ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"CONTROL MEDIANTE JOYSTICK DE TARJETA AVR BUTTERFLY (CON MICROCONTROLADOR ATMEGA169) USANDO MÓDULOS DE RADIO FRECUENCIA PARA COMUNICACIÓN CON TARJETA LPCX1769 CONTROLADORA DE MOTOR BLDC".

TESINA DE SEMINARIO

Previo a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Andrea Carolina Guerra Tapia

María José Morocho Plaza

GUAYAQUIL – ECUADOR

2012

AGRADECIMIENTO

A Dios, por ser el pilar fundamental de nuestras vidas, fuente de sabiduría, guía y luz de nuestro camino.

A nuestros padres, por su apoyo incondicional en cada meta propuesta, gracias por ayudarnos siempre.

A nuestros amigos, por estar siempre en los buenos y malos momentos.

Al Ing. Carlos Valdivieso, por su guía a través del desarrollo de nuestro proyecto.

DEDICATORIA

A nuestros padres, por estar presentes durante todo este tiempo de aprendizaje y formación profesional.

TRIBUNAL DE SUSTENTACIÓN

_____

ING. CARLOS VALDIVIESO A.

PROFESOR DEL SEMINARIO DE GRADUACIÓN

_____

ING. HUGO VILLAVICENCIO V.

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)

Andrea Carolina Guerra Tapia

María José Morocho Plaza

RESUMEN

El siguiente proyecto tiene como finalidad realizar el control de un motor sin escobillas DC mediante comunicación inalámbrica utilizando módulos HM-TR 433 de radio frecuencia para la transmisión de datos a través del joystick del AVR Butterfly que envía las instrucciones que el operador desea que realice el motor, estas instrucciones son: On/Off para el encendido y apagado del motor, aumentar velocidad, disminuir velocidad, e inversión en el sentido de giro del motor.

Estas instrucciones se receptan por la tarjeta LPC1769 que se encarga de procesar las tramas enviadas para ejecutar cada una de las instrucciones y generar el movimiento del motor.

Para la implementación del proyecto se utiliza el Kit AVR Butterfly que cuenta con un microcontrolador ATMega169, el cuál fue programado con el software AVR Studio 4, así como también una pantalla LCD que nos permite visualizar las instrucciones que están siendo enviadas y recibidas; mientras que para la tarjeta LPC1769 se utilizó la herramienta de programación LPCXpresso.

La comunicación inalámbrica entre el AVR Butterfly y la tarjeta LPC1769 la realizamos con los módulos de radiofrecuencia HM-TR433 mediante la plataforma interactiva UART para esta transmisión de datos se utiliza la modulación FSK.

ÌNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Para la elaboración de este proyecto que consiste en el control de un motor BLDC, recurrimos al uso de la tarjeta AVR Butterfly con microcontrolador ATMega 169, que consta de un joystick mediante el cual el usuario podrá operar las diferentes instrucciones que se encuentran programadas en dicha tarjeta.

La primera etapa es la parte emisora del proyecto que se encarga de transmitir las instrucciones que debe seguir nuestro motor BLDC.

Entre las instrucciones que debe seguir el motor se encuentran: encender y apagar el motor, aumentar la velocidad, disminuir la velocidad e invertir el sentido de giro del motor.

La siguiente etapa es la comunicación inalámbrica entre la tarjeta AVR Butterfly y la tarjeta LPC 1769 a través de los módulos de radiofrecuencia HM-TR 433, los cuales pueden alcanzar distancias de hasta 330mts.

La etapa de recepción consta de la tarjeta LPC 1769, la cual se encarga de receptar cada una de las instrucciones que son enviadas por la tarjeta AVR Butterfly.

Finalmente la tarjeta LPC 1769, se conecta al kit del motor BLDC a través de sus puertos de salida de dicha tarjeta para proceder a ejecutar cada una de las instrucciones especificadas anteriormente.

CAPÍTULO 1

DESCRIPCIÓN GENERAL DEL PROYECTO

El presente capítulo se encuentran los pasos previos al desarrollo del proyecto tales como:

Antecedentes basados para el planteamiento de nuestro proyecto, así como también lo que nos motivó realizarlo.

En cuanto a la identificación del problema nos hemos fijado que las comunicaciones son de gran importancia en cualquier área, en esta sección damos a conocer cuál es la necesidad que queremos cubrir.

Las limitaciones encontradas al realizar nuestro proyecto también han sido planteadas en este capítulo, las cuales han sido desarrolladas al realizar las respectivas investigaciones sobre los distintos temas que abordamos en el proyecto.

Las limitaciones encontradas al realizar nuestro proyecto también han sido planteadas en este capítulo, las cuales han sido desarrolladas al realizar las

respectivas investigaciones sobre los distintos temas que abordamos en el proyecto.

Finalmente en este capítulo se plantea los objetivos principales que se desean alcanzar

1.1 ANTECEDENTES

Al abordar el diseño de un sistema electrónico surge la necesidad de implementar partes utilizando hardware dedicado con varias de las opciones existentes.

Una opción innovadora es el MICROCONTROLADOR, que es un computador de limitadas prestaciones contenido en el chip de un circuito integrado programable y que se destina a gobernar una sola tarea con el programa que reside en su memoria.

De este, se puede elegir diferentes tipos de configuraciones, tanto en tamaño y tipo de memoria como en el tipo de puertos de entrada/salida y módulos internos (ADC, DAC, Timers/Counters,…).

Es por estas razones que el presente proyecto surgió de la necesidad de implementar los conocimientos adquiridos acerca de los microcontroladores, ya que gracias a su versatilidad se han convertido en los dispositivos más utilizados en el diseño de sistemas electrónicos; logrando así, mediante la investigación desarrollar un sistema de control de motores BLDC.

El proyecto se basa en el uso de la tarjeta AVR Butterfly que es un kit de desarrollo con el microcontrolador AVR ATMega169, el cual consta de un joystick mediante el cual se ejecutan las ordenes de control para el motor, para luego ser enviadas mediante los módulos de radio frecuencia y proceder a realizar la comunicación (UART) con la tarjeta LPCX1769 que es la encargada de controlar el motor BLDC.

La UART es un dispositivo programable en el que pueden establecerse las condiciones que se utilizarán para la transmisión. Su 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.

1.2 MOTIVACIÓN

Lo que nos motivó a realizar este proyecto es la inquietud de investigar, conocer sobre de los motores, las diferentes formas de programar distintas clases de pics. Aplicar conocimientos adquiridos y así mismo obtener nuevos.

Poder adquirir nuevas habilidades en la programación como futuros ingenieros, aprender a trabajar en grupo y conocer métodos de investigación. Desarrollar nuestra capacidad para enfrentarnos ante cualquier situación

laboral, planteando soluciones a problemas que puedan surgir durante la elaboración de algún proyecto.

Adquirir experiencia en cuanto al manejo de microcontroladores modernos como el LPC1769 y el AVR ATmega169 y sus respectivos softwares de programación LPCXpresso, AVRStudio4 y Win AVR.

1.3 IDENTIFICACIÓN DEL PROBLEMA

Uno de los problemas al realizar el control de motores es la distancia, muchas veces es necesario recorrer varios kilómetros para el control de ciertos dispositivos, los motores no son la excepción, para esto se ha pensado en utilizar una tecnología que permita el control de dichos motores sin la necesidad de movilizarse para realizarlo.

Para llevar a cabo este proyecto recurrimos al AVR Butterfly, que es un kit de desarrollo, entrenamiento y aprendizaje de microcontroladores Atmel, que contiene el siguiente hardware: un microcontrolador ATmega 169V, LCD, joystick, altavoz, cristal de 32 KHz, Data Flash de 42 Mbit, convertidor de nivel RS-232, interfaz USART, interfaz USIT, sensor de temperatura, sensor de luz, ADC, conectores para accesos a periféricos, y batería de 3V.

Para el funcionamiento de la Butterfly se emplean dos nuevas herramientas gratuitas de desarrollo AVR, estas son el IDE AVR Studio 4 y el compilador C de Win AVR, los cuales nos sirven para realizar la programación respectiva

del Joystick que funciona como una entrada para el usuario, el cual opera en cinco direcciones, incluyendo presión en el centro.

A continuación se detalla el funcionamiento de un Motor para el cual se va a 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, invierte el giro del motor.
- Si se presiona el botón derecho en el Joystick, enciende y apaga el motor.
- 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 aumenta la velocidad.

Una vez realizada la programación para el control del joystick se procede a la transmisión de datos mediante los módulos de radiofrecuencia que nos permiten realizar la comunicación inalámbrica con la tarjeta LPCX1769.

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.

Las UART o USART se diseñaron para convertir las señales que maneja la CPU y transmitirlas al exterior, ya que es un dispositivo de comunicación serial altamente flexible, sus principales características son presentadas a continuación:

- Operación Full Duplex.
- 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.

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.

1.4 OBJETIVOS PRINCIPALES

Entre los objetivos principales de nuestro proyecto tenemos:

- Controlar un motor BLDC de manera inalámbrica mediante el uso de Joystick.

- Poder realizar la comunicación usando módulos RF entre las tarjetas LPC1769 y AVR butterfly distintas.
- Desarrollar la capacidad de programar distintos pic's de las tarjetas usando los programas LPCXpresso y AVR studio 4.
- Mostrar el funcionamiento de la comunicación UART.

1.5 LIMITACIONES

En nuestro proyecto encontramos algunas limitaciones entre las primeras fueron conocer los programas usados para programar los microcontroladores, entre los cuales tenemos al programa LPCxpresso, este se usó para la tarjeta LPC1769 y usa lenguaje C para la programación del mismo, por otro lado de tenemos el programa AVR studio4 que se usó para programar la tarjeta AVR Butterfly ATmega169. En ambos casos para entender la programación hubo que buscar y entender distintos ejemplos, así mismo conocer el funcionamiento de las distintas tarjetas LPC1769 y Butterfly ATmega169, la configuración de pines y su estructura, para el entendimiento del mismo fue necesario el análisis de la hoja de datos de cada uno de ellos.

Entre las limitaciones físicas del proyecto, se debe conocer que este es un prototipo, no es aplicable a la industria ya que trabaja con un voltaje máximo de +5v. Sí se desea una aplicación se debe realizar los cambios respectivos

CAPÍTULO 2

FUNDAMENTO TEÓRICO

Los fundamentos teóricos son la base para la realización de cualquier tipo de proyecto o estudio. En este capítulo se ha se va a tratar sobre los recursos de hardware y software disponibles en la familia de microcontroladores de ATMEL, tales como Avr Studio 4, Win AVR, 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.

Los microcontroladores AVR proporcionan todos los beneficios habituales de arquitectura RISC, la característica que identifica a estos microcontroladores de ATMEL es la memoria flash y Eprom. Adicionalmente, ATMEL también proporciona en línea el entorno software (AVR estudio) que permite editar, ensamblar y simular el código fuente.

LA LPC es un ARM Cortex-M3 basado en microcontroladores para aplicaciones integradas que ofrecen un alto nivel de integración y bajo consumo de energía.

Los motores BLDC o motores sin escobillas, son de gran uso en sectores industriales gracias a su gran funcionalidad ya que son menos ruidosos y además de permitir gran ahorro de energía, en este capítulo se dará a conocer sus principales características.

2.1 AVR BUTTERFLY

2.1.1 Descripción

El Kit AVR Butterfly (Figura 2.1) se diseñó para demostrar los beneficios y las características importantes de los microcontroladores ATMEL.

El AVR Butterfly utiliza el microcontrolador AVRAT mega169V, que combina la Tecnología Flash con el más avanzado y versátil microcontrolador de 8 bits disponible.

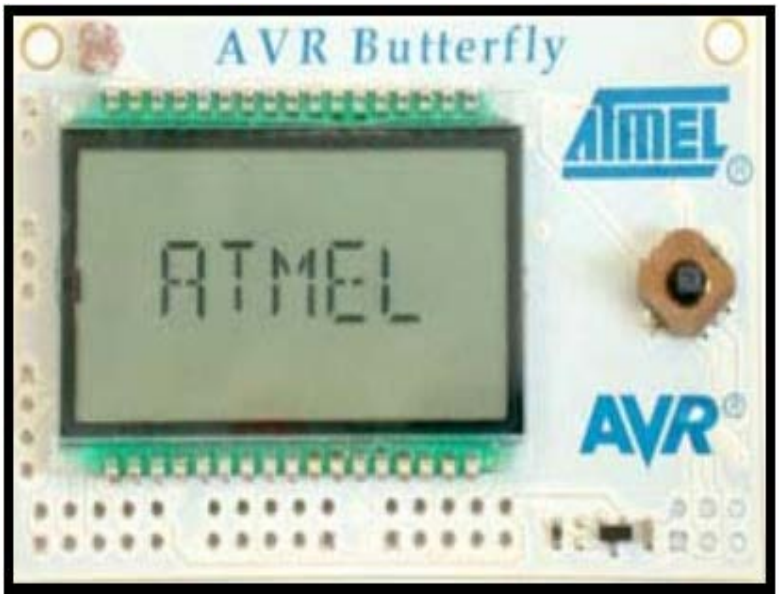


Figura 2.1: Tarjeta AVR Butterfly. [1]

En la figura 2.1 podemos observar los conectores de la tarjeta, 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.

El Kit AVR Butterfly expone las siguientes características principales:

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

    Flash, EPROM, SRAM.

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

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

2.1.2 ATMEGA 169

El ATmega 169 es un microcontrolador de baja potencia CMOS de 8 bits basado en el AVR mejorado de 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 microcontroladores CISC.

El microcontrolador ATmega 169 proporciona las siguientes características:

- 16k bytes de sistema programable.
- Flash con lectura y escritura mientras que las capacidades, 512 bytes de EPROM, 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.
- El modo Power down guarda el contenido del registro.

2.1.3 Hardware

Los siguientes recursos están disponibles en el Kit AVR Butterfly, tal como se muestra en el diagrama de bloques de la Figura 2.1.2a (Vista frontal) y 2.1.2b (Vista Posterior):

- 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 Data Flash 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 pre programada.
- Compatibilidad con el entorno de desarrollo AVR Studio4.

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

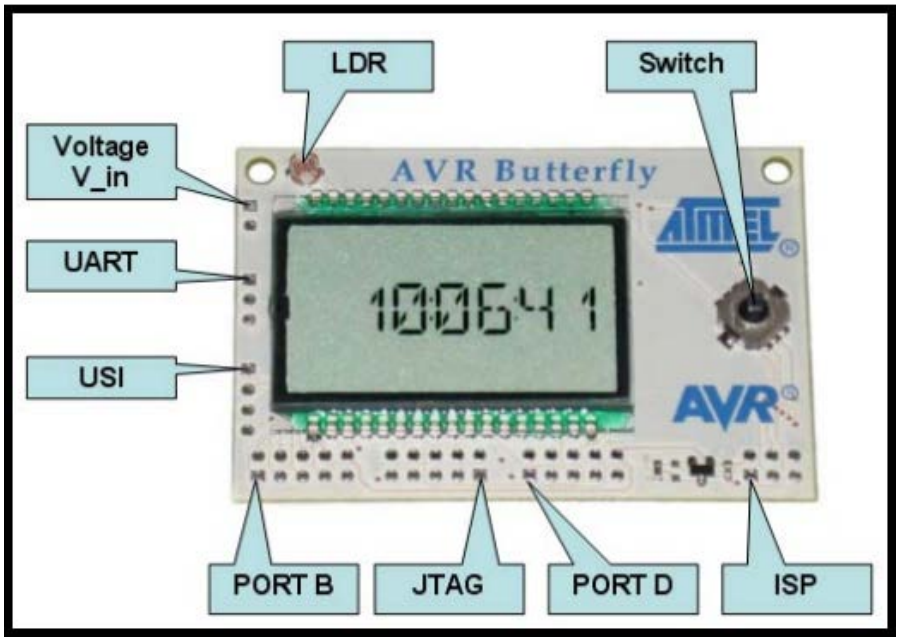


Figura 2.1.3.1: Hardware AVR Butterfly parte frontal. [2]

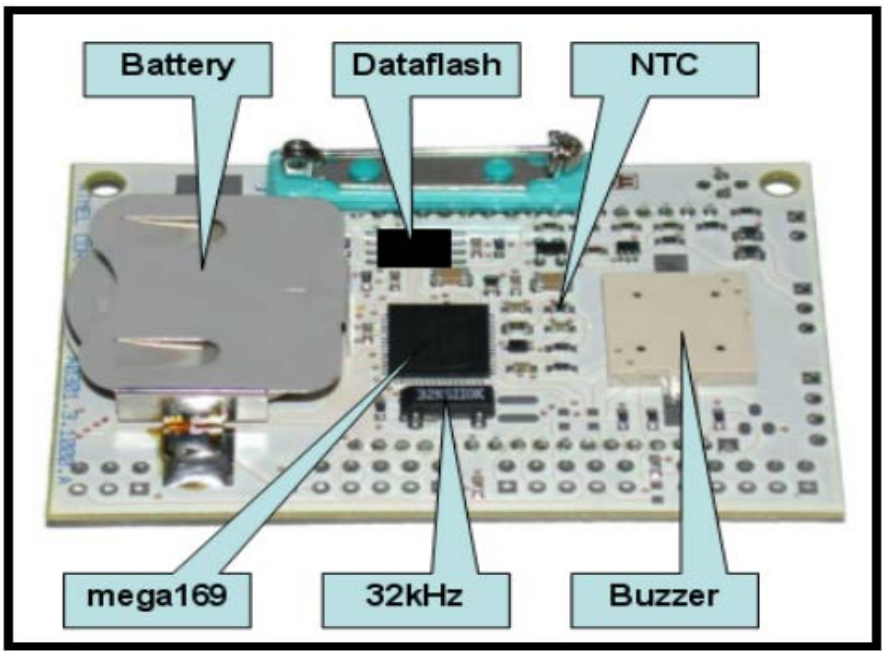


Figura 2.1.3.2: Hardware AVR Butterfly parte posterior [2]

2.1.4 Firmware

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 de 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 Data Flash de 4Mb el usuario podrá almacenar una cantidad grande de datos.

2.1.5 Programación mediante conexión serial (uart) con la pc

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

La comunicación con la PC requiere de tres líneas: TXD, RXD y GND. TXD es la línea para transmitir datos desde la PC hacia el AVR Butterfly, RXD es la línea para recepción de datos enviados desde el AVR Butterfly hacia la PC y GND es la tierra común. En la Tabla 2.1.4 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

Tabla1: Distribución de pines AVR Butterfly Vs. PC. [1]

En la Figura 2.1.5 se observa cómo se debe hacer el cableado para la comunicación, a través de la interfaz serial RS-232, entre el AVR Butterfly y la PC. A la izquierda se aprecia un conector DB9 hembra soldado a los cables que se conectan a la interfaz USART del AVR Butterfly (derecha).

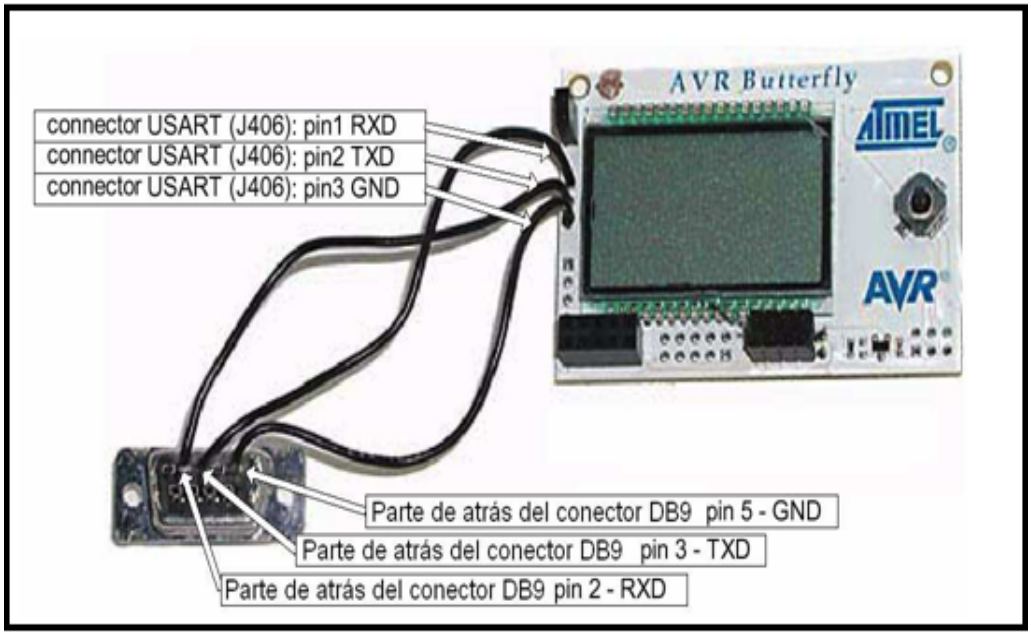


Figura 2.1.5: Conexiones para interfaz USART del AVR Butterfly. [3]

2.2 TARJETA LPC1769

2.2.1 Descripción

El LPC1769 es un ARM Cortex-M3 basado en microcontroladores para aplicaciones integradas que ofrecen un alto nivel de integración y bajo consumo de energía.

El procesador ARM Cortex-M3 es un núcleo de próxima generación que ofrece mejoras en el sistema como funciones avanzadas de depuración y un mayor nivel de apoyo a la integración del bloque.

El LPC1769 funciona en las frecuencias de la CPU de hasta 120MHz. El procesador ARM Cortex-M3 de la CPU incorpora una tubería de 3 etapas y utiliza una arquitectura de Harvard, con locales separados instrucción y los buses de datos, así como un bus tercero para periféricos.

El procesador ARM Cortex-M3 CPU también incluye una unidad de captación previa interna que soporta especulativa ramificación.

El complemento periférico de la LPC1769/68 incluye hasta 512 KB de de memoria flash, hasta 64 KB de memoria de datos, Ethernet MAC, dispositivo USB/Host/OTG interfaz, 8 canales de propósitos generales, controlador DMA, 4 UART, 2 canales de CAN, 2 SSP controladores, interfaces SPI, 3 interfaces de bus I2C, 2 interfaces de buses I2S entrada-salida, 8 canales de ADC de 12 bits, 10 bits DAC, el motor de control PWM, la interfaz de codificador de cuadratura, cuatro temporizadores de uso general, de 6 de salida de propósito general PWM, de ultra bajo de energía en tiempo real Reloj (RTC) con suministro de la batería por separado, y hasta el 70 de propósito general pines I/O.

Los LPC1769 son pin a pin compatible con el LPC236x de 100 pines basado en la serie ARM7.

2.2.2 Características y Beneficios

- ARM Cortex-M3, que funciona a frecuencias de hasta 100 MHz. (LPC1768/67/66/65/64/63) o de hasta 120 MHz (LPC1769). Una unidad de protección de memoria (MPU) a que soporta hasta 8 regiones.
- Permite hasta 512 kB de programación en la memoria flash. Con una velocidad de operación hasta 120 MHz.
- Poseen ocho canales DMA de propósitos generales (GPDMA). Esta puede ser usada con SSP, I2S, UART, convertidores Analógico-Digital y Digital-Analógico.
- Interconexión de múltiples capas de matriz AHB ofrece un autobús para cada maestro de AHB. Los maestros AHB incluyen una CPU, un controlador DMA de propósitos generales, MAC Ethernet y una interfaz USB. Esta interconexión permite una comunicación sin retrasos.
- Interfaces Seriales
  - Mac Ethernet con interface RMII y controlador DMA.
  - USB 2.0 de alta velocidad del dispositivo / Host / OTG controlador con el controlador DMA.
  - Cuatro UARTS con generador de fracción baudrate, FIFO interno y soporte para DMA.
  - Comunicación SPI sincrónico, serial, full duplex y longitud de datos programable.

- Dos controladores con capacidades FIFO y multiprotocolo. Los interfaces SSP pueden ser usados con el controlador GPDMA.
- Tres interfaces mejoradas del bus I2C con salida open-drain.
- Interface I2S para entrada y salida de audio, con control de frecuencia. El bus I2S soporta 3 y 4 hilos de transmisión de datos además de un master clock input/output.

➢ Otros periféricos:

- 70 (100 paquete de pines) de uso general de I/O (GPIO) con pines configurable
  para resistencias pull-up/down.
- Convertidor Analógico-Digital (ADC) con ocho pines de multiplicación, las tasas de conversión de hasta 200Khz. El ADC de 12 bits puede ser usado con GPDMA.
- Convertidor Digital-Analógico de 10 bits, se usa para la conversión en tiempo y soporta DMA.
- Cuatro timers/contadores con ocho entradas de capturas y diez salidas comparadoras.
- Un PWM para control de motores trifásicos.
- Interfaz de codificador de cuadratura que puede controlar un codificador de cuadratura externo.
- Un bloque de estándar PWM/temporizador con entrada externa.

- Watch Dog Timer (WDT). El WDT puede ser ajustado con oscilador RC, RTC o el reloj APB.
- Sistema Timertick ARM Cortex-M3, incluye la opción de un reloj externo.
- Estándar JTAG test/debug de interfaz para la compatibilidad con las herramientas existentes.
- Cuatro modos de ahorro de energía: Sleep, Deep-sleep, Power-down, and Deeppower-down.
- Fuente de alimentación de 3,3 V (2,4 V a 3,6 V).
- Cuatro entradas de interrupción configurable como edge/level. Todos los pines en los Port 0 y Port 2 pueden ser usados como fuente de interrupción.
- Función de reloj, este puede reflejar el reloj oscilador principal, reloj IRC, reloj RTC, reloj del CPU y reloj del USB.
- Power-On Reset (POR).
- Cristal Oscilador con rango de operación de 1Mhz a 25Mhz.
- Oscilador interno RC de 4MHz recortado a 1% de precisión, que opcionalmente se puede utilizar como un reloj del sistema.
- PLL permite el funcionamiento de la CPU a la tasa máxima de la CPU sin la necesidad de un cristal de alta frecuencia. Puede ser ejecutado desde el oscilador principal, el oscilador RC interno, o el oscilador de RTC.

- USB PLL para añadir flexibilidad.
- Código de protección de lectura (PCR) con diferentes niveles de seguridad.
- Número de dispositivo de serie único para fines de identificación.
- Disponible as LQFP100 (14 mm x 14 mm x1.4 mm) and TFBGA1001 (9 mm x 9 mm x 0.7 mm) como paquete.

2.2.3 Aplicaciones

- EMetering.
- Alumbrado.
- Redes Industriales.
- Sistemas de Alarma.
- Productos de línea blanca.
- Control de motores.

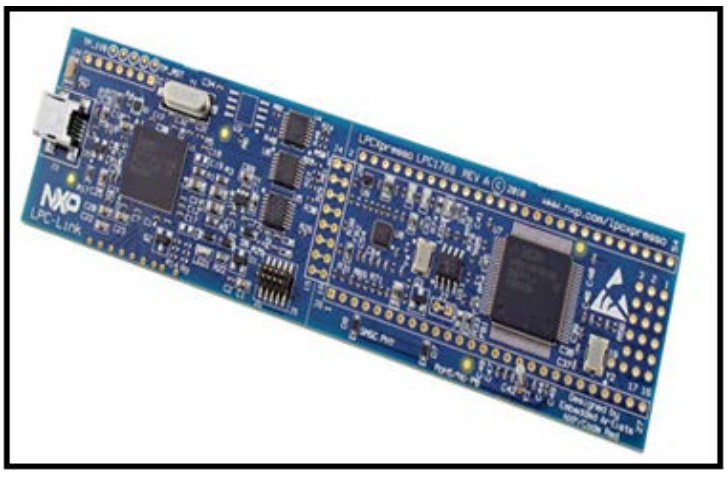


Figura 2.2.3: Foto física de la tarjeta LPC1769. [4]

2.3 MÓDULOS DE RADIO FRECUENCIA (HMTR 433)

Estos módulos de radiofrecuencia están diseñados para aplicaciones que necesiten transmisión de datos a altas velocidades, de larga distancia , frecuencia programables.

El HTMR es un módulo transparente de datos inalámbricos de enlace que se desarrolla por la microelectrónica, dedicada a las aplicaciones que necesita la transmisión de datos inalámbrica. Cuenta con alta velocidad de datos, el protocolo de comunicación es auto controlado y completamente transparente para la interfaz de usuario.

El módulo puede ser incorporado a su diseño actual, de modo que la comunicación inalámbrica se puede configurar fácilmente.

Los módulos HMTR-433 trabajan como transmisor y receptor, su función es la modulación y demodulación de datos, existen en lógica TTL y RS232

Entre las características del transceiver HMTH-433 podemos mencionar:

- Modulación FSK, alta inmunidad a interferencias.
- Bandas ISM 315/434/868/915 MHz.
- Conversión de RF a UART auto controlado, confiable y fácil de usar.
- Formato UART configurable, con data rate de 300 – 19200 bps.
- Niveles TTL y RS232.

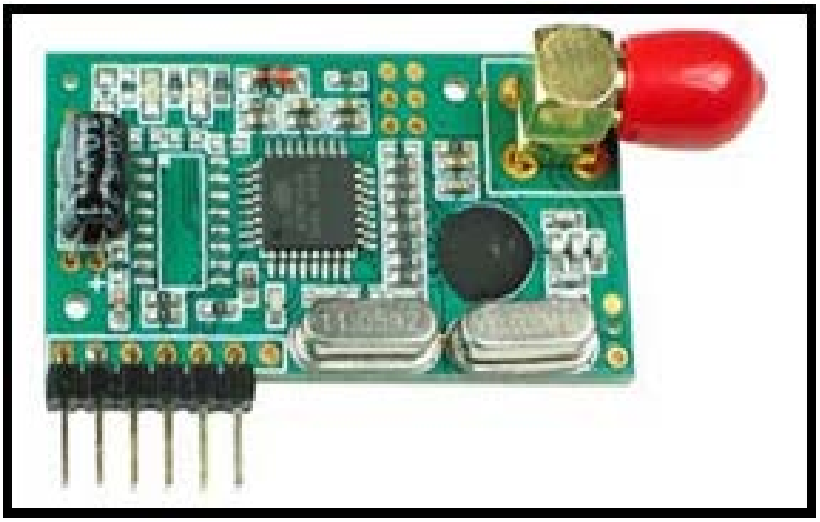


Figura 2.3: Módulo HM-TR 433. [5]

2.4 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.

Aun 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 Studio4 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).

  -MemoriaRAMde64Mb.

  -DiscoDurocon50Mbdeespaciodisponible.

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 microcontroladoresAVRde8bits.

AVR Studio4 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.

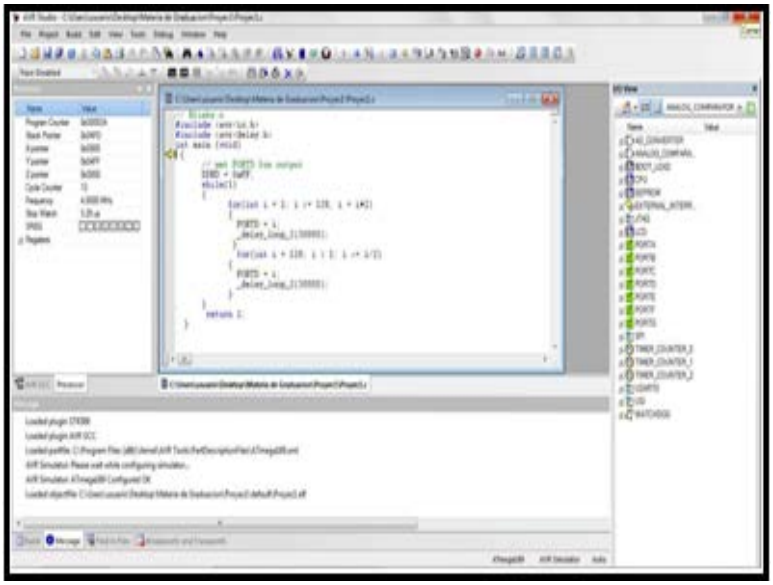


Figura2.4: Ventana Principal IDE. [3]

2.5 WIN AVR

La distribución Win Avr, que es una recopilación de programas de software libre diseñados para facilitar las tareas de programación y desarrollo de los microcontroladores Avr. Dicha distribución Win Avr 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 Win AVR 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. Win AVR 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.
- Avr dude, la interfaz para programación.
- Avarice, la interfaz para JTAG ICE.
- Avr-gdb, el depurador.
- Programmers Notepad, el editor.
- MFile, generador de archivo make file.

La distribución Win Avr 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.

2.6 LPCXpresso

LPCXpresso, Figura 2.6, una plataforma de bajo desarrollo de NXP. El software consiste en un aumento, IDE basado en Eclipse, un compilador de C de GNU, linkers, librerías, y un debugger GDB mejorado. El hardware consiste en la placa de desarrollo LPCXpresso que tiene una interfaz de depuración LPC-Link y un NXP LPC basado en ARM ARM-based microcontroller target. LPCXpresso es una solución de extremo a extremo, permitiendo a los ingenieros desarrollar sus aplicaciones desde al principio hasta la producción final.

El IDE LPCXpresso, impulsado por Code Red Technologies, se basa en la plataforma de desarrollo Eclipse popular e incluye varias mejoras específicas de LPC. Se trata de un estándar de la industria con un conjunto de herramientas GNU optimizado biblioteca de C que permite a los ingenieros desarrollar soluciones de alta calidad de software de forma rápida y rentable. El entorno de programación C incluye características de nivel profesional. La tarjeta del LPCXpresso, desarrollada conjuntamente por NXP y Code Red Technologies, incluye un depurador JTAG integrado (LPC-Link), así que no hay necesidad de una JTAG debug probe. A la tarjeta también se le puede

conectar tarjetas de expansión para ofrecer una mayor variedad de interfaces y dispositivos I/O. La LPC-link depurador proporciona un USB de alta velocidad para JTAG/SWD con interfaz para el IDE y se puede conectar a otros objetivos de depuración, como un prototipo del cliente. Los usuarios también pueden usar el IDE LPCXpresso con el adaptador Red Probe JTAG de Code Red Technologies.

Los productos compatibles con la plataforma de LPCXpresso son:

LPC1100: Soporta todas las clases.

OM11049: LPC1114/302

OM13014: LPC11U14

OM13012: LPC11C24

LPC1200: Soporta todas las clases.

OM13008: LPC1227

LPC1300: Soporta todas las clases.

OM11048: LPC1343

LPC1700: Soporta todas las clases.

OM13000: LPC1769

LPC1800: Soporta todas las clases.

LPC2000: LPC2109, LPC2109/01, LPC2134, LPC2142, LPC2362, LPC2929

LPC3000: LPC3130, LPC3250

LPC4000: Soporta todas las clases.

LPCXpresso base board products:

OM11083: Embedded Artists Base Board for LPCXpresso and mbed

OM13009: Embedded Artists Motor Control Kit for LPCXpresso

OM13016: NGX mbed - LPCXpresso base board.

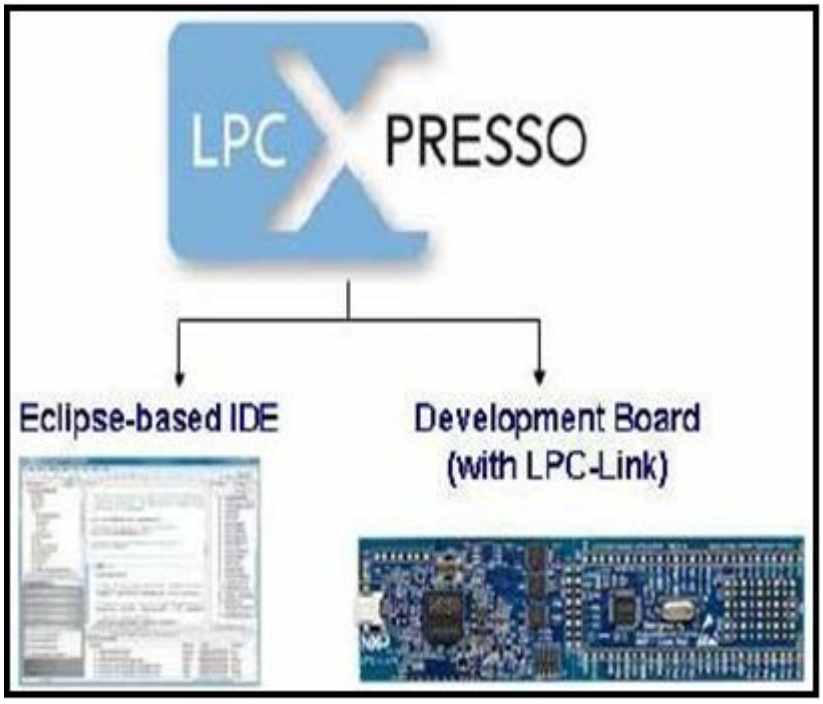


Figura 2.6: Componentes LPCXpresso. [6]

2.6.1 LPCXpresso IDE

IDE LPCXpresso es un entorno para desarrollar software altamente integrado para LPC de NXP Microcontroladores, el mismo que las herramientas necesarias para poder desarrollar soluciones de alta calidad en

software de una manera efectiva en tiempo y costo. LPCXpresso está basado en Eclipse con muchas mejoras específicas para LPC. También cuenta con la última versión de herramientas estándar de industria GNU que posee una biblioteca propia en C optimizado que proporciona herramientas profesionales de alta calidad a bajo costo. El IDE LPCXpresso puede construir un ejecutable de cualquier tamaño con la optimización del código completo y es compatible con un límite de descarga de 128 kB después del registro. LPCXpresso es un equipo completo de diseño de productos ya que nos permite programar las tarjetas LPC mediante el software necesario y así mismo estas tarjetas permiten ir más allá de las placas y apoyar el desarrollo de chips con componentes externos.

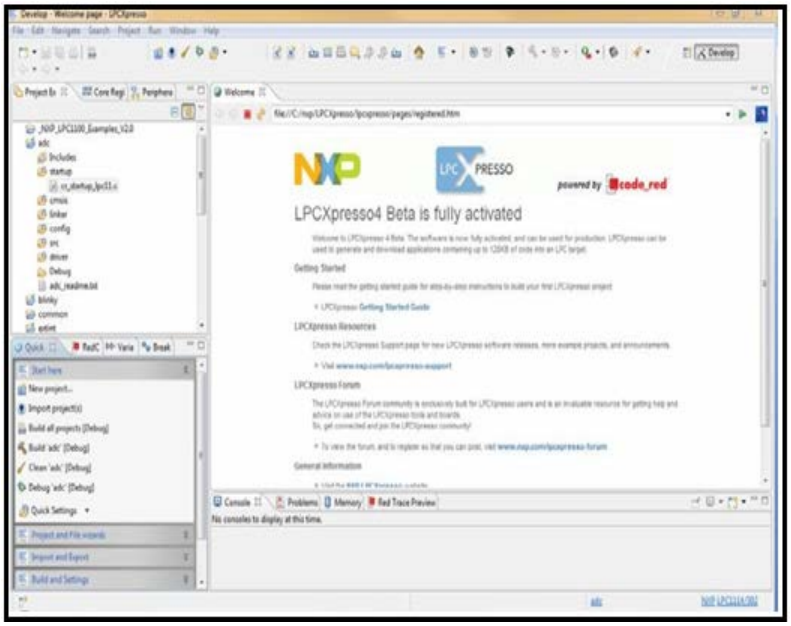


Figura 2.6.1: Ventana del programa LPCXpresso. [6]

2.6.2 LPCXpresso Development Board

En la figura 11, se observa a una tarjeta LPCxpresso y como está compuesto por: LPC – link y un Traget.

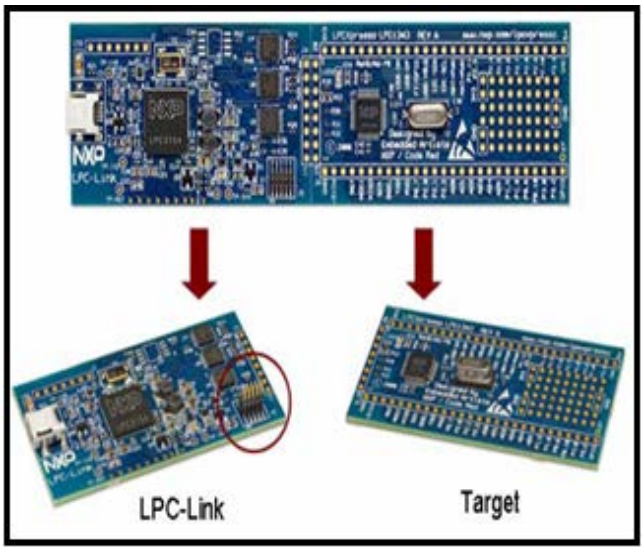


Figura 2.6.2: Tarjeta LPC. [6]

2.6.2.1 Lpc-Link Jtag/swd debugger

La tarjeta LPCXpresso contiene un debugger JTAG/SWD denominada "LPC-Link" y un MCU de destino. LPC-Link está equipado con una cabecera de 10 pines JTAG, figura 11, y se conecta sin problema a través de un puerto USB. En la tarjeta el LPC-link y el target se encuentran divididos. Esto permite a la plataforma LPCXpresso ser conectado a un medio exterior y así desarrollar aún más la tarjeta puede ser conectada con una amplia variedad de aplicaciones basadas en otros NXP's como Cortex-M0, Cortex-M3, y ARM7/9.

2.6.2.2 Integrated evaluation target

El target incluye pequeñas aéreas para la expansión del mismo

La tarjeta LPCXpresso con target puede ser usada para:

- Para desarrollo de software y autoevaluación.
- Permite conexión externa de dispositivos a la placa para realizar pruebas rápidas.
- Permite conexión del usuario a la tarjeta para un completo diseño.

2.7 MOTOR BRUSHLESS (SIN ESCOBILLAS)

BLDC son las siglas para motor brushless DC, se traduce como motores DC sin escobillas, estos motores se emplean en sectores industriales tales como: Consumo Médico, Equipos de Automatización e Instrumentación, Automóvil, Aeroespacial.

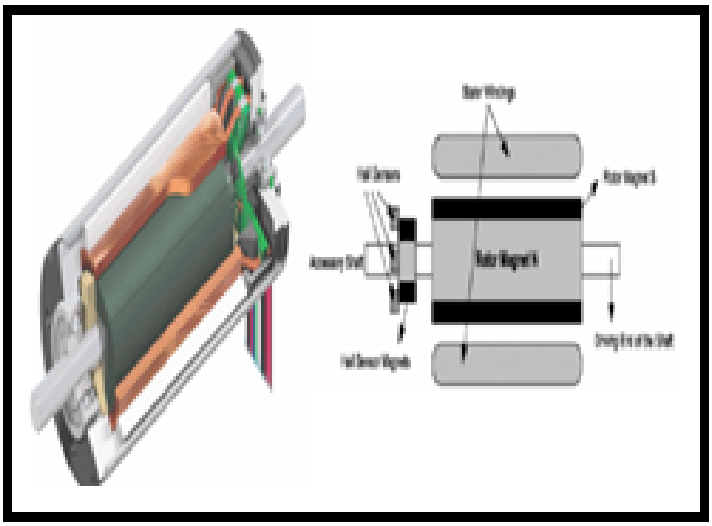


Figura 2.7: Estructura de un Motor. [7]

En la Figura 2.7 mostrada a continuación se puede observar la estructura de un motor BLDC, este tipo de motor puede realizar la misma función que la de un motor DC y se remplaza el conmutador de los motores y de las escobillas por switches. En un motor las escobillas son elementos que hacen contacto la parte fija (estator) con la parte rotatoria (rotor) de un motor.

La gran ventaja de los motores sin escobillas es que no requieren de mantenimiento constante, a diferencia de los motores que cuentan con conmutador y escobillas, las cuales se desgastan, y por esta razón necesitan de un frecuente mantenimiento.

2.7.1 Características de los motores brushless

Como se mencionó los motores BLDC tienen la característica de no emplear escobillas en la conmutación para la transferencia de energía; en este caso, la conmutación se realiza electrónicamente. Las escobillas producen rozamientos, disminuyen el rendimiento, desprenden calor, son ruidosos y requieren una sustitución periódica. Todos estos problemas se eliminan con el motor BLDC, por ende los motores BLDC tienen muchas ventajas frente a los motores DC con escobillas y frente a los motores de inducción. Algunas de estas ventajas son:

- Mejor relación velocidad-par motor.
- Mayor respuesta dinámica.

- Mayor eficiencia.
- Mayor rango de velocidad.
- Mayor vida útil.
- Menor ruido.

La relación par motor-tamaño es mucho mayor, lo que permite emplear en los lugares donde se trabaje con un espacio reducido. En contraparte, las desventajas de los motores BLDC son las siguientes:

- Tienen un mayor coste.
- Requieren un control bastante más complejo.

2.7.2 Construcción y operación

La construcción de los Motores Brushless DC es muy similar a un motor AC.

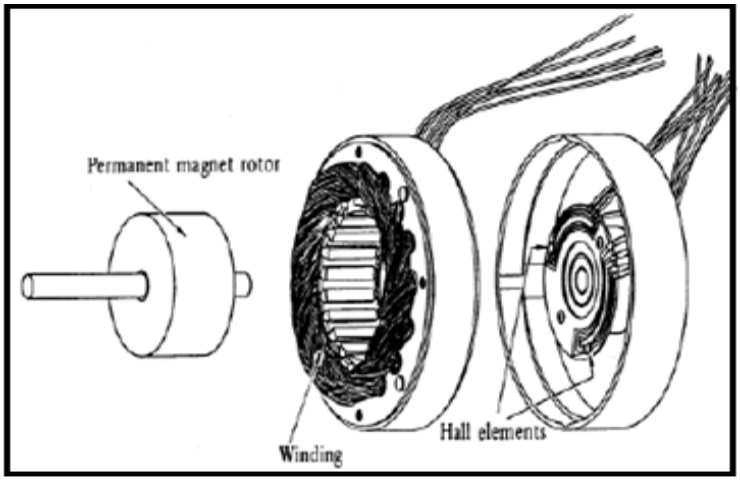


Figura 2.7.2: Despiece del Motor Brushless DC. [8].

La Figura2 muestra el despiece de un motor BLDC, en donde el rotor es un la parte que se encuentra magnetizada permanentemente; el estator está formado por embobinados de varias fases similar a un motor AC. Estos motores se diferencian en la forma de detectar la posición del rotor, para conocer cómo se encuentran los polos magnéticos y así poder generar una señal de control mediante el uso de switches electrónicos.

2.7.3 Funcionamiento

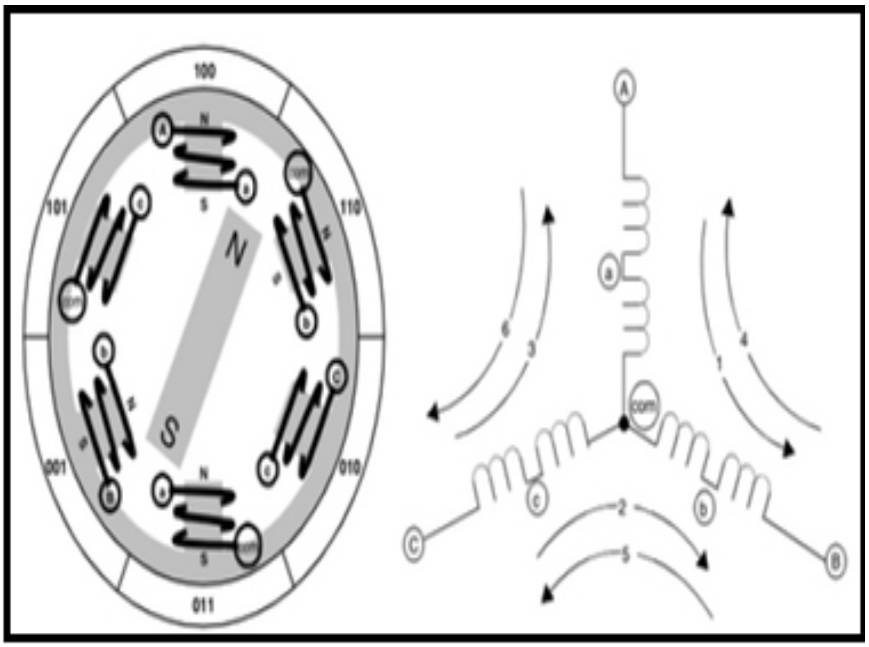


Figura 2.7.3: Bobinado del motor BLDC en conexión estrella. [9].

Dentro de los motores se encuentran divididos en tres fases conectados en estrella con un punto en común, estas fases son excitadas por pulsos sinusoidales, Figura3, en la vida real son pulso rectangulares, estos golpes

junto a la posición del rotor son los que permiten al motor BLDC ser impulsado. Cada circuito electromagnético tiene su referencia en el centro, permitiendo así que el imán permanente del rotor a moverse en el medio del campo magnético inducido. La posición del rotor puede es determinada por códigos únicos.

Así mismo variando el voltaje a través del motor se puede controlar la velocidad del rotor. Esto puede lograrse por ancho de pulso (PWM) de la tensión de fase. Esto puede lograrse por ancho de pulso (PWM) de la tensión de fase. Al aumentar o disminuir el ciclo de trabajo, la corriente fluirá a través de las bobinas del estator. Esto afecta la densidad de flujo del estator, que cambia la fuerza entre el rotor y el estator. Esto significa que la velocidad de rotación está determinada por la carga del rotor, la corriente en cada fase, y el voltaje aplicado.

2.7.4 Tipos de motores BLDC

Una primera clasificación de estos motores se realiza en base a la forma de la onda de tensión inducida y los divide en dos grupos. En el primero se encuentran aquellos cuya onda de tensión inducida es senoidal, también llamados "Motores Síncronos de Imanes Permanentes". Son motores de altas prestaciones y se emplean sobre todo en servosistemas. En un segundo grupo se incluyen los de onda trapezoidal, conocidos como “Motores de Corriente Continua sin Escobillas, o "BRUSHLESS DC”. Suelen

ser motores de pequeña potencia y de prestaciones dinámicas no muy exigentes.

En el motor trapezoidal la FEM inducida en las bobinas del estator tiene una forma trapezoidal y las fases deben ser suplidas con corrientes casi cuadradas para una operación libre de rizado del par.

El motor sinusoidal tiene una FEM inducida formada sinusoidalmente y requiere corrientes de fase sinusoidales para que la operación de troque del par esté libre de rizado. La forma de la EM inducida es determinada por la forma de los imanes del rotor y la distribución de las bobinas del estator. El motor sinusoidal necesita un sensor de posición de alta resolución porque la posición del rotor deber ser conocida en todo instante de tiempo para una operación óptima. También requiere software y hardware más complejos. El motor trapezoidal es una alternativa muy atractiva para más aplicaciones debido a que tiene mayor simplicidad, menor precio y alta eficiencia. En el presente documento trataremos solamente el motor de tipo trapezoidal, más específicamente el de cuatro polos en su rotor.

2.7.5 Técnicas de control

Las técnicas de control para motores brushless se pueden clasificar según el algoritmo de conmutación implementado. Las más utilizadas actualmente son:

- Conmutación trapezoidal
- Conmutación sinusoidal y
- Control vectorial

El objetivo de estas técnicas es estimar la excitación optima de cada una de las fases del y se diferencian una de otra principalmente por su complejidad de implementación que se traduce en un incremento de presentaciones.

El control basado en conmutación trapezoidal es uno de los métodos más simples. En este esquema se controla la corriente que circula por los terminales del motor, excitando un par simultáneamente y manteniendo el tercer terminal desconectado. Sucesivamente se va alternando el par de terminales a excitar hasta completar las seis combinaciones posibles.

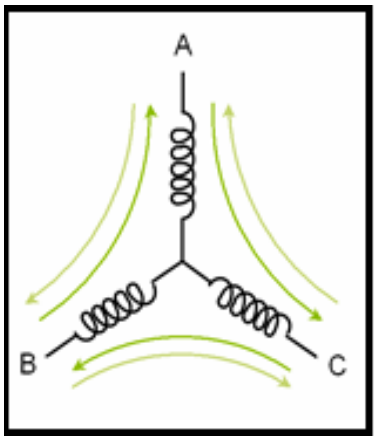


Figura 2.7.5.1: Caminos de circulación de corriente. [10]

Tres sensores de efecto hall situados en el motor son utilizados para proporcionar a posición aproximada del rotor al controlador y que este pueda determinar el próximo par de terminales a excitar. La siguiente figura

muestra el diagrama de bloques de un controlador trapezoidal típico con lazo cerrado de corriente.

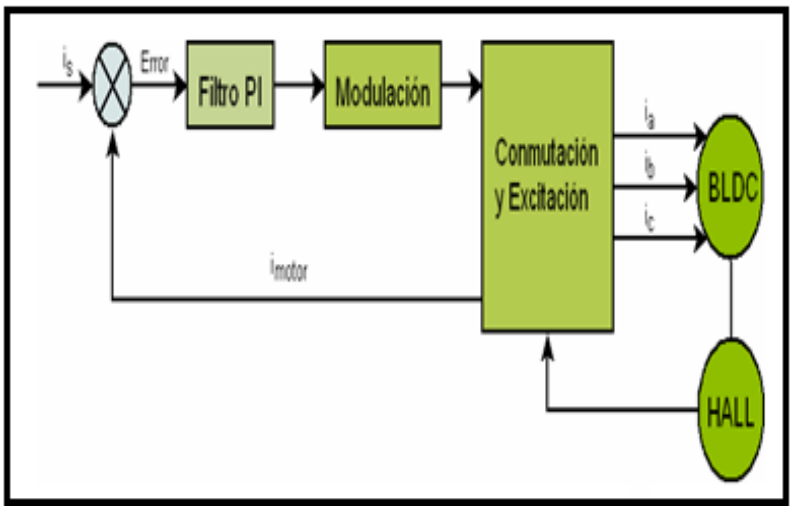


Figura 2.7.5.2: Control con conmutación trapezoidal. [10]

La corriente que circula por el par de terminales activos es comparada con la corriente deseada y el error resultante es aplicado a un Filtro PI (Proporcional - Integrador). La salida de este filtro intenta corregir la desviación y por tanto minimizar el error. Con esta técnica se consigue mantener constante la corriente que circula por cualquiera de los bobinados del motor.

2.7.6 Control basado en conmutación sinusoidal

La conmutación sinusoidal es vista como un control más avanzado y exacto que el trapezoidal, ya que intenta controlar la posición del rotor continuamente.

Esta continuidad se consigue aplicando simultáneamente tres corrientes sinusoidales desfasadas 120º a los tres bobinados del motor. La fase de estas corrientes se escoge de forma que el vector de corrientes resultante siempre esté en cuadratura con la orientación del rotor y tenga un valor constante.

Como consecuencia de este procedimiento se obtiene un par más preciso y sin el rizado típico de la conmutación trapezoidal.

No obstante, para poder generar dicha modulación sinusoidal es necesaria una medida precisa de la posición del rotor.

Debido a que los sensores de efecto hall solo proporcionan una posición aproximada es necesario el uso de otro dispositivo que aporte mayor precisión angular como puede ser un encoder.

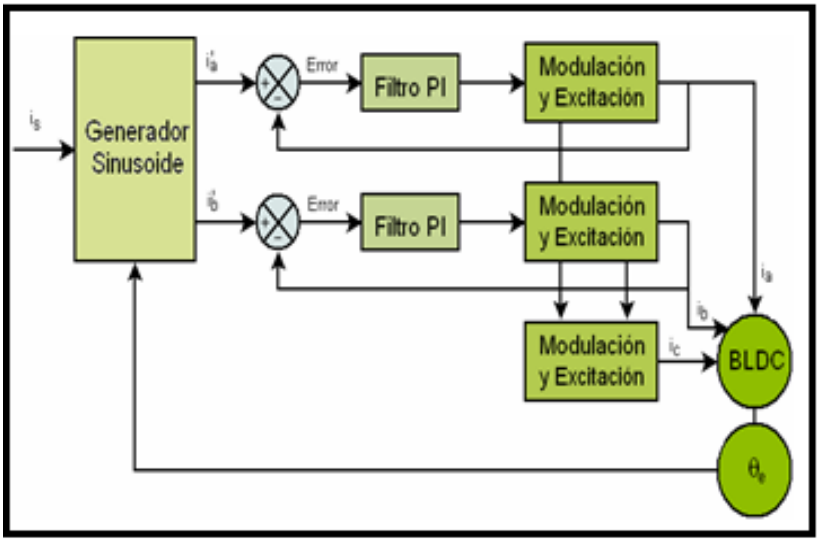


Figura 2.7.6: Controlador con conmutación sinusoidal. [10]

2.7.7 Control vectorial

El control vectorial es el más complejo y el que requiere mayor potencia de cálculo de las tres técnicas. A su vez también es la que mejor control proporciona.

El problema principal que presenta la conmutación sinusoidal es que intenta controlar directamente las corrientes que circulan por el motor, las cuales son intrínsecamente variantes en el tiempo.

Al aumentar la velocidad del motor, y por tanto la frecuencia de las corrientes, empiezan a aparecer problemas.

El control vectorial o Field Oriented Control (FOC) soluciona el problema controlando el vector de corrientes directamente en un espacio de referencia ortogonal y rotacional, llamado espacio D-Q (Direct-Quadrature).

Dicho espacio de referencia está normalmente alineado con el rotor de forma que permite que el control del flujo y del par del motor se realice de forma independiente.

La componente directa permite controlar el flujo y la componente en cuadratura del par.

Debido a que el vector de corrientes en el espacio de referencia D-Q es estático los filtros PI trabajan en continua y se eliminan por tanto los problemas frecuenciales de la conmutación sinusoidal (verFiguraII-7).

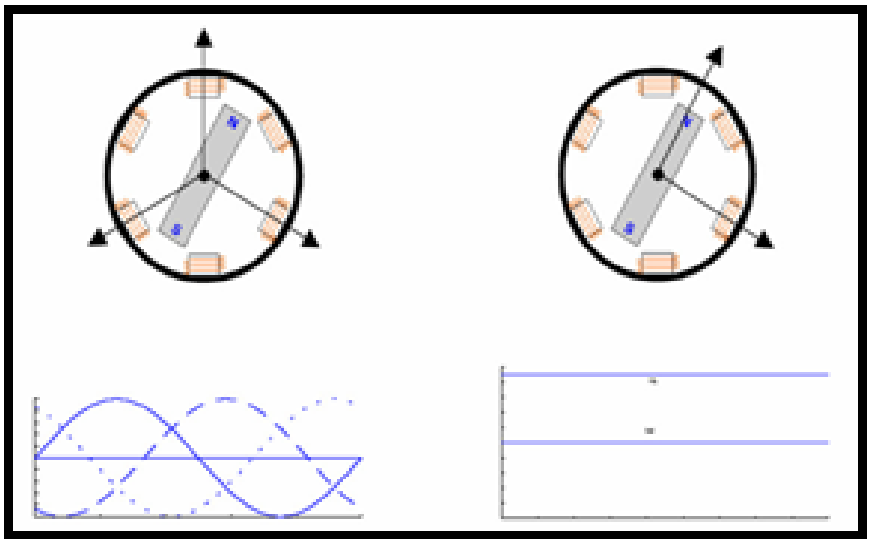


Figura 2.7.7.1: Comparativa entre el espacio estático de las bobinas y el espacio rotacional D-Q. [10]

Una vez aplicadas las dos transformaciones el control del motor se simplifica considerablemente. Dos Filtros PI son utilizados para controlar la componente directa y la cuadratura de forma independiente.

La componente en cuadratura es la única que proporciona par útil, por tanto, la referencia de la componente directa suele fijarse a cero. De esta forma se fuerza al vector de corrientes a situarse en la dirección de la componente de cuadratura maximizando la eficiencia del sistema.

Posteriormente se realizan las transformadas inversas para regresar al espacio estacionario de las bobinas y se aplica la excitación correspondiente a cada una de las fases mediante modulación.

El diagrama de bloques del control vectorial es el siguiente:

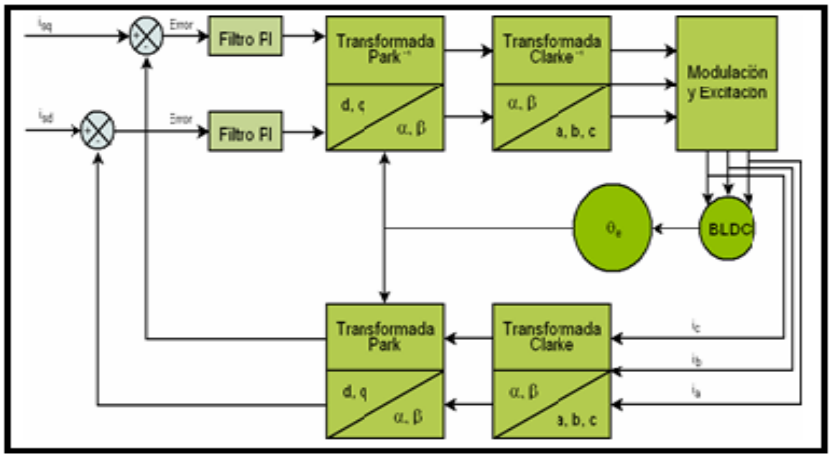


Figura 2.7.7.2: Diagrama de bloques del control vectorial. [10]

2.7.8 Comparativa de motores con escobilla y sin escobilla

A continuación se presenta en la tabla 1, una comparación de las características principales de los motores.

	Motor DC convencional	Motor DC sin escobillas
Estructura Mecánica	Elementos magnéticos en el estator	Elementos magnéticos en el rotor
Principales Características	Respuesta rápida y excelente controlabilidad	Fácil mantenimiento
Conexión de los embobinados	Conexión Δ	conexión Δ o Y
Método de Conmutación	Contacto mecánico entre las escobillas y el conmutador	Conmutación electrónica por medio de transistores
Método para detectar la posición del rotor	Detectada automáticamente por las escobillas	Sensor de *efecto Hall, encoder* óptico, etc.
Método de reversa	Cambiando la polaridad del voltaje	Cambiando la lógica

Tabla 2: Tabla comparativa entre Motor DC convencional y Motor DC sin escobillas. [8]

CAPÍTULO 3

IMPLEMENTACIÓN DE EJERCICIOS Y PROYECTO

En el siguiente capítulo se presenta un conjunto de ejercicios utilizados para relacionarnos con las respectivas tarjetas utilizadas tanto la AVR Butterfly como la LPC1769; las cuales nos permitieron realizar avances en nuestro proyecto final.

Se realizaron tres ejemplos con el propósito de capacitarnos con el manejo de estas nuevas herramientas entre los que se encuentran:

- Comunicación uart entre dos tarjetas lpc1769.
- Encendido de leds mediante la tarjeta lpc1769.
- Encendido de leds mediante la tarjeta AVR Butterfly.
- Control del motor BLDC a través de comunicación uart entre el AVR Butterfly y la LPC1769 utilizando módulos de radio frecuencia.

Con la finalidad de analizar y comprender el funcionamiento de cada uno de los ejercicios realizamos las respectivas descripciones, diagrama de bloques, diagramas ASM, análisis del algoritmo y códigos fuente.

3.1 COMUNICACIÓN UART ENTRE DOS TARJETAS LPC1769

3.1.1 Descripción

En este ejemplo vamos a ver cómo comunicar dos tarjetas LPC 1769 entre si vía serie RS-232 haciendo uso del módulo UART (Universal Asynchronous Receiver Transmiter.), que incorporan la mayoría de los PIC's de la gama media / alta. Con este módulo hardware se puede implementar una comunicación serie del tipo asíncrona.

Se pretende realizar la comunicación entre LPC's usando la comunicación UART con el objetivo de familiarizarse con las tarjetas conocer los puertos y entender cómo funciona la comunicación UART.

Al presionar la botonera, la LPC de transmisor envía el carácter 'A' y este es recibido por la LPC receptora.

3.1.2 Diagrama de bloques

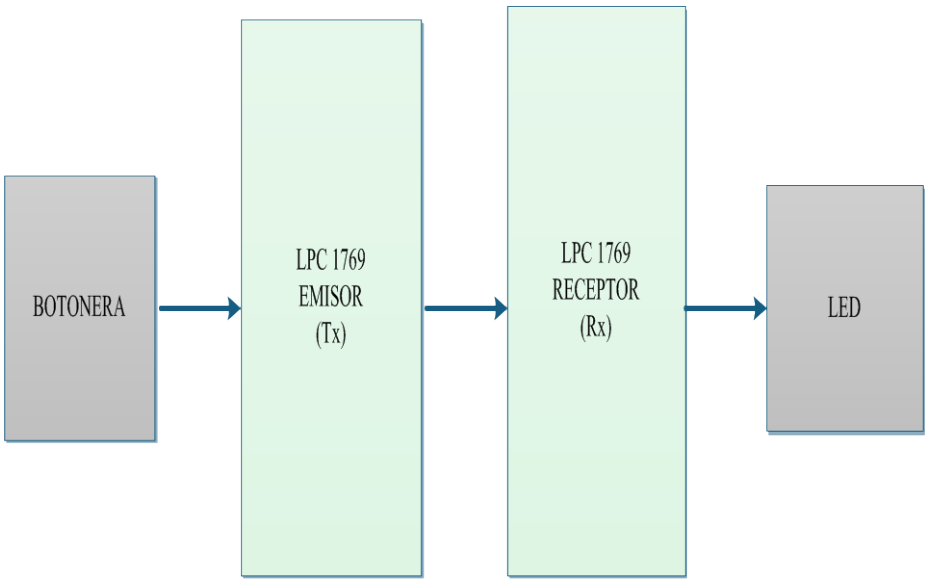


Figura 3.1.2: Diagrama de Bloques

3.1.3 Diagrama ASM

EMISOR

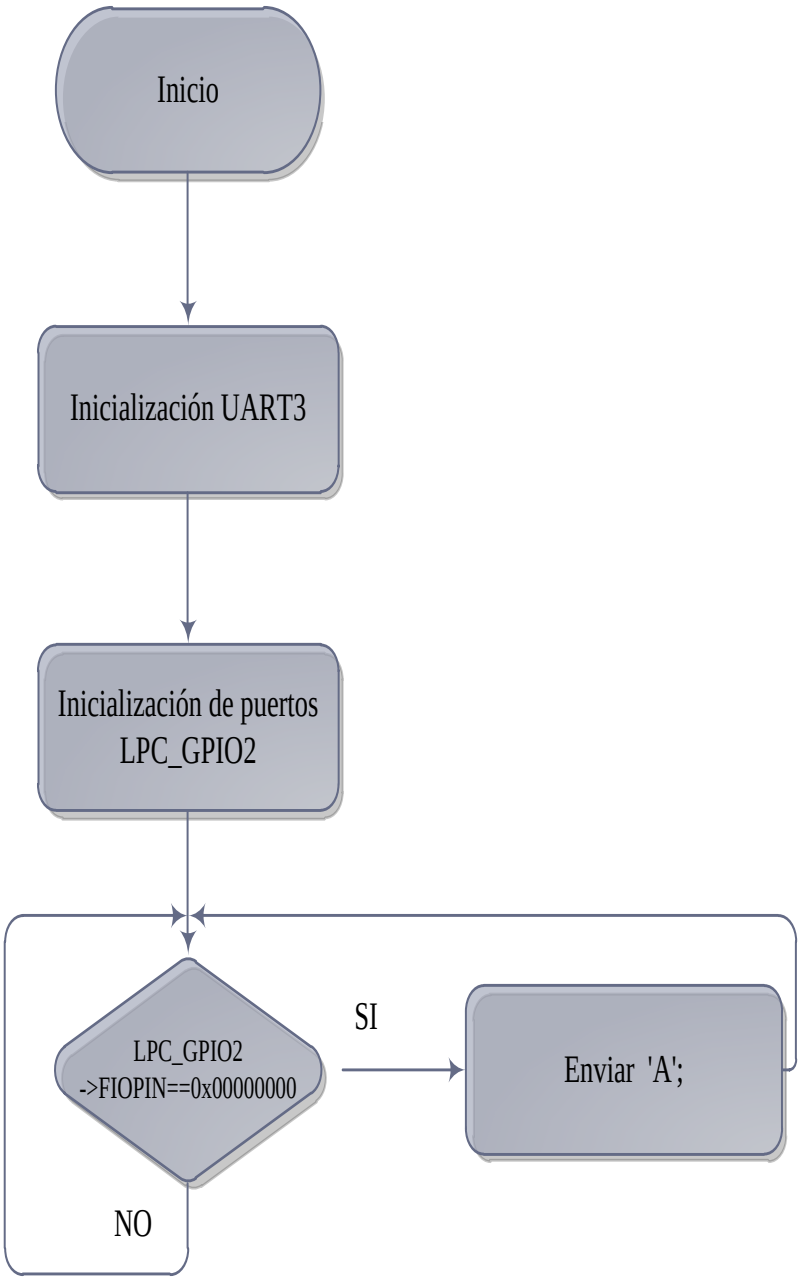


Figura 3.1.3.a: Diagrama ASM Emisor

RECEPTOR

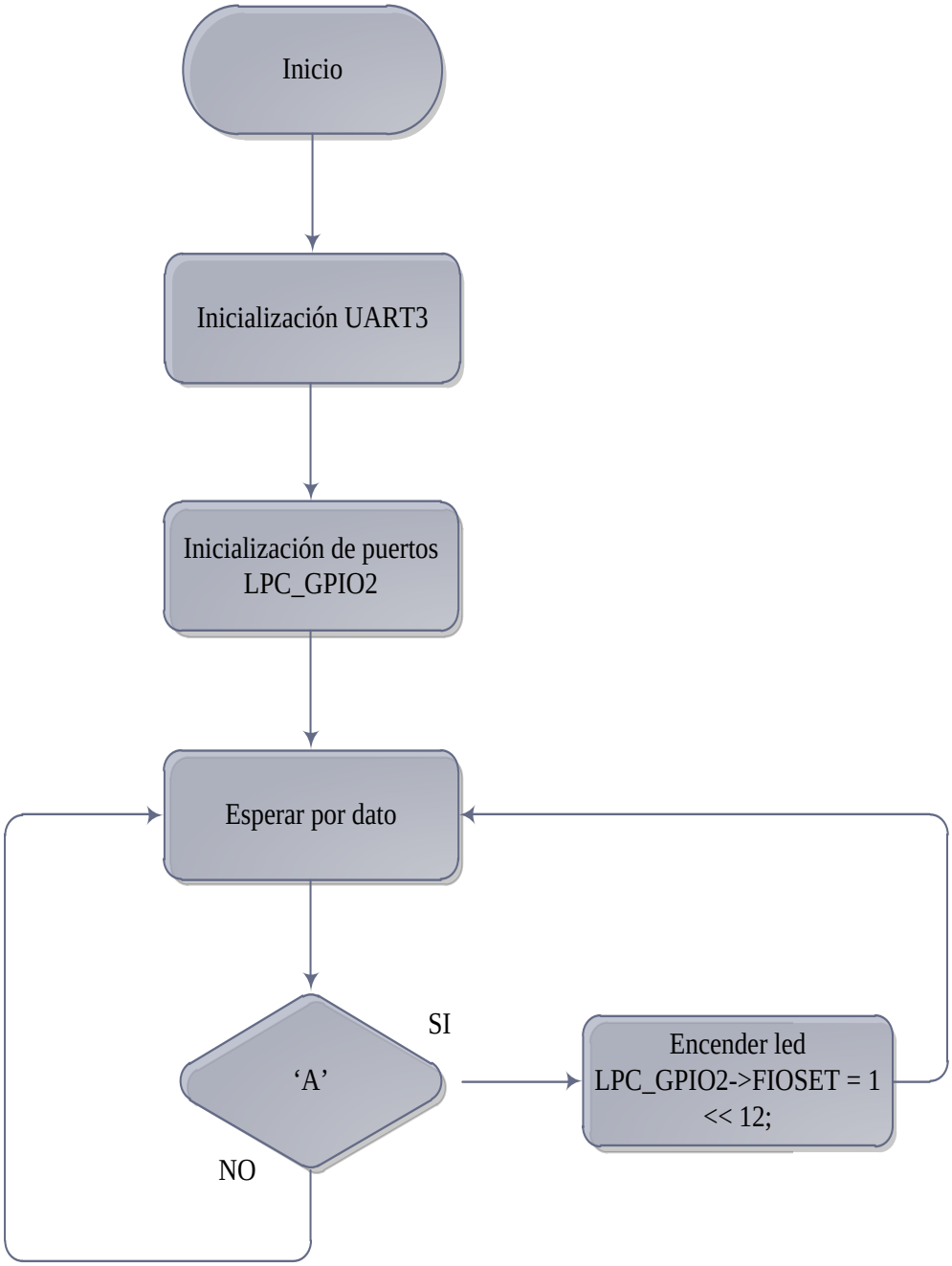


Figura 3.1.3.b: Diagrama ASM Receptor

3.1.4 Descripción del algoritmo

EMISOR

- Se inicializan los puertos a usar.
- Enviar letra ‘A’.

RECEPTOR

- Se inicializan los puertos a usar.
- Cuando recibe la letra ‘A’ habilita con un alto el puerto P2[12].

3.1.5 Código Fuente

EMISOR

```
#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* encendido = 'A';
```

```
UARTInit(3, 9600);        /* baud rate setting */
  LPC_GPIO2->FIODIR = 0xFFFFEFFF;                  /*  P2.xx  defined  as
Outputs */
          LPC_GPIO2->FIOCLR = 0xFFFFFFFF;                  /*  turn  off  all  the
LEDs */

LPC_GPIO2->FIOMASK= 0x00000000;
         while(1)
          {
                if(LPC_GPIO2->FIOPIN==0x00000000 )
                {
                UARTSend(3, (uint8_t *)encendido , strlen(encendido) );
                }
          }
}
```

RECEPTOR

```
#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)
{
        LPC_GPIO2->FIODIR = 0xFFFFFFFF;          /*  P2.xx  defined  as
Outputs */

        LPC_GPIO2->FIOCLR = 0xFFFFFFFF;
         LPC_GPIO2->FIOSET = 0x00000FFF;
        UARTInit(3, 9600);  /* baud rate setting */
        while (1)/* Loop forever */
        {
                if ( UART3Count != 0 )
                {
                        LPC_UART3->IER = IER_THRE | IER_RLS;
                /* Disable RBR */
                        if(*UART3Buffer==0x41)
                        {
                   LPC_GPIO2->FIOSET = 1 << 12;
                        }
                        LPC_GPIO2->FIOCLR = 0x00000FFF;
                        UART3Count = 0;
LPC_UART3->IER = IER_THRE | IER_RLS | IER_RBR; }
                }
}
```

3.2 ENCENDIDO DE LEDS MEDIANTE LA TARJETA LPC1769

3.2.1 Descripción

Mediante este ejercicio se pretende entender y analizar los distingos registros y funciones de la LPCxpresso, así mismo conocer físicamente sus puertos y relacionarlos con las respectivas funciones. La LPC1769 realiza el movimiento de 8 diodos leds conectados a GPIO2, Puerto 2 de Propósito General, estos rotaran en un sentido hasta la mitad y luego cambiaran de dirección, dando la apariencia de que se encuentran y luego se separan. Estas secuencias son controladas mediante una botonera ubicado en el puerto P2[12].

3.2.2 Diagrama de bloques

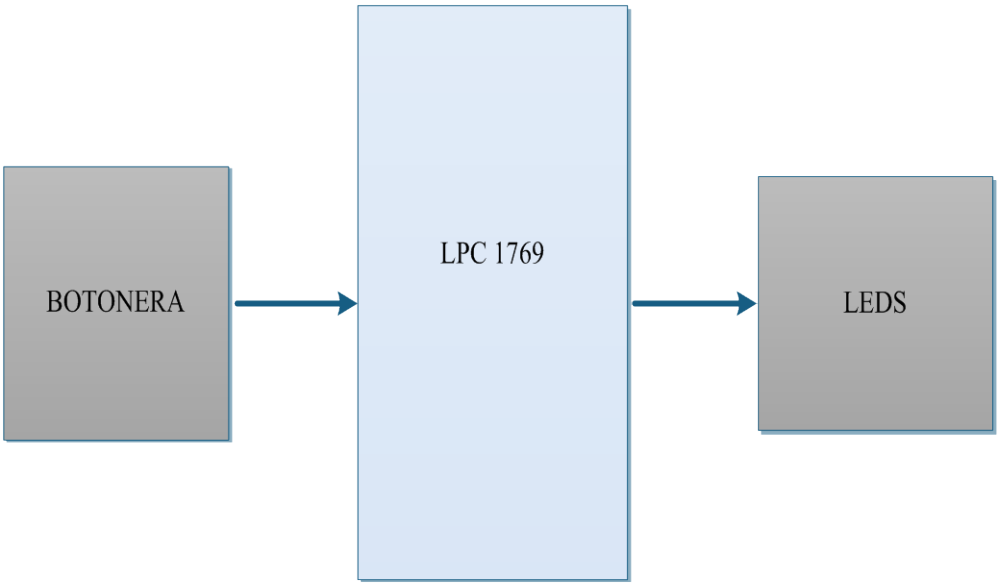


Figura 3.2.2: Diagrama de Bloques

3.2.3 Diagrama ASM

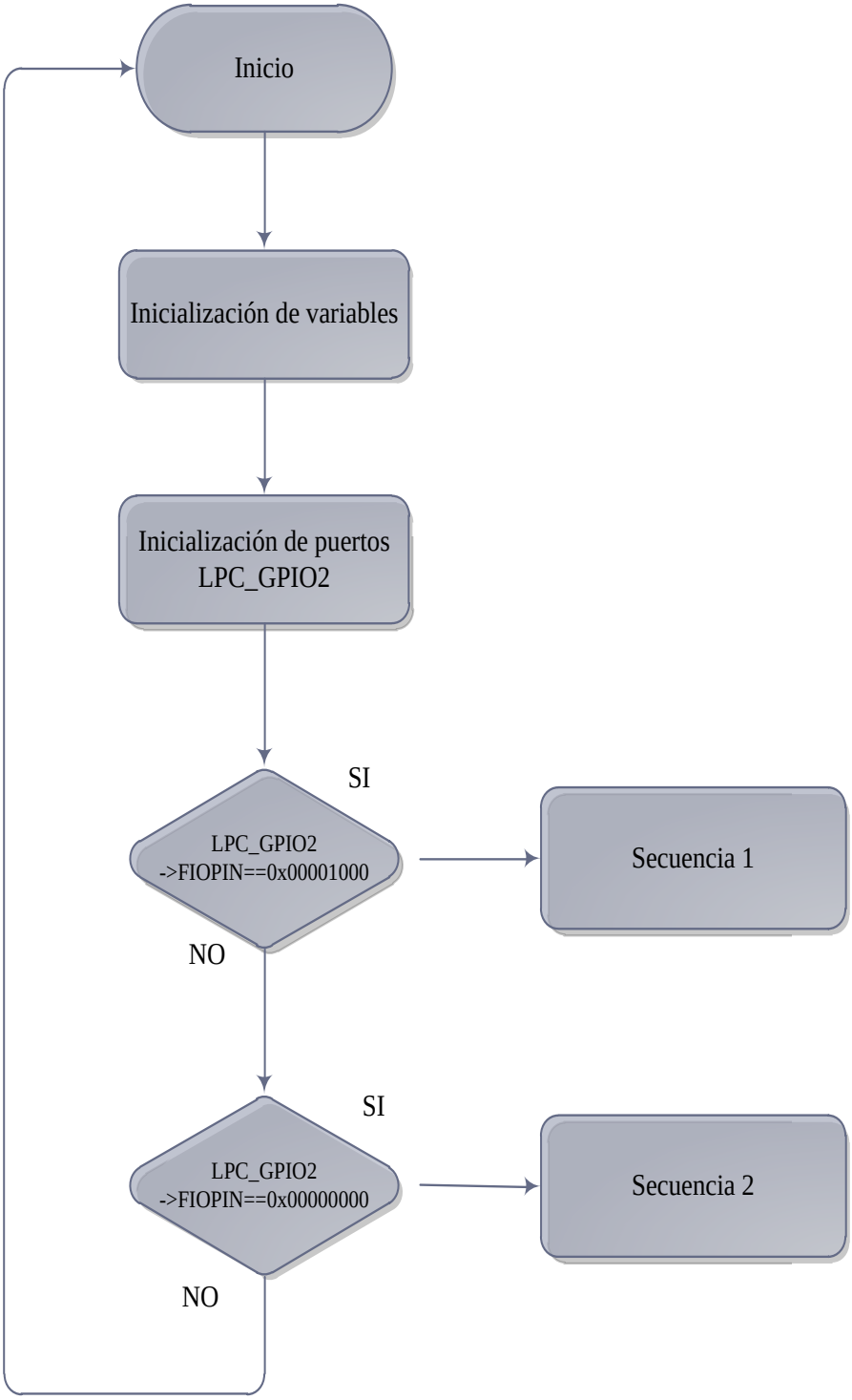


Figura 3.2.3: Diagrama ASM

Subrutina: Secuencia 1

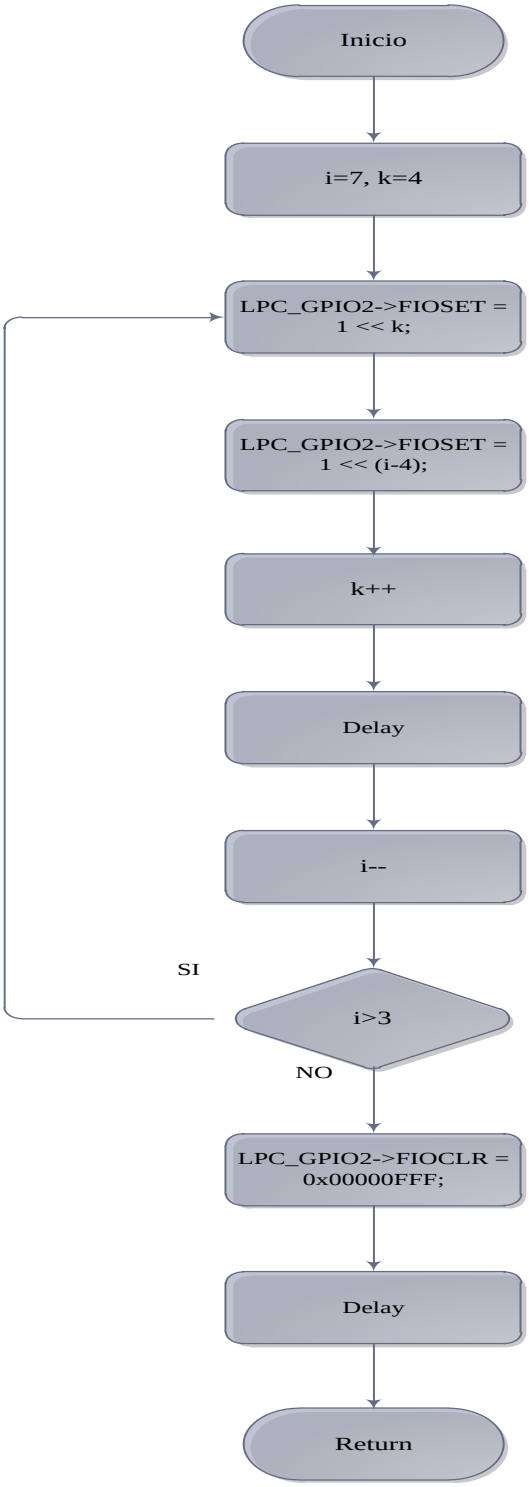


Figura 3.2.3.a: Diagrama ASM secuencia 1

Subrutina: Secuencia 2

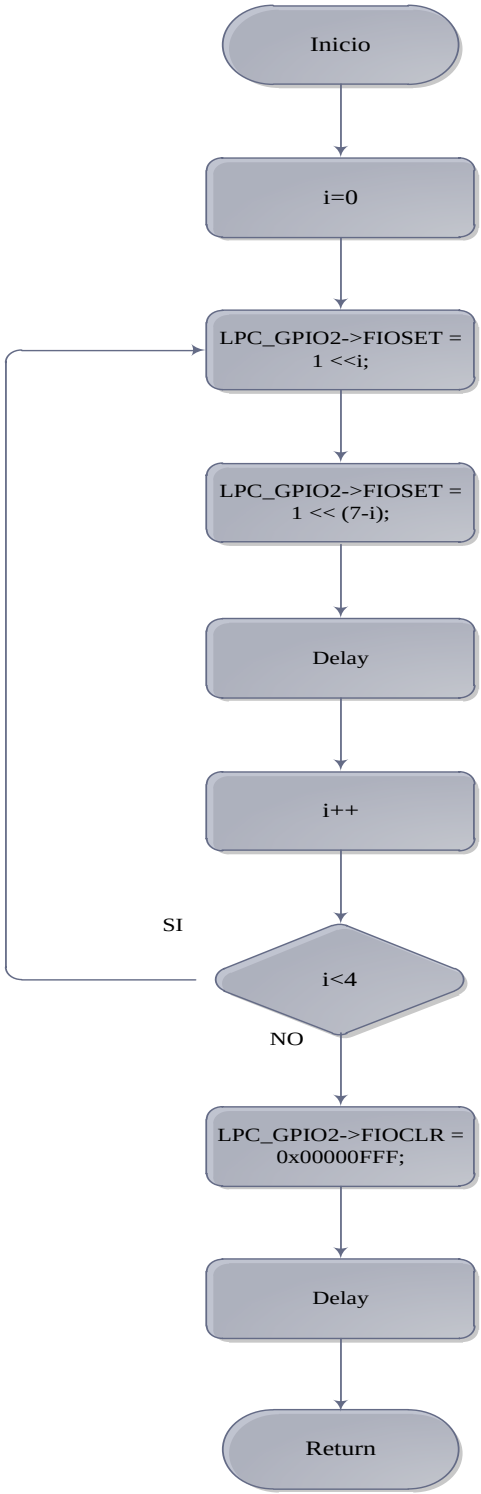


Figura 3.2.3.b: Diagrama ASM secuencia 2

3.2.4 Descripción del algoritmo

- Se inicializan las variables a usar.
- Se inicializan los puertos a usar.
- Cuando recibe un alto del puerto P2[12], realiza la secuencia 1.
- Cuando recibe un bajo del puerto P2[12], realiza la secuencia 2.

Descripción de las subrutinas

- **Secuencia 1**
  - Se inicializan las variables a usar.
  - En el primer ciclo del lazo for se encienden los leds GPIO2 [7] y GPIO2 [0].
  - En el segundo ciclo del lazo for se encienden los leds GPIO2 [6] y GPIO2 [1].
  - En el tercer ciclo del lazo for se encienden los leds GPIO2 [5] y GPIO2 [2].
  - En el cuarto ciclo del lazo for se encienden los leds GPIO2 [4] y GPIO2 [3].

- **Secuencia 2**
  - Se inicializan las variables a usar.
  - En el primer ciclo del lazo for se encienden los leds GPIO2 [4] y GPIO2 [3].

- En el segundo ciclo del lazo for se encienden los leds GPIO2 [5] y GPIO2 [2].
- En el tercer ciclo del lazo for se encienden los leds GPIO2 [6] y GPIO2 [1].
- En el cuarto ciclo del lazo for se encienden los leds GPIO2 [7] y GPIO2[0].

3.2.5 Código Fuente

```
#include<cr_section_macros.h>
#include <NXP/crp.h>
#include "lpc17xx.h"
#include "type.h"
int main (void)
{
uint32_ti, j, k;
/* SystemClockUpdate() updates the SystemFrequency variable */
SystemClockUpdate();
LPC_GPIO2->FIODIR = 0xFFFFEFFF;          /* P2.xx defined as Outputs */
LPC_GPIO2->FIOCLR = 0xFFFFFFFF;          /* turn off all the LEDs */
LPC_GPIO2->FIOMASK= 0x00000000;
while(1) {
        k=4;
        if(LPC_GPIO2->FIOPIN==0x00000000)
```

```
        {
        for(i = 7; i>3; i--){
                                LPC_GPIO2->FIOSET = 1 << k;
                                LPC_GPIO2->FIOSET = 1 << (i-4);
                                k++;
                                for(j = 1000000; j > 0; j--);
                                }
                                LPC_GPIO2->FIOCLR = 0x00000FFF;
                                for(j = 1000000; j > 0; j--);
                                }
    if(LPC_GPIO2->FIOPIN==0x00001000)
    {
     for(i = 0; i< 4; i++)
                    {       LPC_GPIO2->FIOSET = 1 <<i;
                             LPC_GPIO2->FIOSET = 1 << (7-i);
                             for(j = 1000000; j > 0; j--);
                    }

                    LPC_GPIO2->FIOCLR = 0x00000FFF;
                    for(j = 1000000; j > 0; j--);
            }
    }
}
```

3.3 ENCENDIDO DE LEDS MEDIANTE LA TARJETA AVR BUTTERFLY

3.3.1 Descripción

El principal objetivo de este ejercicio es dar a conocer las interrupciones externas del ATMega 169 y validar la funcionalidad del joystick mediante el uso de las mismas. Empleamos el joystick para según la manipulación de éste desplazar los bits que se mostrarán en los LEDs conectados al puerto D. Las acciones que se generan luego de seleccionar la instrucción con el joystick son las siguientes:

- Arriba: desplaza la luz del led dos posiciones a la derecha.
- Abajo: desplaza la luz del led dos posiciones a la izquierda.
- Izquierda: desplaza la luz del led una posición a la izquierda.
- Derecha: desplaza la luz del led una posición a la derecha.
- Centro: borra el contenido del puerto D, y carga un bit en el primer pin del mismo puerto, logrando así activar únicamente la luz del led de aquel pin.

3.3.2 Diagrama de bloques

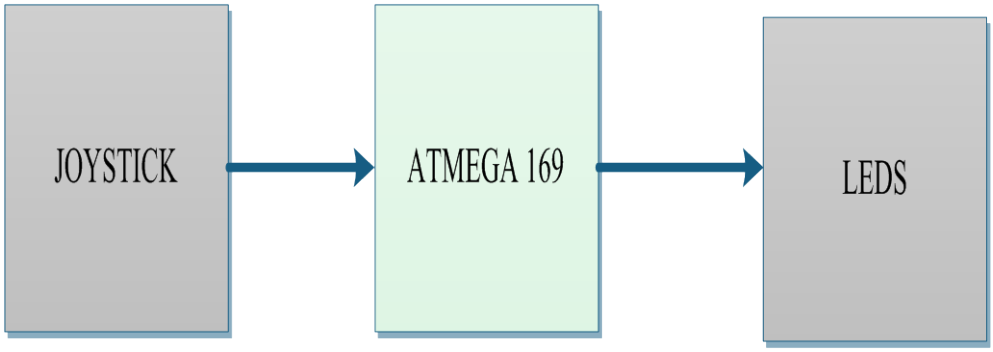


Figura 3.3.2: Diagrama de Bloques

3.3.3 Diagrama ASM

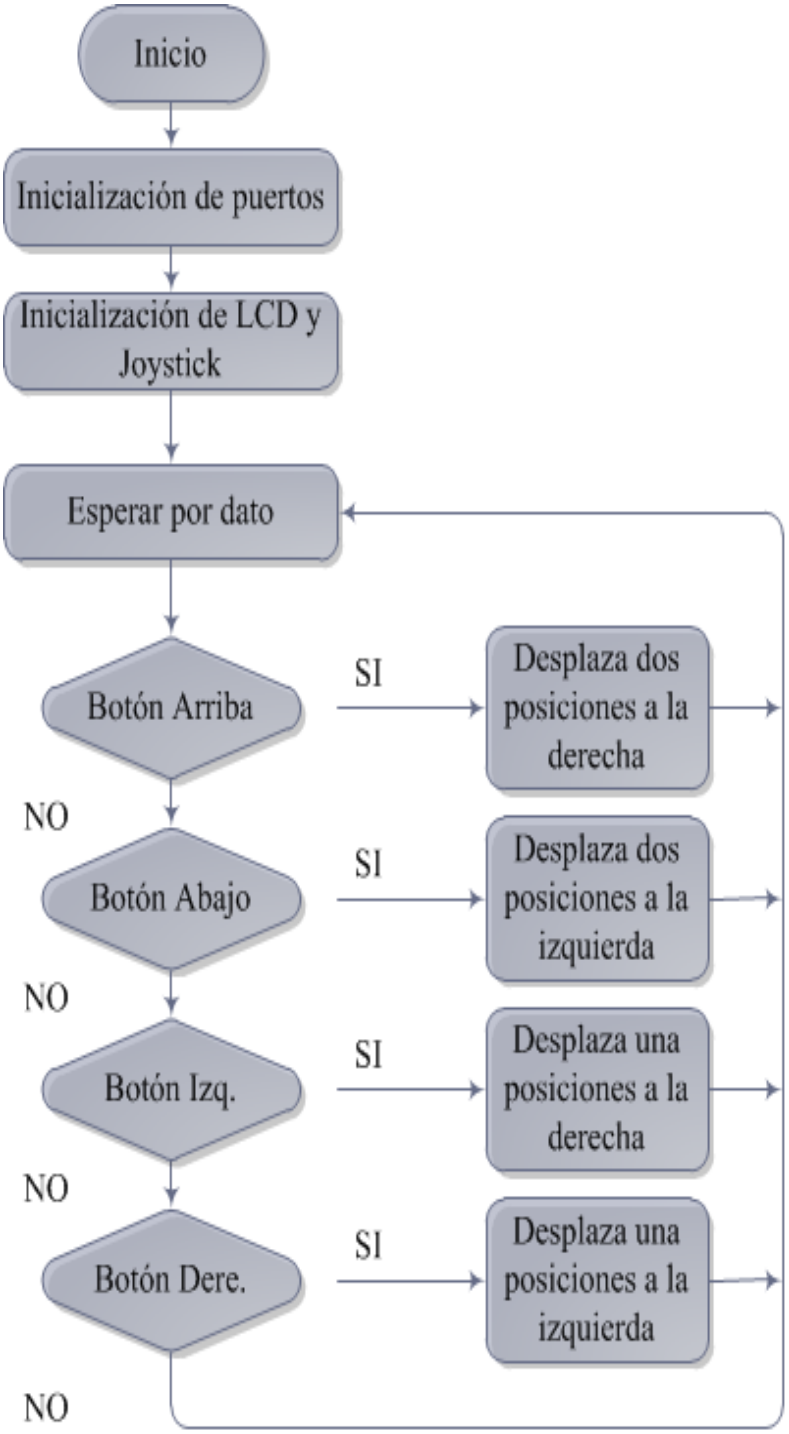


Figura 3.3.3: Diagrama ASM

3.3.4 Descripción del algoritmo

- Se inicializan los puertos a usar.
- Se inicializan el LCD y el Joystick.
- Esperar por el dato enviado del Joystick.
  - Botón Arriba: Desplaza dos posiciones a la derecha
  - Botón Abajo: Desplaza dos posiciones a la izquierda
  - Botón Izquierda: Desplaza una posiciones a la derecha
  - Botón Derecha: Desplaza una posiciones a la izquierda

3.3.5 Código Fuente

```
#include <avr/io.h>

#include <avr/interrupt.h>

#include <avr/signal.h>

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

Pines del ATmega169 conectados con el Joystick:

-------------------------------------------------------------

Bit 7 6 5 4 3 2 1 0

-------------------------------------------------------------

PORTB B A O
```

```
PORTE D C

-------------------------------------------------------------

PORTB | PORTE B A O D C => posición

-------------------------------------------------------------

*/

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

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

#define ARRIBA 0

#define ABAJO 1

#define IZQUIERDA 2

#define DERECHA 3

#define CENTRO 4

#define NO_VALIDA 5

#define posicion_A 6 //ARRIBA

#define posicion_B 7 //ABAJO

#define posicion_C 2 //DERECHA

#define posicion_D 3 //IZQUIERDA

#define posicion_O 4 //CENTRO
```

```
#define VERDADERO 1

#define FALSO 0

volatile unsigned char SELECCION = 0;

volatile unsigned char SELECCION_VALIDA = 0;

void inicializar(void);

void manejar_interrupcion(void);

void obtener_seleccion(void);

int main(void)

{

inicializar();

sei();

while(1)

{

SMCR = ((1<<SM1)|(1<<SE));

}

return 0;

}
```

```
void inicializar(void)

{

CLKPR = (1<<CLKPCE);

CLKPR = (1<<CLKPS3);

while(CLKPR & (1<<CLKPCE));

DDRB |= 0xD0;

PORTB |= MASCARA_PINB;

DDRE |= 0x0C;

PORTE |= MASCARA_PINE;

DDRB = 0;//entrada

PORTB = MASCARA_PINB;//habilitar PULL-UPs

DDRE = 0;//entrada

PORTE = MASCARA_PINE;//habilitar PULL-UPs

PCMSK1 |= MASCARA_PINB;

PCMSK0 |= MASCARA_PINE;

EIFR = ((1<<PCIF1)|(1<<PCIF0));

EIMSK = ((1<<PCIE1)|(1<<PCIE0));
```

```
DDRD = 0xFF;

seleccion = ARRIBA;

else if((joystick & (1<<B)))

seleccion = ABAJO;

else if((joystick & (1<<C)))

seleccion = DERECHA;

else if((joystick & (1<<D)))

seleccion = IZQUIERDA;

else if((joystick & (1<<O)))

seleccion = CENTRO;

else seleccion = NO_VALIDA;

if(seleccion != NO_VALIDA)

{

if(!SELECCION_VALIDA)

{

SELECCION = seleccion;

SELECCION_VALIDA = VERDADERO;
```

```
}

}

EIFR = ((1<<PCIF1)|(1<<PCIF0));

obtener_seleccion();

}

void obtener_seleccion(void)

{

unsigned char seleccion;

cli();

if(SELECCION_VALIDA)

{

seleccion = SELECCION;

SELECCION_VALIDA = FALSO;

}

else seleccion = NO_VALIDA;

if(seleccion != NO_VALIDA)

{
```

```
switch(seleccion)
{
case ARRIBA:
PORTD <<= 2;
break;
case ABAJO:
PORTD >>= 2;
break;
case IZQUIERDA:
PORTD <<= 1;
break;
case DERECHA:
PORTD >>= 1;
break;
case CENTRO:
PORTD = 1;
default:
```

```
break;
}
}
sei();
}
SIGNAL(SIG_PIN_CHANGE0)
{
manejar_interrupcion();
}
SIGNAL(SIG_PIN_CHANGE1)
{
manejar_interrupcion();
}
PORTD = 0x00;
}
void manejar_interrupcion(void)
{
```

unsigned char joystick;

unsigned char seleccion;

joystick = ((~PINB) & MASCARA_PINB);

joystick |= ((~PINE) & MASCARA_PINE);

if((joystick & (1<<A))). [3]

3.4 ESPECIFICACIÓN DEL PROYECTO

3.4.1 Descripción

La realización del proyecto se basa en el funcionamiento de un motor BLDC. El programa implementado para la tarjeta electrónica AVR Butterfly (con microcontrolador ATMega169) transmite datos, los cuales son seleccionados mediante las respectivas instrucciones del joystick, la emisión y recepción de dichos datos se efectúa mediante módulos de radio frecuencia para la comunicación con la tarjeta LPCX769 que controla al motor BLDC.

EMISOR

El circuito emisor está formado por el AVR Butterfly que cuenta con un joystick que es el encargado de seleccionar la trama que se desea enviar, en este caso se envían 4 diferentes tipos de señales a través de la comunicación UART presente en el ATMega 169, la trama enviada contiene

un carácter que da una instrucción para que la recepte la LPC para mover el motor.

Las instrucciones y caracteres que se envían por tramas son los siguientes:

- On//Off (Izquierda): Se envía el carácter “O” enciende y apaga el motor.
- Acelerar (Arriba): Se envía el carácter “A”. aumenta la velocidad del motor.
- Bajar (Abajo): Se envía el carácter “B”. disminuye la velocidad del motor.
- Invertir Giro (Derecha): Se envía el carácter “I”. invierte el sentido de giro.
- Cada trama enviada está compuesta por 8 bits y se le agrega un bit de inicio y un bit de fin que indican al receptor el inicio y el final de la transmisión entre una trama y otra.

Para proceder con la trasmisión es necesario que tanto la tarjeta AVR Butterfly como el módulo RF se encuentren operando a una tasa de 9600 baudios, para lograr así que las tramas sean captadas correctamente por el transceiver,

La señal a trasmitir la obtenemos de la AVR Butterfly, a través del puerto Tx, el cual se conecta a una etapa de inversión constituida principalmente por un transistor (2N222).

Luego de pasar por este filtro se conecta al pin Rx del transceiver y se procede a la modulación de la señal que va a ser transmitida.

Mediante la LCD presente en el AVR Butterfly mostramos por pantalla la instrucción que ha sido seleccionada por el usuario.

RECEPTOR

La tarjeta LPC1769 se encarga de recibir las instrucciones del AVR Butterfly, mediante radiofrecuencia a través del módulo de recepción HM-TR 433, dichas instrucciones son utilizadas para realizar las funciones elegidas por el usuario y hacer funcionar el motor según la opción escogida.

Luego de que la instrucción es captada por el módulo, la señal llega al pin Tx del transceiver y se demodula, luego esta se conecta al puerto de recepción RX que es el pin 10 de la tarjeta LPC1769, dicha recepción está configurada para recibir datos a una tasa de 9600 baudios que nos permite asegurar que los datos del transmisor están siendo interpretados correctamente.

Al recibir las respectivas instrucciones el motor opera de la siguiente manera:

- On//Off: enciende y apaga el motor.
- Acelerar: aumenta la velocidad del motor.
- Bajar: disminuye la velocidad del motor.
- Invertir Giro: cambia el sentido de giro.

3.4.2 Diagrama de bloques

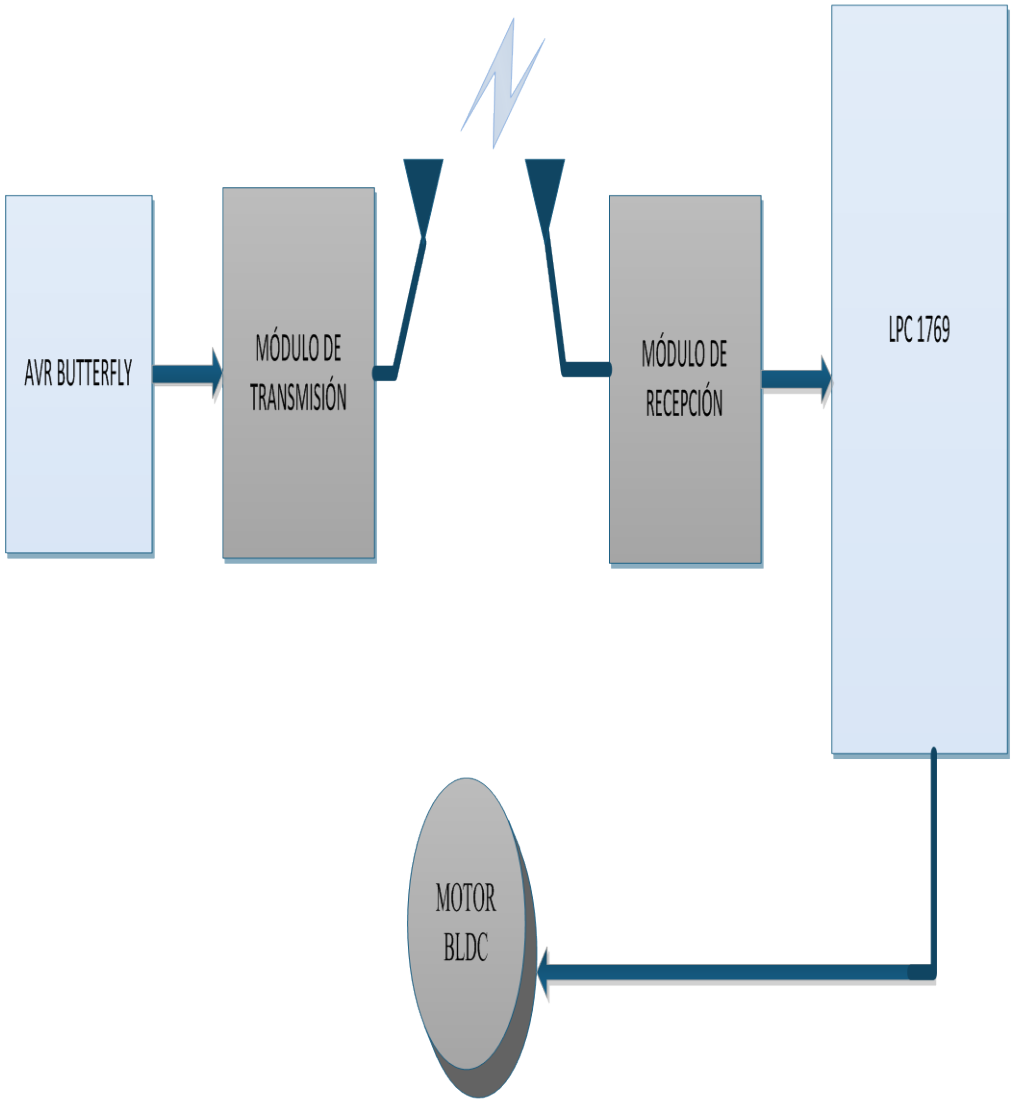


Figura 3.4.2: Diagrama de Bloque

3.4.3 Diagrama ASM

EMISOR

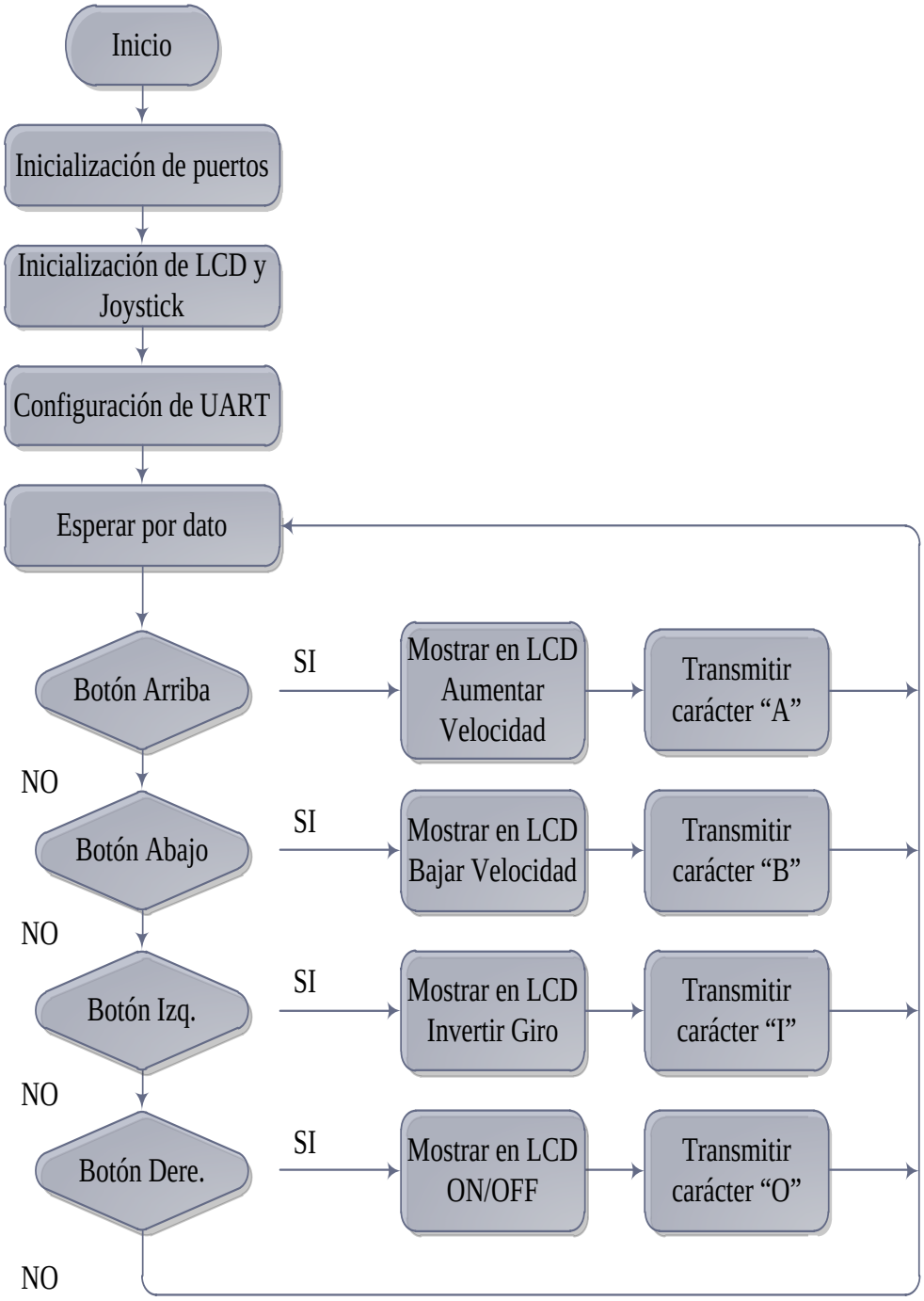


Figura 3.4.3.a: Diagrama ASM Emisor

RECEPTOR

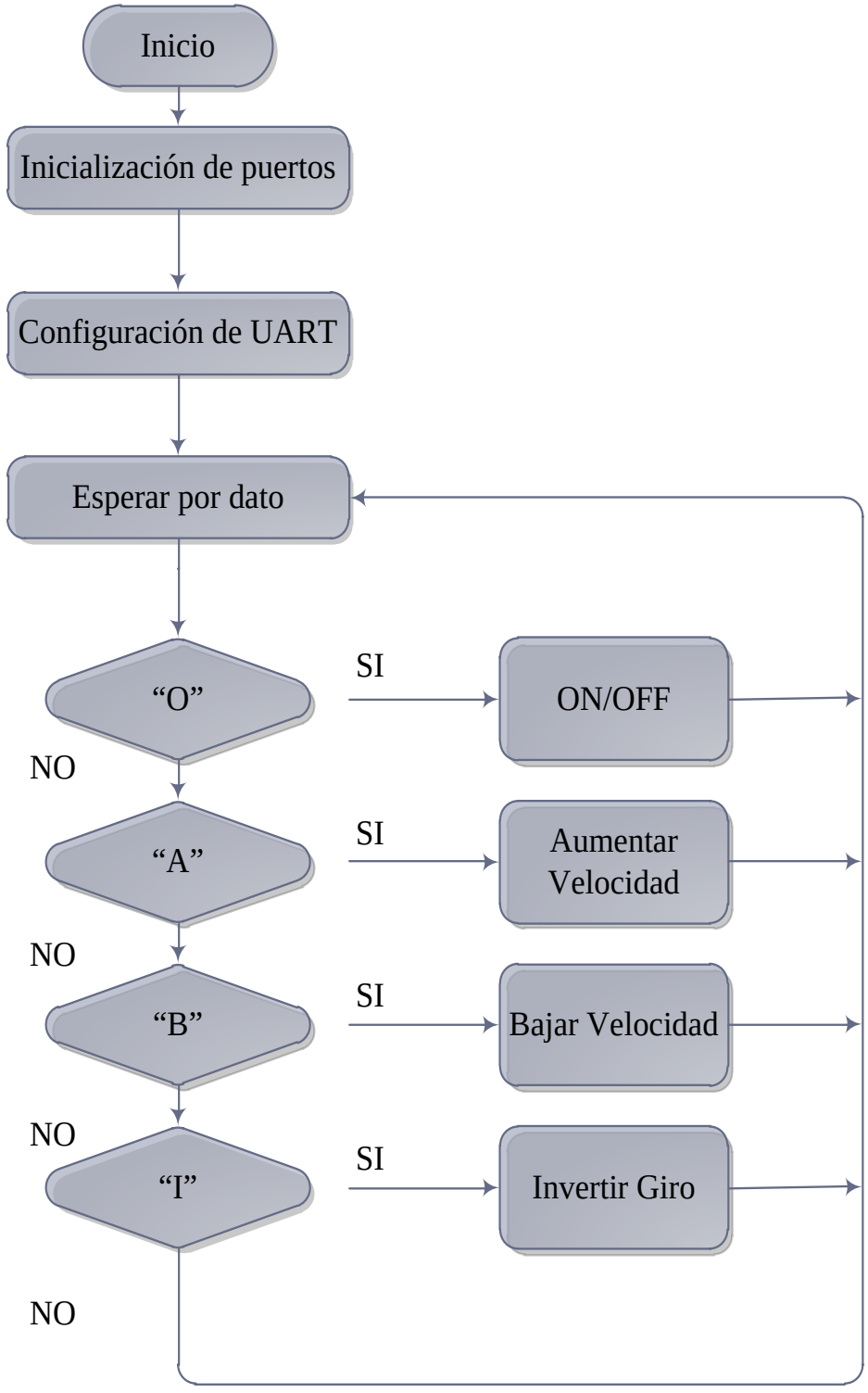


Figura 3.4.3.b: Diagrama ASM Receptor

3.4.4 Descripción del algoritmo

EMISOR

- Se inicializan los puertos a usar.
- Se inicializan el LCD y el Joystick.
- Configuración del puerto UART.
- Esperar por el dato enviado del Joystick.
  - Botón Arriba: Muestra en la pantalla Aumentar velocidad y transmite el carácter “A”.
  - Botón Abajo: Muestra en la pantalla Bajar velocidad y transmite el carácter “B”.
  - Botón Izquierda: Muestra en la pantalla Invertir Giro y transmite el carácter “I”.
  - Botón Derecha: Muestra en la pantalla ON/OFF y transmite el carácter “O”.

RECEPTOR

- Se inicializan los puertos a usar.
- Configuración del puerto UART.
- Esperar por el dato enviado por la AVR Butterfly.

  - Letra O: Encender y/o Apagar motor BLDC.
  - Letra A: Aumenta la velocidad del motor BLDC.
  - Letra B: Baja la velocidad del motor BLDC.

- Letra I: Invierte el giro del motor BLDC.

3.4.5 Código Fuente

EMISOR

```
#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("M_J_MOROCHO-ANDREA_GUERRA");

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);



while (1)
```

```
   {
if (statetext) //Imprime en la LCD
{
LCD_puts_f(statetext, 1);
LCD_Colon(0);
statetext = NULL;
}
input = getkey();
switch (input)
{
case KEY_ENTER:
statetext = PSTR("CENTRO");
Usart_Tx('C');
break;

case KEY_NEXT:
statetext = PSTR("ON/OFF");
```

```
        Usart_Tx('O');
break;
case KEY_PREV:
statetext = PSTR("INVERTIR GIRO");
Usart_Tx('I');
break;
case KEY_PLUS:
statetext = PSTR("AUMENTAR VELOCIDAD");
        Usart_Tx('A');
break;
case KEY_MINUS:
statetext = PSTR("BAJAR VELOCIDAD");
Usart_Tx('B');
break;
default:
break;
        }
```

```
    }
return 0;
}
```

RECEPTOR

```
#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;
 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==0x41){// ASCII DE LA LETRA A
                 LPC_GPIO2->FIOSET = 0x000000FF;
                LPC_GPIO2->FIOCLR = 1 << 1;
                }
if(*UART3Buffer==0x42){// ASCII DE LA LETRA B
                        LPC_GPIO2->FIOSET = 0x000000FF;
                        LPC_GPIO2->FIOCLR = 1 << 2;
```

```
                    }
                    if(*UART3Buffer==0x4F){// ASCII DE LA LETRA O
                        LPC_GPIO2->FIOSET = 0x000000FF;
                        LPC_GPIO2->FIOCLR = 1 << 3;
                    }
                    if(*UART3Buffer==0x49){// ASCII DE LA LETRA I
                        LPC_GPIO2->FIOSET = 0x000000FF;
                        LPC_GPIO2->FIOCLR = 1 << 4;
                    }
                    if(*UART3Buffer==0x43){// ASCII DE LA LETRA C
                        LPC_GPIO2->FIOSET = 0x000000FF;
                    }
                    UART3Count = 0;
                    LPC_UART3->IER = IER_THRE | IER_RLS | IER_RBR;
                /* Re-enable RBR */
                }
        }
}
```

CAPÍTULO 4

PRUEBAS Y SIMULACIONES

En este capítulo se muestra los diagramas de conexiones de cada uno de los ejercicios realizados y detallados en el capítulo anterior, se presenta el listado de materiales necesarios para cada ejercicio con su respectiva foto.

Las simulaciones correspondientes al transmisor también forman parte de este capítulo, con el objetivo de observar que la transmisión de datos mediante comunicación UART se realiza en forma correcta.

Los diagramas de conexiones son basados en las hojas de datos esquemáticos de las tarjetas LPC1769 y el Kit AVR Butterfly, estos diagramas han sido diseñados bajo nuestra creatividad luego de estudiar dichas tarjetas físicamente. La simulación del transmisor fue realizada en ISSIS – Proteus esta se la obtuvo de los ejemplos del Kit AVR Butterfly y cargada con él .hex de nuestro proyecto.

4.1 INTRODUCCIÓN

A continuación se pone en manifiesto detalle a detalle cada uno de los ejercicios que fueron realizados en la plataforma interactiva, con la finalidad de demostrar el funcionamiento y los recursos disponibles en la comunicación UART.

4.2 COMUNICACIÓN UART ENTRE DOS TARJETAS LPC1769

En este ejercicio nuestro principal objetivo es analizar la comunicación serial UART entre dos tarjetas LPC1769 utilizando una para la transmisión y otra para la recepción de datos.

Para la transmisión hacemos uso de una botonera que se conecta al pin 50 de la tarjeta LPC1769, y envía la señal desde el pin 9 de la primera tarjeta al pin 10 de la tarjeta receptora y gracias al encendido del led conectado al pin 50 de la segunda tarjeta podemos observar que se realiza la comunicación de manera óptima.

4.2.1 Lista de Componentes

- 2 Tarjeta LPC1769.
- Cable Adaptador USB.
- 1 Diodo LED.
- 2 Resistencias de 330 Ω.
- Botonera.

4.2.2 Diagrama de conexiones

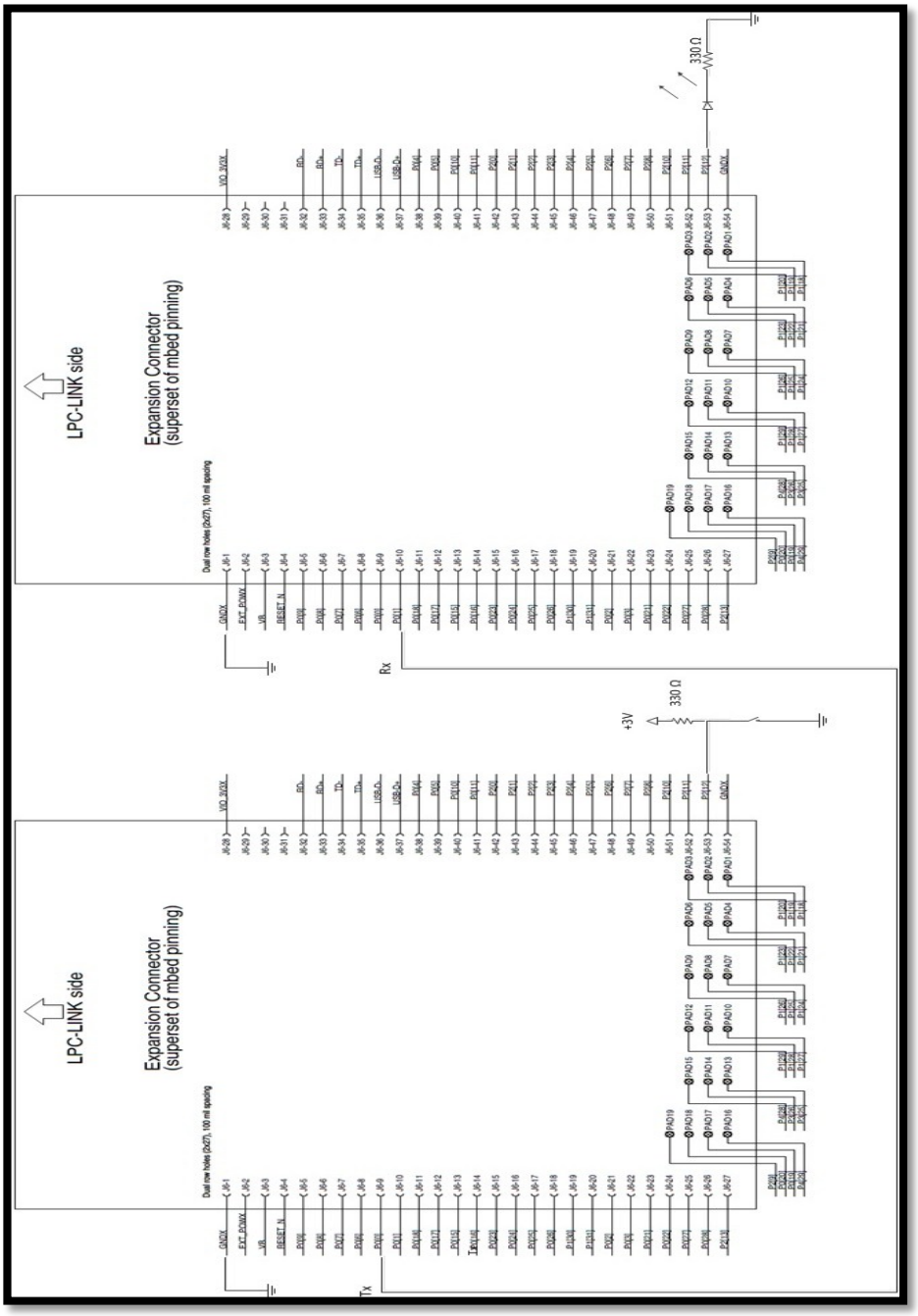


Figura 4.2.2: Diagrama de conexiones para la comunicación UART entre dos tarjetas LPC1769.

4.2.3 Implementación

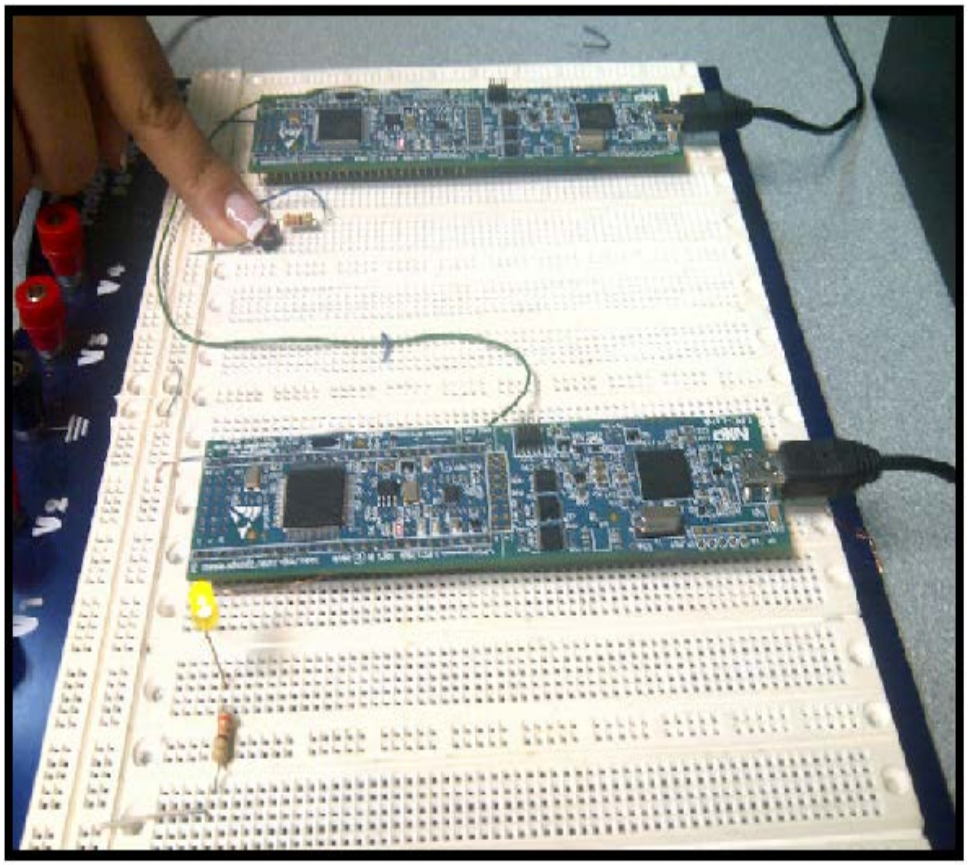


Figura 4.2.3: Comunicación UART entre dos tarjetas LPC1769.

4.2.4 Conclusiones

Se pudo establecer una comunicación UART usando el GPIO0, así mediante P0[0] correspondiente al pin 9 se usó para la transmisión de datos y el mediante P0[1] correspondiente al pin 10 para la recepción de datos. Se usó el UART3 y un baudrate de 3600 para la comunicación entre las tarjetas, fueron inicializados mediante la función UARTInit(x, y) el parámetro x para el uart a utilizar y el baudrate.

4.3 ENCENDIDO DE LEDS MEDIANTE LA TARJETA LPC1769

El desarrollo de este ejercicio tiene como propósito familiarizarnos con los registros de la tarjeta LPC1769 e interactuar con las diferentes funciones que esta nos ofrece.

El funcionamiento consiste en el desplazamiento de 8 leds, los cuales se encienden dos a la vez de acuerdo a la siguiente secuencia:

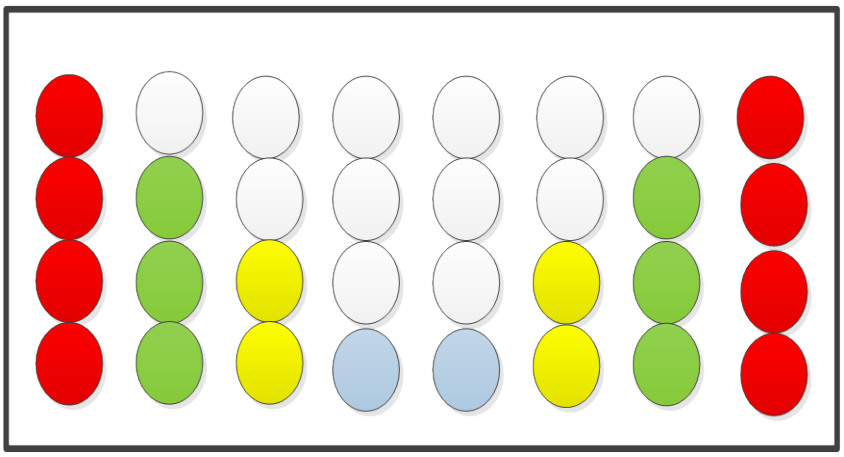


Figura 4.3: Secuencia de encendido de leds.

4.3.1 Lista de Componentes

- 1 Tarjeta LPC1769.
- Cable Adaptador USB.
- 8 Diodos LED’s.
- 9 Resistencias de 330 Ω.
- Botonera.
- Cable UTP

4.3.2 Diagrama de conexiones

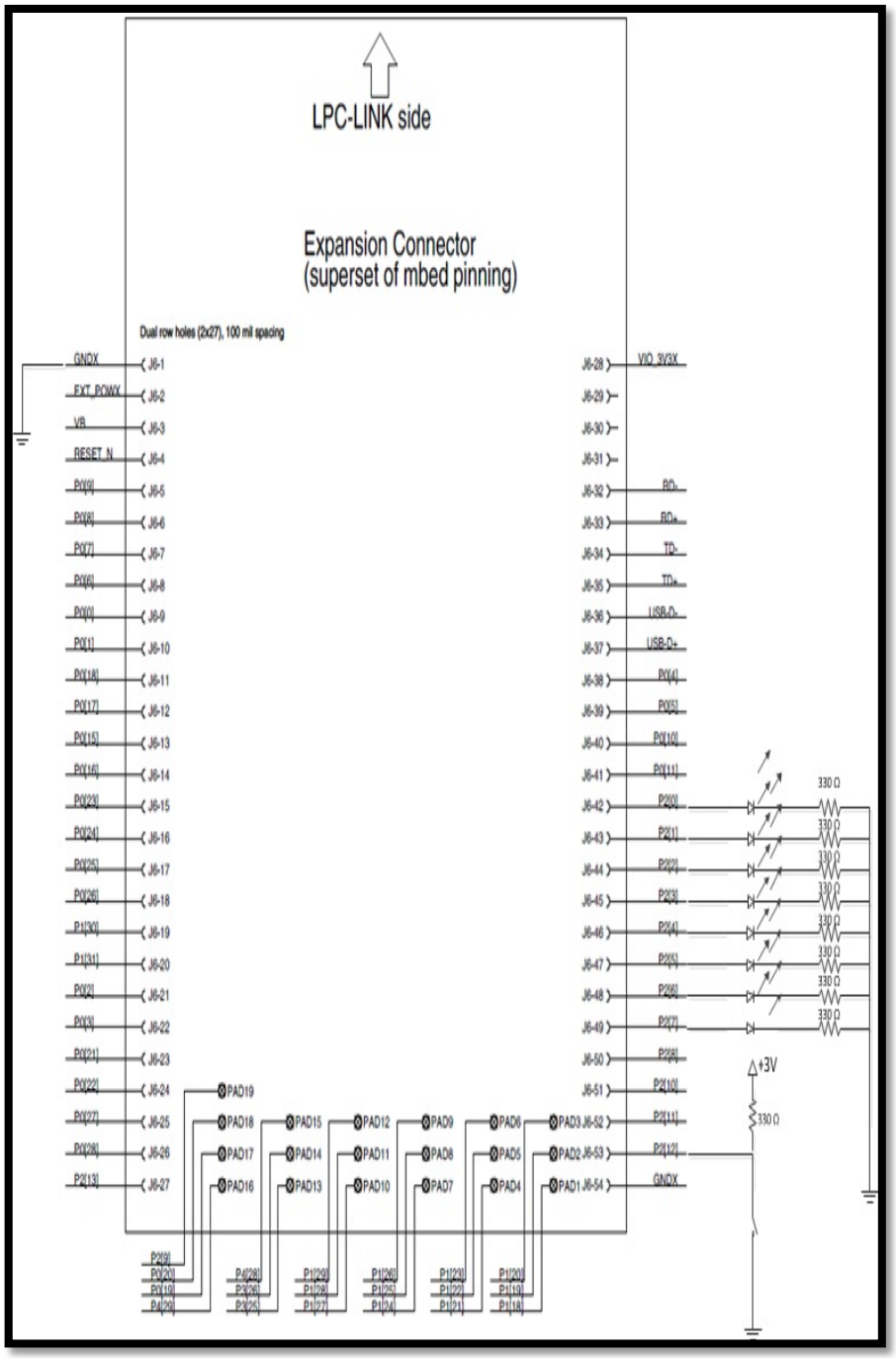


Figura 4.3.2: Diagrama de conexiones para e el desplazamiento de leds usando la tarjeta LPC1769.

4.3.3 Implementación

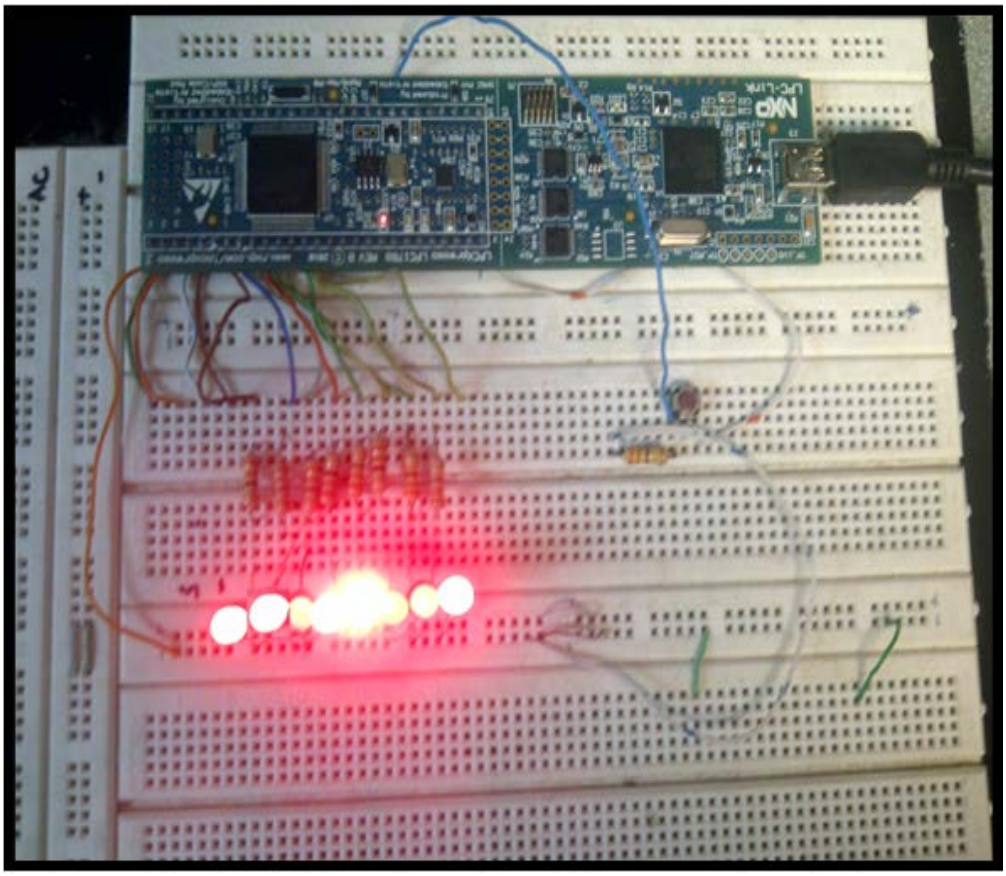


Figura 4.3.3: Desplazamiento de leds con la tarjeta LPC1769.

4.3.4 Conclusiones

La asignación de puertos es vital a la hora de programar la LPC1769, hay que revisar las hojas de datos y fijarse en los puertos que vienen pre asignados. La función FIOPIN asigna puertos como entradas o salidas, FIOSET setea el puerto y FIOCLR encera el puerto; estas tres funciones se colocan en la cabecera del programa y es la manera de inicializar los puertos en esta tarjeta.

4.4 ENCENDIDO DE LEDS MEDIANTE LA TARJETA AVR BUTTERFLY

Este ejercicio es fundamental para comprender el funcionamiento de la tarjeta AVR Butterfly; el desarrollo del mismo consiste en emplear los puertos B y E del ATmega 169 para acceder al Joystick, así como también se usará el puerto D para poder acceder a las conexiones externas del AVR Butterfly .

Finalmente mediante los leds situados en dichas conexiones externas se observará la acción seleccionada por el usuario a través del joystick.

Los leds se encenderán uno a uno de acuerdo a la selección mediante el joystick: arriba, desplaza 2 posiciones a la derecha; abajo, desplaza 2 posiciones a la izquierda; izquierda, desplaza una posición a la izquierda; derecha, desplaza una posición a la derecha y el centro para inicializar y finalizar con el ejercicio.

4.4.1 Lista de Componentes

- 1 Tarjeta AVR Butterfly.
- Cable UTP.
- 8 Diodos LED’s.
- 8 Resistencias de 330 Ω.
- 2 Baterías AA – 1.5v.
- 1 Porta baterías AA.
- 1 Protoboard

4.4.2 Diagrama de conexiones

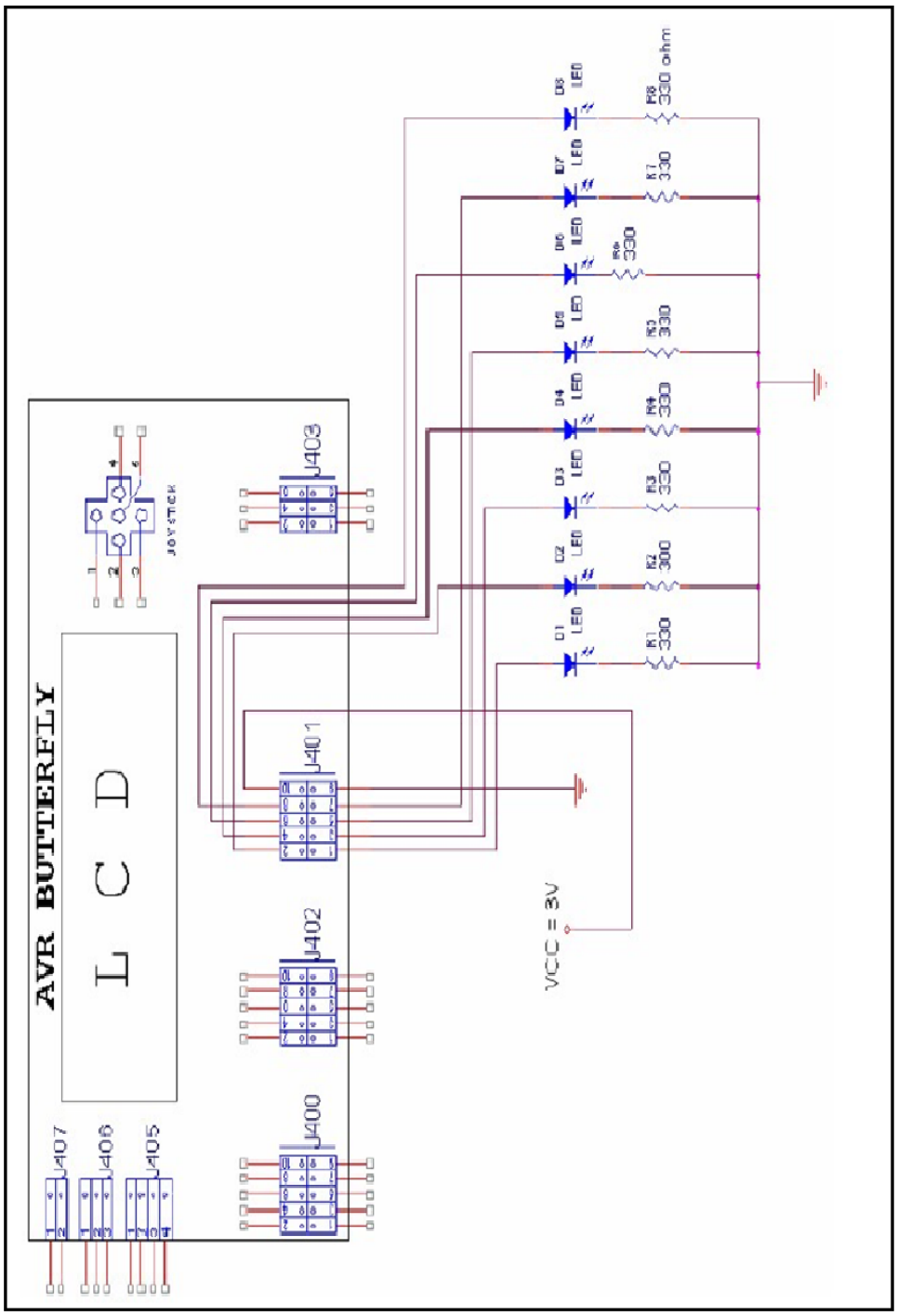


Figura 4.4.2: Diagrama de conexiones para e el desplazamiento de leds usando la tarjeta AVR Butterfly.[3].

4.4.3 Implementación

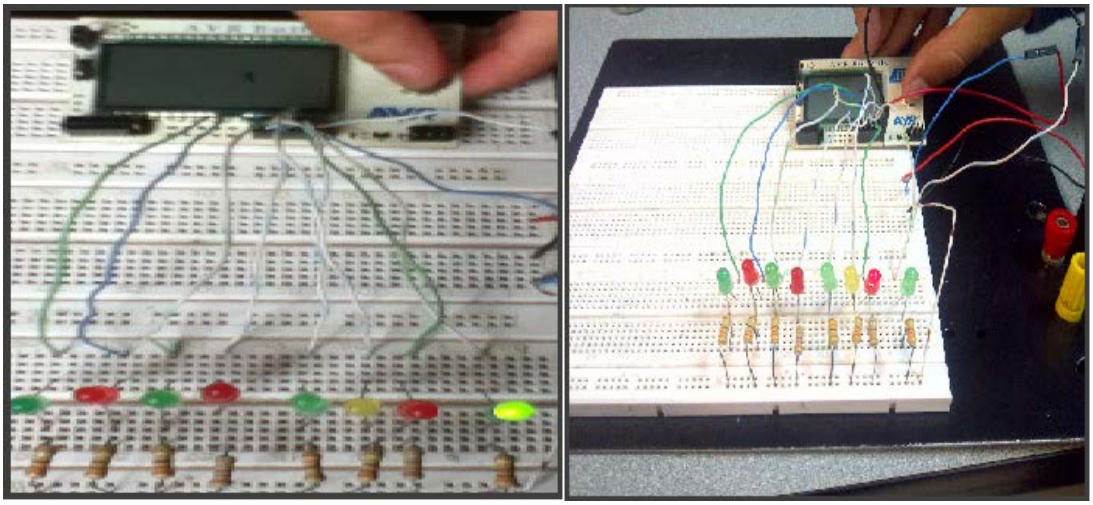


Figura 4.4.3: Desplazamiento de leds con la tarjeta LPC1769.

4.4.4 Conclusiones

El manejo del joystick se emplea como una entrada para el usuario con el propósito de operar el AVR Butterfly para realizar diferentes instrucciones.

Este ejercicio ofrece un manejo sencillo que resulta de gran utilidad para la interacción con el usuario, ya que a través de las interrupciones externas podemos conectar cualquier dispositivo que queramos controlar.

4.5 PROYECTO: APLICACIÓN DE LA COMUNICACIÓN UART PARA EL CONTROL DE MOTORES BLDC.

Este ejercicio contiene la implementación del proyecto final asignado, cuyo objetivo principal es el control de motores BLDC mediante módulos RF,

haciendo uso de las tarjetas AVR Butterfly, LPC1769 y módulos de radio frecuencia.

Se basa fundamentalmente en dos etapas; la etapa de transmisión está formada por el kit AVR Butterfly que gracias a su respectiva programación realizada en el AVR Studio 4 nos permite enviar las instrucciones controladas por el joystick. Una vez que se genera la señal ingresa al módulo de transmisión RF que es el encargado de modular la señal y luego transmitirla.

La etapa de recepción está constituida por el módulo receptor que se encarga de demodular la señal enviada, que por medio del UART de la LPC1769 son procesadas, produciendo de esta manera los respectivos cambios en el motor BLDC dependiendo de los comandos que sean presionados por el joystick.

4.5.1 Lista de Componentes

- 1 Tarjeta LPC1769.
- 1 AVR Butterfly.
- 2 HM-TR 433.
- 1 Resistencia de 1KΩ.
- 1 Resistencia de 220Ω.
- 1 Transistor 2N222.
- Cable Adaptador USB.

- Cables UTP.
- 2 Baterías de 1.5V.
- 2 Fuentes de 5V.
- Porta baterías AA.
- Motor BLDC.

4.5.2 Diagrama de conexiones

Simulaciones del Emisor

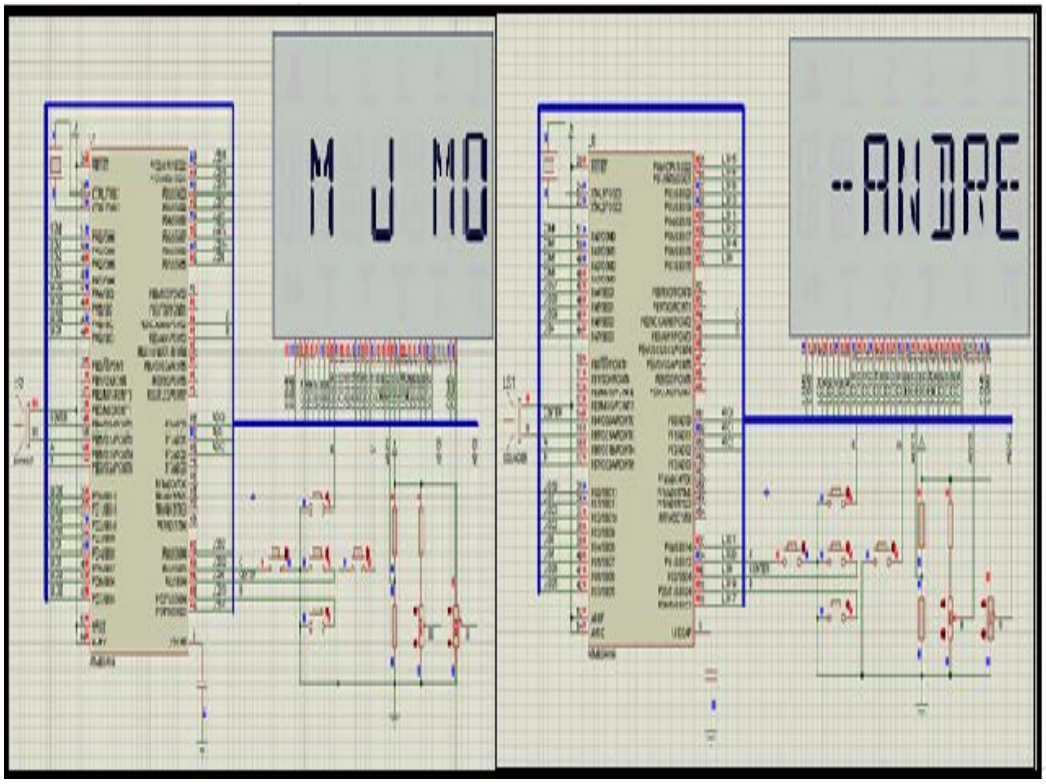


Figura 4.5.2a: Simulación de la tarjeta AVR Butterfly con los nombres de las integrantes.

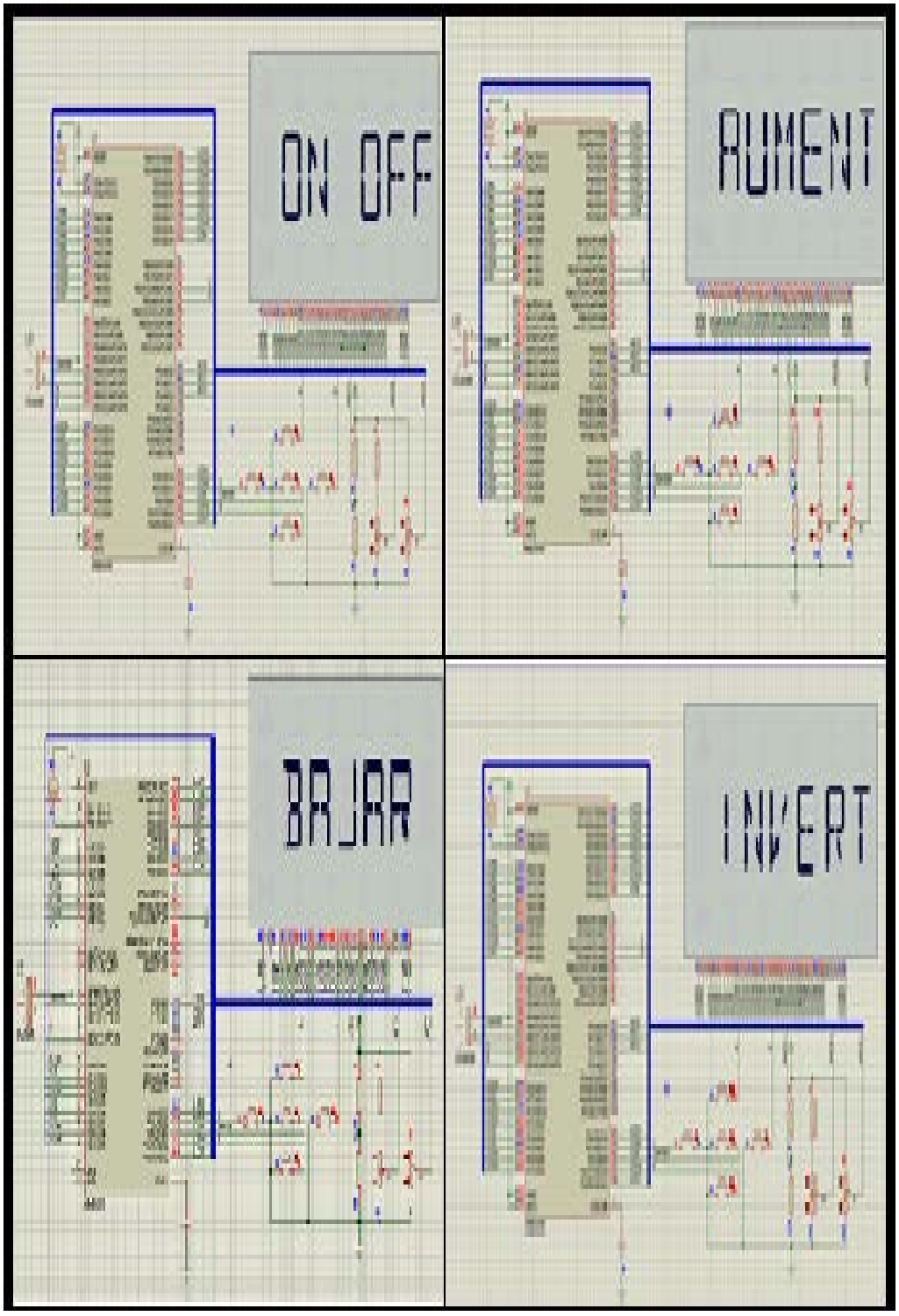


Figura 4.5.2b: Simulación de la tarjeta AVR Butterfly con las instrucciones para el control del motor BLDC.

EMISOR

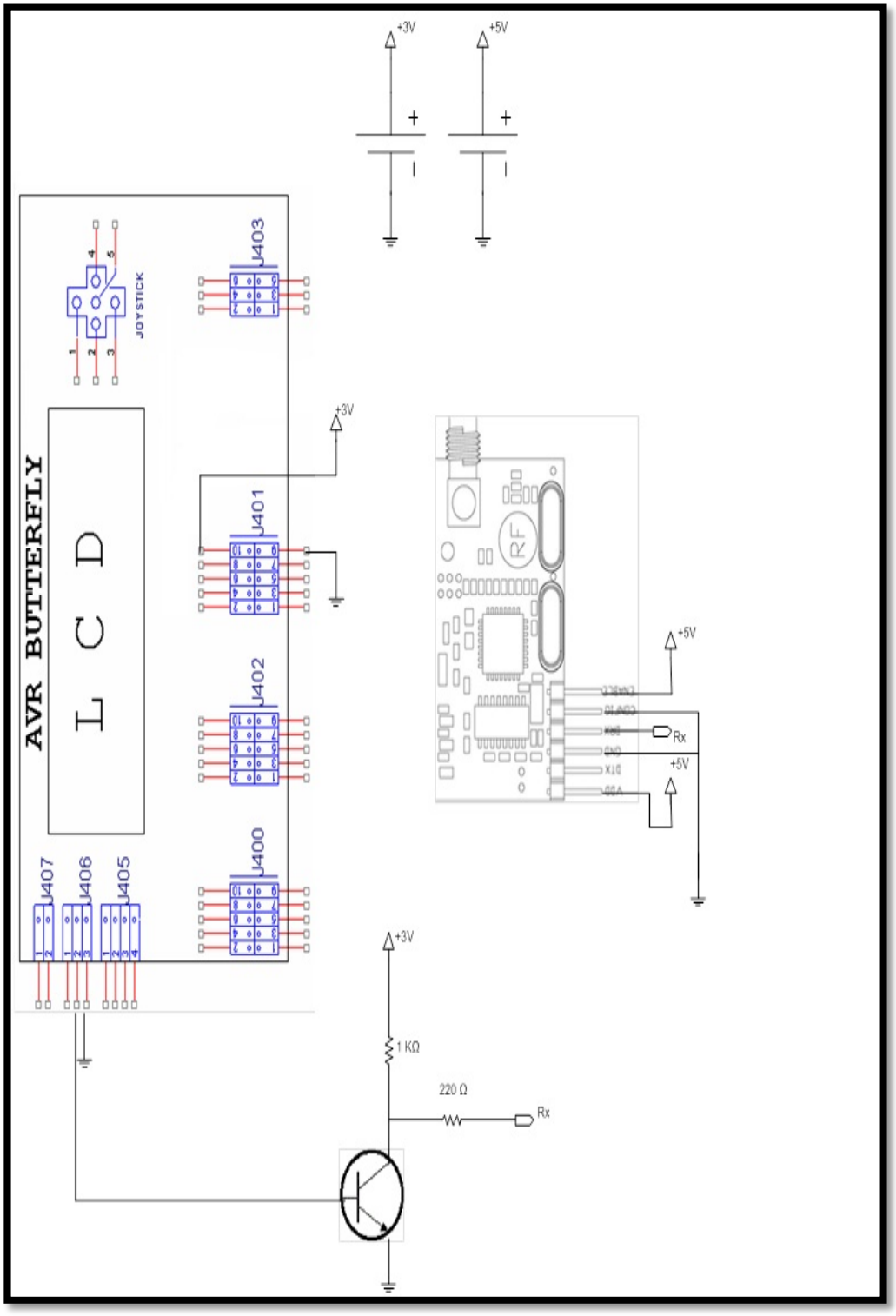


Figura 4.5.2.1: Diagrama de conexiones del circuito emisor

RECEPTOR

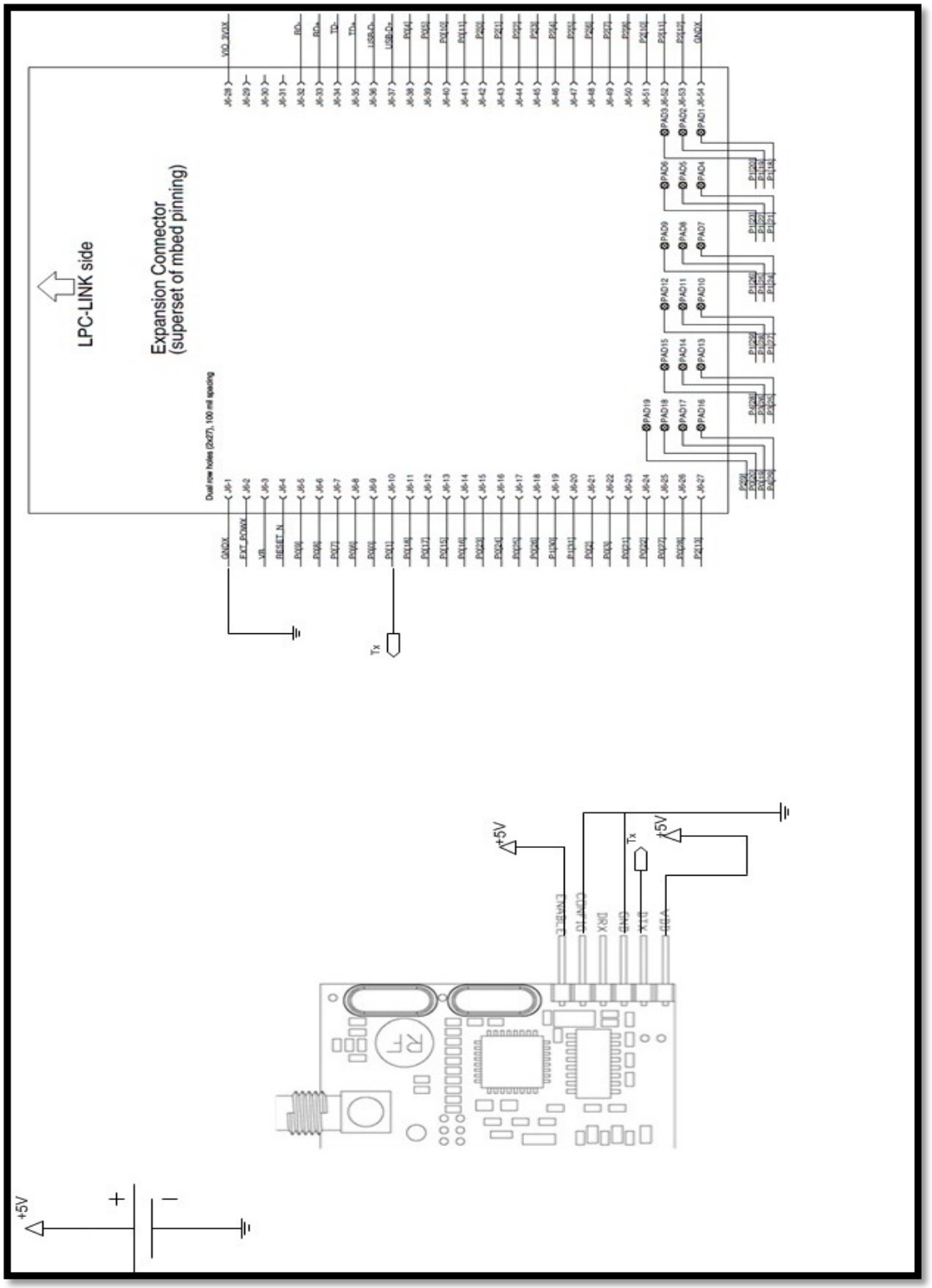


Figura 4.5.2.2: Diagrama de conexiones del circuito receptor.

4.5.3 Implementación

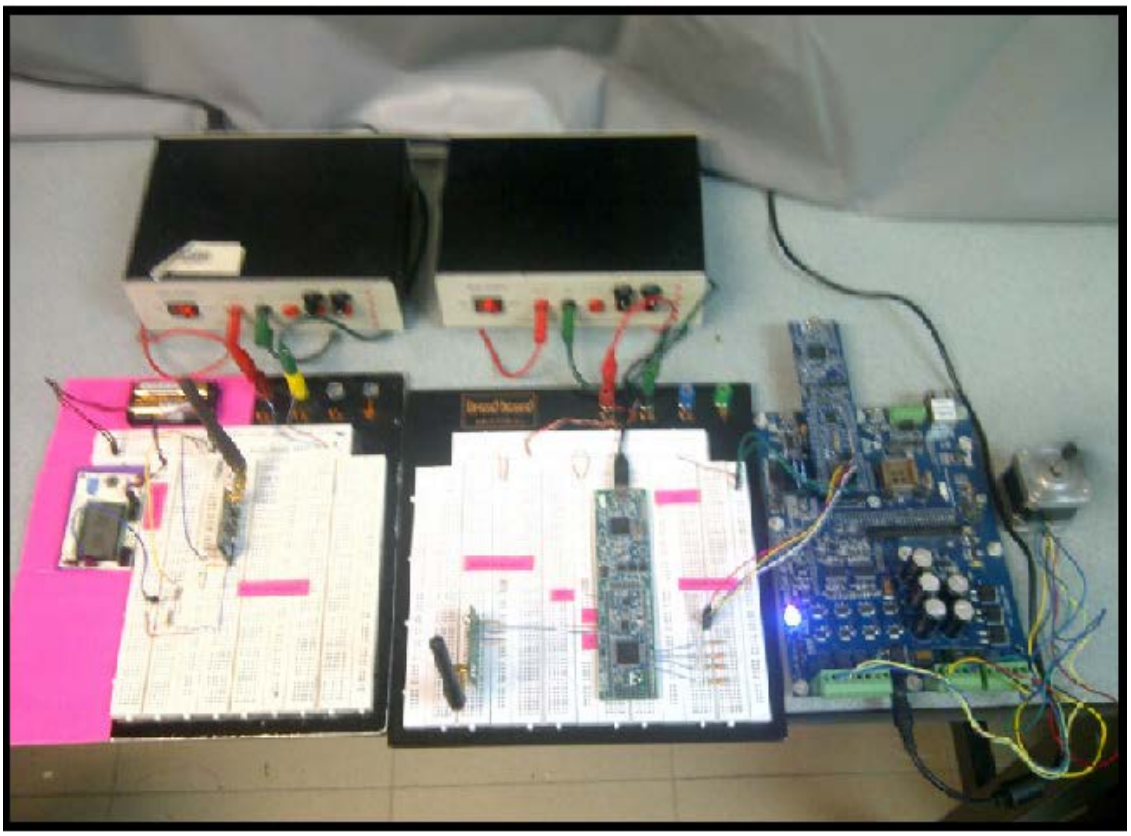


Figura 4.4.3: Implementación del proyecto final.

CONCLUSIONES

1. El objetivo principal del proyecto es la comunicación inalámbrica, para esto se hizo uso de módulos de radiofrecuencia HM-TR 434 TTL, estos nos permitieron realizar un control de los motores BLDC sin necesidad de cableado y de manera más simple al usuario, ya que elJoystick integrado en el Kit AVR Butterfly facilita la interacción del usuario con el motor.

2. Para la implementación de los módulos de radiofrecuencia se utilizó los puertos UART del Kit AVR Butterfly y de la tarjeta LPCXpresso, en ambas tarjetas se programó dicha comunicación además de establecer conexión entre ellas mediante cableado. Una vez que estas podían enviar y recibir datos entre sí, lo cables que las unían fueron remplazados por módulos de comunicación.

3. Este proyecto fue desarrollado con lenguaje C, utilizando AVR Studio 4 para programar el microcontrolador ATMega169 del Kit AVR Butterfly, y el programa LPCXpresso para la tarjeta LPC1769. Las librerías del

LPCXpresso fueron de gran ayuda ya que sirvieron de ejemplo y guía a la hora de programar. La ventaja de usar lenguaje C es que ha utilizado en nuestra formación profesional.

4. Con este prototipo hemos destacado que la comunicación inalámbrica es muy importante hoy en día, y esta nos permite controlar de manera remota y sencilla cualquier prototipo a usar, en nuestro caso motores BLCD. Al integrar la tarjetas y separar un módulo de transmisión y otro de recepción, permite que dichos motores sean controlados a distancia.

RECOMENDACIONES

1. Se recomienda la lectura de las hojas de datos de las diferentes tarjetas tanto para la AVR Butterfly y LPCxpresso 1769, además de la hoja de datos del módulo HMTR-433 ya que es importante conocer su estructura, funcionamiento y limitaciones, además de la configuración en la cual estos funcionan, sobre todo para el módulo de transmisión ya que es un elemento muy sensible y si no se toman las precauciones, este se podría quemar.

2. Para la comunicación inalámbrica se ha utilizado el módulo HMTR-433 estos trabajan con una fuente DC a +5v, se recomienda que estos tengan su propia fuente ya que las tarjetas utilizadas trabajan a +3v, además de tener en cuenta de realizar de manera correcta la referencia de los circuitos con GND para así evitar lecturas erróneas y daños de los módulos y elementos utilizados.

3. No utilizar los módulos HMTR-433 rs232 para comunicar inalámbricamente las tarjetas AVR Butterfly y LPCxpresso puesto que las tarjetas utilizan señales TTL y no es necesario el uso de estos tipos de módulos de comunicación, tampoco es necesario usar un max232, se puede hacer uso de este integrado si va a realizar una comunicación con una PC, ya que este eleva el voltaje y puede quemar los módulos utilizados ya que supera las limitaciones del mismo.

BIBLIOGRAFÍA

[1]. Atmel Corporation, AVR Butterfly, guía para el usuario.

http://http://www.atmel.com/Images/doc4271.pdf

Fecha de consulta: 25/03/2012

[2].AVR Butterfly Training

http://www.siwawi.arubi.uni-kl.de/avr_projects/Butterfly_training.pdf

Fecha de consulta: 25/03/2012

[3]. Escuela Superior Politécnica del Ejército

Características del Kit AVR Butterfly en español

http://www.espe.edu.ec/repositorio/T-ESPE-014271.pdf

Fecha de consulta: 27/03/2012

[4]. LPCxpresso 1769

http://www.microshadow.com/lw_lpcxpresso1769.php

Fecha de consulta: 30/03/2012

[5]. Módulo RF 4333

http://www.roboeq.com/PDF/0501018.pdf

Fecha de consulta: 7/04/2012

[6]. LPCxpresso 1769 guía para el usuario.

http://ics.nxp.com/support/documents/microcontrollers/pdf/lpcxpresso.getting.started.pdf

Fecha de consulta: 30/03/2012

[7]. AN10898 – BLDC Motors with LPC1700 ,

http://es.scribd.com/doc/70224846/BLDC-Fundamentals

Fecha de consulta: 3/03/2012.

[8]. Motor – Brushless DC (BLDC)

http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis89.pdf

Fecha de consulta: 3/03/2012

[9]. T. Kenjo, “Permanent Magnet and Brushless DC Motors”, Oxford, 1985.

Fecha de consulta: 12/03/2012

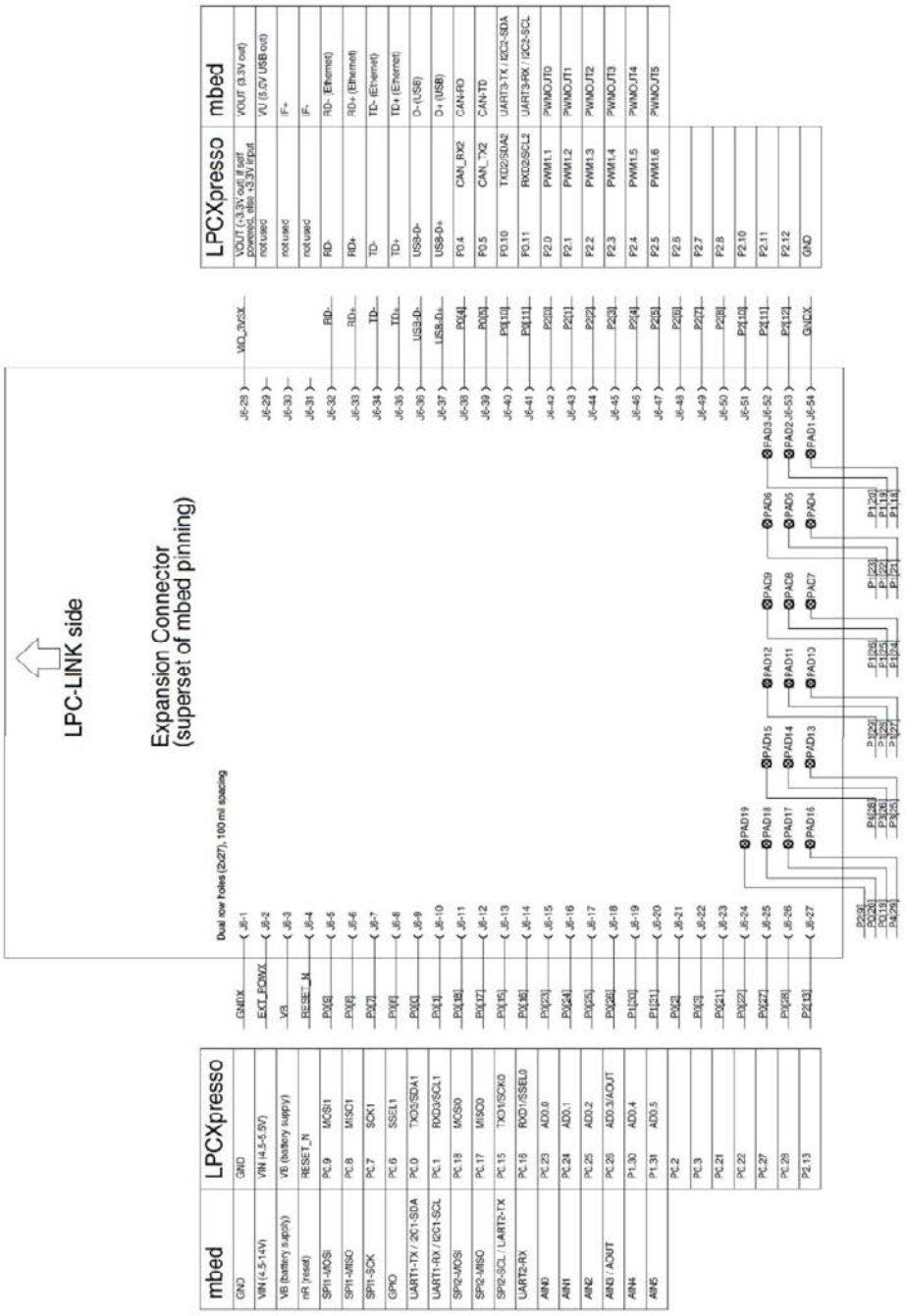
[10]. Técnicas de control para motores BLDC

http://www.ingeniamc.com/Es/-Control-techniques-for-brushless-motors.pdf

Fecha de consulta: 11/03/2012

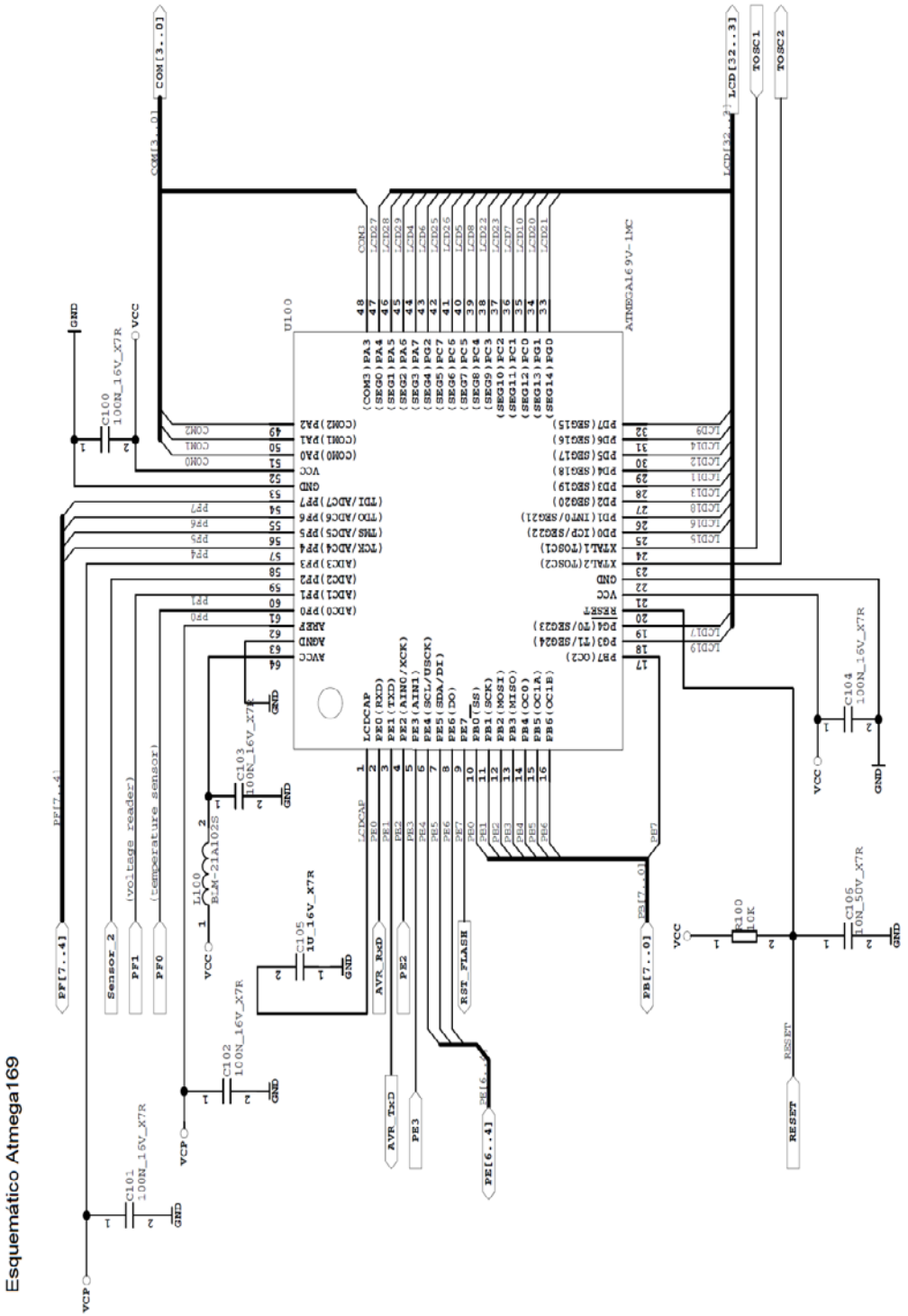
ANEXOS

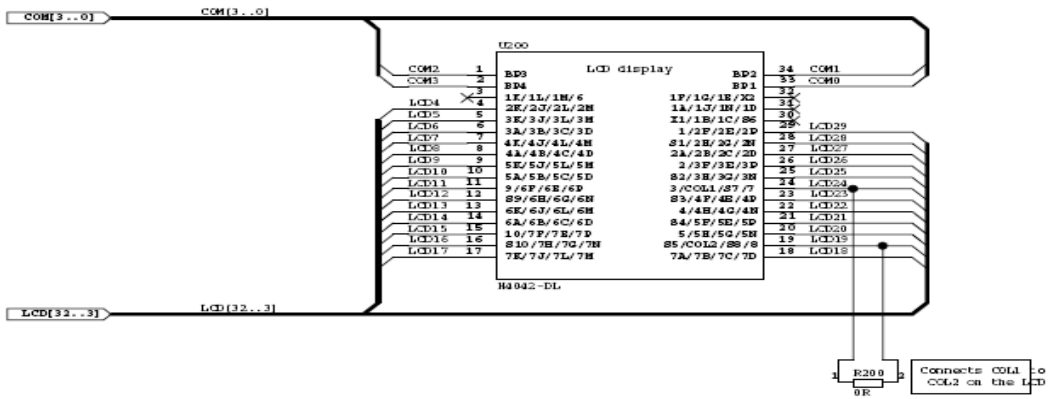
Esquemático de la tarjeta LPC1769

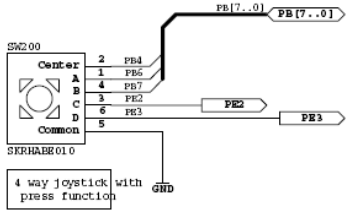


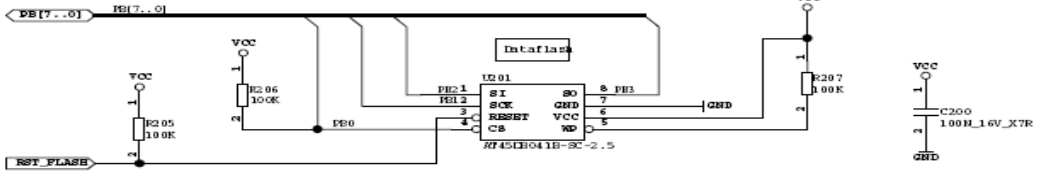
Esquemático del AVR Butterfly

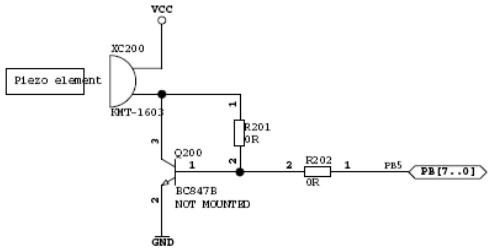
ATmega169

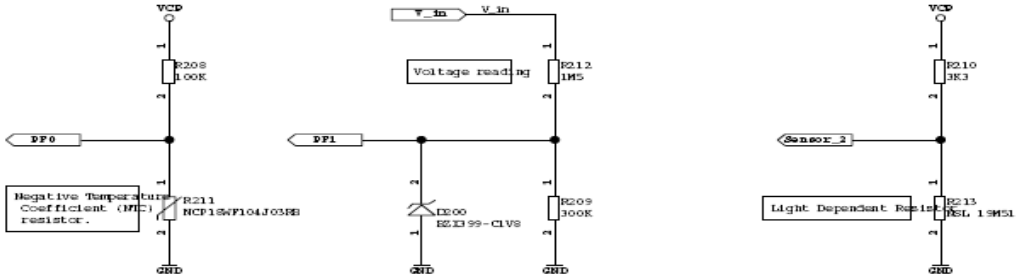


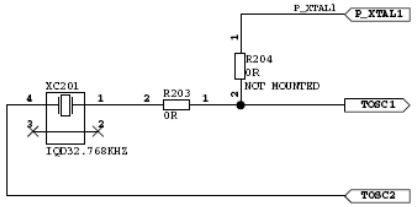

COM[3..0]
COM[3..0]
U200
LCD display
COM2 1
COM3 2
LCD4 4
LCD5 5
LCD6 6
LCD7 7
LCD8 8
LCD9 9
LCD10 10
LCD11 11
LCD12 12
LCD13 13
LCD14 14
LCD15 15
LCD16 16
LCD17 17
BP3
BP4
1K/1L/1M/6
2K/2J/2L/2M
3K/3J/3L/3M
3A/3B/3C/3D
4K/4J/4L/4M
4A/4B/4C/4D
5K/5J/5L/5M
5A/5B/5C/5D
9/6P/6E/6P
89/6E/6Q/6N
6K/6J/6L/6M
6A/6B/6C/6D
10/7P/7E/7D
810/7H/7Q/7N
7K/7J/7L/7H
BP2
BP1
1P/1Q/1E/X2
1A/1J/1N/1D
I1/1B/1C/86
1/2P/2E/2D
81/2H/2Q/2N
2A/2B/2C/2D
2/3P/3E/3D
82/3H/3Q/3N
3/COL1/87/7
83/4P/4E/4D
4/4H/4Q/4N
84/5P/5E/5D
5/5H/5Q/5N
85/COL2/88/8
7A/7B/7C/7D
34 COM1
33 COM0
29 LCD29
28 LCD28
27 LCD27
26 LCD26
25 LCD25
24 LCD24
23 LCD23
22 LCD22
21 LCD21
20 LCD20
19 LCD19
18 LCD18
H4042-DL
LCD[32..3]
LCD[32..3]
R200
0R
Connects COL1 to COL2 on the LCD

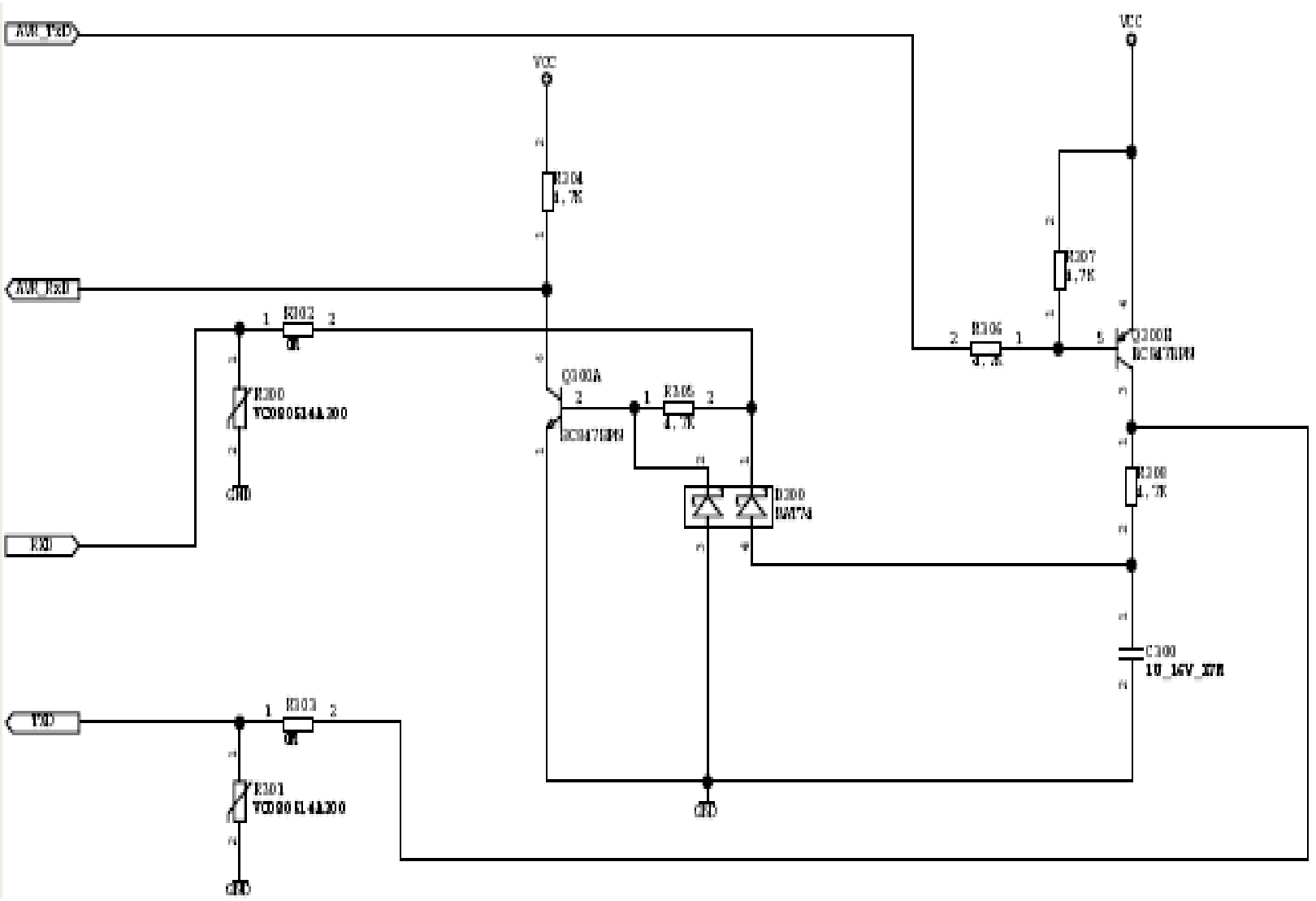

PB[7..0]
PB[7..0]
SW200
Center
A
B
C
D
Common
2 PB4
1 PB6
4 PB7
3 PE2
6 PE3
5
PE2
PE3
SKRHABE010
4 way joystick with press function
GND

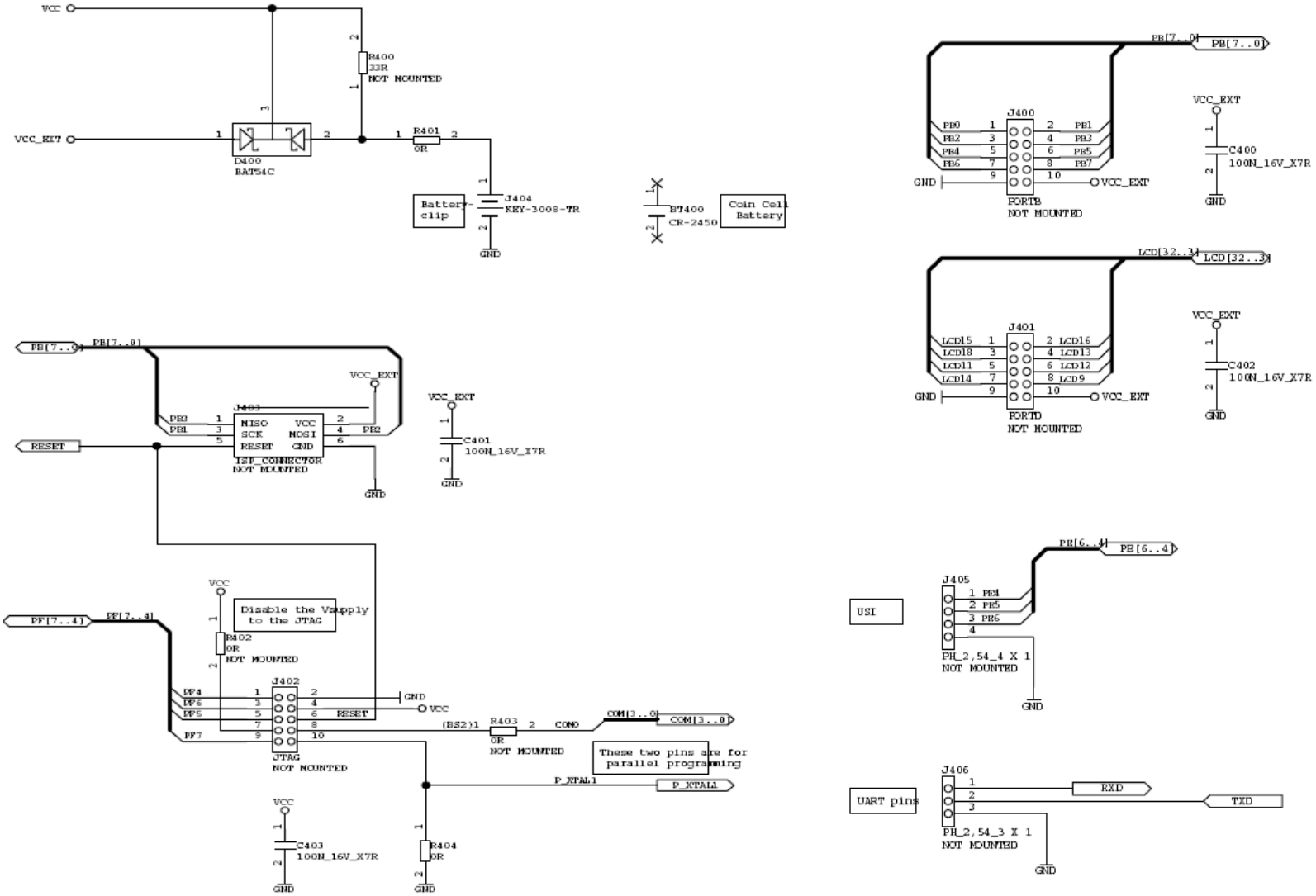

PB[7..0]
PB[7..0]
VCC
R205
100K
VCC
R206
100K
PB0
RST_FLASH
Dataflash
U201
SI
SCK
RESET
CS
SO
GND
VCC
WP
8 PB3
7
6
5
PB2 1
PB1 2
3
4
AT45DB041B-SC-2.5
GND
VCC
R207
100K
VCC
C200
100N_16V_X7R
GND


VCC
Piezo element
XC200
EMT-1603
R201
0R
Q200
BC847B
NOT MOUNTED
R202
0R
2
1
PB5
PB[7..0]
GND


VCC
R208
100K
DF0
Negative Temperature Coefficient (NTC) resistor.
R211
NCP18WF104J03RB
GND
V_in
V_in
Voltage reading
R212
1M5
DF1
D200
BZX99-C1V8
R209
300K
GND
GND
VCC
R210
3K3
Sensor_2
Light Dependent Resistor
R213
MSL 19M51
GND


P_XTAL1
P_XTAL1
R204
0R
NOT MOUNTED
XC201
IQD32.768KHZ
R203
0R
TOSC1
TOSC2

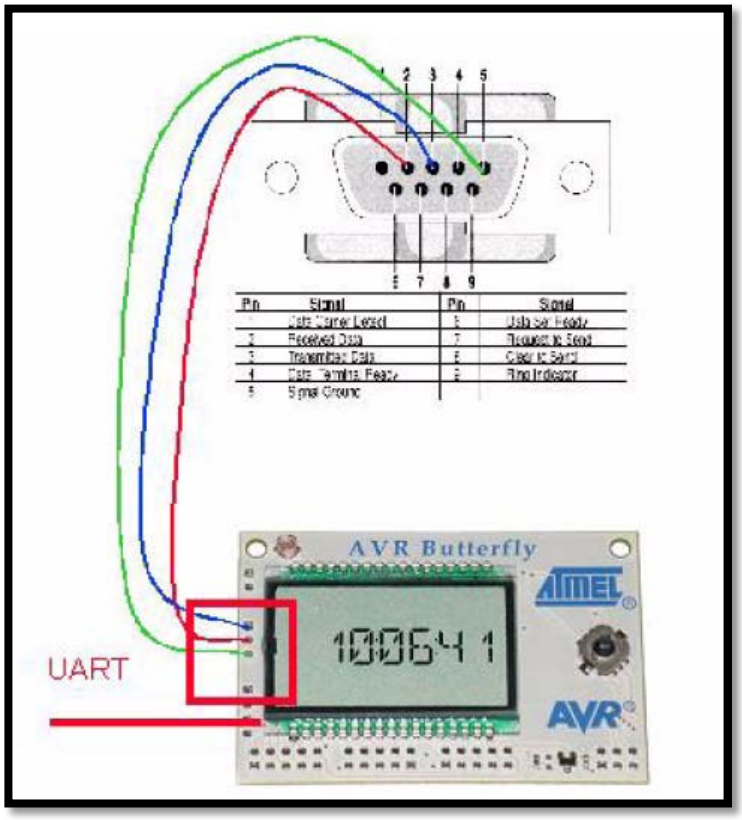

VCC
R304
4.7K
R307
4.7K
R306
4.7K
Q300B
R302
0R
R300
Q300A
R305
4.7K
D300
C300
10_16V_X7R
R303
0R
R301
GND


VCC
R400
33R
NOT MOUNTED
VCC_EXT
D400
BAT54C
R401
0R
J404
KEY-3008-TR
Battery clip
GND
BT400
CR-2450
Coin Cell Battery
PB[7..0]
J400
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PORTB
NOT MOUNTED
C400
100N_16V_X7R
LCD[32..3]
J401
LCD15
LCD16
LCD18
LCD13
LCD11
LCD12
LCD14
LCD9
PORTD
C402
RESET
J403
MISO
VCC
SCK
MOSI
RESET
ISP_CONNECTOR
C401
PE[6..4]
J405
PE4
PE5
PE6
PH_2,54_4 X 1
USI
PF[7..4]
R402
Disable the Vapply to the JTAG
J402
JTAG
R403
(BS2)1
CON0
COM[3..0]
These two pins are for parallel programming
P_XTAL1
C403
R404
J406
RXD
TXD
PH_2,54_3 X 1
UART pins

GUÍA PARA PROGRAMAR EL AVR BUTTERFLY

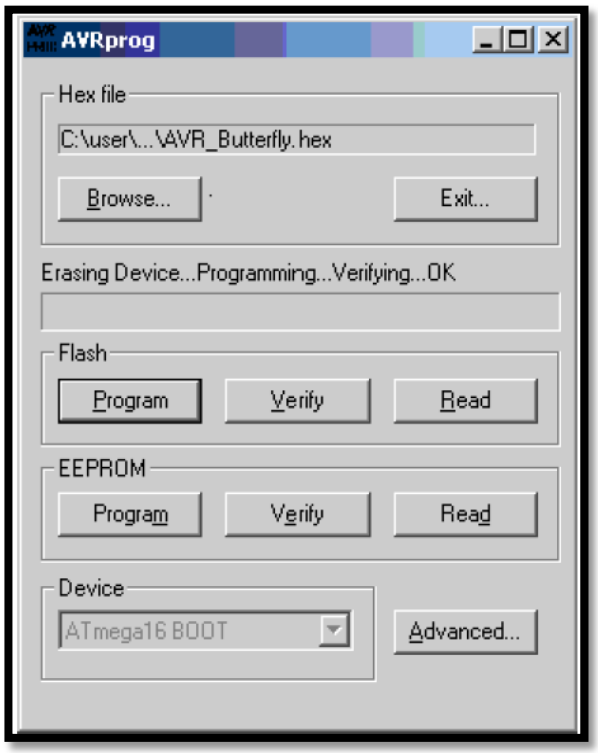
Para realizar la programación del AVR Butterfly con la PC, daremos a conocer al usuario el siguiente procedimiento:

1. Realizar la conexión del cable, para la interfaz serial RS-232 entre la PC y el AVR Butterfly, de la siguiente manera:



2. Ejecutar el software AVR Studio 4
3. Ejercer presión en el joystick del AVR Butterfly, en el centro y mantenerlo así.
4. Energizar el AVR Butterfly con una fuente de voltaje de 3 V

5. En AVR Studio 4, en la barra de herramientas, abrir el menú Tools. En este menú se visualiza la función AVRProg.



6. Quitar la presión que se mantiene sobre el joystick del AVR Butterfly
7. Clic en el botón Browser de la Ventana AVRprog, para localizar y cargar el archivo HEX generado con la compilación en el directorio del proyecto.
8. clic en el botón Program de la ventana AVRprog, para programar el microcontrolador ATmega169V del AVR Butterfly y actualizarlo con la nueva aplicación.
9. Clic el botón Exit de la ventana AVRprog, para salir del modo de programación.

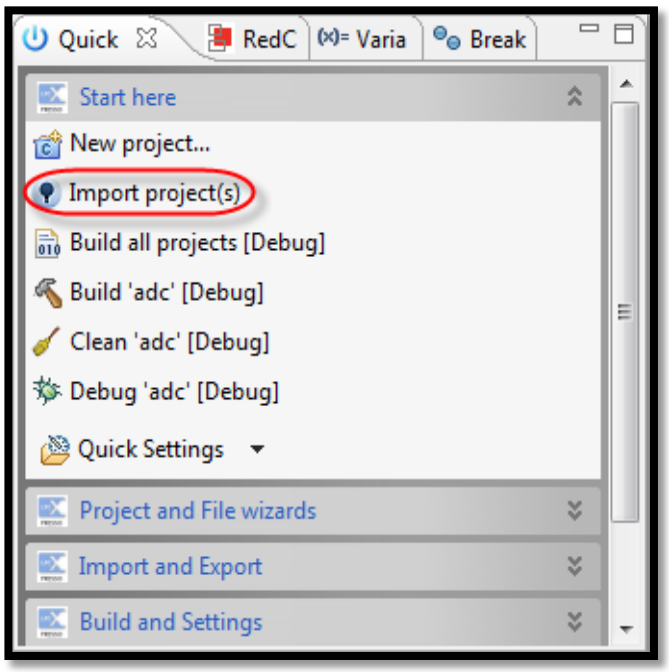
GUÍA PARA PROGRAMAR EL LPCXPRESSO.

Para realizar la programación de la tarjeta LPCxpresso con la PC, daremos a conocer al usuario el siguiente procedimiento mediante pasos sencillos para llevara cabo una correcta programación.

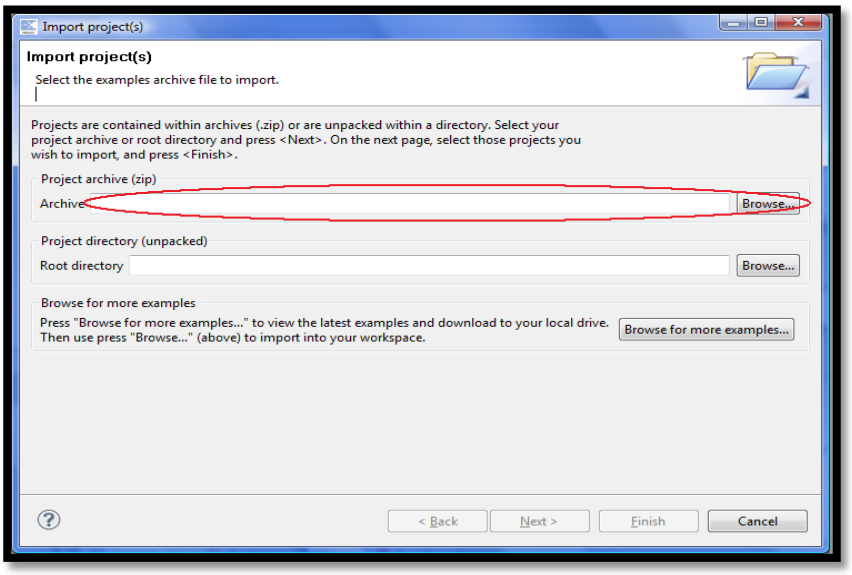
En nuestro caso, lo hicimos a partir de un ejemplo, así como se va indicar a continuación.

1. Para comenzar el desarrollo, la LPCXpresso se puede conectar a un PC mediante un puerto USB 2.0 A / Mini-B del cable.

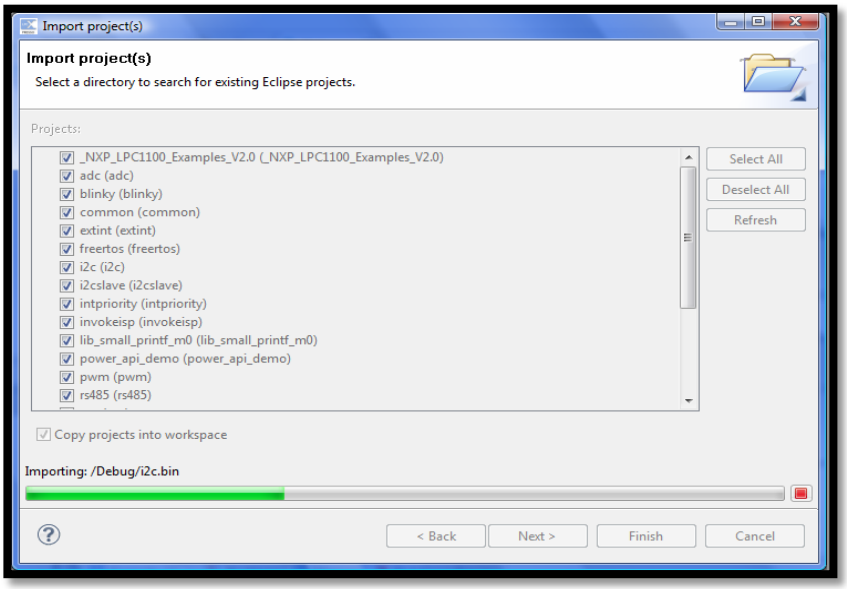
2. Para trabajar con un ejemplo de los que ofrece NXP, seleccione “Import project(s)” desde el panel Quickstart en la esquina inferior izquierda de la pantalla.



3. Seleccionamos Browser en Proyect Archive (zid), para poder acceder al directorio donde se encuentran por default los ejemplos que ofrece el programa.

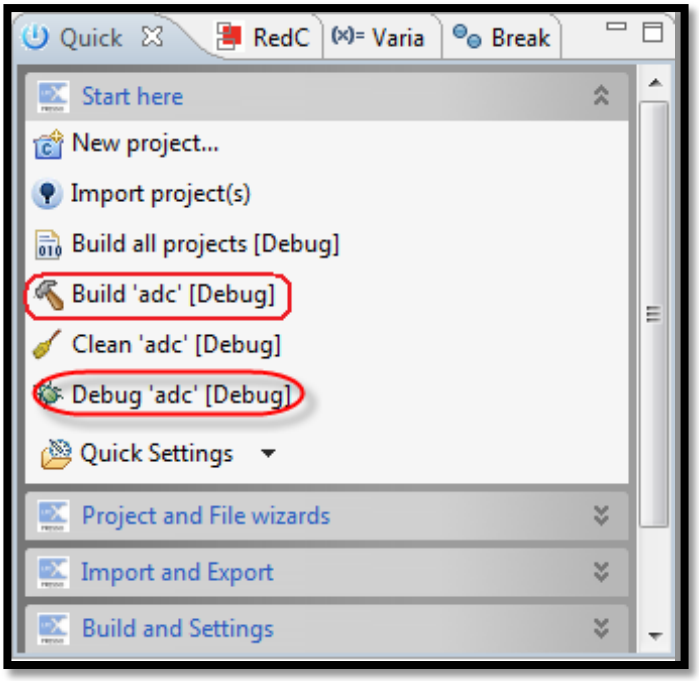


4. Seguido cuando ya se haya importado el archivo damos NEXT y aparecerá una pantalla para seleccionar el ejemplo una vez hecho es damos.

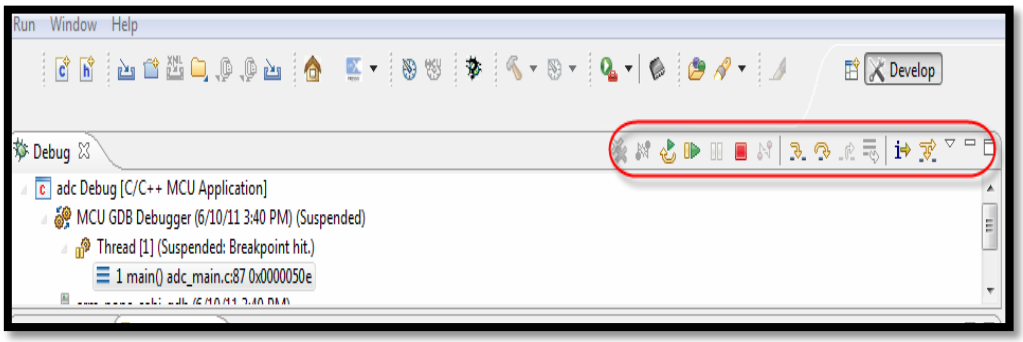


5. Finalmente en el panel Quickstart damos clic en “Built” y luego en “Debug”. En LPCXpresso, cuando se empieza a depurar, el programa

se descargará automáticamente a la tarjeta y es programado en la memoria flash.



6. Aquí se presenta la barra de herramientas para tener control sobre el código que se ejecuta.



HOJA DE DATOS DE LOS MÓDULOS RF

HM-TR Transparent Wireless Data Link Module

1. General

HM-TR series transparent wireless data link module is developed by Hope microelectronics Co. Ltd, dedicated for applications that needs wireless data transmission. It features high data rate, longer transmission distance. The communication protocol is self controlled and completely transparent to user interface. The module can be embedded to your current design so that wireless communication can be set up easily.

2. Features

1. FSK technology, half duplex mode, robust to interference
2. ISM band, no need to apply frequency usage license
3. Operation frequency can be configured and can be used in FDMA applications
4. Transmitting frequency deviation and receiver bandwidth can be

selected.

5. Protocol translation is self controlled, easy to use.

6. Data rate can be select from a wide range.

7. Provide ENABLE pin to control duty-cycle to satisfy different application requirements

8. High sensitivity, long transmission range.

9. Standard UART interface, TTL or RS232 logic level selectable

10. Very reliable, small size, easier mounting.

11. No tuning in producing.

3. Application

1.Remotecontrol,remote measurement system

2. Wireless metering

3. Access control

4. Identity discrimination

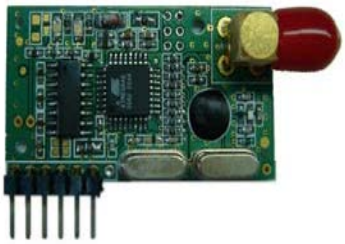
5. Data collection

6. IT home appliance

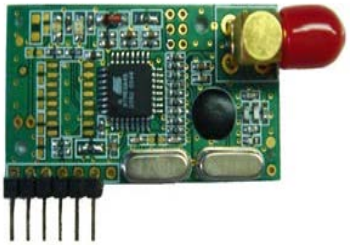
7. Smart house products

8. Baby monitoring

4. Mechanical appearance

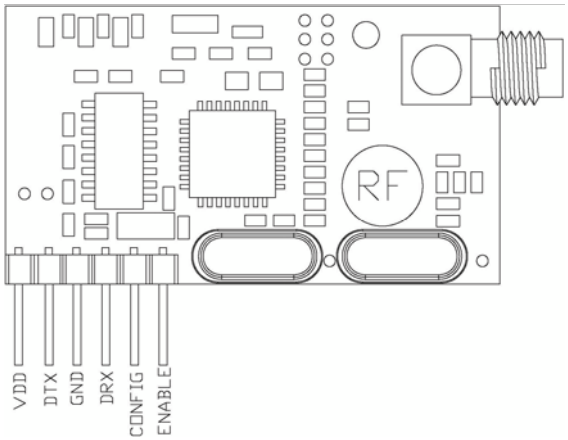


HM-TRXXX-232



HM-TRXXX-TTL

5. Pin definition



Pins	title	description
VCC	Power supply	+5V
DTX	Data	Module data transmission
DRX	Data receiving	Module data receiving
CONFIG	Configure mode	If CONFIG pin is high at power on, the module enter
ENABLE	Working funtion	If config pin is low at power on, the module will enter

6. Module parameters

Basic Parameters

Working	Description	Min.	Typ.	Max.	5V
Working temperature		-35	25	80	℃
Working frequency	4 standard frequency	310.24	-	929.27	MHz
power	Pmax depends on the specific frequency	Pmax-21	Pmax	Pmax	dBm
frequency deviation		15		240	kHz
Receiving bandwith		67		400	kHz
UART Baud		300	9600	19200	bps
UART data bit		5	8	9	bit
Check bit	No check or Parity				
Stop bit		1	1	2	bit
Antenna Connector					SMA (female)
Module size					24×43mm

Working frequencies

Module P/N	description	minimum	typical	maximum	Unit
HM-TR315		310.24	315	319.75	MHz
HM-TR433		430.24	434	439.75	MHz
HM-TR868		860.48	869	879.51	MHz
HM-TR915		900.72	915	929.27	MHz

Maximum transmission power

Module P/N	description	minimum	typical	maximum	Unit
HM-TR315			8		dB
HM-TR433			8		dB
HM-TR868			4		dB
HM-TR915			4		dB

Receiving sensitivity

Module P/N	description	minimum	typical	maximum	Unit
HM-TR315			-109	-100	dBm
HM-TR433			-109	-100	dBm
HM-TR868			-109	-100	dBm
HM-TR915			-109	-100	dBm

Working current in transmitting

Module P/N	description	minimum	typical	maximum	Unit
HM-TR315	TTL Output connect or			48	mA
HM-TR433				48	mA
HM-TR868				50	mA
HM-TR915				50	mA

Working current in receiving

Module P/N	description	minimum	typical	maximum	Unit
HM-TR315	TTL Output connect			34	mA
HM-TR433				34	mA
HM-TR868				36	mA
HM-TR915				36	mA

Static Current

Module P/N	description	minimum	typical	maximum	Unit
HM-TR315	TTL Output connect or		0.5	1	uA
HM-TR433			0.5	1	uA
HM-TR868			0.5	1	uA
HM-TR915			0.5	1	uA

Reliable communication distance

Module P/N	description	minimu	typica	maximu	Unit
HM-TR315	Tested in free open area by keeping the modules 1 meter above the ground			230	m
HM-TR433				330	m
HM-TR868				220	m
HM-TR915				230	m

7. Module application

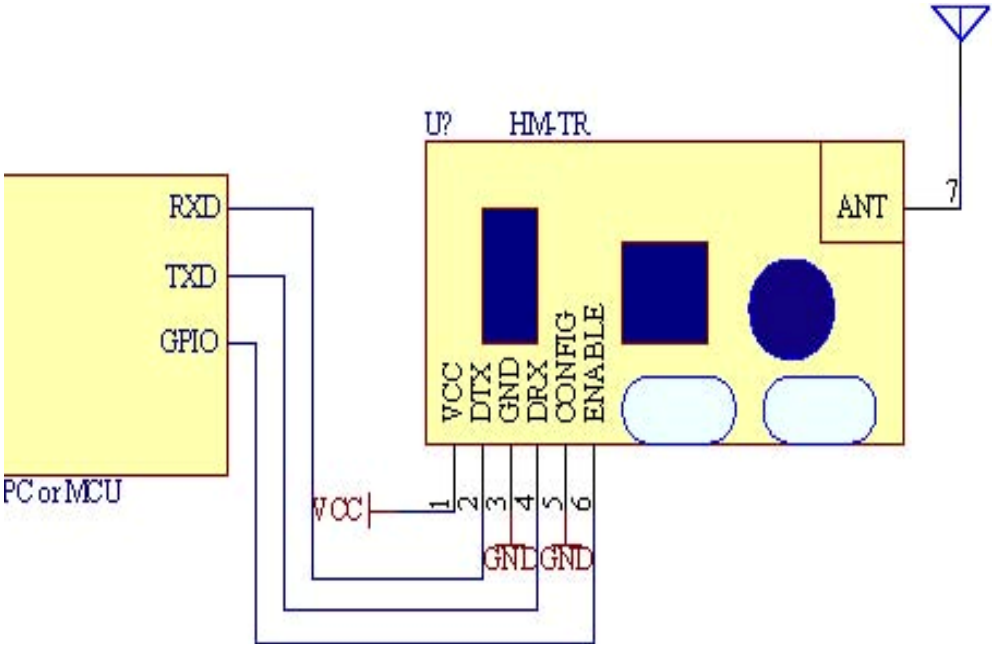
Module has two modes: communication mode and configure mode, it is determined by the status of CONFIG pin when power on:

CONFIG=LOW：It enter communication mode for data transmission

CONFIG=HI：It enter configure mode to setup work parameters

1．Communication mode

If CONFIG pin is low when powering on, the module will enter into communication mode. The module provide RS232 connector to connect with PC or TLL level with MCU directly



Communication Diagram

It can work properly with the default configuration (default configure is 9600，8，N，1）. the module work parameters can be set up via HM-TR setup tool.

When the serial data rate is below 9600bps，HM-TR module supports continuous transmission and the maximum data stream can reach 1000000bytes; however, the data transmitted each time should not exceed 32bytes in high-speed applications（>9600bps）.

HM-TR module work in half-duplex mode. When receiving 32 Bytes from the serial port, it will send data out at once. If the data package received is below 32 Bytes, the module will wait for about 30 ms and then send it. In order to send data immediately, 32 Bytes data per transmission is necessary.

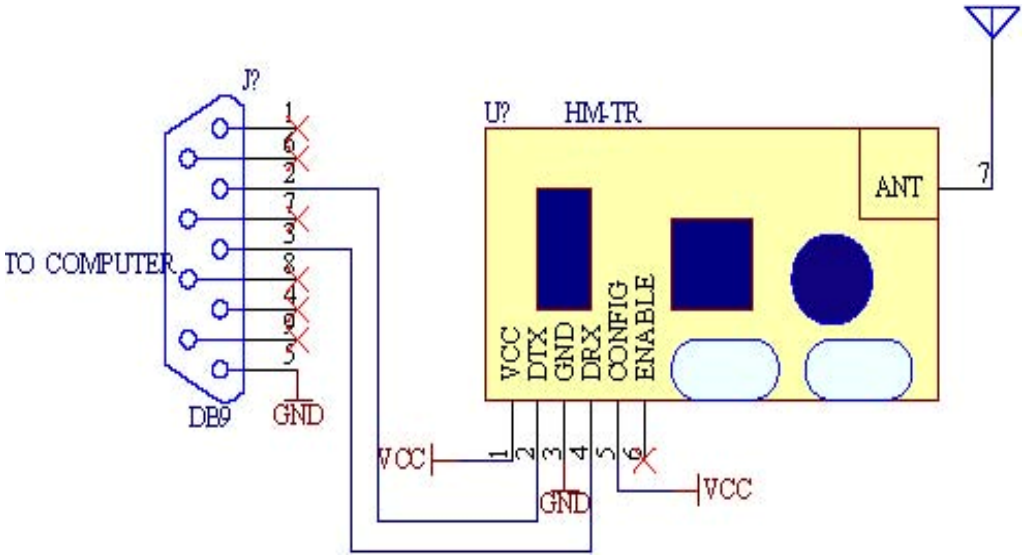
After each transmission, HM-TR module will be switched to receiver mode automatically. The switch time is about 5ms.

ENABLE pin is used to control the power consumption. Once this pin is pulled down, the module will enter into sleep mode immediately. Users can use this pin to control the receiving duty circle.

2．Configuration mode

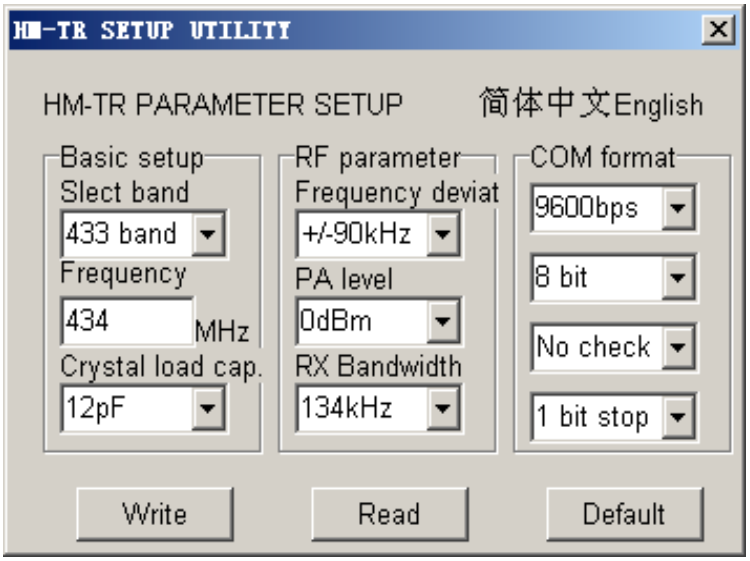
If the CONFIG pin is in high level when powering on, the module will enter into configuration mode automatically. In this mode the module

communicates with the host in fixed serial format（9600，8，N，1）



Configure mode connection

You can check the parameters of HM-TR and set up the parameters via HM-TR setup software below:

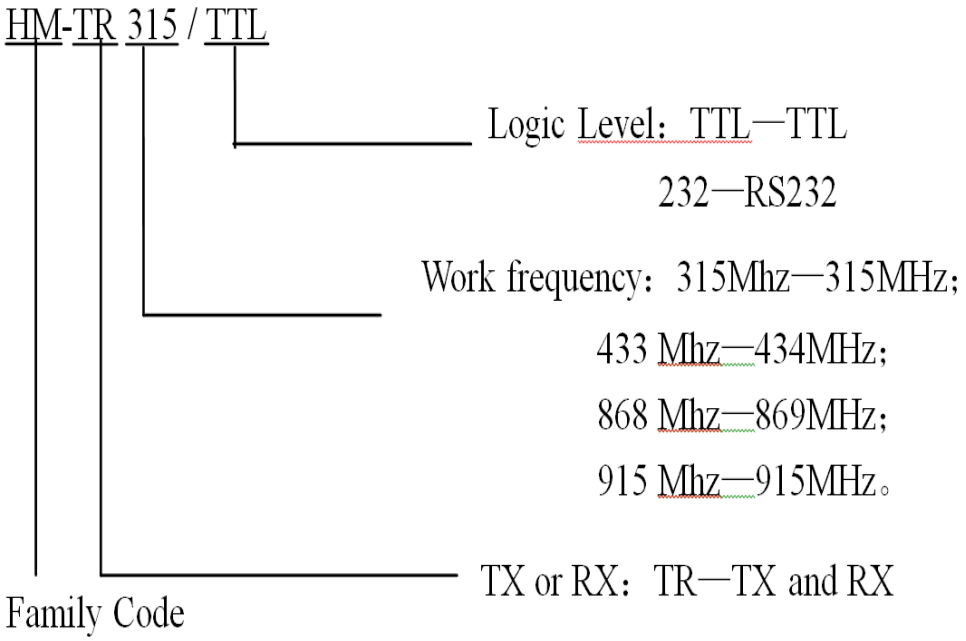


“Read” button: Read the parameters the module currently use; “Write” button: Write new configuration to module; “Default” button: Recover default value；

8. Ordering information

P/N	Logic Level
HM-TRxxx/TTL	TTL
HM-TRxxx/232	RS232

9. Module naming rule



10. Mechanical outline