Sistema de co-procesamiento para él envió de datos desde un computador hacia FPGA Spartan 3E para ejecutar el movimiento de dos servo motores.

System of co-prosecution for sent data from computer to FPGA Spartan 3E, to execute the movement of two servo motors.

Sergio Flórez Galeano, Cristian Steven Chavarro

*Ingeniería de Sistemas y Computación, Universidad Tecnológica de Pereira, Pereira, Colombia*

serfloga@gmail.com

*Resumen*— **El paso de datos por medio de puerto serial fue muy útil para la conexión de dispositivos externos a una terminal (CPU), aunque era un poco lento para su transferencia era la mejor que existía en su tiempo. Para el uso de estos se necesita un controlador para el dispositivo y también un software el cual ayudara a interactuar entre el dispositivo y la terminal, como mandar peticiones o recibir de este. La implementación presentada en este documento será la interfaz a utilizar para mandar información a dos servomotores conectados a través de una UART y dos memorias RAM implementadas sobre una FPGA Spartan 3E.**

***Palabras clave — Com,* FPGA, Giovynet, Hardware, RAM, ROM . RS232, Serial, Software, UART.**

*Abstract*— Passing data through serial port was very useful for connecting external devices to a terminal (CPU), though a bit slow for transfer was the best that existed in his time. To use these you need a driver for the device and software which will help to interact with the device and the terminal, how to send or receive requests for this. The implementation presented here will be the interface used to send information to two actuators connected via a UART and two RAM memories implemented on a Spartan 3E FPGA.

*Key Word* — ***Com,* FPGA, Giovynet, Hardware, RAM, ROM . RS232, Serial, Software, UART.**

### INTRODUCCIÓN

El paso de datos por medio de puerto serial fue muy útil para la conexión de dispositivos externos a una terminal (CPU), aunque era un poco lento para su transferencia era la mejor que existía en su tiempo. Para el desarrollo de un sistema de co-procesamiento de datos por puerto serial, para envió de información a dos servomotores, se llevara a cabo un diseño por etapas de los diferentes componentes involucrados con su respectiva implementación en una FPGA Spartan 3E de Xilinx.

Se realizaran en total cinco etapas de diseño e implementación para afianzar los conocimientos de cada componente.

1. Diseño e Implementación UART para Comunicación con puerto serial RS-232. Usando Hyperterminal de Windows como sistema de transferencia de datos.
2. Desarrollo Software para envió y recepción de datos por puerto Serial usando el Hardware del UART.
3. Almacenamiento de datos en memorias RAM, y verificación de datos mediante display de siete segmentos de la FPGA.
4. Creación del Hardware del PWM.
5. Integración del PWM al resto del hardware para el movimiento de los servomotores a través de la información suministrada por las memorias RAM.

### CONTENIDO

**1. Diseño e implementación de una de UART**

Un transmisor/receptor serie asíncrono (Universal Asynchronous Receiver-Transmitter) que siga la norma RS232. Es un módulo que permitirá comunicar la placa con el puerto serie del PC. Aunque en la norma se definen más señales, sólo son imprescindibles tres señales: la línea de transmisión de datos (TxD), la de recepción (RxD), y la línea de masa (GND).

Existen distintas velocidades de transmisión, que se definen en bits por segundo (bps) o baudios (921600, 460800, 230400, 115200, 57600, 38400, 19200, 9600, 4800, …).

La línea serie permanece a nivel alto ('1') mientras no se envían datos. Cuando el transmisor va a empezar la transmisión, lo hace enviando un bit de inicio que está a '0'. Posteriormente se envían consecutivamente los bits del dato empezando por el menos significativo. Después del último bit de dato se envía el bit de paridad en caso de que se haya especificado. Por último, se cierra la trama con uno o dos bits de fin con valor '1'. En la figura 5-4 se muestra el cronograma de un envío RS232 con 8 datos, un bit de paridad (par) y un bit de fin.

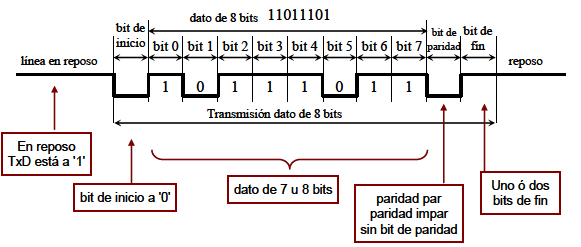


Figura 1: Trama de un envio en Rs232 con 8 bits

**Procedimiento**

El dato de entrada será de 8 bits, con un total de 10 bits con bit de inicio y bit de parada. No se tendrá en cuenta un bit de paridad.

Lo primero que se tuvo en cuenta fue como se iba a implementar los dos métodos de transferencia de RX y TX, para eso se crearon dos maquinas de estado. En la TX cada vez que ocurre un cambio en tx\_ready a 1 el empieza a pasar el dato\_i de entrada bit por bit. Cuando lo transfiere tendrá siempre salida 1 el cual lo mantiene a la UART en estado IDLE.

En la RX su trabajo es invertido; el empieza a recibir el dato rx, durante este tiempo en que no se envía nada se encuentra en estado IDLE; cuando se empieza a enviar empieza a recibir cada bit del dato y lo empieza a unir. Al final cuando halla unido todo el dato en uno solo envía una señal de ’1’ en el rx\_ready y empieza a recibir el dato.

Después se creó el divisor de frecuencia el cual trabaja con un periodo de 434 ciclos del reloj de la FPGA. Este valor se da como resultado, debido a que se va a transmitir a una velocidad de 115200 Baudios.

Velocidad de Transmisión



Frecuencia de la FPGA 1 ciclo de reloj

f = 50Mhz

Periodo de la FPGA

 20ns

Queremos saber la velocidad de transferencia de 1 bit. A 115200 baudios.



115200bits 1s

1bit x

 Tiempo para transmitir 1 bit

Sabiendo el tiempo de transmisión de 1 bit para 115200baudios queremos saber cuantos ciclos de reloj son necesarios para transmitir ese bit a 20ns el cual es el tiempo que dura un ciclo en la FPGA.

 1ciclo de reloj

 x



Por lo tanto debemos adaptar nuestro divisor a un periodo de 434 ciclos de reloj.

Podemos observar que hay un desfase de 0.02… , por lo tanto cada cierto tiempo ocurrirá un error en la transmisión y se imprimirá basura. Debido a que este desfase se va acumulando a lo largo del tiempo.



Esto quiere decir que cada 15625 ciclos de reloj se acumulara el desfase y ocurrirá un error de transmisión.

En base a lo anterior, la UART tendrá tres componentes como se muestra en la figura 2:

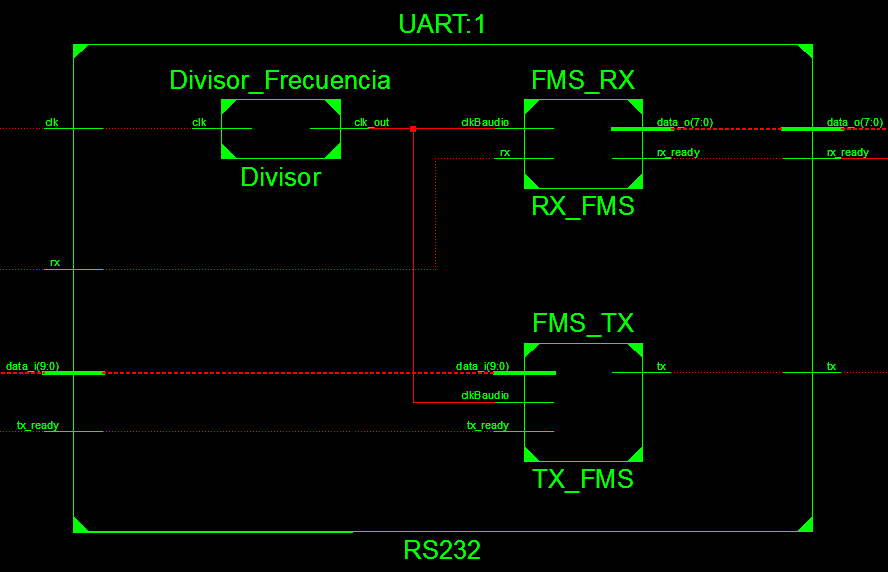


Figura 2: UART (Universal Asynchronous Receiver-Transmitter)

* Divisor de Frecuencia: Divisor con periodo igual a 434 ciclos de reloj de la FPGA.
* Maquina de estados RX: Receptor datos por el puerto serial.
* Maquina de estados TX: Transmisor de datos por puerto serial

**2. Desarrollo Software para envió y recepción de datos por puerto Serial usando el Hardware del UART.**

Para desarrollar un software que cumpla con los protocolos RS232, entre PC y UART. El lenguaje de programación utilizado es JAVA con ayuda de la librería Gyovinet para manejar puertos Seriales RS323.

Giovynet, es una librería creada para hacer observación, medición y control de sistemas electrónicos usando Tecnologías de Java. De esta librería se usaron los paquetes de ***SerialPort, Parameters, Com.***

* ***SerialPort:*** Esta clase permite crear una conexión a el puerto serial para el envió de datos.
* ***Parameters:*** Esta clase determina los parámetros que se vana usar en la comunicación serial.
* ***Com:*** Esta clase representa el puerto serial, al cual se le asignan los parámetros deseados.

Esta librería se escogió por su fácil manejo y el rápido entendimiento de su de las clases.

Con el fin de probar su funcionamiento se añadió al diseño de hardware, una conexion directa entre los datos recibidos por la maquina RX, y la entrada de la maquina TX, con la finalidad de crear un eco entre la maquina y el hardware, y se agrego un registro sujeto a cambios en la recepción para mostar en los leds los datos recibidos, como se muestra en la figura 3.

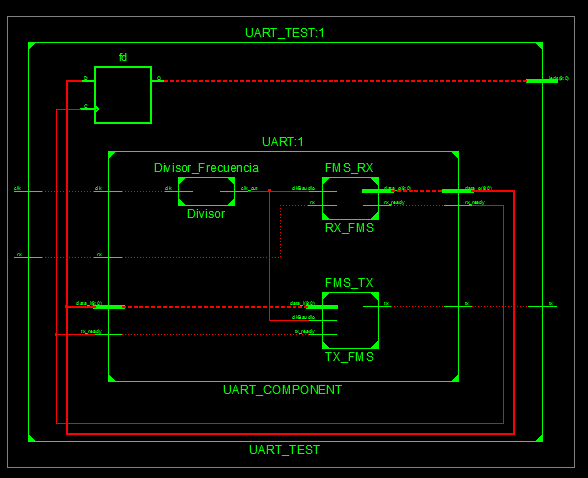


Figura 3: Diseño de UART con eco y registro leds.

Archivo Java.

<Serial.java>

**3. Almacenamiento de datos en memorias RAM, y verificación de datos mediante display de siete segmentos de la FPGA.**

Se describirán 2 memorias RAM las cuales almacenaran 256 datos cada una, enviados desde el software (hyperterminal) implementado en la etapa anterior. Cada memoria contiene datos los cuales va a ser usados para generar un movimiento en dos servomotores. A cada servomotor se le asigno una memoria.

El objetivo de esta etapa es recibir datos enviados desde el software, almacenarlos en las memorias y a través de los 7-segmentos mostrar el valor de cada posición en estas.

Se enviaran 512 datos aleatorios de 8 bits, cada dato se almacenara secuencialmente en las memorias y de forma circular. Las memorias tendrán dos modos: escritura y lectura, lo cual estará controlado por un Switch en la tarjeta de desarrollo. La modalidad funciona de la siguiente manera: Cuando este en modo escritura (0 lógico) los datos que envíen se almacenaran automáticamente y se mostrara una línea de ceros en los 7-segmentos, cuando este en modo lectura (1 lógico) empezara a obtener datos desde la posición 0 en la primera y segunda memoria. En modo lectura, los datos se van a mostrar de forma secuencial y automática cada 1 ms de 0 a 255 en las dos memorias.

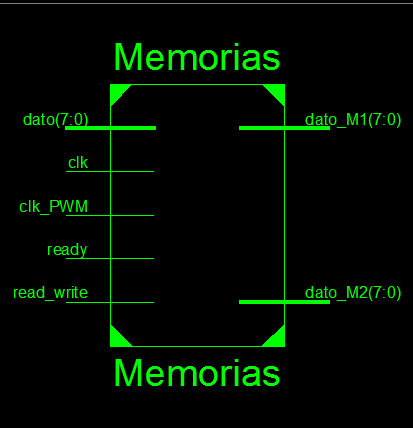
El diseño consistirá en 2 memorias RAM tal y como se muestran en la figura 5.

Figura 4: Descripción de modulo memorias, con sus entradas y salidas.

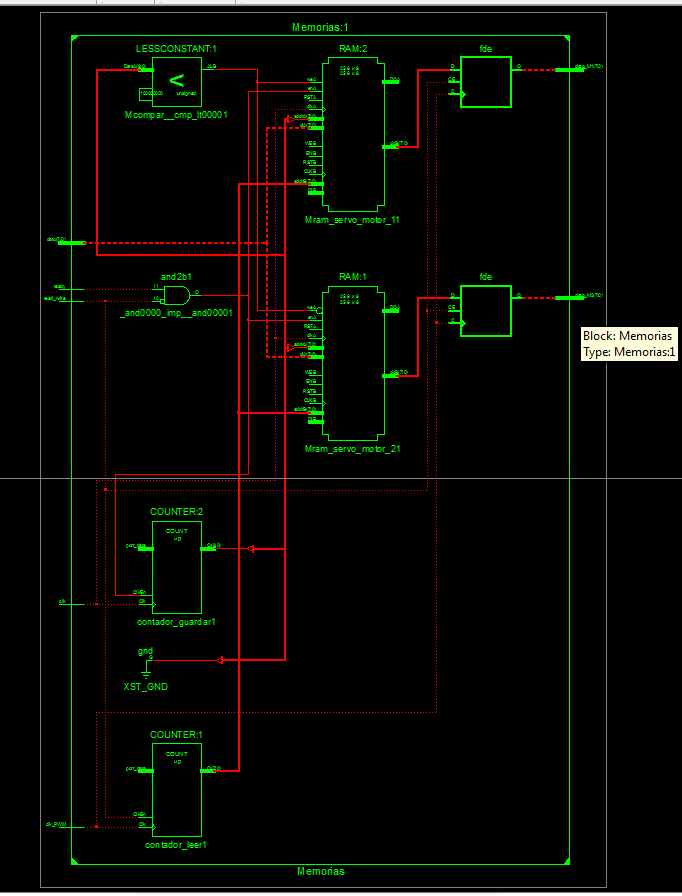
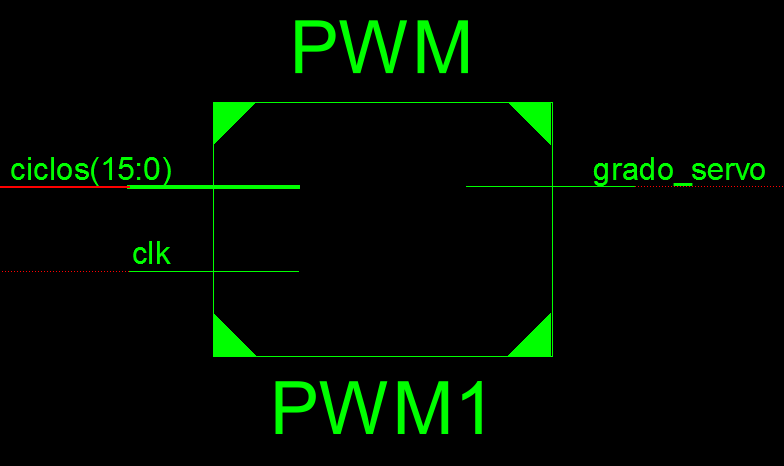
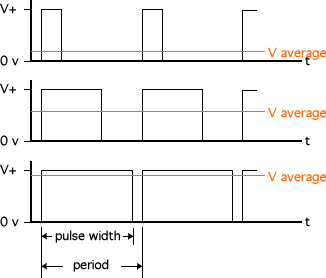


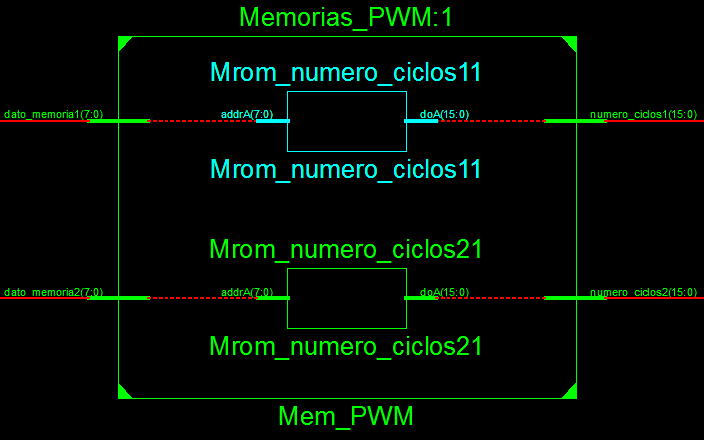
Figura 5: Descripción de la figura 4, mostrando Memorias RAM

**4. Creación del Hardware del PWM.**

La implementación del PWM para el manejo de los servomotores, no es más que el envió de una señal de periodo constante y ancho de pulso variable.

Los servomotores comerciales más comunes giran en ángulos de 0 a 180 grados y su señal de movimiento proviene del ancho de pulso de la señal, con un periodo mínimo de trabajo de los servomotores de T=20ms, y con un ancho de pulso inicial de 1ms, necesario para energizar el movimiento del servo.

Figura 6: Descripción grafica del PWM

Para él envió de datos al servo es necesario de unos anchos de pulso específicos para representar el ángulo de giro. Los servomotores necesitan el primer milisegundo para energizarse y el ancho desde este primer (ms) hasta el segundo (ms) representara el ángulo de giro. Por ello este ancho de 1ms a 2ms se dividirá en 255 partes y cada uno. Esto se describirá a continuación.

**Periodo de la FPGA:** T=50MHZ

**Periodo del servo:** T=20ms

**1 ciclo del servo 🡸🡺 1.000.000 ciclos de la FPGA**

Esto quiere decir que cada ángulo enviado al servo será la suma entre el primer milisegundo de energización mas 0,00390625ms multiplicado por el ángulo a modular.

1ms 🡸===========🡺 5000ciclos de la FPGA

0,00390625 🡸=====🡺 195 ciclos FPGA

A manera de ejemplo, si se desea enviar un ángulo de 135 grados de deberán enviar

Se diseñaran dos módulos para el uso del PWM. Un modulo para la modulación por ancho de pulso variable según una señal de entrada que especificara la cantidad de ciclos que durara activa la señal. Y un modulo con una memoria ROM con 256 datos que especificaran la cantidad de ciclos correspondientes a cada ángulo. El diseño se especificara como se muestra en la figura 7 y 8.

Figura 7: Modulo PWM

Figura 8: Memorias ROM PWM

**5. Integración del PWM al resto del hardware para el movimiento de los servomotores a través de la información suministrada por las memorias RAM.**

Finalmente la integración de todos los componentes del hardware necesarios para el sistema de co-procesamiento de

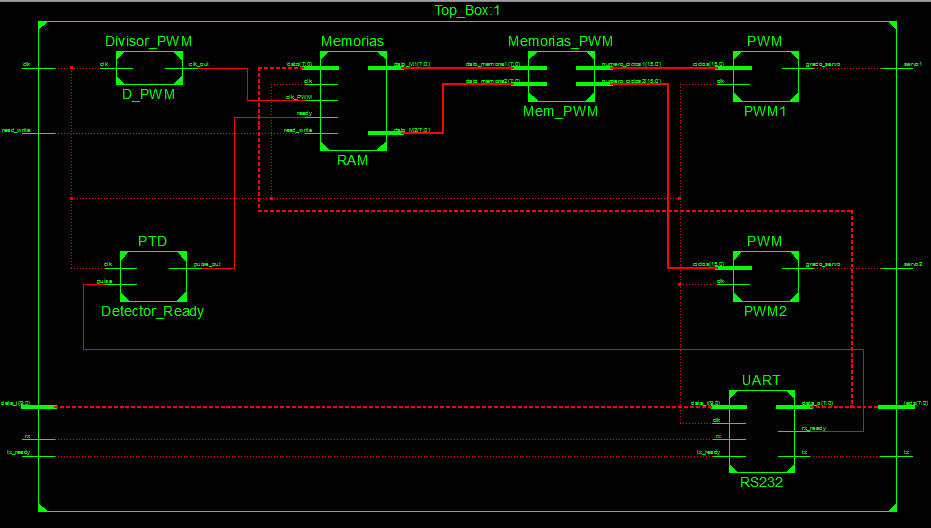


Figura 9: Sistema de co-procesamiento de datos por puerto serial.

datos por puerto serial, será diseñado tal y como se muestra en la figura 9.

1. CONCLUSIONES

1. Por medio de la implementación una UART podemos crear un puente de transferencia entre dos terminales y así intercambiar información entre estos.

2. La transferencia de forma serial es una forma muy útil y eficiente para transmitir datos por sus bajo porcentaje de error en la transferencia, pero ya bastante lenta para los sistemas de información de esta época.

3. El uso del Puerto COM3 no es útil para realizar pruebas reales ya que este es un puerto virtual que creado por el puerto serial real.

4. El manejo de memorias RAM para el almacenamiento de los datos nos permitirá tener una organización de envío del dato a los servomotores.

### 5. Con el uso de un Modulador de ancho de pulsos podemos controlar cargas de voltaje en dispositivo eléctrico.

### 6. Para generar movimientos en el servomotor se necesita un tiempo de para cargar el servomotor y después el ciclo de trabajo.

### VI. REFERENCIAS

Ingeniería en Micro-controladores

Tutorial del Protocolo RS-232

<<cselectrobomba.googlecode.com/files/Serial_RS232.pdf>>

Manejador RS232- Universidad de Las Palmas de Gran Canaria

<<sopa.dis.ulpgc.es/ii-dso/lecminix/manejado/rs232/rs232.pdf>>

RS-232 TRAINING

<<http://www.doranscales.com/connectivity-bulletins/all-about-RS232.pdf>>

Giovanynet.com – Java Electronic Solutions

<<http://www.giovynet.com/>>

LAS PATAS DEL CHIP. Héctor Eduardo Ugarte Rojas <<http://es.scribd.com/doc/49480537/45559031-ENVIO-Y-RECEPCION-DE-DATOS-A-TRAVES-DEL-PUERTO-SERIAL-DE-LA-PC-APLICACION-CONEXION-ENTRE-2-PCS-Y-PRUEBAS-HACIENDO-USO-DE-JAVA>>

Giovanynet Artículos. Manejo de Puertos Seriales RS-232 con Java para Windows

<<https://sites.google.com/site/articulosgiovynet/manejo-de-puertos-seriales-rs-232-con-java--para-windows>>

## Servomotores – Centro de Formación del Profesorado e Innovación Educativa de Valladolid

<<http://cfievalladolid2.net/tecno/cyr_01/robotica/sistema/motores_servo.htm>>