Adaptador USB a LPT para la recuperación de equipos de rehabilitación

Javier Barragan; Fernando Anaut, Jorge Osio*[1]; José Rapallini[1]; Flavio Ferrari [2];
Facultad de Ingeniería - Universidad Nacional de La Plata

jorge.osio@ing.unlp.edu.ar.
1 Centro de Técnicas Analógico – Digitales (CeTAD)
2 UNITEC
*Becario CIC – Comisión de Investigaciones Científicas de la Prov. de Bs. As

Resumen — El proyecto contempla el diseño de un Sistema Emulador de un puerto LPT a través de la interfaz USB.

Debido a la necesidad de recuperar los equipos que requieren una interfaz LPT y a la extinción de esta interfaz en las computadoras modernas, surge esta propuesta orientada a la recuperación de equipos de rehabilitación.

En este proyecto solo se trata el modo unidireccional SPP del protocolo LPT, dejando las bases necesarias para tratar los demás modos en un futuro proyecto.

Una vez seleccionados y estudiado dichos componentes, se procede con el diseño de cada componente que forma parte del Sistema emulador. La creación de una Interfaz Gráfica de Usuario para que el usuario pueda controlar desde la PC el dispositivo LPT conectado al sistema, la creación de un archivo INF único para que la PC reconozca el sistema emulador, la selección de los archivos DLL necesarios para manejar los puertos de la PC, y la escritura de un firmware con las funciones necesarias para que el microcontrolador maneje las diferentes tareas de la interfaz; forman parte de las tareas que comprenden la segunda etapa.

Palabras Clave – Sistemas Embebidos, Protocolo USB, Protocolo LPT, Codiseño HW/SW .

I. INTRODUCCIÓN

Este proyecto está orientado a la reutilización de dispositivos electrónicos utilizados por personas que sufran algún tipo de discapacidad. Integrándose a otros proyectos (distintos tipos de switch, control de entorno, etc.) que surgieron dentro del UNITEC (Unidad de Investigación y Desarrollo para la Calidad de la Educación en Ingeniería con orientación al uso de TIC), con el fin de brindar los elementos necesarios para solucionar las dificultades que les impiden en normal aprendizaje a personas que sufran algún tipo de discapacidad.

A partir de la necesidad de que ciertos dispositivos no queden obsoletos, el proyecto contempla el estudio del funcionamiento de los diferentes protocolos para luego diseñar, tanto en Hardware como en Software, la solución necesaria para que dichos dispositivos puedan seguir teniendo la utilidad a pesar de la modernización de los demás elementos dentro del entorno en que son utilizables. La elección del puerto USB como interfaz de adaptación, tiene que ver con la existencia del mismo en todas las computadoras actuales.

Tanto el Protocolo USB como el Protocolo LPT, forman parte fundamental del trabajo porque son los medios por los que la información viaja hacia los “extremos” de la Interfaz. La información se transmite por el medio USB entre la PC y la interfaz, luego en el dispositivo se añade el dato al medio LPT.

El microcontrolador MC9S08JM60 y la placa de desarrollo DEMOJM, conforman la totalidad del Hardware de la Interfaz. El microcontrolador se caracteriza por tener un Módulo controlador USB, con lo cual se justifica su elección debido al tipo de comunicación que se realiza en este proyecto. Particularmente el microcontrolador forma parte de la DEMOJM, ya que está última es la que acondiciona el camino de la información, que llega tanto de la PC como del dispositivo, para que el microcontrolador tenga la oportunidad de analizarla y procesarla según la función asignada.

Una GUI (Graphical User Interface) es una interfaz gráfica en la PC, en la que el usuario puede controlar o manipular el dispositivo conectado mediante el envío de diferentes tipos de datos. Para poder lograr este fin, la GUI utiliza archivos dll. Los archivos DLL, son los archivos que dan la posibilidad de manipular a gusto los periféricos de la PC, a través de funciones propias que cada dll brinda al usuario. En este caso, se utilizan dos archivos dll para poder manipular tanto los puertos paralelos LPT como los puertos USB de la PC a través de la creación de una GUI particular para el proyecto.

El archivo INF es un archivo de texto que contiene toda la información necesaria para que la PC

reconozca que tipo de dispositivo se conectó a ella y además se encarga de la instalación de drivers de dispositivos para los componentes de hardware. Cada tipo de dispositivo debe tener un archivo INF único. En este proyecto, se diseñó el archivo INF que contiene los datos necesarios para que a pesar de que la interfaz se conecte vía USB, la PC reconozca el dispositivo conectado como si fuera un LPT.

El firmware es el encargado de indicarle al microcontrolador como debe funcionar según la aplicación a la cual fue destinado. Está compuesto por archivos que contienen las configuraciones y las funciones de todos los elementos que componen al microcontrolador.

Tanto el Firmware, como el archivo INF y la GUI; forman parte fundamental de la comunicación entre la PC y el dispositivo. Para lograr este fin, tiene que existir algo que los vincule y que asegure la comunicación entre ellos. Como se observa en la figura 1, para lograr la conexión entre el archivo INF y la PC existe un identificador llamado GUID que figura tanto en la GUI como en el archivo INF y por el cual se asegura la conexión entre estos dos bloques. Para la conexión entre el archivo INF y el Firmware, los identificadores que deben estar presente tanto en el archivo INF como en el firmware para asegurar la conexión, son los descriptores USB como por ejemplo el PID y el VID.

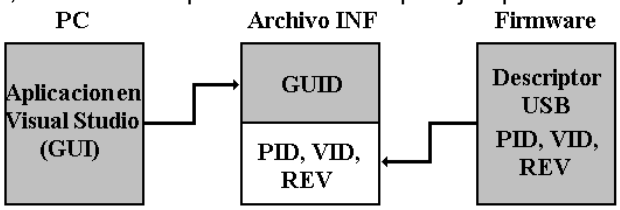

Figura 1. Diagrama en Bloques PC-INF-Firmware.

II. DESCRIPCIÓN DE DRIVERS Y LA GUI DE APLICACIÓN

La Interfaz Gráfica de Usuario, a la que se referirá como GUI hace el manejo de la computadora más intuitivo usando metáforas gráficas. La GUI es la herramienta necesaria que el Usuario tiene para poder manipular desde la PC el puerto paralelo LPT y el dispositivo LPT que se conecte al Sistema emulador. Esto se logra debido a que la GUI contiene los drivers necesarios para cumplir con las funciones de escribir y leer el puerto LPT de la PC y además enviar o recibir vía USB los diferentes tipos de información que provienen o se envían hacia el sistema emulador.

La composición de la GUI de este proyecto, se muestra en la figura 2, en donde se observa el registro de datos, el de estados y control.

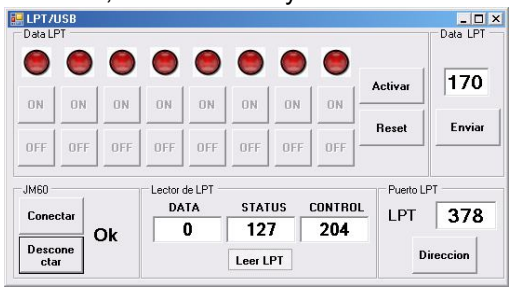

Figura 2. GUI creada para el proyecto

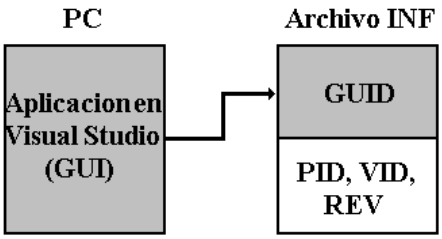

Figura 3. Conexión entre la GUI y archivo INF

La forma de vincular nuestra GUI con el archivo INF, es a través del identificador GUID, que es un número único que debe estar presente tanto en el archivo INF como en la GUI. Para esta última parte, este número se agrega en las propiedades de la dll que vincula la conexión USB [1], el archivo SimpleUSB.dll [2].

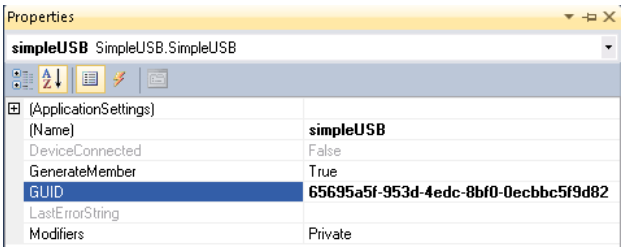

Figura 4. Declaración del GUID en el Visual Studio

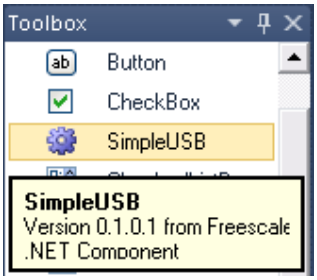

Figura 5. Icono del driver SimpleUSB

Por lo tanto, si se configura correctamente el GUID en la GUI y el INF se soluciona una parte de la conexión USB.

Para vincular el driver simpleusb.dll a una GUI, no hace falta más que arrastrarlo dentro del cuadro de herramientas (toolbox) del Visual Studio. Luego, aparecerá un ícono que representará al driver, como muestra la figura siguiente.

Luego, con solo arrastrar dentro del cuadro de diseño el ícono ya se tiene a disposición el driver en nuestra aplicación. Por lo tanto, con solo utilizar dentro del código las funciones que el driver tiene se podrá manipular el puerto USB como se necesite, tanto como para enviar como para recibir datos.

Es un poco más complicado vincular el archivo inpout32.dll ([4] y [5]). Para lograr dicha vinculación es necesario crear un archivo .cs, donde se contiene el llamado al archivo y la declaración de sus funciones. En la figura siguiente se observa dicha declaración que se encuentra en el archivo .cs creado.

```
[DllImport("inpout32.dll", EntryPoint = "Out32")]
public static extern void Output(int adress, int value);
[DllImport("inpout32.dll", EntryPoint = "Inp32")]
public static extern int Input(int adress);
```

Figura 6. Declaración del driver Inpout32.dll

Luego con solo llamar en el código principal de la aplicación a sus funciones, se podrán manipular sin ningún problema los puertos LPT.

III. FIRMWARE DEL MICROCONTROLADOR

El firmware es el software que contiene el microcontrolador, el cual contiene las diferentes funciones necesarias para manejar o controlar las tareas que el microcontrolador debe realizar. En esta aplicación el firmware es el encargado de contener toda la configuración y funciones necesarias para comunicar la placa DEMOJM, con la PC y con el dispositivo LPT. Además de encargarse de recibir los datos en formato USB provenientes de la PC y transformarlos para enviarlos en formato paralelos al dispositivo LPT.

Como se indica anteriormente, y como indica la figura 7, existen identificadores presentes tanto en el archivo INF como en el Firmware que deben coincidir sus valores para poder asegurar la conexión. Estos identificadores se conocen con el nombre de Descriptores USB [6],[7].

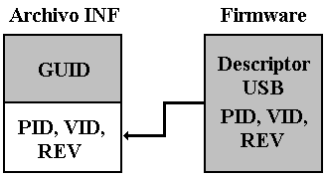

Figura 7: Conexión entre Archivo INF y el Firmware

A. *DESCRIPTORES DE USB*

Para lograr la conexión entre el archivo INF y el firmware que se encuentra en el microcontrolador, se requieren unos números identificadores característicos en una comunicación USB que proporcionan la información que define qué tipo de dispositivo se está queriendo conectar al host USB de la PC. A este tipo de números se los denomina Identificadores o Descriptores [6].

Todos los dispositivos USB tienen una jerarquía de descriptores que describen el dispositivo al host, con el siguiente tipo de información:

- Que dispositivo es.
- Quién lo fabrico.
- Qué versión de USB es compatible con él.
- De cuántas maneras puede ser configurado.
- El número de endpoints y sus tipos, etc.

Cada uno de estos ofrece una pequeña descripción del comportamiento del dispositivo. Los descriptores se encuentran en el archivo Usb_Description.c del firmware.

A continuación, se realiza una descripción de los descriptores que se modificaron para la aplicación.

B. *DESCRIPTORES DE DISPOSITIVOS*

Cada dispositivo USB conectado a un host USB debe tener un identificador único de proveedor o

fabricante (VID) [6], un identificador de producto (PID), la combinación de cadenas de serie (Serial Strings), entre otros identificadores. Estos en particular se encuentran dentro de los descriptores de dispositivo. Con el fin de funcionar correctamente estos valores deben figurar y coincidir tanto en el archivo INF como en el firmware, como se ve en la Figura 8 y 9.

```
; ========== Manufacturer/Models sections ===========

[Manufacturer]
%ProviderName% = Tesis,NTx86,NTamd64

[Tesis.NTx86]
%USB\MyDevice.DeviceDesc% =USB_Install, USB\VID_15A2&PID_0050&REV_0100

[Tesis.NTamd64]
%USB\MyDevice.DeviceDesc% =USB_Install, USB\VID_15A2&PID_0050&REV_0100
```

Figura 9. Declaración de los descriptores en el archivo INF

Usb_Descriptor.c

```
#include "Usb_Descriptor.h"

const USB_DEV_DSC Device_Dsc =
{
    sizeof(USB_DEV_DSC),    /* Size of this descriptor in bytes     */
    DSC_DEV,                /* DEVICE descriptor type               */
    0x0002,                 /* USB Spec Release Number in BCD format*/
    0x07,                   /* Class Code                           */
    0x01,                   /* Subclass code                        */
    0x00,                   /* Protocol code                        */
    EP0_BUFF_SIZE,          /* Max packet size for EP0,             */
    0xA215,                 /* Vendor ID    //little-endian for USB */
    0x5000,                 /* Product ID:  //little-endian for USB */
    0x0001,                 /* Device release number in BCD format  */
    0x01,                   /* Manufacturer string index            */
    0x02,                   /* Product string index                 */
    0x00,                   /* Device serial number string index    */
    0x01                    /* Number of possible configurations    */
};
```

Figura 8. Declaración de los descriptores en el Firmware

Los VIDs se utilizan generalmente para distinguir entre los diferentes proveedores o fabricantes de productos. En este caso se utiliza el VID que caracteriza a los productos de Freescale. Su número hexadecimal de VID es 15A2.

Los PIDs normalmente se utilizan para distinguir entre diferentes productos de un mismo fabricante. En este caso, el PID y el REV son a elección del desarrollador y sus valores hexadecimales son 0050 y 0100 respectivamente.

Otros valores importantes que se encuentran en los descriptores de dispositivos, son Class Code, Subclass Code y Protocol Code. Como se observa en la Figura 8, toman los valores 07, 01 y 00 respectivamente. Estos tres valores en conjunto indican que tipo de dispositivo se conecta, en este caso un dispositivo de comunicación de puerto paralelo.

C. DESCRIPTORES DE ENDPOINTS

Cuando se declara un endpoint, se define si es para una transferencia IN (dispositivo a la PC) o OUT (PC al dispositivo), el tipo de transferencia que soporta ya sea por interrupciones, isocrónicas, o a granel (bulk), cuantos datos soporta cada endpoint, etc [6].

En esta aplicación se utiliza solo dos endpoints del tipo bulk, ya que es el tipo de transferencia es más conveniente si se quiere enviar o recibir datos en formato del protocolo LPT. Los endpoints contienen una especie de paquetes de tamaño de 1 byte, por donde se envían los datos o información.

El endpoint 3 es el encargado del transportar los valores del Registro de Datos y de Control. Como en este proyecto el modo LPT a tratar es el modo unidireccional SPP, los valores de estos dos registros se transportan desde la PC al dispositivo LPT, por la tanto se configura al endpoint 3 como del tipo OUT.

Pero en el caso del endpoint 4 que se encarga de transportar los valores del Registro de Estados, el tipo de transferencia es del tipo IN porque en este registro el dispositivo LPT envía su estado a la PC. En la figura 10, se observa la declaración en el firmware de los endpoints a utilizar.

```
/* Endpoint Descriptor*/
/* Size of descriptor
                    Endpoint descriptor identifier
                            In or Out
                                    Endpoint type
                                            Max packet size
                                                                Interval
{sizeof(USB_EP_DSC), DSC_EP, _EP03_OUT, _BULK, (word)(UEP3_SIZE << 8), 0},
{sizeof(USB_EP_DSC), DSC_EP, _EP04_IN , _BULK, (word)(UEP3_SIZE << 8), 0}
```

Figura 10. Declaración del endpoint 3

```
struct
{
  byte bLength;
  byte bDscType;
  byte string[sizeof("Tesis")*2];
} const sd001
  ={
    sizeof(sd001),DSC_STR,
    'T',0,
    'e',0,
    's',0,
    'i',0,
    's',0
  };
```

```
struct
{
  byte bLength;
  byte bDscType;
  byte string[sizeof("LPT Virtual")*2];
} const sd002
  ={
    sizeof(sd002),DSC_STR,
    'L',0,
    'P',0,
    'T',0,
    ' ',0,
    'V',0,
    'i',0,
    'r',0,
    't',0,
    'u',0,
    'a',0,
    'l',0
  };
```

Figura 11. Descriptores de cadenas del proyecto

D. DESCRIPTORES DE CADENAS

Contienen la descripción del dispositivo en formato Unicode (strings) [6]. Está compuesto de tres estructuras de código, en las cuales se envía la información adicional del dispositivo. En este caso, se envía “LPT Virtual” que es la información que lee Windows apenas detecte el dispositivo cuando todavía no se le instalo el archivo INF. En la Figura 11, se observa el fragmento referido en el código del firmware.

IV. DESCRIPCIÓN DEL SISTEMA INTEGRADO

Para corroborar la funcionalidad del sistema emulador de puerto LPT, se utilizó una placa paralela que emula el funcionamiento del dispositivo LPT SPP a conectarse. En la figura siguiente se observa el diagrama en bloques del sistema para estas pruebas.

A través de la GUI se escribe y se lee el puerto LPT de la PC y sus datos son enviados hacia el firmware del microcontrolador a través de la conexión USB entre la PC y la interfaz DEMOJM. Este firmware se encarga de recibir losdatos y distribuir cada bit de información en los puertos configurados como salidas del microcontrolador. Los datos llegan al conector DB25 de la placa paralela y se reproducen sus valores en los leds que representan los tres registros del puerto paralelo.

B. Firmware

A. Conexión con el conector LPT

En este caso, se utilizan 8 pines del puerto PTD del microcontrolador para enviar los 8 bits del Registro de Datos, 5 pines del puerto PTE por el cual se envían los 5 bits más altos del Registro de Estados, y los 4 pines más bajos del Registro de Control se representan a través del puerto PTF.

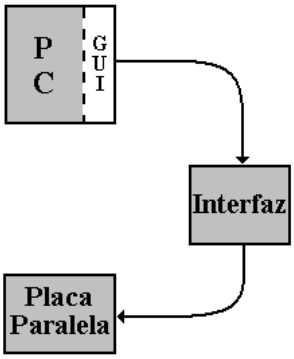

Figura 12. Diagrama en bloques

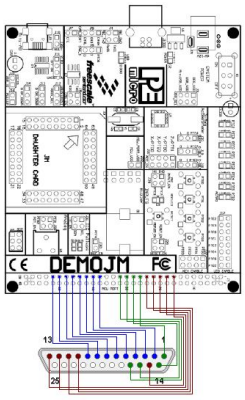

Figura 13. Conexión de los pines

En la figura 13, se observa la conexión entre los pines utilizados del microcontrolador y los pines del conector paralelo DB25. Los pines del conector DB25 del 2 al 8 son los correspondientes al Registro de Datos, del 10 al 13 y el 15 son los pines referidos al Registro de Estado y los pines 1, 14, 15 y 16 son los del Registro de Control. Además, los pines libres en el conector DB25 son líneas de masa o tierra (GND).

Los cambios necesarios para la aplicación, son básicamente los que se realizaron en el archivo principal (main.c), donde se habilitan y configuran los puertos a utilizar como salida, este tipo de configuración se realiza en el archivo de entradas y salidas de propósito general (GPIO.c). Otro archivo importante es el USB_User_API, este archivo tiene la funcionalidad de encargarse de toda la configuración referida a los endpoints que utiliza el firmware para comunicarse con la aplicación GUI.

V. RESULTADOS OBTENIDOS

A. Placa de Pruebas

Las pruebas se realizaron con una placa o entrenador que emula el funcionamiento de un dispositivo paralelo. Esta placa es parte de una serie de dispositivos que la Editorial Cekit editó como Cursos de Electrónica Aplicada. El cual contiene todos los elementos necesarios para que el usuario pueda armar la placa por su propia cuenta.La placa cuenta con la posibilidad de ser alimentada a través de una fuente incorporada a ella, pero en este caso se alimenta a través del pin 1 (Vdd) de la DEMOJM.

B. Pruebas del Registro de Datos

Existen dos formas de escribir el Registro de Datos del puerto LPT a través de la GUI. La primera es mediante la opción de modificar bit a bit el byte de dicho registro y la segunda es escribiendo el valor decimal que el usuario desee enviar como dato.

Básicamente, las pruebas realizadas consisten en testear los dos modos de escritura del Registro de Datos que tiene la GUI, y observar si los valores enviados son realmente los que llegan a la placa. Para

ello se utiliza otro programa llamado Monitor LPT. En la figura siguiente se realiza el envío del dato 169 desde el módulo que permite manipular bit por bit el registro. Los resultados son los mismos que se ven en el Monitor LPT.

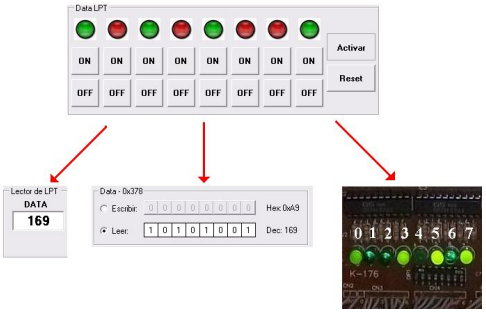

Figura 15. Envío de datos bit por bit al Registro de Datos

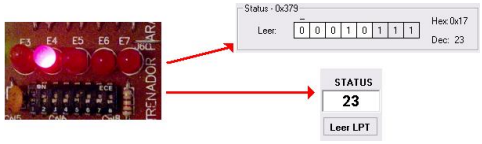

Figura 16. Prueba con el Registro de Estados

C. Pruebas del Registro de Estados

Se define al Registro de Estados como el registro encargado de ser el indicador del estado actual de los dispositivos LPT, tales como el estado ocupado, errores de la interfaz o de periférico. Este registro es solamente de lectura, por lo tanto el encargado de manipularlo es el mismo dispositivo LPT conectado.

La placa paralela tiene la opción de mediante un interruptor simular el valor que el dispositivo enviaría a la PC como dato de estado. En este caso se envía el valor que el dispositivo envía para indicar que todavía no está preparado para recibir datos, el valor BUSY se representa por el seteo del bit 4 del Registro de Estados en modo SPP. En la figura siguiente se observan los leds de la placa paralelo junto al interruptor DIP utilizado para simular el valor de BUSY y además se observan los resultados correctos obtenidos tanto en la GUI como en el programa Monitor LPT.

D. Pruebas Del Regisrto De Control

El registro de control es utilizado como control de la interfaz y señalización de establecimiento de comunicación (handshaking) de la PC al dispositivo.

En esta prueba, se setea el bit cero del registro de control con el fin de avisarle al dispositivo LPT que hay datos válidos en la línea de datos para serle enviados desde la PC. Esta función se la llama STROBE.

El seteo del bit se realiza con el programa Monitor LPT para constatar que también puede utilizarse otro programa en simultáneo a la GUI, siendo esta ultima la encargada de comunicarle al dispositivo los cambios realizados en el puerto físico paralelo a través del Monitor LPT.

En la figura siguiente se observa la escritura en el Monitor LPT y las predecibles lecturas en los Leds de la placa paralela y en el cuadro de lectura del registro de control de la GUI.

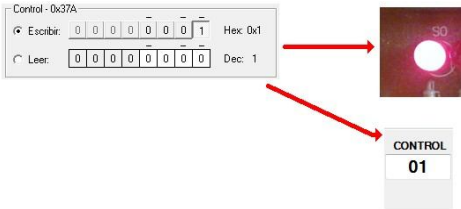

Figura 17. Prueba con el Registro de Contro

En base a estas pruebas realizadas se puede afirmar que los resultados obtenidos fueron más que satisfactorios, en cuanto a la implementación del protocolo LPT mediante el puerto USB. Pues, se demuestra que mediante los drivers diseñados y el firmware implementado en el Microcontrolador se puede implementar un puerto LPT virtual, en donde mediante el envío de datos se configuran los registros de dicho protocolo y se verifican los cambios mediante los programas mencionados.

VI. CONCLUSIONES

Se diseñó un programa de aplicación básico que utiliza el puerto paralelo virtual para el envío de datos. Dejando sentadas las bases para el diseño de cualquier programa que requiera una interfaz de datos con un dispositivo cualquiera que contenga un puerto paralelo. Mediante un conjunto de drivers genéricos se estableció una conexión entre las direcciones de los registros del puerto paralelo asignados por Windows con un puerto USB. Logrando que al conectar un dispositivo en el puerto USB, el mismo se enumere como un LPT virtual.

Por otro lado se configuró un sistema integrado básicamente por un JM60, mediante el

cual se reciben los datos por el puerto USB y se mapean al puerto paralelo de manera correcta. Dicho funcionamiento se plasmó en la placa de pruebas del puerto paralelo Cekit k-176 que permite visualizar el estado de los registros del puerto paralelo mediante leds.

De todos los modos de comunicación del LPT se implementó el unidireccional SPP. Aunque la idea principal fue tratar todos los modos de comunicación, se optó por solo tratar el modo SPP debido a la complejidad y extensión del código que hubiera sido necesario para tratar todos los módulos de comunicación del LPT.

VII. TRABAJO A FUTURO

La implementación del modo unidireccional SPP deja las bases necesarias para la implementación de los demás modos. Como diferencia principal, en los modos EPP y ECP el Registro de Datos es bidireccional. Por lo tanto deberá cambiarse tanto el firmware como la GUI con el fin de poder tener la posibilidad de recibir datos enviados desde el dispositivo paralelo. En este caso el Registro de Estados lo escribe el dispositivo LPT, por lo tanto el proceso de transferencia de datos de este registro deja el conocimiento para hacer las modificaciones necesarias en el Registro de Datos.

Se realizó la implementación en hardware mediante un kit de desarrollo que contiene el JM60 y algunos periféricos. Quedaría como trabajo a futuro el diseño de una placa que contenga el microcontrolador con el hardware necesario tanto para la comunicación USB, como para la comunicación con dispositivos con puerto LPT.

La utilización de los archivos DLL seleccionados nos permiten utilizar la GUI creada para este proyecto solamente en sistemas operativos de la firma de Windows. La tarea a futuro sería crear los propios archivos DLL o buscar nuevos DLL, e implementarlos con el fin que permitan utilizar la GUI en otros Sistemas Operativos.

REFERENCIAS

[1] “WinUSB Device”, msdn.microsoft.com, 2011.
[2] “SimpleUSB”, David Arnold, 2003.
[3] “Introducción a Visual Studio 2010”, msdn.microsoft.com, 2010.
[4] “Inpout32”, Logix4U.net, 2000.
[5] “Archivos DLL”, msdn.microsoft.com, 2011.
[6] Jan Axelson, USB Complete. Madison, Lakeview research,1999-2005.[7] Jan Axelson, USB Mass Storage. Madison, Lakeview research, 2006.