ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Universidad Politécnica de Cartagena

Proyecto Fin de Carrera

Diseño y desarrollo de un simulador de adquisición de entradas digitales y actuadores basado en microcontrolador PIC.

AUTOR: Alejandro Daniel Páez Espinosa

DIRECTOR: Alejandro Santos Martínez Sala

Noviembre / 2012

Universidad
Politécnica
de Cartagena

Autor	Alejandro Daniel Páez Espinosa
E-mail del Autor	alejandro.paez.espinosa@gmail.com
Director(es)	Alejandro Santos Martínez Sala
E-mail del Director	alejandro.martinez@upct.es
Título del PFC	Diseño y desarrollo de un simulador de adquisición de entradas digitales y actuadores basado en microcontrolador PIC.

Resumen:

El proyecto final de carrera consistirá en el diseño y desarrollo de un simulador de adquisición de entradas digitales gestionada mediante de comunicación puerto serie y Ethernet mediante protocolo UDP. El servicio UDP estará basado en microcontrolador PIC18F mediante el modulo ENC28J60.

En este proyecto asimismo se realiza el diseño de un controlador (API) que emulará los drivers de la tarjeta basada en PIC18F y que abstraerá la lógica de bajo nivel, siendo posible su utilización en aplicaciones de alto nivel.

Titulación	Ingeniero de Telecomunicación, especialidad Telemática
Departamento	Tecnologías de la Información y las comunicaciones (TIC)
Fecha de Presentación	

Índice general

Índice de figuras

Índice de tablas

Capitulo 1

Introducción

1.1.- Antecedentes.

Hoy en día, en el mercado de la automatización existe un abanico de productos que nos pueden ayudar ahorrar en mano de obra, dando lugar a que estas operaciones la realicen maquinas de forma automática, con lo que se realizaría una gestión mas rápida y eficiente.

Este proyecto surge a partir de un proyecto previo titulado “Diseño y desarrollo de un sistema automatizado de control de entradas y salidas de camiones”, que consiste en una aplicación telemática de gestión y control que emplea una tarjeta de adquisición de datos I/O comercial (Advantech USB 4761) para gestionar los sensores de detección de paso y el accionamiento, mediante relé, de barreras y luces.

Figura 1: Tarjeta comercial Advantech USB-4751

El proyecto mencionado anteriormente, presenta varias limitaciones en la tarjeta de adquisición de datos (de ahora en adelante llamada DAQ). Por un lado, la distancia física entre el PC y la DAQ para realizar la comunicación entre ellos, esta limitada a

unos 2 metros aproximadamente. Por otra lado, existen problemas de compatibilidad en los drivers proporcionados por el distribuidor de la DAQ, con el lenguaje C# .net.

Finalmente, el principal inconveniente de la DAQ, es que presenta un solo modo de funcionamiento, llamado modo polling. Este modo, consiste en que el PC esta constantemente enviando comandos para detectar el estado de las entradas digitales, lo cual provoca un mayor consumo de recursos. Como solución, interesaría disponer de un modo evento consistente en enviar mensajes al PC, solamente cuando la tarjeta detecta un cambio. De esta forma, se reduciría el consumo de recursos.

1.2.- Objetivo

1.2.1.- General

El objetivo de este proyecto es diseñar y desarrollar un prototipo de tarjeta de I/O con adquisición de entradas digitales y accionamiento de salidas relé, basado en microcontroladores que mejoren las limitaciones de la tarjeta Advantech comercial.

Para desarrollar el prototipo de tarjeta de I/O se utilizará un entorno de desarrollo de Microchip con un PIC18 con interfaz de comunicaciones por Puerto Serie y Ethernet. A este entorno se le conectará una placa prototipo con entradas digitales y relés para que sean accionadas por pines I/O del PIC.

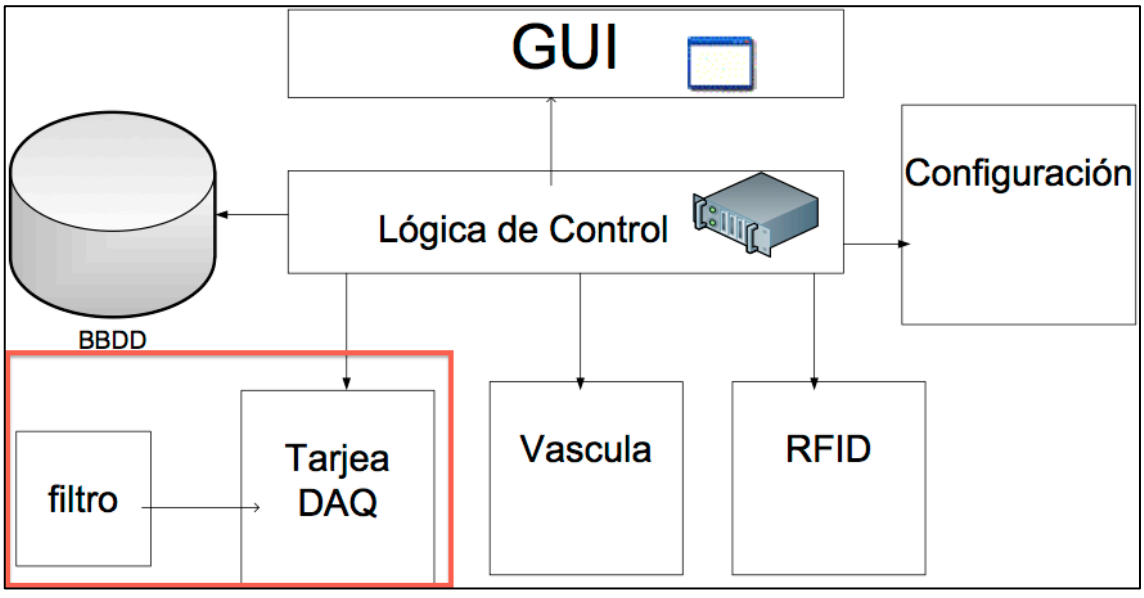

Figura 2: Arquitectura General del Sistema

En el desarrollo de la tarjeta de I/O, se diseñará un controlador que emulará los drivers de la tarjeta basada en PIC y que abstraerá la lógica de bajo nivel de la aplicación que usa dicha tarjeta. En la comunicación entre la tarjeta de I/O y la estación de trabajo, se implementará un protocolo de comunicación y un programa en C# que controlará a bajo nivel la tarjeta. Este programa será una API simplificada y sencilla para que la tarjeta sea usada por una aplicación de alto nivel.

1.2.2.- Específico

- Estudiar el marco teórico referente a la tecnología PIC-Ethernet, y PIC-RS232.
- Diseñar un protocolo de comunicación PIC → PC.
- Diseño y fabricación de una placa prototipo con entradas digitales y relés.
- Implementación de los drivers de la tarjeta de desarrollo basada en una API en C#.
- Determinar la eficiencia del sistema, en base a pruebas del funcionamiento.

1.3.- Herramientas y entornos de desarrollo.

Para el desarrollo del proyecto se usarán diferente herramientas. En este apartado se explicara resumidamente, haciendo mayor hincapié en las herramientas y entornos de desarrollo.

- MPLAB

MPLAB-IDE es un entorno de desarrollo especialmente diseñado para desarrollar aplicaciones para microcontroladores del fabricante Microchip y controladores de señales digitales. El entorno proporciona herramientas integradas, lo que permite desarrollar código para microcontroladores embebidos.

Este entorno de desarrollo nos concede la capacidad de programar en diferentes lenguajes como son ensamblador o C. A si también nos permite ensamblar, gestionar proyectos, depurar y simular, ya que se integra con diferentes herramientas como puede ser MATLAB, ISIS Proteus, etc. Mas información http://www.microchip.com/.

- VISUAL STUDIO

Para el desarrollo de la API, que sirve como controlador de bajo nivel de la tarjeta de I/O, se ha utilizado la Suite Visual Studio 2010 Express, que es un entorno de desarrollo gratuito proporcionado por Microsoft para el desarrollo de aplicaciones de

escritorio para sistemas Windows. Dentro de este entorno de desarrollo hay varios paquetes que permite usar varios lenguajes tales como C++, C#, Visual Basic y otros.

Dentro de esta suite se ha optado por Visual C# debido a que la aplicación final a la que proporciona la función de driver ya estaba desarrollada en este lenguaje. Mas información http://www.microsoft.com/visualstudio/

- OTRAS HERRAMIENTAS

Otras de las herramientas y aplicaciones que se han llegado a utilizar han sido: analizadores de puerto serie, CCS C Compiler y programador USB Pitkit2 de Microchip:

- **232Analyer**: Es un programa que nos ha permitido controlar, monitorear y revisar errores en la comunicación serie.. Este programa permite ver todos los datos de la comunicación entre dos dispositivos que utilicen el puerto serie. Mas información http://www.commfront.com/

- **CCS C Compiler**: Es una plataforma de desarrollo para microcontroladores, el cual permite programar en lenguaje C. Este compilador se integra como un plug-in en MPLAB-IDE, lo que facilita la programación convirtiendo el código C en lenguaje especifico para microcontroladores. Mas información http://www.ccsinfo.com/

- **Programador PitKit2**: Es un programador para microcontroladores PIC realizado por Microchip. Se utiliza para programar y depurar microcontroladores, así también como programar EEPROM. La ventaja que nos ofrece este programador es que puede programar los microcontroladores directamente en la placa de desarrollo y/o protoboard sin necesidad de sacar el microcontrolador, cada vez que se modifique la programación. Mas información http://www.microchip.com/pitkit2

1.4.- Estructura del contenido del proyecto

El trabajo desarrollado en el proyecto final de carrera que se presenta se divide en los siguientes capítulos:

- **Capítulo 2**. Desarrollo de la Capa de Control

  En este capitulo se explica el funcionamiento principal del Microcontrolador, el formato de las tramas serie y Ethernet transmitidas entre el microcontrolador y el PC y viceversa.

- **Capítulo 3**. Implementación del Protocolo de comunicaciones, Puerto serie y Ethernet.

  Se explicara, la comunicación llevada a cabo entre el PC y el microcontrolador desde el punto de vista de la implementación, como así también explicación del desarrollo de la API, para el control de la DAQ.

- **Capitulo 4**. Pruebas y resultados.

  En este capítulo se mostrara la interfaz grafica que se ha desarrollado para el control de la API para llevar a cabo las pruebas de funcionamiento y las pruebas realizadas en la integración con la aplicación principal.

Capitulo 2

Desarrollo de la Capa de Control

2.1.- Maquina de estados del microcontrolador

Antes de entrar a explicar en detalle el funcionamiento del sistema, es importante conocer cual es la maquina de estados de la tarjeta de I/O, ya que de ella ha dependido la implementación de la unidad de control del sistema, es decir, del microcontrolador PIC18F97J60.

La siguiente figura muestra la maquina de estados del la tarjeta, que llevara a cabo durante su ejecución:

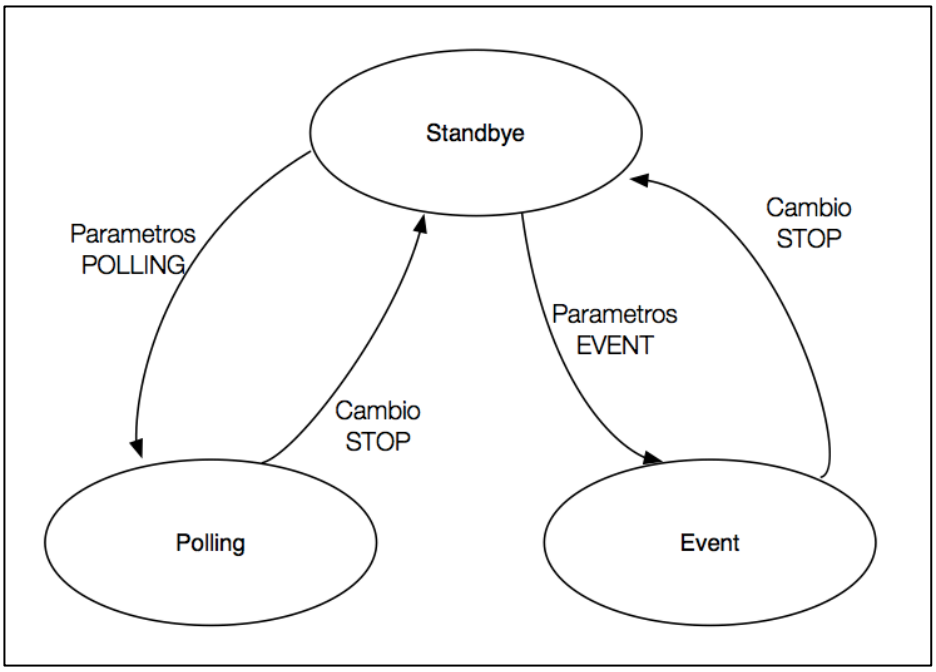

Figura 3: Maquina de Estados de la tarjeta de I/O

La maquina de estados que incorpora el PIC se utiliza para cambiar entre los diferentes estados de operación o modos de funcionamiento. El cambio de un estado a otro se realiza mediante el intercambio de una serie de comandos, en necesidad de la funcionalidad que se requiera.

Los comandos que se han empleado son:

- **STOP**: Modo de arranque principal del PIC donde solo responderá a petición de comandos, por ejemplo comando ECHO o cambio de estado de las salidas digitales de la tarjeta de I/O.
- **POLLING**: El modo polling consiste que el PIC esta a la espera de comandos, donde este solo responderá, cuando es consultado por el PC maestro, respondiendo con el estado actual de las entradas.
- **EVENT**: Este modo de funcionamiento consiste que cuando la tarjeta detecta un cambio de estado en sus entradas, este notifica de los cambios de tarjeta al PC maestro.

2.2.- Descripción del funcionamiento de la tarjeta de I/O

Como ya se ha comentado, el objetivo principal de este Proyecto Final de Carrera es diseñar un prototipo de tarjeta de I/O, que simule la tarjeta de adquisición de datos comercial Advantech USB 4751, abstrayendo la lógica de bajo nivel. A continuación se hace una breve descripción de la implementación llevada a cabo en el PIC con el fin de conocer mejor el funcionamiento del mismo.

La tarjeta comenzara permaneciendo a la espera del comando, vía puerto serie, necesario que indicara el modo de comunicación que se llevara a cabo durante todo el . Los modos de comunicación permitidos se clasifican en puerto serie o puerto Ethernet.

Una vez seleccionado el modo de comunicación, el PIC iniciará en un modo “reposo” o “standbye” donde solo podrá recibir comandos donde se le solicite información sobre estado de entradas, comprobación de la comunicación, o simplemente cambiar el modo de funcionamiento de la tarjeta de I/O.

La tarjeta de I/O dispone de dos modos mas de funcionamiento. El modo “evento”, el cual el PIC comprueba constantemente las entras y salidas digitales y al detectar un cambio en una de sus entradas enviará una trama el PC donde notificará el cambio y el nuevo estado de las entradas.

El otro modo de funcionamiento disponible en la tarjeta de I/O, es el llamado modo “polling”, que se caracteriza en que el PC maestro es quien realiza las peticiones al microcontrolador para conocer el estado de las entradas y salidas, estando siempre el microcontrolador a la espera de que acción realizar.

A la hora de establecer estos modos de funcionamiento, es necesario establecer unas características que son necesarias para el funcionamiento de la tarjeta,. Una de estas características es el tiempo de muestreo, que es el periodo en el que el microcontrolador realizar las comprobaciones de las entras y salidas de la tarta de I/O. Este periodo esta predefinido en useg.

Otra de las características es el umbral de tiempo. Este umbral es útil a la hora de realizar las comprobación de cambio de estado de las salidas. La tarjeta durante este procedimiento toma muestras basándose en este umbral. Estas muestras son útiles para detectar que las señales que detecta el microcontrolador proceden de una pulsación de un sensor o de una fuente de ruido. La funcionalidad principal es para la detección de espurios.

La ultima característica de que dispone es keepalive. Esta propiedad tiene como finalidad enviar periódicamente mensajes al PC maestro para confirmar que el extremo (tarjeta) sigue estando activo. Estos mensajes de keepalive deben generarse si el tiempo establecido es mayor que 0. Esta funcionalidad es útil cuando el sistema esta inactivo durante un largo periodo de tiempo.

2.3.- Tipos y Formatos de tramas entre Microcontrolador y el PC

Es necesario conocer el formato de la trama que enviará la tarjeta de I/O teniendo en cuenta los campos de interés que se desean transmitir. Los campos que forman la trama se muestras en la siguiente figura.

La trama que envía el microcontrolador al PC es la siguiente:

Id_emisor (1byte)	**Id_receptor (1byte)**	**Longitud (1byte)**	**Tipo (1byte)**	**Datos (8bytes)**	**Contador (2bytes)**

Figura 4: Trama enviada por el microcontrolador

La siguiente tabla muestra la descripción y la utilidad de cada uno de los campos que forman la trama que luego es procesada por el PC maestro:

CAMPO	DESCRIPCION
Id_emisor	Este campo contiene el identificador del microcontrolador que envía la trama.
Id_receptor	Este campo contiene el identificador del PC receptor que recibirá la trama.
Longitud	El campo longitud informará, como su nombre indica, sobre el numero de bits que se están transmitiendo. Este campo es útil a la hora de que la trama sufriera alguna modificación en el tamaño de alguno de sus campos o alguna incorporación de un nuevo campo.
Tipo	Este campo "tipo" indica el tipo de trama que se esta enviando, pudiendo tener diferente valores: ▪ ACK_ECHO → 1 ▪ SEND_IN → 3 ▪ ACK_WRITE → 5 ▪ NO_ACK_WRITE → 6 ▪ ACK_WRITE_ONE → 12 ▪ ACK_NO_WRITE_ON → 13.
Datos	Este campo como bien indica su nombre, contendrá la información del estado de las entradas y salidas digitales del microcontrolador. Cada uno de los bytes que lo forman corresponde a cada una de las puertas del microcontrolador.
Contador	Este campo se utilizará para numerar las tramas. Este contador es de 2 bytes, teniendo como valor máximo 65535. Este campo se utiliza para comprobar si ha habido perdida de alguna trama.

Tabla 1: Campos de la trama enviada por la tarjeta de I/O

Como se ha menciona, la trama contiene un campo "tipo" podrá tener diferentes valores:

- ACK_ECHO → '1' : Petición de respuesta desde el PIC al PC en el cual se utiliza para la comprobación de la comunicación entre Pc y Microcontrolador.

- SEND_IN → '3' : Petición de respuesta a la lectura de entradas y salidas digitales, en donde también se incorpora en el campo datos el estado de las entrada y salidas.

- ACK_WRITE → '5' : Petición de respuesta de escritura satisfactoria en las salidas digitales.

- NO_ACK_WRITE → '6' : Petición similar a tipo de trama anterior. Este tipo de trama, se utiliza para indicar que se quiere escribir la salidas digitales, y estas ya se encuentran en el estado actual que se está indicando.

- ACK_WRITE_ONE → '12' : Petición de respuesta de escritura satisfactoria de una única salida digital.

- ACK_NO_WRITE_ONE → '13' : Petición de respuesta de escritura no satisfactoria, la salida digital indicada no sea la correcta o que la salida ya se encuentra en el estado indicado.

2.4.- Tipos y Formatos de trama entre PC y Microcontrolador

A continuación se explicara los formatos de la trama que se enviara desde el PC al Microcontrolador. Estos son de mayor importancia ya que sin las peticiones del PC maestro, no se iniciaría el funcionamiento de la tarjeta de I/O.

2.4.1.- Etapa configuración

Existes varias tramas de comunicación que el PC maestro envía al microcontrolador. La primera trama y no menos importante es la de modo de configuración, en donde se indica el modo de comunicación que se va a utilizar.

ID_PIC (1 byte)	ID_PC (1 byte)	MODO (1 byte)	CONFIG (8 bytes)

Figura 5: Trama de modo de configuración inicial

Dependiendo del modo de funcionamiento que se utilice, el campo CONFIG contendrá una sub-trama con el resto de información de configuración.

La siguiente tabla muestra la descripción de cada uno de los campos que forman la trama de configuración inicial:

CAMPO	DESCRIPCION
Id_pic	Este campo contendrá el identificador que se establecerá al PIC.
Id_pc	Este campo contiene el identificador del pc que se usara de PC maestro.
Modo	Este campo como su nombre indica, indica el modo de comunicación que se usará, pudiendo tener los siguientes valores: ▪ SERIE → '20' ▪ ETHERNET → '30'
Config	Este campo es utilizado como campos de datos donde irán el resto de parámetros necesarios para la configuración de la tarjeta de I/O. Este campo de configuración varia dependiendo del modo de comunicación que se seleccione.

En la tabla anterior se ha mencionado que el campo "config" varia dependiendo de el modo de comunicación que se utilice, a continuación se detalla la información que corresponde con este campo:

VELOCIDAD (1 byte)

Figura 6: Sub-trama modo serie

- SERIE:

Si se usa el modo puerto serie, solo será necesario indicar la velocidad con la que se llevara a cabo con el PC maestro. Los valores pueden ser de 1 a 7, siendo velocidades desde 9600bps a 256000bps respectivamente.

Valor	Velocidad
1	9600bps
2	14400bps
3	19200bps
4	38400bps
5	57600bps
6	115200bps
7	256000bps

Tabla 2: Velocidades de comunicación por puerto serie.

- ETHERNET

En el modo Ethernet existen 2 opciones posibles, modo “default” en donde al tarjeta obtendrá el valor de la IP establecida por programación, siendo 192.168.1.254. Para usar el modo “default” será necesario poner los valores de los campos a 0. La otra opción disponible es “IP fija” en donde se puede optar por cambiar la IP por defecto y usar una dirección adecuada a nuestra red. En la Figura 7, se puede apreciar los diferentes campos que la componen, siendo cada uno de estos, los octetos que de una dirección IP.

IP_1 (1 byte)	IP_2 (1 byte)	IP_3 (1 byte)	IP_4 (1 byte)	NET_1 (1 byte)	NET_2 (1 byte)	NET_3 (1 byte)	NET_4 (1 byte)

Figura 7: Sub-trama modo Ethernet

2.4.2.- Etapa Ejecución

Una vez ya establecida la comunicación el microcontrolador procede a la etapa de ejecución a la espera de peticiones por parte del PC maestro. Los campos que forman la trama son los siguientes:

Id_emisor (1byte)	Id_receptor (1byte)	Longitud (1byte)	Tipo (1byte)	Datos (8bytes)

Figura 8: Trama enviada por el PC maestro

Como se puede apreciar el formato de trama que el PC maestro que envía a la tarjeta de I/O, no difiere mucho con respecto a la que el PIC envía al PC. Lo relevante son los tipos de comandos que existen en la comunicación y los datos que la componen.

Los valores que el campo "tipo" podrá tener son:

- ECHO → '0' : Se realiza una petición de echo para comprobar la comunicación con la tarjeta de I/O. Permaneciendo la aplicación a la espera de un ACK_ECHO. En este proceso de intercambio de mensajes no se transmite dato alguno, solo la trama con su petición y en el sentido microcontrolador - PC.

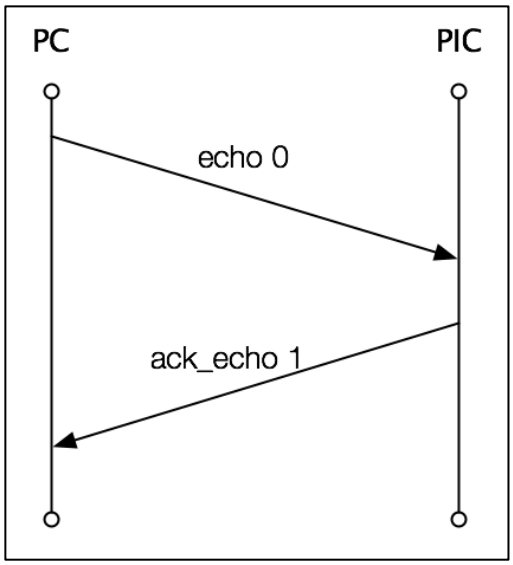

Figura 9: Secuencia petición – respuesta echo

- READ_IN → '2' : Se solicita a la tarjeta la lectura de las entradas y salidas digitales. La tarjeta una vez recibido el comando, se observan las entradas y salidas, añadiendo sus estados en el campo de datos. Una vez finalizado el proceso, se añade al campo tipo el comando "SEND_IN" como respuesta a la solicitud.

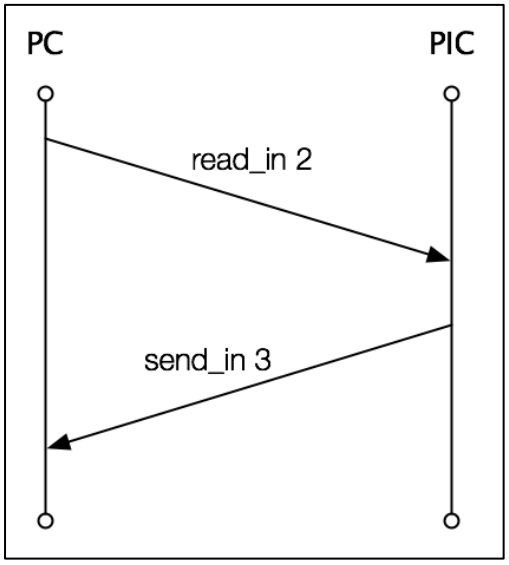

Figura 10: Petición-respuesta de comprobación estado de entradas y salidas digitales

- WRITE_IN → '4' : Solicita el cambio de las salidas digitales permaneciendo a la espera la respuesta por parte de la tarjeta, si ha sido correcto o no el cambio de estados en las puertas logicas. Si ha sido correcta la variación de estados se recibirá un ACK_WRITE y en caso contrario un NO_ACK_WRITE, incluyendo en el campo datos el nuevo estado de las entradas y salidas digitales.

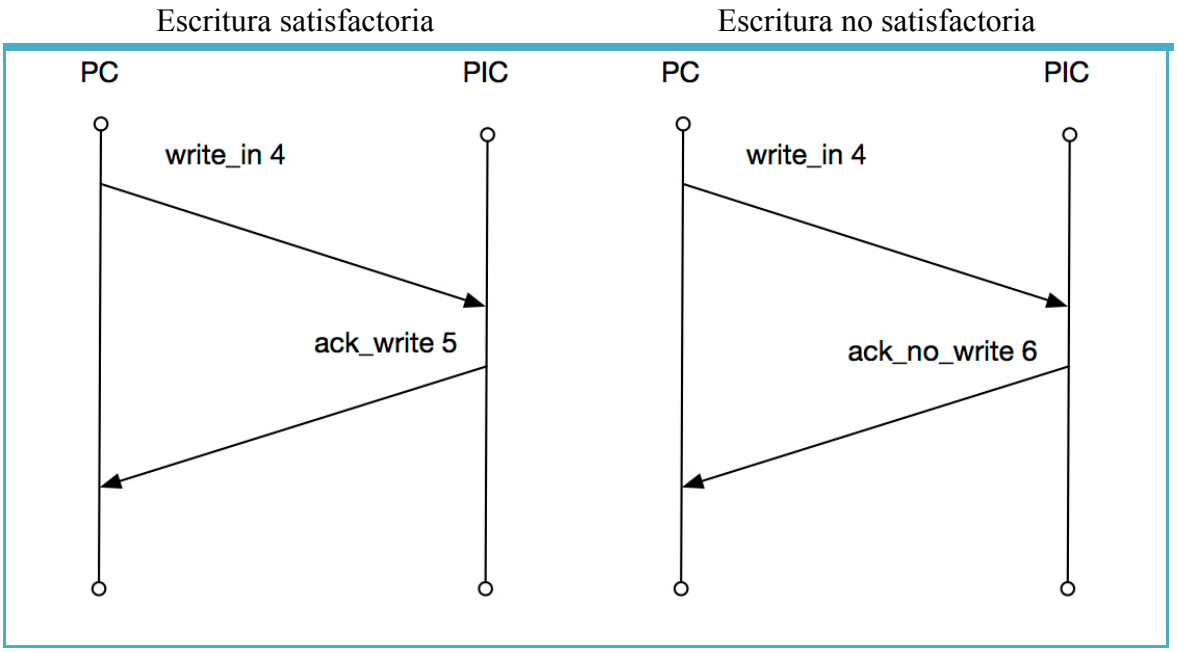

Figura 11: Petición-respuesta de escritura de salidas digitales

- WRITE_IN_ONE → '11' : Solicitud similar a WRITE_IN, en donde solo se quiere modificar el estado de una única salida digital, si el cambio de estado se realiza correctamente, se compone la trama con respuesta ACK_WRITE_ONE, y en caso contrario, ACK_NO_WRITE_ONE. También incluyéndose, en el campo de datos el nuevo estado que compone las salidas y entradas digitales.

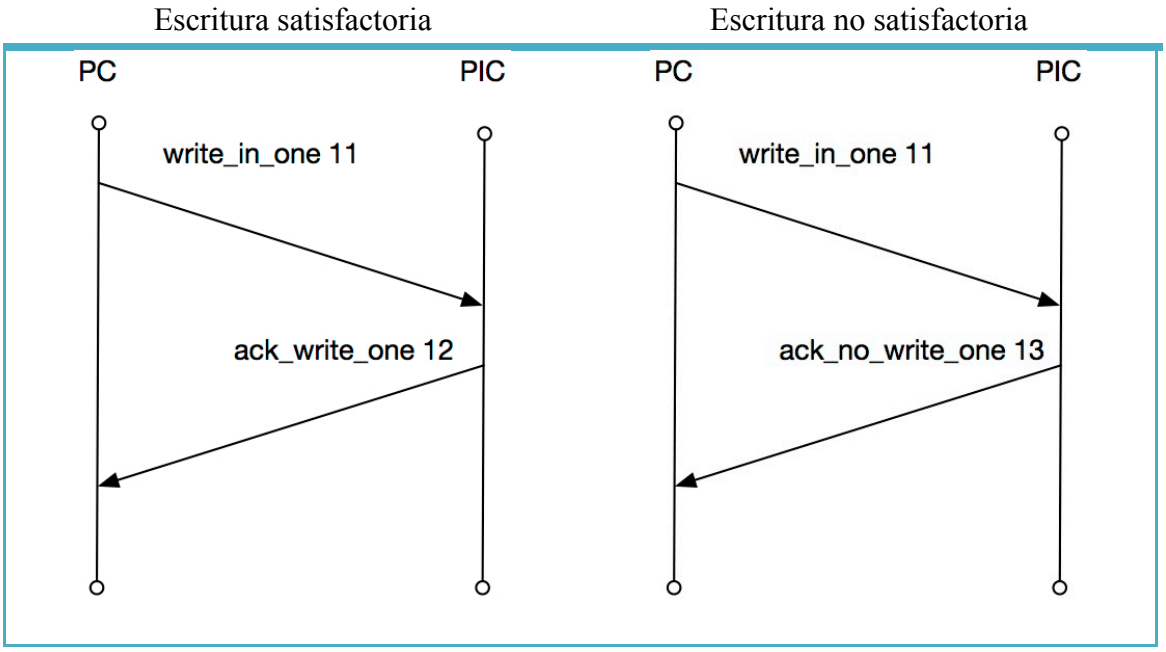

Figura 12: Petición-respuesta de cambio del estado de una salida digital.

- POLLING → '8' : Petición de cambio de estado a modo polling. En apartados anteriores se ha explicado, que con este comando el PC pregunta periódicamente a la tarjeta de I/O sobre el estado de las entradas y salidas digitales.

- EVENT → '9' : Se solicita a petición del PC el cambio de estado a modo evento , en el cual el microcontrolador cuando detecta un cambio de estado en una de sus salidas envía una trama con el nuevo estado de las entradas y salidas al PC maestro.

- STOP → '10' : Este comando indica una petición de cambio de estado de operación o modo de funcionamiento a estado "STAND_BYE". Si bien se ha comentado en apartados anteriores este modo, es el estado inicial de arranque de la tarjeta una vez establecido los parámetros de configuración de la tarjeta. Como se podrá apreciar en la Figura 13, no solo tiene este estado en el modo de inicio de la tarjeta sino que también puede cambiarse a este modo en cualquier momento. Así también si se desea cambiar, por ejemplo, de modo polling a modo evento es requisito indispensable cambiar a modo stop, y luego nuevamente enviar la instrucción evento.

A continuación se presenta un pequeño ejemplo de envió de paquetes de cambio de estado:

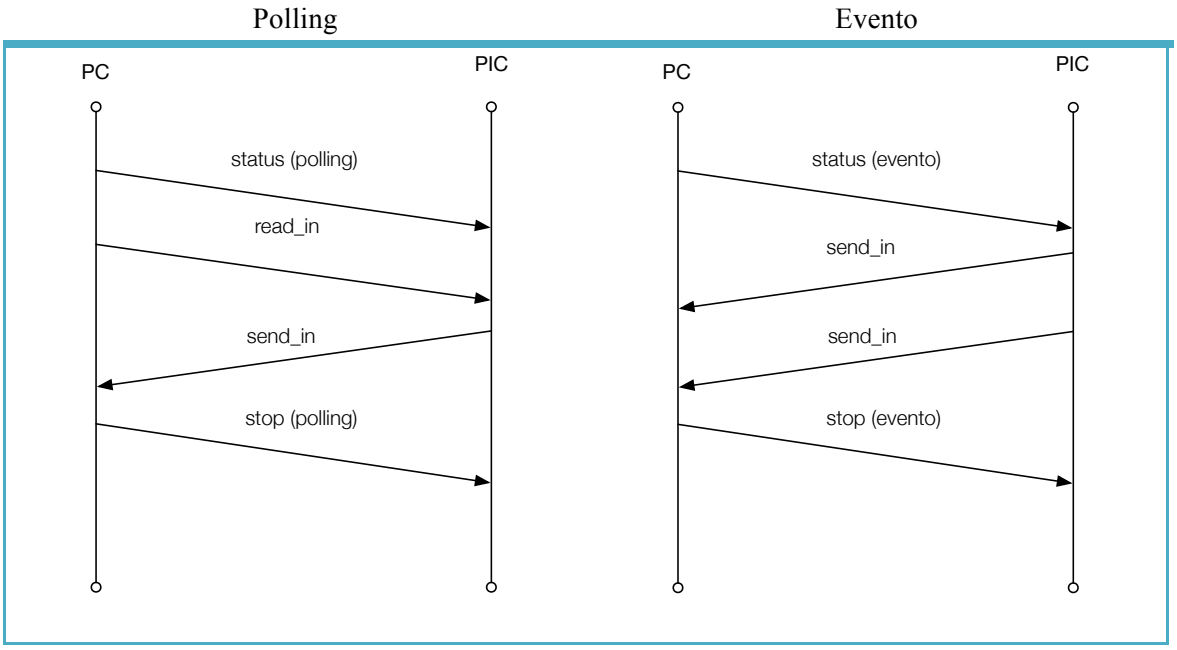

Figura 13: Diagrama de ejemplo cambios de estados.

Estos tres últimos comando de cambio estado explicados, envían en el campo de datos una sub-trama de gran importancia para la realización de las comprobaciones de las entradas y salidas. Esta sub-trama esta compuesta por 3 campos:

tn (1 byte)	th_in (1 byte)	keep_alive (1 byte)

Figura 14: Sub-trama de parámetros

Si bien esta trama es utilizada para establecer varios parámetros para la realización del muestreo de las entradas y salidas en el microcontrolador.

La descripción de cada uno de los campos:

Campo	Descripción
tn	Definido en useg. Este campo es utilizado para establecer el tiempo de muestreo, es decir, cada cuantos useg se realiza la comprobación del muestreo, pudiendo tomar 4 valores: ▪ 10,4 useg → '1' ▪ 20,9 useg → '2' ▪ 41,9 useg → '3' ▪ 83,8 useg → '4'
th_in	Este campo se utiliza para establecer el umbral de muestreos, el cual es de vital importancia para la detección de espurios (señales digitales no deseadas). Se debe establecer valores superiores a 5 para obtener un resultado optimo, ya que en valores inferiores puede obtenerse respuesta de funcionamiento no deseado de la tarjeta de I/O.
Keep_alive	Este campo como su nombre lo indica, se utiliza para indicar si se activará el servicio de keepalive. Indicando con el valor 0, el servicio estaría desactivo, y si es un valor mayor que 0, indicará que esta activo y cada cuanto segundos se desea recibir paquetes READ_IN.

Tabla 3: Explicacion de Subtrama cambio de estado

Capitulo 3

Implementación del Protocolo de Comunicaciones, Serial y Ethernet.

3.1.- Implementación del Microcontrolador

Este apartado se centrará en el funcionamiento del PIC, desde el punto de vista de su implementación. Como se ha comentado en el capítulo 1, se utilizó la herramienta MPLAB con su plug-in de CCS para su compilación en lenguaje C.

Para poder llegar a entender todo el funcionamiento se hará uso de diagramas de flujo, lo que ayudará a entender de manera clara y concisa el funcionamiento del PIC en cada momento.

3.1.1.- Estructura del Proyecto

Debido a la extensión del desarrollo de la implementación en el PIC y con el fin de que fuera lo mas claro posible, el proyecto se dividió en diferentes ficheros (.c y .h).

La siguiente figura presenta la estructura del proyecto.

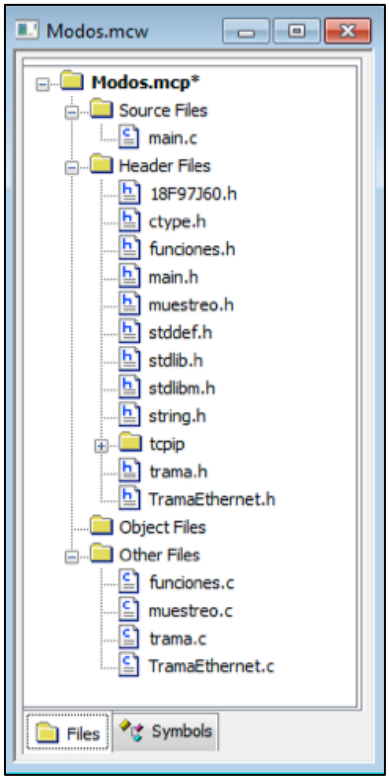

Figura 15: Estructura del Proyecto

- Source Files

Main.c

Contiene la clase principal, la cual contiene el método main. Este método main es donde se tendrá el control total de toda la aplicación. Este fichero también contiene, para el mayor entendimiento del mismo, las funciones llamadas en la ejecución de los diferentes estados.

- Header Files

Los archivos de cabecera "hearder files" son archivos cuya extensión es .h, (ejemplo main.h), y en principio uno incluye en su programa aquellos ficheros necesarios. Un archivo de cabecera contiene declaraciones de variables y constantes, prototipo de funciones, macros, etc.

Dentro de los archivos .h cabe destacar los mas importantes utilizados en el presente proyecto.

Fichero	Descripción
Main.h	Contiene todas las constantes, variables globales, configuración de la placa PICDEM.net 2 y configuración del puerto serie.
TramaEthernet.h	Contiene la configuración y definición de los pines utilizados para el modulo ENC28J60, como así también los protocolos que se utilizaran en las comunicaciones en red.
Stacktsk.h	Ubicado dentro de la carpeta "tcpip". Este fichero se utiliza para habilitar y deshabilitar funciones de la pila TCP/IP tales como DNS, ARP, etc, dependiendo de las funciones que se desea desarrollar en el proyecto.

Tabla 4: Ficheros .h Header Files

Dentro de "Header Files" se podrá observar que se incluyen todas las librerías necesarias para el funcionamiento de la pila TCP/IP. Para mas información consultar la documentación de Microchips Solutions.

- Other Files

Los ficheros ubicados en la carpeta "Other Files" contiene la implementación del resto del proyecto. La estructuración del proyecto en distintos ficheros fue útil para permitir el desarrollo claro, fácil de entender y facilitar la depuración de errores.

Fichero	Descripción
Trama.c	Fichero útil para el análisis de la trama recibida, donde se observará que tipo de trama recibimos. Contiene funciones para la formación y envió de trama ya sea mediante puerto serie o Ethernet. También podemos encontrar un par de funciones con la que se realizan el cambio de estado de las puertas lógicas.
Muestreo.c	Fichero encargado del muestro como del escaneo de todas las puerta lógicas especificas del PIC.
TramaEthernet.c	Fichero utilizado para las comunicaciones UDP. Configuración de MAC, IPAddress, etc. Inicialización del protocolo como recepción y envió de mensajes UDP.
Funciones.c	Fichero en el que se incluyen funciones varias tales como: ▪ config_pic(): Función donde se incluyen instrucciones de configuración del PIC. ▪ proceso_rx(): Función utilizada para el proceso de Keep Alive, si este esta activo, cada X segundos se encarga de enviar un mensaje sobre el estado de las puertas. ▪ ConfigPuertas(): Establece el estado de las puertas, si fueran salidas o entradas. ▪ Temporizador(): Establece los distintos tiempos de muestreo disponibles, según se indique.

Tabla 5: Ficheros .c Other Files

3.1.2.- Interrupciones empleadas

Los microcontroladores no pueden realizar varias acciones simultaneas como atender el cambio de estado de una puerta lógica y atender al temporizador que se ha

desbordado al mismo tiempo, por lo que para poder realizar estas acciones existen las interrupciones.

Las interrupciones han jugado un papel muy importante en la implementación del proyecto, ya que de ellas dependen que se pueda realizar la escucha del puerto serie o Ethernet cada media segundo, o que cuando se produzca un cambio de estado en una puerta lógica, el microcontrolador sea notificado, y pueda enviar las notificaciones necesarias al PC maestro.

- TIMER0

El Timer0 es un temporizador/contador ascendente de 8 bits. Este temporizador ha sido de gran importancia para el funcionamiento general del programa del microcontrolador. Según el modo de funcionamiento realizará una función u otra. Si el modo de funcionamiento es en modo puerto serie, se encargara de desactivar la escucha del puerto serie aproximadamente cada 500ms. Si el modo de funcionamiento es en modo red, este es encargado de temporizar los tickets para el funcionamiento de la red. También es el encargado de incrementar el contador de keep alive, si este estuviera activo.

- TIMER1

El Timer1 es un temporizador/contador ascendente parecido al TMR0, pero con algunas peculiaridades que lo hacen muy interesante a la hora de incluir temporizaciones en nuestro programa. Una de ellas es que se trata de un contador de 16 bits.

Para el proyecto se ha utilizado como temporizador, el cual es parametrizable mediante parámetros, como se ha mencionado en el capitulo anterior, en el cual hace referencia al umbral de muestro y es posible cambiarlo mediante parámetros.

- INTERRUPCION RDA DE LA USART (PUERTO SERIE)

La interrupción RDA se produce cada vez que en el puerto serie hay disponible un carácter para ser leído. Esta interrupción será deshabilitada si el usuario a la hora de establecer el modo de comunicación con la tarjeta de I/O indica modo Ethernet, ya que una vez establecida la configuración esta es desactivada.

3.1.3.- Diagramas de Flujo

Con el objetivo de llegar a entender correctamente el funcionamiento general del proyecto, se hará uso de diagramas de flujo que mostrarán de manera clara y concisa el

comportamiento de los pasos llevados a cabo. Cabe destacar que se mostrarán algunas de las subrutinas implementadas en el proyecto puesto que el objetivo principal es dar a entender la implementación.

A continuación se mostrarán las rutinas del funcionamiento de los diferentes modos de funcionamiento de la tarjeta de I/O.

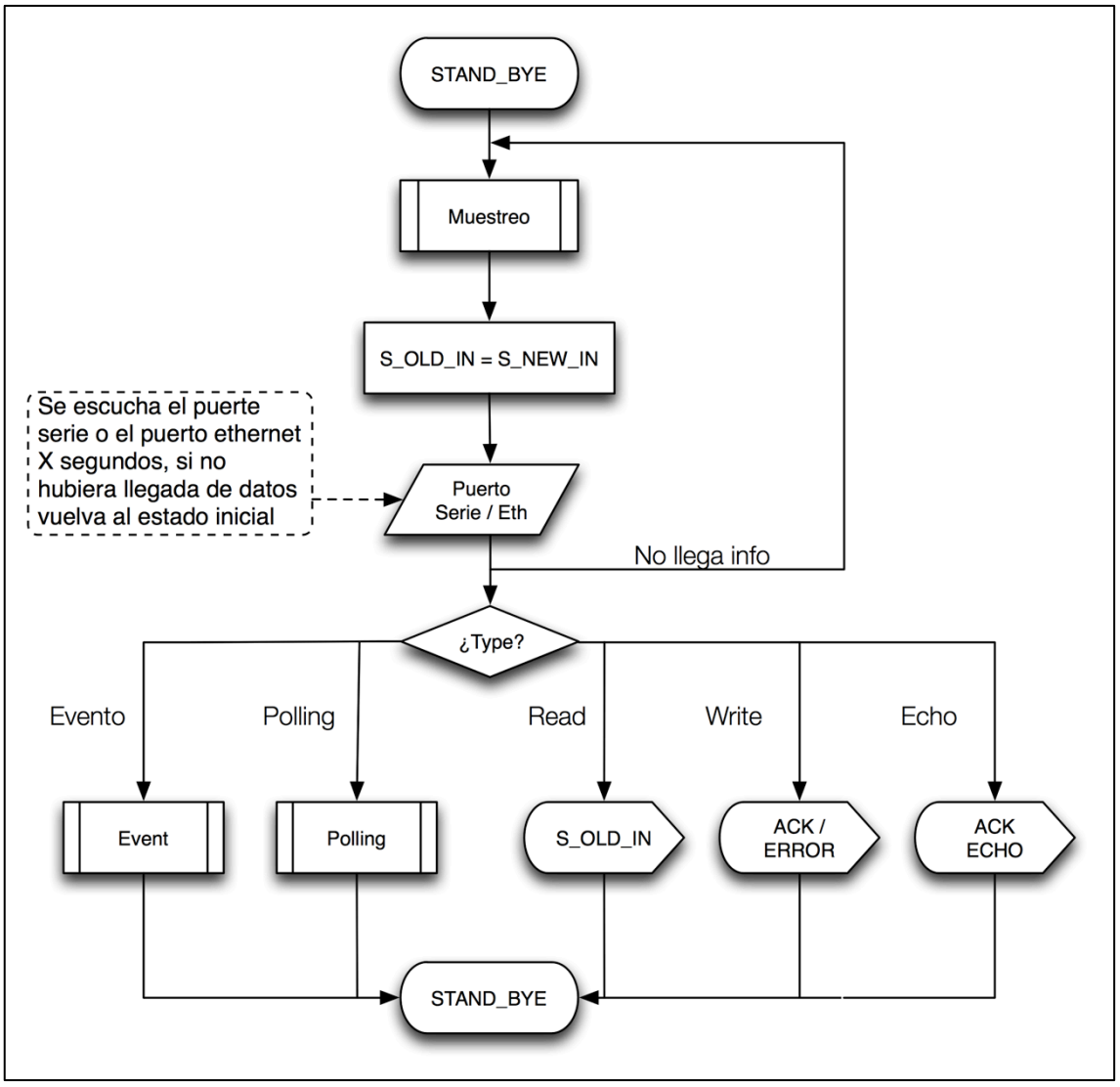

Figura 16: Diagrama de Flujo. Rutina de estado STAND BYE

Como se ha comentado anteriormente, el estado en el que inicia la tarjeta de I/O es en modo STAND_BYE. Observando la Figura 16, una vez entramos en el modo "stand_bye", primero se realiza un muestreo de las puertas lógicas, para posteriormente almacenar en S_OLD_IN el nuevo estado de las puestas lógicas. A continuación la tarjeta se sitúa a la escucha durante aproximadamente 500ms, esperando recibir alguna de las tramas de tipo.

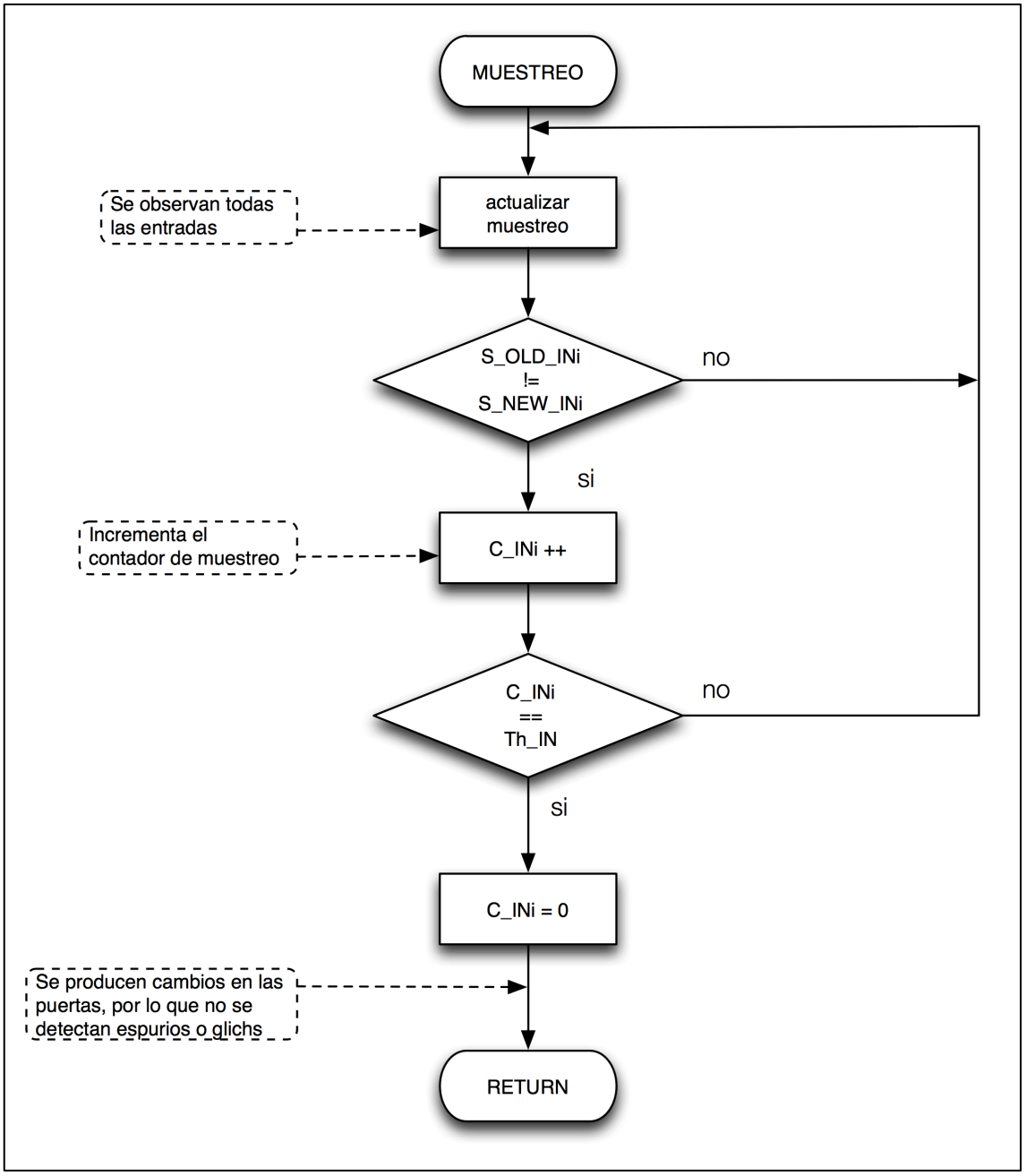

Figura 17: Diagrama de Flujo. Rutina de Muestreo.

La rutina muestreo es la encargada de realizar las comprobaciones de los estados de las puertas lógicas. En la subrutina “actualizar muestreo” es donde se realiza la operación de escaneo de las puertas lógicas. En esta rutina “muestreo” se realiza una comprobación comparando el umbral de muestreo con un contador temporal llamado C_IN. Esta comprobación se realiza para comprobar que las señales que llegan a la tarjeta sean correctas, y no espurios.

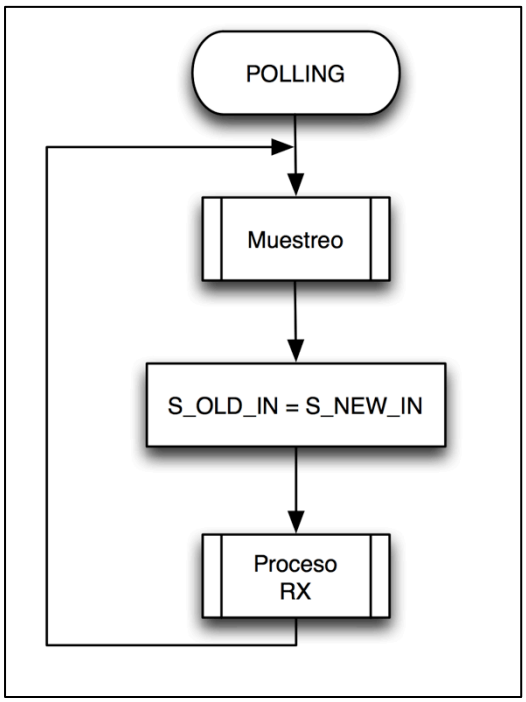

Figura 18: Diagrama de Flujo. Rutina de estado POLLING

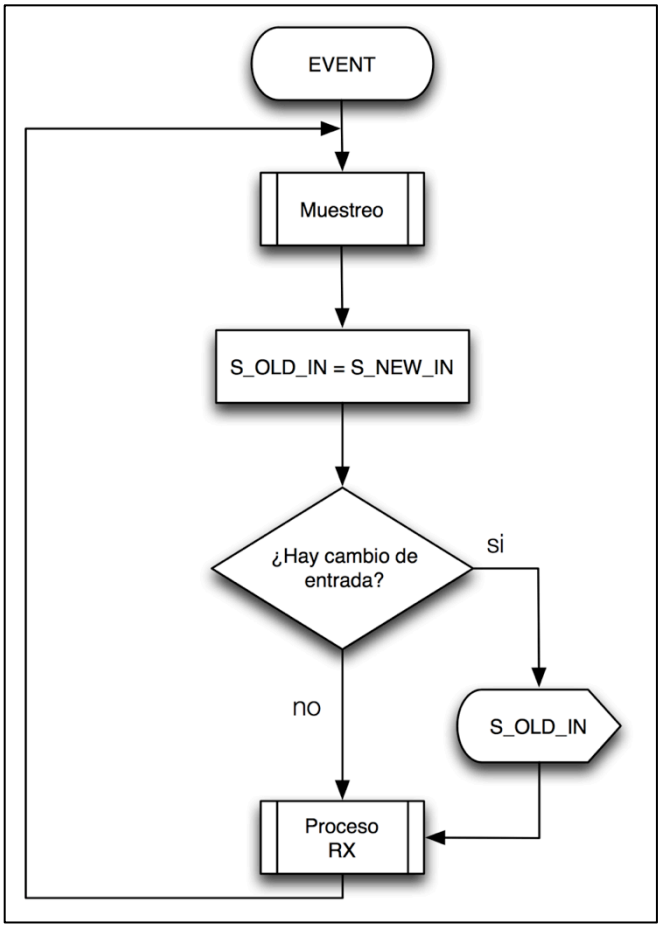

Figura 19: Diagrama de Flujo. Rutina de estado EVENT.

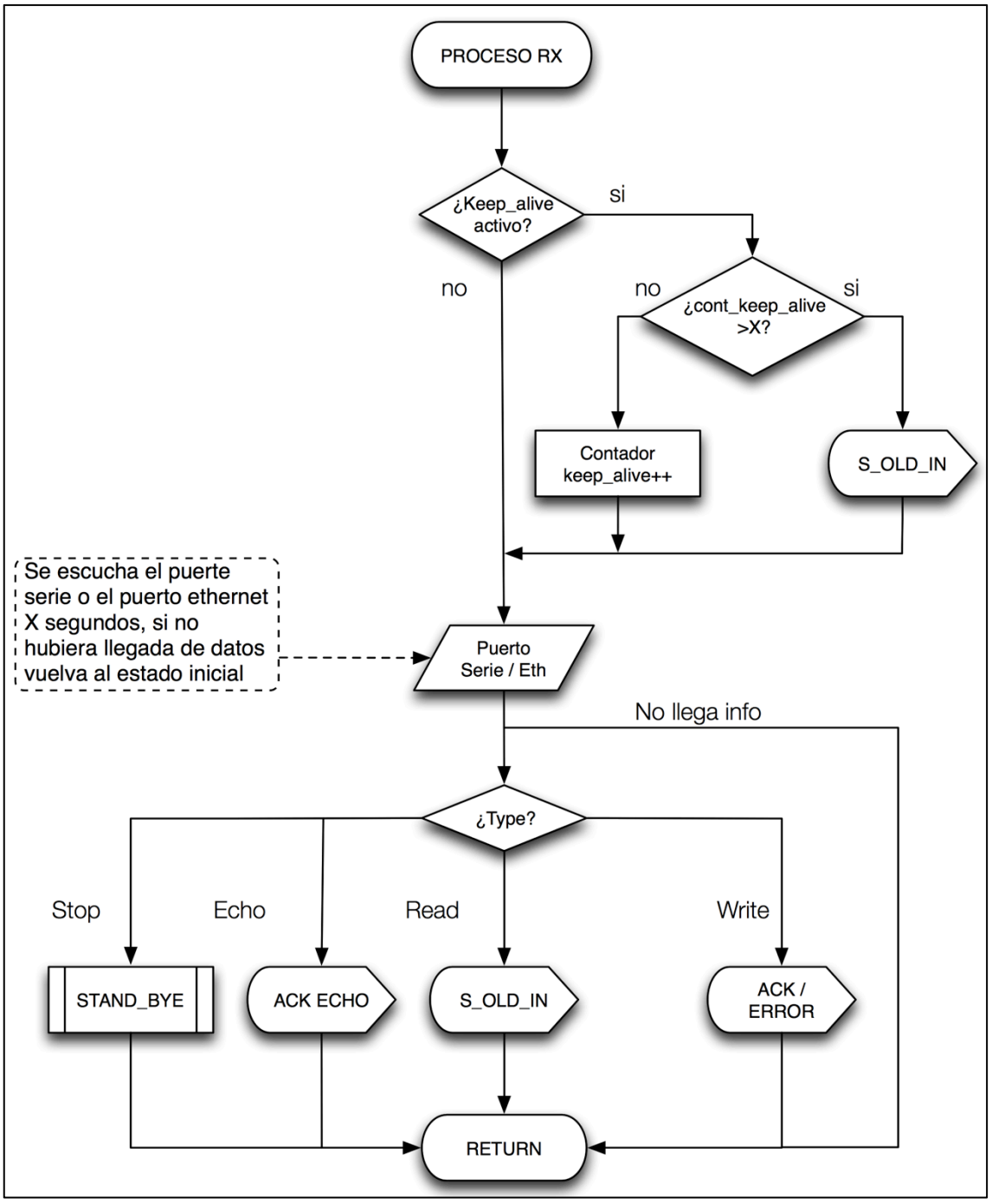

Figura 20: Diagrama de Flujo. Rutina proceso_rx

Observando la Figura 18, notamos que el modo POLLING es muy sencillo. En la rutina “polling” simplemente se realiza un muestreo y una copia del nuevo estado de las puertas lógicas.

En el estado EVENT, como se puede ver en la Figura 19, el proceso que lleva a cabo es similar a la rutina polling. Esta rutina con diferencia a la de polling es que si se

detecta un cambio de estado de las puertas lógicas ésta envía al PC maestro los nuevos estados de las puertas lógicas almacenados en S_OLD_IN. Como se podrá ver, tanto en la rutina polling como event el resto de operaciones son llevadas a cabo por la rutina proceso_rx. Esto se ha realizado ya que simplifica el desarrollo y las operaciones que realizan a posteriori son las mismas.

En la rutina proceso_rx, ver Figura 20, es en donde llevamos a cabo de las operaciones de comprobación de keepalive. Estas operaciones que realiza sería primero, comprobar si estuviera activo, en caso afirmativo se comprueba que el contador haya alcanzado el valor preestablecido, y si es así, envía al PC maestro el estado de las puertas lógicas. Al igual que la rutina STAND_BYE, el siguiente paso es mantenerse a las espera por si llegase alguna petición por parte del PC maestro.

3.2.- Implementación de la API en C#

Para empezar, ¿qué es una API?. Una IPA, interfaz de programación de aplicaciones o API (del ingles) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente como librerías.

Para llevar a cabo el control de la tarjeta de I/O se ha tenido que desarrollar una API, que realizará la tarea de driver, controlando a bajo nivel la tarjeta basado en PIC. Esta API implementa todas las funciones necesarias para tener el control total de la tarjeta de I/O y poder ser utilizada por otra aplicación de alto nivel.

3.2.1.- Descripción API

A continuación se realizará una explicación del funcionamiento y estructuración de la API. Además, se realizará una aclaración de los pasos que hay que realizar para poder ser utilizada en una aplicación de alto nivel.

EVENTOS

Los eventos proporcionan un medio de que una clase u objeto informa a otras clases u objetos cuando sucede algo relevante. La clase que envía (o produce) el evento recibe el nombre de editor y las clases o métodos que reciben (o controlan) el evento se denomina suscriptores.

Por lo tanto dentro de la API han sido necesario la utilización de eventos para el control de recepción de tramas, o sea cuando recibo mediante puerto serie o Ethernet.

Los eventos tienen las propiedades siguientes:

- El editor determina cuándo se produce un evento; los suscriptores determinan qué operación se realiza en respuesta al evento.
- Un evento puede tener varios suscriptores. Un suscriptor puede controlar varios eventos de varios editores.
- No se llama nunca a los eventos que no tienen suscriptores.
- Los eventos se utilizan normalmente para señalar acciones del usuario como hacer clic en un botón o seleccionar un menú en interfaces gráficas de usuario.
- Si un evento tiene varios suscriptores, se invocan los controladores de eventos sincrónicamente cuando se produce el evento.
- Los eventos se pueden utilizar para sincronizar subprocesos.
- En la biblioteca de clases .NET Framework, los eventos se basan en el delegado EventHandler y en la clase base EventArgs.

La suscripción a un evento publicado por otra clase se realiza cuando se desea escribir código personalizado al que se llama cuando se produce ese evento. Por ejemplo, puede suscribirse al evento "click" de un botón para que la aplicación realice alguna operación cuando el usuario haga clic en el botón.

Dentro de la API se han implementado los siguientes eventos:

- Evento “EventoCambioEntradas”. Este evento es el encargado de notificar a la aplicación de alto nivel, que ha habido un cambio de estrada, o sea que se ha recibido el comando SEND_IN por parte del PC maestro. Al tratar este evento se notifica con array de bytes el estado de las puertas lógicas y un entero que representa la secuencia de trama.
- Evento “EventoRespuestaEco” encargado de notificar la llegada de una trama de tipo ACK_ECHO.
- Evento “EventoRespuestaWrite” encargado de notificar la llegada de una trama de tipo ACK_WRITE.
- Evento “EventoRespuestaNoWrite” encargado de notificar la llegada de una trama de tipo ACK_NO_WRITE.
- Evento “EventoRespuestaWriteOne” encargado de notificar la llegada de una trama de tipo ACK_WRITE_ONE.
- Evento “EventoRespuestaNoWriteOne” encargado de notificar la llegada de una trama de tipo ACK_NO_WRITE_ONE.

Si bien estos eventos deben ser tratados en la aplicación de alto nivel, la cual será notificada para realizar las operaciones pertinentes.

DESCRIPCIÓN DE FUNCIONES

A continuación se comentarán las funcionalidades de las funciones empleadas en la API. Comenzaremos con los constructores ya que dependiendo de que tipo de comunicación que se desea o que configuración utilizar, es necesario utilizar un tipo de constructor u otro.

Antes de empezar a explicar los diferentes tipos cabe destacar que el modo de configuración siempre se realizá desde un primer momento mediante puerto serie, y siempre a una velocidad de 9600bps, por lo que se tendrá que tener cautela a la hora de realizar ciertas acciones. Los problemas que pueden llegar a surgir es que si se establece una velocidad de trabajo mayor a 9600bps y el desarrollador de la aplicación de alto nivel no controla la ejecución de envió de configuración nuevamente, podrá producir inestabilidad en la tarjeta de I/O. Esto es debido a que la velocidad de trabajo entre el PC maestro y la tarjeta no son las mismas. Este problema solamente podrá surgir cuando se desee trabajar en modo puerto serie.

Dentro de la API, de ahora en adelante librería, existen tres tipos de constructores:

1. Si se utiliza puerto serie, el cual se le será necesario pasarle el nombre del puerto COM que utilizaremos, velocidad de trabajo definida en bps, identificador que empleará la tarjeta y por ultimo el identificador del PC.

```
TarjetaPic(string puertoCom, int velocidadBps, byte idPic, byte
idPc)
```

2. Este segundo constructor es para utilizar la IP por defecto, con lo que hay que pasarle el puerto COM que utilizaremos para establecer la configuración y los identificadores del PIC y el PC. Si se utiliza comunicación por red hay que tener en cuenta si se desea utilizar una IP de nuestra red o la dirección que se le fue asignada por defecto (192.168.1.254).

```
TarjetaPic(string puertoCom, byte idPic, byte idPc)
```

3. Como ultima opción hay un tercer constructor el cual se utiliza para usar una IP fija. Ha este constructor habría que pasarle el puerto COM, un array de byte el cual contendrá la dirección IP y la mascara de red, identificador que tendrá la tarjeta y el identificador del PC.

```
TarjetaPic(string puertoCom, byte[] ip, byte idPic, byte idPc)
```

Una vez realizada la creación del objeto en la aplicación principal, es necesario establecer el medio de comunicación mediante el uso de la función "EstablecerMedio()", la cual crea la trama necesaria para establecer el medio de comunicación que se utilizará en el PIC para la comunicación con el PC. Un vez ya realizados estos pasos, la tarjeta de I/O, ya estaría configurada y lista para recibir peticiones. El resto de las funciones que componen la API no necesitan mayor explicación ya que la misma contiene una explicación de su función, pero para descartar dudas se realizará una pequeña explicación de las mismas. Se hará hincapié en las principales funciones ya que el resto son para funcionamiento interno de la API.

- Conectar(): Esta función realiza la comunicación con la tarjeta de I/O. Si se realiza por puerto serie, esta abre el puerto, y si es por red, la librería ejecuta un Thread, el cual se encargará de abrir el puerto UDP y estar a la escucha de recibir peticiones.

- Desconectar(): Como bien indica su nombre realiza la desconexión con la tarjeta de I/O.

- EstablecerModo(): Establece el modo en el que se desea trabajar en el PIC, ya sea polling, evento o stand_bye.

- enviarEcho(): Envía una trama ECHO.

- enviarRead(): Envía una trama READ_IN.

- setSalidas(): Función que envía una trama WRITE_IN. A esta función es necesario pasarle un array de Bits con el estado de las puertas.

- setSalida(): Realiza el cambio de estado de la puerta deseada. Hay disponible 30 salidas:
    - PUERTA A --> 0 - 5
    - PUERTA E --> 8 - 15
    - PUERTA H --> 16 - 23
    - PUERTA J --> 24 - 31
    - NOTA: Los valores 6 y 7 no son validos, ya que no corresponde a salidas validas.

Capitulo 4

Pruebas y Resultados

4.1.- Implementación

En la fase de implementación de este proyecto hubo que empezar con el estudio de el microcontrolador PIC18FJ60. Una vez comprendido el funcionamiento, se designo por abordar desde aplicaciones pequeñas hasta las mas complejas. Se opto por realizar una aplicación en el microcontrolador en donde se implemento una primera fase del protocolo de comunicación PIC-PC mediante puerto serie. Esto permitió observar cual era el rendimiento de este, es decir, comprobar las velocidades soportadas mediante puerto serie enviando y recibiendo tramas sin que se produzcan perdida de datos.

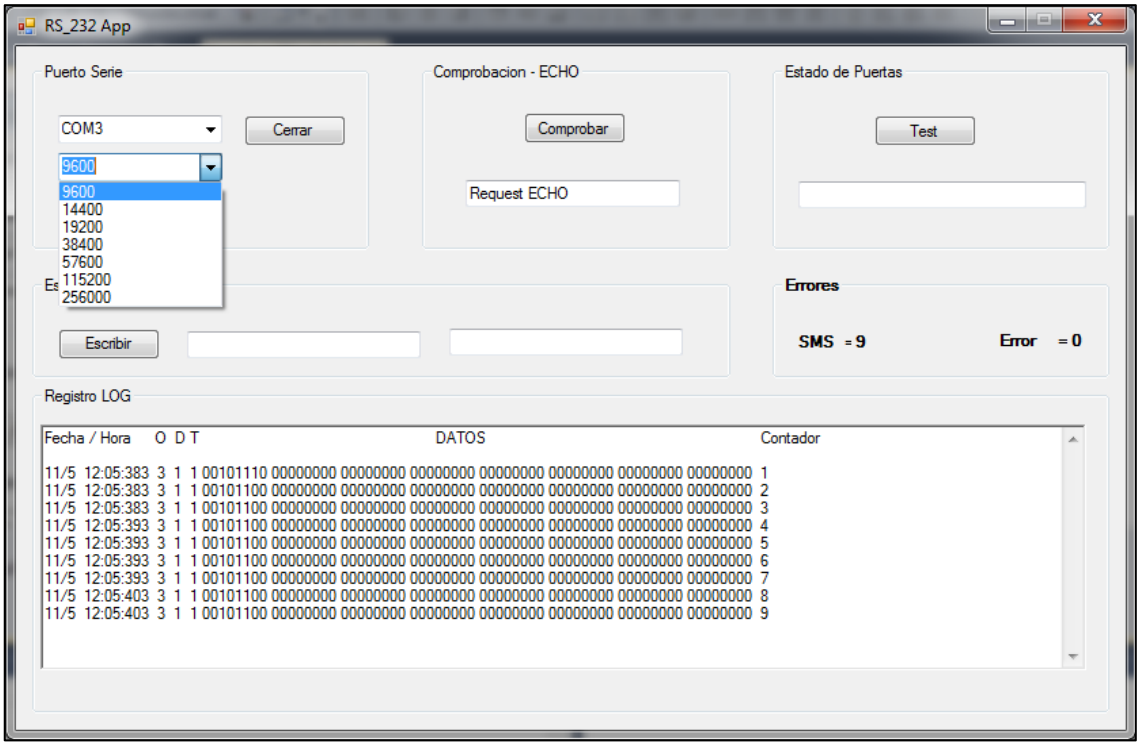

Figura 21: Primer Fase de la Aplicación de Pruebas

En la aplicación de la primera fase, ver Figura 21, se consiguió realizar pruebas de comunicación en todas las velocidades disponibles (desde 9600bps a 256000bps), enviando aproximadamente 50.000 tramas teniendo una tasa de errores nula. Esto llego

a comprobarse debido a la creación de un registro log, que a su vez fue almacenado en un fichero de texto.

Mediante la primera fase de la aplicación, en la cual hemos conseguido buenos resultados, nos llevó a efectuar una segunda aplicación. En esta segunda aplicación hemos optado por implementar las comunicaciones con el PIC mediante Ethernet-UDP. Para el desarrollo de las comunicaciones UDP se ha tenido que hacer un estudio del manejo de la pila TCP/IP, como desarrollar sockets en microcontroladores y como configurar el PIC para poder comunicarse con el modulo ENC28J60 integrado en la tarjeta de pruebas.

Tras el estudio y pequeñas aplicaciones de pruebas, para testear las comunicaciones mediante Ethernet, se llego a una aplicación funcional en la cual se podían mantener comunicación mediante UDP o puerto serie.

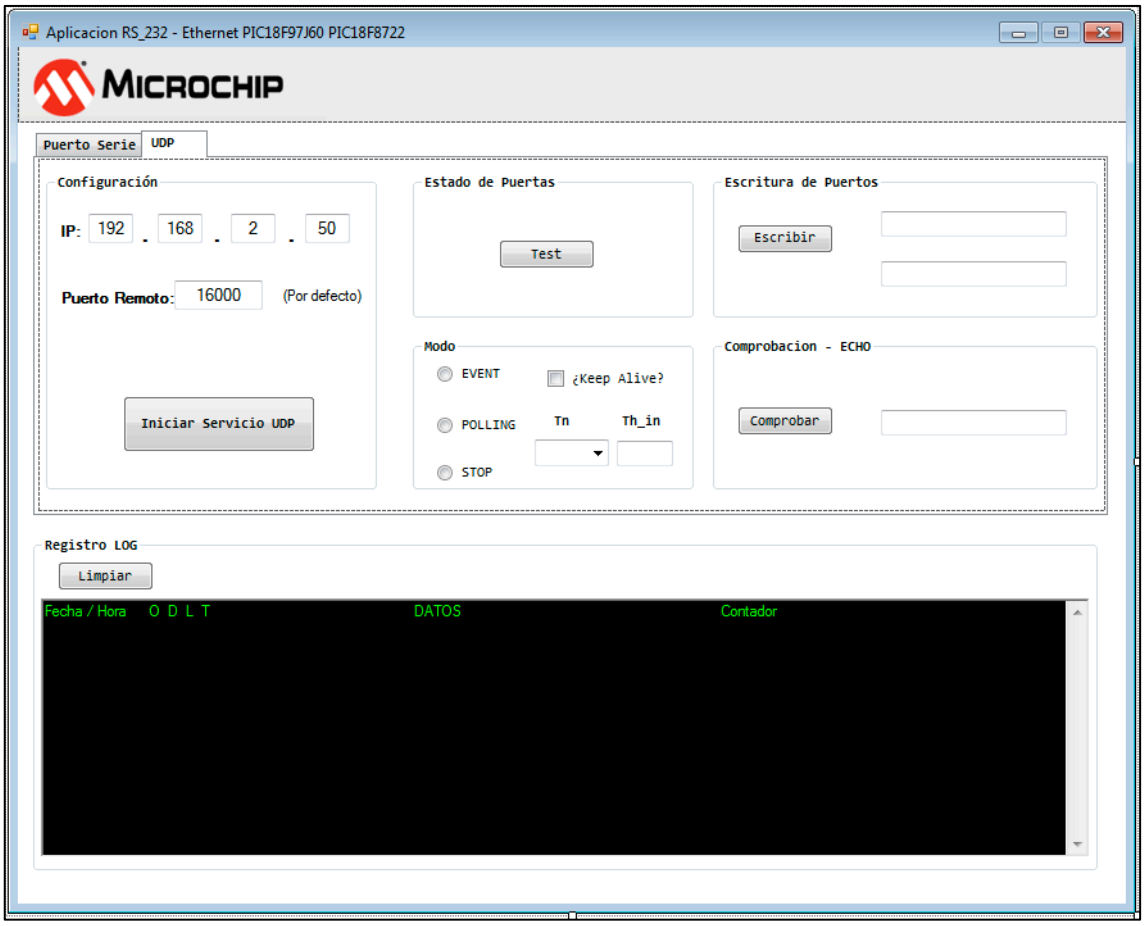

Figura 22: Segunda Fase de Aplicación de Prueba mediante comunicación Ethernet y Serie.

4.2.- Aplicación de Prueba

Tras haber comentado el desarrollo llevado a cabo durante la ejecución del proyecto, se llevó a cabo el desarrollo de la librería comentado en el capitulo 3. Para las realizaciones de las pruebas de esta librería y pruebas de funcionamiento general se decidió realizar una aplicación que consta de dos partes, una de ellas es la configuración de la tarjeta de I/O y otra para el funcionamiento.

- Configuración

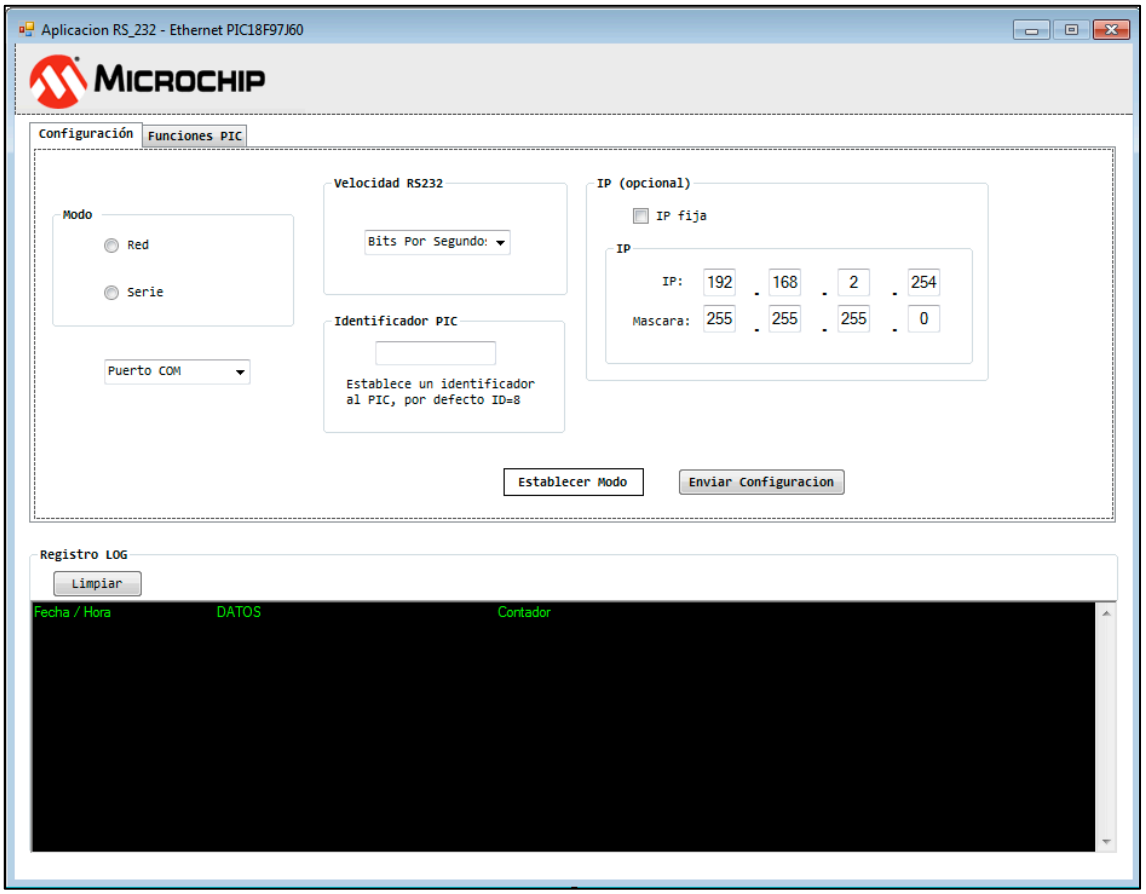

Figura 23: Interfaz Grafica para la configuración de la tarjeta de I/O

Como se observa en la Figura 23, se pueden configurar los parámetros necesario para la iniciación de la tarjeta de I/O. Además se han establecido dos botones, “establecer modo” que indica al programa el medio que se utilizara y “enviar configuración” el cual envía a la placa los datos de configuración. Esto se ha realizado de este modo para que si la tarjeta ya se encuentra en ejecución poder retomar el estado en el que se encuentra.

- Funcionamiento Tarjeta I/O

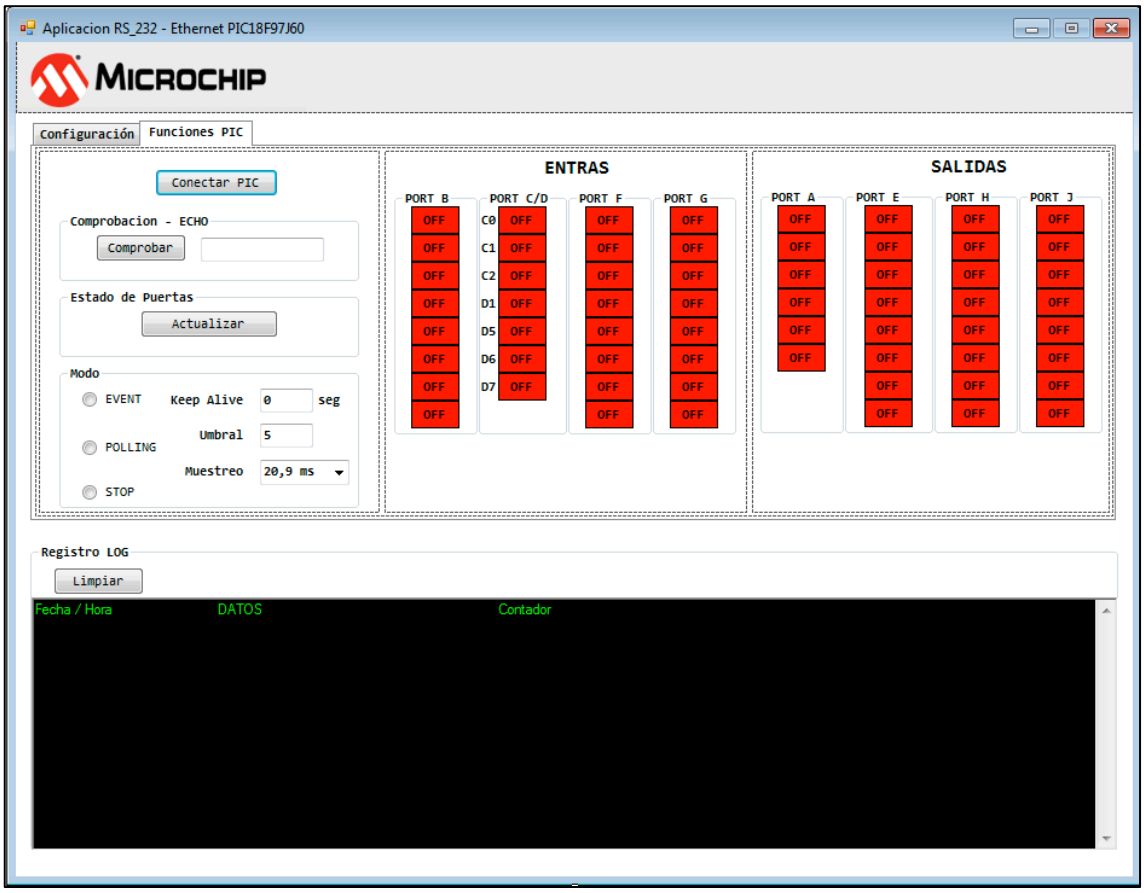

Figura 24: Interfaz Grafica del Funcionamiento de la Tarjeta de I/O

Para llevar a cabo la comunicación con la tarjeta será necesario realizar una conexión. Una vez realizada la comunicación, se podrán enviar comandos como los que aparecen en la Figura 24. Estos serán, comprobar la comunicación, comprobar estado actual de las puertas lógicas y como ultimo el modo de funcionamiento con sus opciones de configuración. En las dos columnas de la derecha, entradas y salidas, se podrá observar el estado de cada una de las puertas lógicas en tiempo real.

La interfaz también incorpora una zona de registro log o zona de reporte donde se mostrarán errores que puedan surgir en la comunicación y todos los registros de datos recibidos.

Capitulo 5

Conclusiones

5.1.- Valoración de Resultados.

Como resultado de este trabajo se ha ve por concluido los objetivos propuestos:

- Estudiar el marco teórico referente a la tecnología PIC-Ethernet, y PIC-RS232.
- Diseñar un protocolo de comunicación
- Diseño y fabricación de una placa prototipo con entradas digitales y relés.
- Implementación de los drivers de la tarjeta de desarrollo basada en una API en C#.
- Determinar la eficiencia del sistema, en base a pruebas del funcionamiento.

Valorando los resultados obtenidos puede afirmarse que se han alcanzado los objetivos planteados inicialmente, que como objetivo final era realizar un prototipo de una tarjeta de adquisición de datos, para luego implementarlo en otro sistema de mayor envergadura.

No obstante, al tratarse de un primer prototipo esta sujeto a posibles modificaciones y mejoras en futuras versiones, en la que se podrán añadir nuevas funcionalidades y la posibilidad adaptarse a un sistema en escala.

5.2.- Valoración Personal

Personalmente me he visto realizado con el resultado final del proyecto y no solo porque se hayan cumplidos los objetivos planteados, sino porque los conocimientos adquiridos en los años de carrera no han sido en vano. Durante el transcurso del proyecto me he sentido muy cómodo gracias a la materia del proyecto, especialmente por el hecho de que combinara el desarrollo de software (programación en lenguaje C para microcontroladores como Visual Studio C#.net) con hardware y electrónica.

Personalmente algo que me ha sorprendido mucho es que a pesar de que este trabajo es la cima de una carrera universitaria, he aprendido muchísimo, y de muchos temas. A pesar de haberme enfrentado a conceptos técnicos nuevos para mi, con mis propios conocimientos he sido capaz de abordarlos y poderlos llevar a cabo. Supongo que los conocimientos que me han sido aportados en esta carrera empieza a dar sus frutos.

Referencias

- Guia de Programacion de C Sharp
http://msdn.microsoft.com/es-es/library/67ef8sbd(v=vs.80).aspx

- Ayuda y Soporte Tecnico de Microsoft España
http://support.microsoft.com

- Data Sheet PIC18FJ60
http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026445

- Data Sheet PICDEM.net 2
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en028217

- Data Sheet ENC28J60
http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en022889

Anexo

Modulo de Comunicación Ethernet ENC28J60

Descripción del producto

El ENC28J60 es un controlador independiente Ethernet con una interfaz estándar Serial Peripheral (SPI). Se ha diseñado para servir como una interfaz de red Ethernet para cualquier controlador equipado con SPI. El ENC28J60 cumple con todas las especificaciones IEEE 802.3. Se incorpora una serie de esquemas de filtrado de paquetes para limitar los paquetes entrantes. También proporciona un módulo interno de DMA para la transferencia de datos rápida y cálculo de suma de comprobación de hardware asistida, que se utiliza en varios protocolos de red. La comunicación con el controlador de host se implementa a través de un pin de interrupción y el SPI, con velocidades de reloj de hasta 20 MHz. Dos pines dedicados se utilizan para la conexión de LED y la indicación de actividad de la red.

Este modulo integrado en la placa de desarrollo PICDEM.net 2 es una solución perfecta para controlar tu aplicación de forma remota a través de tu red, ya sea mediante aplicaciones web o a través de correos electrónicos para advertir de incidencias.

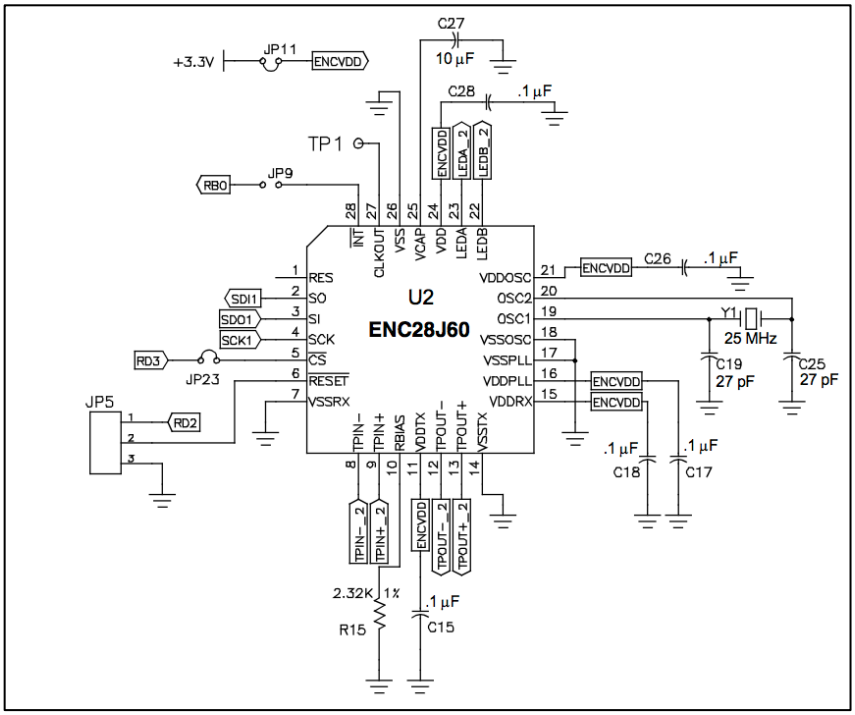

Figura 25: Esquema eléctrico del modulo Ethernet integrado en PICDEM.net 2

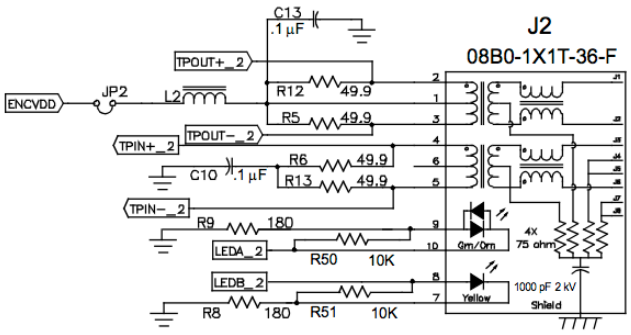

Figura 26: Esquema eléctrico adaptador RJ-45 placa PICNET.net 2

Descripciones de los Pines

Pin	Descripción
VCC	Conexión de alimentación 5V o 3V3
GND	Conexión de masa
CSK	Entrada de reloj SPI interface
RST	Pin de reset a nivel bajo
CS	Selección de chip mediante SPI
MOSI	Pin de entrada de datos SPI
MISO	Pin de salidas de datos SPI
WOL	No conectado
INT	Pin de interrupción
CLKO	Reloj de salida programable

Tabla 6: Descripción de conexión de pines

Ejemplo de código modulo-entrenadora

```
#define  PIN_ENC_MAC_SO    PIN_C4   // Conectar con PIN MISO del ENC28J60.
#define  PIN_ENC_MAC_SI    PIN_C5   // Conectar con PIN MOSI del ENC28J60.
#define  PIN_ENC_MAC_CLK   PIN_C3   // Conectar con PIN SCK del ENC28J60.
#define  PIN_ENC_MAC_CS    PIN_D3   // Conectar con PIN CS del ENC28J60.
#define  PIN_ENC_MAC_RST   PIN_D2   // Conectar con PIN RST del ENC28J60.
#define  PIN_ENC_MAC_INT   PIN_D0   // Conectar con PIN INT del ENC28J60.
#define  PIN_ENC_MAC_WOL   PIN_D4   // Conectar con PIN WOL del ENC28J60.
```

Figura 27: Ejemplo código conexión modulo-entradora