# Analizador de Protocolo RS-232

Steven Caicedo [1], Jeanneth Paredes [2], Ronald Ponguillo [3]
Facultad de Ingeniería en Electricidad y Computación [1] [2] [3]
Escuela Superior Politécnica del Litoral (ESPOL)[1] [2] [3]
Campus Gustavo Galindo, Km 30.5 Vía Perimetral
Apartado 09-01-5863. Guayaquil, Ecuador [1] [2] [3]
skcaiced@espol.edu.ec [1], jeamipar@espol.edu.ec [2], rponguil@espol.edu.ec [3]

## Resumen

*En el presente trabajo se desarrolló una herramienta que servirá para capturar y analizar las tramas que se intercambian entre dos dispositivos que se comunican usando el protocolo de comunicación serial RS-232. La información capturada se muestra en una Computadora Personal o en un Dispositivo Móvil con Sistema Operativo Android; esto nos permitirá entender e interpretar el funcionamiento del protocolo antes mencionado, además nos ayudará en la tarea de detectar y resolver problemas relacionados con este tipo de comunicación.*

*Esta herramienta tiene la capacidad de detectar los parámetros de configuración de la comunicación RS-232 que analizaremos para posteriormente auto configurarse a ellos y así recolectar las tramas de forma correcta.*

*Para esto se desarrolló un Sistema Embebido basado en el Microprocesador NIOS II el cual fue implementado en la tarjeta de desarrollo y educación DE0 nano de Terasic Inc., cuyo componte principal es la FPGA EP4CE22F17C6N de la familia Cyclone IV E de Altera.*

**Palabras claves:** *Analizador de Protocolo, RS-232, FPGA, NIOS II.*

## Abstract

In this paper we were developed a tool that will be used to collect and analyze the frames that are being exchanged between two devices that communicate using the RS-232 serial communication protocol. The information collected is displayed in a Personal Computer or a Mobile Device with Android OS; this will allow us understand and interpret the operation of the RS-232 protocol, also help us in the task of detecting and solving problems related to this type of communication.

This tool has the ability to detect the parameters of configuration of the RS-232 communication that we want to analyse, auto configured with this parameters to collect the frames correctly.

Then, we developed an embedded system based on NIOS II Processor; this was implemented in the Board of development and education DE0 nano of Terasic Inc, whose main component is the FPGA EP4CE22F17C6N of the family Cyclone IV E of Altera.

**Keywords:** *Protocol Analyzer, RS-232, FPGA, NIOS II.*

## 1. Introducción

Desde la antigüedad los hombres han tenido la necesidad de interactuar e intercambiar información entre sí, a este proceso se le llama comunicación en la cual es muy necesaria la correcta interpretación de la información por parte del receptor del mensaje emitido.

Con la aparición de los dispositivos electrónicos surgió también la necesidad de comunicarse entre ellos, para esto los expertos desarrollaron distintos métodos de comunicación, entre las cuales está el protocolo RS232.

En una comunicación electrónica es muy importante que los dispositivos apliquen de forma correcta las reglas definidas por los protocolos. He aquí la importancia de los analizadores de protocolos.

Estas herramientas nos permiten analizar la comunicación, detectar fallos y verificar el uso inequívoco del protocolo.

Para cubrir esta necesidad se presenta el analizador de protocolo RS-232 desarrollado en una computadora de propósito específico embebida en una FPGA con un teclado 4x4 como interfaz de entrada y un PC y un dispositivo móvil para la visualización de la información analizada.

## 2. Objetivos

### 2.1 Objetivo General

Aplicar conceptos adquiridos en el área de sistemas embebidos, comunicación serial y su implantación en una FPGA, instruirnos en el manejo de un LCD 16x2 y un teclado numérico hexadecimal usando la tarjeta de desarrollo y educación DE0 nano, además de experimentar con la comunicación entre dispositivos que usan diversas tecnologías

### 2.1 Objetivo Específico

El objetivo principal con este proyecto es diseñar un analizador de comunicación serial RS-232 desarrollando un Sistema Embebido basado en el Microprocesador NIOSII.

Además, nos enfocaremos en los siguientes puntos:

- Conocer y entender el funcionamiento de la tarjeta DE0 nano, sus características principales y módulos incorporados.
- Conocer el protocolo de comunicación serial RS-232.
- Hacer uso de los programas Quartus II para el diseño de hardware de nuestro sistema y NIOS II IDE para Eclipse para el desarrollo del software que controlará dicho hardware.
- Usar módulos UART-USB y UART-Bluetooth, ambos externos a la tarjeta DE0 nano para la comunicación con una computadora y un dispositivo Android.
- Hacer uso del programa NetBeans IDE para el desarrollo de las aplicaciones para el usuario final en la computadora y en el dispositivo Android.

## 3. Alcance y Limitaciones

El proyecto cumple con las siguientes características:

Se realizará un analizador de protocolo RS-232 cuyos parámetros de comunicación sean configurables.

- La configuración de nuestro analizador puede ser automática como manual.
- El análisis se concentrará exclusivamente en los pines de trasmisión y recepción de datos y se lo realizará a niveles TTL.
- La trama RS-232 se la analizará después de haberla recibido completamente.
- La visualización del estado del analizador se mostrara en un LCD 16x2.
- La visualización de los datos se lo podrá realizar en una computadora y en un teléfono celular, tableta o cualquier dispositivo con sistema operativo Android.
- La aplicación para el usuario final en la computadora se la desarrollara en el lenguaje de programación java.
- Los datos llegarán al dispositivo Android a través de una comunicación inalámbrica usando el protocolo Bluetooth.
- Los datos llegarán a la computadora a través de una comunicación alámbrica usando una interfaz UART-USB.
- Las pruebas se las harán solo en ambiente de laboratorio.

## 4. Fundamento Teórico

### 4.1 Protocolo RS-232

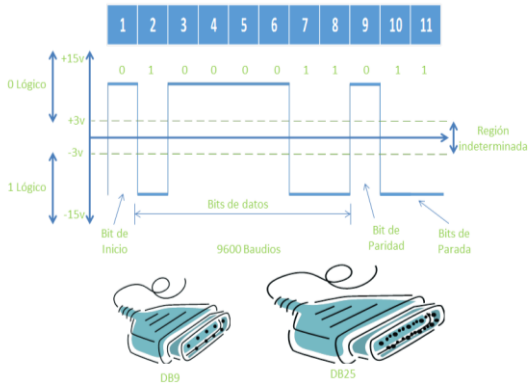



**Figura 1.** Especificaciones del protocolo RS-232

El protocolo RS-232 es una norma de comunicación serial que define especificaciones mecánicas, eléctricas, regula la transmisión de datos, el cableado, las señales eléctricas y los conectores en los que debe basarse. [1]

Especificaciones:

- Mecánicas: El RS-232 consiste en un conector tipo DB-25 (de 25 pines), se lo puede encontrar también en la versión de 9 pines (DB-9). El estándar define que el conector hembra se situará en los DCE y el macho en el DTE. [1]
- Eléctricas: Los estados lógicos son definidos por los siguientes niveles de voltaje: 1 lógico entre -3V y -15V, 0 lógico entre +3V y +15V. La interfaz RS-232 está diseñada para distancias cortas, de hasta 15 metros y para velocidades de comunicación bajas, de no más de 20 Kb/s.[1]
- Lógicas: La trama comienza con un bit de inicio, luego se envía el dato desde el bit menos significativo hasta el mas significativo, si se ha configurado paridad se añadirá el bit de paridad dependiendo del tipo de paridad (par o impar) y finalmente se añadirá el o los bits de parada según se haya configurado La duración de cada bit está dado por la velocidad de trasmisión definida para la comunicación.

## 4.2 Módulo NIOS II UART

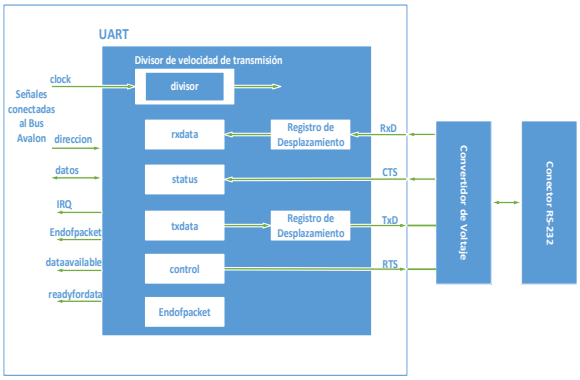



**Figura 2.** Diagrama de bloques del Módulo UART.

El módulo NIOS II UART es un componente de la librería SOPC Builder de Altera que implementa al protocolo RS-232 asíncrono. El módulo UART envía y recibe datos de forma serial a través de dos pines externos (RxD y TxD). Para el control por Software del módulo UART se usan 5 registros de 16 bits (ver figura 2).

Para cumplir con los voltajes especificados en la norma RS-232 se necesita que los pines RxD y TxD estén conectados a un convertidor de voltaje y este a su vez al correspondiente conector DB9 o DB25 especificado en la Norma. El rango de voltaje aceptable de los pines de entrada y salida del módulo depende sobre cómo se configuren los pines de E/S en el dispositivo de Altera. El módulo UART trabaja con una entrada de reloj síncrona, clk. [2]

## 4.3 Bluetooth

Bluetooth es un protocolo de comunicación para redes inalámbricas de área personal (WPAN) que hace posible la trasmisión de voz y datos entre diferentes dispositivos mediante un enlace de radiofrecuencia en la banda de los 2.4 GHZ.

## 4.4 Módulo Bluetooth HC-05

Es un dispositivo electrónico (ver figura 3) que incorporado a un sistema le añade la capacidad de poder establecer enlaces de comunicación Bluetooth. La comunicación entre el Módulo y el Sistema puede ser RS-232, SPI, USB según se configure. [3]

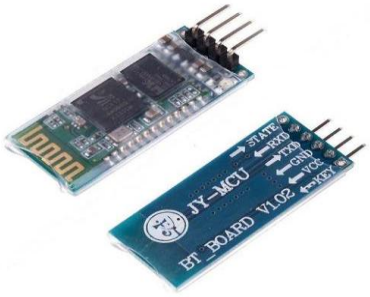



**Figura 3.** Módulo Bluetooth HC-05.

## 4.5 Módulo UART-USB I&T 02

Es un módulo de comunicación serial por puerto USB para hacer interface con un computador o PC, permite alimentar circuitos de 5V.

**Figura 4.** Módulo UART-USB I&T 02.[4]

## 4.6 LCD 16x2

El LCD (Liquid Crystal Display) es básicamente un dispositivo que sirve para la visualización de datos en un sistema electrónico (ver figura 4). Está compuesto básicamente por una pantalla de cristal líquido y un circuito microcontrolador que regula los parámetros necesarios para la presentación de caracteres ASCII, Kan ji y griego. El LCD 16x2 dispone de 2 filas de 16 caracteres cada una y cada carácter dispone de una matriz de 5x7 pixeles. Puede almacenar 40 caracteres por línea de pantalla, además puede ser controlado por un procesador usando un interfaz de 4 u 8 bits.

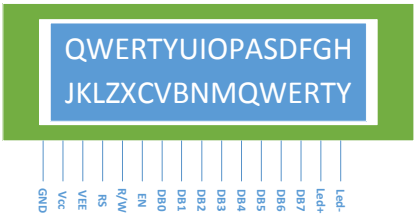



**Figura 5.** LCD 16x2.

## 4.7 Teclado Matricial

Es un dispositivo de entrada de datos que consta de 16 teclas o pulsadores, dispuestos e interconectados en filas y columnas. En la siguiente figura vemos el esquema de conexionado interno del teclado matricial y sus correspondientes pines de salida. Cuando se presiona un pulsador se conecta una fila con una columna, teniendo en cuenta este hecho es muy fácil averiguar que tecla fue pulsada.

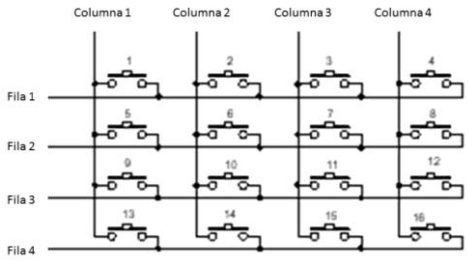



**Figura 6.** Teclado Matricial 4x4.

### 4.8 Tarjeta DE0 nano

La tarjeta DE0-nano es una plataforma de desarrollo basado en FPGA adecuado para una amplia variedad de proyectos de diseños portables, como robots y proyectos móviles. [5]

La DE0-nano es ideal para diseño con procesadores embebidos, además posee incorporada una herramienta para la programación de la FPGA llamada USB Blaster. La tarjeta puede ser alimentada eléctricamente mediante este puerto USB o mediante una fuente de alimentación externa.

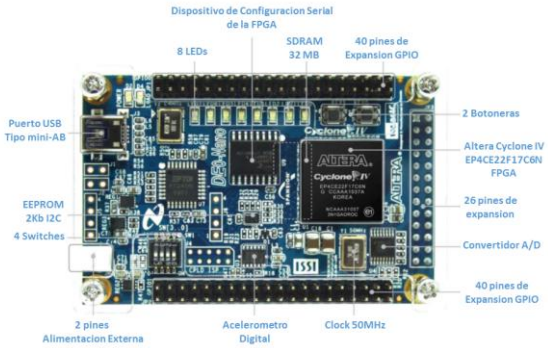



**Figura 7.** Vista superior de Tarjeta DE0 nano.

### 4.9 Java

Java es un lenguaje compilado e interpretado. Todo programa en Java ha de compilarse y el código que se genera bytecodes es interpretado por la máquina virtual de Java (JVM). De este modo se consigue la independencia de la plataforma, el código compilado se ejecuta en la JMV que si es dependiente de la plataforma.

### 4.10 Android

Android es un sistema Operativo desarrollado por Google basado en Linux para teléfonos móviles, tabletas, portátiles e incluso PCs. Android permite programar en Java para el desarrollo de sus aplicaciones.

## 5. Diseño e Implementación

En esta sección se presenta el diseño de nuestro analizador de protocolos así como la implementación del mismo. Este fue realizado desarrollando un sistema embebido, el cual consta de componentes de hardware y software que administra dicho hardware.

### 5.1 Diseño de Hardware Físico

La figura 8 muestra los componentes de hardware usados en este proyecto, así como sus respectivas conexiones. En el rectángulo central tenemos a la tarjeta DE0 nano, la cual tiene la FPGA EP4CE22F17C6N en donde se implementó el sistema embebido NIOS II.

Tenemos un LCD 16x2 que sirve para la visualización del estado del analizador, un teclado matricial de 4x4 para configurar el analizador y elegir opciones.

Además un módulo UART-USB para la comunicación con el PC que muestra los datos recolectados para el análisis, un módulo Bluetooth para la comunicación con el celular que será otra alternativa para la visualización de los datos recolectados.

En la figura 8 los dispositivos 1y 2 son los analizados.

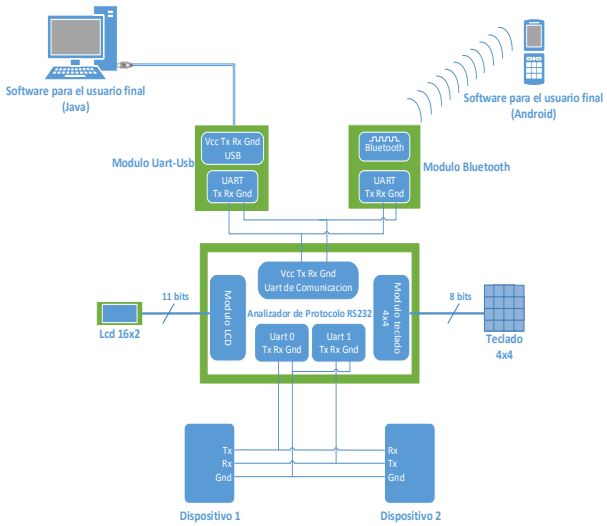



**Figura 8.** Hardware del sistema.

## 5.2. Diseño de Hardware embebido en la FPGA

La figura 9 muestra claramente el hardware que se embebió en la FPGA, el cual es un Sistema basado en el procesador NIOS II, diferenciándolo así del hardware físico que lo complementa. El diseño se lo realizó basándonos en un sistema computacional funcional: Procesador, memoria e interfaces de entrada y salida, adicionando módulos de comunicación UART, controlador de LCD, etc., adecuados a la necesidad de nuestro proyecto.

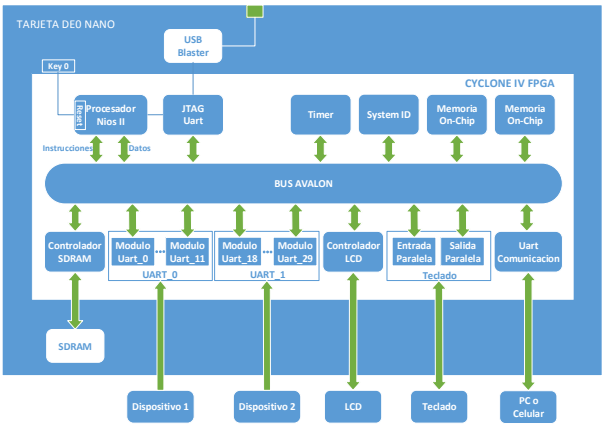



**Figura 9.** Sistema NIOS II.

Uno de los principales inconvenientes en la creación de este sistema es que el módulo NIOS II UART no permite la

modificación de los parámetros de configuración como tamaño en bits de datos, paridad y número de bits de parada después de la creación del sistema, lo que limitaría a nuestro proyecto a analizar dispositivos comunicándose bajo un solo tipo de configuración.

Para darle solución a este problema analizamos todas las combinaciones posibles de configuración. Para tamaño de datos: 7, 8, 9 bits; para la paridad: none, even y odd; y para el números de bits de parada: 1 y 2; entonces el número de combinaciones posible es 18.

Esto quiere decir que necesitamos 18 módulos NIOS II UART para analizar un dispositivo bajo cualquier configuración, debido a que analizamos dos dispositivos la cantidad de módulos necesario asciende a 36 agrupados en dos bloques de 18, un bloque para cada dispositivo.

La instanciación de todos estos módulos provoca un error que no se había tomado en cuenta en el diseño, el Procesador NIOS II solo permite 32 fuentes de interrupción externa y para instanciar todos estos módulos se necesitan 36 ya que es necesaria una interrupción por cada módulo.

Entonces hay que disminuir la cantidad de interrupciones externas, es decir disminuir la cantidad de módulos seriales, pero mantener la posibilidad de analizar todos los tipos de configuraciones.

La solución fue eliminar los módulos correspondientes a la configuración de paridad odd. Las razones son simples: un dato puede ser recibido correctamente con un receptor configurado con la paridad even si el emisor la envió con la paridad odd, siempre y cuando los demás parámetros de configuración son los mismos en el receptor y emisor. Esto se debe a que el bit de paridad solo se usa para verificar la validez de la trama recibida.

Por ejemplo, un emisor con configuración 7O1 quiere enviar el dato 11100100, entonces el trasmisor primero envía el bit de inicio que es un 0 lógico, luego el dato desde el bit menos significativo y luego envía el bit de paridad que es un 0 lógico en este caso ya que hay un número par de unos, por ultimo añade el bit de parada que es un 1 lógico, entonces la trama completa es 00010011101.

El receptor configurado con 7E1 detecta el bit de inicio, los siguientes 7 bits son de datos leídos desde el menos significativo por lo que el dato recibido es 11100100, es decir el mismo enviado, luego busca un 1 lógico como bit de paridad pero encuentra un 0 por lo que habilita la interrupción por error de bit de paridad y finalmente encuentra el 1 lógico como bit de parada.

Fuera del error provocado por error en el bit de paridad el receptor configurado con 7E1 recibe exactamente el mismo dato enviado por un receptor configurado con 7O1.

Entonces para configuraciones con paridad odd usamos la configuración even aunque nos dé una desventaja al provocar una interrupción por error de paridad los beneficios son mayores ya que reducimos a 24 la cantidad de interrupciones necesarias para el grupo de módulos seriales de nuestro analizador.

Cabe recalcar que cada bloque de módulos seriales va conectado a un solo pin de entrada de la FPGA, entonces a cada módulo de un mismo bloque de módulos seriales le llega la misma señal eléctrica, pero cada uno interpretara los datos de acuerdo a su configuración. El software se encarga de elegir un solo módulo de cada bloque para el análisis de los datos de acuerdo a la configuración de los dispositivos a analizar.

## 5.3 Diseño de Software

Una vez terminado con el diseño de la parte física de nuestro proyecto lo siguiente es el diseño del software que administra dicho hardware. Esta sección está comprendida por: la programación del procesador que se la realizara usando la herramienta NIOS II IDE para Eclipse, el desarrollo del software para el usuario final en Java y en Android usando NetBeans IDE.

## 5.4 Programación del Procesador NIOS II

Ahora lo principal para establecer una comunicación RS-232 es que los dispositivos a comunicarse deben tener los mismos parámetros de configuración. Entonces para poder interceptar los datos que se están intercambiando dos dispositivos que están usando el protocolo de comunicación RS-232 es necesario que nuestro analizador este usando la misma configuración. Los parámetros de configuración son velocidad de trasmisión, tamaño en bits de datos, paridad y número de bits de parada.

Para esto nuestro analizador de protocolos tiene dos opciones: un modo de configuración manual y un modo de autoconfiguración.

En el modo de configuración manual al usuario se le solicitara que ingrese todos los parámetros de configuración de la comunicación.

En el modo de configuración automático el analizador auto configura todos los parámetros de configuración de la comunicación.

La figura 10 muestra el diagrama de flujo del programa principal del analizador. Como vemos primero pide que ingrese el modo de configuración si es automático o manual.

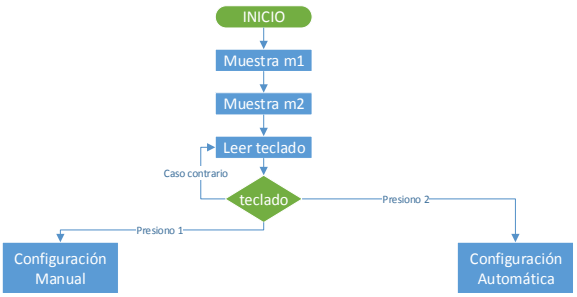



**Figura 10.** Flujo del programa Principal NIOS II.

El flujo de la configuración manual se muestra en la figura 11, simplemente el analizador pide al usuario que ingrese por teclado todos los parámetros de configuración, luego de haberse configurado correctamente el analizador estará listo para recolectar información, cuando llegue un dato, este es enviado al PC y al dispositivo Android para su respectiva visualización.

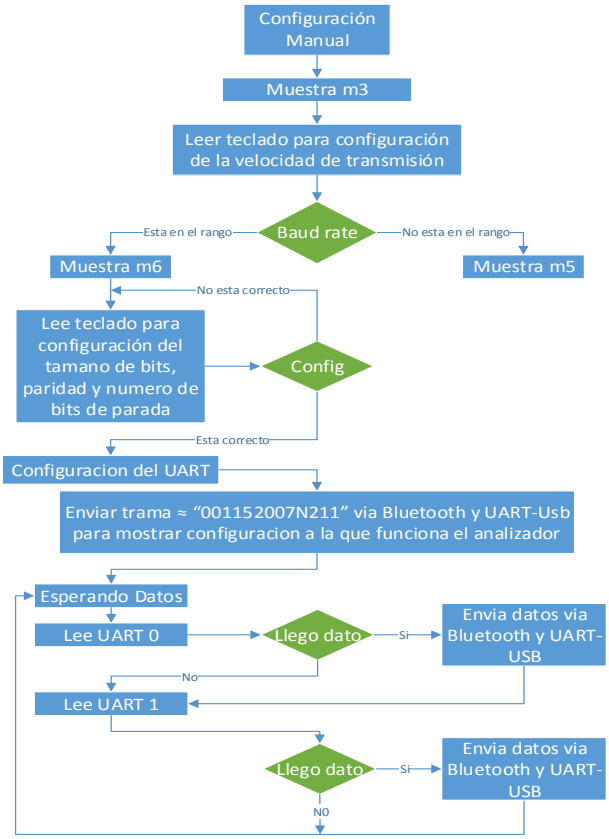



**Figura 11.** Configuración Manual.

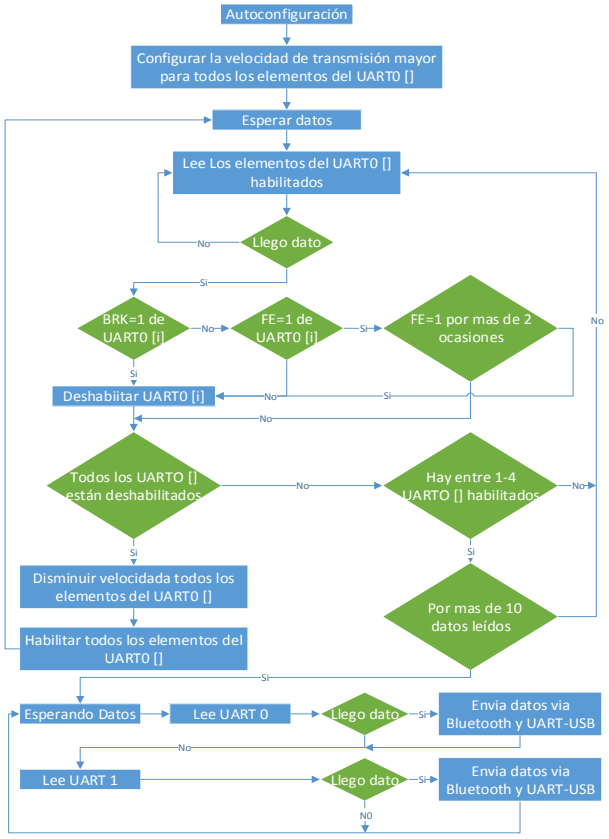



**Figura 12.** Configuración Automática.

El analizador tiene la capacidad de auto configurarse, para esto debe esperar que el o los dispositivos a analizar estén trasmitiendo datos para poder analizarlos y de esta forma determinar la velocidad con que se están trasmitiendo los bits y la configuración de tamaño de bits, paridad y parada. Para esta operación nos vamos a valer de dos banderas o bits de error que se encuentran en el registro de estado de nuestro módulo NIOS UART, el bit BRK y el bit FE.

Por otra parte el analizador auto configura velocidades de trasmisión puntuales y estándares como lo son de mayor a menor 115200, 57600, 28800, 19200, 9600, 4800, 2400 y 1200 baudios.

Cuando se menciona UART0 [ ] en la figura 12 se refiere al grupo de UARTs que lo conforman (ver figura 9) además cuando se menciona UART0 [ i ] se refiere a un elemento en particular de ese grupo de UARTs.

Inicialmente en la configuración automática se analizan todos los elementos del UART0 y se establece la velocidad mayor, cuando llega un dato y un elemento de este grupo cumple con la cantidad máxima de errores (FE y BRK), se deshabilita, es decir, se descarta esa configuración. Si se han descartado todas las configuraciones se disminuye la velocidad a la siguiente velocidad estándar. Si se mantienen 4 UARTs habilitados por más de 10 datos recibidos, se escoge el primero de la lista de los 4 UARTs habilitados y la velocidad de trasmisión actual para la lectura de los datos y se muestra en el LCD las 4 posibles configuraciones a la que se comunican los dispositivos y la posible velocidad de trasmisión.

## 5.5 Desarrollo de aplicación para el PC

Una de las opciones para la presentación de los datos analizados al usuario final es en una PC. Para esto se desarrolló una aplicación en el lenguaje de programación Java que muestra en una tabla los datos.

La recepción de los datos a mostrar se realiza a través de una conexión UART-USB (ver figura 8), la conexión con la PC es USB pero los datos llegan usando el protocolo RS232 con velocidad de trasmisión 115200 y configuración 8N1.

Nuestra necesidad es el uso del puerto serial por lo que se añadió una librería llamada RxTx para el acceso a este puerto desde la aplicación desarrollada.

La figura 13 muestra la interfaz gráfica del software mientras la figura 14 muestra el diagrama de flujos del mismo.

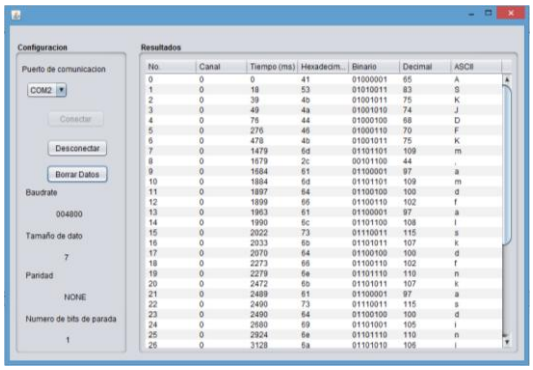



**Figura 13.** Interfaz gráfica del software para el PC.

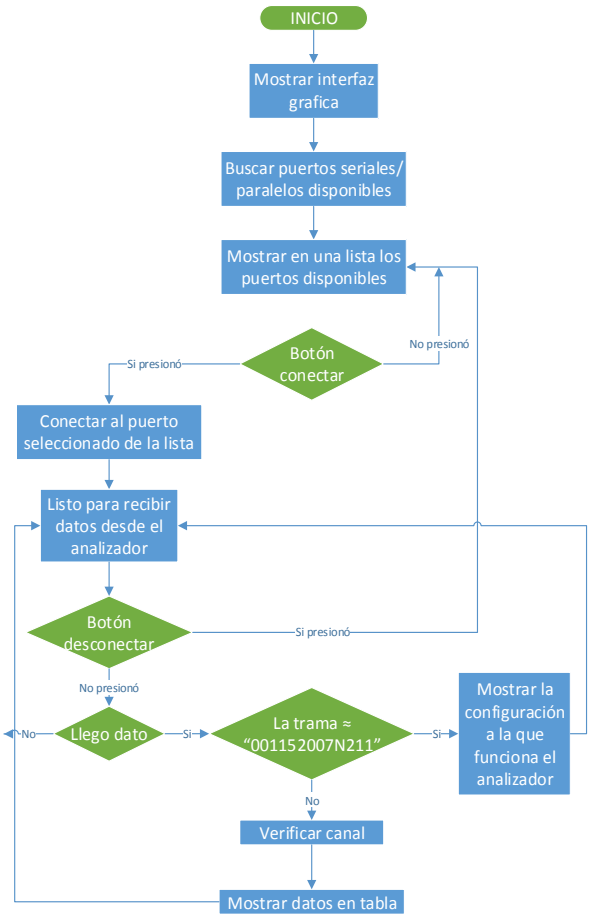



**Figura 14.** Flujo del software para el PC.

## 5.6 Desarrollo de aplicación para dispositivo Android

Para darle un toque de portabilidad a nuestro proyecto, una aplicación para un dispositivo móvil Android es la alternativa para mostrar los datos analizados los cuales se reciben vía Bluetooth desde el analizador.

Ahora, una aplicación en Android está formada por un conjunto de elementos básicos de visualización, conocidos como actividades. Estas actividades son las que controlan el ciclo de vida de las aplicaciones.

Nuestra aplicación tendrá una sola interfaz gráfica es decir una sola actividad, por tanto hablar del ciclo de vida de esta actividad es como hablar del ciclo de vida de nuestra aplicación. Para crear una aplicación estable en Android es necesario conocer perfectamente el ciclo de vida de una actividad.

Una actividad pasa por varios estados en su ciclo de vida, cuando existe un cambio de estado se dispara un evento el cual puede ser capturado por métodos o funciones definidos para dicha operación.

La figura 15 muestra claramente el método que se usa para capturar cada evento que se dispara dado un cambio de estado en la actividad dentro de su ciclo de vida.

Estos métodos pueden ser sobrescritos de acuerdo a la necesidad de la aplicación. Los métodos que sobrescribiremos en nuestra aplicación son: onCreate(), onResume() y onPause(), que son básicamente los más importantes en el desarrollo de una actividad.

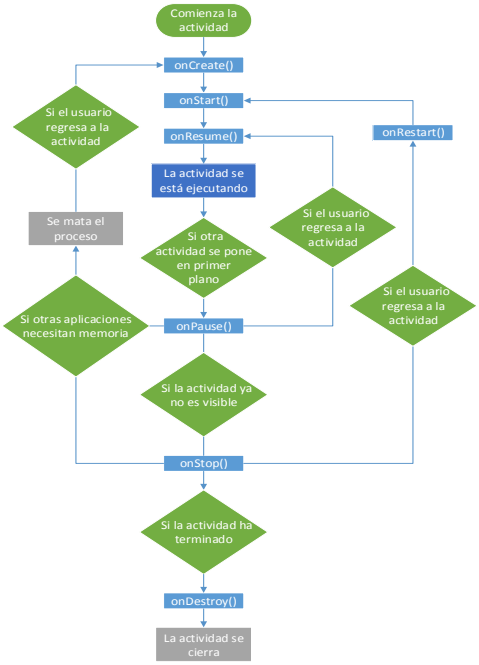



**Figura 15.** Ciclo de vida de una aplicación Android.

El método onCreate() es llamado en la creación de la actividad. Lo usaremos para establecer la interfaz de usuario e inicializar las variables que manejaran dicha interfaz además de la verificación y la activación del Bluetooth del dispositivo, también se crea el manejador para la estructura de datos que se usa para la recepción de la información vía Bluetooth.

La figura 16 muestra el flujo que tomara del método onCreate() en nuestro proyecto.

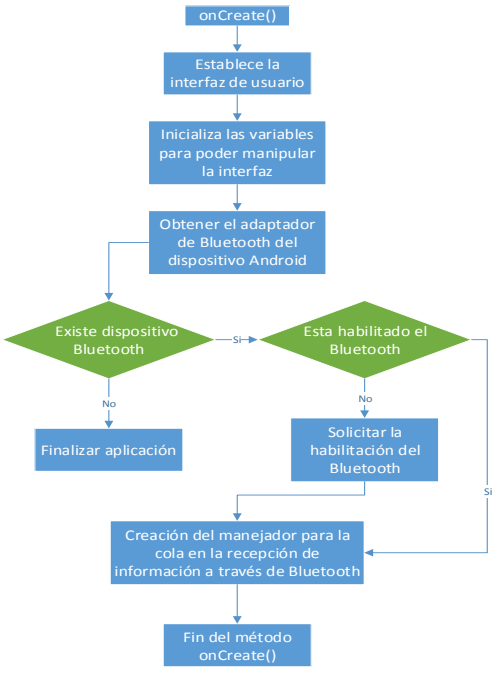



**Figura 16.** Flujo del método onCreate().

El método onResume() es llamado cuando la actividad ha terminado de cargarse en el dispositivo y comienza a interactuar con el usuario. En este método nuestra aplicación intenta establecer una conexión con el dispositivo Bluetooth de nuestro analizador y se crea el hilo de conexión para poder recibir o enviar información vía Bluetooth. La figura 17 muestra el flujo que toma del método onResume() en nuestro proyecto.

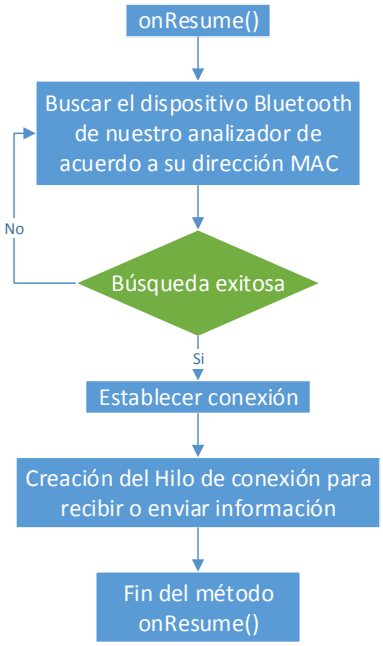



**Figura 17.** Flujo del método onResume().

La creación de un hilo de conexión se usa para que el proceso de lectura del flujo de entrada en la recepción vía Bluetooth sea constante e independiente del flujo del programa. El hilo de conexión es un bucle del que solo se sale si existe una excepción, la cual puede ser causada por cualquier fallo en la conexión Bluetooth. La figura 18 muestra el flujo que toma nuestro hilo de conexión.

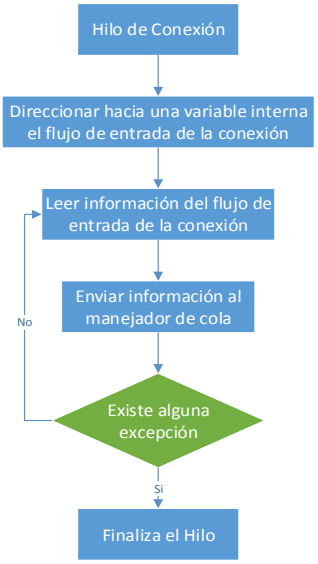



**Figura 18.** Flujo del hilo de conexión.

Dentro del flujo del hilo encontramos un proceso en el que se envía la información al manejador de la estructura de datos, el cual fue creado al inicio de la aplicación. La estructura de dato que se usa para la recepción de la información vía Bluetooth es una cola que usa el método FIFO para la manipulación de los datos. La figura 19 muestra el flujo del manejador de la cola que se crea al recibir datos.

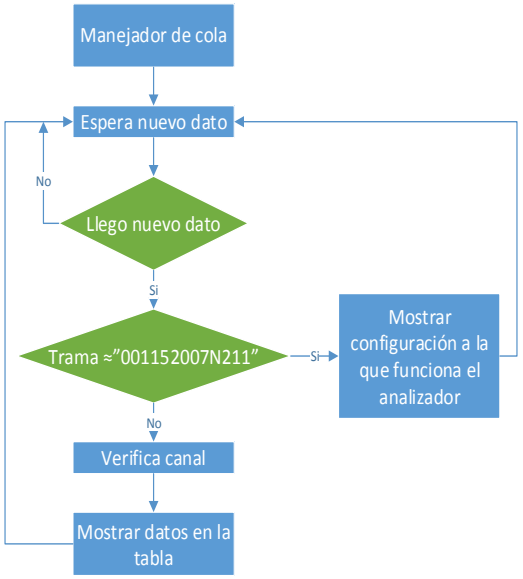
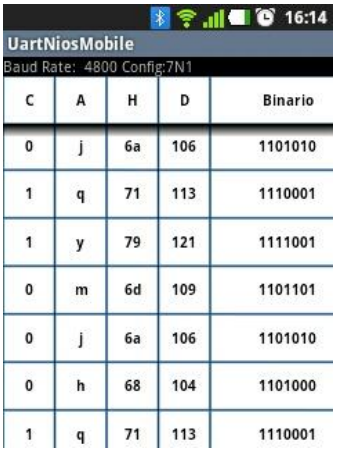



**Figura 19.** Flujo del manejador de cola.

Finalmente la figura 20 muestra la interfaz gráfica de la aplicación para dispositivos Android desarrollada para este proyecto.




| C | A | H | D | Binario |
|---|---|---|---|---|
| 0 | j | 6a | 106 | 1101010 |
| 1 | q | 71 | 113 | 1110001 |
| 1 | y | 79 | 121 | 1111001 |
| 0 | m | 6d | 109 | 1101101 |
| 0 | j | 6a | 106 | 1101010 |
| 0 | h | 68 | 104 | 1101000 |
| 1 | q | 71 | 113 | 1110001 |

**Figura 20.** Interfaz gráfica de aplicación Android

## 6. Pruebas y Resultados

Finalmente, esta sección muestra algunas pruebas de funcionamiento del analizador de protocolo RS232, además de un análisis de la eficiencia del mismo.

El diagrama esquemático de todas las pruebas es el mismo, y como podemos observar en la figura 21, lo que cambia son los equipos que participan en la comunicación a analizar (dispositivo 1 y 2).

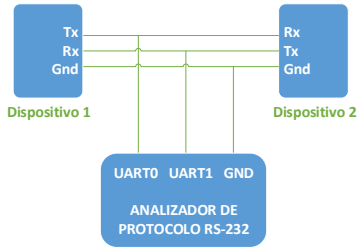



**Figura 18.** Diagrama de análisis.

Los dispositivos que se usaron para realizar las pruebas son:

- 1 Tarjeta de desarrollo DE0 nano.
- 1 Tarjeta de desarrollo DE2.
- 2 PC con el software AccesPort.
- 1 Modulo de comunicación UART-Bluetooth.

Con estos dispositivos se efectuarán todas las combinaciones de pares posibles para efectuar el análisis de dichas comunicaciones.

## 6.1 Resultados de efectividad

Para un entendimiento más profundo de la efectividad del analizador, analizamos los resultados agrupados por tres características de las pruebas: la velocidad de trasmisión, la comunicación y la configuración conformada por el tamaño de datos, paridad y número de bits de parada.

Los resultados de efectividad agrupados por la velocidad de trasmisión y dispositivos analizados se muestran en la tabla 1.

**Tabla 1.** Efectividad por velocidades.

| Vel. Real | Vel. Ok | Conf. Ok | Datos Ok | No. Datos |
|---|---|---|---|---|
| 115200 | 100% | 60% | 90% | 9x20 |
| 57600 | 100% | 60% | 70% | 12x20 |
| 28800 | 100% | 50% | 90% | 16x20 |
| 19200 | 80% | 30% | 70% | 17x20 |
| 9600 | 100% | 30% | 90% | 22x20 |
| 4800 | 100% | 60% | 90% | 16x20 |
| 2400 | 100% | 60% | 80% | 7x20 |
| 1200 | 100% | 60% | 80% | 7x20 |

Como vemos las pruebas realizadas a 19200 baudios son las que tienen la efectividad más baja, esto se debe al algoritmo usado para determinar la velocidad (descarte), ya que todas las velocidades excepto esta son la mitad de su antecesor, ejemplo 57600=115200/2, pero 19200!= 28800/2, esto dificulta descartar la velocidad 28800. Las pruebas realizadas a las demás velocidades mantienen un valor de efectividad muy parecido.

Por otro lado la tabla 2 muestra los resultados de efectividad agrupados por la configuración usada.

**Tabla 2.** Efectividad por configuración.

| Conf. Real | Vel. Ok | Conf. Ok | Datos Ok | No. Datos |
|---|---|---|---|---|
| 7N1 | 100% | 100% | 100% | 6x20 |
| 7N2 | 83% | 50% | 83% | 12x20 |
| 7E1 | 100% | 100% | 100% | 12x20 |
| 7E2 | 100% | 22% | 44% | 19x20 |
| 7O1 | 100% | 0% | 37% | 13x20 |
| 7O2 | 100% | 0% | 0% | 15x20 |
| 8N1 | 100% | 66% | 100% | 11x20 |
| 8N2 | 100% | 45% | 90% | 16x20 |
| 8E1 | 100% | 100% | 100% | 11x20 |
| 8O1 | 100% | 16% | 100% | 11x20 |
| 9N1 | 100% | 66% | 66% | 7x20 |
| 9N2 | 100% | 33% | 33% | 10x20 |
| 9E1 | 100% | 66% | 100% | 21x20 |
| 9E2 | 100% | 66% | 66% | 9x20 |
| 9O1 | 100% | 33% | 33% | 17x20 |

Al agrupar los resultados por configuración usada verificamos que las pruebas realizadas con configuración de paridad Odd son las que tienen menos efectividad. La causa de esto es que se eliminaron los módulos UARTs con configuración Odd para dar solución al problema de limitación de interrupciones externas del procesador Nios II.

La tabla 3 muestra los resultados agrupados por el dispositivo 1 usado en la comunicación analizada, ya que el analizador se auto configura de acuerdo a la configuración de este dispositivo que está conectado al UART0.

**Tabla 3.** Efectividad por comunicación.

| Comunicación | | Vel. | Conf. | Datos | No. |
|---|---|---|---|---|---|
| Disp1 | Disp2 | Ok | Ok | Ok | datos |
| PC | * | 98% | 43% | 73% | 14x20 |
| DE0 nano | * | 100% | 68% | 87% | 14x20 |
| DE2 | * | 94% | 56% | 68% | 9x20 |
| Bluetooth | * | 100% | 70% | 100% | 12x20 |

Agrupando las pruebas por dispositivos a analizar verificamos que los resultados mantienen valores de efectividad muy cercanos.

Y finalmente analizando las pruebas como un todo, obtendremos la efectividad general del analizador, la cual mostraremos en la tabla 4.

**Tabla 4.** Efectividad del analizador RS-232

| Velocidad | Configuración | Lectura de datos | No de datos |
|---|---|---|---|
| 97% | 53% | 78% | 13x20 |

## 7. Conclusiones

La tarjeta DE0 nano nos proporciona un entorno de desarrollo flexible en hardware. Esto debido a dos razones: en la FPGA integrada se pueden implementar sistemas computacionales funcionales ajustados a la medida de la aplicación específica que se le quiera dar, sus puertos de expansión permiten la integración de hardware adicional para la complementación de dichos sistemas.

El protocolo RS-232 aparentemente en desuso, aun es uno de los principales métodos de comunicación entre dispositivos periféricos como módulos GPS, módulos Bluetooth, módulos GSM, esto se debe a la sencillez de la configuración y al fácil entendimiento del método usado para la comunicación.

Los analizadores de protocolos son herramientas útiles para verificar si una determinada comunicación entre dos dispositivos se está efectuando de forma correcta y determinar la configuración de dicha comunicación.

El analizador de protocolo RS-232 que se diseñó en el presente trabajo es una herramienta que tiene una efectividad del 97% para la determinación de la velocidad de trasmisión en una comunicación, y del 53% en parámetros de configuración como tamaño en bits de datos, paridad y número de bits de parada.

El procesador NIOS II soporta hasta 32 fuentes de interrupción externa, esto limita la cantidad de módulos que pueden ser instanciados en el sistema y que además necesiten interrumpir al procesador, la mayoría de los dispositivos que poseen interfaces de entrada de datos necesitan interrumpir al procesador; en caso de necesitar más de 32 fuentes de interrupción, el diseño debería cambiar a Multi-core, es decir varios procesadores NIOS II instanciados en el mismo sistema ya que esto nos daría 32 fuentes de interrupción externa por cada procesador NIOS II instanciado.

## 8. Referencias


[1]. Ronald Mijail Dueñas, El estándar RS-232. http://interface-serial-rs232.blogspot.com/ Fecha de consulta: Septiembre 2013

[2]. Altera, NIOS UART Data Sheet. http://studies.ac.upc.edu/EPSC/SED/Practicas/ds_nios_uart.pdf Fecha de consulta: Septiembre 2013.

[3]. HC Serial Bluetooth Products,User Instructional Manual http://www.exp-tech.de/service/datasheet/HC-Serial-Bluetooth-Products.pdft71.pdf Fecha de consulta: Enero 2014.

[4]. Ideas y Tecnologías, Página HTML. http://ideastechnology.com/ Fecha de consulta: Enero 2014.

[5]. Altera, DE0 nano User Manual. http://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=593&FID=75023fa36c9bf8639384f942e65a46f3 Fecha de consulta: Septiembre 2013.