ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"CONTROL MEDIANTE JOYSTICK DE TARJETA AVR BUTTERFLY (CON MICROCONTROLADOR ATMEGA169) MEDIANTE COMUNICACIÓN RS232 CON TARJETA LPCXPRESSO CONTROLADORA DE MOTOR BLDC."

TESINA DE SEMINARIO

Previa a la obtención del título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

PRESENTADO POR:

Harry Daniel Gómez Rubio

Carlos Augusto Zúñiga Reyes

GUAYAQUIL- ECUADOR

AÑO 2012

AGRADECIMIENTO

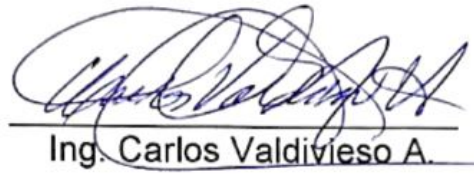
A Dios primeramente por darnos las ganas de superarnos y la inteligencia para llegar a ser personas productivas para la sociedad de forma profesional y humana.

A nuestros padres por ser ejemplo de superación y perseverancia, guiándonos siempre por el camino del bien con amor y respeto.

DEDICATORIA

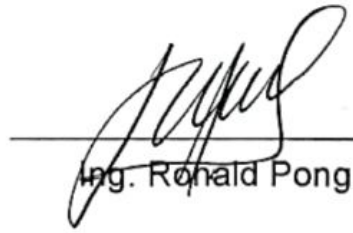
A nuestros padres por enseñarnos que nada en la vida es imposible si hay perseverancia, además de su apoyo incondicional en toda nuestra vida académica.

TRIBUNAL DE SUSTENTACIÓN



Ing. Carlos Valdivieso A.

PROFESOR DEL SEMINARIO DE GRADUACIÓN



Ing. Ronald Ponguillo I.

PROFESOR DELEGADO POR LA UNIDAD ACADÉMICA

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)

Carlos Augusto Zúñiga Reyes

Harry Daniel Gómez Rubio

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE ABREVIATURAS

MC	Microcontrolador
CC	Corriente Continua
CA	Corriente Alterna
PWM	Modulación de ancho de pulso
BLDC	Motor CC sin escobillas
TM	Par motor
I	Intensidad De Inducido
W	Velocidad Angular

INTRODUCCIÓN

Nuestro proyecto se enfoca en el aprendizaje del microcontrolador ARM Cortex3 de 32 bits de LPCXpresso, mediante la aplicación del control de motores Brushless DC sin sensores, para esto pondremos en práctica todo lo aprendido en la materia de graduación de Microcontroladores Avanzados además de los conocimientos previos de microcontroladores adquiridos a lo largo del pregrado, se necesitará de la herramienta LPCXpresso V2.0_264 la cual es un software con lenguaje de programación en “C” para el desarrollo de programas que controlen nuestro proyecto.

En el capítulo 1 se describirá el proyecto de forma general, exponiendo los antecedentes, motivación, objetivos y limitaciones para su realización. El capítulo 2 estará enfocado a los a la fundamentación teórica y las herramientas usadas para la realización de nuestro proyecto. En el capítulo 3 será explicado el diseño preliminar del proyecto así mismo los ejercicios que ayudaron a su realización. Y en el capítulo 4, se mostrarán las pruebas, simulaciones además del análisis de resultados.

CAPÍTULO 1

DESCRIPCIÓN GENERAL DEL PROYECTO

En el capítulo 1 detallaremos de forma general, las características del proyecto a realizar, tales como los objetivos planteados, motivos para la implementación de sistemas embebidos, explicar de qué manera interactuamos con los proyectos de nuestros compañeros así como los problemas y limitaciones que tuvimos en la transmisión de datos.

1.1 ANTECEDENTES

Muchos desarrollos recientes en la industria han impulsado el uso de motores eléctricos en sus prototipos por razones ambientales y económicas, entre otras, pues los niveles de contaminación en el planeta y la escasez de los recursos no

renovables están a punto de generar una crisis global. Quizás el problema más significante de los prototipos desarrollados hasta el momento ha sido la baja autonomía que presentan, dificultando la incursión de estos en el mercado. Investigaciones recientes se centran en mejorar la eficiencia de los motores eléctricos, utilizando nuevo tipos de motores y configuraciones de control. Entre los motores eléctricos con mayor acogida para esta finalidad se encuentran los motores Brushless DC ,que son motores sin escobilla, también llamados motores DC sincrónicos, que por su construcción ofrecen ventajas frente a otros motores eléctricos.

1.2 MOTIVACIÓN

Uno de los principales motivos para el uso de la Tarjeta AVR butterfly como Joystick es la facilidad con la esta puede ser usada, de manera que el control de los motores se torna muy sencillo y de forma muy práctica que hasta una persona sin conocimientos de electrónica puede manejar, ya que en la tarjeta AVR butterfly viene incluido un joystick con cinco botones, esto nos motivó a usarla e investigar más acerca de esta tarjeta que luego sería usada junto a la tarjeta 1769 para el control de motores BLDC.

Una motivación que tuvimos para el uso de los motores BLDC en nuestro proyecto es que poseen un mayor tiempo de vida útil frente a otro motores, como por ejemplo los motores con escobillas debido a que en los motores BLDC en ausencia de escobillas no presentan el desgaste que es característico en los motores con escobillas, lo que a su vez implicaría también la disminución de ruido y calor generado por el frecuente rozamiento.

1.3 IDENTIFICACIÓN DEL PROBLEMA

El presente trabajo, se trata sobre el control mediante el uso de la tarjeta AVR butterfly como un joystick, la cual enviara datos mediante la comunicación RS232 a la tarjeta 1769 que servirá como codificador y esta a su vez mandara los códigos correspondientes a la tarjeta madre encargada de hacer mover los motores, a su vez se mostrara las opciones de control mediante el display incorporado en la tarjeta AVR butterfly.

El problema a solucionar consiste en saber comunicar la tarjeta AVR butterfly con la tarjeta 1769, ya que para esto necesitaremos de la comunicación RS232, además de que la información enviada desde la tarjeta AVR butterfly debe ser interpretada de manera correcta por la tarjeta 1769 la cual será codificada y

enviada de forma entendible para la tarjeta madre que controlara los motores BLDC, para esto se deberá analizar los retardos de envíos y demás parámetros que optimicen la comunicación de las tarjetas y ayuden con el correcto funcionamiento del controlador de motores BLDC.

1.4 OBJETIVOS

OBJETIVO GENERAL

- Adquirir la destreza necesaria para poder comunicar tarjetas electrónicas mediante protocolos de comunicación ya establecidos y así poder enviar datos y a su vez estos ser codificados para diversos usos ya sea el control de aparatos electrónicos

OBJETIVOS ESPECÍFICOS

- Implementar un Joystick de control de motores
- Comunicar dos tarjetas mediante la comunicación RS232
- Dominar el manejo correcto de los microcontroladores LPCXPRESSO de 32 bits para propósitos generales.

- Manejar correctamente el microcontrolador AVR Butterfly y realizar la comunicación serial con microcontrolador LPCEXPRESSO para envió de datos.
- Lograr realizar un código en la tarjeta AVR butterfly que permita mostrar las distintas opciones de control de los motores BLDC, mediante el uso del joystick que viene incorporado en esta tarjeta.

1.5 LIMITACIONES

Una de las principales limitaciones en el desarrollo de nuestro proyecto fue trabajar por primera vez con los controladores de la familia ARM CORTEX de LPCXPRESSO ya que esto implicaba el aprendizaje de nuevas arquitecturas y dispositivos de trabajo

Otra limitación fue el uso de la tarjeta AVR butterfly, ya que a lo largo del seminario no se había analizado esta tarjeta ni su funcionamiento, esto implicaba investigar en tesis antiguas para saber su correcto funcionamiento, comunicación y como quemar códigos en la tarjeta.

Otra limitación adicional es tener todas las herramientas disponibles instaladas en una PC, en este caso el del laboratorio y tener que hacer uso de ella para

poder realizar diferentes pruebas y poder depurar hasta lograr nuestros objetivos. Esto atrasaba un poco el avance del proyecto debido a que no siempre era accesible el ingreso al laboratorio.

CAPÍTULO 2

SUSTENTACIÓN TEÓRICA

2.1 RESUMEN

En el presente capítulo daremos una breve descripción de cada uno de los elementos que serán utilizados para el desarrollo de nuestro proyecto. La herramienta de Software será LPCXpresso 2.0_2.64, AvrStudio4Setup y entre las herramientas de hardware tenemos: La tarjeta LPC1769, el motor BLDC con sensores y la tarjeta AVR butterfly que nos serviría como joystick para enviar señales a la tarjeta LPC1769 y así poder codificar esa información y enviarla a la tarjeta madre que permitirá el movimiento de los motores BLDC

2.2 HERRAMIENTAS DE SOFTWARE

2.2.1 LPCXPRESSO 2.0_2.64

LPCXpresso es una nueva plataforma de bajo costo de desarrollo disponible de NXP. El software consiste en un aumento, IDE basado en Eclipse, un compilador de C de GNU, enlazador, librerías, y un mayor depurador GDB. El hardware consiste en la placa de desarrollo LPCXpresso que tiene una interfaz de depuración LPC-Link y un NXP LPC basado en ARM microcontrolador objetivo.

LPCXpresso es una solución de extremo a extremo, permitiendo a los ingenieros incorporados a desarrollar sus aplicaciones desde la evaluación inicial hasta la producción final.

Figura 2.1: LPCXpresso V2.0_264 [7]

Es una herramienta de desarrollo para programar tarjetas creadas por la compañía LPCXpresso en la cual la programación se la realiza en el lenguaje C/C++ entre sus características principales podemos mencionar que posea un compilador paso a paso que nos permite ir observando línea por línea el desarrollo del programa.

El LPCXpresso IDE ha sido desarrollado por Code Red junto a NXP. El mismo incluye un entorno de Eclipse específicamente adaptado para interactuar con la tarjeta.

Eclipse utiliza algunos conceptos comunes a otros programas de desarrollo por lo que vamos a ver algunos detalles.

Workspace: Es donde se encuentran nuestros proyectos. Estos proyectos pueden ser aplicaciones y/o bibliotecas. También almacena todas las configuraciones del entorno por lo que se puede mover muy fácilmente de computadora en computadora con solo almacenar una copia del el workspace.

Proyecto: Este puede ser de dos tipos. Biblioteca estática o una aplicación ejecutable. Aquí se contiene archivos de código fuente (.c), encabezados (.h) y cualquier otro archivo que se desee.

En general workspace es utilizado para intercambiar proyectos (en el sentido convencional de la palabra) ya que el mismo incluirá todas las bibliotecas necesarias.

El workspace se encuentra ubicado en la parte encerrada en el cuadro de color rojo y el área en cerrada en el rectángulo en azul corresponde a los proyectos.

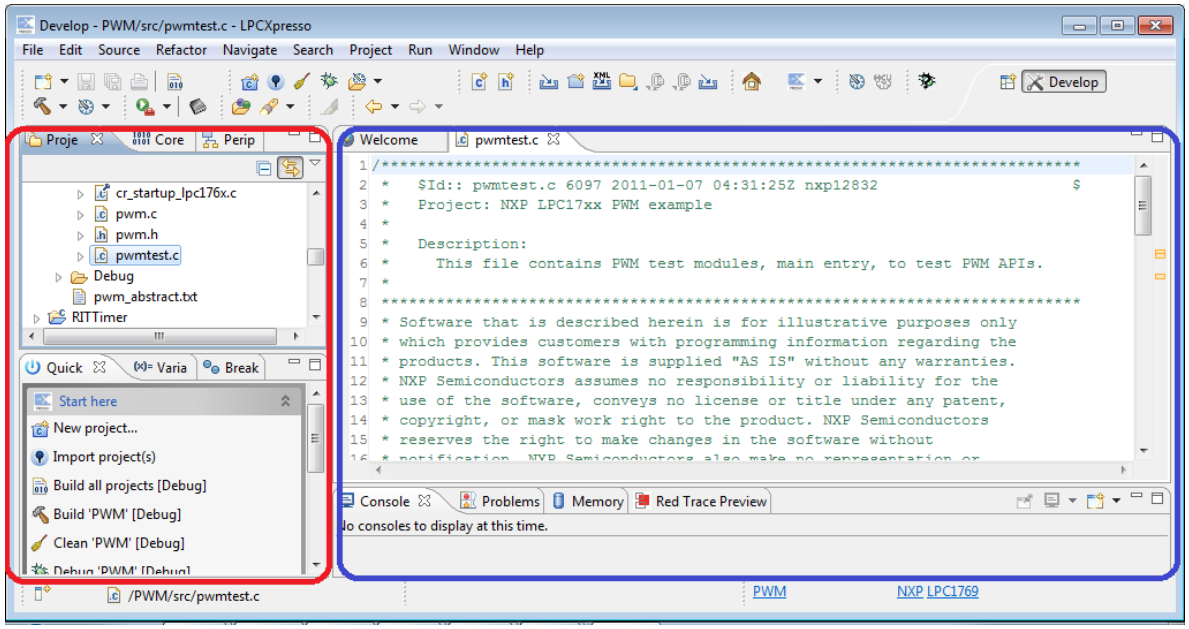


Figura 2.2: Lpcxpresso interfaz [7]

Los proyectos pueden ser de dos tipos:

- Aplicaciones: Se compilan y se pueden descargar directamente al target.
- Bibliotecas estáticas: Se pueden compilar, pero para usarlas, un proyecto de tipo aplicación debe hacer llamadas a las funciones que este contiene. Es decir, no puede tener un main(). Este tipo de proyectos no se puede descargar por si solo al microcontrolador.

2.2.2 AVR STUDIO 4

Es un entorno de desarrollo IDE ensamblador y programador de software para el desarrollo de aplicaciones de Atmel AVR de 8 bits en Windows NT, Windows 2000, Windows XP, Windows Vista y Windows 7.

El IDE soporta todas las herramientas de Atmel que apoyan a la arquitectura AVR 8 bit. AVR studio incorpora un depurador que permite el control de ejecución con fuente y nivel de instrucción, paso a paso y puntos de interrupción, el registro, la memoria y E/S puntos y configuración y gestión, y apoyo a la programación completa para los programadores independientes además permite crear archivos assembler (asm) y archivos .C véase la Figura 3

Características Principales

- Se integra con el compilador GCC plug-in
- AVR RTOS plug-in de apoyo
- Soporta AT90PWM1 y ATtiny 40.
- Herramientas de CLI al día con el apoyo de TPI
- Ayuda en línea.

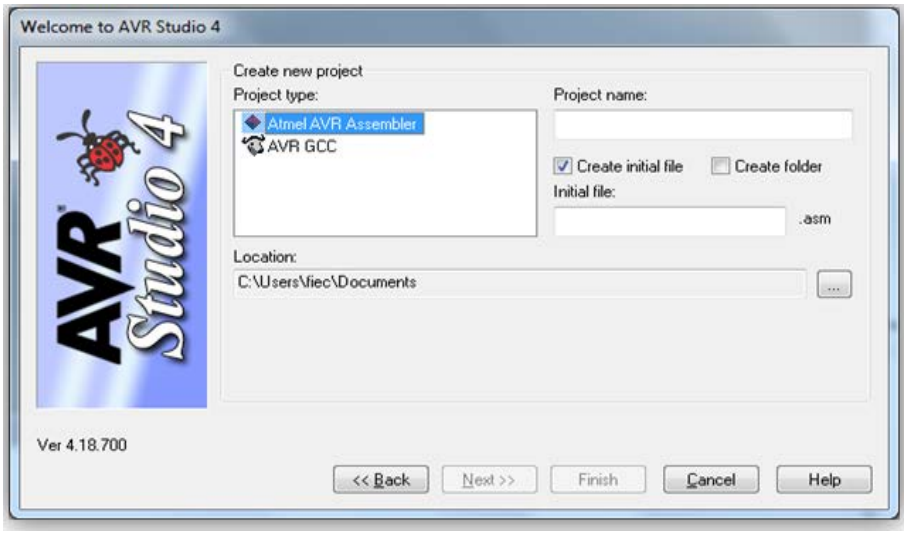


Figura 2.3: selección del lenguaje [10]

AVR Studio cuenta con algunas formas para poder programar los microcontroladores de la familia ATMEL, para la realización de este trabajo se utilizaran los siguientes.

La programación ISP, a la cual se accede mediante la opción AVRISP, permite grabar el microcontrolador tanto del Robot Pololu 3pi así también como del AVR Butterfly. Se hace uso del Pololu USB AVR Programmer el cual se conecta al puerto ISP de los módulos a través de un cable de 6 líneas.

Se conecta el modulo a programar al PC se verifica la conexión luego comienza el proceso de grabación y luego de verificación del microcontrolador.

2.3 HERRAMIENTAS DE HARDWARE

2.3.1 TARJETA LPC1769

La tarjeta LPC1769 es un ARM Cortex M3 basado en microcontroladores para sistemas embebidos con un alto nivel de integración. El ARM Cortex M3 es la siguiente generación de procesadores usados en un sinnúmero de aplicaciones.

Las versiones de alta velocidad del LPC1769 operan hasta una frecuencia de CPU de 120 MHz otras versiones trabajan solamente hasta una frecuencia de 100 MHz, esta tarjeta trabaja con arquitectura Harvard con las instrucciones locales separadas de los buses de datos además un tercer bus para los periféricos

Los periféricos que complementan al LPC incluyen hasta 512Kb de memoria flash, hasta 64Kb de memoria de datos, una Ethernet MAC, una interfaz usb que puede ser configurada tanto como un dispositivo host o como dispositivo OTG(OnTheGo), tiene incorporado 4 UARTs,2 canales para CAN ,2 controles SSP, interfaz SPI, 3 interfaces I2C,interfaz ADC de 8 canales con una resolución de 12 bits y de 10 bits en DAC, Control de motores por PWM, 4 timers de propósito general y hasta 70 pines con propósito de I/O

CARACTERÍSTICAS DE LA TARJETA LPC1769

- Posee un procesador ARM Cortex M3 operando a velocidades de hasta 120Mhz.
- La memoria flash tiene una capacidad de 512 Kb.
- 64Kb de memoria SRAM
- Posee interfaces de comunicaciones: SPI, RS232,I2C,Ethernet
- Cuenta con comunicación USB OTG (Onthego)
- Soporta debug realizado en real time
- Alimentación de 3.3 V (2.4V hasta 3.6V) y un rango de temperatura desde -40 C hasta 85C

- Cada periférico tiene su propio divisor de reloj para un mayor ahorro de energía
- Sistema PWM capaz de controlar motores de tres fases

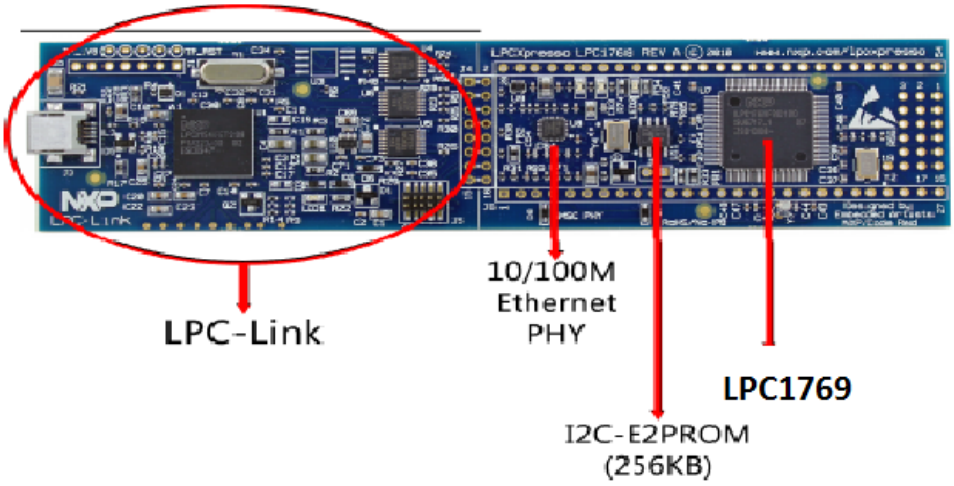


Figura 2.4: Tarjeta 1769 [6]

2.3.3 AVR BUTTERFLY

Los kits de AVR Butterfly están diseñados para demostrar los beneficios y las principales características de los microcontroladores AVR este es un módulo de soporte que puede ser utilizado en numerosas aplicaciones Fig 2.5.

El AVR Butterfly contiene un microcontrolador ATMega169, el cual va a realizar el comando de las diferentes funciones de las que es capaz éste kit.

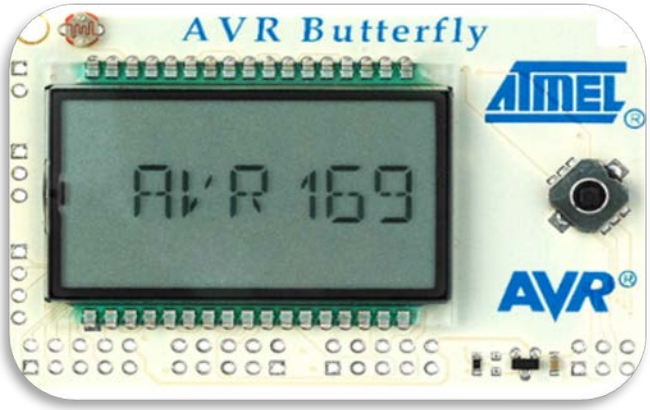


Figura 2.5: AVR Butterfly [1]

2.3.3.1 CARACTERÍSTICAS DEL BUTTERFLY

- Diseño de bajo poder
- El tipo de paquete MLF
- Controlador de LCD
- Memorias
- Flash, EEPROM, SRAM, DATAFLASH externos
- Interfaces de comunicación
- UART, SPI, USI
- Convertidor analógico a digital (ADC)
- Temporizadores / contadores
- Reloj en tiempo real (RTC)

- Modulación por impulsos (PWM)

2.3.3.2 ELEMENTOS QUE CONTIENE EL AVR BUTTERFLY

Los siguientes recursos están disponibles en el kit del butterfly Atmega 169

- LCD en la pantalla de vidrio con 120 segmentos, para demostrar la ATMEGA 169 controlador LCD.
- Joystick de 4 direcciones con empuje el centro, como la entrada del usuario
- Elemento piezoeléctrico, para reproducir sonidos
- 32KHZ Xtal para la RTC
- 4 Mbit DATAFLASH, para el almacenamiento de datos
- RS-232-convertidor de nivel, para comunicarse con las unidades fuera de borda
- Coeficiente de temperatura negativo (NTC) termistor, para medir la temperatura
- Resistencia depende de la luz (LDR) para medir la intensidad de la luz 3V pila de botón (600mAh) para proporcionar energía de funcionamiento
- Emulación JTAG, para la interfaz de comunicación adicional

El ATMEGA 169 en el juego de controles de los periféricos externos, y también se puede utilizar para hacer la lectura de voltaje de 0 a 5 voltios.

El kit se puede reprogramar una serie de maneras diferentes, incluyendo programación en serie a través del puerto JTAG. La mayoría de usuarios prefieren utilizar el gestor de arranque precargado con el estudio de AVR para descargar nuevo código.

El AVR Butterfly viene con una aplicación reprogramada. En esta sección se pasará a través de los fundamentos de esta solicitud.

JOYSTICK

El AVR Butterfly tiene un joystick en miniatura para operar la entrada de usuario. Maneja en cinco direcciones, incluyendo arriba, abajo, derecha, izquierda y centro. La línea común de todas las direcciones es GND. Esto significa que pull-up interna debe estar habilitado en el ATMEGA 169 a detectar a partir de la entrada de la palanca de mando, véase la figura 2.6

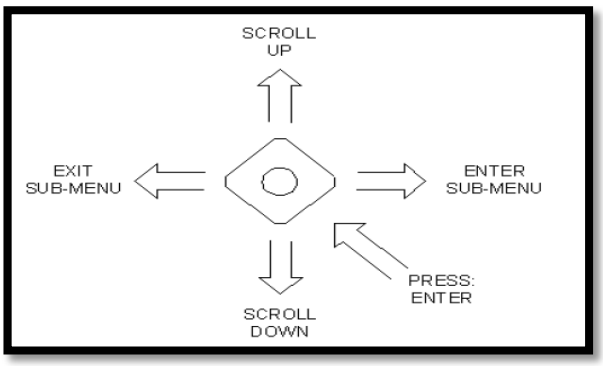


Figura 2.6: Diagrama del Joystick [1]

PANTALLA LCD

La pantalla LCD del AVR Butterfly es la misma que la utilizada en la disposición STK502 de Atmel. Las conexiones entre el ATmega 169 y la pantalla LCD también son las mismas.

STK502 es un módulo superior diseñado para añadir soporte ATmega169 a la placa de desarrollo STK500 de Atmel Corporation. Incluye una pantalla LCD. Cuenta con seis dígitos de 14 segmentos, y algunos segmentos adicionales. En general, la pantalla es compatible con 120 segmentos. La pantalla está diseñada para la tensión de funcionamiento de 3V véase la Figura 2.7

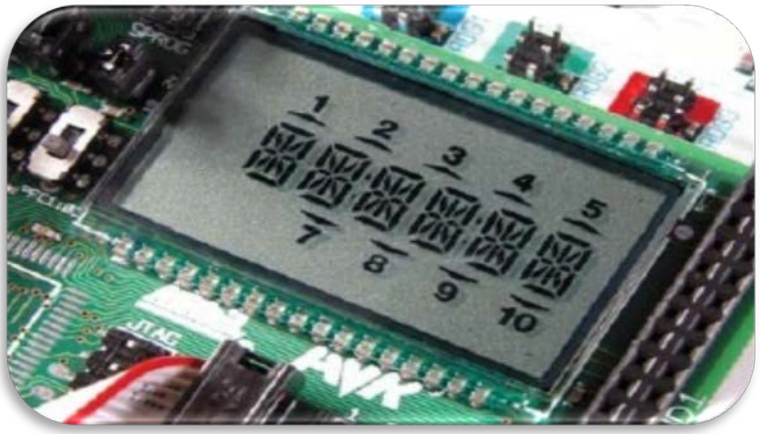


Figura 2.7: Pantalla LCD [3]

CONEXIÓN DE LA PANTALLA LCD STK502 AL ATMEGA169

Los segmentos de pines de ATmega 129 se encuentran en PORTA, PORTC, PORTD y PORTG. Por razones de simplicidad en el uso de todos ellos son unidos en la cabecera de la etiqueta “pasadores de segmento de ATmega 169”. La cabecera a su lado, la etiqueta “STK502 pines LCD” tiene todos los segmentos pines para la pantalla LCD en el STK502.

Al utilizar el cable de 34 derivaciones que viene con el STK502-kit, los dos pines de conexión se pueden conectar, permitiendo que el ATmega 169 para controlar la pantalla LCD. **[3]**

MICROCONTROLADOR ATMEGA 169

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

El núcleo AVR combina un amplio conjunto de instrucciones con 32 registros de propósito general de trabajo.

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

El ATmega 169 proporciona las siguientes características:

- 16k bytes de sistema programable
- Flash con lectura y escritura mientras que las capacidades, 512 bytes de EEPROM, SRAM bytes 1K.
- 54 registros de propósito general
- 32 registros de propósito general de trabajo
- Controlador de LCD con la resistencia de step-up de tensión
- Una serie UART programable, serie universal
- Sistema de interrupción
- Interfaz con el inicio de condición del detector

2.3.4 MOTORES BLDC

2.3.4.1 EVOLUCIÓN DE LOS MOTORES DC

Los primeros motores sin escobillas fueron los motores de corriente alterna asíncronos. Hoy en día, gracias a la electrónica, se muestran muy ventajosos, ya que son más baratos de fabricar, pesan menos y requieren menos mantenimiento, pero su control era mucho más complejo. Esta complejidad prácticamente se ha eliminado con los controles electrónicos.

Los motores eléctricos solían tener un colector de delgas o un par de anillos rozantes. Estos sistemas, que producen rozamiento, disminuyen el rendimiento, desprenden calor y ruido, requieren mucho mantenimiento y pueden producir partículas de carbón que manchan el motor de un polvo que, además, puede ser conductor.

Los motores brushless han derivado de los motores de los CD ROM, los DISCOS RIGIDOS y los ventiladores de computación, son motores trifásicos de alto rendimiento y bajo peso.

2.3.4.2 CONTROL DEL MOTOR DE F.E.M. SENOIDAL

Los motores de f.e.m. senoidal han de ser alimentados con un sistema de tensiones e intensidades también senoidales, y sincronizadas en todo momento con la f.e.m. inducida. El control de estos motores es complejo y se recurre a

técnicas similares a las empleadas en los motores asíncronos, incluidas las técnicas de control vectorial.

2.3.4.3 CONTROL DEL MOTOR DE F.E.M. TRAPEZOIDAL

La figura 8 muestra las ondas de tensión y de intensidad correspondiente a una fase para un motor de este tipo. En la figura 2.9 se han dibujado las tres tensiones para un motor trifásico y los transistores que intervienen en cada intervalo de funcionamiento. Hay que destacar que la conducción se realiza siempre a través de dos transistores; uno de los del grupo superior (T1, T2 o T3) y otro de los del inferior (T4, T5 o T6), de forma que siempre hay una bobina desactivada.

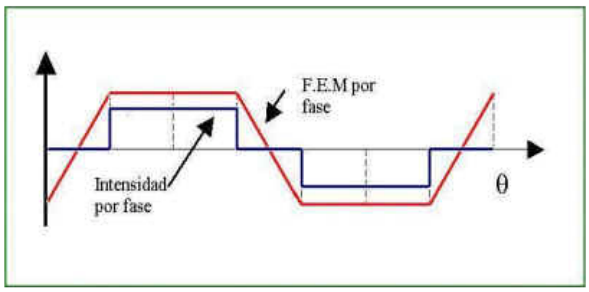


Figura 2.8: Formas básicas de tensión e intensidad de un motor de CC sin escobillas

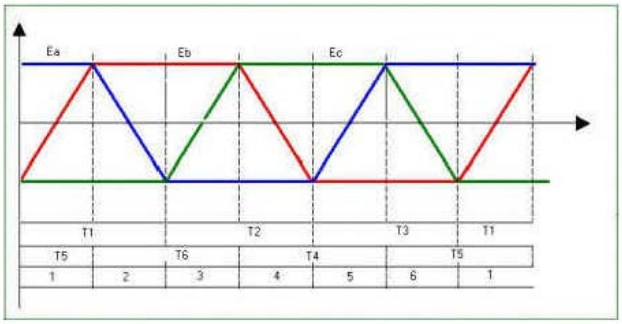


Figura 2.9: Ondas de ff.ee.mm. e intervalos de conducción

2.3.4.4 INTRODUCCIÓN A MOTORES BRUSHLESS DC (BLDC)

Los motores DC con escobillas son altamente eficientes y tienen grandes características para hacerlos funcionar como servo motores. Pese a estas grandes ventajas cuenta con un conmutador y con unas escobillas las cuales están sujetas al desgaste y por esta razón, se hace necesario un calendario de mantenimiento de las mismas.

La característica principal de los motores DC sin escobillas es que realiza la misma función de un motor DC normal pero reemplazando el conmutador y las escobillas por switches de estado sólido que funcionan con una lógica para la conmutación de los embobinados. Se puede concluir que la gran ventaja de los motores DC sin escobillas. Frente n a los demás motores de alimentación continua, es que no requieren un mantenimiento periódico.

Los motores DC electrónicamente conmutados (brushless DC) destacan particularmente por sus excelentes características de par, altas prestaciones, rango de velocidades muy amplio y, por supuesto, su insuperable duración en servicio.

Sin conmutación mecánica al no tener escobillas. Sin embargo requieren una electrónica externa (o integrada) para realizar la conmutación.

Prolongada vida útil, limitada únicamente por los rodamientos, mínimo 20.000 horas a carga máxima.

Línea velocidad-par largamente lineal, permitiendo una excelente regulación.

Elevada eficiencia alcanzando el 90%: Aprovechan la energía eléctrica, convirtiéndola en potencia mecánica y generando menos calor. Idóneo para aplicaciones alimentadas por baterías, o donde el consumo sea importante.

Muy baja constante eléctrica de tiempo y reducida inductancia, por lo tanto, mínimo ruido eléctrico, o interferencias eléctricas prácticamente inexistentes.

Campos de aplicación: entornos explosivos (sin chispas), salas limpias (sin desgaste) y cualquier otra aplicación que requiera velocidades de giro elevadas y larga vida en servicio.

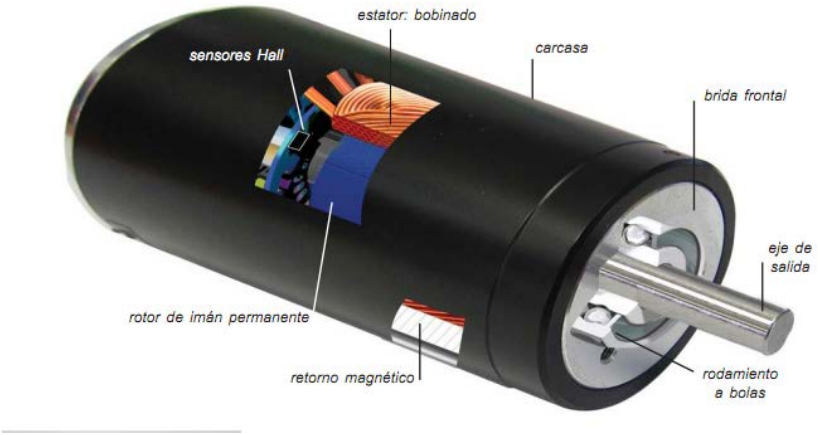


Figura 2.10: Elementos de un motor brushless

2.3.4.5 ESTRUCTURA BÁSICA

La construcción de motores DC sin escobillas modernos es muy similar a la de los motores AC, que es mostrada en la figura

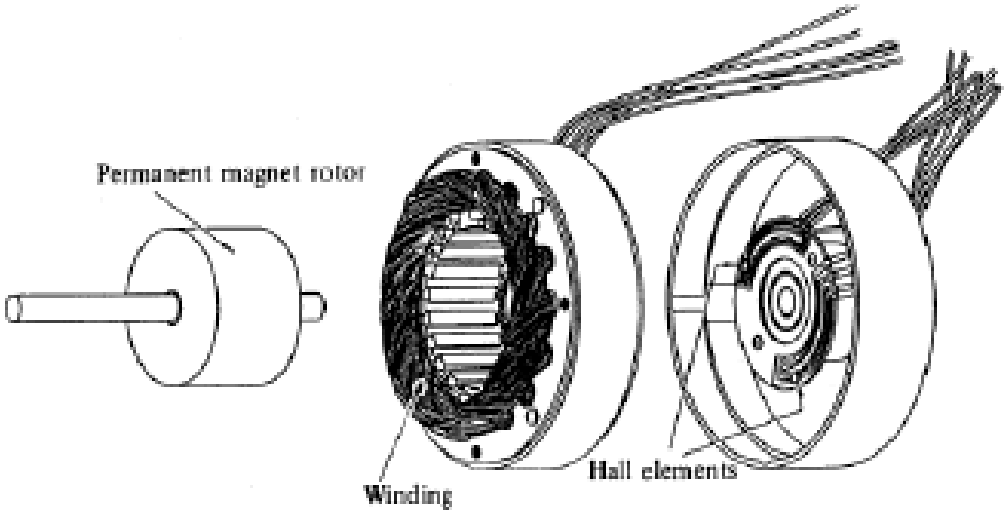


Figura 2.11: Partes de un motor Brushless DC

El rotor es un elemento magnético permanente, y el estator está formado por embobinados al igual que un motor AC de varias fases. La gran diferencia entre estos dos tipos de motores es la forma de detectar la posición del rotor, para poder saber cómo se encuentran los polos magnéticos y así generar la señal de control mediante switches electrónicos.

Este censado de la ubicación de los polos magnéticos en los motores DC sin escobillas normalmente se hace con sensores de efecto hall aunque existen modelos que utilizan sensores ópticos, que funcionan de manera similar a los encoders.

2.3.4.6 PRINCIPIO DE FUNCIONAMIENTO

En un motor de corriente continua con escobillas, se obtiene par motor gracias a la interacción del campo magnético inductor, estacionario, y la intensidad del arrollamiento inducido giratorio. Campo y corriente eléctrica se mantienen siempre en la misma posición relativa gracias al mecanismo de conmutación formado por el colector de delgas y las escobillas. En motores de pequeña potencia suele obtenerse la excitación mediante imanes permanentes. En este caso, solo se dispone de dos terminales para el control y la alimentación del motor. Las relaciones básicas electromecánicas son en este caso las siguientes

$$T_m = K \cdot i$$
$$E = K \cdot \Omega \qquad (2.1)$$

Siendo, Tm: Par motor; i: intensidad de inducido; E: tensión inducida; W: velocidad angular. El hecho de tener control directo sobre el par mediante la intensidad de inducido, y sobre la velocidad a través de la tensión, convierte a este motor en el modelo de referencia para la regulación de velocidad. No obstante, la alimentación del inducido a través de las escobillas y el colector presenta muchos inconvenientes, hasta el punto que en algunos casos se hace inviable su utilización.

El motor que nos ocupa es similar al de corriente continua con escobillas, con las siguientes salvedades: a) la conmutación se realiza de forma electrónica en lugar de mecánica; b) los imanes permanentes van alojados en el rotor en lugar de en el estator y c) las bobinas van alojadas en el estator, constituyendo un devanado monofásico o polifásico.

Su funcionamiento se basa en la alimentación secuencial de cada una de las fases del estator de forma sincronizada con el movimiento del rotor. De esta forma, los imanes permanentes siguen el movimiento del campo magnético estatórico, cuyo desplazamiento depende a su vez del giro del rotor (figura 12).

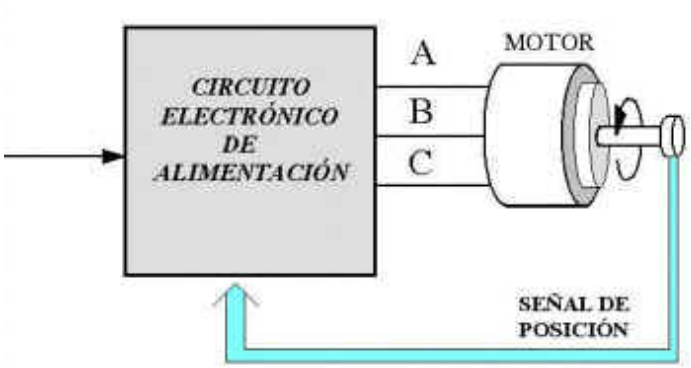


Figura 2.12: Esquema de funcionamiento de un motor de corriente continúa sin escobillas

LÓGICA DE FUNCIONAMIENTO

Como ejemplo para explicar la lógica de funcionamiento de un motor DC sin escobillas, se utilizó un motor con un rotor (elemento magnético), tres embobinados en el estator y tres foto-transistors encargados de la detección de la posición del rotor.

El rotor del motor se encuentra sujeto a una especie de lámina que va girando con este y que es el objeto que obstruye la luz a la foto-transistor, con lo que se obtiene los estados de los sensores, que determinan las variables de entrada a la lógica que realiza el movimiento. Esto se ve mejor representado en la figura 2.13

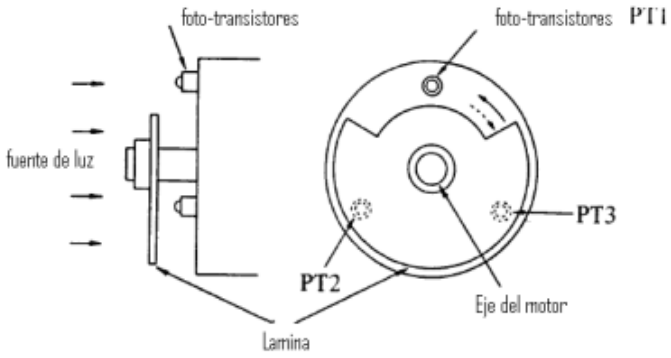


Figura 2.13: Sensores de efecto Hall

Por ejemplo, en la gráfica se puede observar que mientras PT1 está recibiendo luz, PT2 y PT3 están tapados por la lámina, y de e3sta forma se sabe en qué posición se encuentra el rotor al momento de la toma de datos.

Una vez que se conoce la posición del rotor, se comienza a seguir la lógica secuencial para moverlo a una velocidad determinada, esto se logra energizando las bobinas del estator en diferentes tiempos. Para alimentar los embobinados se usa un control, seguido de una etapa de salida compuesta por transistores, que cumplan con los requerimientos de velocidad y potencia, y se hace pasar corriente por las fases dependiendo de la posición del rotor, este esquema se observa en la figura. Es decir para el ejemplo que se tomó anteriormente de PT1 prendido, mientras PT2 y PT3 se encuentran apagados, la

lógica decide por cual embobinado hacer pasar corriente para que gire en uno u otro sentido como se muestra en la figura 2.14.

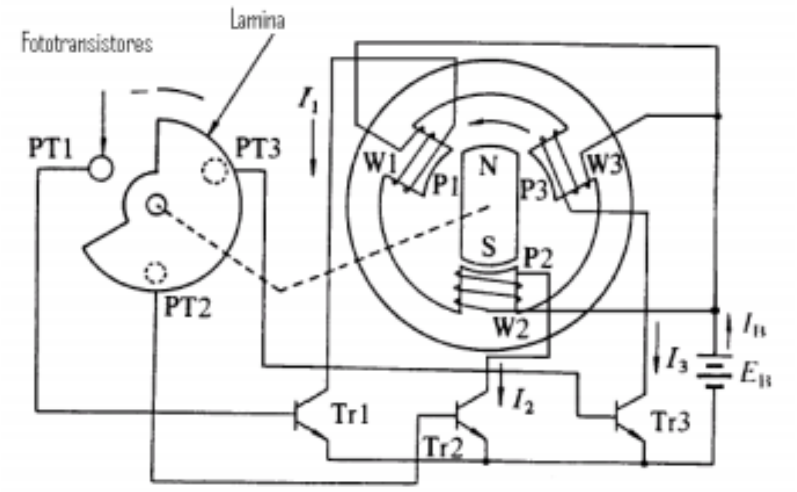


Figura 2.14: Acople sencillo de sensores y fases

2.3.4.7 SENSOR DE EFECTO HALL (O SONDA HALL)

Solo para motores brushless DC. Efecto Hall: Cuando fluye una corriente a través de un sensor Hall y este se aproxima a un campo magnético perpendicular, entonces se crea un voltaje saliente proporcional al producto de la fuerza del campo magnético y de la corriente. Gracias a este principio, mediante un disco magnético acoplado al eje del servomotor podemos censar la posición del rotor. Principalmente, estos sensores se usan para que la

electrónica pueda conmutar las tres bobinas del motor de acuerdo a la posición de los polos del imán del rotor. Así por ejemplo un motor brushless de dos polos con 3 sensores hall (a 120 º), tiene una resolución en posición de 6 pulsos por vuelta (60 º de conmutación). En caso de los servomotores multipolares esta resolución aumenta. Ocasionalmente, se pueden accionar los motores brushless sin sensores Hall para determinadas aplicaciones muy sencillas, como por ejemplo ventiladores y bombas. Sin sensores Hall, el problema es que el arranque del servomotor es un poco brusco.

Algunas veces se utilizan los sensores Hall para aplicaciones de posicionamiento de baja resolución. Si tenemos una reductora acoplada al servomotor esta resolución se multiplica.

2.3.4.8 VENTAJAS Y DESVENTAJAS

Los motores brushless tienen muchas ventajas por sobre los motores brushed (con escobillas) entre ellas las más nombradas son:

- Mayor eficiencia (menos perdida por calor)
- Mayor rendimiento (mayor duración de las baterías para la misma potencia)
- Menor peso para la misma potencia

- Conmutación electrónica basada en sensores de posición de efecto Hall
- Requieren menos mantenimiento al no tener escobillas
- Relación velocidad/par motor es casi una constante
- Mayor potencia para el mismo tamaño
- Mejor disipación de calor
- Rango de velocidad elevado al no tener limitación mecánica.
- Menor ruido electrónico (menos interferencias en otros circuitos)
- Desventajas de un motor brushless
- Mayor costo de construcción
- El control es mediante un circuito caro y complejo
- Siempre hace falta un control electrónico para que funcione, que a veces duplica el costo

2.3.4.9 CLASIFICACIÓN

Básicamente, hay dos tipos de motores brushless, los inrunner y los outrunner. Los primeros son de más velocidad, su torque máximo lo tienen a muy altas revoluciones, por lo que se usan con reductoras o con ducted funs. Los outrunner tienen su torque máximo a baja velocidad, por lo que no necesitan reductoras, van directamente a la hélice.

MOTORES DC BRUSHLESS: INRUNNER

La forma en la que están construidos estos motores es disponiendo los imanes directamente en torno al eje, mientras que el bobinado es exterior y rodea el eje con los imanes, este tipo de motores tiene la ventaja de proporcionar un alto número de revoluciones por lo que su Kv (revoluciones por voltio) es muy alto, sin embargo esto tiene la desventaja de proporcionar un par muy bajo y si queremos utilizar una hélice grande no tendremos más remedio que emplear una reductora. Este tipo de motor es muy popular en alas Zagi, donde se requiere una hélice pequeña funcionando con un gran número de revoluciones. Aunque el empleo de la reductora se podría tomar como un inconveniente, ya que puede ser una fuente más de averías y requiere de un mantenimiento, si queremos potencia y efectividad en un motor inrunner la reductora es una buena solución.

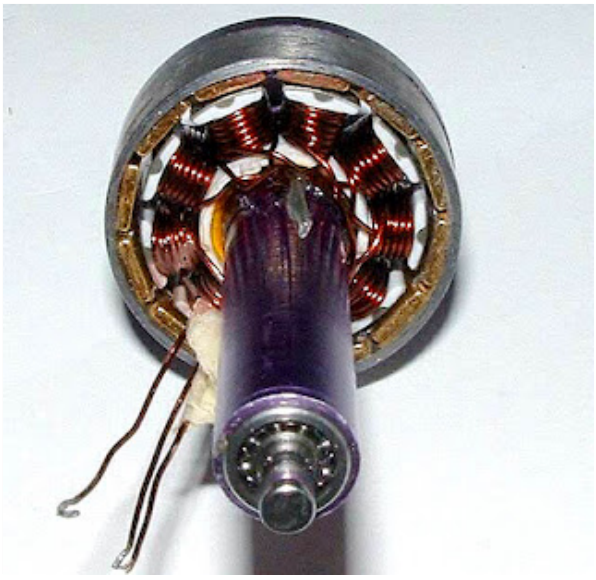


Figura 2.15: Motor inrunner

MOTORES DC BRUSHLESS: OUTRUNNER

En este tipo de motores los bobinados de cobre se disponen en la parte interior central, mientras que los imanes están situados en una campana exterior que rodea al bobinado y a la que se conecta el eje, cuando la campana gira lo hace también el eje, estos motores producen un menor número de revoluciones, sin embargo la campana al actuar como un volante de inercia, les dota de un mayor par por lo que dan más potencia sin necesidad del empleo de una reductora, esto les hace ser más ligeros, silenciosos y económicos.

PARÁMETRO KV

Uno de los parámetros que casi siempre aportan los fabricantes sobre un motor brushless es el Kv, no es más que las revoluciones por minuto y por voltio que da un motor. Se podría hacer un símil con los motores de dos y de cuatro tiempos, es decir para una potencia dada, un alto kv proporcionará muchas más revoluciones que un bajo kv. Un alto Kv está indicado para aviones rápidos con una hélice pequeña, mientras que un Kv mayor es adecuado para un motovelero que lleva una hélice más grande.

Si estamos limitados por el voltaje de la batería podemos jugar con el Kv del motor para adecuar el comportamiento del conjunto a nuestro gusto, mientras que si no estamos limitados por la potencia podemos usar motores de un Kv menor y simplemente para adecuar el número de revoluciones a nuestro gusto aumentaremos el voltaje de la batería. Un ejemplo típico sería las siete pilas de 1,2 v de un velero de competición, es necesario utilizar un motor con un alto Kv ya que con 8,4 v si no tenemos alto Kv el motor no sería lo suficientemente competitivo.

Figura 2.16: Motor outrunner

2.4 COMUNICACIÓN SERIAL

2.4.1 ¿QUÉ ES LA COMUNICACIÓN SERIAL?

La comunicación serial es un protocolo muy común (no hay que confundirlo con el Bus Serial de Comunicación, o USB) para comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. La mayoría de las computadoras incluyen dos puertos seriales RS-232. La comunicación serial es también un protocolo común utilizado por varios dispositivos para instrumentación; existen varios dispositivos compatibles con GPIB que incluyen un puerto RS-232. Además, la comunicación serial puede ser

utilizada para adquisición de datos si se usa en conjunto con un dispositivo remoto de muestreo.

El concepto de comunicación serial es sencillo. El puerto serial envía y recibe bytes de información un bit a la vez. Aun y cuando esto es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias. Por ejemplo, la especificación *IEEE 488* para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20 metros, con no más de 2 metros entre cualesquier dos dispositivos; por el otro lado, utilizando comunicación serial el largo del cable puede llegar a los 1200 metros.

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión: (1) Tierra (o referencia), (2) Transmitir, (3) Recibir. Debido a que la transmisión es asincrónica, es posible enviar datos por una línea mientras se reciben datos por otra. Existen otras líneas disponibles para realizar *handshaking*, o intercambio de pulsos de sincronización, pero no son requeridas. Las características más importantes de la comunicación serial son la velocidad de transmisión, los bits

de datos, los bits de parada, y la paridad. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

a. **Velocidad de transmisión *(baud rate)*:** Indica el número de bits por segundo que se transfieren, y se mide en baudios (*bauds*). Por ejemplo, 300 baudios representa 300 bits por segundo. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión. Por ejemplo, si el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800 Hz, lo que significa que el puerto serial está muestreando las líneas de transmisión a 4800 Hz. Las velocidades de transmisión más comunes para las líneas telefónicas son de 14400, 28800, y 33600. Es posible tener velocidades más altas, pero se reduciría la distancia máxima posible entre los dispositivos. Las altas velocidades se utilizan cuando los dispositivos se encuentran uno junto al otro, como es el caso de dispositivos GPIB.

b. **Bits de datos:** Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía

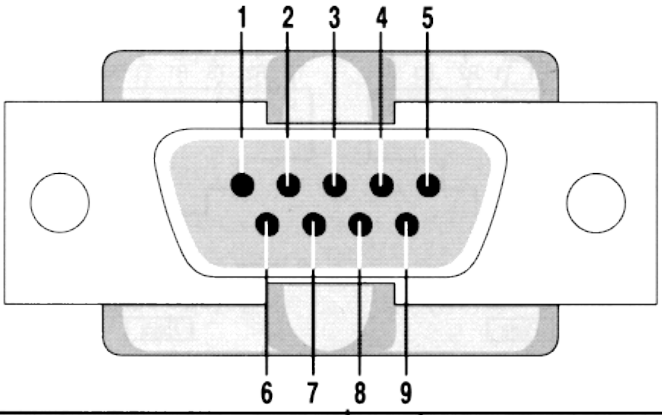
depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII extendido es de 0 a 255, lo que utiliza 8 bits. Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits por paquete para la comunicación. Un paquete se refiere a una transferencia de byte, incluyendo los bits de inicio/parada, bits de datos, y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se usar para referirse a todos los casos.

c. **Bits de parada:** Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes. Mientras más bits de parada se usen, mayor será la tolerancia a la sincronía de los relojes, sin embargo la transmisión será más lenta.

d. **Paridad:** Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible. Para paridad par e impar, el puerto serial fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par. Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para tener 3 bits en estado alto lógico. La paridad marcada y espaciada en realidad no verifican el estado de los bits de datos; simplemente fija el bit de paridad en estado lógico alto para la marcada, y en estado lógico bajo para la espaciada. Esto permite al dispositivo receptor conocer de antemano el estado de un bit, lo que serviría para determinar si hay ruido que esté afectando de manera negativa la transmisión de los datos, o si los relojes de los dispositivos no están sincronizados.

2.4.2 ¿QUÉ ES RS-232?

RS-232 (Estándar ANSI/EIA-232) es el conector serial hallado en las PCs IBM y compatibles. Es utilizado para una gran variedad de propósitos, como conectar un ratón, impresora o modem, así como instrumentación industrial. Gracias a las mejoras que se han ido desarrollando en las líneas de transmisión y en los cables, existen aplicaciones en las que se aumenta el desempeño de RS-232 en lo que respecta a la distancia y velocidad del estándar. RS-232 está limitado a comunicaciones de punto a punto entre los dispositivos y el puerto serial de la computadora. El hardware de RS-232 se puede utilizar para comunicaciones seriales en distancias de hasta 50 pies.



Pin	Signal	Pin	Signal
1	Data Carrier Detect	6	Data Set Ready
2	Received Data	7	Request to Send
3	Transmitted Data	8	Clear to Send
4	Data Terminal Ready	9	Ring Indicator
5	Signal Ground		

Figura 2.17: Pines del conector DB-9

Funciones de los pines en RS-232:

Datos: TXD (pin 3), RXD (pin 2)

Handshake: RTS (pin 7), CTS (pin 8), DSR (pin 6), DCD (pin 1), DTR (pin 4)

Tierra: GND (pin 5)

Otros: RI (pin 9)

2.4.3 ¿QUÉ ES RS-485?

RS-485 (Estándar EIA-485) es una mejora sobre RS-422 ya que incrementa el número de dispositivos que se pueden conectar (de 10 a 32) y define las características necesarias para asegurar los valores adecuados de voltaje cuando se tiene la carga máxima. Gracias a esta capacidad, es posible crear redes de dispositivos conectados a un solo puerto RS-485. Esta capacidad, y la gran inmunidad al ruido, hacen que este tipo de transmisión serial sea la elección de muchas aplicaciones industriales que necesitan dispositivos distribuidos en red conectados a una PC u otro controlador para la colección de datos, HMI, u otras operaciones. RS-485 es un conjunto que cubre RS-422, por lo que todos los dispositivos que se comunican usando RS-422 pueden ser controlados por RS-485. El hardware de RS-485 se puede utilizar en comunicaciones seriales de distancias de hasta 4000 pies de cable.

Funciones de los pines en RS-485 y RS-422:

Datos: TXD+ (pin 8), TXD- (pin 9), RXD+ (pin 4), RXD- (pin 5)

Handshake: RTS+ (pin 3), RTS- (pin 7), CTS+ (pin 2), CTS- (pin 6)

Tierra: GND (pin 1)

2.4.4 ¿QUÉ ES HANDSHAKING?

El método de comunicación usado por RS-232 requiere de una conexión muy simple, utilizando sólo tres líneas: Tx, Rx, y GND. Sin embargo, para que los datos puedan ser transmitidos correctamente ambos extremos deben estar sincronizados a la misma velocidad. Aun y cuando este método es más que suficiente para la mayoría de las aplicaciones, es limitado en su respuesta a posibles problemas que puedan surgir durante la comunicación; por ejemplo, si el receptor se comienza a sobrecargar de información. Es en estos casos cuando el intercambio de pulsos de sincronización, o *handshaking*, es útil. En esta sección se describirán brevemente las tres formas más populares de *handshaking* con RS-232: handshaking por software, handshaking por hardware, y XModem.

a. **Handshaking por software:** Ésta será la primera forma de *handshaking* que discutiremos. Esta forma de sincronización utiliza bytes de datos como caracteres de control, de manera similar a como GPIB utiliza las cadenas de caracteres como comandos. Las líneas necesarias para la comunicación siguen siendo Tx, Rx, y GND, ya que los

caracteres de control se envían a través de las líneas de transmisión como si fueran datos. La función SetXMode permite al usuario habilitar o deshabilitar el uso de dos caracteres de control: XON y XOFF. Estos caracteres son enviados por el receptor para pausar al transmisor durante la comunicación.

A manera de ejemplo, asúmase que el transmisor comienza a enviar datos a alta velocidad. Durante la transmisión, el receptor se da cuenta que el búfer de entrada se está llenando debido a que el CPU está ocupado con otras tareas. Para pausar temporalmente la transmisión, el receptor envía XOFF (cuyo valor es típicamente 19 decimal, o 13 hexadecimal) hasta que el búfer se vacíe. Una vez que el receptor está preparado para recibir más datos envía XON (cuyo valor es típicamente 17 decimal, u 11 hexadecimal) para continuar la comunicación. LabWindows enviará un XOFF cuando el búfer de entrada se encuentre a la mitad de su capacidad. Además, en caso que la transmisión inicial de XOFF haya fallado, LabWindows enviará de nuevo un XOFF cuando el búfer alcance un 75% y 90% de su capacidad. Para que funcione correctamente, es necesario que el transmisor esté utilizando el mismo protocolo.

La mayor desventaja de este método es además lo más importante a considerar: los números decimales 17 y 19 son ahora los límites para la transmisión. Cuando se transmite en ASCII, esto no importa mucho ya que estos valores no representan carácter alguno. Sin embargo, si la transmisión de datos es en binario, lo más probable es que estos valores sean transmitidos como datos regulares y falle la comunicación.

b. **Handshaking por hardware:** El segundo método de *handshaking* utiliza líneas de hardware. De manera similar a las líneas Tx y Rx, las líneas RTS/CTS y DTR/DSR trabajan de manera conjunta siendo un par la entrada y el otro par la salida. El primer par de líneas es RTS (por sus siglas en inglés, *Request to Send*) y CTS (*Clear to Send*). Cuando el receptor está listo para recibir datos, cambia la línea RTS a estado alto; este valor será leído por el transmisor en la línea CTS, indicando que está libre para enviar datos. El siguiente par de líneas es DTR (por sus siglas en inglés, *Data Terminal Ready*) y DSR (*Data Set Ready*). Estas líneas se utilizan principalmente para comunicación por modem, permiten al puerto serial y modem indicarse mutuamente su estado. Por ejemplo, cuando el modem se encuentra preparado para que la PC envíe datos, cambia la

línea DTR a estado alto indicando que se ha realizado una conexión por la línea de teléfono. Este valor se lee a través de la línea DSR y la PC comienza a enviar datos. Como regla general, las líneas DTR/DSR se utilizan para indicar que el sistema está listo para la comunicación, mientras que las líneas RTS/CTS se utilizan para paquetes individuales de datos.

En LabWindows, la función SetCTSMode habilita o deshabilita el uso de *handshaking* por hardware. Si el modo CTS está habilitado, LabWindows aplica las siguientes reglas:

Cuando la PC envía datos: La librería de RS-232 debe de detectar que la línea CTS se encuentra en estado alto antes de enviar datos.

Cuando la PC recibe datos: Si el puerto está abierto y el búfer de entrada puede contener más datos, la librería envía a RTS y DTR a estado alto.

Si el búfer de entrada está al 90% de su capacidad, la librería manda a estado bajo RTS pero mantiene DTR en alto.

Si el búfer de entrada está casi vacío, la librería manda a estado alto RTS y mantiene DTR en alto.

Si el puerto se cierra, la librería manda a estado bajo a RTS y DTR.

c. **Handshaking por XModem:** El modo de handshaking presentado es el protocolo de transmisión de archivos XModem. Este protocolo es muy común en comunicación por modem. Aun y cuando es más utilizado para comunicación por modem, el protocolo XModem puede ser utilizado directamente entre otros dispositivos. En LabWindows, la implementación de XModem se mantiene oculta para el usuario. Mientras la PC se conecte a otro dispositivo que utilice el protocolo XModem, se pueden utilizan las funciones de LabWindows para transferir datos de un lado a otro. Estas funciones son XModemConfig, XModemSend, y XModemReceive.

   XModem utiliza un protocolo basado en los siguientes parámetros: start_of_data, end_of_trans, neg_ack, ack, wait_delay, start_delay, max_tries, packet_size. Estos parámetros deben de ser comunes en ambos lados de la comunicación, y el estándar XModem contiene la definición estándar de éstos; sin embargo, se pueden modificar utilizando la función XModemConfig de LabWindows para cumplir cualquier otro requerimiento. Los parámetros en XModem funcionan de la siguiente manera: el receptor envía el caracter "neg_ack". Esto indica al transmisor

que ya está listo para recibir datos. El receptor continuará enviado el caracter "neg_ack" en intervalos de tiempo de duración de "start_delay" hasta que iguale la cuenta de "max_tries" o reciba "start_of_data" del transmisor. Si el receptor intenta comunicarse con el transmisor la misma cantidad de veces que "max_tries", informará al usuario que no fue posible comunicarse con el transmisor. Si el receptor sí recibe el "start_of_data" del transmisor, leerá el paquete de información que sigue. Este paquete contiene el número de paquete, el complemento del número de paquete para fines de verificación de errores, el paquete actual de datos con una cantidad de bytes igual a "packet_size", y un checksum para más verificación de errores. Después de recibir el paquete, el receptor mandará llamar el "wait_delay", y luego enviará el "ack" al transmisor. Si el transmisor no recibe el "ack", intentará de reenviar el paquete de datos una cantidad de veces igual a "max_tries" o hasta que reciba el "ack". Si nunca recibe el "ack", informará al usuario que hubo un fallo al momento de querer transferir el archivo.

Los datos deben de ser enviados en paquetes con una cantidad de bytes igual a "packet_size". Debido a esto, cuando se está enviando el último paquete y no se tiene la cantidad suficiente de información válida para llenarlo, el protocolo llenará el paquete con el caracter ASCII nulo (0).

Esto puede causar que el archivo recibido sea más grande que el original. Es importante recordar que no hay que usar XON/XOFF con el protocolo XModem, ya que el número de paquete durante la transferencia por XModem se incrementará conforme se envían los caracteres XON/XOFF, lo que puede causar una falla en la comunicación.

CAPÍTULO 3

TRABAJO REALIZADO POR CADA GRUPO

En este capítulo se observara el proceso mediante el cual se ha completado el proyecto combinando ciertos ejercicios los cuales nos permitirán realizar en etapas los requerimientos solicitados para el correcto funcionamiento de nuestro controlador.

Se tratara ejercicios tales como el manejo del LCD del AVR Butterfly para la interacción con el usuario, el manejo del joystick para la posibilidad de elegir opciones por medio de un menú, manejo del módulo UART en el AVR, Transmisión UART entre tarjetas LPC Y finalmente manejo de comunicación UART entre la tarjeta AVR y la LPC

3.1 EJERCICIO 1: ENCENDIDO DE UN LED MEDIANTE UNA BOTONERA CONECTADA AL MICROCONTROLADOR 1769

Mediante el uso de una botonera y la tarjeta LPC EXPRESSO 1769 seremos capaces de trabajar con señales de control en este caso será comprobado por medio de un led conectado a la salida de la tarjeta que será encendido al detectar algún cambio en el valor lógico de la botonera con este ejercicio seremos capaces de manipular las entradas y salidas de nuestra tarjeta

3.1.1 DIAGRAMA DE FLUJO

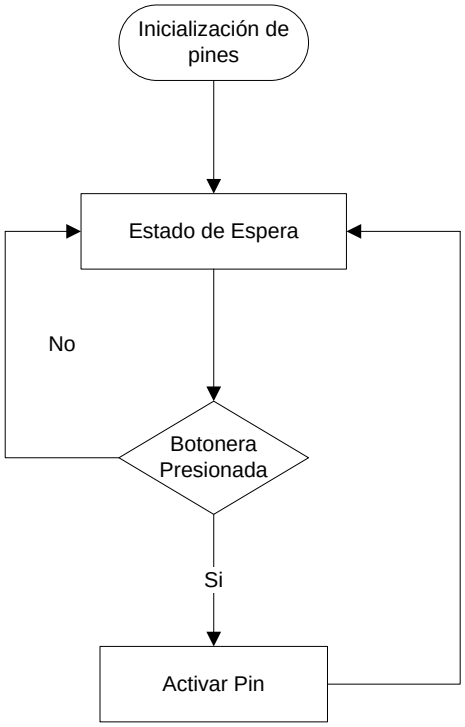


Figura 3.1: Esquema del encendido de un led mediante la LPC

3.1.2 ALGORITMO

1. Configurar los pines como entradas y salidas
2. Presionar la botonera
3. Encender el pin establecido en este caso un led

3.1.3 CÓDIGO FUENTE

```
**********************************************************************/
#include<cr_section_macros.h>
#include <NXP/crp.h>

// Variable to store CRP value in. Will be placed automatically
// by the linker when "Enable Code Read Protect" selected.
// See crp.h header for more information
__CRP const unsigned int CRP_WORD = CRP_NO_CRP ;

#include "lpc17xx.h"
#include "type.h"
int main (void)
```

```
{
        uint32_ti, j, BOTON;
        /* SystemClockUpdate() updates the SystemFrequency variable */
        SystemClockUpdate();
        LPC_GPIO2->FIODIR = 0xFFFFFEFF;              /* P2.xx defined as Outputs */
        LPC_GPIO2->FIOCLR = 0xFFFFFFFF;              /* turn off all the LEDs */
        LPC_GPIO1->FIOCLR = 0xFFFFFFFF;

        LPC_GPIO2->FIOMASK= 0x00000000;
        while(1)
        {
                                        if(LPC_GPIO2->FIOPIN==0x00000100 )
                {
                        for(i = 0; i< 8; i++)
                        {
                        LPC_GPIO2->FIOSET = 1 <<i;
                        for(j = 1000000; j > 0; j--);
                        }
                        LPC_GPIO2->FIOCLR = 0x000000FF;
                        for(j = 1000000; j > 0; j--);
                }
```

```
                if(LPC_GPIO2->FIOPIN == 0x00000000)
                {
                        for(i = 8; i> 0; i--)
                        {
                                LPC_GPIO2->FIOSET = 1 <<i;
                               for(j = 1000000; j > 0; j--);
                        }
                        LPC_GPIO2->FIOCLR = 0x000000FF;
                        for(j = 1000000; j > 0; j--);
                }
        }
}
```

CONCLUSIÓN

Este ejercicio nos permite comprender el correcto manejo de los puertos de entrada y salida de la tarjeta LPCXPRESSO ya que en el podemos configurar los pines a fin de usarlos en este caso como indicadores visuales.

3.2 EJERCICIO 2: ENVIÓ DE SEÑAL ENTRE DOS MICROCONTROLADORES POR MEDIO DE LA INTERFAZ RS232

Para este ejercicio debemos trabajar con dos tarjetas LPCXPRESSO 1769 y conectarlas mediante la interfaz RS232 que posee cada tarjeta, por lo que usaremos la conexión UART, para este diseño usamos los puertos 3 de UART.

Para confirmar si la información está transmitiéndose de la manera correcta se conectara un juego de leds a la salida del GPIO de la tarjeta receptora los cuales ejecutaran orden dependiendo de la botonera presionada en la tarjeta transmisora.

3.2.1 CONFIGURACIÓN DE INTERFAZ RS232

Debemos conectar los pines correspondientes en este caso el pin de transmisión del UART3 (TX) de una de las tarjetas configuradas con el pin de recepción del UART3 (RX) de la tarjeta configurada como receptora .

3.2.2 DIAGRAMAS DE FLUJO

3.2.2.1 TRANSMISIÓN

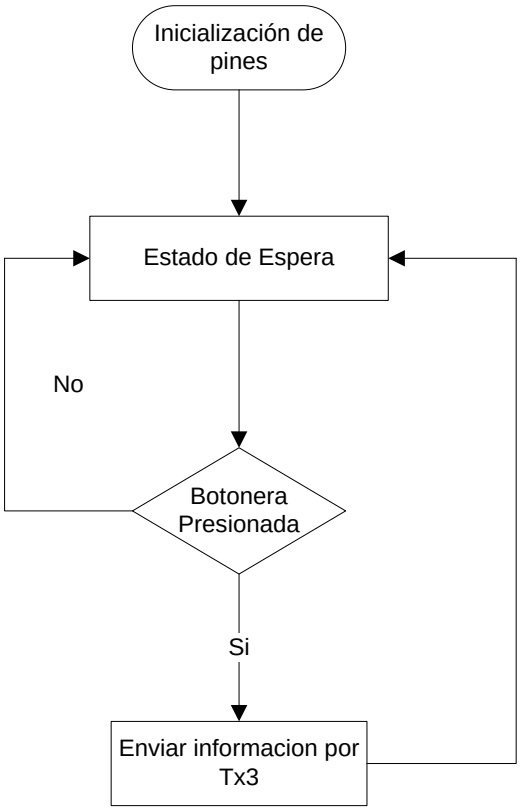


Figura 3.2: Esquema de la transmisión UART mediante la tarjeta LPC

3.2.2.2 RECEPCIÓN

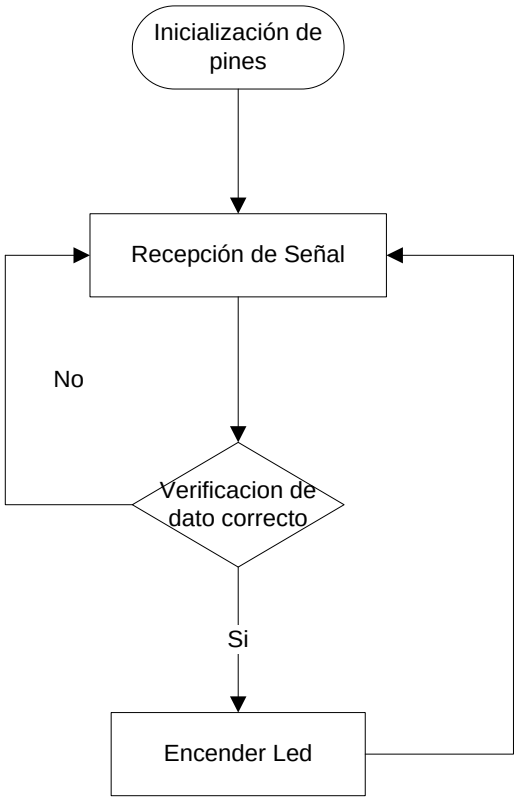


Figura 3.3: Esquema de recepción UART mediante la tarjeta LPC

3.2.3ALGORITMO

3.2.3.1 TRANSMISIÓN

1. Se configuran los pines que van a ser usados como entradas y salidas incluyendo los pines para el UART3

2. Se mantiene en un estado de espera a la tarjeta mientras el usuario no envié la orden preestablecida la cual será transmitida luego de presionar la botonera
3. El dato a transmitir se guarda en el buffer de transmisión y es transmitido de manera serial

3.2.3.2 RECEPCIÓN

1. Configurar los pines como entradas o salidas incluyendo los pines del UART3
2. Nos encontramos en un estado de espera hasta que el microcontrolador detecte que algún dato ha sido enviando por medio de comunicación serial
3. Nos encargamos de verificar el dato en caso de no ser el correcto se procede a descartarlo y no tomar ninguna acción hasta recibir el dato correcto
4. Una vez que el dato ha sido verificado se procede a realizar la opción correspondiente al dato en este caso a encender el led indicando que la transmisión resulto un éxito

3.2.4 CÓDIGO FUENTE

3.2.4.1 CÓDIGO DE TRANSMISIÓN

```
#include<cr_section_macros.h>
#include<NXP/crp.h>
#include "lpc17xx.h"
#include "type.h"
#include "uart.h"
#include <string.h>
/*extern volatile uint32_t UART3Count;
extern volatile uint8_t UART3Buffer[BUFSIZE];*/

int main (void)
{
	const char* encender = 'a';
	uint32_ti, j, BOTON;
	UARTInit(3, 9600);	/* baud rate setting */
	LPC_GPIO1->FIODIR = 0xFFFFFEFF;		/* P2.xx defined as Outputs
*/
	LPC_GPIO1->FIOCLR = 0xFFFFFFFF;		/* turn off all the LEDs */
	LPC_GPIO1->FIOMASK= 0x00000000;
```

```
        while(1)
        {
                if(LPC_GPIO1->FIOPIN==0x00000000)
                {
                        UARTSend(3, (uint8_t *)encender , strlen(encender) );
                }
         }
}
```

3.2.4.2 CÓDIGO DE RECEPCIÓN

```
#include "LPC17xx.h"
#include "type.h"
#include "uart.h"
#include <string.h>

extern volatile uint32_t UART3Count;
extern volatile uint8_t UART3Buffer[BUFSIZE];
.
*****************************************************************************/
int main (void)
```

```
{
        uint32_ti, j;
        LPC_GPIO2->FIODIR = 0xFFFFFFFF;          /* P2.xx defined as Outputs
*/
        LPC_GPIO2->FIOCLR = 0xFFFFFFFF;
        UARTInit(3, 9600);  /* baud rate setting */


        /* Loop forever */
        while (1)
        {
                if ( UART3Count != 0 )
                {
                        LPC_UART3->IER = IER_THRE | IER_RLS;
                        /* Disable RBR */
                        if(UART3Buffer=="a")
                        {
                                LPC_GPIO2->FIOSET = 1 << 1;
                                for(j = 1000000; j > 0; j--);
                                 LPC_GPIO2->FIOCLR = 0x000000FF;
```

```
                        for(j = 1000000; j > 0; j--);
                    }
                    UART3Count = 0;
                    LPC_UART3->IER = IER_THRE | IER_RLS | IER_RBR;
                    /* Re-enable RBR */
                }
            }
        }
        ************************************************************/
```

CONCLUSIÓN

Por medio de este ejercicio podemos comprender el uso del módulo UART de las tarjetas LPCXPRESSO lo que será parte fundamental para la realización del proyecto final aquí hemos comprobado que para establecer una correcta comunicación se debe configurar los parámetros adecuadamente.

3.3 ENVIÓ DE DATOS POR MEDIO DE LA TARJETA AVR BUTTERFLY HACIA EL MICROCONTROLADOR 1769

En este ejercicio debemos programar la tarjeta AVR butterfly con el código adecuado para así luego mediante el uso del joystick que viene incluido en esta tarjeta poder enviar los respectivos códigos hacia el microcontrolador localizado en otra tarjeta.

Para este envío de datos debemos utilizar la transmisión UART y basarnos en el ejercicio 3.2, así que fácilmente podíamos enviar datos, solo que esta vez no es entre dos Tarjetas LPCXPRESSO sino entre una Tarjeta LPCXPRESSO y una tarjeta AVR BUTTERFLY.

El envío de la tarjeta AVR BUTTERFLY sería el siguiente:

- Arriba
- Abajo
- Derecha
- Izquierda
- Centro

Mediante estos comandos nosotros encendimos 5 leds conectados a la salida del puerto 1 del microcontrolador, cada led representa un comando, así que solo debe encenderse uno a la vez

3.3.1 DIAGRAMA DE FLUJO

3.3.1.1 TRANSMISIÓN DESDE BUTTERFLY

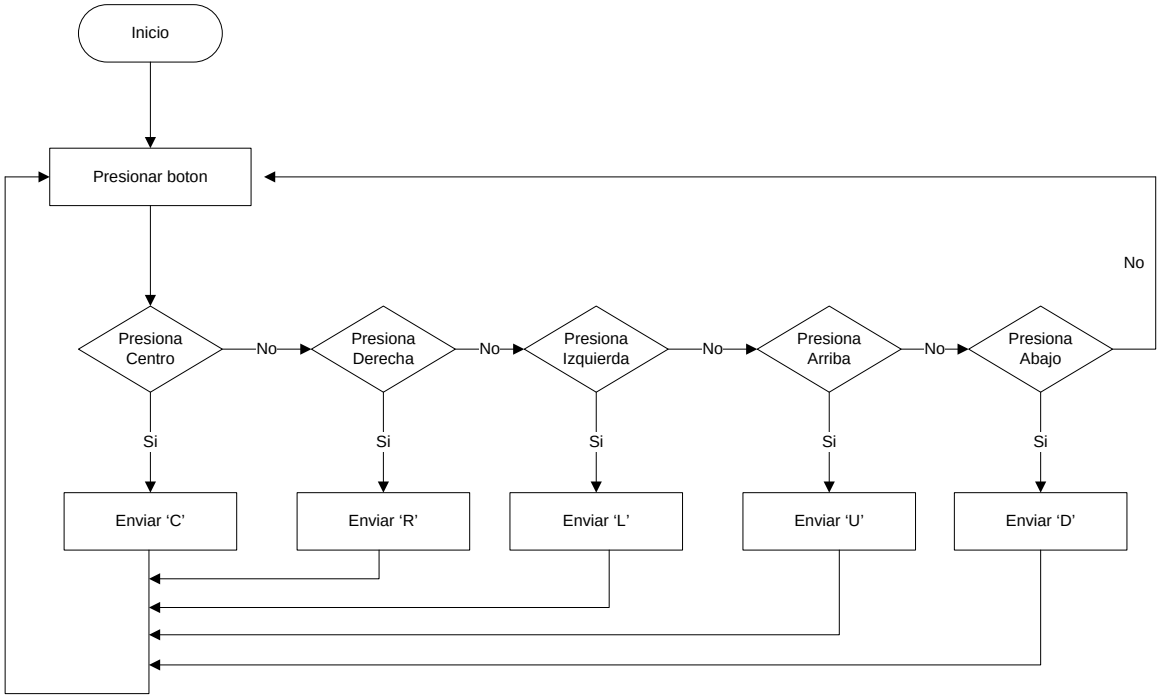


Figura 3.4: Esquema de transmisión UART en la tarjeta BUTTERFLY

3.3.1.2 RECEPCIÓN LPC

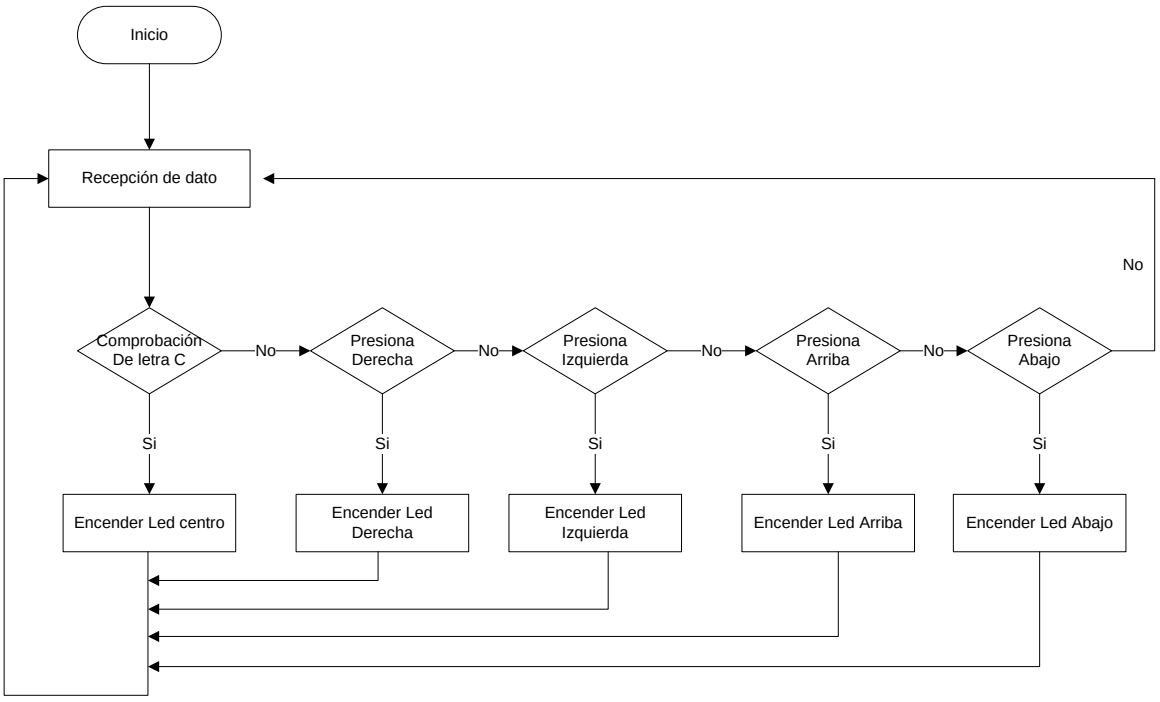


Figura 3.5: Recepción UART mediante la tarjeta LPC y la BUTTERFLY

3.3.2 ALGORITMOS

3.3.2.1 TRANSMISOR

1. Energizar la butterfly
2. Inicialización de las variables entradas y salidas y parámetros de velocidad para trabajar con comunicación serial
3. Seleccionar cualquier botón

4. Se transmite una trama el cual está formada por un caracter el cual será leído por medio de la tarjeta receptora
5. Regresamos a un estado de espera en el caso de que se desee realizar otra opción

3.3.2.2 RECEPTOR

1. Energizar la LPC
2. Configurar los pines necesario como entradas y salidas
3. Espera a recibir un dato de la BUTTERFLY cada letra corresponde a una acción diferente.
4. En caso de obtener una trama correcta se realizara la activación de un pin dependiendo del dato recibido y nos mantenemos en un estado de espera hasta recibir otra instrucción.

3.3.3 CÓDIGO FUENTE

3.3.3.1 TRANSMISOR BUTTERFLY

```
#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"

int main(void)
{

        PGM_P statetext = PSTR("CONTROL DE MOTORES");
        uint8_t input;
        char *cadena;
```

```
inti;
 // Disable Analog Comparator (power save)
 ACSR = (1<<ACD);
// Disable Digital input on PF0-2 (power save)
 DIDR0 = (7<<ADC0D);
 // Enable pullups
PORTB = (15<<PB0);
PORTE = (15<<PE4);
Button_Init();              // Initialize pin change interrupt on joystick
LCD_Init();                 // initialize the LCD
CLKPR = (1<<CLKPCE);        // set Clock Prescaler Change Enable
 // set prescaler = 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();          // Read buttons
switch (input)
        {
                case KEY_ENTER:
                statetext = PSTR("CENTRO");
                Usart_Tx('C');
                break
                case KEY_NEXT:
                statetext = PSTR("RIGHT");
                Usart_Tx('R');
                break;
                case KEY_PREV:
                statetext = PSTR("LEFT");
                Usart_Tx('L');
                break;
                case KEY_PLUS:
                statetext = PSTR("UP");
                Usart_Tx('U');
                break;
                case KEY_MINUS:
```

```
                    statetext = PSTR("DOWN");
                    Usart_Tx('D');
                    break;
                    default:
                    break;
                }
            }
            return 0;
}
```

3.3.3.2 RECEPTOR LPC

```
**************************************************************************/
#include "LPC17xx.h"
#include "type.h"
#include "uart.h"
#include <string.h>

extern volatile uint32_t UART3Count;
extern volatile uint8_t UART3Buffer[BUFSIZE];

/*****************************************************************************
**   Main Function  main()
```

```
This program has been test on LPCXpresso 1700.
*****************************************************************************/
int main (void)
{
	uint32_t i, j;

	LPC_GPIO2->FIODIR = 0xFFFFFFFF;		/* P2.xx defined as Outputs */
	LPC_GPIO2->FIOCLR = 0xFFFFFFFF;
	 LPC_GPIO2->FIOSET = 0x000000FF;

	UARTInit(3, 9600);	/* baud rate setting */


	/* Loop forever */
	while (1)
	{
		if ( UART3Count != 0 )
		{
			LPC_UART3->IER = IER_THRE | IER_RLS;
	/* Disable RBR */
			if(*UART3Buffer==0x55){
				 LPC_GPIO2->FIOSET = 0x000000FF;
				LPC_GPIO2->FIOCLR = 1 << 1;
```

```
}
if(*UART3Buffer==0x44){
        LPC_GPIO2->FIOSET = 0x000000FF;
                        LPC_GPIO2->FIOCLR = 1 << 2;

                }
if(*UART3Buffer==0x52){
        LPC_GPIO2->FIOSET = 0x000000FF;
                        LPC_GPIO2->FIOCLR = 1 << 3;

                }
if(*UART3Buffer==0x4c){
        LPC_GPIO2->FIOSET = 0x000000FF;
                        LPC_GPIO2->FIOCLR = 1 << 4;

                }
if(*UART3Buffer==0x43){

        LPC_GPIO2->FIOSET = 0x000000FF;

                }
```

```
                        UART3Count = 0;

                        LPC_UART3->IER = IER_THRE | IER_RLS | IER_RBR;
        /* Re-enable RBR */

                }

        }

}


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

**                      End Of File

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

CAPÍTULO 4

PRUEBAS Y SIMULACIONES

4.1 RESUMEN DEL CAPÍTULO

En este capítulo presentamos lo que corresponde a la simulación del correcto funcionamiento de las partes que conforman al transmisor y al receptor del proyecto desarrollado con el fin de tener una mejor comprensión sobre el mismo al revisar minuciosamente su comportamiento en una plataforma de simulación que nos permitirá observar errores antes pasar a su ejecución

4.2 SIMULACIÓN DEL TRANSMISOR

El circuito transmisor está conformado por la tarjeta BUTTERFLY que contiene un microcontrolador ATMEGA169 que se encargara de ejecutar todas las instrucciones que le hayamos cargado en la memoria previamente, de una

pantalla LCD que nos permitirá tener una interacción con el usuario y un joystick que también está incluido en la tarjeta el cual nos permitirá hacer movimientos en 5 direcciones

Como se puede observar en la figura al ejecutar la simulación de lo que es la tarjeta AVR Butterfly en el LCD se muestra un mensaje en este caso el tema del proyecto “CONTROL DE MOTORES” que nos indicara que la tarjeta estará totalmente funcional y lista para recibir una instrucción seleccionada desde el joystick la cual provocara que el mensaje de la pantalla cambie con el fin de conocer la instrucción enviada

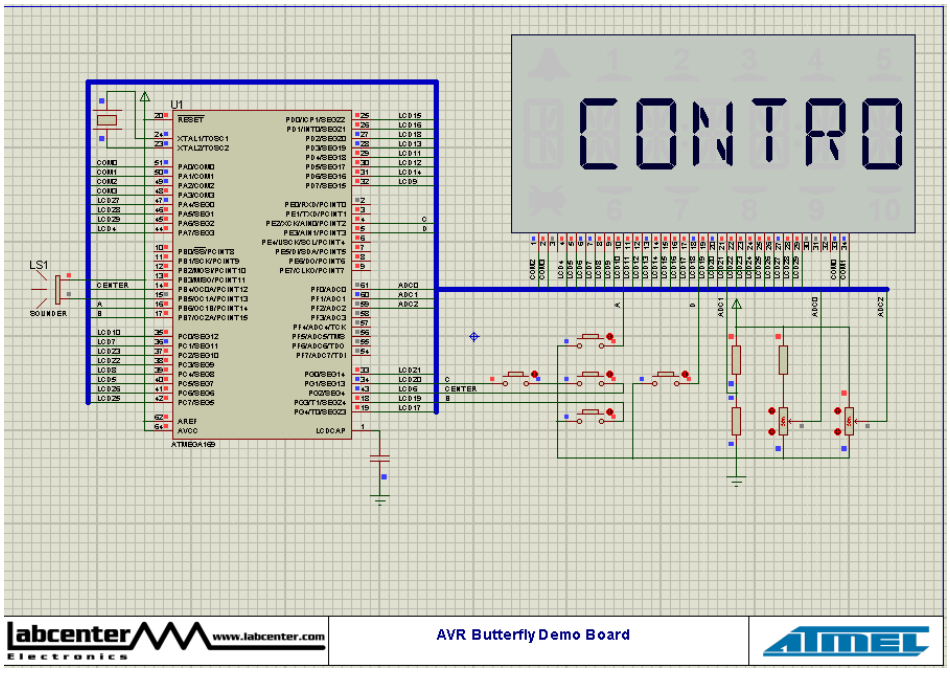


Figura 4.1: .Mensaje de bienvenida en el LCD de la AVR Butterfly (Control).

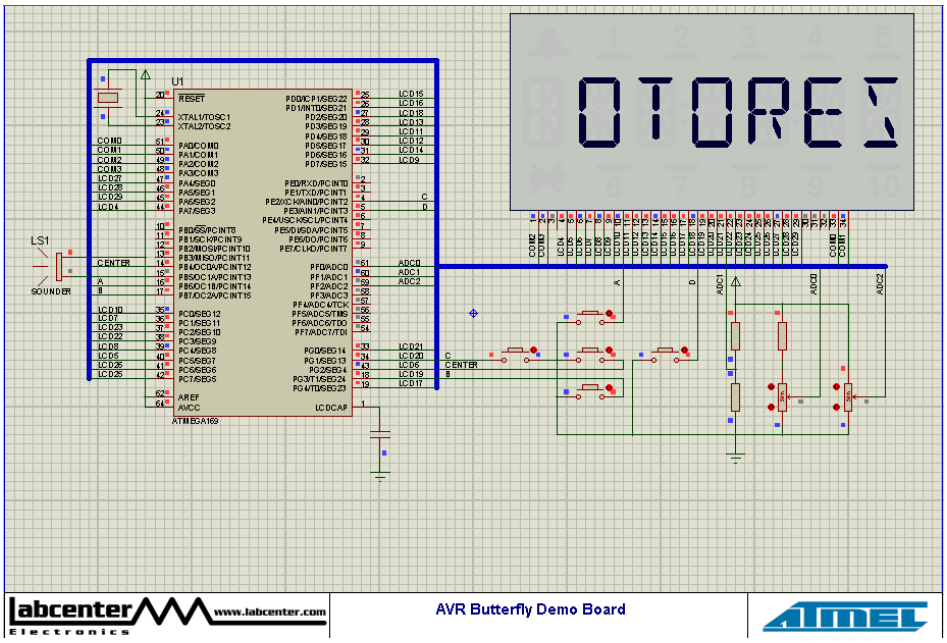


Figura 4.2: Mensaje de bienvenida en el LCD de la AVR Butterfly (Motores).

A continuación que sabemos que el proyecto arranco de una manera correcta e indica que esta lista para trabajar con el mensaje de bienvenida procedemos a operar los controles que nos permitirán transmitir cada una de las opciones necesarias para manejar el funcionamiento de un motor BLCD los cuales son detallados a continuación

La primera opción a considerar es la opción UP la cual se mostrara en el LCD si el usuario llegase a mover el joystick hacia arriba con esta opción podremos controlar la velocidad a la que el motor gira ya que al ser recibida aumentara la velocidad de rotación.

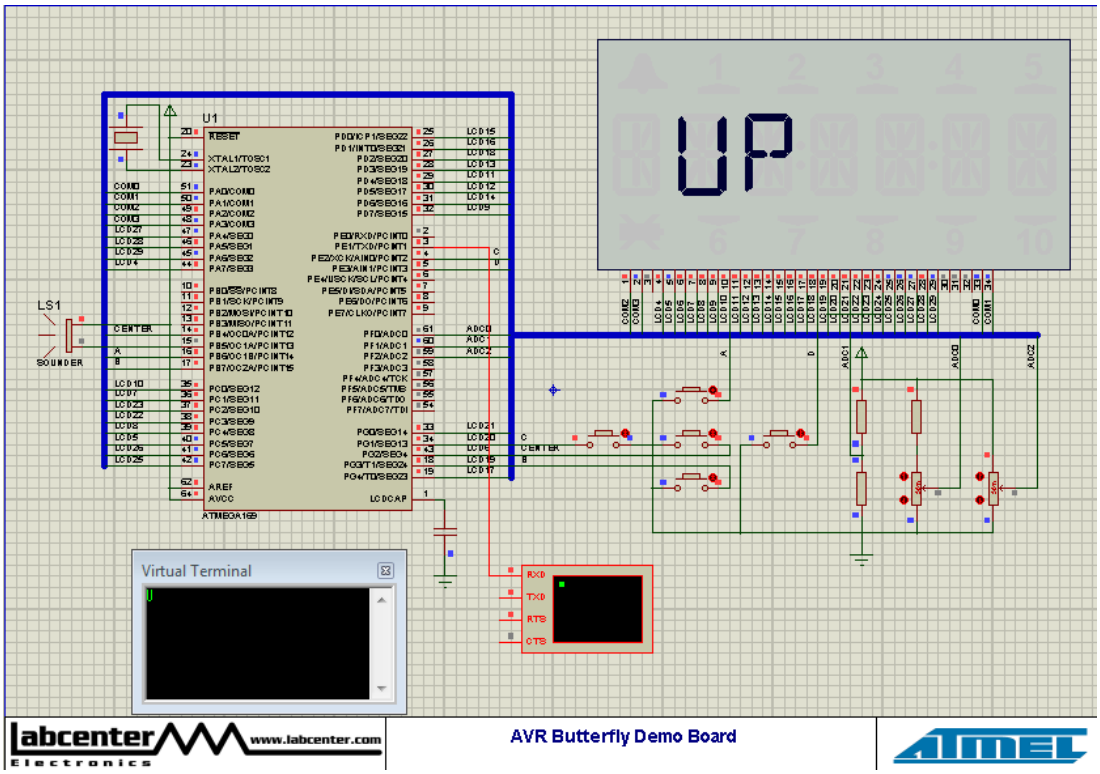


Figura 4.3: Instrucción “UP” en LCD.

La segunda opción es la instrucción DOWN la cual se mostrara en la pantalla del LCD al mover el joystick hacia abajo con esta opción también se podrá controlar la velocidad a la que el motor gira pero con la diferencia de que no aumentara su velocidad más bien esta se verá reducida con la misma razón en la que fue aumentada.

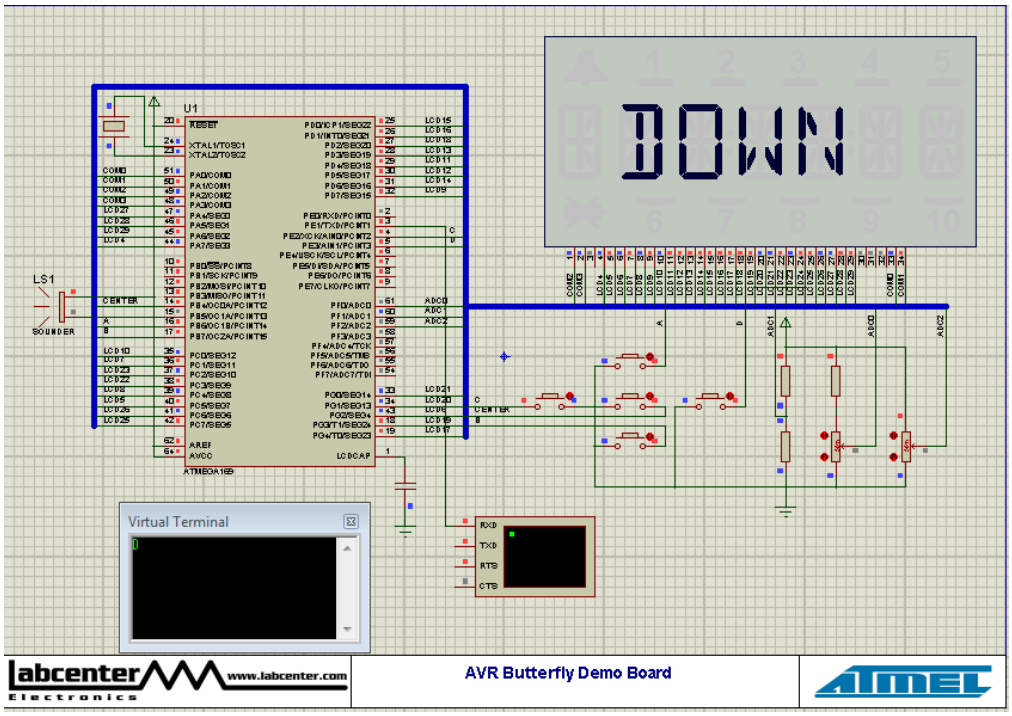


Figura 4.4: Instrucción “DOWN” en LCD.

La tercera opcion es la intruccion RIGHT la cual sera mostrada en al pantalla al mover el joystick hacia la derecha esta opcion nos permitira controlar el sentido de giro del motor asi al ser presionada en motor girar de manera horaria.

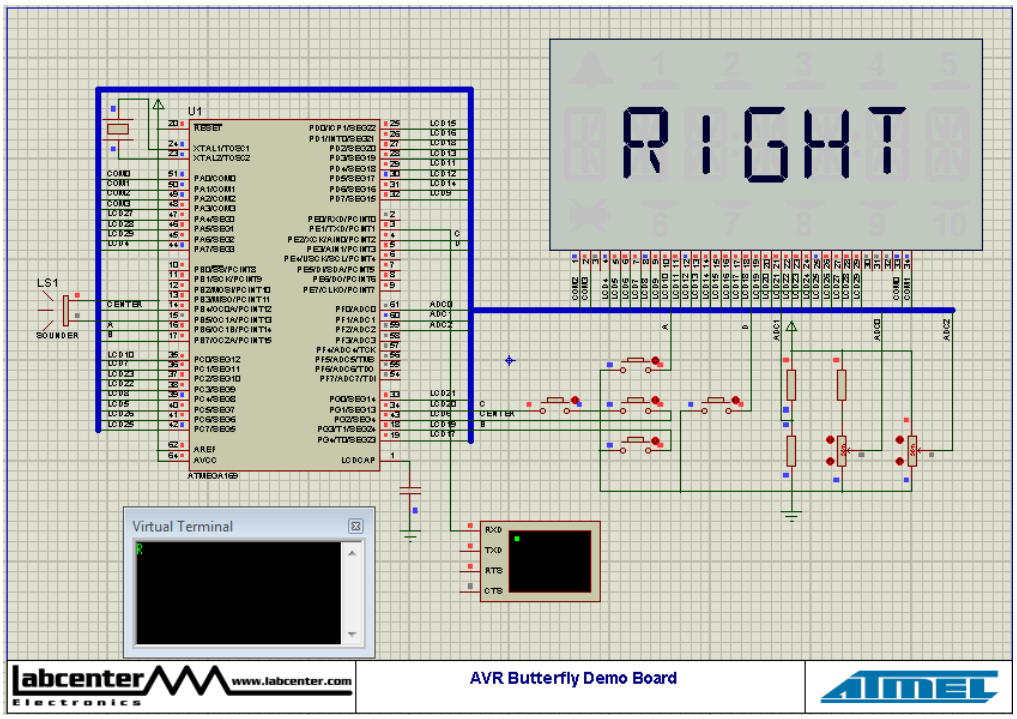


Figura 4.5: Instrucción “RIGHT” en LCD.

La cuarta opcion es la intruccion LEFT la cual sera mostrada en al pantalla al mover el joystick hacia la izquierda esta opcion nos permitira controlar el sentido de giro del motor asi al ser presionada en motor girar de manera antihoraria.

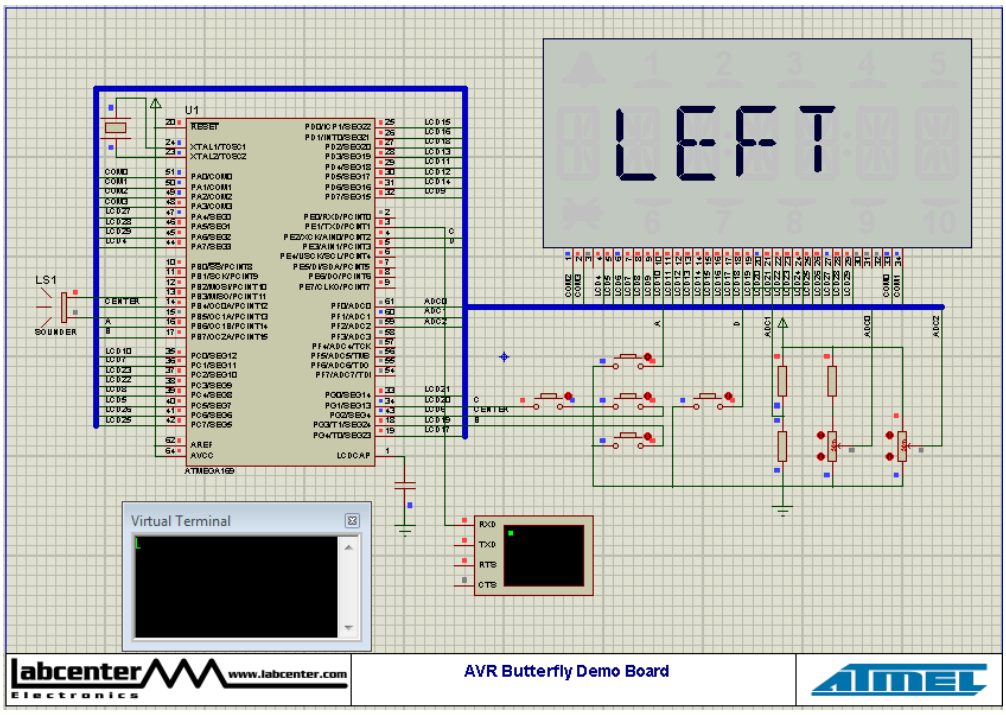


Figura 4.6: Instrucción “LEFT” en LCD.

La quinta opcion es la intruccion CENTRO la cual sera mostrada en al pantalla al presionar el joystick como si fuera un boton esta opcion nos permitira detener por completo el motor llegando a ser una funcion de apagado.

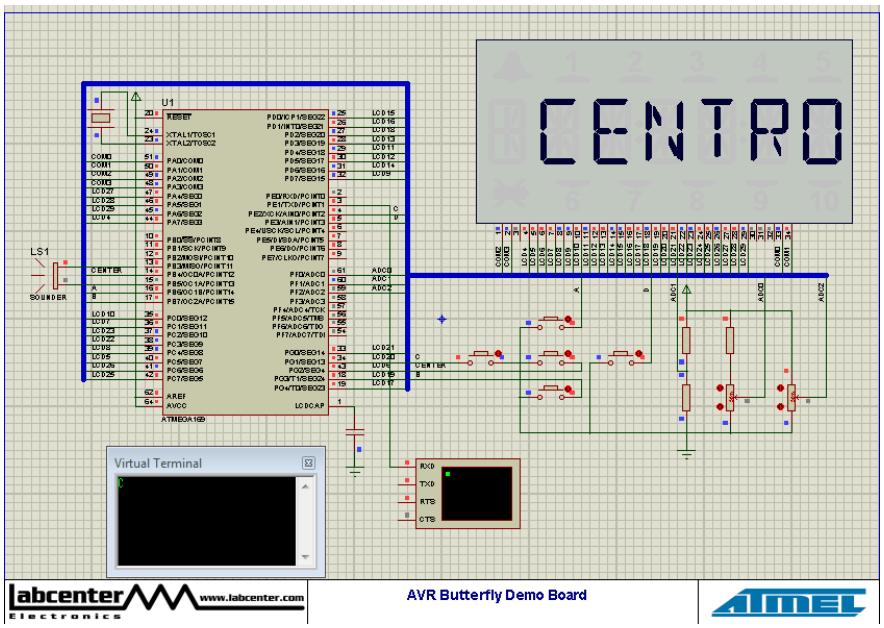


Figura 4.7: Instrucción “CENTRO” en LCD.

Para observar la información transmitida por medio de la interfaz rs232 del AVR BUTTERFLY en nuestra simulación será necesario usar un instrumento de medición virtual por lo que hemos utilizado la herramienta Virtual terminal que nos permitirá observar en este caso los caracteres enviados por comunicación serial en la figura 25 podemos ver los caracteres que se transmiten para cada opción observando de izquierda a derecha tenemos la instrucción UP, DOWN, LEFT, RIGHT, CENTRO.

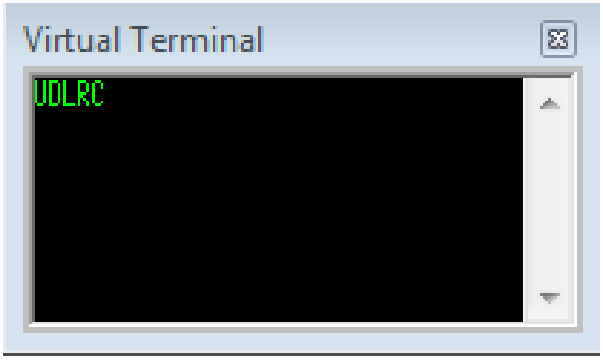


Figura 4.8: Visualización de los caracteres de transmisión

4.3 RESULTADOS DE LA SIMULACIÓN

Por medio de cada una de las simulaciones que se han realizado en este capítulo podemos llegar a la conclusión de que ya tenemos una gran perspectiva sobre el funcionamiento del proyecto debido a que en cada una de ellas pudimos comprobar que los datos que preparamos para ser enviados por medio de comunicación serial han sido transmitidos de manera exitosa, por lo que observamos que se ha logrado cumplir el objetivo planteado en nuestro proyecto, ofrecer una forma práctica de controlar un motor aplicando un tipo de comunicación en este caso la comunicación serial y logrando también una forma de comunicarnos con el usuario por medio de un LCD para lograr manejar el programa sin contratiempos.

4.4 SIMULACIÓN PROYECTO

La figura 4.9 muestra el esquema de funcionamiento del Proyecto en general, el AVR Butterfly se encarga de transmitir la instrucción a la tarjeta LPCXpresso 1769 que procesa los datos recibidos y ejecuta la instrucción que se le envió por medio de una tarjeta controladora de motores.

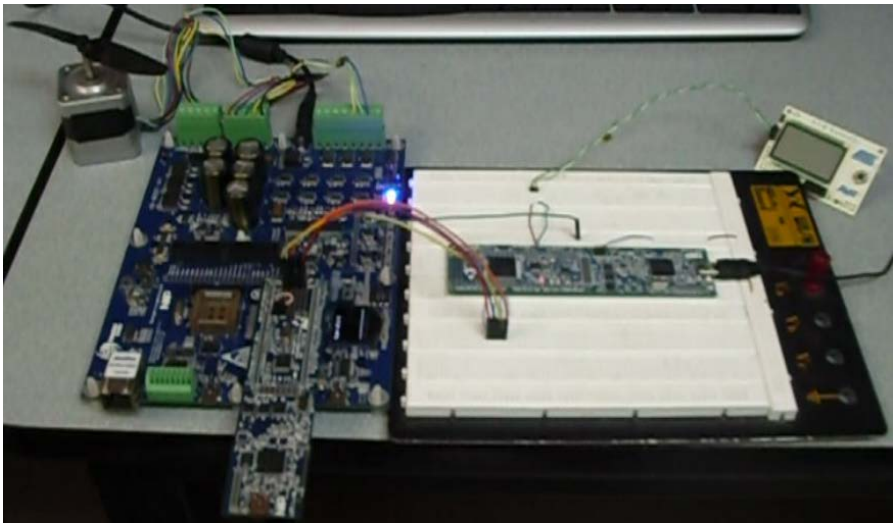


Figura 4.9: Proyecto funcionando

CONCLUSIONES

1. La comunicación UART utilizada en este proyecto es de mucha utilidad ya que podemos lograr un control a distancia desde cualquier tipo de dispositivo que soporte la tecnología de comunicación serial además de que podemos transmitir sin ningún tipo de riesgo ya que se toma las debidas precauciones al momento de enviar una señal lo que nos da seguridad al manejar un equipo.

2. Trabajar con motores BLDC resulta muy eficiente ya que son motores inteligentes y nos permiten manejarlos de una forma más sencilla, son fácilmente controlables, podemos regular su velocidad y mantenerla constante por que no existen perdidas por rozamiento como es el caso de otros motores además son motores potentes muy utilizados para cualquier tipo de aplicación por su bajo consumo de energía.

3. Podemos resolver problemas de aplicación muy interesantes como en este caso el manejo de motores ya que usando microcontroladores de gama alta tenemos muchas más posibilidades de resolver problemas de diseño ya que usan una mayor resolución como en este caso la tarjeta LPCXPRESSO usa 32 bits, lo que las hace muy útiles para aplicaciones con exigencias de precisión.

4. Innovar en el uso de nueva tecnología como es el caso de estos tipos de tarjetas resulta indispensable ya que nos familiarizamos con diferentes protocolos de comunicación con los que se puede lograr trabajos más eficientes y podemos acoplarnos en este caso al uso de un motor BLCD que es lo más utilizado en cualquier tipo de industria lo que nos hace más competitivos en el ámbito profesional.

RECOMENDACIONES

1. Al momento de programar la LPC tener mucho cuando con su manejo es decir evitar manipularla con los dedos ya que esta podría afectar el correcto funcionamiento de la tarjeta ya que los componentes son muy, también es recomendable alimentar la tarjeta solo desde una computadora ya que la polarización incorrecta de la misma podría dañarla

2. Es recomendable usar resistencias de un valor adecuado al momento de transmitir por los puertos de comunicación serial ya que cualquier cable mal conectado podría dañar los módulos de comunicación de la tarjeta.

3. La programación del AVR se debe realizar de la manera indicada mucho cuidado al momento de energizarla ya que la incorrecta manipulación de la misma podría dañar la tarjeta permanentemente además revisar si al momento de programar el cable está correctamente conectado al AVR

BIBLIOGRAFÍA

[1] ATMEL. AVR Butterfly Kit de Evaluación – Guía de Usuario

http://www.atmel.com/dyn/resources/prod_documents/doc4271.pdf Marzo 10, 2012

[2] Guerrero, Richard. Características del Kit AVR Butterfly en español

http://www.espe.edu.ec/repositorio/T-ESPE-014271.pdf Marzo 15, 2012

[3] Escobar, Carlos. Control de un motor BLDC con frenado regenerativo

http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis89.pdf. Marzo 5,2012

[4] Microchip Technology Inc. Brushless DC (BLDC) Motor Fundamentals

http://electrathonoftampabay.org/www/Documents/Motors/Brushless%20DC%20(BLDC)%20Motor%20Fundamentals.pdf Marzo 18, 2012

[5] Master ingenieros S.A. Motor brushless (sin escobillas).Características fundamentales

www.masteringenieros.com/master/Ficheros/File/motor.pdf Marzo 21, 2012

[6] NXP Semiconductors. DatasheetLPC17xx User manual

http://es.scribd.com/doc/37637774/User-manual-lpc17xx Marzo 15, 2012

[7] NXP Semiconductors. LPC1769/68/67/66/65/64/63 datasheet

www.nxp.com/documents/data_sheet/LPC1769_68_67_66_65_64_63.pdf Marzo 17,2012

[8] National Instruments. Comunicación serial

http://www.ni.com/white-paper/2895/es. Marzo 20, 2012

[9] National Instruments. Conceptos generales de la Comunicación Serial.

http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1. Marzo 20, 2012

[10] DTE. Introducción a AVR-STUDIO

http://www.todopic.com.ar/foros/index.php?topic=24309.0

ANEXOS

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.

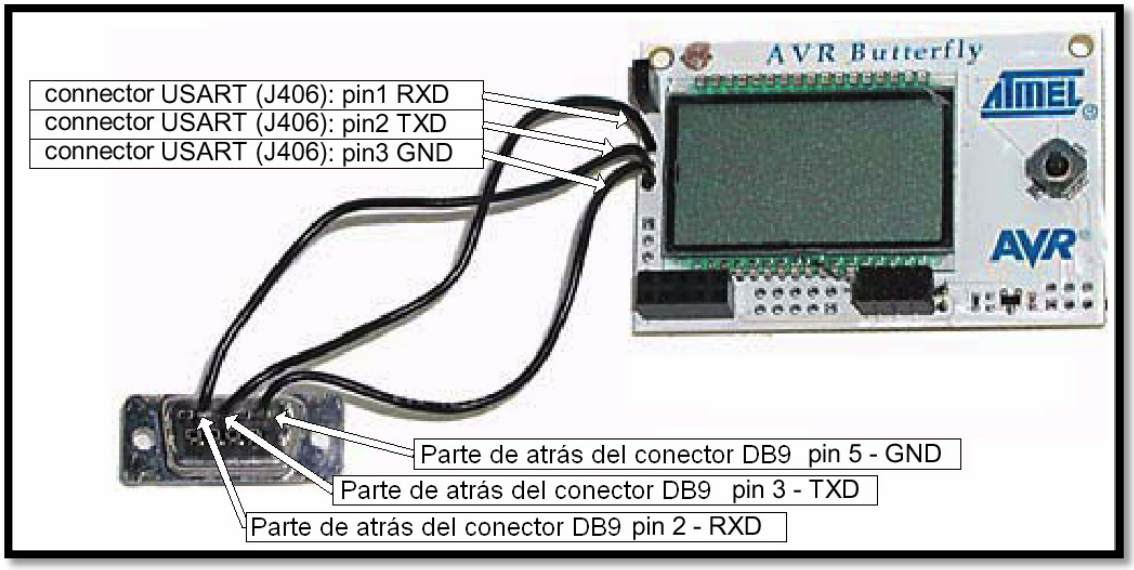
Distribución de pines para la comunicación serial entre el AVR y la PC

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

Distribución de pines, AVR Butterfly Vs. PC

En la Figura 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).

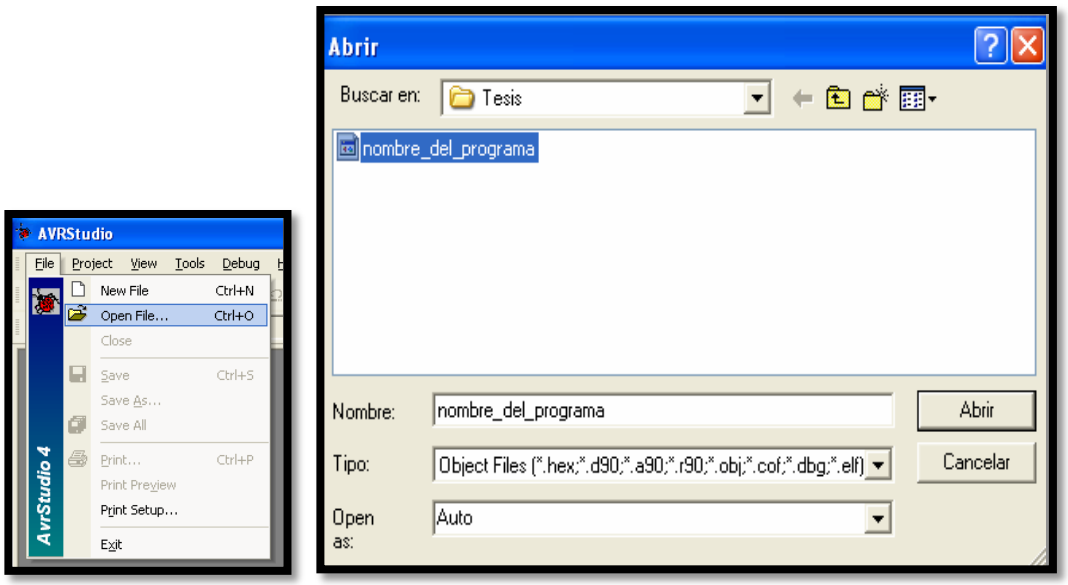


Conexiones para interfaz USART del AVR Butterfly

PROGRAMACIÓN DEL AVR BUTTERFLY

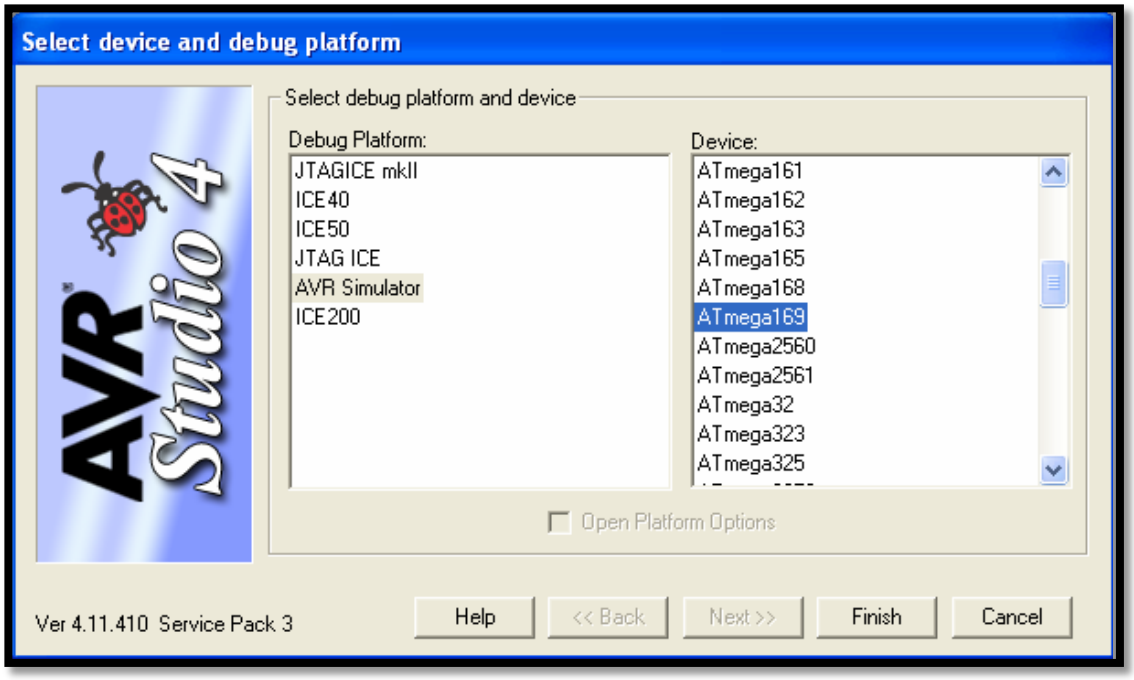
Los pasos necesarios para la Programación del Kit AVR Butterfly son los siguientes:

1. En la PC, localizar y ejecutar el AVR Studio.
2. En el menú “File” del AVR Studio seleccionar “Open File“, luego seleccionar el archivo con el que se desea programar al AVR Butterfly, tal como indica la Figura; por ejemplo: …\nombre_del_programa.cof.



AVR Studio, selección del archivo COF para depuración

3. Seleccionar el AVR Simulator y luego el ATmega169 como en la siguiente Figura



Butterfly Selección del AVR Simulator y Dispositivo ATmega169

4. Presionar “Finish”.
5. Conectar el cable serial entre la PC y el AVR Butterfly como se indica en la primera figura.
6. Resetear el AVR Butterfly cortocircuitando los pines 5 y 6 en el conector J403, conector ISP, o quitando y aplicando nuevamente la fuente de alimentación.
7. 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.

8. Una vez que se haya iniciado el AVR Prog, soltar el joystick del AVR Butterfly. Desde el AVR Prog, utilizar el botón “Browse” para buscar el archivo con la extensión *.hex con el que desea actualizar al AVR Butterfly. Una vez localizado el archivo *.hex, presionar el botón “Program”. Se notará que “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.

9. Para que empiece a ejecutarse la nueva aplicación, resetear el AVR Butterfly cortocircuitando los pines 5 y 6 en el conector J403 y presionar el joystick hacia ARRIBA. **[2]**

DISTRIBUCIÓN DE LA TARJETA LPCXPRESSO1769