ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"ALMACENAMIENTO DE DATOS DE TEMPERATURA DE MOTOR BLDC PARA GRAFICACIÓN Y ANÁLISIS EN DISPLAYS DISPONIBLES EN TARJETA AVR BUTTERFLY Y EN TARJETA CONTROLADORA LPCXPRESSO"

TESINA DE SEMINARIO

Previa a la obtención del título de:

INGENIERO EN ELECTRICIDAD ESPECIALIZACION ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

Presentado por:

Tamara Alexandra Samaniego Soto

Fernando Adrián Rodríguez González

GUAYAQUIL – ECUADOR

AÑO 2012

AGRADECIMIENTO

Doy gracias a Dios en primer lugar por haberme permitido culminar de manera exitosa mis estudios superiores. A mis padres por su apoyo incondicional. Y a mis maestros por sus conocimientos impartidos durante la carrera.

Tamara Samaniego Soto

Agradezco primeramente a Dios por permitirme haber alcanzado una meta más en mi vida. A mi madre por su apoyo incondicional. A mi esposa e hijo quienes son los principales motivos de lucha para seguir adelante. A mi ángel por ser la luz al final del camino.

Fernando Rodríguez González

DEDICATORIA

A mi familia por estar siempre brindándome esa confianza y apoyo.

Tamara Samaniego Soto

A mi madre por ser quien con esfuerzo y dedicación supo guiarme por el camino del saber y a mi esposa e hijo por esa confianza que siempre me brindan.

Fernando Rodríguez González

TRIBUNAL DE SUSTENTACIÓN

Ing. Carlos Valdivieso A.

PROFESOR DEL SEMINARIO DE GRADUACIÓN

Ing. Hugo Villavicencio

PROFESOR DELEGADO POR LA UNIDAD ACADÉMÍCA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de exámenes y títulos profesionales de la ESPOL)

_____

Tamara Samaniego Soto

_____

Fernando Rodríguez González

RESUMEN

El siguiente proyecto de graduación expone el diseño e implementación de un control de temperatura utilizando las tarjetas LPCXpresso y AVR Butterfly para así demostrar una de las diversas aplicaciones que tienen estas tarjetas.

En los párrafos siguientes describiremos el desarrollo de comunicación entre las tarjetas así como los análisis de las mediciones de temperatura realizados al motor Brushless.

Luego, detallaremos las herramientas tanto de hardware como de software que utilizaremos para desarrollar nuestro proyecto. Entre estas herramientas se encuentran el microcontrolador ATmega169 de la tarjeta AVR Butterfly, así como su respectiva configuración de puertos tanto de entrada como de salida, la pantalla LCD en la cual visualizaremos los datos de temperatura; la tarjeta LPCxpresso la cual nos ayudará con su convertidor análogo digital (ADC) a leer correctamente los datos de temperatura -(provenientes de un sensor)- de tal forma que luego los podamos enviar por medio del protocolo de comunicación I2C. Además de los equipos antes mencionados también utilizaremos un sensor

de temperatura LM35 el cual que estará midiendo la temperatura de un motor 42BLF01.

Finalmente los datos serán mostrados en el LCD que está disponible en la tarjeta AVR Butterfly para su respectivo análisis.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

GLOSARIO

AVR Advanced Virtual RISC, RISC Virtual Avanzado.

Code Red Empresa que proporciona herramientas de desarrollo innovador para habilitar sistemas integrados con microcontroladores de 32 bits.

Código fuente Conjunto de líneas las cuales son las instrucciones que se deben seguir o ejecutarse para realizar un tarea específica, estas instrucciones deber ser traducidas a lenguaje máquina por medio de un compilador para entenderse y ejecutarse en el microcontrolador.

Debugger Programa usado para probar y depurar códigos desarrollados en distintas plataformas de programación.

Embedded Artists Empresa que provee productos y servicios para plataformas que crean prototipos para sistemas

	embebidos basados en NXP, así como también colaboran con la construcción y elaboración de las tarjetas LPCXpresso.
Hardware	Partes o componentes físicos que integran una herramienta; inclusive ella misma como una unidad.
HEX	Archivo de tipo hexadecimal compuesto de registros que le especifican al microcontrolador datos para la memoria del programa.
JTAG	Joint Test Action Group. Usado para la norma ieee 1149.1-1990 llamada Standard Test Access Port and Boundary-Scan Architecture. Usada en los circuitos impresos tanto para el seteo así como para la depuración de la aplicación desarrollada en aquel circuito.
Microcontrolador	Computador de limitadas prestaciones que está contenido en el chip de un circuito integrado programable y que destina a gobernar una sola tarea con el programa que reside en su memoria. Sus

	líneas de entrada/salida soportan el conexionado de los sensores y actuadores del sistema a controlar.
Programador	Dispositivo electrónico que permite cargar un programa al microcontrolador.

INTRODUCCIÓN

Desde que nació la tecnología ha ido evolucionando continuamente, parte de esta evolución está enfocada en los microcontroladores que actualmente esta proyectándose en todos los campos donde puedan emplearse. Los microcontroladores en nuestro entorno nos ayudan entre tantas cosas a optimizar recursos de hardware.

El uso de los microcontroladores en este tipo de aplicaciones ha tomado gran fuerza en lo que respecta a la incorporación de lenguajes de alto nivel, además del bajo coste de adquisición de dichos dispositivos.

El propósito de nuestro proyecto es dar a conocer las diversas opciones que se tiene para realizar una comunicación I2C entre dos tarjetas que realizan trabajos diferentes así como también la lectura de un sensor mediante una de estas tarjetas, para lograr así que sean sencillos de manejar.

Para efectos prácticos y de aprendizaje esta tesina se ha dividido en 4 capítulos los cuales contendrán la información necesaria para su entendimiento.

El primer capítulo trata de la descripción general del proyecto en el cual encontraremos los antecedentes, la motivación que nos llevo a realizar este proyecto, que limitaciones podríamos tener, entre otras cosas.

En el capítulo 2 encontraremos información de los componentes y/o equipos que vamos a utilizar en el transcurso del desarrollo de nuestro proyecto.

En el capítulo 3 se detallaran parte de los ejercicios realizados previos a la elaboración del proyecto final con sus respectivos diagramas de bloques y flujo.

En el capítulo 4 se desarrollará de manera explicativa y visual la ejecución de los ejercicios mencionados en el capítulo anterior así como también la realización del proyecto final.

CAPÍTULO 1

DESCRIPCIÓN GENERAL DEL PROYECTO

1.1 ANTECEDENTES.

Desde la aparición de la automatización, el hombre no se ha detenido y sigue implementando controles automáticos. La primera aparición de estos controles fue de tipo análogos, pero a medida como iban pasando los años fue mejorando la tecnología digital y así es como surgen los microcontroladores y que mejor forma de explicarlos sino mediante un análisis de temperatura.

Hubo muchos investigadores para la realización de un sistema de control automático de temperatura. Entre ellos están: J. Kepler, J.J. Becher, entre otros.

Siendo Bonnemain allá por el año 1977 quien desarrolló un regulador conveniente de temperatura para el uso industrial el cual lo utilizó en una incubadora. Su dispositivo fue instalado más adelante en el horno de una planta de calefacción de agua caliente

En la actualidad los controladores de temperatura están en varios campos como por ejemplo: las refrigeradores domésticas, frigoríficos, etc.

El procesador ARM Cortex ™-M3, el primero de la nueva generación de procesadores Cortex de ARM, está centralizado en el mercado microcontrolador de 32 bits. El procesador Cortex-M3 proporciona un rendimiento excelente en el número de puertas bajas y viene con muchas nuevas características antes sólo estaban disponibles en procesadores de gama alta.

1.2 MOTIVACIÓN

La motivación principal para realizar este proyecto es implementar un sistema de comunicación y la medida de la temperatura de un motor que sea útil en el

campo de la ingeniería, y que utilice elementos electrónicos modernos y tecnologías actualizadas para ser más competitivos en nuestra área.

1.3 DESCRIPCIÓN DEL PROYECTO.

La realización de nuestro proyecto se basa en el almacenamiento de datos de temperatura de motores BLDC, así como también su graficación y respectivo análisis en displays los cuales están disponibles en la tarjeta AVR Butterfly, siendo esta tarjeta de la familia ATMEL.

Gran parte de los elementos que se necesitan para desarrollar este proyecto vienen incorporados en la tarjeta, tales como: el microcontrolador ATmega169, la pantalla LCD para la visualización de las medidas. Para adquirir los datos de temperatura utilizaremos un sensor LM35 cuya lectura será realizada con la tarjeta LPC1769, con la cual trabajaremos en modo MASTER y mediante el protocolo de comunicación I2C se transmitirán las datos al dispositivo ESCLAVO que en nuestro caso será la tarjeta AVR Butterfly.

1.4 LIMITACIÓN OPERACIONAL.

Debido a que el motor no puede ser forzado para aumentar su temperatura, las pruebas de nuestro proyecto se las realizaron externamente sensando la temperatura de un cautín.

1.5 SITUACIÓN ACTUAL.

Hoy en día y con los avances tecnológicos, los controladores digitales de temperatura han reemplazado parcialmente a los controladores analógicos sin desmerecer su amplio campo de aplicación. Los argumentos para cambiar de tecnología son el bajo costo de los circuitos digitales y el reducido espacio que estos ocupan.

Un sistema analógico tiene mucho más limitaciones que los sistemas digitales; por ejemplo, un sistema analógico debe respetar unos criterios de calidad que afectan a la transmisión de la señal. Como la señal transmitida debe ser una réplica análoga de la señal original, es necesario que esta forma no se distorsione.

Es importante que tengamos en cuenta el hecho que la mayoría de las veces los sensores de temperatura son utilizados en hospitales, salas de recuperación e incluso en quirófanos, ya que debemos tener en cuenta que la temperatura del ambiente tiene que ser ideal evitando que la misma interfiera de algún modo con la recuperación del paciente. Cabe recalcar que quienes más utilizan estos controladores de temperatura son los neonatos.

Fig. 1.1 Incubadora Rarosch2. [3]

Los sensores de temperatura son de gran ayuda hoy en la actualidad debido a que no se establecen en un campo específico sino que se lo puede instalar en distintos sitios dependiendo de su rango de temperatura y sobre todo de su uso.

Por otro lado, una desconcertante variedad de proveedores, dispositivos y arquitecturas está compitiendo en el mercado de los microcontroladores. El requisito para microcontroladores de mayor rendimiento ha sido impulsado globalmente por el cambio de la industria necesidades, por ejemplo, en la actualidad, los microcontroladores son necesarios para manejar más trabajo sin aumentar la frecuencia de un producto o de alimentación. Además, los microcontroladores son cada vez más conectado, ya sea por bus serie universal

(USB), Ethernet o inalámbrica de radio, y por lo tanto, el tratamiento necesario para apoyar a estos canales de comunicación y periféricos avanzados están creciendo. Del mismo modo, la complejidad de aplicación general va en aumento, impulsada por las interfaces de usuario más sofisticadas, los requisitos de multimedia, la velocidad del sistema, y la convergencia de funcionalidades.

CAPÍTULO 2

BASES TÉCNICAS DEL PROYECTO

Para poder llevar a cabo el desarrollo de nuestro proyecto es necesario haber conocido el funcionamiento de los equipos a utilizarse. A continuación daremos a conocer en forma resumida cada equipo usado en este proyecto. Lo dividiremos en 2 partes: herramientas de software y herramientas de hardware.

2.1 HERRAMIENTAS DE SOFTWARE.

Los programas con los cuales vamos a trabajar son los siguientes: LPCXpresso, AVR Studio 4, Proteus.

2.1.1 LPCXPRESSO

LPCXpresso es una nueva plataforma de desarrollo de bajo costo disponible para NXP. Es compatible con microcontroladores basados en ARM LPC de NXP. La plataforma está compuesta por un IDE basado en eclipse y tiene una tarjeta de bajo costo que incluye un depurador JTAG.

Diseñado para la simplicidad y facilidad de uso, el LPCXpresso IDE (powered by Code Red) proporcionará a los ingenieros de software una forma rápida y sencilla de desarrollar sus aplicaciones. Las tarjetas LPCXpresso son desarrolladas en conjunto por Embedded Artists, Code Red y NXP. LPCXpresso ha lanzado al mercado dos tarjetas: LPC1114, es la primera tarjeta que tiene en integrado un Cortex M0; otra de las tarjetas es la LPC1343 con un dispositivo USB 2.0 de alta velocidad.

2.1.2 AVR STUDIO 4

Fig. 2.1.: Logo de la herramienta de software AVR Studio 4.

AVR Studio 4 es un Entorno de Desarrollo Integrado (IDE) para escribir y depurar aplicaciones AVR en el entorno de Windows. Soporta varias de las fases por las cuales se atraviesa al crear un nuevo producto basado en un microcontrolador AVR.

Este software ayuda al programador en el diseño, desarrollo, depuración y parte de la comprobación del proceso. Es actualizado continuamente y está disponible para descargarlo desde la siguiente página web “www.atmel.com”. Además tiene una arquitectura modular completamente nueva que incluso permite interactuar con software de otros fabricantes.

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

2.1.3 PROTEUS

Proteus es un software de diseño electrónico desarrollado por Labcenter Electrónicas que consta de dos módulos: Ares e Isis y que incluye un tercer módulo opcional denominado Electra.

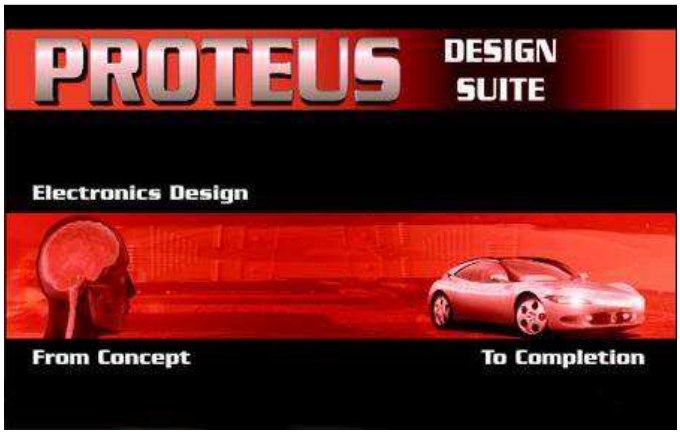


Fig. 2.2.: Logo de la herramienta de software PROTEUS 7.7

2.1.3.1 ISIS

Mediante este programa podemos diseñar el circuito que deseemos con componentes muy variados, desde una simple resistencia hasta algún que otro microprocesador o microcontrolador, incluyendo fuentes de alimentación, generadores de señales y muchas otras prestaciones. Los diseños realizados en Isis pueden ser simulados en tiempo real. Una de estas prestaciones es VSM,

una extensión de la aplicación con la cual podremos simular, en tiempo real, todas las características de varias familias de microcontroladores, introduciendo nosotros mismos el programa que queramos que lleven a cabo. Se pueden simular circuitos con microcontroladores conectados a distintos dispositivos, como motores, lcd´s, displays, etc. El módulo VSM incluye, entre otras, las familias PIC10, PIC12, PIC16, PIC18, PIC24 y dsPIC33. ISIS es el corazón del entorno integrado PROTEUS. Es mucho más que un simple programa de dibujo de esquemas electrónicos. Combina un entorno de diseño de una potencia excepcional con una enorme capacidad de controlar la apariencia final de los dibujos.

ISIS es la herramienta ideal para una rápida realización de complejos diseños de esquemas electrónicos destinados tanto a la construcción de equipos electrónicos como a la realización de tareas de simulación y prueba.

Además, encontrará en ISIS una herramienta excepcional para la realización de atractivos esquemas electrónicos destinados a su publicación en libros, manuales o documentos técnicos.

2.1.3.2 ARES

Ares es la herramienta de rutado de Proteus, se utiliza para la fabricación de placas de circuito impreso, esta herramienta puede ser utilizada de manera

manual o dejar que el propio programa trace las pistas, aunque aquí podemos también utilizar el tercer módulo, Electra (Electra Auto Router), el cual, una vez colocados los componentes trazará automáticamente las pistas realizando varias pasadas para optimizar el resultado. Con el módulo Ares también podemos tener una visualización en 3D del PCB que se ha diseñado.

2.2 HERRAMIENTAS DE HARDWARE

Las partes tangibles de nuestro proyecto serán las tarjetas de desarrollo tanto de la familia de NXP como de la familia ATMEL. Estas son: LPCXpresso que tiene integrado un chip LPC1769 y la Butterfly AVR que está formado por un microcontrolador ATmega169PV. Además haremos uso de los motores brushless (sin escobillas) para realizar la medición de temperatura.

2.2.1 LPC 1769

La LPCXpresso es una tarjeta de evaluación completa para el desarrollo con microcontroladores de NXP.

Contiene:

- LPCXpresso IDE y "development tools".
- IDE basado en Eclipse.
- Compiler y linker GNU.

- GDB debugger.
- LPCXpresso target board (stick).
- BaseBoard o hardware adicional (opcional).

|El target board es un microcontrolador con todo lo necesario para encender y también es una herramienta que incluye un debugger y un programador.

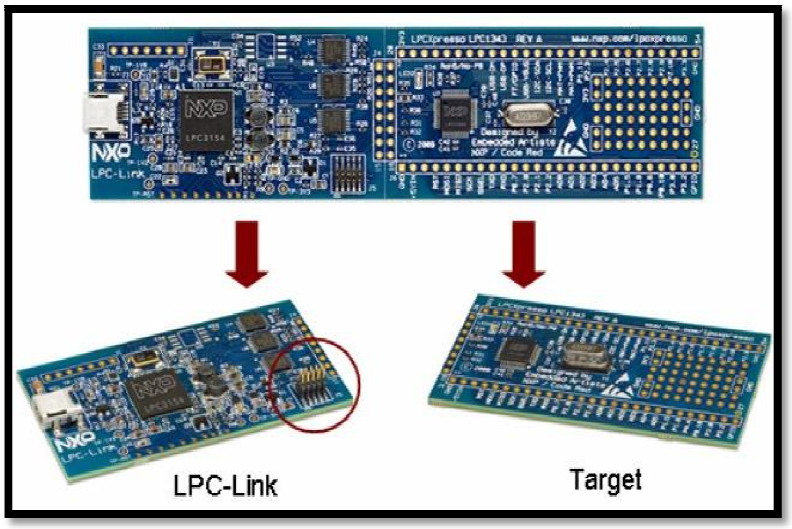


Fig. 2.3: Tarjeta LPCXpresso. [4]

Del lado de la tarjeta están incluidos algunos periféricos básicos y se comercializan con diferentes tipos de microcontroladores. Pero en nuestro caso usaremos el siguiente:

- LPC1769: ARM Cortex-M3, 512KB flash, 64KB SRAM,

Ethernet, USB On the go.

La placa contiene periféricos para desarrollo y experimentación:

Generales:[5]

- Socket para LPCXpresso y un módulo mbed.
- Pequeña batería de alimentación en forma de moneda.
- Interface USB.
- Pulsador de Reset.

Analógicos:

- Potenciómetro trimmer para entrada analógica.
- Salida PWM y entrada analógica.
- Salida para speaker (salida PWM).
- Etapa de entrada para probador de osciloscopio.

Digitales:

- Led RGB (puede ser controlado con PWM)
- 5-key joysticks switch.
- 2 pulsadores, uno para activar el bootloader.
- Sensor de temperatura con salida PWM.

- Interruptor giratorio con la codificación de cuadratura.

Serial – SPI:

- Registro de desplazamiento conductor de led de 7 segmentos.
- SD/MMC interface de tarjeta de memoria.
- DataFlash SPI-NOR.

Serial – I2C.

- Puerto expandido PCA9532 conectado a 16 leds.
- 8kbit E2PROM.
- Sensor de luz
- MMA7455L acelerómetro con interface I2C.

2.2.2 Butterfly AVR

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

El AVR Butterfly utiliza el microcontrolador AVR ATmega169PV, que combina la Tecnología Flash con el más avanzado y versátil microcontrolador de 8 bits disponible. En la siguiente figura se puede apreciar el Kit AVR Butterfly.

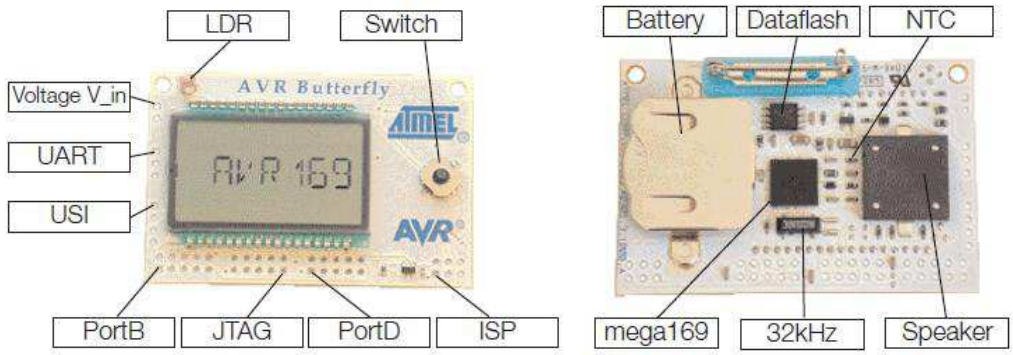


Fig. 2.4: Hardware disponible en el Kit AVR Butterfly. [6]

2.2.2.1 CARACTERÍSTICAS DE LA TARJETA AVR BUTTERFLY

El Kit AVR Butterfly expone las siguientes características principales:

- La arquitectura AVR en general y la ATmega169 en particular.
- Diseño de bajo consumo de energía.
- El encapsulado tipo MLF.
- Periféricos:
  - ✓ Controlador LCD
  - ✓ Memoria Interfaces de comunicación:
    - o UART, SPI, USI.

- Métodos de programación
  - ✓ Self-Programming/Bootloader, SPI, Paralelo, JTAG.
- Convertidor Analógico Digital (ADC).
- Timers/Counters:
  - ✓ Contador de Tiempo Real (RTC).
  - ✓ Modulación de Ancho de Pulso (PWM).

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

Los siguientes recursos están disponibles en el Kit AVR Butterfly:

- Microcontrolador ATmega169V (en encapsulado tipo MLF).
- Pantalla tipo vidrio LCD de 120 segmentos, para demostrar las capacidades del controlador de LCD incluido dentro del ATmega169.
- Joystick de cinco direcciones, incluida la presión en el centro.
- Altavoz piezoeléctrico, para reproducir sonidos.
- Cristal de 32 KHz para el RTC.
- Memoria DataFlash de 4 Mbit, para el almacenar datos.

- Convertidor de nivel RS-232 e interfaz USART, para comunicarse con unidades fuera del Kit sin la necesidad de hardware adicional.
- Termistor de Coeficiente de Temperatura Negativo (NTC), para sensar y medir temperatura.
- Resistencia Dependiente de Luz (LDR), para sensar y medir intensidad luminosa.
- Acceso externo al canal 1 del ADC del ATmega169, para lectura de voltaje en el rango de 0 a 5 V.
- Emulación JTAG, para depuración.
- Interfaz USI, para una interfaz adicional de comunicación.
- Terminales externas para conectores tipo Header, para el acceso a periféricos.
- Batería de 3 V tipo botón (600mAh), para proveer de energía y permitir el funcionamiento del AVR Butterfly.
- Bootloader, para programación mediante la PC sin hardware especial.
- Aplicación demostrativa preprogramada.
- Compatible con el Entorno de Desarrollo AVR Studio 4.

2.2.2.2 MICROCONTROLADOR ATMEGA169

El ATmega169 es un Microcontrolador de 8 bits con arquitectura AVR RISC, este posee las siguientes características: [7]

- Arquitectura RISC avanzada.
  - ❖ Conjunto de 130 instrucciones ejecutables en un solo ciclo de reloj.
  - ❖ 32 x 8 registros de trabajo de propósito general.
  - ❖ Rendimiento de hasta 16 MIPS a 16 MHz.
- Memoria no volátil para Programa y Datos.
  - ❖ Flash de 16 K bytes, auto-programable en el sistema.
  - ❖ Resistencia: 10 000 ciclos de Escritura/Borrado.
  - ❖ Sección Opcional para Código de Arranque con Lock Bits Independientes.
  - ❖ Programación en el Sistema a través del Programa de Arranque residente en el chip.
  - ❖ Operación Real de Lectura Durante la Escritura.
  - ❖ EEPROM de 512 bytes.
  - ❖ Resistencia: 100 000 ciclos de Escritura/Borrado.

- ❖ SRAM Interna de 1 Kbyte.
- ❖ Bloqueo de Programación para Seguridad del Software.

- Interfaz JTAG (conforme el Standard IEEE 1149.1)

  - ❖ Capacidad de Boundary-Scan Acorde al Standard JTAG.
  - ❖ Soporta Depuración On-chip.
  - ❖ Programación de la FLASH, EEPROM, Fusibles y Lock Bits a través de la Interfaz JTAG.

- Características de los Periféricos.

  - ❖ 6 puertos de I/O de 8-bits y 1 de 5-bits.
  - ❖ Controlador de LCD de 4 x 25 segmentos.
  - ❖ Dos Temporizadores/Contadores de 8 bits con Preajustador (Prescaler) separado y Modo de comparación.
  - ❖ Un Temporizador/Contador de 16 bits con Preajustador (Prescaler) separado, Modo de Comparación y Modo de Captura.
  - ❖ Contador de Tiempo Real con Oscilador Separado.
  - ❖ Cuatro canales PWM.
  - ❖ Ocho canales ADC de 8 bits cada uno.

- ❖ Interfaz Serial USART Programable.
- ❖ Interfaz Serial SPI Maestro/Esclavo.
- ❖ Interfaz Serial Universal con Detector de Condición de Inicio.
- ❖ WatchDogTimer Programable con Oscilador Separado incluido en el chip.
- ❖ Comparador Analógico.
- ❖ Interrupción y Salida del Modo de Sleep por Cambio en Pin.

- Características especiales del microcontrolador.

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

- Entradas/Salidas y Tipo de Encapsulado

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

- Rangos de Velocidad

  - ATmega169V: 0–4 MHz a 1.8–5.5 V, – 8 MHz a 2.7–5.5 V
  - ATmega169: 0–8 MHz a 2.0–5.5 V, 0–16 MHz a 4.5–5.5 V

- Rango de Temperatura

  - Desde -40 º C a 85 ºC.

- Consumo de Energía

  - En el Modo Activo:
    - 1 MHz, 1.8 V: 350 uA.
    - 32 KHz, 1.8 V: 20 uA (incluyendo Oscilador).
    - 32 KHz, 1.8 V: 40 uA (incluyendo Oscilador y LCD)

  - En el Modo Power-Down:
    - 0.1uA a 1.8 V.

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

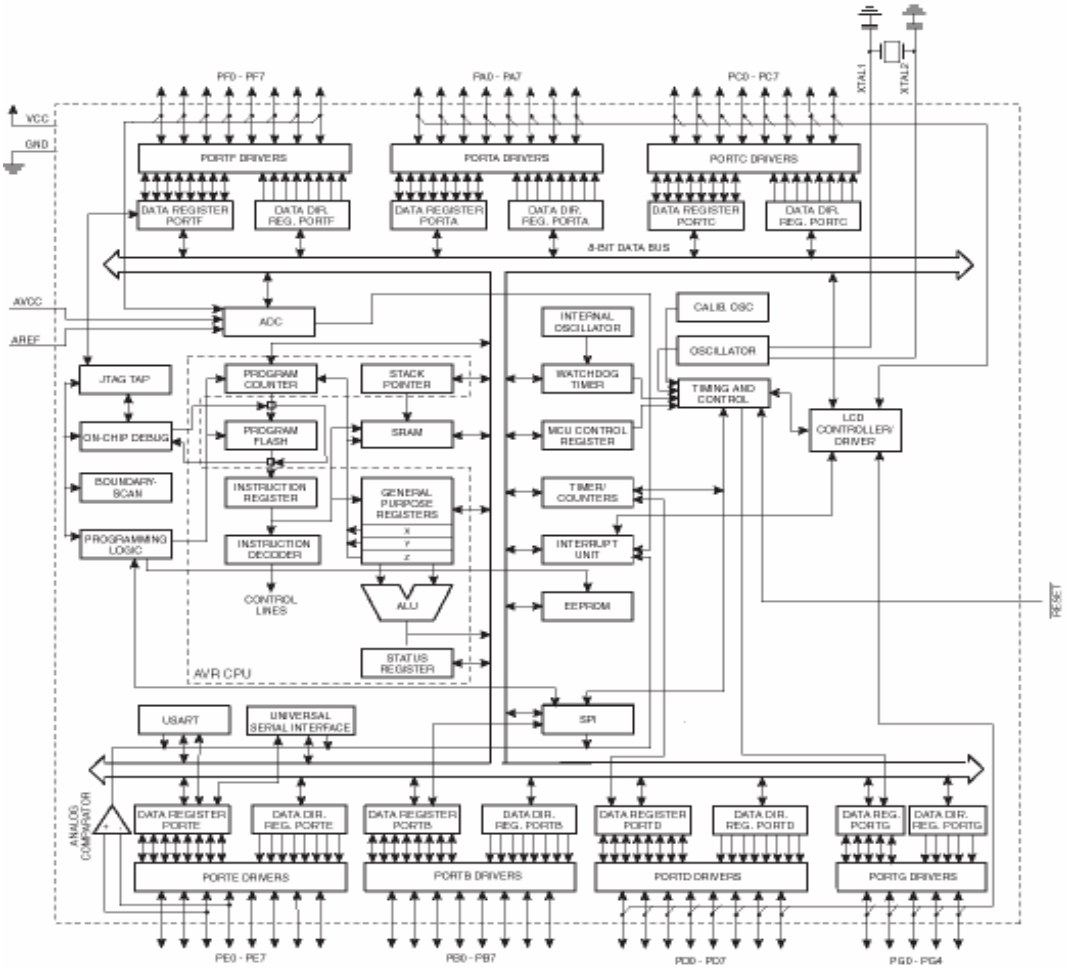


Fig. 2.5: Diagrama de bloques del microcontrolador ATmega169. [8]

El núcleo AVR combina un rico conjunto de instrucciones con 32 Registros de Trabajo de Propósito General (General Purpose Working Registers). Todos los 32 registros están conectados directamente a la Unidad de Lógica Aritmética (ALU), permitiendo que se acceda a dos registros independientes en una sola

instrucción ejecutada en un ciclo de reloj. La arquitectura da como resultado código más eficiente mientras que el rendimiento alcanzado es diez veces más rápido que los convencionales microcontroladores CISC.

Este dispositivo se fabrica empleando tecnología Atmel de memoria no volátil de alta densidad. La Flash ISP en el Chip permite que la memoria de programa se reprograme en el sistema a través de una interfaz serial SPI, por un programador convencional de memoria no volátil ó por un programa de Arranque (BootProgram) residente en el Chip ejecutándose en el núcleo del AVR. El programa de Arranque puede usar cualquier interfaz (SPI, USART, UART) para descargar el programa de la aplicación en la memoria Flash de Aplicación. El Software en la sección Flash de Arranque continuará ejecutándose mientras la sección Flash de Aplicación esté actualizándose, proporcionando la operación real de Lectura durante la Escritura.

Al combinar una CPU RISC de 8 bits con una Flash Auto programable en el Sistema sobre un chip monolítico, el ATmega169 de Atmel se convierte en un microcontrolador poderoso que provee una solución altamente flexible y de efectividad de costo para muchas aplicaciones de control integrado.

2.2.2.3 CONFIGURACIÓN DE LOS PINES DEL MICROCONTROLADOR ATMEGA169.

En la figura 2.4 se ilustra la distribución de pines del microcontrolador ATmega169V, en éste se aprecia el encapsulado MLF de 64 pines.

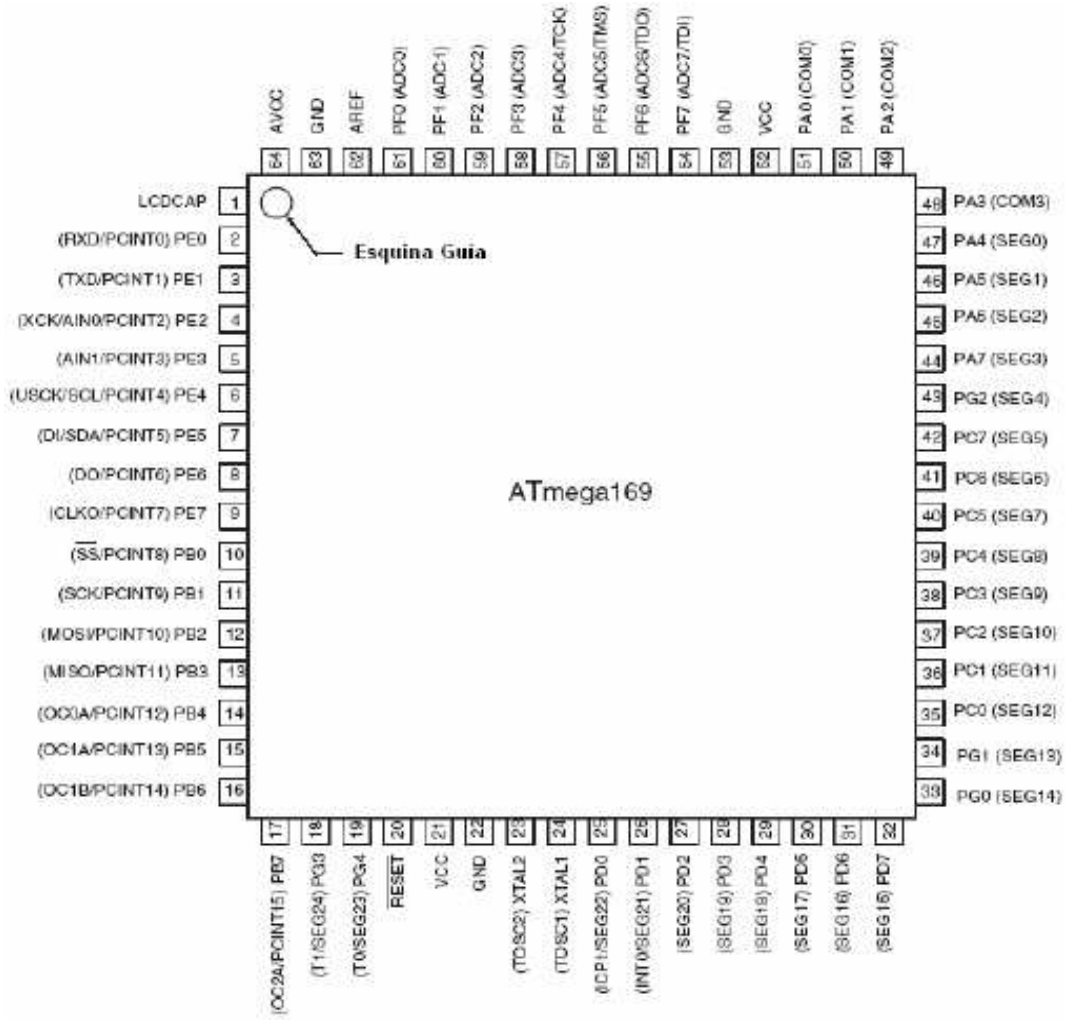


Fig. 2.6: Distribución de pines del microcontrolador ATmega169V. [8]

2.2.2.4 JOYSTICK

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

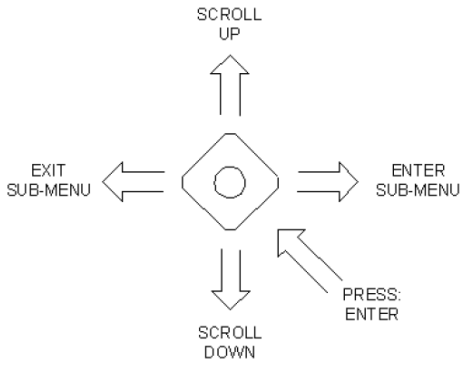


Fig. 2.7: Entrada tipo Joystick. [6]

2.2.2.5 CONECTORES

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

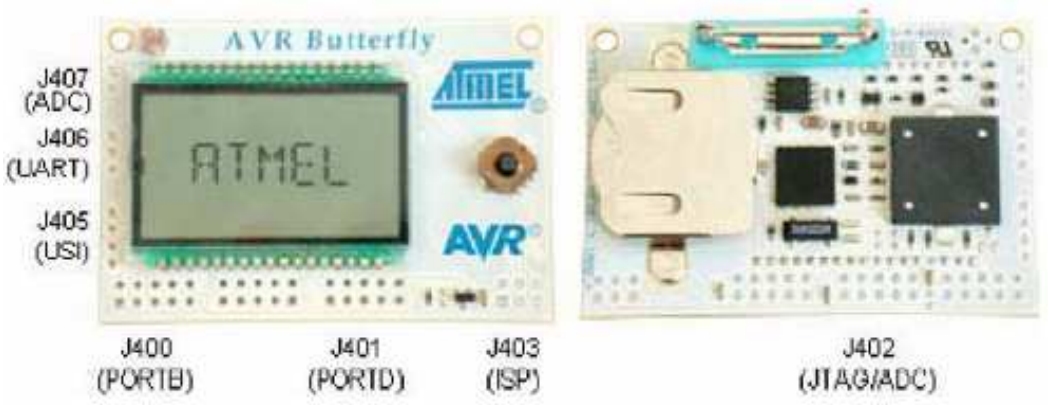


Fig. 2.8: Conectores del AVR Butterfly para acceso a periféricos. [6]

2.2.2.6 EL LCD

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

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

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

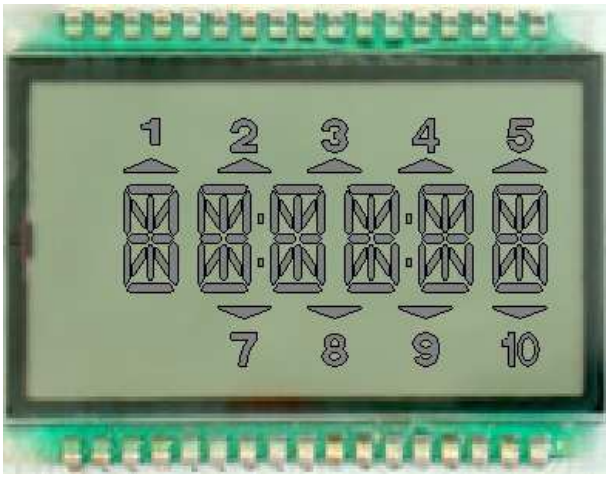


Fig. 2.9: Pantalla LCD. [9]

2.2.3 SENSOR DE TEMPERATURA LM35

Para poder sensar la temperatura de la planta haremos uso del integrado LM35, el cual posee un factor de escala lineal de $10.0\,{}^{mV}\!/_{^\circ C}$ Ahora, para obtener una relación temperatura – voltaje del sensor, de la siguiente manera:

$$\begin{aligned} 1000mV \quad & a \quad 100^\circ C \\ 600mV \quad & a \quad 60^\circ C \\ 300mV \quad & a \quad 30^\circ C \end{aligned}$$

El fabricante propone la siguiente configuración para calibrar el sensor:

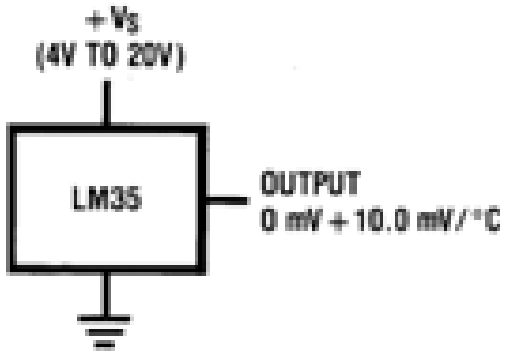


Fig. 2.10: Configuración del sensor de temperatura (+2ºC a +150ºC). [10]

2.3 PROTOCOLO DE COMUNICACIÓN I2C

2.3.1 Definición

Es un bus serial de comunicaciones compuesto únicamente de dos líneas: SDA y SCL. Su nombre viene de *Inter-Integrated Circuit* (Inter-Circuitos Integrados).

Este protocolo fue diseñado en 1992 por la compañía Philips, y este es muy utilizado hasta la actualidad en la industrial principalmente para comunicar microcontroladores o microprocesadores con otros dispositivos que permitan este protocolo, pero de preferencia lo utilizan entre microcontroladores.

2.3.2 Funcionamiento del I2C

Como habíamos mencionado el protocolo de comunicación I2C está compuesto por 2 líneas las cuales son drenador abierto por lo cual es necesario la utilización de resistencias pull-up, es decir que estas lineas en reposo estan a nivel alto.

Los nombres específicos de estas líneas son SDA y SCL.

SDA: es la linea por la cual se transmitiran los datos hacia el otro dispositivo.

SCL: es la señal de reloj

En este protocolo se especifica a cada uno de los dispositivos que se comunican sea como maestro o como esclavos.

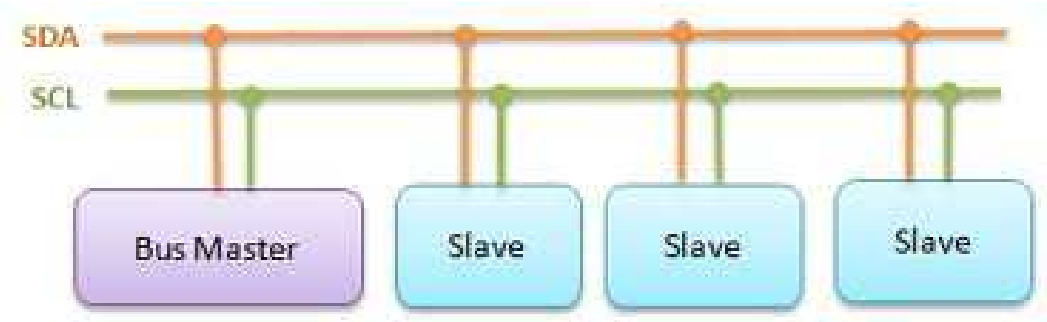


Fig. 2.11: Esquema protocolo I2C

Los diversos dispositivos que se interconectan al bus I2C tienen una dirección única para cada uno de ellos, así el dispositivo que hace de maestro los reconoce y puede transmitir la información.

Aunque en el protocolo sea necesario un maestro los demás elementos o microcontroladores conectados no serán solo esclavos pueden ser maestros así como esclavos, claro uno a las vez ya que no pueden haber 2 maestros al mismo tiempo, este tipo de configuración hace que se diga que el bus es multimaestro.

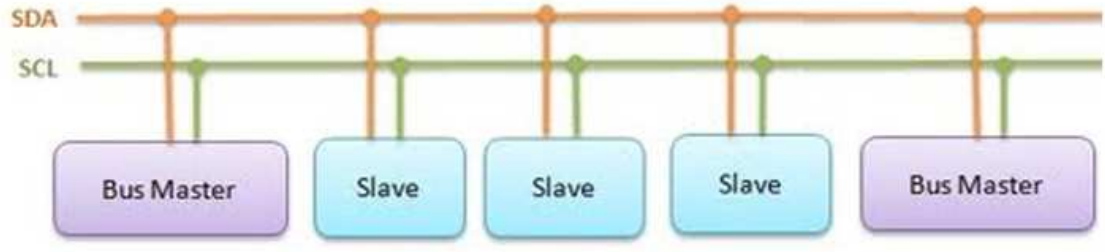


Fig. 2.12: Topología Multimaestro

2.3.3 Transmisión de Datos del protocolo I2C

Para realizar la transmisión hay que tener en cuenta que los bits de datos serán enviados por la SDA, para la transmisión de cada bit es necesario un pulso de reloj es decir que exista un cambio en la SCL. Para realizar un cambio de bit en la transmisión el estado de SCL debe ser en bajo.

Para iniciar la transmisión es necesaria la activación del bus es decir el envío del START, esto pondrá en alerta a todos los dispositivos esclavos que se encuentran conectados al bus I2C.

La condición de START es básicamente que el SDA este en bajo mientras el SCL está en alto.

Debemos recordar que el maestro es quien controla la señal SCL.

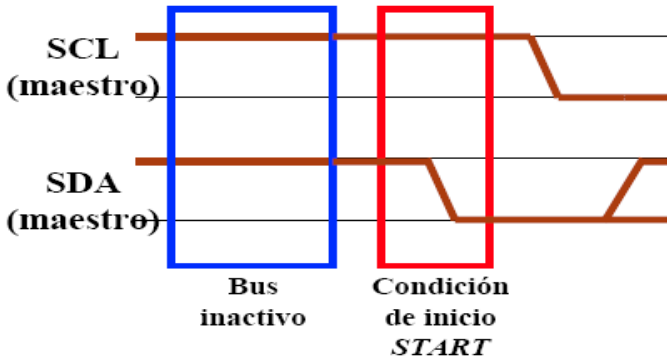


Fig.2.13: Condición START.

Luego de la señal de START el dispositivo maestro envía una dirección, de esta manera se podrá identificar al esclavo con el cual se comunicará, así solamente este esclavo podrá contestar ante las señales del maestro.

Luego de la dirección y la respuesta del esclavo se transmite el byte, cuando se recibió un byte el dispositivo esclavo envía un acknowledge lo que representa un bajo de la señal SDA.

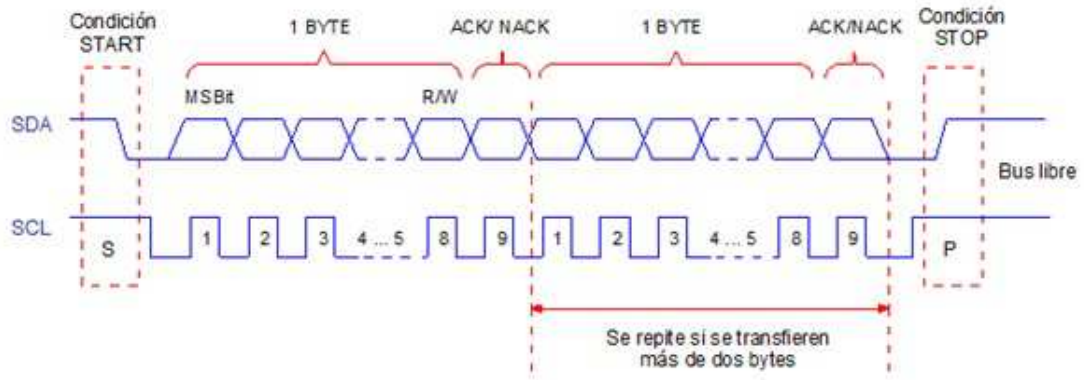


Fig. 2.14: Esquema de la Transmisión de datos

Para finalizar la transmisión es necesario la condición de STOP la cual es de la siguiente manera:

SDA: pasa a estado alto mientras el SCL se encuentra en alto.

Recordemos que mientras la señal de SCL se encuentra en alto no hay transmisión de datos.

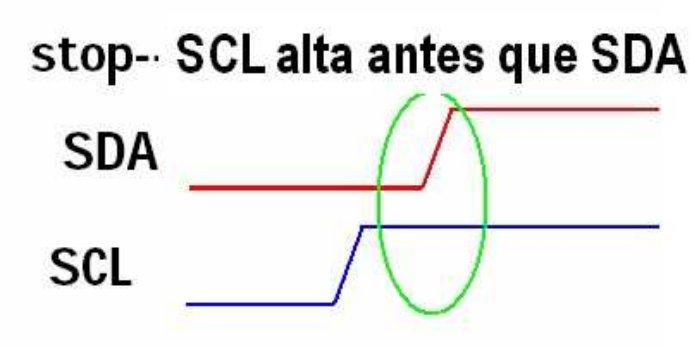


Fig.2.15: Condición STOP

2.4 MOTORES BRUSHLESS

Los motores de corriente alterna asíncronos fueron los primeros motores brushless. Los motores BLDC vienen siendo comerciales aproximadamente desde 1962. Los motores brushless (en español, motores sin escobillas) son motores que no emplean escobillas, están distribuidos a lo largo del estator en múltiples fases, conocidos también como motores de conmutación electrónica, trifásicos de alto rendimiento y bajo peso. Estos motores también pueden alimentarse directamente con corriente continua.

Esta señal trifásica idealmente seria sinusoidal, pero en realidad son pulsos, debido a estos pulsos la señal se vuelve continua pulsante (continua con muchos componentes AC)

Al ser estos motores trifásicos, entre las líneas presentan un desfase de 120°. En estos motores brushless las corrientes y voltajes aplicados deben ser controlados de forma independiente con una conmutación electrónica. Para esto existe un dispositivo al cual se lo denomina controlador del motor.

La relación existente entre Corriente-Torque y Frecuencia-Velocidad con los motores BLDC es que son lineales.

Existen dos tipos de motores brushless:

- Inrunner, su torque máximo lo tiene a muy altas revoluciones por lo que son usadas junto con reductoras.

- Outrunner, su torque máximo lo tiene a baja velocidad van directamente a la hélice. Lo que gira en este tipo de motores brushless es la parte exterior, donde los imanes están pegados quedando los bobinados fijos.

2.4.1 FUNCIONAMIENTO PRINCIPAL

Los motores BLDC se caracterizan por no tener escobilla (brushless) para el cambio de polaridad lo cual es conocido como conmutación que es para la

transferencia de energía, en estos motores se usa una conmutación electrónica que se realiza sin contacto.

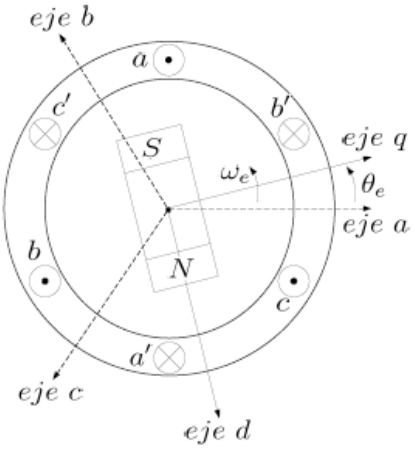
Como es una conmutación electrónica esta proporciona a la vez diversa información como es la velocidad de giro (revoluciones), si esta en movimiento o esta frenado o parado incluso se puede proteger el motor de tal forma que si está parado cortar la corriente para que éste no sufra ningún tipo de daño como es el quemarse.

2.4.2 CONSTRUCCIÓN DE LOS MOTORES BRUSHLESS (MB)

Su construcción es similar a la de cualquier motor: posee un estator, un rotor y una carcasa.

- ✓ El estator contiene varias bobinas por cada devanado de fase, distribuidas a su alrededor.
- ✓ El rotor está formado por uno o varios imanes permanentes con la intención de generar el campo magnético del rotor.
- ✓ La carcasa cumple con el objetivo de proteger contra el medio ambiente y la corrosión al motor.

La estructura del motor brushless es la que se muestra a continuación:



$$w_e = 2\pi f \quad donde\ f\ es\ la\ frecuencia$$

Fig. 2.16: Estructura del motor brushless. [11]

2.4.3 TÉCNICAS DE CONTROL PARA MOTORES BRUSHLESS

Se clasifican dependiendo del algoritmo de conmutación implementado.

Actualmente se utilizan las tres que vamos a mencionar:

- *Control basado en conmutación trapezoidal*, también conocida como 6-steps mode o basada en sensores HALL.

  Este esquema se encarga de controlar 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.

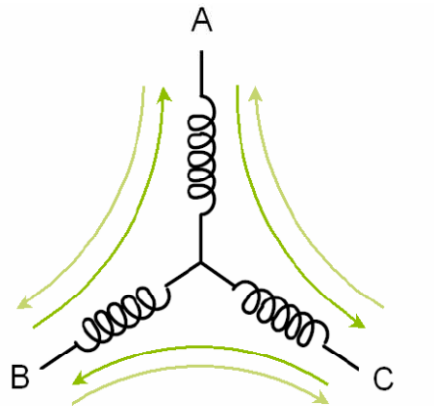


Fig. 2.17: SEIS POSIBLES CAMINOS DE CIRCULACIÓN DE CORRIENTE. [11]

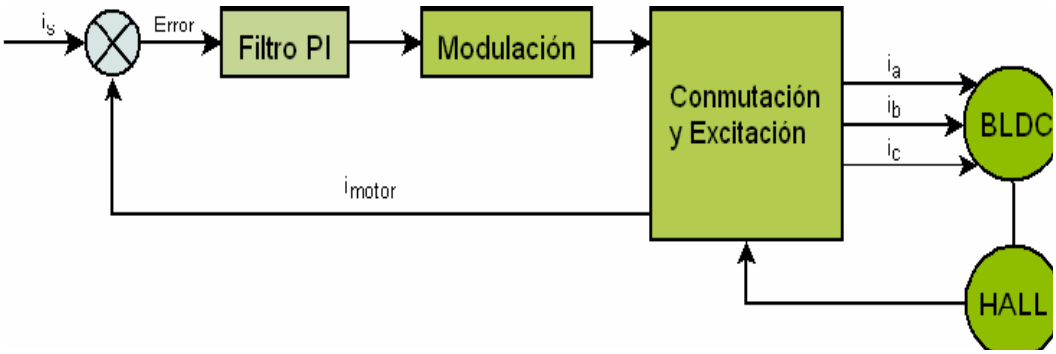


Fig. 2.18: ESQUEMA DE UN CONTROLADOR CON CONMUTACIÓN TRAPEZOIDAL. [11]

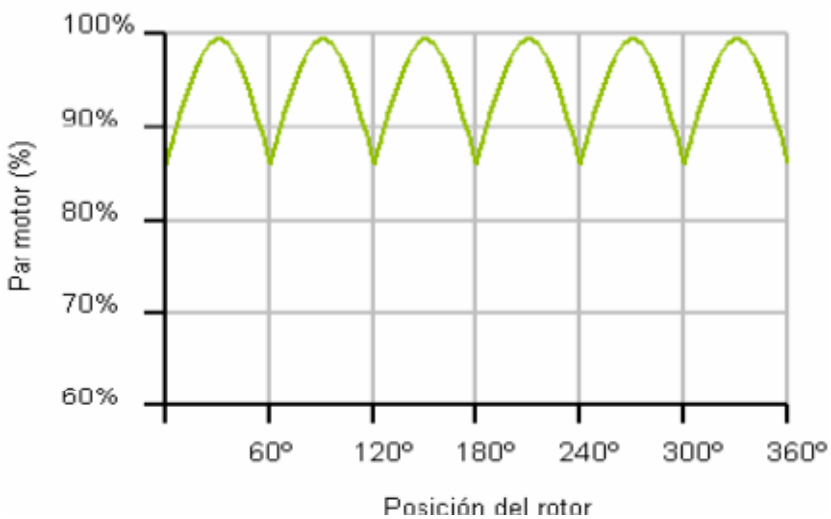


Fig. 2.19: RIZADO DEL PAR MOTOR RESPECTO A LA POSICIÓN DEL ROTOR EN UNA CONMUTACIÓN TRAPEZOIDAL. [11]

- *Control basado en conmutación sinusoidal*, intenta controlar la posición del rotor continuamente por lo que se la ve como un control más avanzado y exacto que el trapezoidal.

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

El principal problema que presenta esta conmutación 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.

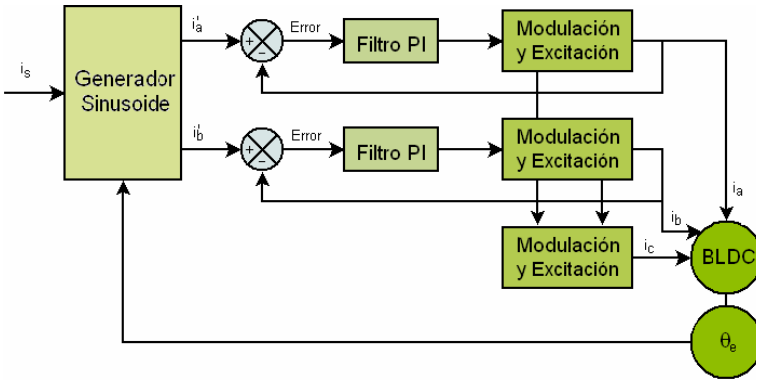


Fig. 2.20: ESQUEMA DE UN CONTROLADOR CON CONMUTACIÓN SINUSOIDAL. [11]

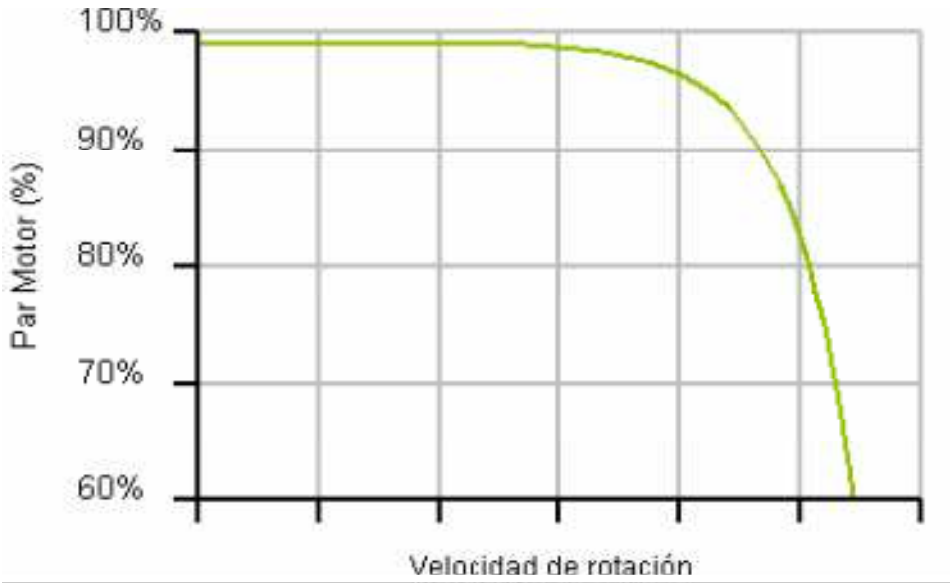


Fig. 2.21: PAR MOTOR EN FUNCIÓN DE LA VELOCIDAD DE ROTACIÓN. [11]

- *Control vectorial*, es el más complejo y el que requiere mayor potencia de cálculo de las tres técnicas. A su vez es la que mejor control proporciona.

  Este control soluciona el inconveniente que se genera en el control sinusoidal controlando el vector de corrientes directamente en un espacio de referencia ortogonal y rotación, llamado espacio D-Q (Direct-Quadrature).

  Dicho espacio de referencia esta normalmente alineado con el rotor de forma que permite que el control de flujo y del par del motor se

realice de forma independiente. La componente directa permite controlar el flujo y la componente en cuadratura el par.

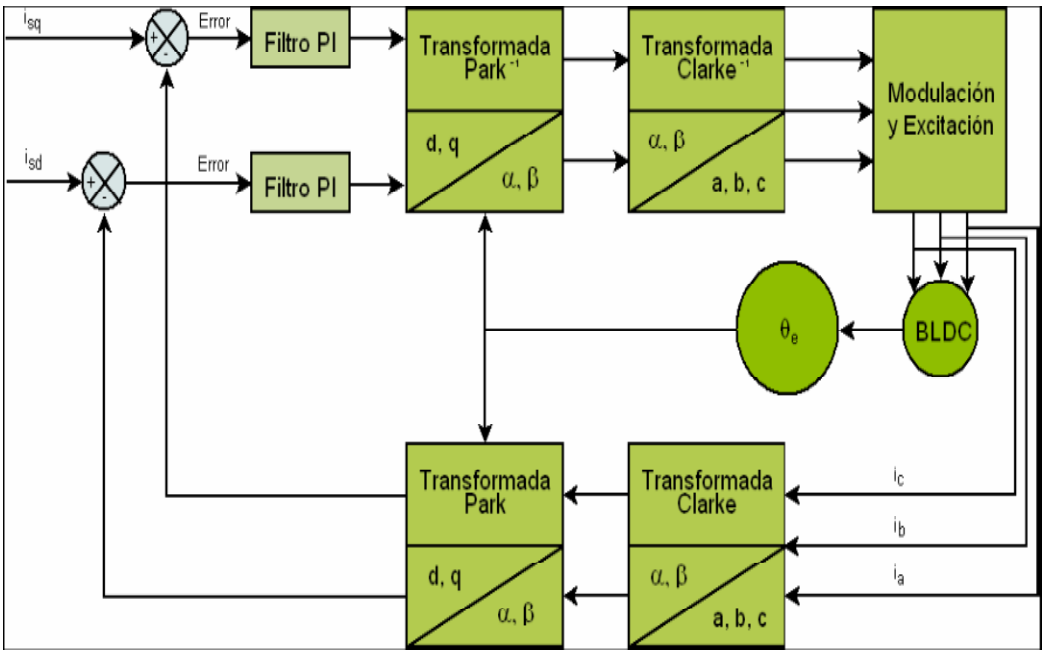


Fig. 2.22: ESQUEMA DE CONTROLADOR CON CONTROL VECTORIAL. [11]

2.4.4 DIFERENCIAS ENTRE MOTORES CON ESCOBILLAS (MOTOR BRUSHED) Y SIN ESCOBILLAS (MOTOR BRUSHLESS).

Hay muchas diferencias entre los motores con escobillas y sin escobillas entre las más destacadas se puede mencionar las siguientes:

Motores con escobillas como todos los motores están formados por 2 partes fundamentales, la estática ESTATOR (parte fija del motor) y la que rota conocida como ROTOR o INDUCIDO. En el estator se encuentra imanes permanentes, y el rotor está formado por un eje metálico con polos que sobresalen en los cuales hay bobinas, produciendo un campo magnético y con el campo magnético del estator se producirá el giro del rotor y así giraría el eje del motor.

Motores sin escobillas se puede decir que son parecidos a los motores con escobillas pero a la inversa ya que las bobinas las encontramos en el estator y los imanes permanentes en el rotor. En estos motores la corriente eléctrica pasa por un conductor de la bobina (ubicada en el estator) produciendo el campo electromagnético y así giran los imanes del rotor y por lo tanto el eje del motor. Por esto, no son necesarios ni las escobillas ni el conmutador, pero usan un controlador electrónico que suministra la corriente y determina la posición del rotor, esto se realiza con sensores instalados en el motor.

	MOTOR BLDC	MOTOR CON ESCOBILLAS
Conmutacion	Conmutacion electrónica basada en sensores de posición de efecto Hall	Conmutación por escobillas
Mantenimiento	Mínimo	Periódico
Durabilidad	Mayor	Menor
Curva Velocidad/par	Plana. Operación a todas las velocidades con la carga definida	Moderada. A altas velocidades la fricción de las escobillas se incremente, reduciendo el par.
Eficiencia	Alta. Sin caída de tensión por las escobillas	Moderada.
Potencia de salida/tamaño	Alta. Menor tamaño debido a mejores características térmicas porque los bobinados están en el estator, que al estar en la carcasa tiene una mejor disipación de calor.	Baja. El calor producido en la armadura es disipado en el interior aumentando la temperatura y limitando las características.
Inercia del rotor	Baja. Debido a los imanes permanentes en el rotor	Alta. Limita las características dinámicas.
Rango de velocidad	Alto. Sin limitaciones mecánicas impuestas por escobillas/conmutador.	Bajo. El límite lo imponen principalmente las escobillas
Ruido eléctrico generado	Bajo.	Arcos en las escobillas.
Coste de contrucción	Alto. Debido a los imanes permanentes.	Bajo.
Control	Complejo y caro	Simple y barato.
Requisitos de control	Un controlador es requerido siempre para mantener el motor funcionando. El mismo puede usarse para variar la velocidad.	No se requiere control si no se requiere una variación de velocidad.

[12]

2.4.5 USOS Y APLICACIONES

Estos motores BLDC han ganado mucho campo en la actualidad siendo utilizados en diversos campos como lo son:

- Pequeños aparatos de baja potencia como lectores de CD-ROM, o en ventilador (fan) de las PCs.
- Sectores industriales como:
- Automotriz.
- Aeroespacial.
- Barcos
- Equipos Médicos.
- Aire acondicionado
- Mecanismos o máquinas automatizadas.
- Instrumentación.
- En la industria manufacturera se usan para el movimiento de sistemas de control o de posicionamiento.

2.4.6 VENTAJAS Y DESVENTAJAS

Una de las grandes ventajas es que al no tener escobilla ya no hay reducción del rendimiento, no se desprende calor, se aceleran más rápidamente, no son

tan ruidosos debido a que no hay chispas al contacto de las escobillas con el conmutador y al no producirse el rozamiento ya no necesitan un mantenimiento con tanta frecuencia dado que no hay desgaste de dichas escobillas, además que no se producen aquellas partículas de carbón que ensucian el motor y ese polvo el cual puede ser conductor produciendo así grandes inconvenientes si no se le da un mantenimiento frecuente.

Estos motores son ideales para muchas aplicaciones debido a su densidad de potencia y sus características de Torque vs Velocidad, una eficiencia superior al 90% frente a un 60% de los motores con escobillas, así logran mayor rpm en un factor de 4 a 5 veces con respecto a los motores con escobillas, y claro no se puede dejar de mencionar el ahorro de energía que se produce al usar motores sin escobillas, además que tienen un menor peso que los otros motores.

Algo muy importante es que gracias al control electrónico se puede regular el torque y las revoluciones.

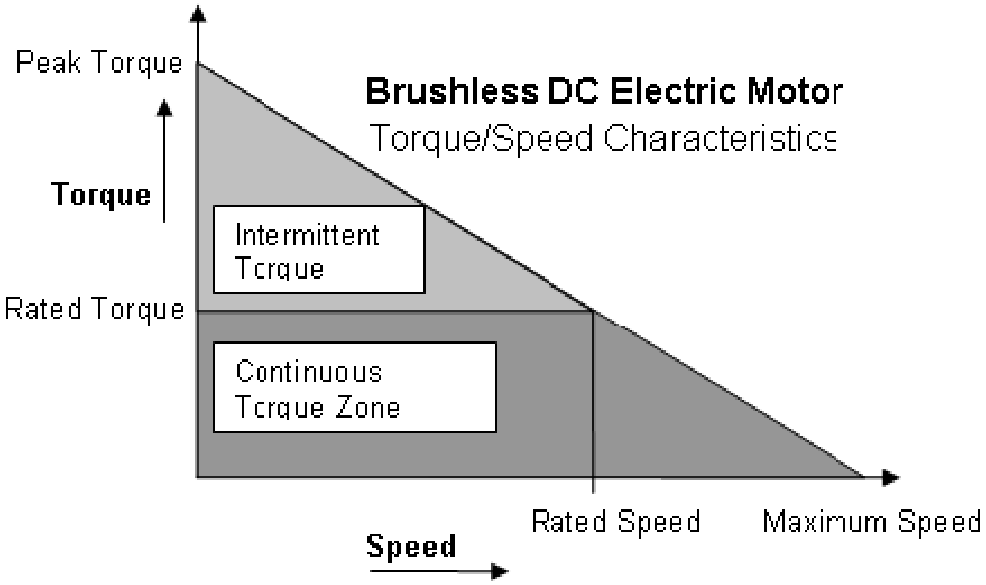


Fig. 2.23: Características Torque vs Velocidad. [13]

En resumen se podrían decir que las ventajas son:

- Mayor respuesta dinámica.
- Mayor vida útil.
- Mayor eficiencia (menos pérdida por calor).
- Mayor rendimiento (mayor duración de las baterías para la misma potencia).
- Menor peso para la misma potencia.
- Conmutación electrónica basada en sensores de posición de efecto Hall.

- Requieren menos mantenimiento al no tener escobillas.
- Relación velocidad/par motor es casi una constante.
- Mayor potencia para el mismo tamaño.
- Mejor disipación de calor.
- Rango de velocidad elevado al no tener limitación mecánica.
- Menor ruido electrónico (menos interferencias en otros circuitos).

Los motores brushless prácticamente no hacen ruido!!

Además, la relación par motor-tamaño es mucho mayor, lo que implica que se puedan emplear en aplicaciones donde se trabaje con un espacio reducido.

Por otra parte, los motores BLDC tienen desventajas, que son las siguientes:

1. Tienen un mayor costo de construcción.
2. Requieren un control bastante complejo.
3. Pueden tener problemas al enfrentarse a situaciones extremas debido a la electrónica que se utiliza.
4. Siempre hace falta un control electrónico para que funcione, que a veces duplica el costo.

5. Tienen la desventaja de que no giran al revés al cambiarles la polaridad (+ y -). Para hacer el cambio se deberían cruzar dos conductores del sistema electrónico.

CAPÍTULO 3

EJERCICIOS PREVIOS A LA ELABORACIÓN DE NUESTRO PROYECTO

3.1 Introducción

En el presente capítulo se detallará y analizará las principales pruebas realizadas utilizando el protocolo de comunicación I2C y muy necesarias para nuestro proyecto de una forma fácil de comprender, estas pruebas son tanto en la LPCXpresso así como la AVR Butterfly ya sea como maestro así como

esclavo, lo cual nos ayudará en el desarrollo del el tema de proyecto enfocado a la medición de la temperatura de motores BLDC utilizando el protocolo de comunicación antes mencionado.

En cada prueba que hemos realizado se hará una explicación del funcionamiento utilizando diagramas tanto de bloques como los diagramas de flujo, así como también se presentará el código de programación de cada una de las tarjetas usadas para dicha prueba.

3.2 Transmisión y almacenamiento de datos entre LPCXpresso y EEPROM

Esta prueba consiste en almacenar cierta cantidad de datos transmitidos en una memoria, para la transmisión usaremos el protocolo de comunicación I2C y haremos actuar la tarjeta LPCXpresso como maestro es decir que será la encargada de enviar los datos a almacenar a una memoria eeprom 24LC32A, dado que se realizará también una lectura de los datos almacenados en la eeprom se han colocado una cantidad suficientes de leds en la tarjeta LPCXpresso para que así se pueda visualizar los datos ledos.

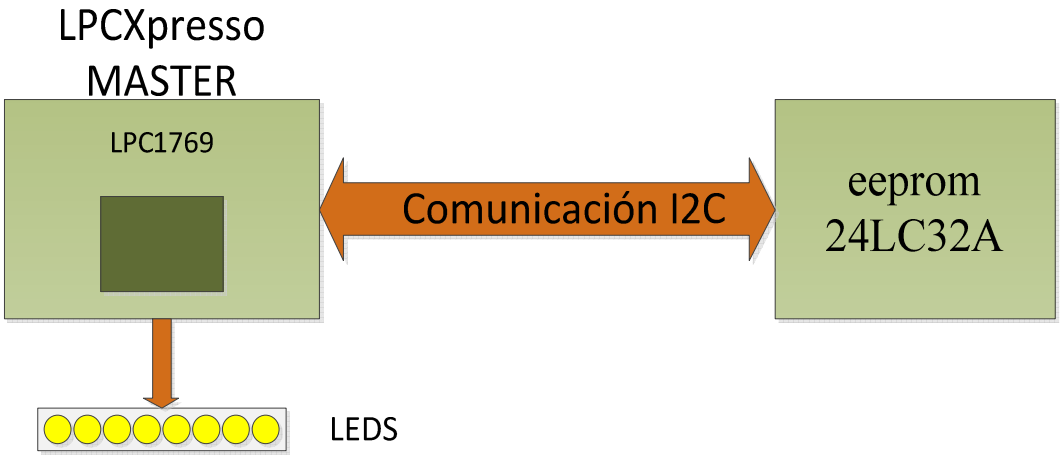


Fig. 3.1: Diagrama de bloques transmisión y almacenamiento de datos.

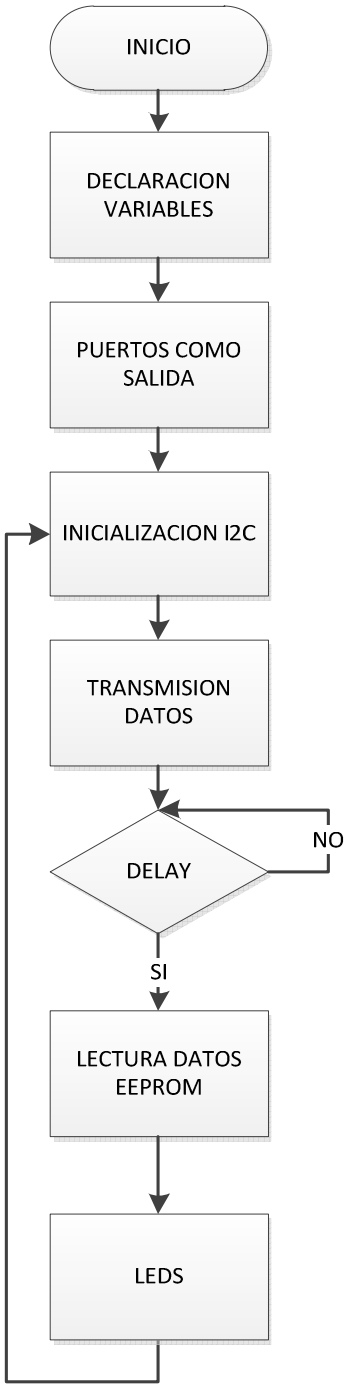


Fig.3.2: Diagrama de flujo transmisión y almacenamiento de datos

3.2.1. Código fuente en C

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

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

#include "lpc17xx.h"
#include "type.h"
#include "i2c.h"

extern volatile uint8_t I2CMasterBuffer[I2C_PORT_NUM][BUFSIZE];
extern volatile uint8_t I2CSlaveBuffer[I2C_PORT_NUM][BUFSIZE];
extern volatile uint32_t I2CReadLength[I2C_PORT_NUM];
extern volatile uint32_t I2CWriteLength[I2C_PORT_NUM];

#define PORT_USED              2

//********************  Main Function  main()**********************
int main (void)
{
  uint32_t i=0;

  uint32_t j=0;

  SystemClockUpdate();

  LPC_GPIO2->FIODIR = 0xFFFFFFFF;          /* Puerto 2 como salida*/
  LPC_GPIO2->FIOCLR = 0xFFFFFFFF;          /* turn off all the LEDs */

  I2C2Init( );                  /* initialize I2c2 */

  /* Write SLA(W), address and one data byte */
  I2CWriteLength[PORT_USED] = 11;
  I2CReadLength[PORT_USED] = 0;
  I2CMasterBuffer[PORT_USED][0] = PCF8594_ADDR;
  I2CMasterBuffer[PORT_USED][1] = 0xFF;                /* address */
  I2CMasterBuffer[PORT_USED][2] = 0x06;                /* address */
```

```
  I2CMasterBuffer[PORT_USED][3] = 0x01;          /* Data0 */
  I2CMasterBuffer[PORT_USED][4] = 0x02;          /* Data1 */
  I2CMasterBuffer[PORT_USED][5] = 0x04;          /* Data0 */
  I2CMasterBuffer[PORT_USED][6] = 0x08;          /* Data1 */
  I2CMasterBuffer[PORT_USED][7] = 0x10;          /* Data1 */
  I2CMasterBuffer[PORT_USED][8] = 0x20;          /* Data1 */
  I2CMasterBuffer[PORT_USED][9] = 0x40;          /* Data1 */
  I2CMasterBuffer[PORT_USED][10] = 0x80;         /* Data1 */
  I2CEngine( PORT_USED );

  for ( i = 0; i < 0x200000; i++ );          /* Delay after write */
  for ( i = 0; i < BUFSIZE; i++ )
  {
        I2CSlaveBuffer[PORT_USED][i] = 0x00;
  }
  /* Write SLA(W), address, SLA(R), and read one byte back. */
  I2CWriteLength[PORT_USED] = 3;
  I2CReadLength[PORT_USED] = 8;
  I2CMasterBuffer[PORT_USED][0] = PCF8594_ADDR;
  I2CMasterBuffer[PORT_USED][1] = 0xFF;          /* address */
  I2CMasterBuffer[PORT_USED][2] = 0x06;          /* address */
  I2CMasterBuffer[PORT_USED][3] = PCF8594_ADDR | RD_BIT;
  I2CEngine( PORT_USED );

  /* Check the content of the Master and slave buffer */
  while ( 1 ){
        for ( i =0; i < 8; i++ )
          {
              LPC_GPIO2->FIOPIN = I2CSlaveBuffer[PORT_USED][i];
              for(j = 6000000; j > 0; j--);
          }
  }
}
```

3.3 Conversión Analógica Digital usando la Tarjeta LPCXpresso

En esta prueba se utiliza un potenciómetro para simular los valores analógicos que ingresan a la tarjeta LPCXpresso por medio de su respectivo puerto

analógico, dado que no se ha colocado una forma de visualización externa hemos hecho uso de los recursos del programa para la programación de la LPCXpresso como lo es la herramienta para mostrar los valores de las variables además del uso de un voltímetro para conocer el valor exacto de voltaje que ingresa al puerto analógico de nuestra tarjeta de esta forma hemos podido corroborar que la conversión analógica es correcta.

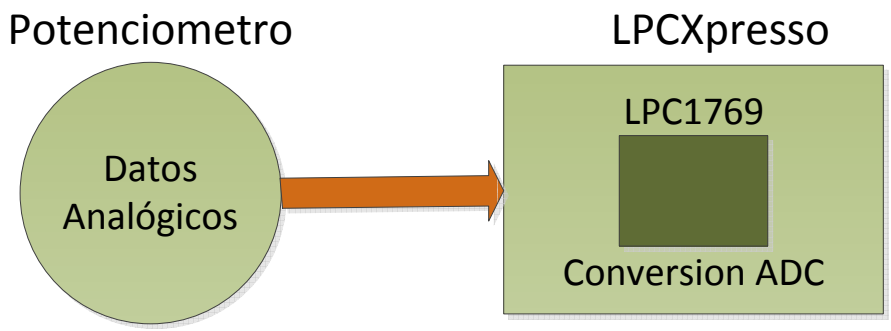


Fig. 3.3: Diagrama de Bloques de la conversión Analógica digital en LPCXpresso.

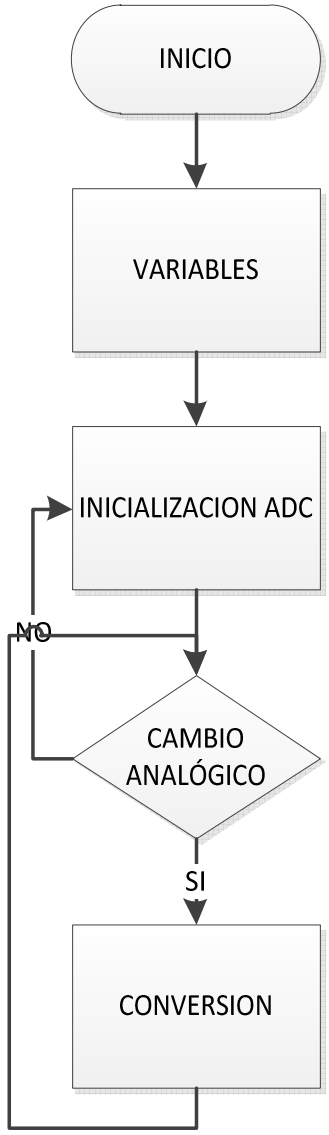


Fig.3.4: Diagrama de flujo conversión analógica digital en LPCXpresso.

3.3.1. Código fuente en C

```
/*****************************************************************
**        ADC USANDO POTENCIOMETRO Y LA LPCXPRESSO
*****************************************************************/

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

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

#include "lpc17xx.h"
#include "type.h"
#include "adc.h"

extern volatile uint32_t ADCValue[ADC_NUM];
extern volatile uint32_t ADCIntDone;
extern volatile uint32_t OverRunCounter;

/*****************************************************************
**                                Main Function  main()
*****************************************************************/
int main (void)
{
          /* SystemClockUpdate() updates the SystemFrequency variable */
          SystemClockUpdate();
           int k;

          for(k = 45000000; k > 0; k--);

          ADCInit( ADC_CLK );          /* Initialize ADC  */

           int i=0;
```

```
        while(1)
        {
                #if BURST_MODE                  /* Interrupt driven only */
                        ADCBurstRead();
                        while ( !ADCIntDone );
                        ADCIntDone = 0;
                        if ( OverRunCounter != 0 )
                        {
                                while ( 1 );
                        }

                #else                                   /* Not burst mode */
                        #if ADC_INTERRUPT_FLAG   /* Interrupt driven */
                                for ( i = 0; i < ADC_NUM; i++ )
                                {
                                        ADCRead( i );
                                        while ( !ADCIntDone );
                                        ADCIntDone = 0;
                                }
                        #else                                   /* Polling */
                                for ( i = 0; i < ADC_NUM; i++ )
                                {
                                        ADCValue[i] = ADCRead( i );
                                }
                        #endif          /* Endif interrupt driven */
                #endif  /* Endif BURST mode */
        }
}
```

3.4. Código para el uso del LCD de la tarjeta AVR Butterfly

Para la realización de este ejercicio nos hemos valido de un contador el cual se incrementa cada cierto tiempo y se muestra este valor en el LCD de la butterfly, esta prueba se la realizó ya que en el proyecto de grado se utilizará el LCD de la

tarjeta AVR Butterfly así que se han hecho las respectivas pruebas tanto de las librerías del LCD como el uso de variables y retardos en la programación C para la AVR Butterfly.

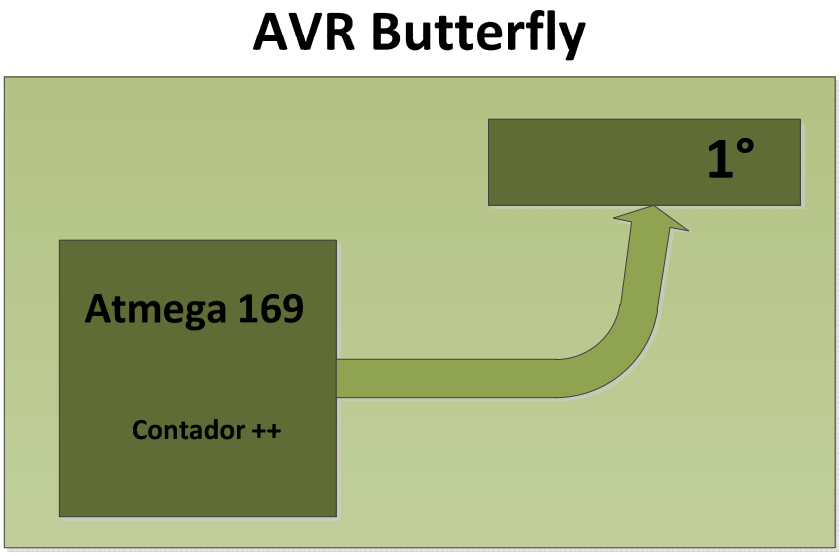


Fig. 3.5: Diagrama de bloques de butterfly usando LCD

Fig. 3.6: Diagrama de flujo de butterfly usando LCD

3.4.1. Código fuente en C

```
// Programa principal

#define __ATmega169__
# define F_CPU 1000000UL

// Librerias que usaremos
#include        <avr/io.h>
#include        <avr/interrupt.h>
#include        <util/delay.h>
#include        <string.h>
#include        <stdlib.h>
#include         "LCD_driver.h"
#include        "LCD_functions.h"

// const char LCD_START_msg[] PROGMEM = "Iniciando\0";

char *LCD_START_msg = "Iniciando\0";
char *LCD_PROCESS_msg = "__0. -\0";

int Counter;
int Flag;
char *Text;
int intTemp;
char *charTemp = "000\0";

int main( void )
{
        unsigned char  temp;

        DDRA = 0xFF;  // L2 Used to time interrupts.
        Counter = 0;
        Flag = 0;
        intTemp = 100;

         // initialize the LCD
         LCD_Init();

         // Habilita interrupciones
         sei();

        LCD_puts(LCD_PROCESS_msg, 0);
```

```
        for(;;)
        {
                Counter = Counter + 1;
                if(Counter == 10000)
                {
                        Counter = 0;
                        if(Flag==1){
                                LCD_PROCESS_msg[5]='-';
                                Flag=0;
                        }
                else{
                        LCD_PROCESS_msg[5]='+';
                        Flag=1;
                }
                itoa(intTemp, charTemp, 10);
                if(strlen(charTemp)>1) {
                        if(strlen(charTemp)>2) {
                                LCD_PROCESS_msg[0] = charTemp[0];
                                LCD_PROCESS_msg[1] = charTemp[1];
                                LCD_PROCESS_msg[2] = charTemp[2];
                        }
                        Else
                        {
                                LCD_PROCESS_msg[0] = '_';
                                LCD_PROCESS_msg[1] = charTemp[0];
                                LCD_PROCESS_msg[2] = charTemp[1];
                        }
                }
              else              {
                   LCD_PROCESS_msg[0] = '_';
                   LCD_PROCESS_msg[1] = '_';
                   LCD_PROCESS_msg[2] = charTemp[0];
               }
               LCD_puts(LCD_PROCESS_msg, 0);
               if(intTemp<180)         {
                   intTemp++;
               }
               else              {
                   intTemp=0;
               }
        }
}
```

3.5. Comunicación I2C entre LPCXpresso y Butterfly.

En este ejercicio hemos realizado la comunicación entre 2 tarjetas las cuales son AVR Butterfly y la LPCXpresso, para esta prueba se utilizó la tarjeta LPCXpresso como transmisor es decir que esta configurada como master y se ha usado para receptar los datos osea configurada como esclavo la tarjeta AVR Butterfly, los datos que serán transmitidos se podrán visualizar por medio de leds colocados en uno de los puertos con los que cuenta la AVR Butterfly, así también se ha colocado un led en uno de los puertos de la LPCXpresso el cual se encenderá cada que se transmite un dato por medio de la comunicación I2C hacia el esclavo.

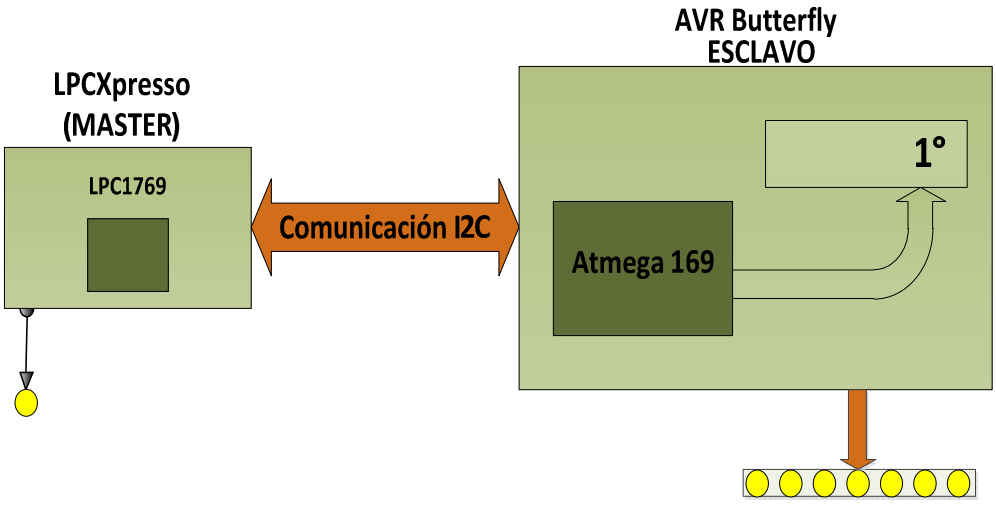


Fig. 3.7: Diagrama de bloques de comunicación entre LPCXpresso y Avr butterfly.

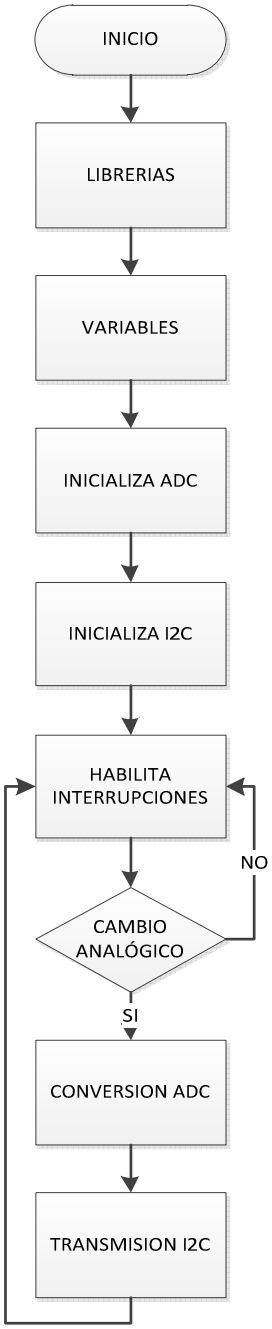


Fig. 3.8: Diagrama de flujo de comunicación entre LPCXpresso como Master.

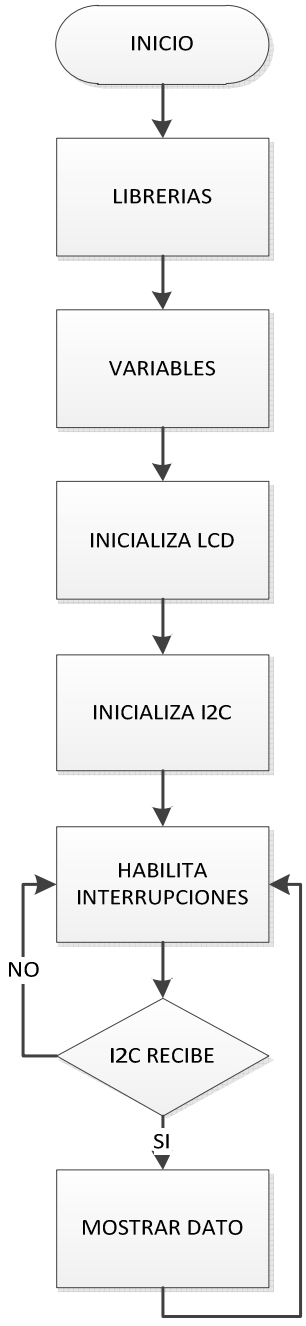


Fig. 3.9: Diagrama de flujo de comunicación Avr Butterfly como Esclavo.

3.5.1. Código para Avr Butterfly (Esclavo)

```
// Librerias
//#include <avr/cpufunc.h>

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <string.h>
#include <stdlib.h>
#include "main.h"
#include "i2c.h"
#include "LCD_driver.h"
#include "LCD_functions.h"

volatile uint8_t Temp;

char *LCD_PROCESS_msg = "__1. -\0";
int intTemp;
char *charTemp = "000\0";

int main( void )
{

        MCUCR |= (1<<PUD);
        PORTB = 0x00000000;
        DDRB = 0xFF;
        cli();

         // inicialización del LCD
        LCD_Init();

        I2CInit();
        sei();

         LCD_puts(LCD_PROCESS_msg, 0);
        DELAY_2M;
```

```
        // programa principal
        while(1)
        {
                cli();
                receiveI2C();

         intTemp = Temp;
         itoa(intTemp, charTemp, 10);
         if(strlen(charTemp)>1)

         {
                if(strlen(charTemp)>2)
                {

                        LCD_PROCESS_msg[0] = charTemp[0];
                        LCD_PROCESS_msg[1] = charTemp[1];
                        LCD_PROCESS_msg[2] = charTemp[2];
           }
                else
                {
                        LCD_PROCESS_msg[0] = '_';
                        LCD_PROCESS_msg[1] = charTemp[0];
                        LCD_PROCESS_msg[2] = charTemp[1];
                }
   }
         else
         {
                LCD_PROCESS_msg[0] = '_';
                LCD_PROCESS_msg[1] = '_';
                LCD_PROCESS_msg[2] = charTemp[0];
         }
           DELAY_2M;
           sei();
           LCD_puts(LCD_PROCESS_msg, 0);
           DELAY_2M;
          }
        }
        ISR(BADISR_vect) {
}
```

3.5.2. Código para la LPCXpresso (Maestro)

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

__CRP const unsigned int CRP_WORD = CRP_NO_CRP ;

#include "lpc17xx.h"
#include "type.h"
// #include "i2c.h"
#include "main.h"
#include "adc.h"
#include "i2cmini.h"
volatile uint32_t ADCValue;
volatile uint32_t xDelay;

#define PORT_USED              1
#define ADC_PORT_USED     0

/*****************************************************************
**   Main Function  main()
*****************************************************************/
int main (void)
{

  uint32_t xTemp;

  /* SystemClockUpdate() updates the SystemFrequency variable */
  SystemClockUpdate();

  LPC_GPIO2->FIOPIN |= 0x00000000;
  LPC_GPIO2->FIODIR |= 1 << 7;

  I2CInit();

  /* Initialize ADC  */
  ADCInit( ADC_CLK );

  DELAY_2M;  // esperamos para que el voltaje se estabilice a lo largo del
             circuito
  DELAY_2M;  // esperamos para que el voltaje se estabilice a lo largo del
             circuito
```

```
 DELAY_2M;  // esperamos para que el voltaje se estabilice a lo largo del
                circuito

/* Check the content of the Master and slave buffer */
 while ( 1 ){
         // Reading ADC
           ADCValue = ADCRead( ADC_PORT_USED );

        // Aqui se debe hacer el cálculo de conversión 12bit a temperatura
           xTemp = (ADCValue * 30) / 457;

           ADCValue = xTemp;

          /* Write SLA(W), address and one data byte */

 LED_ON;

           sendI2C();

           LED_OFF;
           DELAY_2M;
 }


/*********************************************************************
**                      End Of File
*********************************************************************/
```

CAPÍTULO 4

DESARROLLO DEL PROYECTO DE GRADUACIÓN

En el siguiente capítulo demostraremos mediante ejercicios realizados como pudimos realizar la comunicación I2C entre la AVR Butterfly y la LPCXpresso así como también las diferentes pruebas que se hicieron con estas tarjetas.

4.1. Comunicación I2C entre LPCXpresso y EEPROM.

En este ejercicio realizaremos la comunicación I2C entre las dos tarjetas: la LPCXpreso que actuara como maestro y la EEPROM 24LC32A como esclavo. Esta última receptara en sus registros datos que le enviara la tarjeta master, datos que luego serán leídos por la LPC1769 (tarjeta master) y mostrados en leds.

Lista de Materiales:

- Un protoboard.
- Una EEPROM 24LC32A.
- Una LPC1769.
- 4 leds.
- 4 resistencias de 330Ω.
- 2 resistencias de 10kΩ.

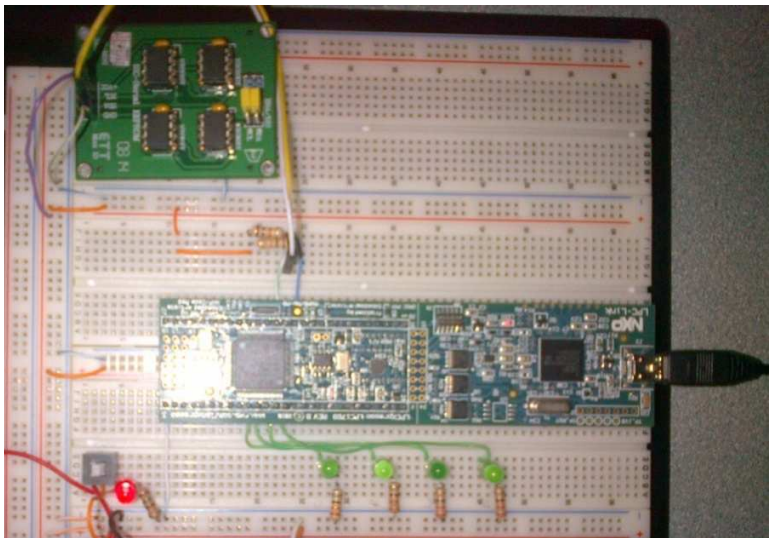


Fig. 4.1: Prueba de comunicación I2C entre LPCXpresso y EEPROM.

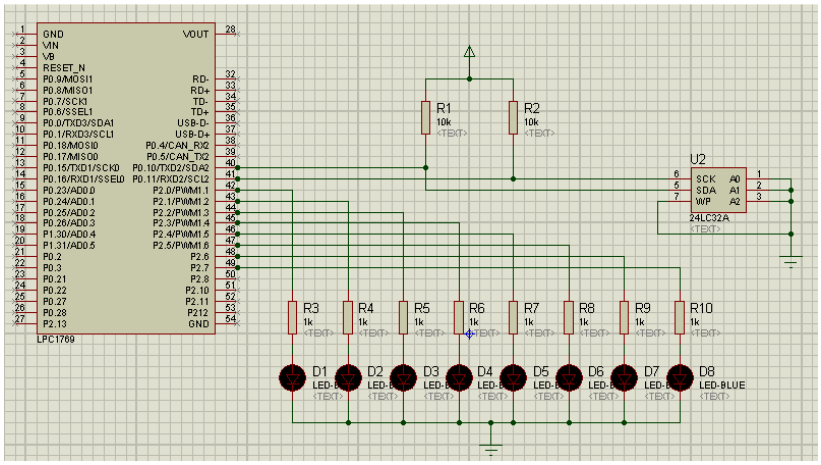


Fig. 4.2: Diagrama eléctrico de la LPC1769 y la EEPROM.

Conclusión

Para realizar la comunicación I2C se hace uso de 2 puertos específicos de la tarjeta: el puerto SCL y el SDA. Con el cableado correcto y el encendido de los diferentes leds se pudo comprobar que se enviaban y recibían los datos que previamente fueron enviados por la tarjeta master LPC1769.

4.2. Uso del convertidor analógico/digital de la LPCXpresso.

Para este ejercicio en particular haremos uso de un potenciómetro que será debidamente cableado al puerto ADC de la tarjeta LPCXpresso para registrar la respectiva conversión.

Lista de materiales:

- Un protoboard.
- Un potenciómetro de 10kΩ.
- Una LPC1769.

Fig.4.3: Prueba de ADC en la LPCXpresso con un potenciómetro

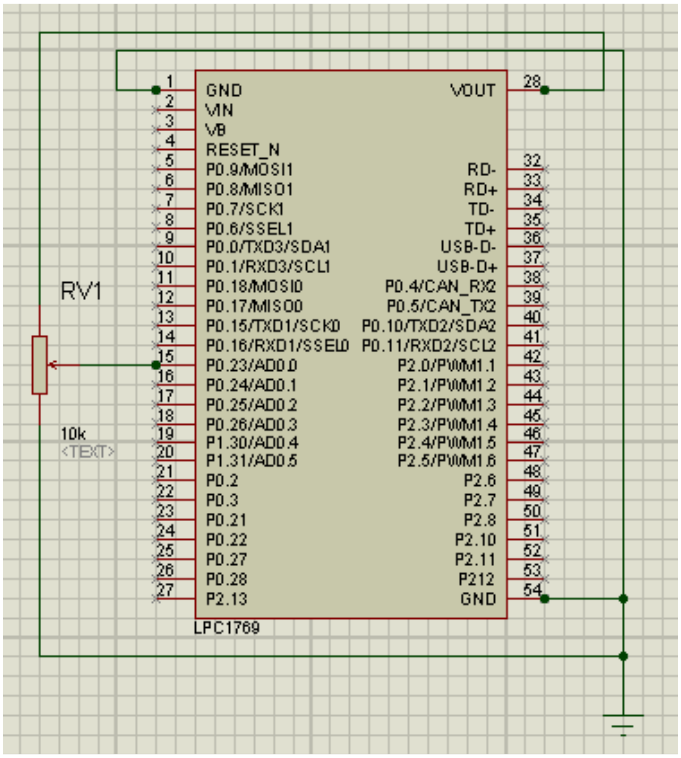


Fig. 4.4: Diagrama eléctrico de la LPC1769 con un potenciómetro.

Conclusión.

Este ejercicio nos ayudó para poder realizar un análisis y por ende corroborar el funcionamiento del ADC de la tarjeta LPCXpresso, simulando el cambio de voltaje con un potenciómetro que fue conectado a la entrada ADC de la tarjeta en mención.

4.3. Uso del LCD de la Tarjeta AVR Butterfly.

Este ejercicio lo hemos realizado para aprender y hacer uso de varios de los recursos que nos ofrece la tarjeta AVR Butterfly como en este caso lo es la LCD.

A continuación detallaremos los componentes que utilizamos para esta prueba.

Lista de materiales:

- Una tarjeta AVR Butterfly

Fig. 4.5: Prueba LCD en tarjeta AVR Butterfly.

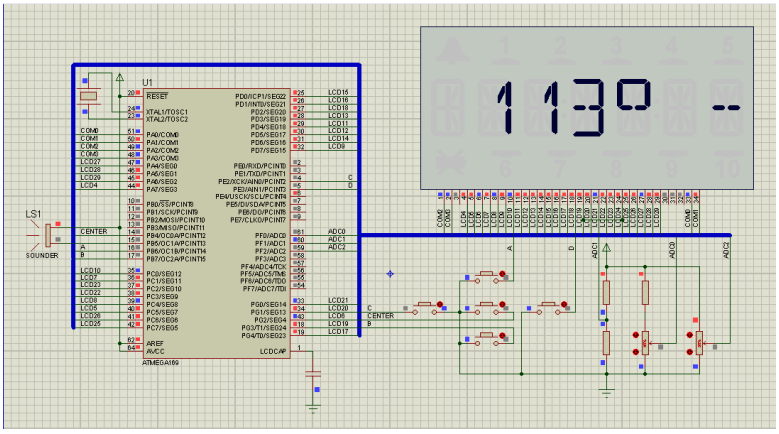


Fig. 4.6: Diagrama eléctrico de la AVR Butterfly.

Conclusión.

A medida que va evolucionando la tecnología, nos encontramos que en la actualidad los componentes cada vez son más pequeños y vienen implantados dentro de dispositivos. Este es el caso de esta tarjeta que tiene incorporado una LCD que nos ayudara a mostrar los datos y así haber probado dicha pantalla de cristal liquido.

4.4. Comunicación I2C entre LPCXpresso y AVR Butterfly.

Nuestro proyecto se basa en la medición de temperatura a un motor BLDC y la comunicación entre dos tarjetas LPCXpresso y AVR Butterfly. Haremos uso de varios de los recursos que nos brindan estas tarjetas.

Lista de materiales:

- Un protoboard.
- Una LPCXpresso.
- Una AVR Butterfly
- 2 resistencias de 10kΩ.
- 1 LM35.
- 1 resistencia de 330Ω.
- 1 led
- 1 capacitor de 10uF
- 1 botonera.

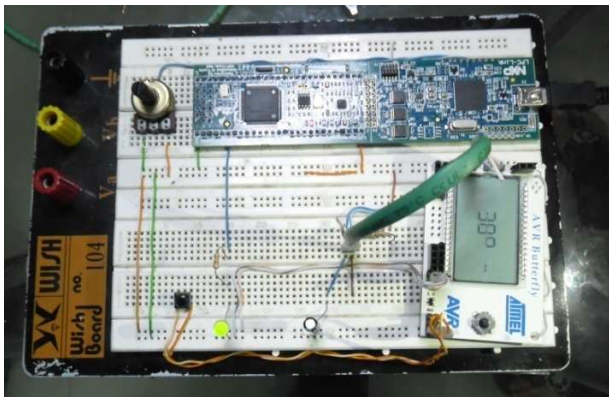


Fig. 4.7: Implementación del proyecto final

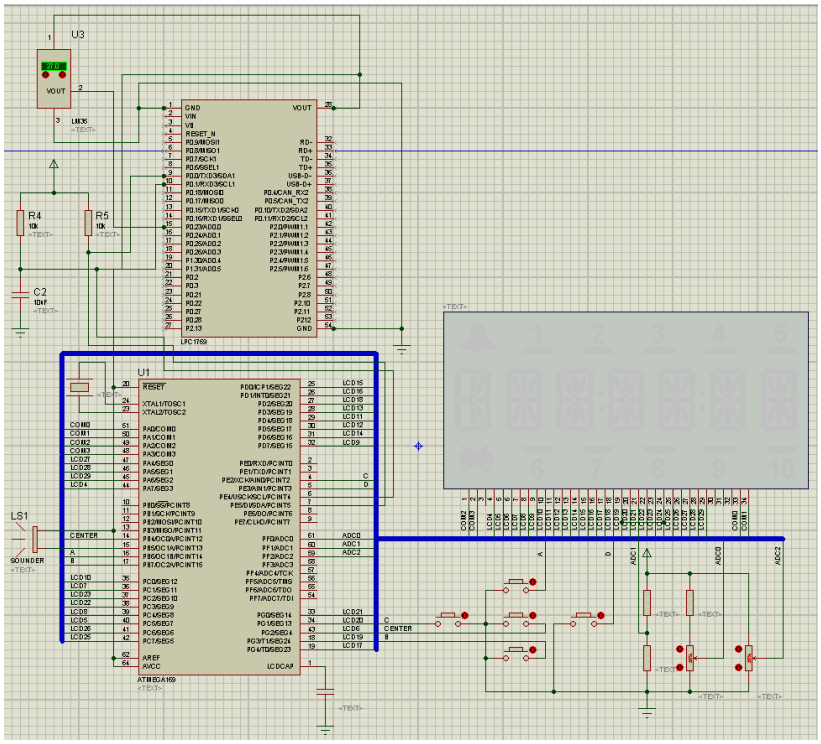


Fig. 4.8: Diagrama eléctrico entre AVR Butterfly y LPCXpresso

Conclusión

Como podemos darnos cuenta la comunicación entre las dos tarjetas se efectuó de excelente forma y así se logró realizar el proyecto con éxito. Surgieron inconvenientes en este proyecto como por ejemplo la distorsion y perdida de señal, pero ésta se supo sobrellevar y corregir a tiempo.

CONCLUSIONES

1) El protocolo I2C es muy útil para transmisión de datos entre dispositivos así sean de diferente clase, en este caso hemos realizado esta comunicación entre las tarjetas AVR BUTTERFLY y LPCXpresso con gran éxito; su comprobación se efectuó tomando datos de temperatura mediante un sensor LM35 a un motor BLDC y visualizándolos en una LCD propia de la tarjeta AVR BUTTERFLY. Algo muy importante es recalcar que estas tarjetas cuentan con microcontroladores que tienen módulos internos para la transmisión y recepción por I2C.

2) La sincronización en la comunicación de las tarjetas por medio del protocolo I2C fue algo muy necesario dado que se nos presentaron algunos inconvenientes debido a las velocidades con las que trabajan cada una de ellas, dicho problema fue muy bien superado gracias a diferentes validaciones que le permitía tanto al maestro como al esclavo cuando era transmisión y cuando se podía enviar otro dato.
3) El Kit AVR Butterfly es una poderosa herramienta de aprendizaje, es práctica, eficaz y muy amigable; que con el desarrollo de las prácticas el usuario va descubriendo progresivamente las características del microcontrolador ATmega169.
4) Fue de gran ayuda el que la tarjeta LPCXpresso tenga incluido en su software ejemplos, ya que estos fueron de gran ayuda aprender a utilizarla de una mejor manera así como también en la realización de nuestro proyecto.
5) La característica de controlador LCD, del ATmega169, permite abaratar costos en la implementación de aplicaciones que necesitan despliegue de información a través de LCD. Esta característica permite controlar pantallas de cristal líquido de bajo costo, que por ser básicas no poseen ni driver interno, ni interfaces de comunicación como la mayoría de los costosos módulos LCD.

RECOMENDACIONES

1) No apoyar el Kit AVR Butterfly en superficies conductivas tales como metal, líquidos, etc., puesto que podrían causar daños en el mismo.
2) Se recomienda colocar espadines hembra o macho en los espacios destinados para conexiones externas de este kit en lugar de colocar cables directamente ya que esto podría causar un corto circuito.
3) Obtener información suficiente sobre los microcontroladores a usar en este caso AVR Butterfly y LPCXpresso para conocer la correcta distribución y funcionamiento de cada uno de los pines.

4) Se recomienda revisar el datasheet de todos los componentes para asi revisar la configuración y conexión de todos sus pines para evitar cualquier daño irreversible en sus componentes.

BIBLIOGRAFÍA

[1] Code red

www.**code**-**red**-tech.com/

Nombre del autor: Code Red Technologies Ltd

Fecha de consulta: Mayo 2012

[2] Embedded Artists

http://www.embeddedartists.com

Nombre del autor: Embedded Artists

Fecha de consulta: Mayo 2012

[3] Incubadora Rarosch2

http://www.avserviciosmedicos.com/web/index.php?option=com_content&view=article&id=47&Itemid=1

Nombre del autor: Norman

Fecha de consulta: Marzo 2012

[4] “Getting starts with NXP LPCXpresso”, USER GUIDE, LPCXpresso.

Nombre del autor: LPCXpresso

Fecha de consulta: Marzo 2012

[5] LPCXpresso - ICs - NXP Semiconductors

http://ics.nxp.com/lpcxpresso/

Nombre del autor: NXP Semiconductors.

Fecha de consulta: Marzo 2012

[6] AVR Butterfly Evaluation Kit - User Guide

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

Nombre del autor: NXP Semiconductors.

Fecha de consulta: Marzo 2012

[7] Características del Kit AVR Butterfly en español

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

http://www.datasheetcatalog.org/datasheet/atmel/2514S.pdf

Nombre del autor: Atmel Corporation 2003.

Fecha de consulta: Marzo 2012

[8] “8-bit Mocrocontroller with 16K Bytes In-System Programmable Flash”, USER GUIDE, ATMEL.

http://www.atmel.com/Images/doc2514.pdf,

Nombre del autor: 2006 Atmel Corporation.

Fecha de consulta: Marzo 2012

[9] Guerrero Richard, (2006) “Kit de desarrollo AVR Butterfly, Desarrollo de guía de prácticas de laboratorio y tutoriales”, Proyecto grado para ingeniería, Sangolqui, Ecuador.

http://repositorio.espe.edu.ec/handle/21000/424

Nombre del autor: Richard Guerrero

Fecha de consulta: Marzo 2012

[10] LM35 -- Sensor de temperatura de precisión en grados centígrados

http://electronica.webcindario.com/componentes/lm35.htm

Nombre del autor: Carlos Díaz.

Fecha de consulta: Abril 2012

[11] Langarica Córdoba Diego, (13 de Octubre de 2010) "*Control de un Motor Brushless para Aplicación a Vehículos Eléctricos*", Maestría en Ciencias en Ingeniería Electrónica, Cuernavaca, Morelos, México, Centro Nacional de Investigación y Desarrollo Tecnológico Departamento de Ingeniería Electrónica. [online], Disponible en:

http://www.cenidet.edu.mx/subaca/web-elec/tesis_mc/243MC_dlc.pdf

Nombre del autor: Diego Langarica

Fecha de consulta: Febrero 2012

[12] Roger Juanpere Tolrà, "Técnicas de control para motores Brushless

Comparativa entre conmutación Trapezoidal, conmutación Sinusoidal y Control Vectorial", [online], España Barcelona, Departamento control de movimiento Disponible en: http://www.ingeniamc.com/Es/-Control-techniques-for-brushless-motors.pdf

Nombre del autor: Roger Juanpere Tolrá

Fecha de consulta: Febrero 2012

[13] Motor DC sin escobillas,

http://www.brushlessmotor.com.ar/Caracteristicas.html

Nombre del autor: Gerardo Man

Fecha de consulta: Febrero 2012

[14] Philips Semiconductors, THE I2C-BUS SPECIFICATION V2.1

http://i2c2p.twibright.com/spec/i2c.pdf

Nombre autor: Philips Semiconductors

Fecha de consulta: Abril 2012

[15] AVR STUDIO 4

http://www.gratisprogramas.org/descargar/avr-studio-4-19-avr-toolchain-4-19-hf/

Nombre del autor: Arkanosant

Fecha de consulta: Abril 2012

[16] Manual de Usuario del LPC1769,

http://www.nxp.com/documents/user_manual/UM10360.pdf,

Nombre del autor: NXP Semiconductors

Fecha de consulta: Marzo 2012