ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería Eléctrica y Computación

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

TESINA DE SEMINARIO

Previo a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Edison Javier Figueroa Romero

Jorge Pablo Saavedra Cepeda

GUAYAQUIL – ECUADOR

2012

A G R A D E C I M I E N T O

A Dios, por ser la guía de nuestro camino y fuente de esperanza, por colmarnos de bendiciones y brindarnos aliento y esperanza.

A nuestros padres, por su apoyo y consejos para nuestro diario vivir, por sus deseos de vernos triunfantes y por inculcarnos metas y sueños.

Además a nuestros amigos que con su apoyo moral y conocimientos, promovieron a la culminación de nuestra carrera.

DEDICATORIA

A nuestros padres, pilar fundamental de nuestras vidas, presentes durante todo este tiempo de aprendizaje y formación profesional.

TRIBUNAL DE SUSTENTACIÓN

_____

ING. CARLOS VALDIVIESO A.

PROFESOR DEL SEMINARIO DE GRADUACIÓN

_____

PROF. IGNACIO MARIN GARCIA MSIS

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)

_____	_____
Edison Javier Figueroa Romero	Jorge Pablo Saavedra Cepeda

RESUMEN

El proyecto consiste en la comunicación entre dos dispositivos, la tarjeta LPC1769 y el kit de desarrollo AVR Butterfly, haciendo uso del módulo SSP en modo de operación SPI, para analizar el ruido generado por un motor BLDC. Se emplearon los programas LPCXpresso y AVR Studio para la programación de los dispositivos, en conjunto con el lenguaje de programación C.

En la implementación del proyecto, la tarjeta LPC1769 toma la función de emisor o maestro, y se encarga de enviar los datos obtenidos por un sensor de impacto y sonido, asimismo, si existe un dato que sea causa de alerta se activará un contador que al llegar al número 5 activará una un grupo de LED’s que funcionarán como alarma del suceso.

La recepción de los datos enviados por la tarjeta LPC1769, la realiza el kit de desarrollo AVR Butterfly, tomando el nombre de receptor o esclavo. Este kit analiza los datos, bajo el mismo criterio que el emisor. Además en la LCD del mismo, se visualizarán mensajes del estado del motor. Para efectos de prueba, el sensor puede ser calibrado de acuerdo a las necesidades del usuario.

ÍNDICE GENERAL

RESUMEN VI

ÍNDICE DE FIGURAS IX

INTRODUCCIÓN XV

CAPÍTULO 1

ANTECEDENTES Y JUSTIFICACION 1

1.1 Antecedentes Teóricos 1

1.2 Antecedentes de campo 3

1.3 Motivación del proyecto 4

1.4 Justificación del proyecto 4

1.5 Objetivo General 5

1.6 Objetivos específicos 5

1.7 Limitaciones 6

CAPÍTULO 2

MARCO TEÓRICO 7

2.1 SPI 8

2.2 Sensor de Impacto de Sonido 29132 9

2.3 Tarjeta LPC1769 10

2.4 Kit de Desarrollo AVR Butterfly 11

2.5 LPCxpresso 12

2.6 AVR Studio 4 13

2.7 Motores BLDC 13

CAPÍTULO 3

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA 16

3.1 Ejercicio 1: Secuencia de encendido de LEDS 17

3.2 Ejercicio 2: Comunicación SPI entre dos tarjetas LPC1769 19

3.3 Almacenamiento de datos de vibraciones de motor BLDC para graficación y análisis en displays disponibles en tarjeta AVR Butterfly y en tarjeta controladora LPCXpresso. 24

CAPÍTULO 4

PRUEBAS Y SIMULACIONES 31

4.1 Ejercicio 1: Rotación de LEDS 31

4.2 Ejercicio 2: Comunicación SPI entre dos tarjetas LPC1769.....34

4.3 Almacenamiento de datos de vibraciones de motor BLDC para graficación y análisis en displays disponibles en tarjeta AVR Butterfly y en tarjeta controladora LPCXpresso 37

CONCLUSIONES 43

RECOMENDACIONES.....45

ANEXOS.....47

ANEXO A.....48

ANEXO B.....50

ANEXO C.....54

BIBLIOGRAFÍA.....60

ÍNDICE DE FIGURAS

Figura 1.1 Esquema Básico de Comunicación Asíncrona 2
Figura 1.2 Esquema de Maestro a Esclavo 2

Figura2. 1 Sensor de Impacto de Sonido 10
Figura2. 2 Tarjeta LPC1769 10
Figura2. 3 Distribución Pines Tarjeta LPC1769 11
Figura2. 4 Kit de Desarrollo AVR Butterfly 12
Figura2. 5 Motor BLDC 14

Figura3. 1 Diagrama de Bloques Ejercicio 1 17
Figura3. 2 Diagrama de Flujo Ejercicio 1 18
Figura3. 3 Código Bucles Ejercicio 1 19
Figura3. 4 Diagrama de Bloques Ejercicio 2 20
Figura3. 5 Diagrama de Flujo Dispositivo Maestro Ejercicio 2 21
Figura3. 6 Tabla del Codigo para visualizacion en display 22
Figura3. 7 Código de transmisión de datos 22
Figura3. 8 Diagrama de Flujo Dispositivo Esclavo Ejercicio 2 23
Figura3. 9 Código de transmisión de datos 24
Figura3. 10 Diagrama de Bloques Proyecto 26
Figura3. 11 Diagrama de Flujo Dispositivo Maestro del Proyecto 27
Figura3. 12 Diagrama de Flujo Dispositivo Esclavo del Proyecto 28
Figura3. 13 Código Dispositivo Maestro Envío de datos y Alarma 29
Figura3. 14 Código Dispositivo Maestro Envío de datos y Alarma 30

Figura4. 1 Diagrama de Conexiones del Ejercicio 1 32
Figura4. 2 Secuencia Normal 33
Figura4. 3 Secuencia inversa presionando botonera 34
Figura4. 4 Diagrama de Conexiones del Ejercicio 2 35
Figura4. 5 Transmisión SPI 36
Figura4. 6 Diagrama de Conexiones del proyecto final 38
Figura4. 7 Proyecto en estado NORMAL 40
Figura4. 8 Proyecto en estado ALERTA 41
Figura4. 9 Visualización Alarma 42

GLOSARIO

Término	Significado
Aplicación Embebida	Es un sistema diseñado para realizar procedimientos específicos, usado en microcontroladores y microcomputadores.
Arquitectura ARM	Es el conjunto de instrucciones de tamaño fijo (32 bits) que detalla que un procesador puede ejecutar.
Arreglo de datos	Es un conjunto o agrupación de variables del mismo tipo cuyo acceso se realiza por índices.
ATMEL	Es una compañía de semiconductores, Atmel sirve a los mercados de la electrónica de consumo, comunicaciones, computadores, redes, electrónica industrial, equipos médicos, automotriz, aeroespacial y militar. Es una industria líder en sistemas seguros, especialmente en el mercado de las tarjetas seguras.
AVR Butterfly	Es un kit de desarrollo para el microcontrolador AVR Atmega169, que puede ser usado en numerosas aplicaciones.
Bit	Un bit es una señal electrónica que puede estar encendida (1) o apagada (0). Es la unidad más pequeña de información que utiliza un ordenador.
Delgas	Láminas, generalmente de cobre, aisladas unas de otras y conectadas a su vez a los terminales de cada una de las bobinas giratorias que forman el rotor de un motor.
Display 7 segmentos	Es un componente que se utiliza para la representación de números en muchos dispositivos electrónicos. Se ensambla o arma de manera que se pueda activar cada segmento (diodo LED) por separado logrando de esta manera combinar los

	elementos y representar todos los números en el display (del 0 al 9).
EEPROM	Responde a “Erasable Programmable Read Only Memory” que se puede traducir como Memoria programable borrable de solo lectura. Como su nombre sugiere, una EEPROM puede ser borrada y programada con impulsos eléctricos.
Full Duplex	Capacidad de transmitir datos simultáneamente entre una estación emisora y una estación receptora.
GCC	Es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran; es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde ha de correr. La sigla GCC significa "GNU Compiler Collection".
GNU	Es un sistema operativo similar a Unix que es software libre y respeta su libertad.
GPIO	General Purpose Input/Output (GPIO) son básicamente portas programables de entrada y salida de datos. Son utilizadas para proveer una interfaz entre los periféricos y los microcontroladores.
Half Duplex	Capacidad de transmitir los datos en una sola dirección a la vez entre una estación emisora y una estación receptora.
Hardware	Conjunto de elementos materiales que constituyen el soporte físico de una computadora.
IDE	Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).
I2C	Inter Integrated Circuit, es un estándar diseñado por Philips que facilita la comunicación entre

	microcontroladores, memorias y otros dispositivos. Sólo requiere de dos líneas de señal y un común.
JTAG	Un acrónimo para Joint Test Action Group, es el nombre común utilizado para la norma IEEE 1149.1 titulada Standard Test Access Port and Boundary-Scan Architecture para test access ports utilizada para testear PCBs utilizando escaneo de límites.
Joystick	Palanca de mando. Dispositivo que se conecta con un ordenador o videoconsola para controlar de forma manual un software, especialmente juegos o programas de simulación.
KIT	Conjunto de las piezas de un objeto o aparato que se venden sueltas y con un folleto de instrucciones para montar con facilidad.
LCD	Liquid crystal display, Pantalla de cristal líquido.
LED	Light-Emitting Diode, Diodo Emisor de luz.
LPC1769	Es una tarjeta que contiene un microcontrolador basado en la Arquitectura ARM con procesador Cortex de núcleo M3, para aplicaciones embebidas.
Macro	Con el fin de evitar al programador la tediosa repetición de partes idénticas de un programa, los ensambladores y compiladores cuentan con macroprocesadores que permiten definir una abreviatura para representar una parte de un programa y utilizar esa abreviatura cuantas veces sea necesario.
Microcontrolador	Es un circuito integrado, o chip, compuesto por tres bloques funcionales: Unidad Central de Procesamiento (UCP), Memoria y Periféricos de entrada y salida. Se caracteriza por ser programable, económico y de dimensiones pequeñas.

Motor BLDC	BrushLess DC Motor. Es un motor de corriente continua sin escobillas, lo cual le permite tener ventaja sobre los motores eléctricos.
NXP	(Next eXPerience) Semiconductors es una empresa fabricante de semiconductores que se creó el 31 de agosto de 2006 a partir de la división de semiconductores de la empresa holandesa Philips.
PIC 16f887	Es un microcontrolador creado por Microchip, conocido por ser de alta calidad, rango de aplicaciones y bajo precio.
Pin	Patilla para conexiones eléctricas.
Potenciómetro	Es un resistor cuyo valor de resistencia es variable. De esta manera, indirectamente, se puede controlar la intensidad de corriente que fluye por un circuito si se conecta en paralelo, o la diferencia de potencial al conectarlo en serie.
Protoboard	Es una especie de tablero con orificios, en la cual se pueden insertar componentes electrónicos y cables para armar circuitos.
RISC	Es una filosofía de diseño de CPU para computadora que está a favor de conjuntos de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse, generalmente utilizado en microprocesadores o microcontroladores.
RS-232	Recommended Standard-232. Es un estándar para la conexión serial de señales de datos binarias entre un equipo terminal de datos y un equipo de terminación del circuito de datos.
Señal de reloj	Es una señal usada para coordinar las acciones de dos o más circuitos, oscila entre estado alto o bajo, y gráficamente toma la forma de una onda cuadrada.
Software	Término genérico que se aplica a los componentes no físicos de un sistema informático, como por ejemplo

	los programa y sistemas operativos que permiten a este ejecutar sus tareas.
SPI	Serial Peripheral Interface, Comunicación serial de Periféricos. Es un estándar para comunicación serial sincronizada por reloj, registrado por la compañía Motorola.
Módulo SSP	Synchronous Serial Port, Puerto Serial Síncrono. Es una interfaz de comunicación serial síncrona entre periféricos.
Torque	La propiedad de la fuerza para hacer girar al cuerpo se mide con una magnitud física que llamamos torque o momento de la fuerza.
USART	Es el acrónimo de Universal Synchronous / Asynchronous Receiver Transmitter, que traducido al español viene a ser algo parecido a Transmisor y Receptor Sincrónico/Asincrónico Universal. Sirve para transmitir datos en serie entre varios dispositivos, un microcontrolador con otro microcontrolador, microcontrolador con el pc (RS232 puerto serie), etc.
USB	Universal Serial Bus que en español significa Bus Universal en Serie. Es un subsistema que transfiere datos o electricidad entre componentes del ordenador dentro de un ordenador o entre ordenadores.

INTRODUCCIÓN

La prevención de posibles fallas en maquinarias es necesaria para una operación confiable y segura de una instalación. El riesgo de fallas y el tiempo en que una maquinaria queda fuera de servicio pueden disminuirse sólo si los problemas potenciales son anticipados y evitados.

En general, las vibraciones en una máquina no son buenas: pueden causar desgaste, fisuras por fatiga, pérdida de efectividad de sellos, rotura de aislantes, ruido, etc. Pero al mismo tiempo las vibraciones son la mejor indicación de la condición mecánica de una maquinaria y pueden ser una herramienta de predicción muy sensible de la evolución de un defecto.

En la actualidad contamos con sistemas avanzados para la colección de datos, procesamiento y transmisión de los mismos, es por tal motivo que en el presente proyecto se pretende implementar un prototipo con sistema de alarma para prevenir la problemática antes mencionada.

CAPÍTULO 1

ANTECEDENTES Y JUSTIFICACION

1.1 Antecedentes Teóricos

La comunicación serial funciona mediante el envío de ceros y unos de forma secuencial sobre uno o más cables, y es utilizada en la mayoría de las comunicaciones, frecuentemente utilizado en computadoras y los dispositivos que se conectan a la unidad central de proceso de la misma. La comunicación se da de emisor o maestro, a uno o varios receptores o esclavos, y puede ser Half Dúplex o Full Dúplex.

Existen dos formas de establecer una comunicación serial: en forma síncrona y en forma asíncrona. De forma asíncrona se refiere a la necesidad de sincronización por medio de hardware o software o ambos. El estándar comúnmente utilizado en este tipo de comunicación es RS-232 que establece niveles de voltaje, velocidad de transmisión de datos, cableado y los conectores que deben usarse, es normalmente utilizado para comunicaciones entre un dispositivo y una computadora.

A continuación se muestra un esquema básico de una comunicación asíncrona.

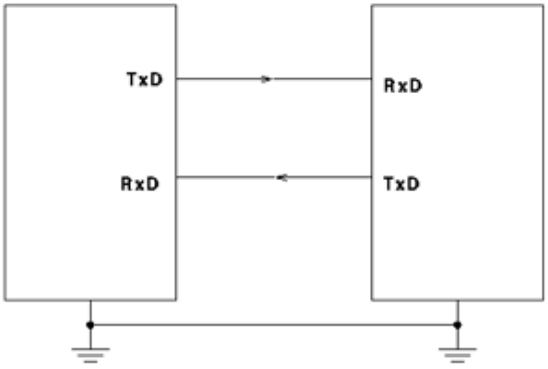

Figura 1.1 Esquema Básico de Comunicación Asíncrona

La comunicación síncrona existe cuando hay una señal de reloj común para los dispositivos maestro y esclavo, aunque los dispositivos esclavos no requieren de base de tiempo propia, ya que la obtienen del maestro. A continuación se muestra un esquema básico de una comunicación síncrona entre maestro y esclavo.

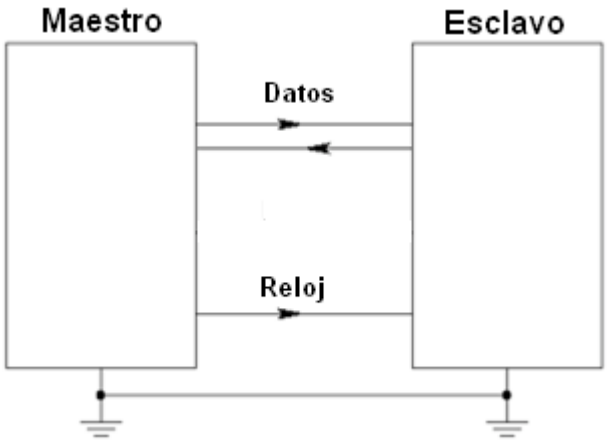

Figura 1.2 Esquema de Maestro a Esclavo

Se puede establecer la comunicación síncrona de dos formas diferentes: SPI e I2C. El estándar aplicado en este proyecto es SPI, el cual fue creado por Motorola y utiliza un bus de cuatro líneas para interconectar dispositivos de baja y media velocidad. La comunicación se realiza siguiendo un modelo maestro/esclavo donde el maestro selecciona al esclavo y comienza el proceso de transmisión o recepción de información. Se caracteriza por ser full dúplex o half dúplex, dependiendo de la necesidad, es decir, que se puede enviar y recibir información de manera simultánea, lo que constituye una elevación de la tasa de transferencia de datos.

1.2 Antecedentes de campo

Dentro del pensum curricular de la carrera de Ingeniería en Electrónica y Telecomunicaciones, se encuentra la enseñanza de los microcontroladores, en especial del microcontrolador PIC 16F887, como uno de los dispositivos más básicos y al mismo tiempo ideal para entender el funcionamiento de los microcontroladores. Cada clase fue complementada por su respectiva práctica de laboratorio, en la que se ejecutaba lo aprendido, para el final del currículo de la materia se realizó la comunicación serial asíncrona bajo el protocolo RS232 vía USART entre el microcontrolador 16F887 y una computadora, así como la comunicación síncrona I2C entre el microcontrolador 16F887 y una memoria EEPROM. Mas, sin embargo, la

comunicación bajo el protocolo SPI, no fue cubierta, es por eso que vimos la oportunidad de aplicarla como precedente para futuras generaciones.

1.3 Motivación del proyecto

El ruido que muchas veces se produce por las vibraciones en los componentes electrónicos nos revela el comportamiento de éstos, además ayuda a comprobar la correcta operación del mismo. Por tal motivo, hemos analizado la posibilidad de implementar un prototipo que genere una alarma visual de acuerdo al ruido generado por la vibración de un motor en mal estado.

1.4 Justificación del proyecto

Esta tesina es elaborada con el fin de desarrollar un sistema que permita determinar cuándo se debe tomar las medidas preventivas y/o correctivas al existir un ruido constante y excesivo generado en los motores. Para esto tomamos en consideración los antecedentes de campo en nuestra carrera en relación a la comunicación síncrona SPI, así como la oportunidad de trabajar con microcontroladores de mayor complejidad y capacidad comparados a los utilizados en las prácticas de laboratorio de la materia Microcontroladores. En la implementación se ejecuta la captura de datos del sensor de impacto y sonido, para la transmisión de éstos desde la tarjeta LPC1769 hacia el kit de

desarrollo AVR Butterfly usando la comunicación mencionada, a fin de visualizar alertas necesarias.

1.5 Objetivo General

Implementar un sistema de análisis de vibraciones generadas por ruido de un motor BLDC mediante la comunicación serial síncrona, SPI, entre dos dispositivos con la finalidad de que a partir del análisis el usuario tome las medidas necesarias.

1.6 Objetivos específicos

- Aplicar la comunicación serial síncrona bajo el esquema maestro esclavo, entre dos tarjetas LPC1769, para familiarizarnos con el funcionamiento de la misma.

- Integrar los datos obtenidos por el sensor de impacto y sonido, a la comunicación entre la tarjeta LPC1769 y el kit de desarrollo AVR Butterfly.

- Complementar el proyecto con alarmas lumínicas de acuerdo al tiempo de vibración del motor y a los parámetros especificados por el usuario.

1.7 Limitaciones

Entre las limitaciones físicas del proyecto, se debe conocer que este es un prototipo, no es aplicable a la industria, si se desea una aplicación más detallada se debe realizar los cambios respectivos.

El motor BLDC, el cual se utilizó como sujeto de estudio para la implementación de este proyecto, es relativamente nuevo y no ha tenido un uso frecuente, por ende no genera ruidos por desperfectos en su funcionamiento. Esto es una limitante debido a que usamos ruidos adicionales para comprobar el funcionamiento del mismo.

Hay que destacar que para la mayoría de los proyectos académicos nos apoyamos en las herramientas de simulación brindadas por varios programas, pero la tarjeta LPC1769, al ser un producto relativamente nuevo, no se ha encontrado una herramienta para la simulación de su funcionamiento.

CAPÍTULO 2

MARCO TEÓRICO

En este capítulo, se realiza la introducción de los conceptos generales de la comunicación síncrona en modo de operación SPI, sobre los que se fundamenta la tesina desarrollada, a continuación tendremos los detalles de cada herramienta utilizada para realizar la implementación del proyecto, entre los cuales tenemos el Sensor de Impacto y Sonido 29132, la tarjeta LPC1769 y el kit de desarrollo AVR Butterfly. Además de las herramientas físicas antes mencionadas, utilizamos lenguajes de programación para lo cual requerimos de dos tipos de programas como: LPCXPRESSO para la tarjeta LCP1769 y AVR Studio Versión 4.0 para el kit de desarrollo AVR Butterfly, en los cuales utilizamos el lenguaje C. Finalmente tenemos el Motor BLDC del cual obtendremos los datos de las vibraciones que está receptando el sensor para la debida transmisión y análisis del mismo.

2.1 SPI

SPI es un estándar que utiliza un bus de cuatro líneas y permite la transmisión sincrónica de datos entre un dispositivo maestro y uno o varios dispositivos esclavos. Es implementado por un módulo de hardware llamado SSP, este módulo permite la comunicación serial síncrona entre dos o más dispositivos a una alta velocidad y es razonablemente fácil de implementar, además que permite una trasferencia de datos no limitada por bloques de 8 bits.

Líneas de control y datos de SPI

- SCK: El dispositivo maestro provee una señal de reloj para la sincronización, ésta controla cuando los datos pueden ser enviados y cuando deben ser leídos, a esta señal se le da el nombre de SCK. Como SPI es síncrono, los datos se envían por cada pulso de reloj y la frecuencia del mismo puede variar sin interrumpir la transmisión de los datos, la velocidad de transmisión cambiará al mismo tiempo que varíe la frecuencia. En SPI solo el dispositivo maestro puede controlar la señal de reloj SCK, ningún dato se transferirá a menos que la señal de reloj no esté presente.

- SS: Usualmente una señal selectora controlará cuando un dispositivo esclavo debe permanecer en la transmisión. Esta señal debe usarse

cuando existe uno o más dispositivos esclavos en el sistema pero puede ser opcional cuando solamente existe uno. Esta señal es conocida como SS y significa selector de esclavo (Slave Select). Con esta señal el dispositivo maestro le indica al esclavo que desea iniciar la transmisión SPI, para esto el maestro baja la señal SS a cero voltios, como resultado el dispositivo esclavo se activa; mientras que a niveles de voltaje superior el dispositivo va a estar inactivo.

- MOSI: Esta línea está definida como salida desde el dispositivo maestro y entrada del dispositivo esclavo. Los datos son transmitidos en una sola dirección desde el maestro hacia el esclavo.

- MISO: Esta línea está definida como entrada desde el dispositivo maestro y salida del dispositivo esclavo. Los datos son transmitidos en dirección opuesta al MOSI, es decir desde el esclavo hacia el maestro.

2.2 Sensor de Impacto de Sonido 29132

Un sensor es un dispositivo diseñado para detectar variaciones de una magnitud física y transformarla en señales útiles capaces de cuantificar y manipular. El sensor, visto en la Figura 2.1, que utilizamos tiene un rango de detección de sonidos de hasta 3 metros de distancia ajustable por un potenciómetro, que en conjunto con un micrófono, el cual recepta los

sonidos, emitirá una señal digital con duración de tiempo igual a la del sonido en la terminal de salida del sensor. El voltaje de alimentación es de 5 voltios.

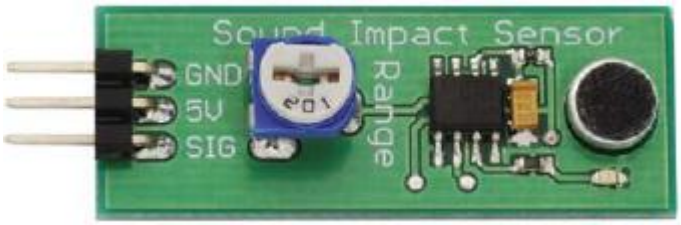

Figura2. 1 Sensor de Impacto de Sonido [8]

2.3 Tarjeta LPC1769

La tarjeta LPC1769 consta de dos partes: el microcontrolador LPC1769 y un depurador JTAG, como se puede observar en la Figura 2.2. La porción de depuración JTAG de la tarjeta LPC se llama LPC-Link. La LPC-Link está conformada por una cabecera de 10 pines JTAG, y se conecta con otros dispositivos a través de la interfaz USB.

Figura2. 2 Tarjeta LPC1769 [9]

El microcontrolador consta de 100 pines los cuales están distribuidos en su contorno (Figura 2.3), esta tarjeta cuenta con 3 puertos configurables (GPIO0, GPIO1, GPIO2) como entradas o salidas de un sistema según sea el caso. Para conocimiento de las funciones de cada uno de estos puertos se puede consultar el manual de usuario el cual se encuentra en la bibliografía de esta tesina.

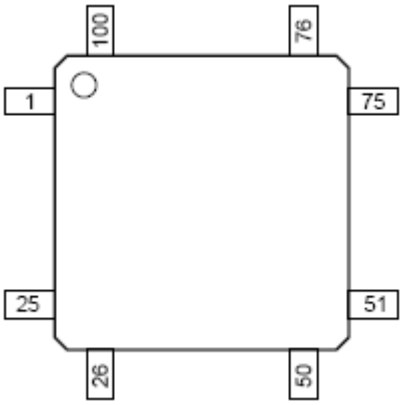

Figura2. 3 Distribución Pines Tarjeta LPC1769

2.4 Kit de Desarrollo AVR Butterfly

El kit de desarrollo AVR Butterfly está diseñado para demostrar los beneficios y características claves de los microcontroladores ATMEL, utiliza el microcontrolador ATmega169 y posee partes y aplicaciones que lo distinguen de la tarjeta LPC1769, tales como un display LCD para mostrar información, reproducción de música, sensor de temperatura y un joystick. El kit se alimenta con tres voltios y posee varios conectores para comunicación y programación, como se puede ver en la Figura 2.4 (editar figura).

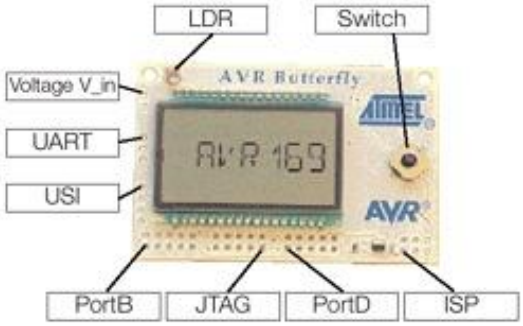

Figura2. 4 Kit de Desarrollo AVR Butterfly [2]

El ATmega169 es un microcontrolador de 8 bits con arquitectura AVR RISC, y es compatible con un completo conjunto de programas y elementos de desarrollo, como compiladores de lenguaje de programación C, ensambladores de macro, simuladores de programas, emuladores de circuito y kits de evaluación.

2.5 LPCxpresso

El LPCXpresso es un programa para evaluación con microcontroladores LPC de NXP, el cual incluye todas las herramientas necesarias para desarrollar una alta cualidad de soluciones de manera efectiva en tiempo y costo. Consiste de un editor de texto para editar código fuente, un compilador y enlazador para transformar el código fuente en un programa ejecutable y un depurador.

También cuenta con la última versión de la cadena de herramientas GNU estándar con una biblioteca propia optimizado en lenguaje de programación

C. Además el IDE LPCXpresso puede construir un ejecutable de cualquier tamaño con la optimización del código completo.

2.6 AVR Studio 4

AVR Studio es un Entorno de Desarrollo Integrado (IDE) para escribir y depurar aplicaciones AVR en el sistema operativo de Windows, tiene una arquitectura modular que permite interactuar con programas de otros fabricantes y soporta todas las herramientas de ATMEL que apoyan a la arquitectura AVR 8 bits.

Para la compilación del código hecho en el AVR Studio se utiliza la distribución WinAvr, que es una recopilación de programas de software libre diseñados para facilitar las tareas de programación y desarrollo de los microcontroladores AVR. Dicha distribución WinAvr incorpora además del compilador GCC de consola, un editor de texto especialmente diseñado para ayudar al programador y hacer el código más legible mediante su resaltado con colores.

2.7 Motores BLDC

Un motor BLDC sin escobillas es una máquina síncrona con la frecuencia de alimentación, capaz de desarrollar altos torques en forma transitoria para oponerse a todo esfuerzo que trate de sacarla de sincronismo. Hoy en día el

motor BLDC es utilizado en muchos sectores desde implementaciones de forma capsular, es decir micro robots quirúrgicos, hasta aplicaciones de gran escala industrial y automotriz

El motor BLDC (Figura 2.5) cuenta con las ventajas de una mayor duración dado que no existe rozamiento en las escobillas, que a su vez se traduce en una emisión mucho menor de interferencias electromagnéticas. Otra de las ventajas, es el mayor control sobre la velocidad del motor y su posición. A la vez, permite introducir un mayor número de elementos sensores en el motor, por lo que se pueden realizar controles, no sólo de velocidad y sentido, sino también de errores entre otros parámetros.

Figura2. 5 Motor BLDC [10]

Estas notables ventajas, tales como la inexistente caída de tensión (y pérdida de energía) entre las escobillas y las delgas de un colector, el bajo ruido generado durante el funcionamiento o las bajas vibraciones mecánicas,

logradas mediante un desarrollo más cuidado, permiten una mejor eficiencia del sistema en general, una mayor fiabilidad de duración y un mejor rendimiento energético [11].

CAPÍTULO 3

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

En este capítulo se examinan los ejercicios que realizamos con el objetivo de familiarizarnos con la tarjeta LPC1769 y el kit de Desarrollo AVR Butterfly, así como de sus IDE's. Se realizaron dos ejercicios para cumplir con los objetivos específicos del proyecto.

Para la descripción de los ejercicios nos regimos bajo los siguientes esquemas: Diagrama de Bloques y Diagrama de Flujo. El diagrama de bloques es la representación gráfica del funcionamiento interno del sistema, que se hace mediante bloques y sus relaciones, y además, define la organización de todo el proceso interno, sus entradas y sus salidas. El diagrama de flujo describe qué operaciones y la secuencia que se requiere para solucionar un problema dado.

3.1 Ejercicio 1: Secuencia de encendido de LEDS

En este ejercicio, la tarjeta LPC1769 realiza un encendido secuencial de 8 LED's, comenzará con una secuencia definida y una vez que terminó de encender los ocho leds, los apaga para continuar con la misma secuencia, en el momento, y durante el tiempo, que se presione la botonera, se apagan los LED's y comienza una secuencia opuesta.

En la Figura 3.1 se muestra el diagrama de bloques correspondiente a este ejercicio, como podemos observar consta de 3 bloques. En el primer bloque tenemos la botonera que es entrada al sistema y que dependiendo de su estado, se procesa en el segundo bloque, tarjeta LPC1769, para establecer la dirección de la secuencia de encendido en el bloque de salida asignado a los LED'S.

Figura3. 1 Diagrama de Bloques Ejercicio 1

En el diagrama de flujo, que tenemos en la Figura 3.2, se detalla el algoritmo usado. Comenzado por definir las variables globales del sistema y asignando un puerto de la tarjeta LPC1769 para las variables de entrada y salida. Para finalizar, se procede a evaluar el estado de la botonera para definir la

dirección de la secuencia de encendido de los LED’s, esto se realizará mientras el sistema este energizado.

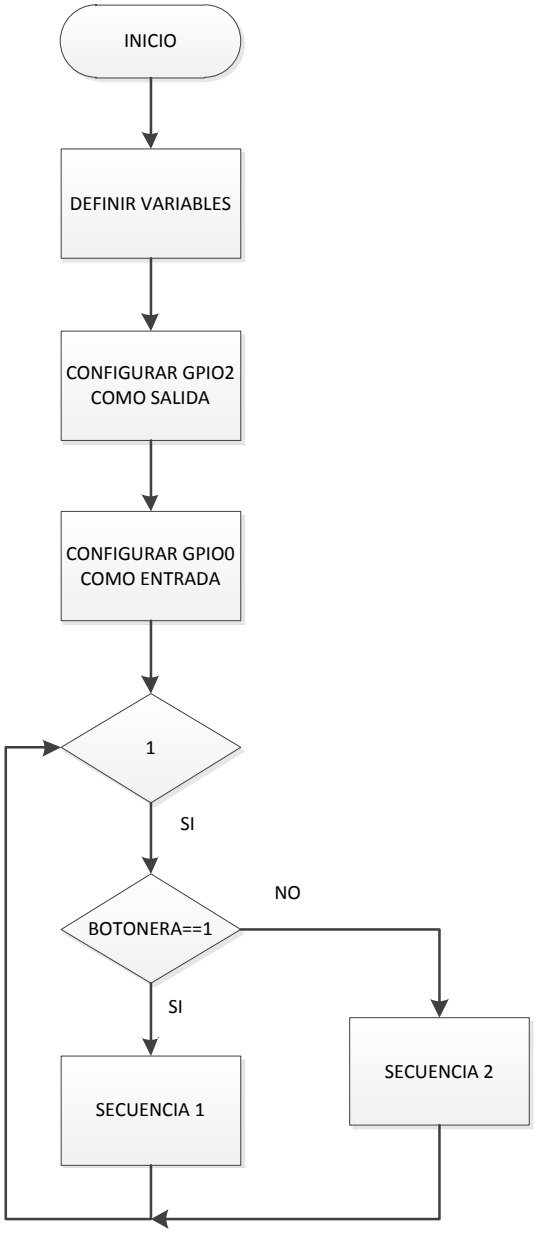

Figura3. 2 Diagrama de Flujo Ejercicio 1

Para realizar la secuencia de encendido en diferentes direcciones se crearon dos bucles que toman efecto bajo la condición de la variable que representa la botonera, la cual está simbolizada por la instrucción “If”, estos bucles se observan en la parte a) y b) de la Figura 3.3.

```
while(1)
  {
    if(LPC_GPIO0->FIOPIN0==0xff)
      {
        for(i = 0; i< 8; i++)
          {
              LPC_GPIO2->FIOSET = 1 <<i; // prende cada led
            for(j = 1000000; j > 0; j--); //Retardo
          }
          LPC_GPIO2->FIOCLR = 0x000000FF;
        for(j = 1000000; j > 0; j--);
    }
    else
      {
        for( k= 7; k >= 0 &&  k<=7; k--)
          {
              LPC_GPIO2->FIOSET = 1 << k;
            for(j = 1000000; j > 0; j--);
          }
          LPC_GPIO2->FIOCLR = 0x000000FF;for(j = 1000000; j > 0; j--);
    }
```

a)

b)

Figura3. 3 Código Bucles Ejercicio 1

Para lectura del código completo utilizado en este ejercicio se puede consultar el Anexo A.

3.2 Ejercicio 2: Comunicación SPI entre dos tarjetas LPC1769

El principal fundamento de este ejercicio es realizar la comunicación SPI entre dos tarjetas LPC1769, para esto se transmitirán datos de una tabla

definida en el código fuente del dispositivo maestro y serán visualizados en un display de siete segmentos conectado en la tarjeta establecida como esclavo.

En la Figura 3.4 se muestra el diagrama de bloques correspondiente a este ejercicio, como podemos observar consta de 3 bloques. En el primer bloque tenemos la tarjeta LPC1769 que tomará la función de maestro o emisor para el segundo bloque que consta de otra tarjeta LPC1769 que toma la función de esclavo o receptor, y con la cual existirá la comunicación SPI. Como último bloque tenemos un display de 7 segmentos para visualización de los datos recibidos.

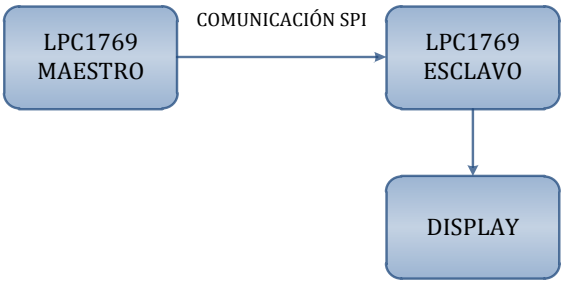

Figura3. 4 Diagrama de Bloques Ejercicio 2

En el diagrama de flujo que tenemos en la Figura 3.5 se detalla el algoritmo usado para el dispositivo maestro. Comenzado por definir las variables globales del sistema, dentro de las cuales se encuentra un arreglo de diez datos que serán transmitidos, inmediatamente se activa la interfaz SPI en la

tarjeta LPC1769. Para el envío de los datos del arreglo se utiliza una variable contador, que aumentará conforme cada dato se envíe y se reiniciará para volverlos a enviar, esto se realizará mientras el sistema este energizado.

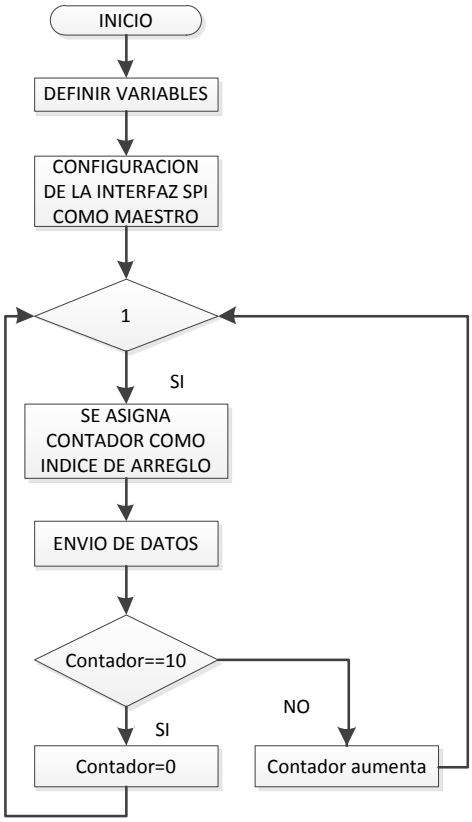

Figura3. 5 Diagrama de Flujo Dispositivo Maestro Ejercicio 2

A continuación, en la Figura 3.6, mostramos la tabla definida en la tarjeta LPC1769 emisora, en la cual tenemos la asignación de bits para la activación de los segmentos de un display ánodo común, conectado a un puerto de salida de la tarjeta mencionada. La utilización de esta tabla nos ayudará para

la visualización de los dígitos del cero al nueve, que posteriormente será usada como contador.

```
static const uint8_t segmentLUT[10]=
{
        //FCPBAGED para anodo comun
                (uint8_t) 0b11011011, // 0
                (uint8_t) 0b01010000, // 1
                (uint8_t) 0b00011111, // 2
                (uint8_t) 0b01011101, // 3
                (uint8_t) 0b11010100, // 4
                (uint8_t) 0b11001101, // 5
                (uint8_t) 0b11001111, // 6
                (uint8_t) 0b01011000, // 7
                (uint8_t) 0b11011111, // 8
                (uint8_t) 0b11011101, // 9

};
```

Figura3. 6 Tabla del Codigo para visualizacion en display

Para realizar el envio de los datos utilizamos funciones existentes en ejemplos proporcionados por NXP que es el desarrollador del programa LPCXpresso, como se puede observar en la Figura 3.7 se inicializa la interfaz SPI y luego se asigna una variable a un dato del arreglo, con la ayuda del contador *"i"*, e inmediatamente transmitirlo. Esto se ejecutará en un bucle infinito.

```
while ( 1 )
  {
    SSP1Init();
    src_addr[0] = (uint8_t)segmentLUT[i];
    SSPSend(portnum, (uint8_t *)src_addr, 1);
    i=i+1;
    if (i==10) i=0;
  }
```

Figura3. 7 Código de transmisión de datos

En el diagrama de flujo que tenemos en la Figura 3.8 se detalla el algoritmo usado para el dispositivo esclavo. Comenzado por definir las variables globales del sistema, asignando un puerto de la tarjeta LPC1769 para la variable de salida, e inmediatamente activar la interfaz SPI en la tarjeta. Se utiliza una variable para la recepción de los datos y visualización en un display de siete segmentos, esto se realizará mientras el sistema este energizado.

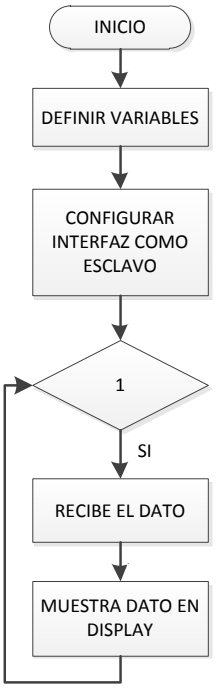

Figura3. 8 Diagrama de Flujo Dispositivo Esclavo Ejercicio 2

Para realizar recepción de los datos utilizamos funciones existentes en ejemplos proporcionados por NXP que es el desarrollador del programa

LPCXpresso, como se puede observar en la Figura 3.9 se inicializa la interfaz SPI y luego se recibe el dato en la variable “dest_addr”, también utilizada como salida en el puerto GPIO2, el cual está configurado como salida para mostrar los datos en un display. Esto se ejecutará en un bucle infinito.

```
while ( 1 )
  {
    SSP1Init();
    SSPReceive (portnum, (uint8_t *)dest_addr, 1 );
    LPC_GPIO2->FIOSET = dest_addr[0];
    for(j = 1000000; j > 0; j--);
    LPC_GPIO2->FIOCLR = 0x000000FF;
    for (j=90100;j>0;j--);
  }
```

Figura3. 9 Código de transmisión de datos

Para lectura del código completo utilizado en este ejercicio se puede consultar el Anexo B.

3.3 Almacenamiento de datos de vibraciones de motor BLDC para graficación y análisis en displays disponibles en tarjeta AVR Butterfly y en tarjeta controladora LPCXpresso.

En este ejercicio se desarrolla el proyecto completo de la tesina, el cual consiste en la transmisión de datos recibidos de un sensor de impacto y sonido en la tarjeta LPC1769 hacia el kit de desarrollo AVR Butterfly, el cual contiene un microcontrolador ATmega169 que está programada para recibir los datos. La LCD mostrará por cada dato recibido un mensaje que nos

indica el estado del motor, los mensajes que se muestran son: “NORMAL” y “ALERTA”.

En el estado de “NORMAL” se reflejará un comportamiento adecuado del motor; mientras que cuando ocurra lo contrario, inmediatamente se mostrará el mensaje “ALERTA” y en un display de 7 segmentos comenzará un contador que aumenta si el ruido producido por el motor no cesa, en caso de llegar al número 5, entonces se activarán unos leds en forma de alarma.

En la Figura 3.10 se muestra el diagrama de bloques correspondiente al proyecto de presente tesina, como podemos observar consta de 3 bloques importantes. En el primer bloque tenemos al sensor de Impacto y Sonido el cual detecta los ruidos generados por el motor y actúa como entrada para el segundo bloque que consta de una tarjeta LPC1769 que toma la función de maestro o emisor, y con la cual existe la comunicación SPI con el tercer bloque, representado por el kit de desarrollo AVR Butterfly que actúa como receptor o esclavo. Para la visualización al usuario de una alarma tenemos los bloques: LCD, Display y LED’s.

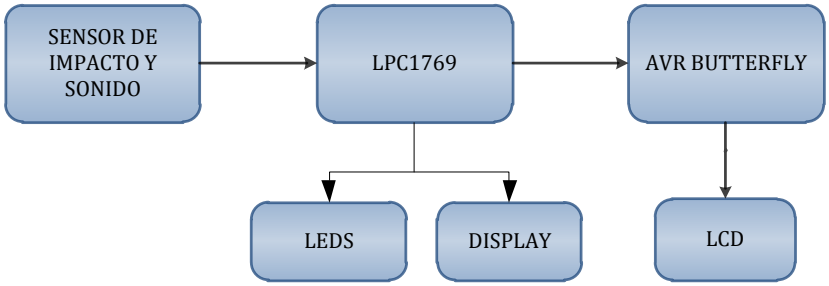

Figura3. 10 Diagrama de Bloques Proyecto

En el diagrama de flujo que tenemos en la Figura 3.11 se detalla el algoritmo usado para el dispositivo maestro del proyecto. Comenzado por definir las variables globales del sistema, asignando dos puertos de la tarjeta LPC1769 para las variables de entrada y salida, y activando la interfaz SPI en la tarjeta. Se recepta un dato con el sensor, el cual es enviado a través de la comunicación SPI y que además es considerado, de acuerdo a su valor, para la activación de una alarma. Esta alarma funciona de la siguiente manera: si el dato sensado no es considerado alerta, continúa con el proceso de recepción y envío de datos, caso contrario se visualizará en un display un contador que aumenta de valor mientras los datos receptados sean motivo de alerta para el sistema; cuando el contador llegue al número cinco se activan LED’s bajo el mismo parámetro que aumenta el contador, es decir, se vuelve a receptar un dato y se verifica que continúe siendo considerado alerta para el sistema, caso contrario se desactivarán los leds y se retorna al estado inicial.

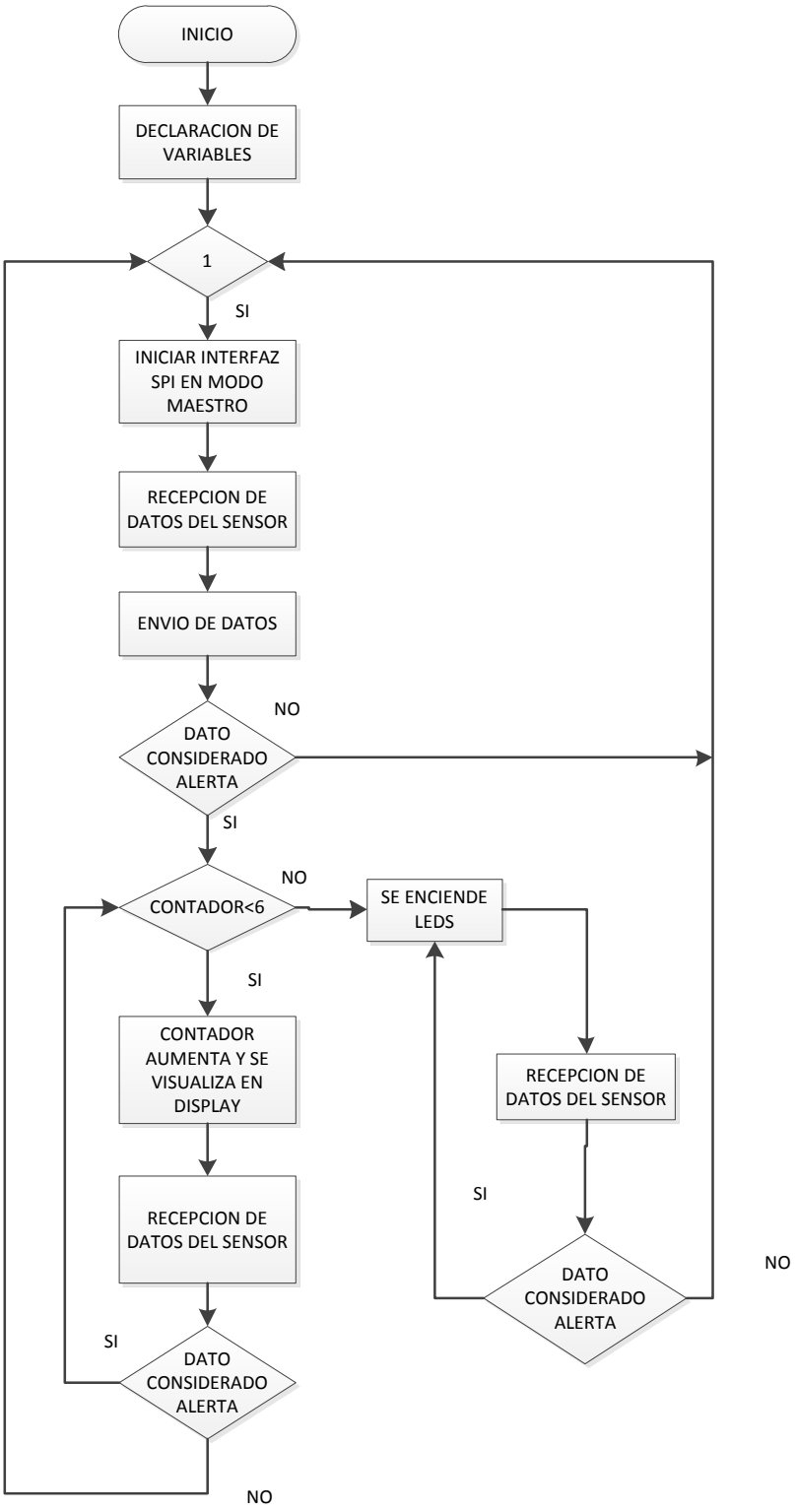

Figura3. 11 Diagrama de Flujo Dispositivo Maestro del Proyecto

En el diagrama de flujo que tenemos en la Figura 3.12 se detalla el algoritmo usado para el dispositivo esclavo del proyecto, kit de desarrollo AVR Butterfly. Comenzado por definir las variables globales del sistema, habilitando el puerto B para la interfaz SPI, así como la LCD. Se procede, entonces, a la recepción de datos por la interfaz SPI, y con cada dato receptado, de acuerdo a su valor, se muestran los mensajes de “NORMAL” o “ALERTA” en la LCD.

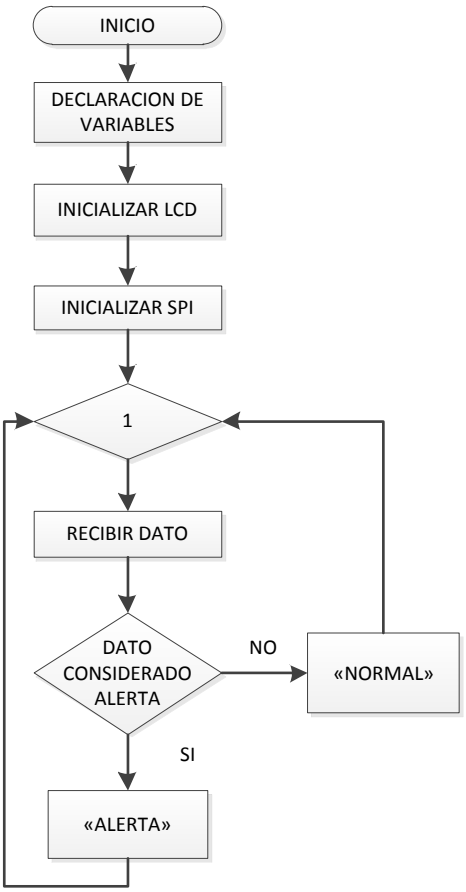

Figura3. 12 Diagrama de Flujo Dispositivo Esclavo del Proyecto

El código presentado en la Figura 3.13, corresponde a los siguientes procesos: el envío del dato recibido por el sensor y dos bucles que se ejecutan de acuerdo a un valor considerado alerta, en el primer bucle se establece un contador que llegará al número cinco, mientras que los datos receptados continúen siendo “alerta”, en el segundo bucle se ejecuta el encendido de leds, considerando el mismo criterio que en el primer bucle.

```
SSPSend( portnum, (uint8_t *)datosensor, 1);
 if (datosensor[0]>35){
     for(i = 0; i< 6; i++)
         {LPC_GPIO2->FIOSET = (uint8_t)segmentLUT[i];//muestra numero en display
         for(j = 10000000; j > 0; j--);//Retardo
         LPC_GPIO2->FIOCLR = 0x000000FF;//apaga display
         for(j = 1000000; j > 0; j--);
         ValorADC[0] = ADCRead(0);
         datosensor[0] = ValorADC[0]/100;
         if (datosensor[0]<35) break;
         }
 }
 while (datosensor[0]>35){
     for(i = 0; i< 2; i++)
         {LPC_GPIO0->FIOSET = 1 <<i; // encendido de led's
         for(j = 1000000; j > 0; j--); //Retardo
         LPC_GPIO0->FIOCLR = 0x000000FF;//apagado de led's
         for(j = 1000000; j > 0; j--);
         }
         ValorADC[0] = ADCRead(0);
         datosensor[0] = ValorADC[0]/100;
     }
 }
```

Figura3. 13 Código Dispositivo Maestro Envío de datos y Alarma

En el código presente en la Figura 3.14, se observa la recepción continua de datos en una variable “input”, esta variable se compara con un valor asignado

como alerta y del resultado de la comparación se mostrarán un mensaje en la pantalla LCD del kit de desarrollo AVR Butterfly.

```
while (1)
{
input=SPI_slave_receive();
switch (input) {
      case 0:
         statetext = PSTR("NORMAL");
         break;
      case 35:
         statetext = PSTR("ALERTA");
         _delay_ms(2);
         break;
      default:
         break;
         }
}
```

Figura3. 14 Código Dispositivo Maestro Envío de datos y Alarma

Para lectura del código completo utilizado en este ejercicio se puede consultar el Anexo C.

CAPÍTULO 4

PRUEBAS Y SIMULACIONES

En el contenido de este capítulo se detallan los diagramas esquemáticos de cada ejercicio realizado, los materiales que usamos y una imagen del ejercicio funcionando.

Para los diagramas de conexiones nos basamos en los esquemáticos de la tarjeta LPC1769 y del kit de desarrollo AVR Butterfly, que en conjunto con los manuales de usuario se explican las funcionalidades de cada pin. Los materiales que usamos son de uso común que se encuentran en cualquier establecimiento de venta de dispositivos de electrónica.

4.1 Ejercicio 1: Rotación de LEDS

El diagrama de bloques de este ejercicio, explicado en el capítulo anterior, indica que la salida del sistema es representada por LED's (Figura 3.1), los cuales están conectados a los pines desde el 42 al 49 de la tarjeta LPC1769

que pertenecen al puerto GPIO2, Figura 4.1, también tenemos una botonera que es entrada del sistema conectada al pin 9 de la tarjeta LPC1769.

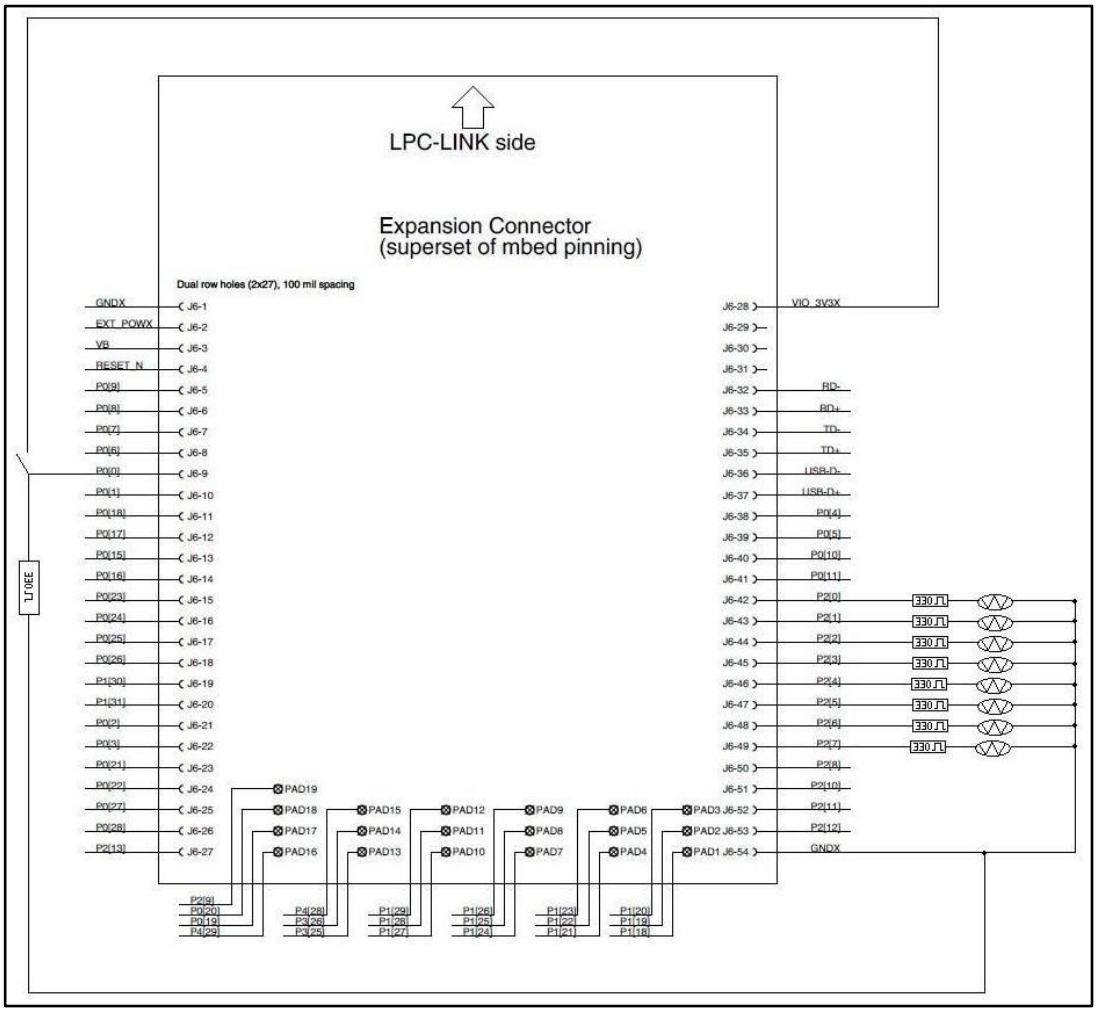

Figura4. 1 Diagrama de Conexiones del Ejercicio 1

Para la implementación del ejercicio se utilizaron los siguientes materiales:

- LPC1769 (Figura 4.2a).
- 8 leds rojos (Figura 4.2b).

- 8 resistencias de 330 Ω (Figura 4.2c).
- Cable Adaptador USB (Figura 4.2d).
- Botonera (Figura 4.2e).

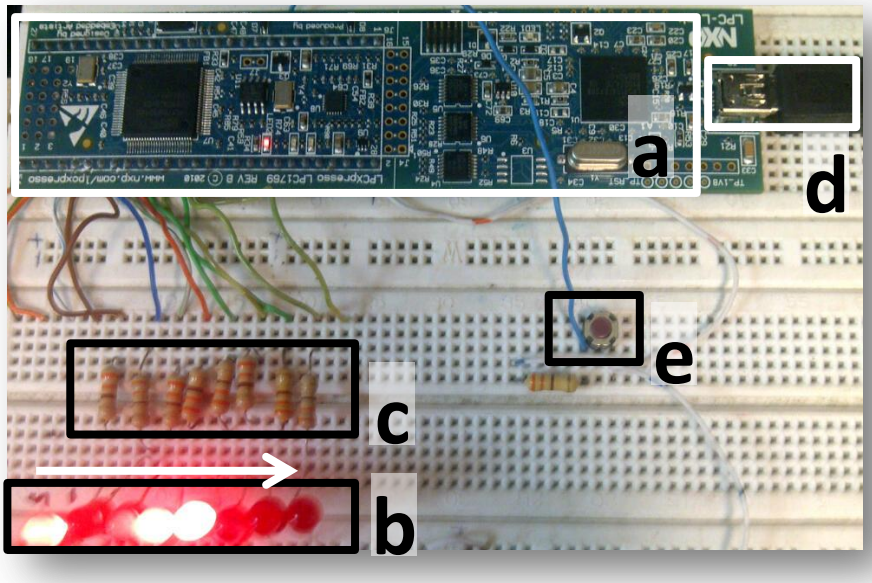

Figura4. 2 Secuencia Normal

Así pues, como observamos en la Figura 4.2, la secuencia de encendido de los LED's toma una dirección de izquierda a derecha hasta que se presione la botonera (Figura 4.3b), que iniciará una secuencia inversa, es decir de derecha a izquierda (Figura 4.3a).

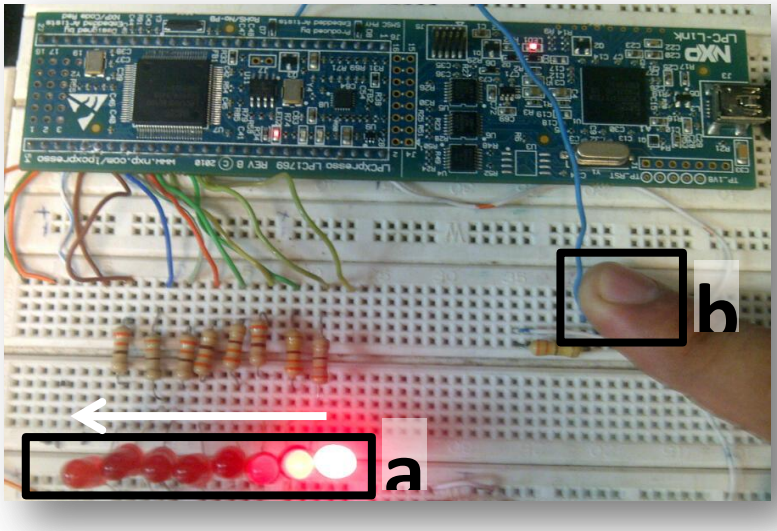

Figura4. 3 Secuencia inversa presionando botonera

4.2 Ejercicio 2: Comunicación SPI entre dos tarjetas LPC1769

Con respecto al diagrama de bloques de este ejercicio, explicado en el capítulo tres, indica que la salida del sistema es representada por un display de siete segmentos (Figura 3.4), conectado a los pines desde el 42 al 49 de la tarjeta LPC1769, dispositivo esclavo del sistema, y pertenecen al puerto GPIO2, Figura 4.4. También tenemos resistencias que unen los pines del 5 al 8 desde el dispositivo maestro hacia el esclavo, estos pines pertenecen al puerto GPIO0, configurados como la interfaz SPI de ambas tarjetas; que sirven de protección de voltaje y corriente. Finalmente, observamos que los dispositivos que forman parte de este sistema comparten el mismo nivel de referencia.

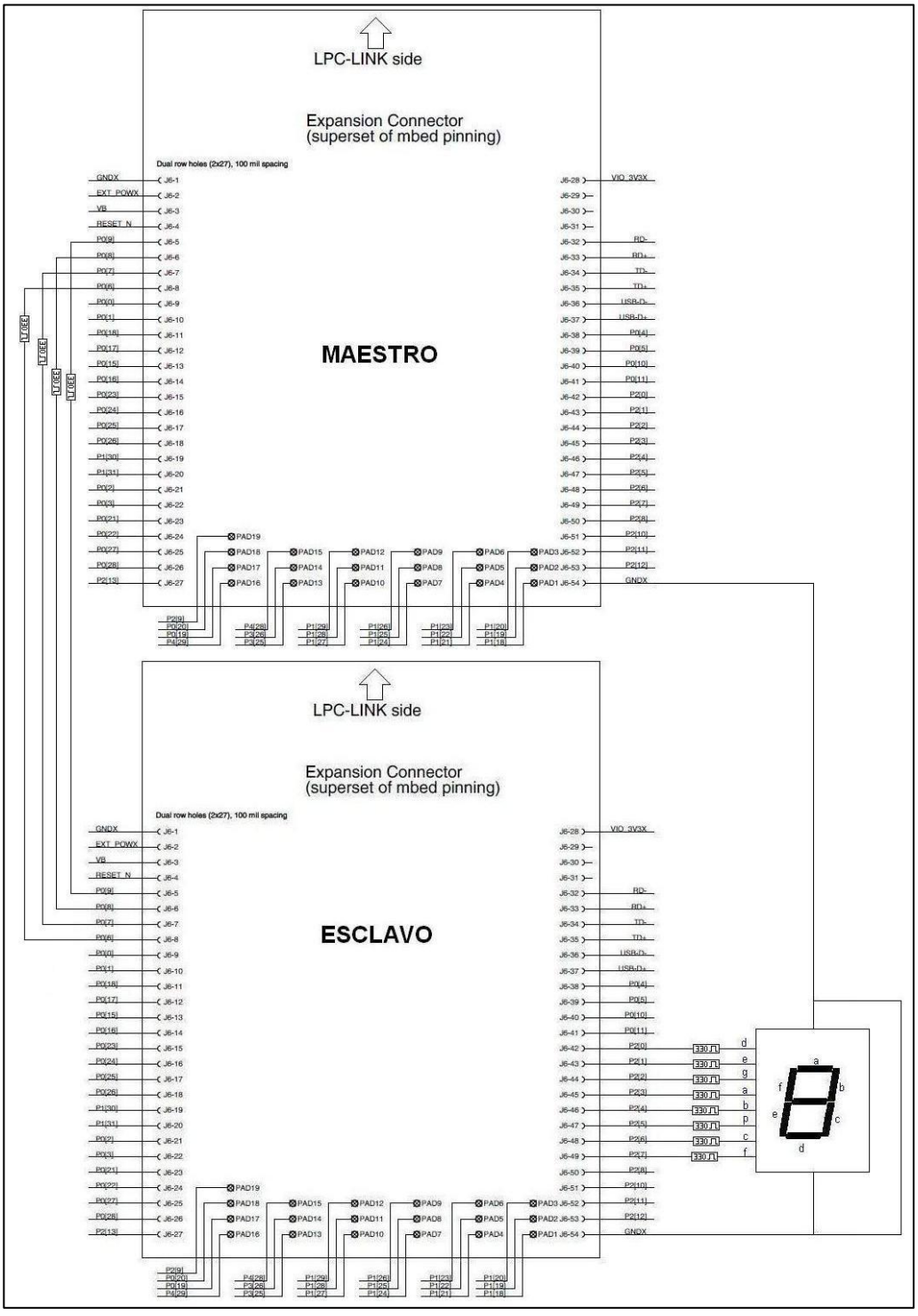

Figura4. 4 Diagrama de Conexiones del Ejercicio 2

Para la implementación del ejercicio se utilizaron los siguientes materiales:

- 2 Tarjetas LPC1769 (Figura 4.5 a y Figura 4.5b).
- 1 Display 7 segmentos (Figura 4.5c).
- 2 Cables Adaptador USB (Figura 4.5d).
- 12 resistencias de 330 Ω (Figura 4.5e).

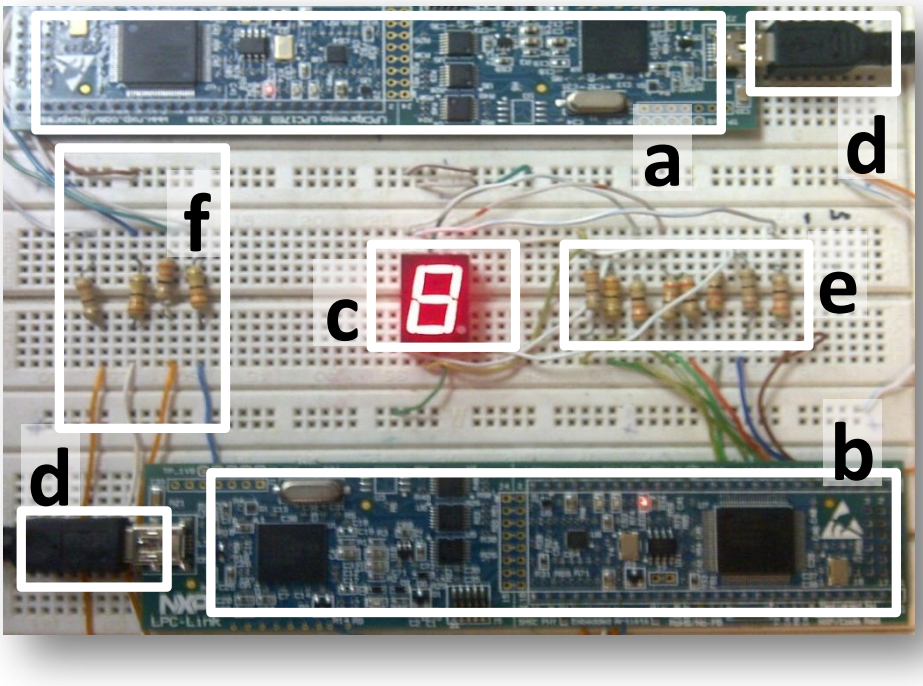

Figura4. 5 Transmisión SPI

Hay que destacar de la Figura 4.5 los siguientes puntos: la parte ***a*** corresponde al dispositivo maestro, la ***b*** al dispositivo esclavo al cual está conectado el display de siete segmentos por medio de unas resistencias para

protección del mismo. La conexión para la comunicación SPI entre maestro-esclavo la observamos en la parte ***f***, relacionados a las líneas de control: MISO, MOSI, SCK y SS. Finalmente, la salida del sistema la observamos en la parte ***c***, que en este caso está mostrando el número 6.

4.3 Almacenamiento de datos de vibraciones de motor BLDC para grafica ción y análisis en displays disponibles en tarjeta AVR Butterfly y en tarjeta controladora LPCXpresso

En relación con el diagrama de bloques del proyecto, explicado en el capítulo anterior (Figura 3.10), las salidas del sistema están representadas por: un display de siete segmentos, conectado a los pines desde el 42 al 49 de la tarjeta LPC1769, dispositivo maestro del sistema, y pertenecen al puerto GPIO2, Figura 4.6; así como un par de LED's bicolores conectados a los pines 9 y 10 de la tarjeta LPC1769, también la LCD del kit de desarrollo AVR Butterfly que internamente está conectada con el microcontrolador ATMEGA169. Otro punto, es la conexión del sensor de impacto y sonido a la tarjeta LPC1769 en el pin 15 correspondiente al puerto GPIO0. Finalmente, observamos que los dispositivos que forman parte de este sistema comparten el mismo nivel de referencia.

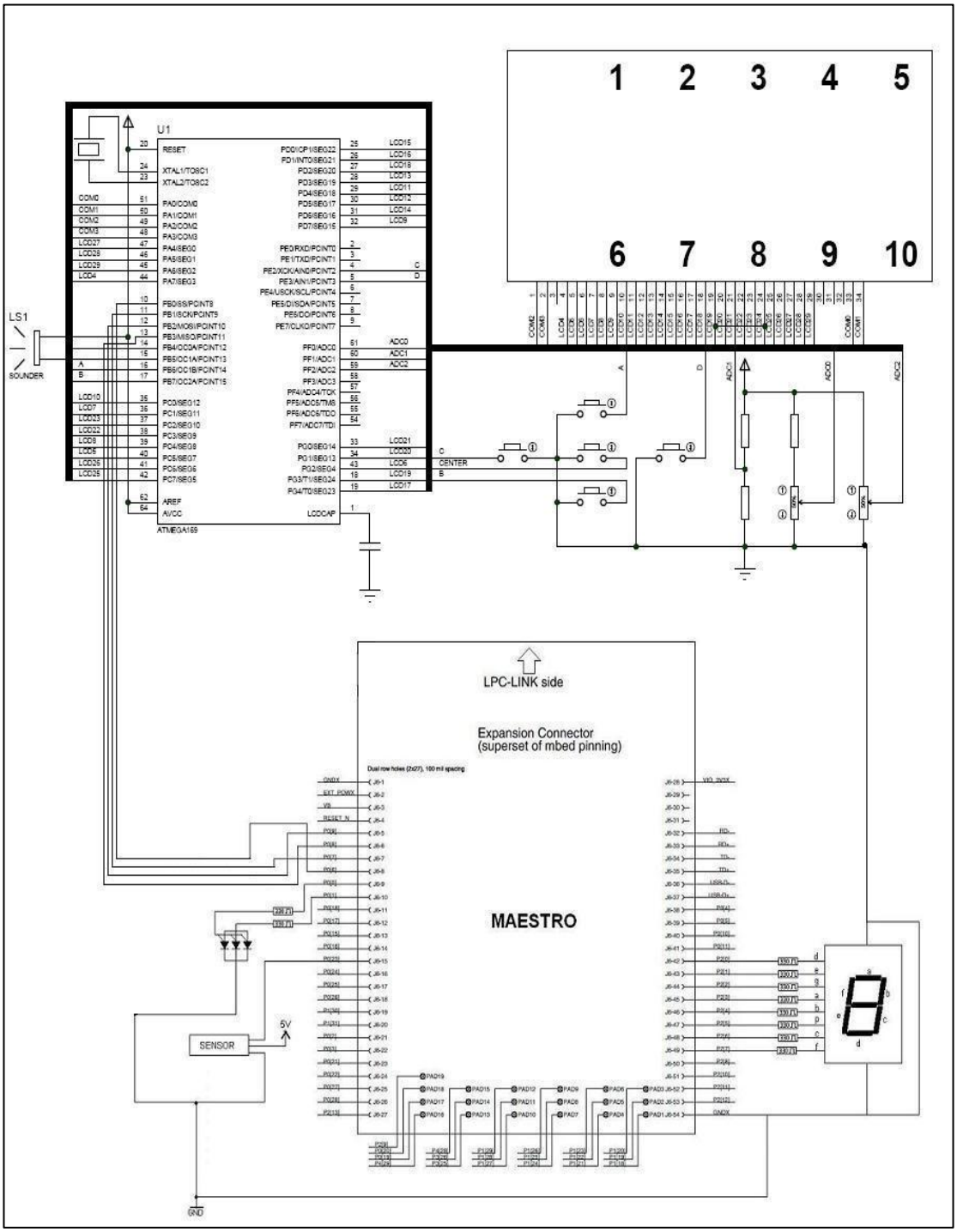

Figura4. 6 Diagrama de Conexiones del proyecto final

Para la implementación del proyecto se utilizaron los siguientes materiales:

- 1 Tarjeta LPC1769 (Figura 4.7a).
- 1 kit de desarrollo AVR Butterfly (Figura 4.7b).
- 1 Sensor de impacto y sonido 29132 (Figura 4.7c).
- 1 Cable Adaptador USB (Figura 4.7d).
- 3 LED’S Bicolor (Figura 4.7e).
- 1 Display de 7 segmentos (Figura 4.7f).
- 1 Batería de 3 V (Figura 4.7g).
- 1 Fuente de 5 V.
- Resistencias de 330 Ω.

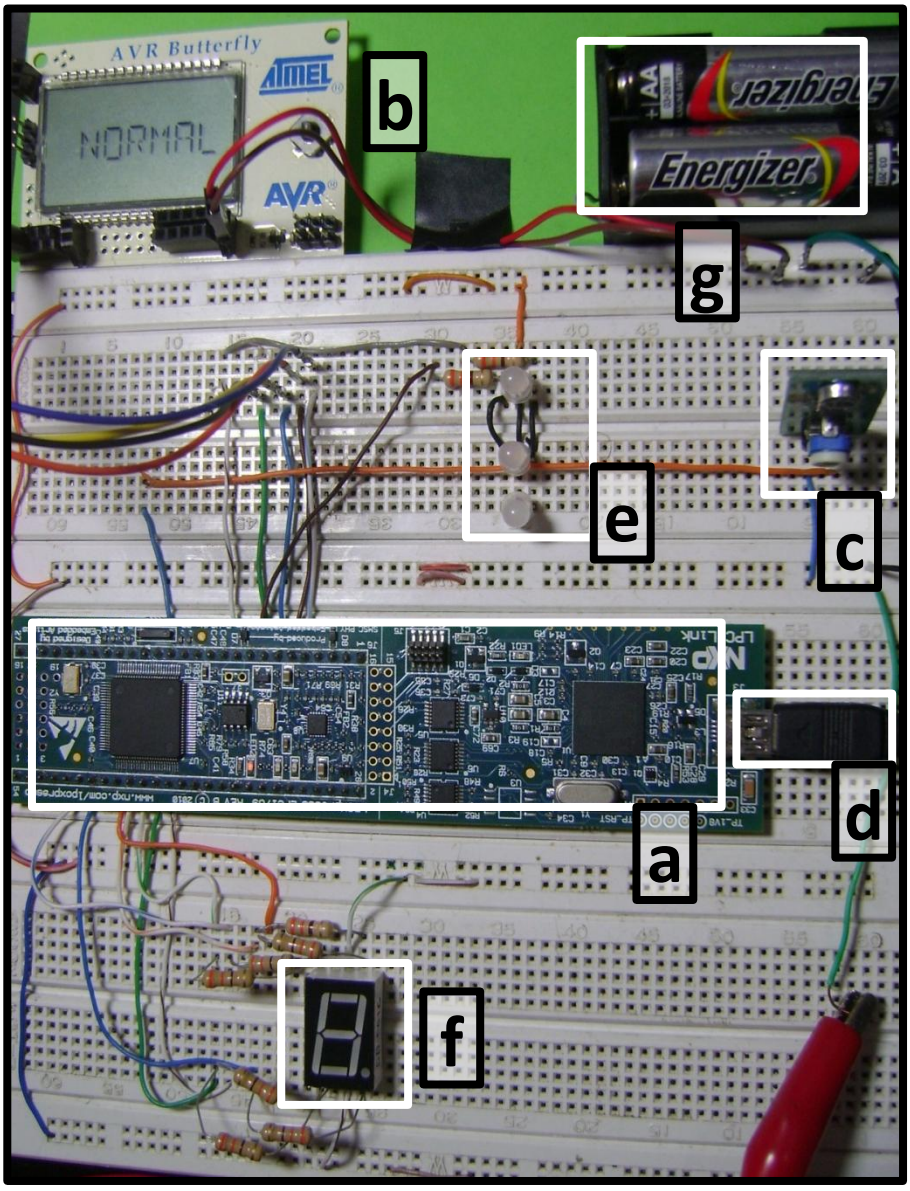

Figura4. 7 Proyecto en estado NORMAL

La Figura 4.7 describe el funcionamiento del sistema sin activación del sensor, es decir, está captando sonido no considerado crítico por lo que en la LCD del kit de desarrollo AVR Butterfly (Figura 4.7b) se muestra el mensaje

“NORMAL”. Ahora bien, cuando el ruido es crítico comienza un contador en el display (Figura 4.8c) y al mismo tiempo se muestra el mensaje “ALERTA” (Figura 4.8b). En cuanto el contador llegue a su fin, se apaga, y comienza la alarma lumínica representada por los LED’s bicolores como se muestra en la Figura 4.9a y Figura 4.9b, manteniéndose el mensaje “ALERTA”.

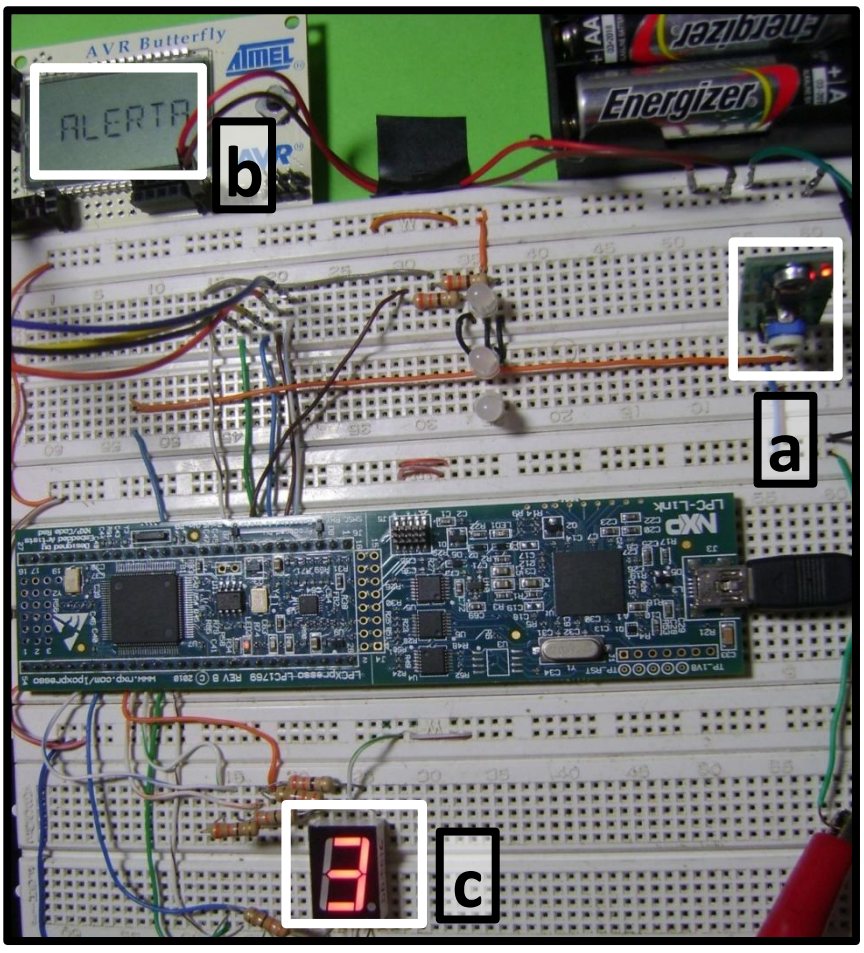

Figura4. 8 Proyecto en estado ALERTA

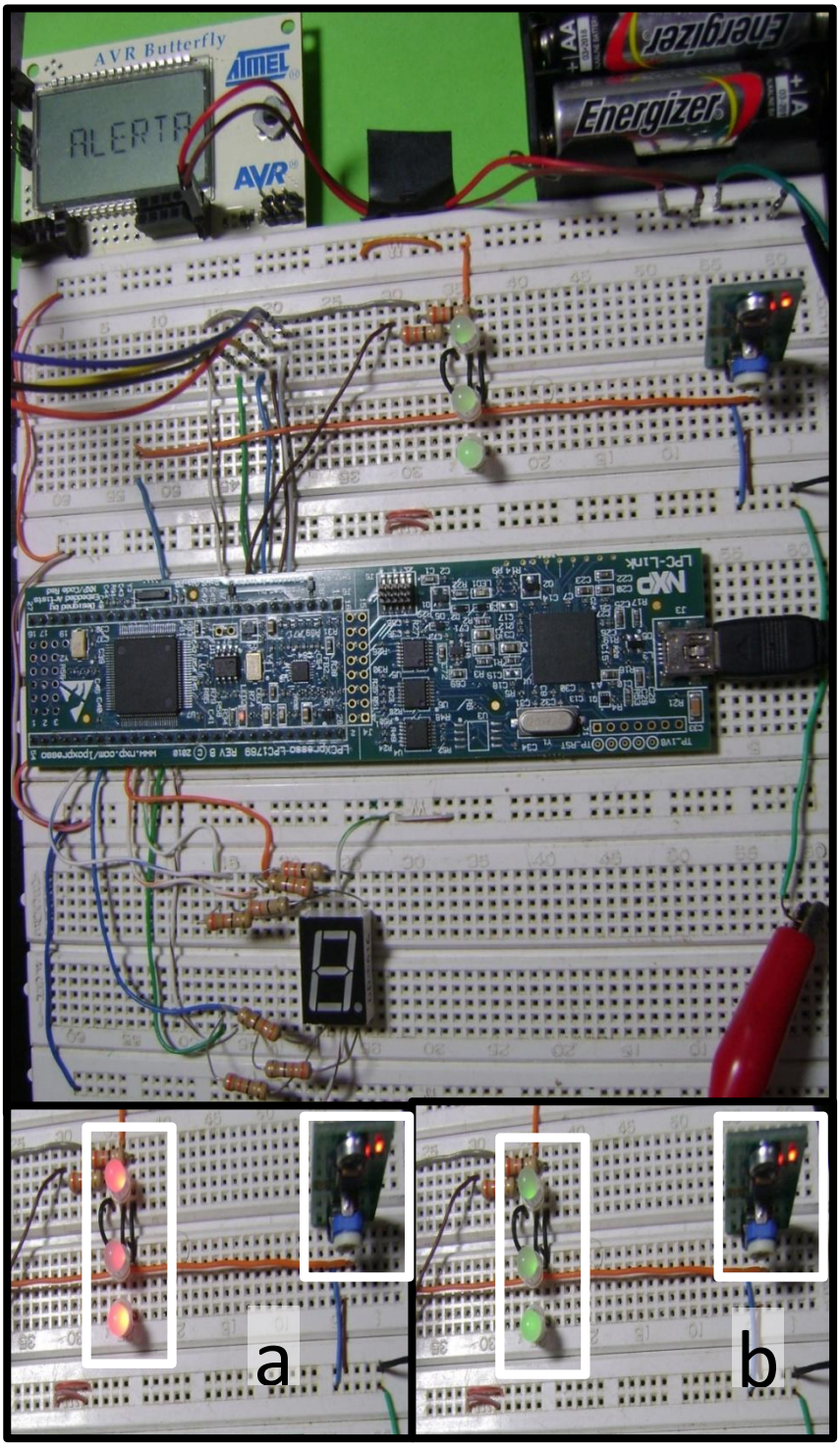

Figura4. 9 Visualización Alarma

CONCLUSIONES

Al terminar con la implementación del proyecto y al haber llevado a cabo una detallada fase de pruebas y demostraciones, se ha llegado a obtener las siguientes resoluciones.

1) Este proyecto demuestra la comunicación serial sincrónica SPI entre microcontroladores de distintos fabricantes, que se dio satisfactoriamente gracias al estudio de los manuales de usuarios de cada tarjeta y a la realización de ejercicios con éstos.

2) Si interpretamos todo lo descrito en los capítulos tres y cuatro comprobamos que para el sistema la distancia a la que se encuentra el motor del sensor, es un factor importante, debido a que el sensor puede ser calibrado por el usuario, y es necesario tener en cuenta el ruido generado por otros agentes para que no influya en el

funcionamiento del sensor, tampoco sobrepasar el rango de los tres metros de cobertura del mismo.

3) Este proyecto fue desarrollado con lenguaje de programación “C”, utilizando la herramienta 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, teniendo en cuenta que no se tuvo un programa para simular en funcionamiento de la implementación.

RECOMENDACIONES

Las recomendaciones que logramos obtener del proyecto son las siguientes:

1. Tener en cuenta que los dispositivos son de uso delicado y aunque hay que tener cuidado de no dañarlo, también hay que asegurarse de que se estén haciendo bien las conexiones para que el funcionamiento sea el correcto. Para asegurar que la transmisión entre los dispositivos se esté dando es necesario que los cables estén bien conectados, para lo cual sugerimos comprar cable UTP de mayor grosor para evitar rupturas internas dentro de los orificios del protoboard que produzcan una falla en la comunicación.

2. Es necesario que los cables que son conectados a la computadora estén funcionando correctamente, tal es el caso del cable de conexión

USB que energiza y ayuda a la programación de la tarjeta LPC1769, asimismo con la comunicación USART entre el kit de desarrollo AVR Butterfly y la computadora. [6]

3. La importancia de unir los niveles de referencia de la tarjeta LPC1769, el kit de desarrollo AVR Butterfly y del sensor fue clave esencial para lograr la meta debido a que aunque los tres son energizados de distintas fuentes y tienen que tener una conexión común para que funcionen correctamente.

ANEXOS

ANEXO A

Código Fuente: Ejercicio 1

```
#include <cr_section_macros.h>
#include <NXP/crp.h>
__CRP const unsigned int CRP_WORD = CRP_NO_CRP ;
#include "lpc17xx.h"
#include "type.h"
int main (void)
{
uint32_ti, j,k;
SystemClockUpdate();
LPC_GPIO2->FIODIR = 0x000000FF;          /* P2.xx Definido como salidas */
  LPC_GPIO2->FIOCLR = 0x000000FF;        /* Apaga los LEDs */
  LPC_GPIO0->FIODIR = 0x00000000;        /* P0.xx definida como entrada */
while(1)
  {
        if(LPC_GPIO0->FIOPIN0==0xff)
          {
                for(i = 0; i< 8; i++)
                  {
                        LPC_GPIO2->FIOSET = 1 <<i; // prende cada led
                       for(j = 1000000; j > 0; j--); //Retardo
                  }
                  LPC_GPIO2->FIOCLR = 0x000000FF;
                for(j = 1000000; j > 0; j--);
        }
        else
          {
                for( k= 7; k >= 0 &&  k<=7; k--)
```

```
                {
                        LPC_GPIO2->FIOSET = 1 << k;
                    for(j = 1000000; j > 0; j--);
                }
                LPC_GPIO2->FIOCLR = 0x000000FF;for(j = 1000000; j > 0; j--);
        }
}
}
```

ANEXO B

Código Fuente: Ejercicio 2

/Programa Principal Dispositivo Maestro/

```
#include <cr_section_macros.h>
#include <NXP/crp.h>
__CRP const unsigned int CRP_WORD = CRP_NO_CRP ;
#include "LPC17xx.h"
#include "ssp.h"
#include "adc.h"
#define PORT_NUM                1
#define LOCATION_NUM            0
#define LED   LPC_GPIO2
uint8_t src_addr[SSP_BUFSIZE];
staticconst uint8_t segmentLUT[10] =
{
                //FCPBAGED
                (uint8_t) 0b11011011, // 0
                (uint8_t) 0b01010000, // 1
                (uint8_t) 0b00011111, // 2
                (uint8_t) 0b01011101, // 3
                (uint8_t) 0b11010100, // 4
                (uint8_t) 0b11001101, // 5
                (uint8_t) 0b11001111, // 6
                (uint8_t) 0b01011000, // 7
                (uint8_t) 0b11011111, // 8
                (uint8_t) 0b11011101, // 9
};
/*****************************************************************************
```

```
Función Principal main()
*********************************************************************/
int main (void)
{
SystemClockUpdate();
uint32_ti = 0,j=0;
uint32_tportnum = PORT_NUM;
LPC_GPIO2->FIODIR = 0x000000FF;
  LPC_GPIO2->FIOCLR = 0x000000FF;
while ( 1 )
  {
        SSP1Init();
        src_addr[0] = (uint8_t)segmentLUT[i];
        LPC_GPIO2->FIOSET = src_addr[0];
        for(j = 1000000; j > 0; j--);
        SSPSend(portnum, (uint8_t *)src_addr, 1);
        LPC_GPIO2->FIOCLR = 0x000000FF;
        for (j=90100;j>0;j--);
        i=i+1;
        if (i==10) i=0;
  }
return 0;
}
/*********************************************************************
Fin de Archivo
*********************************************************************/
```

/*Programa Principal Dispositivo Esclavo*/

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

```
__CRP const unsigned int CRP_WORD = CRP_NO_CRP ;
#include "LPC17xx.h"
#include "ssp.h"
#include "adc.h"
#define PORT_NUM                    1
#define LOCATION_NUM            0
#define LED   LPC_GPIO2
uint8_t src_addr[SSP_BUFSIZE];
uint8_t dest_addr[SSP_BUFSIZE];
/*****************************************************************************
Función Principal  main()
*****************************************************************************/
int main (void)
{
SystemClockUpdate();
uint32_ti = 0,j=0;
uint32_tportnum = PORT_NUM;
LPC_GPIO2->FIODIR = 0x000000FF;
LPC_GPIO2->FIOCLR = 0x000000FF;
while ( 1 )
  {
        SSP1Init();
        SSPReceive (portnum, (uint8_t *)dest_addr, 1 );
        LPC_GPIO2->FIOSET = dest_addr[0];
        for(j = 1000000; j > 0; j--);
        LPC_GPIO2->FIOCLR = 0x000000FF;
        for (j=90100;j>0;j--);
  }
return 0;
}
```

/**

Fin de Archivo

**/

ANEXO C

Código Fuente: Proyecto

/Programa Principal Dispositivo Maestro/

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

__CRP const unsigned int CRP_WORD = CRP_NO_CRP ;

#include "LPC17xx.h"
#include "ssp.h"
#include "adc.h"

#define PORT_NUM                1
#define LOCATION_NUM            0
#define LED   LPC_GPIO2

uint8_t datosensor [SSP_BUFSIZE];
extern volatile uint32_t ValorADC[ADC_NUM];
static const uint8_t segmentLUT[10]=
{
                //FCPBAGED para ánodo común
                        (uint8_t) 0b11011011, // 0
                        (uint8_t) 0b01010000, // 1
                        (uint8_t) 0b00011111, // 2
                        (uint8_t) 0b01011101, // 3
                        (uint8_t) 0b11010100, // 4
                        (uint8_t) 0b11001101, // 5
```

```
};
/*****************************************************************************
  Main Function  main()
*****************************************************************************/
int main (void)
{
 SystemClockUpdate();
  uint32_t j=0, i=0, a=0;
  LPC_GPIO2->FIODIR = 0x000000FF;            /* P2.xx Definido como salidas */
  LPC_GPIO0->FIODIR = 0x000000FF;            /* P0.xx Definido como salidas */
 uint32_t portnum = PORT_NUM;
 LPC_GPIO2->FIOCLR = 0x000000FF;            /* Apaga LEDs */
 LPC_GPIO0->FIOCLR = 0x000000FF;            /* Apaga LEDs */
 while ( 1 )
 {
        SSP1Init();
        ADCInit( ADC_CLK );
        ValorADC[0] = ADCRead(0);
        datosensor [0] = ValorADC[0]/100;
        SSPSend( portnum, (uint8_t *)datosensor, 1);

if (datosensor [0]>35){
                for(i = 0; i< 6; i++)
                                {LPC_GPIO2->FIOSET = (uint8_t)segmentLUT[i];
                                for(j = 10000000; j > 0; j--);
                                LPC_GPIO2->FIOCLR = 0x000000FF;
                                for(j = 1000000; j > 0; j--);
                                ValorADC[0] = ADCRead(0);
                                datosensor [0] = ValorADC[0]/100;
                                if (datosensor [0]<35) break;
```

```
                        }
        }
        while (datosensor [0]>35){
                for(i = 0; i< 2; i++)
                        {
                                LPC_GPIO0->FIOSET = 1 <<i; // prendecada
led
                                for(j = 1000000; j > 0; j--); //Retardo
                                LPC_GPIO0->FIOCLR = 0x000000FF;
                                for(j = 1000000; j > 0; j--);
                        }
                ValorADC[0] = ADCRead(0);
                datosensor [0] = ValorADC[0]/100;
            }
        }
return 0;
  }
/*****************************************************************************
End Of File
*****************************************************************************/
```

/*Programa Principal Dispositivo Esclavo*/

```
/*****************************************************************
      MICROCONTROLADORES AVANZADOS
 *****************************************************************
            Comunicación Serial SPI
 *****************************************************************
               PROYECTO
 * Nombre: Graficacion y analisis de Datos
 * Descripción:
   El esclavo de la comunicación SPI recibe el valor correspon-
```

diente al dato sensado y lo muestra por la pantalla LCD.
Ademas analiza los datos y muestra mensaje de aviso.
**********/

```
#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"

void SPI_slave_init(void)
{
 DDRB=0x08;    // MISO como salida y el resto como entrada
 SPCR=(1<<SPE);  // SPI enable, dispositivo Slave
}

uint8_t SPI_slave_receive(void)
{
 while(!(SPSR & (1<<SPIF))); // espera mientras recibe el dato completo
 return SPDR; // retorna dato recibido
}

int main(void)
{
```

```
  PGM_P statetext = PSTR("FIGUEROA-SAAVEDRA");
  uint8_t input;

  char *cadena;
  int  i;
   LCD_Init();                // initialize the LCD

  SPI_slave_init(); // inicializa SPI
   sei();

   while (1)
   {
      if (statetext){
       LCD_puts_f(statetext, 1);
       LCD_Colon(0);
       statetext = NULL;
                  }
input=SPI_slave_receive();
   switch (input) {
       case 0:
         statetext = PSTR("NORMAL");
         break;
       case 40:
         statetext = PSTR("ALERTA");
                        _delay_ms(2);
         break;
         default:
         break;
         }
   }
```

```
    return 0;
}
```

BIBLIOGRAFÍA

[1]. Padmaraja Yedamale, Microchip Techonology Inc. – AN885: Brushless DC (BLDC) Motor Fundamentals Fecha de consulta: 06/03/12.

[2]. Martin THOMAS, AVR-Projects,

http://gandalf.arubi.uni-kl.de/avr_projects/

Fecha de consulta: 08/03/12.

[3]. ATMEL, ATMEGA169 datasheet,

www.atmel.com/dyn/resources/prod_documents/doc2514.pdf

Fecha de consulta: 09/03/12.

[4]. NXP Semiconductors, Cortex M3, Manual de Usuario 10470

http://www.nxp.com/documents/user_manual/UM10470.pdf

Fecha de consulta: 21/02/12.

[5]. Topic 18c: Serial Peripheral Interface

http://ww1.microchip.com/downloads/en/DeviceDoc/70243b.pdf

Fecha de Consulta: 09/03/12.

[6]. Richard Leonel Guerrero Jumbo, KIT DE DESARROLLO AVR BUTTERFLY, DESARROLLO DE GUÍA DE PRÁCTICAS DE LABORATORIO Y TUTORIALES.

Fecha de Consulta: 09/03/12.

[7] Causas de vibración de los motores de inducción de 2polos mediante análisis espectral

www.motors-electrics.com/pdf/Vibraciones.pdf

Fecha de Consulta: 09/03/12.

[8] Parallax, Sound Impact Sensor Documentation

https://www.parallax.com/Portals/0/Downloads/docs/prod/sens/29132-SoundImpactSesnor-v1.0.pdf

[9] Alan Kharsansky, Introducci_on a LPCXpresso y repaso del lenguaje C, Seminario de Sistemas Embebidos, Agosto 2011

http://laboratorios.fi.uba.ar/lse/seminario/material-2011/Sistemas_Embebidos-2011_2doC-Intro_a_LPCXpresso_y_repaso_lenguaje_C-Kharsansky.pdf

[10] Osmetec BLDC Motor

http://www.osmtec.com/bldc_motor_57bl.htm

[11] Mario Sacco, Motores Brushless o BLDC (Métodos de Accionamiento), Abril 2012

http://www.neoteo.com/motores-brushless-bldc