATIVA PARA PRUEBAS EXPERIMENTALES

Es una placa de uso genérico reutilizable, o semipermanente usado para construir prototipos de circuitos electrónicos con o sin soldadura. Normalmente se utilizan para la realización de pruebas experimentales (Figura 3.1.1**)**. Además de los protoboard plásticos, libres de soldadura, también existen en el mercado otros modelos de placas de prueba.

La tableta experimental, es una herramienta que nos permite interconectar elementos electrónicos, ya sean resistencias, capacidades, semiconductores, etc., sin la necesidad de soldar los componentes. Está lleno de orificios metalizados –con contactos de presión- en los cuales se insertan las componentes del circuito a ensamblar.

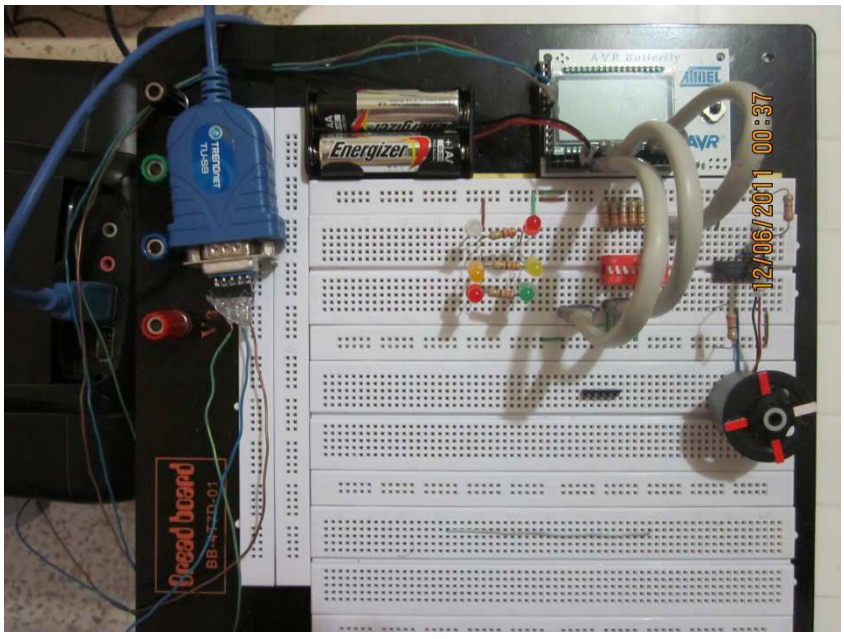


Figura 3.1.1. Plataforma Interactiva

3.1.2 CONECTOR DB9 DEL PC

En los PC´s hay conectores DB9 macho (Figura 3.1.2.1), de 9 pines, por el que se conectan los dispositivos al puerto serie. Los conectores hembra que se enchufan tienen una colocación de pines diferente, de manera que se conectan el pin 1 del macho con el pin 1 del hembra, el pin2 con el 2, etc (Figura 3.1.2.2).

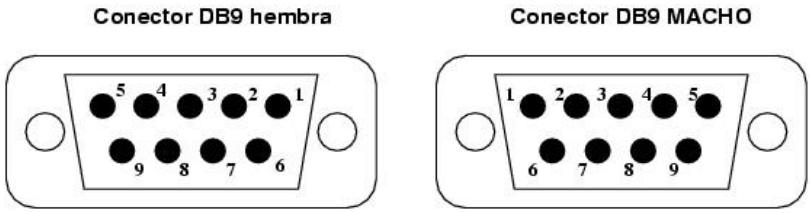


Figura 3.1.2.1 Conectores DB9.

La información asociada a cada uno de los pines es la siguiente:

Número de pin	Señal
1	**DCD** (Data Carrier Detect)
2	**RX**
3	**TX**
4	**DTR** (Data Terminal Ready)
5	**GND**
6	**DSR** (Data Sheet Ready)
7	**RTS** (Request To Send)
8	**CTS** (Clear To Send)
9	**RI** (Ring Indicator)

Tabla 3.1.2.2 Pines de Conector DB9

3.1.3 CONEXIÓN DE UN MICROCONTROLADOR AL PUERTO SERIE DE UNA PC.

Para conectar el PC a un microcontrolador por el puerto serie se utilizan las señales Tx, Rx y GND (Figura 3.1.3). El PC utiliza la norma RS232, por lo que los niveles de tensión de los pines están comprendidos entre +15 y -15 voltios. Los microcontroladores normalmente trabajan con niveles TTL (0-5v). Es necesario por tanto intercalar un circuito que adapte los niveles:

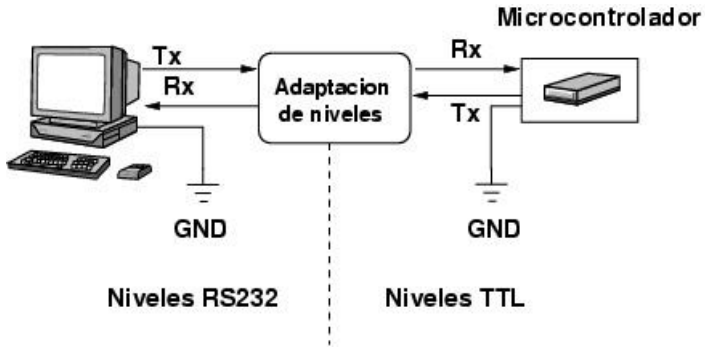


Figura 3.1.3: Esquema de conexión serie

3.2 EJERCICIOS A DESARROLLAR

3.2.1 EJERCICIO 1:

El Programa transmite un Dato desde la tarjeta AVR BUTTERFLY hacia el computador (Virtual Terminal), este dato se genera de acuerdo al valor que se selecciona en los Switchs y en el joystick. El programa funciona de la siguiente manera:

- En los 4 Switchs se podrá seleccionar las decenas en binario, estas pueden ir desde 0 hasta el 15, los cuales se representaran en el virtual terminal con su respectivo símbolo en código ASCII.
- En el Joystick del Butterfly se podrá seleccionar las unidades las cuales van desde el 0 hasta el 5 de la siguiente manera:
  - Presionando el botón del centro PUSH se obtiene el número 1
  - Presionando el botón UP se obtiene el número 2
  - Presionando el botón DOWN se obtiene el número 3
  - Presionando el botón LEFT se obtiene el número 4
  - Presionando el botón RIGHT se obtiene el número 5

3.2.1.1 LISTADO DE COMPONENTES

CANTIDAD	UNIDAD	DETALLE
1	u	Juego de botoneras de cuatro Switch
5	u	Resistencias de 330 Ω
1	u	Plataforma de Trabajo Interactiva
1	m	Cable Serial RS232 USB-DB9
2	u	Baterías AA (1.5 V)
1	u	Porta baterías
1	u	Tarjeta Electrónica AVR-BUTTERFLY

Tabla 3.2.1.1: Listado de Componentes Ejercicio 1

3.2.2 EJERCICIO 2:

El programa transmite datos desde la tarjeta electrónica AVR BUTTERFLY hasta el computador (Virtual Terminal), Al comenzar aparece en la pantalla LCD el mensaje MATERIA DE GRADUACION, y posteriormente si interactuamos con el Joystick se transmite el dato que se genera de la siguiente manera:

- Presionando en el Joystick el botón del centro, aparece en la pantalla Lcd la palabra “Stop” y se transmite la letra (p).
- Presionando en el Joystick el botón de la Izquierda, aparece en la pantalla Lcd la palabra “Izquierda” y se transmite letra (L).
- Presionando en el Joystick el botón de la Derecha, aparece en la pantalla Lcd la palabra “Derecha” y se transmite la letra (R).
- Presionando en el Joystick el botón de Arriba, aparece en la pantalla Lcd la palabra “Arriba” y se transmite la letra (U).
- Presionando en el Joystick el botón de Abajo, aparece en la pantalla Lcd la palabra “Abajo” y se transmite la letra (D).

3.2.2.1 LISTADO DE COMPONENTES

CANTIDAD	UNIDAD	DETALLE
1	u	Plataforma de Trabajo Interactiva
1	m	Cable Serial RS232 USB-DB9
2	u	Baterías AA (1.5 V)
1	u	Porta baterías
1	u	Tarjeta Electrónica AVR-BUTTERFLY

Tabla 3.2.2.1 Listado de componentes Ejercicio 2

3.2.3 EJERCICIO 3:

El siguiente diagrama representa el funcionamiento de un Motor para el cual se va ha controlar los sentidos de los giros y las paradas, al comenzar se espera por el dato que se genera al interactuar con el Joystick en la cual dependiendo de la posición presionada se presenta la acción seleccionada en la pantalla LCD y se realiza la acción en el Motor de la siguiente manera:

- Si se presiona el botón izquierdo en el Joystick, el motor se mueve a la Izquierda
- Si se presiona el botón derecho en el Joystick, el motor se mueve a la Derecha.
- Si se presiona el botón de arriba en el Joystick, el motor aumenta la velocidad.
- Si se presiona el botón de abajo en el Joystick, el motor disminuye la velocidad.
- Si se presiona el botón del centro en el Joystick, el motor se empieza a detener.

3.2.3.1 LISTADO DE COMPONENTES

CANTIDAD	UNIDAD	DETALLE
1	u	Plataforma de Trabajo Interactiva
1	m	Cable Serial RS232 USB-DB9
2	u	Baterías AA (1.5 V)
1	u	Porta baterías
1	u	Tarjeta Electrónica AVR-BUTTERFLY
1	u	Motor de Paso

Tabla 3.2.3.1 Listado de componentes Ejercicio 3

3.2.4 EJERCICIO 4:

El programa transmite datos desde la plataforma interactiva, que está compuesta por la tarjeta electrónica AVR BUTTERFLY y Led's, hasta el computador (Virtual Terminal), El dato que se transmite se genera cada vez que se cumple un ciclo repetitivo en el cual se van activando alternadamente las salidas PD5, PD6 y PD7 del Port D, las cuales se pueden apreciar por los Led's que están conectados a dichas salidas, Al comenzar transmite el dato en código Ascii que representa la letra A, y cada vez que se cumple un ciclo se incrementa en uno y se va transmitiendo los valores en código Ascii.

3.2.4.1 LISTADO DE COMPONENTES

CANTIDAD	UNIDAD	DETALLE
1	u	Plataforma de Trabajo interactiva
1	m	Cable Serial RS232 USB-DB9
2	u	Baterías AA (1.5 V)
1	u	Porta baterías
1	u	Tarjeta Electrónica AVR-BUTTERFLY
3	u	Led´s
3	u	Resistencias 330 Ω

Tabla 3.2.4.1 Listado de componentes Ejercicio 4

CAPITULO 4

EJERCICIOS A DESARROLLAR

4.0 INTRODUCCIÓN

A continuación vamos a enunciar los ejercicios que serán probados en la plataforma interactiva educativa, la cual va a permitir demostrar el funcionamiento y los recursos disponibles en la comunicación UART.

4.1 EJERCICIO 1:

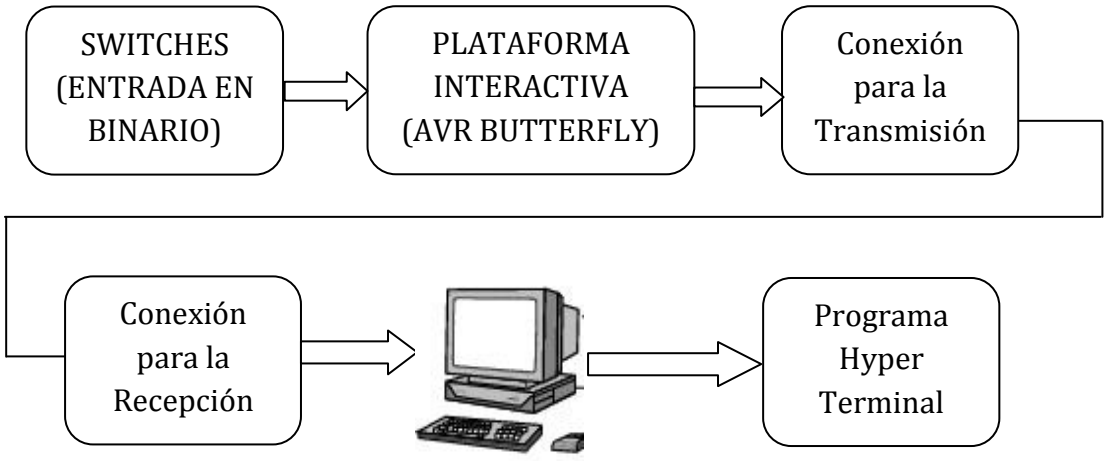
4.1.1 ESPECIFICACIÓN

El Programa transmite un Dato desde la tarjeta AVR BUTTERFLY hacia el computador (Virtual Terminal), este dato se genera de acuerdo al valor que se selecciona en los Switchs y en el joystick. El programa funciona de la siguiente manera:

- En los 4 Switchs se podrá seleccionar las decenas en binario, estas pueden ir desde 0 hasta el 15, los cuales se representaran en el virtual terminal con su respectivo símbolo en código ASCII.

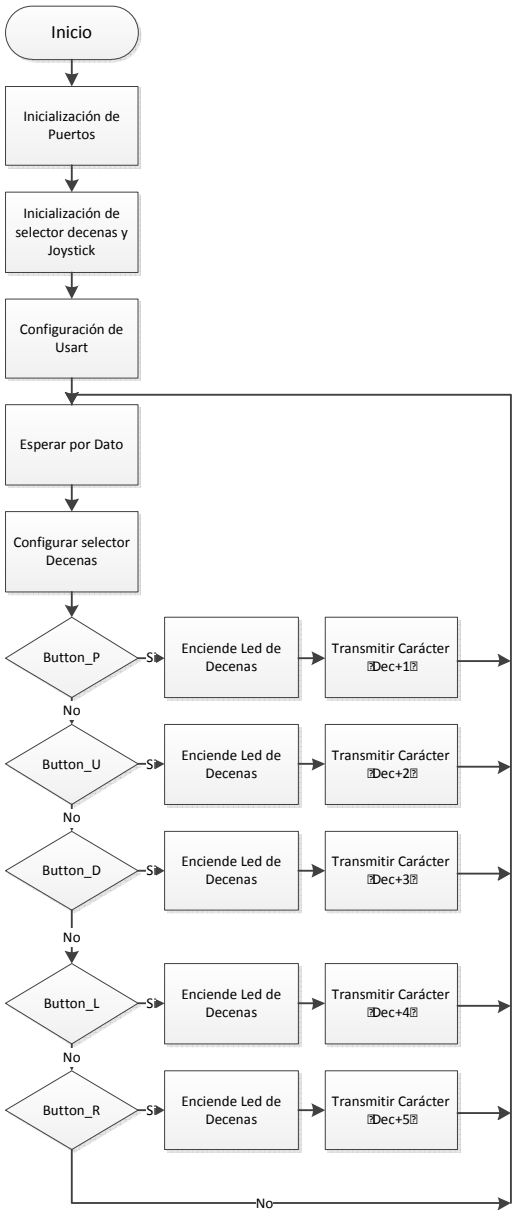
- En el Joystick del Butterfly se podrá seleccionar las unidades las cuales van desde el 0 hasta el 5 de la siguiente manera:
  - Presionando el botón del centro PUSH se obtiene el número 1
  - Presionando el botón UP se obtiene el número 2
  - Presionando el botón DOWN se obtiene el número 3
  - Presionando el botón LEFT se obtiene el número 4
  - Presionando el botón RIGHT se obtiene el número 5

4.1.2 DIAGRAMA DE BLOQUE



4.1.3 DIAGRAMA DE FLUJO

El siguiente diagrama de flujo representa el funcionamiento de un programa que al comenzar inicializa los puertos como entradas o salidas, además configura el Joystick que vamos a utilizar en la ejecución del programa, a continuación se configura la comunicación Usart que va ha permitir la comunicación entre la PC y la Plataforma Interactiva, se espera por el dato que se selecciona en los switchs y se configura como decenas, dependiendo de la posición que se selecciona en el joystick se enciende el led y se transmiten las unidades y las decenas en código Ascii hacia la PC.



4.1.4 CÓDIGO

```
/* ---------------------------MATERIA DE GRADUCACIÓN-----------------------------------

 * Profesor: Ing. Carlos Valdivieso A.

 * Creado: 14/12/2011 15:08:57

 * Autor: Frank Benalcázar & Jorge Arévalo

  */-------------------------------------------------------------------------------------------------------

////////////////////////////////////////////////Librerias a usar/////////////////////////////////////////////////////////

#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>

////////////////////////////////////////////////Declaración de constantes/////////////////////////////////////////////////

#define PINB_MASK ((1<<PINB4)|(1<<PINB6)|(1<<PINB7))

#define PINE_MASK ((1<<PINE2)|(1<<PINE3))



#define BUTTON_A 6 // UP

#define BUTTON_B 7 // DOWN
```

```
#define BUTTON_C 2 // LEFT

#define BUTTON_D 3 // RIGHT

#define BUTTON_O 4 // PUSH

//////////////////////////////////////////Librerías para utilizar LCD//////////////////////////////////////////

#include "lcd_functions.h"

#include "lcd_driver.h"

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

//////////////////////////Prototipo de procedimientos para la transmisión//////////////////////////

void InitUART (unsigned char baudrate);

unsigned char ReceiveByte (void);

void TransmitByte (unsigned char data);

void PinChangeInterrupt(void);

int indice,selector;

char buttons_;

char key;

//////////////////////////////////////////////////////////Mensaje Inicial//////////////////////////////////////////////

PGM_P MENSAJE1;

PGM_P MENSAJE2;

int main(void)

{

        LCD_Init();
```

```
        MENSAJE1 = PSTR("Plataforma Interactiva");

        sei();

////////////////////////////////////// Ajuste de la velocidad de transmisión ////////////////////////////////

        InitUART (51);     /* Set the baudrate to 9600 bps using a 8MHz crystal */

////////////////////////////////////////////////// Seteo de puertos//////////////////////////////////////////////////

        DDRB |= 0XD8;

        PORTB |= PINB_MASK;

        DDRE = 0x00;

        PORTE |= PINE_MASK;

/////////////////////// Habilita Pines de interrupción en PORTB and PORTE /////////////////

        PCMSK0 = PINE_MASK;

        PCMSK1 = PINB_MASK;

        EIFR   = (1<<6)|(1<<7);

        EIMSK  = (1<<6)|(1<<7);

        SREG   = 0X80;              //////// Habilita las interrupciones.

        DDRD  = 0xFF;               /////// Seteo del PORTD como salida

        DDRB  = 0X00;               /////// Seteo del PORTB como entrada

        PORTB = 0xFF;               /////// Activa las resistencias pullup en las en la entrada

       PORTD = 0xFF;               /////// Seteo de los LEDs en off

  _delay_ms(23);

    LCD_puts_f(MENSAJE1,1);
```

```
    for (;;) {
            indice= ReceiveByte ();
            TransmitByte (indice);
            TransmitByte (' ');
    for (indice = 0; indice < 200; indice++);
             _delay_ms(23);              // PORTB = (1<<INDICADOR);
             _delay_ms(23);             //PORTB = (0<<INDICADOR);
  }
  return 0;
}
////////////////////////********** Inicializa UART **********////////////////////
void InitUART (unsigned char baudrate)
{
////////////////////////////////////////////* Establecer la velocidad *////////////////////////////////////////
 UBRRL = baudrate;
 ////////////////////////////////* Habilita UART Receptor y Transmisor *//////////////////////////
 UCSRB = (1 << RXEN) | (1 << TXEN);
////////////////////////////////////* Fija 8 bits de datos, 1 bit de parada *//////////////////////////////
   UCSRC = (0<<USBS) | (0 << UCSZ2) | (1 << UCSZ1) | (1 << UCSZ0);
}
////////////////////////////////////////////*Implementación de función*////////////////////////////////////////
```

//////*Función que espera a la recepción de un carácter y luego retorna el dato recibido al programa principal*//////

```
unsigned char ReceiveByte (void)
{
  /* Wait for incomming data */
  while (!(UCSRA & (1 << RXC)));     //////////////* Espera por lo datos entrantes
  return UDR;                                  //////////////* Retorna el Dato
}
void TransmitByte (unsigned char data)
{
  while (!(UCSRA & (1 << UDRE)));   /// Espera a que el Buffer de transmisión este vacío
  UDR = data;                                  /// Inicia la Transmisión
}
        SIGNAL(SIG_PIN_CHANGE0)
        {
                PinChangeInterrupt();
        }
                SIGNAL(SIG_PIN_CHANGE1)
        {
                PinChangeInterrupt();
        }
        void PinChangeInterrupt(void)
```

```
{
        buttons_ = (~PINB) & PINB_MASK;
        buttons_ |= (~PINE) & PINE_MASK;
```

///// **Botón Arriba marca la unidad en dos y las decenas la marca el interruptor** /////

```
        if (buttons_ & (1<<BUTTON_A))
        {
                PORTD = PINB & 0X0F;
                selector=PORTD;
                selector+=48;
                TransmitByte (selector);
                _delay_ms(1000);
                indice=50;
                TransmitByte (indice);
        }
```

////// **Botón bajo marca la unidad en tres y las decenas la marca el interruptor**////////

```
        if (buttons_ & (1<<BUTTON_B))
        {
                PORTD = PINB & 0X0F;
                selector=PORTD;
                selector+=48;
                TransmitByte (selector);
```

```
                _delay_ms(1000);

                indice=51;

                TransmitByte (indice);

        }

////// Botón Izquierda marca la unidad en cuatro y las decenas la marca el interruptor

        if (buttons_ & (1<<BUTTON_C))

        {

                PORTD = PINB & 0X0F;

                selector=PORTD;

                selector+=48;

                TransmitByte (selector);

                _delay_ms(1000);

                indice=52;

                TransmitByte (indice);

        }

////// Botón Derecha marca la unidad en cinco y las decenas la marca el interruptor/////

        if (buttons_ & (1<<BUTTON_D))

        {

                PORTD = PINB & 0X0F;

                selector=PORTD;

                selector+=48;
```

```
                TransmitByte (selector);
                _delay_ms(1000);
                indice=53;
                TransmitByte (indice);
        }
```

////////// Botón Centro marca la unidad en Uno y las decenas la marca el interruptor//////

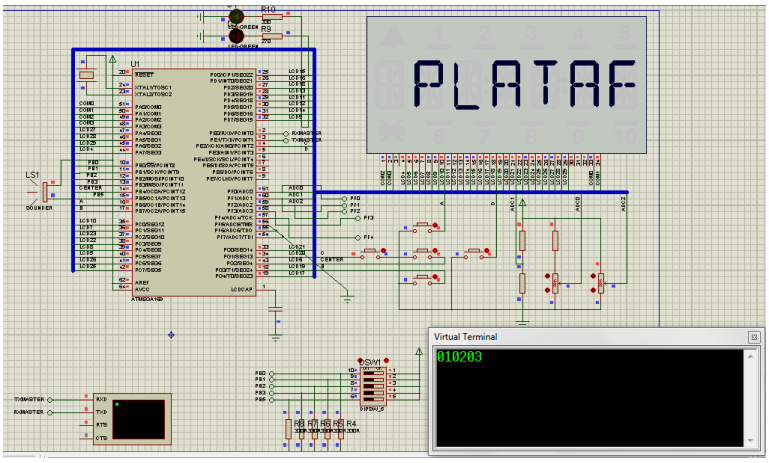
```
        if (buttons_ & (1<<BUTTON_O))
        {
                PORTD = PINB & 0X0F;
                selector=PORTD;
                selector+=48;
                TransmitByte (selector);
                _delay_ms(1000);
                indice=49;
                TransmitByte (indice);
        }
```

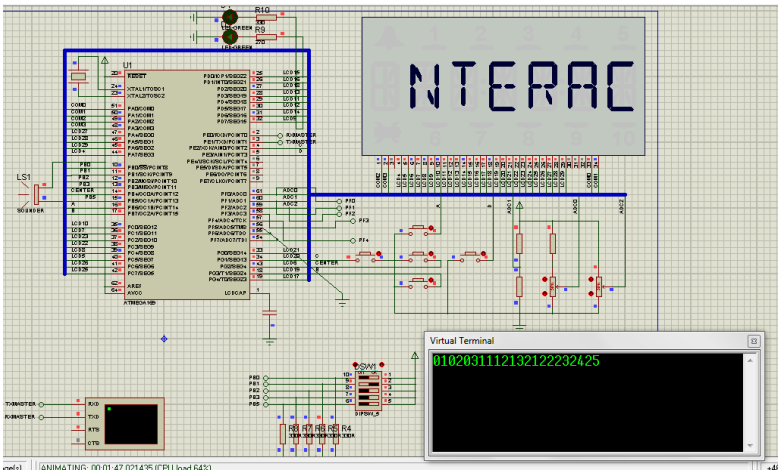
////////////////// Setea la bandera de interrupción//

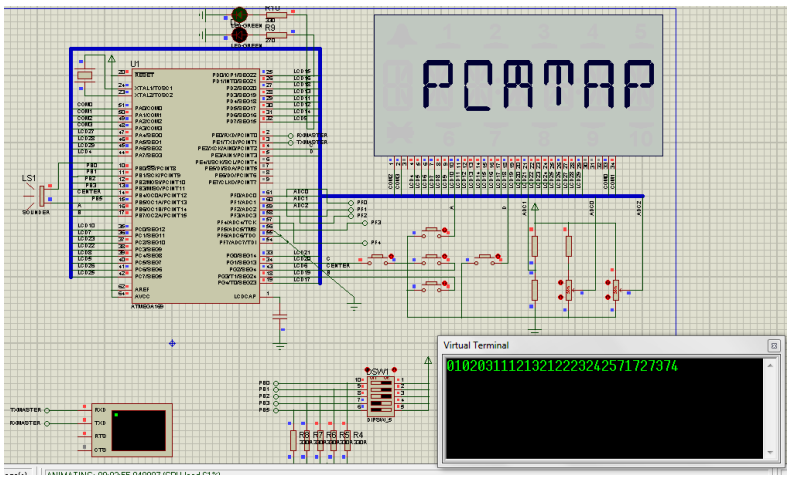
```
        EIFR = (1<<PCIF1) | (1<<PCIF0);
        cli();
        sei();
        }
```

4.1.5 SIMULACIONES

Primeramente se escogen las decenas en los switch, luego se espera hasta seleccionar la posición en el joystick, y dependiendo de esto se transmiten las decenas con las unidades seleccionadas en el joystick en código Ascii hacia la PC.







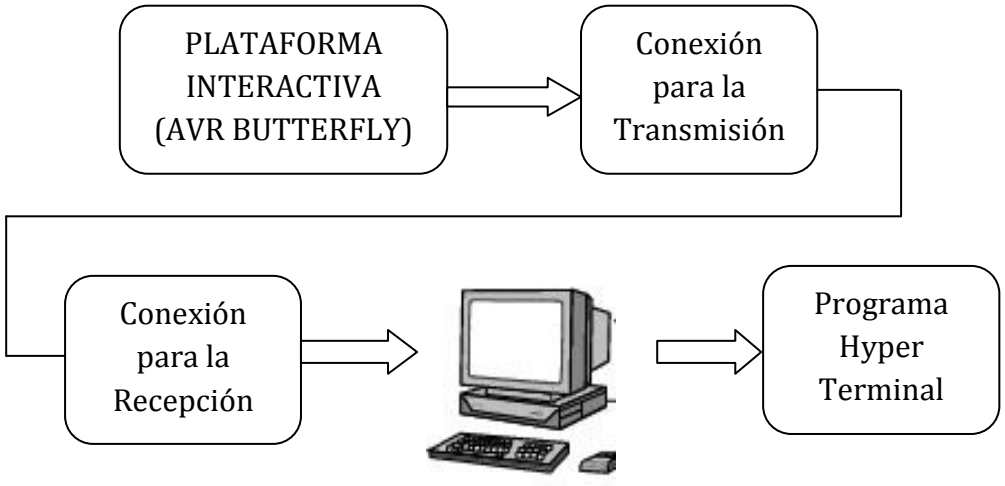
4.2 EJERCICIO 2:

4.2.1 ESPECIFICACIÓN

El programa transmite datos desde la tarjeta electrónica AVR BUTTERFLY hasta el computador (Virtual Terminal), Al comenzar aparece en la pantalla Lcd el mensaje "Materia de Graduación", y posteriormente si interactuamos con el Joystick se transmite el dato que se genera de la siguiente manera:

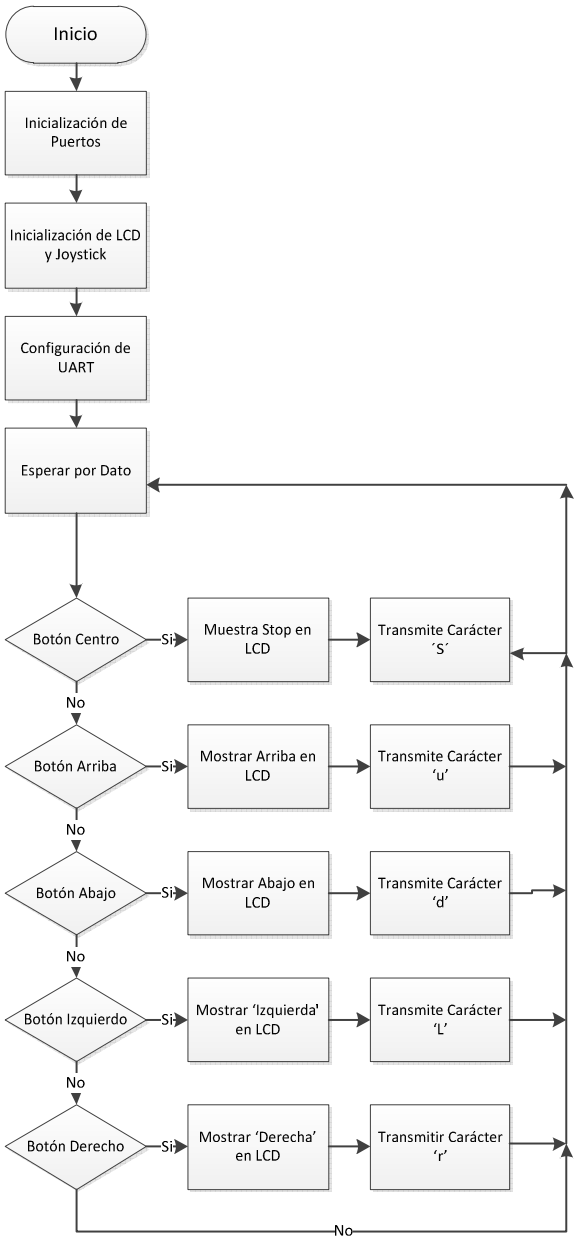
- Presionando en el Joystick el botón del centro, aparece en la pantalla Lcd la palabra "Stop" y se transmite la letra (P).
- Presionando en el Joystick el botón de la Izquierda, aparece en la pantalla Lcd la palabra "Izquierda" y se transmite la letra (L).
- Presionando en el Joystick el botón de la Derecha, aparece en la pantalla Lcd la palabra "Derecha" y se transmite la letra (R).
- Presionando en el Joystick el botón de Arriba, aparece en la pantalla Lcd la palabra "Arriba" y se transmite la letra (U).
- Presionando en el Joystick el botón de Abajo, aparece en la pantalla Lcd la palabra "Abajo" y se transmite la letra (D).

4.2.2 DIAGRAMA DE BLOQUES



4.2.3 DIAGRAMA DE FLUJO

El diagrama de flujo siguiente representa el funcionamiento de un programa que se desarrolla primeramente inicializando los puertos de entrada y salida además de la pantalla LCD y el Joystick, a continuación se configura la comunicación Uart para comunicarnos con la PC, y se espera por el dato que se va ha selecciona al presionar alguna posición del Joystick, luego de esto transmite hacia la PC el primer carácter de la posición seleccionada.



4.2.4 CÓDIGO

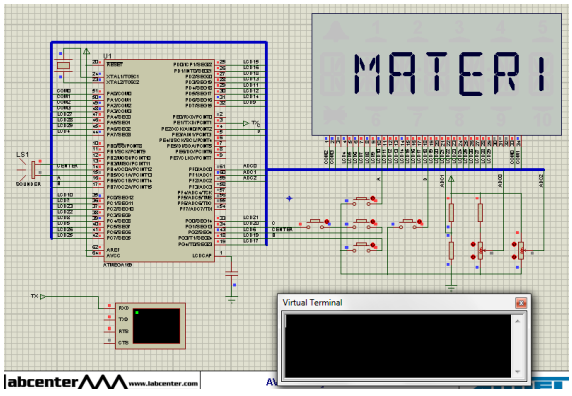
```
//////////////////////////////////////////////////////Librerias a usar//////////////////////////////////////////////////////////////

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <avr/delay.h>
#include <inttypes.h>

#include "mydefs.h"
#include "LCD_functions.h"
#include "LCD_driver.h"
#include "button.h"
#include "usart.h"

////////////////////////////////////////////////////////////// Inicio  ////////////////////////////////////////////////////////////
int main(void)
{
   PGM_P statetext = PSTR("MATERIA DE GRADUACION");
  uint8_t input;
  ACSR = (1<<ACD);            ////////////////// Deshabilita comparador analógico
  DIDR0 = (7<<ADC0D);       ////////////////// Deshabilita entrada digital en PF0-2
  PORTB = (15<<PB0);        ////////////////// Activa Pullups
  PORTE = (15<<PE4);
   Button_Init();                  //// Inicializa interrupción por cambio de pin en el Joystick
     LCD_Init();                   //// inicializa el LCD
  CLKPR = (1<<CLKPCE);   //// Activa el reloj del  Preescalador
//////////////////////////// seteo de preescalador = 8, Inter RC 8Mhz / 8 = 1Mhz ////////////////////////////////
   CLKPR = (0<<CLKPS1) | (1<<CLKPS0);
  USART_Init(25.04);
```

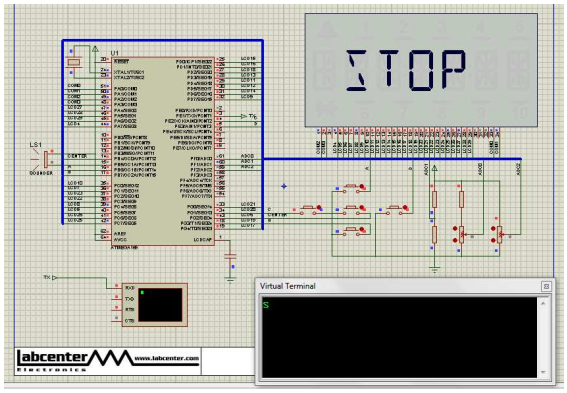
```
//sei();
while (1)
{
   if (statetext){
    LCD_puts_f(statetext, 1);
    LCD_Colon(0);
    statetext = NULL;
                    }
 input = getkey();          // Lee Botones
    switch (input) {
    case KEY_ENTER:
      statetext = PSTR("STOP");
      Usart_Tx('stop');
      break;
    case KEY_NEXT:
      statetext = PSTR("DERECHA");
      Usart_Tx('r');
      break;
    case KEY_PREV:
      statetext = PSTR("IZQUIERDA");
      Usart_Tx('l');
      break;
    case KEY_PLUS:
      statetext = PSTR("ARRIBA");
      Usart_Tx('u');
      break;
    case KEY_MINUS:
      statetext = PSTR("ABAJO");
      Usart_Tx('d');
      break;
```

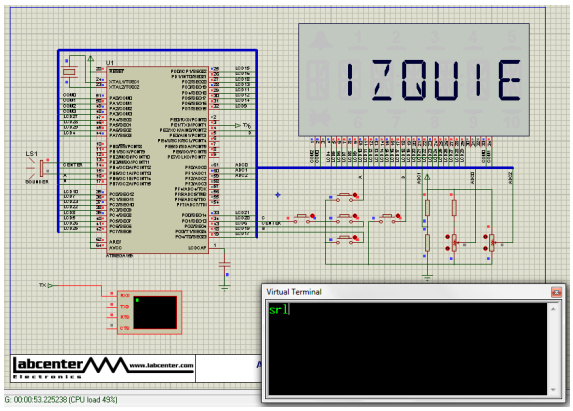
```
        default:
          break;
    }
```

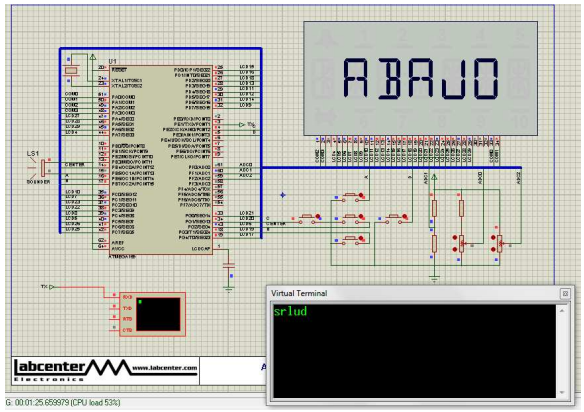
4.2.5 SIMULACIONES

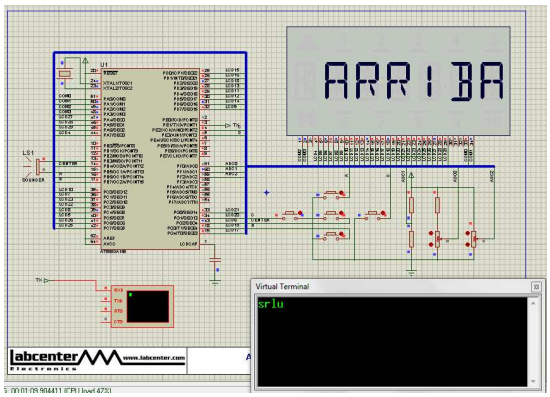
El programa transmite datos desde la tarjeta electrónica AVR BUTTERFLY hasta el computador (Virtual Terminal), Al comenzar aparece en la pantalla LCD el mensaje “Materia de Graduación”, y posteriormente si interactuamos con el Joystick se presenta el la pantalla LCD la posición que se presiono y en la del centro stop, se transmite el primer carácter de la palabra que se presenta en el LCD.

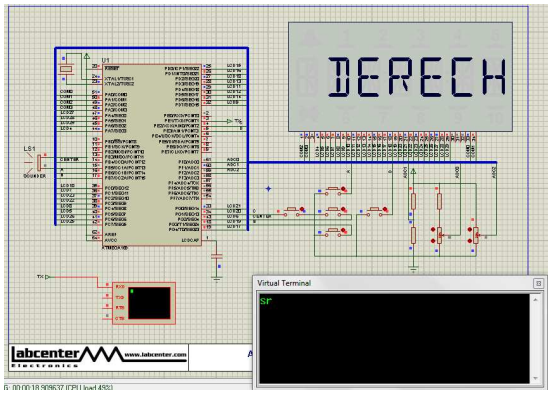












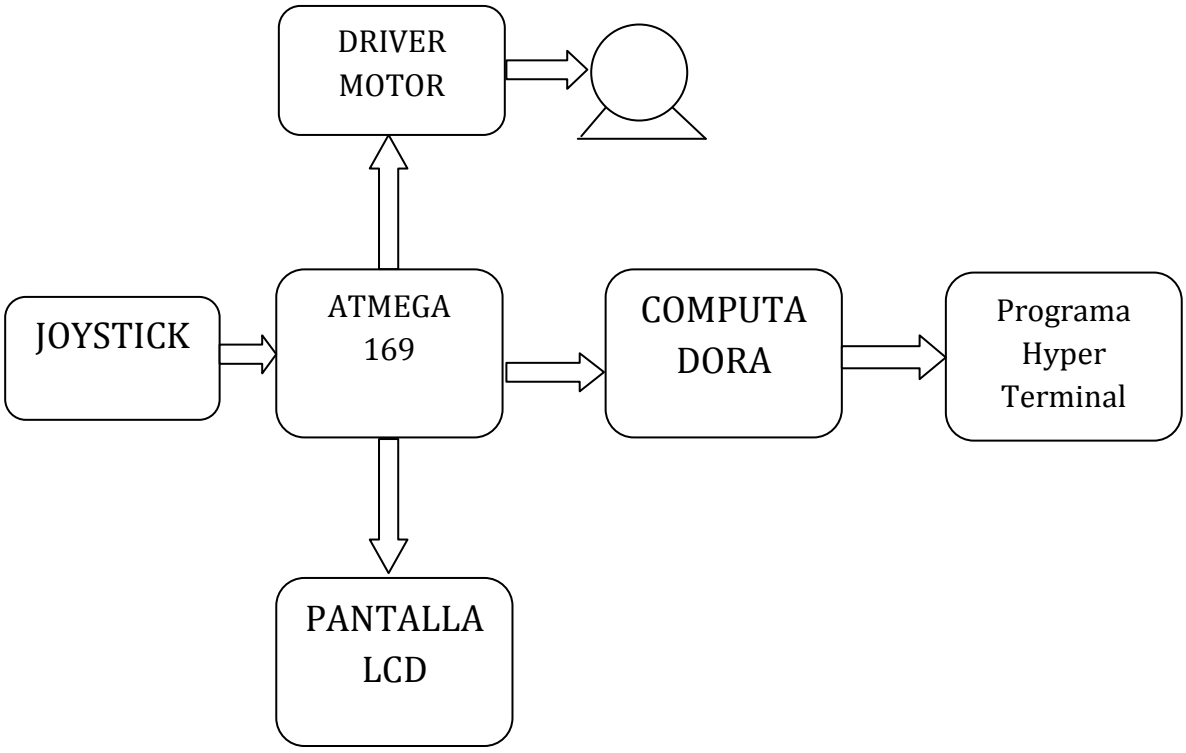
4.3 EJERCICIO 3:

4.3.1 ESPECIFICACIÓN

El siguiente diagrama representa el funcionamiento de un Motor para el cual se va ha controlar los sentidos de los giros y las paradas, al comenzar se espera por el dato que se genera al interactuar con el Joystick en la cual dependiendo de la posición presionada se presenta la acción seleccionada en la pantalla LCD y se realiza la acción en el Motor de la siguiente manera:

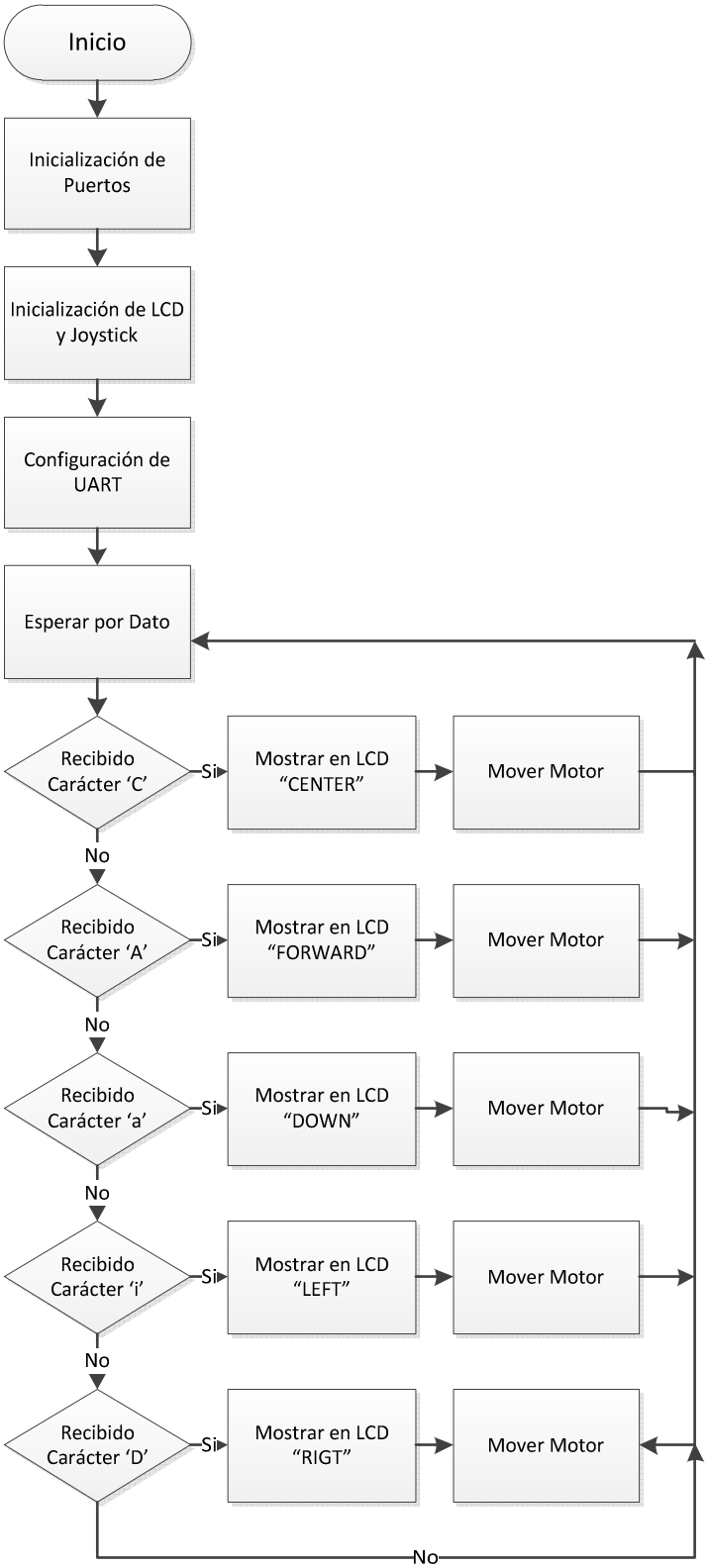
- Si se presiona el botón izquierdo en el Joystick, el motor se mueve a la Izquierda
- Si se presiona el botón derecho en el Joystick, el motor se mueve a la Derecha.
- Si se presiona el botón de arriba en el Joystick, el motor aumenta la velocidad.
- Si se presiona el botón de abajo en el Joystick, el motor disminuye la velocidad.
- Si se presiona el botón del centro en el Joystick, el motor se empieza a detener.

4.2.2 DIAGRAMA DE BLOQUES



4.2.3 DIAGRAMA DE FLUJO

El siguiente diagrama representa el funcionamiento de un Motor para el cual se va ha controlar los sentidos de los giros y las paradas, al comenzar se inicializan los puertos, la pantalla LCD y el Joystick, se configura la comunicación UART para comunicarnos con la PC y se espera por el dato que se genera al interactuar con el Joystick en la cual dependiendo de la posición presionada se presenta la acción seleccionada en la pantalla LCD y se realiza la acción en el Motor.


Inicio
Inicialización de Puertos
Inicialización de LCD y Joystick
Configuración de UART
Esperar por Dato
Recibido Carácter 'C'
Si
Mostrar en LCD "CENTER"
Mover Motor
No
Recibido Carácter 'A'
Si
Mostrar en LCD "FORWARD"
Mover Motor
No
Recibido Carácter 'a'
Si
Mostrar en LCD "DOWN"
Mover Motor
No
Recibido Carácter 'i'
Si
Mostrar en LCD "LEFT"
Mover Motor
No
Recibido Carácter 'D'
Si
Mostrar en LCD "RIGT"
Mover Motor
No

4.3.4 CÓDIGO

```
////////////////////////////////////////////////////////Librerias a usar////////////////////////////////////////////////////////////

#include <avr/io.h>

#include <avr/interrupt.h>

#include <avr/pgmspace.h>

#include <avr/delay.h>

#include <inttypes.h>

#include "mydefs.h"

#include "LCD_functions.h"

#include "LCD_driver.h"

#include "button.h"

#include "usart.h"

////////////////////////////////////////////////////////////* INICIO *////////////////////////////////////////////////////////////

void Initialization(void);

void Delay(unsigned int millisec);

void OSCCAL_calibration(void);

int main(void)

{

   PGM_P statetext = PSTR("BF JS Demo");

            uint8_t input, buttons;

            uint8_t last_buttons = KEY_NULL;
```

```
        int i;
Initialization();
     CLKPR = (1<<CLKPCE);        /////////////// set Clock Prescaler Change Enable
     CLKPR = (1<<CLKPS1) | (1<<CLKPS0); //// set Prescaler = 8, Inter RC 8Mhz / 8 = 1Mhz
       USART_Init(12);
sei();
for (;;)
{
   if (statetext) {
                        LCD_puts_f(statetext, 1);
                        LCD_Colon(0);
                        statetext = NULL;
                }
                input = getkey();          ////////////////// Leer los botones
                switch (input)
   {
                        case KEY_ENTER:
                                statetext = PSTR("ENTER");
                                Usart_Tx('1');
                                _delay_ms(100);
                                Usart_Tx('C');
```

```
            break;
        case KEY_NEXT:
            statetext = PSTR("RIGHT");
            Usart_Tx('1');
            _delay_ms(100);
            Usart_Tx('D');
            break;
        case KEY_PREV:
            statetext = PSTR("LEFT");
            Usart_Tx('1');
            _delay_ms(100);
            Usart_Tx('I');
            break;
        case KEY_PLUS:
            statetext = PSTR("UP");
            Usart_Tx('1');
            _delay_ms(100);
            Usart_Tx('A');
            break;
        case KEY_MINUS:
            statetext = PSTR("DOWN");
```

```
                                        Usart_Tx('1');
                                        _delay_ms(100);
                                        Usart_Tx('a');
                                        break;
                                default:
                                        break;
                        }
////////////////Verifica si el Joystick ha estado en la misma posicion por algun tiempo/////////////////
////////////////////////////////////////////// Activación de Auto Press del Joystick/////////////////////////////////////////
        buttons = (~PINB) & PINB_MASK;
        buttons |= (~PINE) & PINE_MASK;
             if( buttons != last_buttons )
        {
           last_buttons = buttons;
           gAutoPressJoystick = FALSE;
        }
        else if( buttons )
        {
           if( gAutoPressJoystick == TRUE)
           {
              PinChangeInterrupt();
```

```
            gAutoPressJoystick = AUTO;
          }
          else
            gAutoPressJoystick = AUTO;
       }
    }
          return 0;
}
/*****************************************************************************
*  Nombre de la Función: Initialization
*   Retorno:       Ninguno
*   Parámetros:    Ninguno
*   Propósito:     Inicializar los diferentes Módulos
*****************************************************************************/
void Initialization(void)
{
   OSCCAL_calibration();              ////////////////////// Calibra el OSCCAL byte
    CLKPR = (1<<CLKPCE);              /////////// setea Clock Prescaler Change Enable
   CLKPR = (1<<CLKPS1) | (1<<CLKPS0);  /// setea prescaler = 8, Inter RC 8Mhz / 8 = 1Mhz
   ACSR = (1<<ACD);                   ///////// Deshabilita Comparador analógico (power save)
   DIDR0 = (7<<ADC0D);                //////// Deshabilita entrada digital en PF0-2 (power save)
```

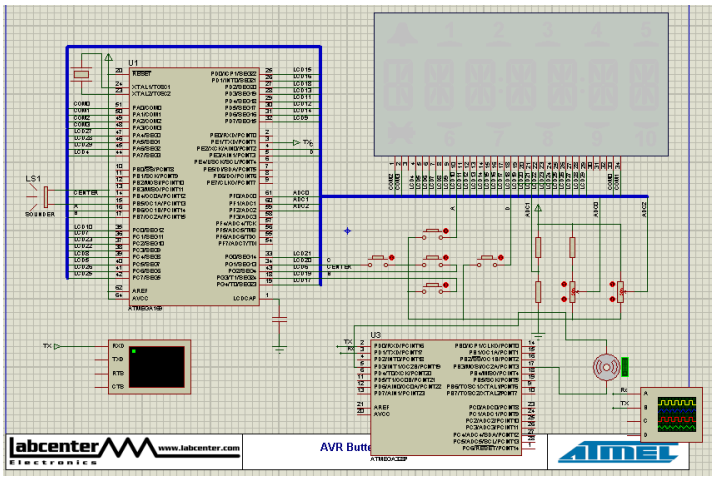
```
        PORTB = (15<<PB0);          ///////// Habilita pullups
        PORTE = (15<<PE4);
    Button_Init();              ////////// Inicializa interrupción por cambio de pin en el Joystick
   LCD_Init();                  ////////// inicializa LCD
}
/*****************************************************************************
*   Nombre la Función: Delay
*   Retorno: Ninguno
*   Propósito:    Lazo de  Delay
*****************************************************************************/
void Delay(unsigned int millisec)
{
 ////////////////////////////////// mt, int i did not work in the simulator:  int i;
        uint8_t i;
   while (millisec--) {
                for (i=0; i<125; i++) {
                        asm volatile ("nop"::);
                }
        }
}
/*****************************************************************************
```

```
* Nombre de la Función: OSCCAL_calibration

* Retorno:      Ninguno

* Proposito :     Calibra el byte del OSCCAL interno, usando el cristal externo de 32,786 KHZ como
referencia

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

void OSCCAL_calibration(void)

{

   unsigned char calibrate = FALSE;

   int temp;

   unsigned char tempL;

   CLKPR = (1<<CLKPCE);       ////// setea el reloj del prescalador

   // setea el  prescalador = 8, Inter RC 8Mhz / 8 = 1Mhz

   CLKPR = (1<<CLKPS1) | (1<<CLKPS0);

  TIMSK2 = 0;            //Deshabilita OCIE2A y TOIE2

   ASSR = (1<<AS2);       //selecciona operacion asynchronous de timer2 (32,768kHz)

  OCR2A = 200;           // setea valor de comparación timer2

 TIMSK0 = 0;            // Elimina alguna fuente de interrupción

 TCCR1B = (1<<CS10);    // Inicializa timer1 sin prescaling

 TCCR2A = (1<<CS20);    // inicializa timer2 sin prescaling

 while((ASSR & 0x01) | (ASSR & 0x04));   //Espera por TCN2UB y TCR2UB para ser borrado

   Delay(1000);    // Espera por por el cristal externo para estabilizarce

   while(!calibrate)
```

```
{
   cli(); // mt __disable_interrupt();    ///// deshabilitador global de interrupciones
   TIFR1 = 0xFF;   ////// Elimina la bandera TIFR1
   TIFR2 = 0xFF;   ////// Elimina la bandera TIFR2
   TCNT1H = 0;     ////// Limpia el contador  timer1
   TCNT1L = 0;
   TCNT2 = 0;      /////// Limpia el contador  timer2
   // shc/mt while ( !(TIFR2 && (1<<OCF2A)) );   ///// Espera por la bandera timer2
   while ( !(TIFR2 & (1<<OCF2A)) );                ////// Espera por la bandera timer2
   TCCR1B = 0; // stop timer1
   sei(); // __enable_interrupt();                 ///// Habilitador Global de Interrupciones
   // shc/mt if ( (TIFR1 && (1<<TOV1)) )
   if ( (TIFR1 & (1<<TOV1)) )
   {
      temp = 0xFFFF;      // if timer1 overflows, set the temp to 0xFFFF
   }
   else
   {   ////////////////////// Lee fuera del valor del contador  timer1
      tempL = TCNT1L;
      temp = TCNT1H;
      temp = (temp << 8);
```

```
        temp += tempL;

      }

      if (temp > 6250)

      {

        OSCCAL--;   //////////// El Oscilador corre rapido, decrese el OSCCAL

      }

      else if (temp < 6120)

      {

        OSCCAL++;   //////////// El oscilador corre despacio, incrementa el OSCCAL

      }

      else

        calibrate = TRUE;     ////////////// El interRC  esta correcto

      TCCR1B = (1<<CS10);  ///////// Empieza timer1

    }

}
```

4.3.5 SIMULACIONES

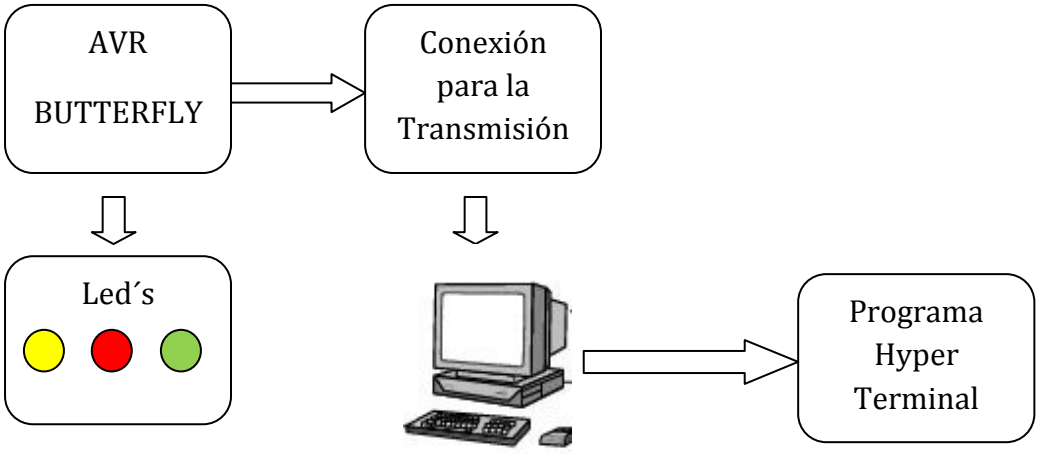


4.4 EJERCICIO 4:

4.4.1 ESPECIFICACIÓN

El programa transmite datos desde la plataforma interactiva, que está compuesta por la tarjeta electrónica AVR BUTTERFLY y Led's, hasta el computador (Virtual Terminal), El dato que se transmite se genera cada vez que se cumple un ciclo repetitivo en el cual se van activando alternadamente las salidas PD5, PD6 y PD7 del Port D, las cuales se pueden apreciar por los Led's que están conectados a dichas salidas, Al comenzar transmite el dato en código Ascii que representa la letra A, y cada vez que se cumple un ciclo se incrementa en uno y se va transmitiendo los valores en código Ascii.

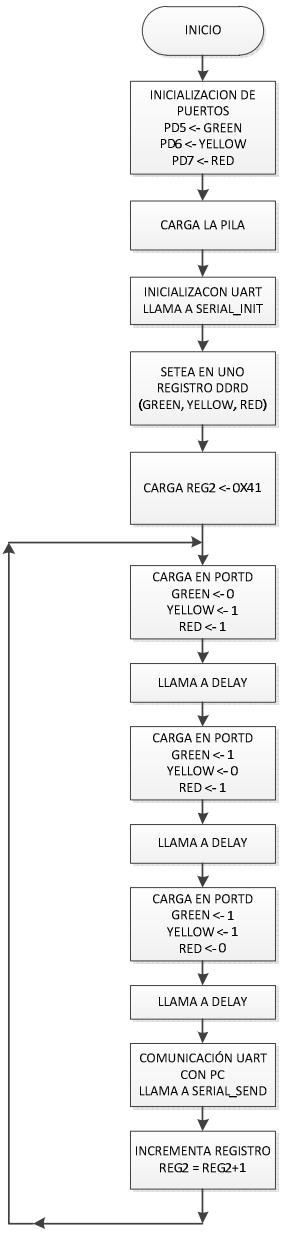
4.4.2 DIAGRAMA DE BLOQUES



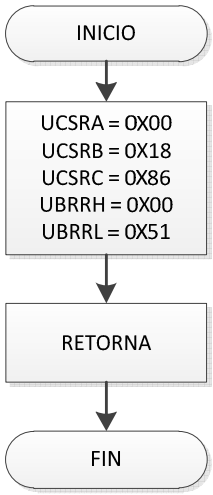
4.4.3 DIAGRAMA DE FLUJO

El siguiente diagrama de flujo representa un programa que transmite un dato que se genera cada vez que se cumple un ciclo repetitivo en el cual se van activando alternadamente las salidas PD5, PD6 y PD7 del Port D, para lo cual inicializamos los puertos de salida y la comunicación UART, Al comenzar transmite el dato en código Ascii que representa la letra A, y cada vez que se cumple un ciclo se incrementa en uno y se va transmitiendo los valores en código Ascii.

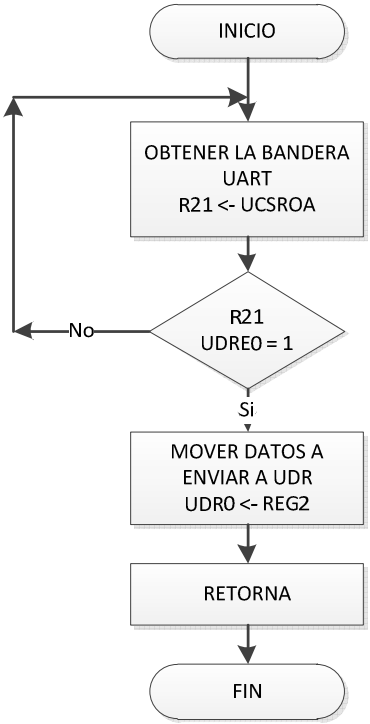
Principal:



Serial_init:



Serial_send:



4.4.4 CÓDIGO

```
////////////////////////////////////////////Librerias a usar////////////////////////////////////////////////////////////////

.include "m169def.inc"

.equ Green  = 5            ; LED en PD5

.equ Yellow = 6            ; LED en PD6

.equ Red    = 7  ; LED en PD7

.def reg1   = r19

.def reg2   = r20

RJMP MAIN                  ;///////////////// Vector Reset

MAIN:

        ldi R16,low(RAMEND)  ; /////////////////Carga la Pila con

        out SPL,R16                ; RAMEND - mas alto valor de la interna memoria SRAM

        ldi R16,high(RAMEND)

        out SPH,R16

        RCALL Serial_Init

        SBI DDRD,Green

        SBI DDRD,Yellow

        SBI DDRD,Red

        ldi reg2,0x41

DO:
```

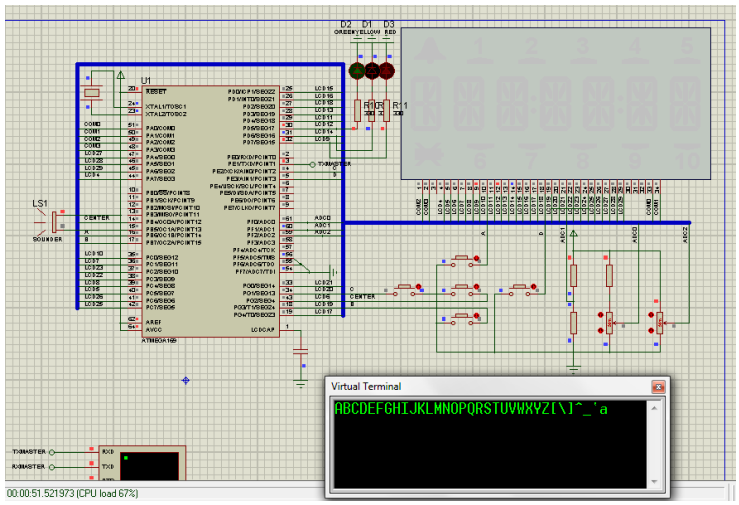
```
CBI PORTD,Green

SBI PORTD,Yellow

SBI PORTD,Red

RCALL DELAY                ; ////////////Espera por algun tiempo

SBI PORTD,Green

CBI PORTD,Yellow

SBI PORTD,Red

RCALL DELAY

SBI PORTD,Green

SBI PORTD,Yellow

CBI PORTD,Red

RCALL DELAY                ; ////////////////Espera por algun tiempo

RCALL Serial_Send

INC reg2

RJMP DO                    ; ////////////////////Lazo infinito

DELAY: LDI R16,$15  ; ////////////////////El rutina de retardo

LOOP1: SER R17             ; ////////////////////Cargar algun valor de retardo

LOOP2: SER R18

LOOP3: DEC R18

BRNE LOOP3            ; ////////////////Hacer R17 como $FF
```

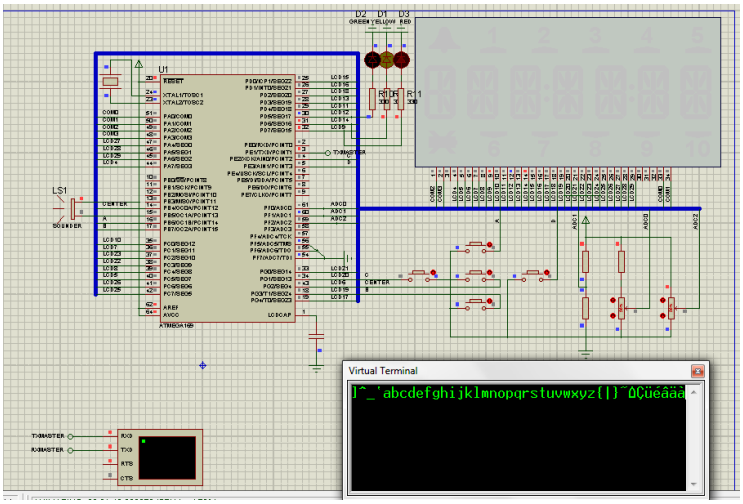
```
        DEC R17                 ; ////////////////Decrementa R17
        BRNE LOOP2
        DEC R16                 ;///////////////// Decrementa R16
        BRNE LOOP1              ;/////////////// Salta si no es cero
        RET
Serial_Init:
        ldi r16,0x00    ; UCSRA=0x00
        sts UCSR0A,r16
        ldi r16,0x18    ; UCSRB=0x18
        sts UCSR0B,R16
        ldi R16,0x86
        sts UCSR0C,R16  ; UCSRC=0x86
        ldi R16,0x00
        sts UBRR0H,R16  ; UBRRH=0x00
        ldi R16,0x33
        sts UBRR0L,R16  ; UBRRL=0x51
        ret
Serial_Send:
;//////////////////Esperar a que la bandera búfer de transmisión este vacía/////////
        lds r21,UCSR0A  ; Obtener las banderas USART
```

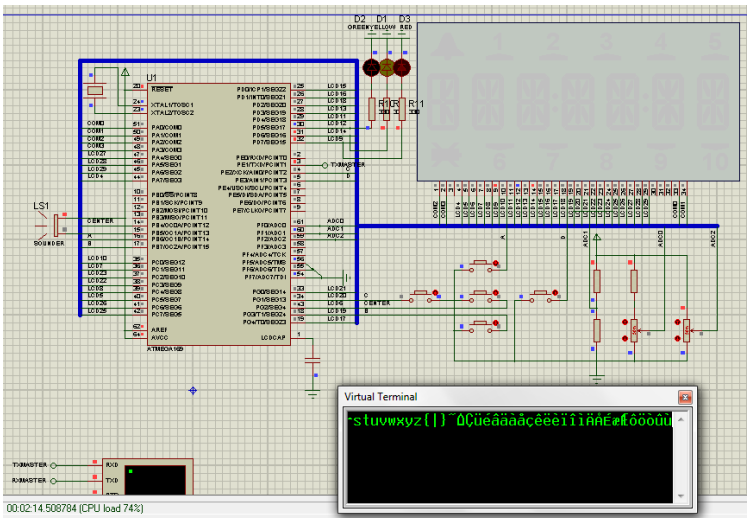
```
        sbrs r21, UDRE0
        rjmp Serial_Send
;////////////////////////////////////Si la bandera esta fijada ////////////////////////////////////////////////
;//////////////////////////Entonces mover los datos a enviar en UDR//////////////////////////////
        sts UDR0,reg2
        ret
Serial_Read:
;////////////////////////////////////////Esperar a recibir la bandera/////////////////////////////////////////
        lds r21,UCSR0A ; Obtener las banderas USART
        sbrs r21,RXC0
        rjmp Serial_Read
;////////////////////////////Si la bandera es fijada/////////////////////////////////////////////////////////
;////////////////////////////Entonces leer los datos de UDR/////////////////////////////////////////
        ldi reg2,UDR0
        ret
```

4.4.5 SIMULACIONES

El dato que se transmite se genera cada vez que se cumple un ciclo repetitivo en el cual se van activando alternadamente las salidas PD5, PD6 y PD7 del Port D, las cuales se pueden apreciar por los Led's que están conectados a dichas salidas, Al comenzar transmite el dato en código Ascii que representa la letra A, y cada vez que se cumple un ciclo se incrementa en uno y se va transmitiendo los valores en código Ascii.






D2 D1 D3
GREEN YELLOW RED
U1
RESET
R10
R11
LS1
SOUNDER
Virtual Terminal
TXMASTER
RXMASTER
RXD
TXD
00:02:14.508784 (CPU load 74%)

CONCLUSIONES

1. En base al desarrollo de este proyecto podemos concluir que fue muy útil y necesario contar con una herramienta como lo es el kit de la compañía Atmel el Avr Butterfly, con el cual se pudo construir y desarrollar la plataforma interactiva la cual cuenta con las herramientas necesarias para la ejecución de diferentes proyectos en el que se tenga que establecer la comunicación UART, por lo tanto fue acertada la decisión de utilizar este Kit para que forme parte de la plataforma interactiva.

2. Con la construcción de la plataforma interactiva educativa se pudo obtener una herramienta útil y a bajo costo, por cuanto esta nos permite apreciar el funcionamiento de los recursos disponibles en la comunicación UART, y en la misma podemos desarrollar diferentes proyectos programándolos y cargándolos en la tarjeta.

3. Se puede concluir que la comunicación Uart a través de puertos series ya sean 232 o 485 es muy importante y necesaria, para que se pueda interactuar entre el computador y otros dispositivos externos, además a través del programa Avr Studio, el cual permite programar nuestros proyectos y grabarlos en el ATmega169 que esta instalado en la tarjeta Avr Butterfly.

4. En la realización de nuestro proyecto pudimos darnos cuenta que para el caso particular del ejercicio en que se utiliza el Motor, se tuvo que poner un Microcontrolador para acoplar la tarjeta Avr Butterfly con el Motor, con esto se verifico el buen funcionamiento de nuestra plataforma interactiva.

RECOMENDACIONES

1. Es importante acotar que en la configuración de la comunicación UART, siempre se tiene que asegurar que tengan los mismos parámetros (Velocidad de transferencia de datos ect.) tanto en el Avr Butterfly como en la computadora, ya que si no se tiene esto no podremos establecer la comunicación entre ambos dispositivos.

2. Es necesario que en la construcción de este tipo de proyectos en los que se tenga que construir plataformas interactivas y educativas, se tome en cuenta que van a ser utilizadas por otras personas, los cuales necesitan tener todo de una forma comprensible y clara en base al funcionamiento de esta plataforma para que ellos la puedan utilizar e interactuar con la misma.

3. Antes de realizar un proyecto con la plataforma interactiva es preferible que las personas lean todo lo relacionado con el microcontrolador Atmega169 y el Kit de desarrollo Avr Butterfly, así como también la documentación de esta tesina, con lo cual podrán sacarle el mayor provecho a la plataforma interactiva y podrán entender mejor todos los atributos de la comunicación UART.

4. Para realizar la comunicación entre la PC y la tarjeta de desarrollo interactiva, es necesario que se cuente con un cable de comunicación serial RS232 de USB a DB9, y que este sea de buena calidad, ya que así nos evitaremos muchos dolores de cabeza a la hora de intentar hacer la comunicación entre estos dos dispositivos.

BIBLIOGRAFIA

- [1]ATMEL. (s.f.). Atmel Products., de Atmel AVR 8- and 32-bit Microcontrollers: http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3146; Fecha de consulta Octubre 2011.

- [2]Computación, F. d. Control por cambio de frecuencia de motor síncrono usando microcontroladores. Guayaquil-Ecuador: Tesina de Seminario de Graduación;ESPOL; Facultad de Ingeniería en Electronica en Electricidad y Computación; fecha de consulta Noviembre 2011.

- [3]Guía de desarrollo de practicas de laboratorio y tutoriales;Kit de desarrollo Avr butterfly. Sangolquí - Ecuador: Departamento de Eléctrica y Electrónica ; fecha de consulta Octubre 2011.

- [4]Hazael, I. (s.f.). Monografias.com., de http://www.monografias.com/trabajos-ppt/tutorial-isis-proteus/tutorial-isis-proteus.shtml; fecha de consulta el 05 de 11 de 2011.

- [5]Pardue, J., Micros, p. p., & Knoxville. C Programing for Microcontrollers. The Next Generation; fecha de consulta 2011.

- [6]Simulación con AVR Studio 4, http://www2.tech.purdue.edu/ecet/courses /ecet309/Reference _Materials/Simulation_AVR_Studio_4.pdf; fecha de consulta Octubre 2011.

- [7]Hazael, I. (s.f.). *Monografias.com*. de http://www.monografias.com/trabajos-ppt/tutorial-isis-proteus/tutorial-isis-proteus.shtml. Fecha de consulta el 2011

- [8]Monterrey, T. d. (s.f.). Maestría con Doble Grado en Tecnologías de Información, de http://www.duiops.net/hardware/modems/modems.htm; ; Fecha de consulta 05 de Diciembre de 2011