Detección de errores en números binarios sea cualquier longitud

Hugo Andrade, Marshal Flores

Universidad de las Fuerzas Armadas "E.S.P.E" Ingeniería en electrónica y Telecomunicaciones

hdandrad@espe.edu.ec, jmflores15@espe.edu.ec

***Abstracto*-La detección de errores en códigos binarios es muy importante para saber identificar si existe algún fallo en sistemas de comunicación digital, los cuales pueden sufrir alteraciones, por diferentes fenómenos físicos, al momento de la transmisión de información que parte desde un transmisor hacia un receptor. Es por ello que nuestro propósito en este trabajo es realizar un programa (app) en el cual se pueda realizar la detección de errores en números binarios sea cualquier longitud que se nos sea presentada, a través de la paridad par o impar, en este programa debe ser implementado un algoritmo propio para realizar la detección del error de cualquier número binario, para realizar este programa no se admite el uso de librerías que ya se encuentren prestablecidas, todo esto con el uso del software online propuesto que es MIT APP INVENTOR 2.**

***Palabras clave- app :** Una App es una aplicación de software que se instala en dispositivos móviles o tablets para ayudar al usuario en una labor concreta, ya sea de carácter profesional o de ocio y entretenimiento, a diferencia de una webapp que no es instalable.*

Para la construcción de este software utilizaremos los siguientes componentes

Componentes utilizados	
Software online MIT APP INVENTOR 2	Software utilizado para crear la app, cuya programación se la realiza mediante un entorno gráfico (programación por bloques).
Compilador móvil AI Companion	Después de realizar la programación en el software MIT APP INVENTOR 2, es necesario realizar las respectivas pruebas mediante el compilador móvil AI Companion para comprobar que el programa se esté ejecutando de manera correcta
Software Java NetBeans (Apoyo)	Software utilizado como método de apoyo para la elaboración del mismo programa en un entorno diferente (No gráfico).

Los diagramas usados en la realización de nuestra aplicación son:

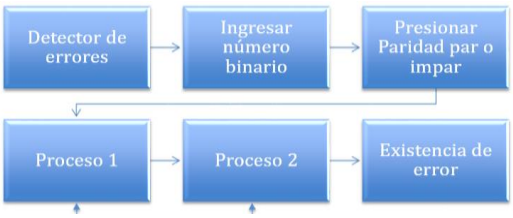


Diagrama 1.1 – Diagrama de bloques

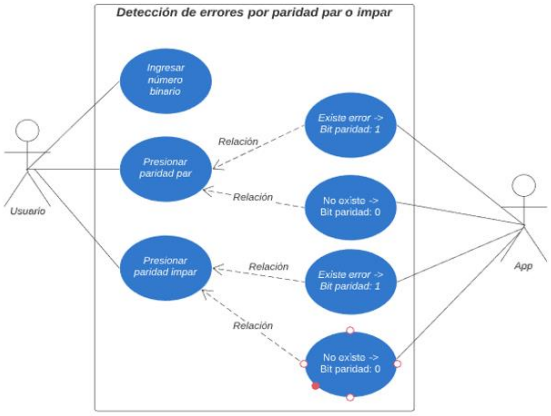


Diagrama 1.2 – Diagrama de uso.

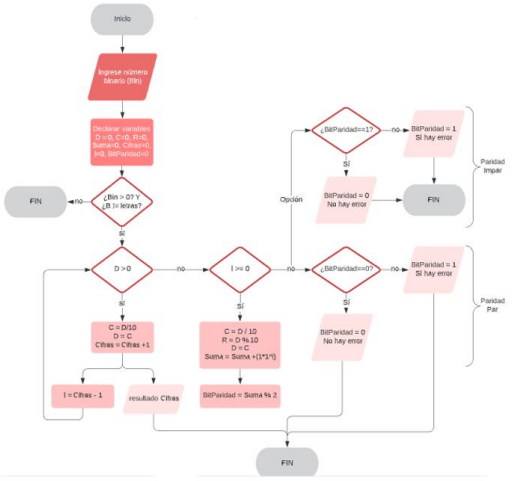


Diagrama 1.3 – Diagrama de flujo.

Ahora veremos cuáles fueron las variables utilizadas para el funcionamiento de la app

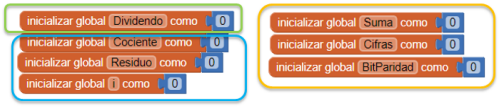


Figura 1.1 – Variables utilizadas.

Como podemos observar la figura 1.1 nos encontramos con todas las variables que han sido utilizadas para el correcto funcionamiento del programa (App). En el recuadro verde podemos observar la variable de entrada, la cual corresponde al número binario ingresado por el usuario.En el recuadro celeste podemos observar variables auxiliares que se han utilizado para realizar todos los cálculos matemáticos que fueron necesarios para que el funcionamiento de la App sea correcto.Y, por último, en un recuadro amarillo podemos observar las variables de salida, las cuales representan a los resultados que se visualizan en la interfaz del programa. Correspondientes al número total de cifras, la suma total de 1´s y el Bit de paridad que determina la existencia de error. (También ilustrados en la figura 1.2.).

Figura 1.2 – Ejemplo - Variables de salida visibles en la interfaz de usuario.

Detector de errores por paridad par o impar		
Variables	**Visibles**	**No visibles**
Dividendo	X	
Cociente		X
Residuo		X
i		X
Suma	X	
Cifras	X	
BitParidad	x	

Tabla 1.2 – Variables visibles y no visibles.

Ahora bien, necesitamos saber el tipo de cada variable que se ha utilizado, además de la función que realiza cada una en el programa. Esto se ilustra en la tabla 1.3.

Detector de errores por paridad par o impar		
Variables	**Tipo**	**Función**
Dividendo	Long	A esta variable se le asigna el valor del número binario ingresado por el usuario.
Cociente	Long	A esta variable se le asigna el resultado de la división, entre "Dividendo" dividido para 10.
		Al final del bucle While, asignamos el valor de "Cociente" a la variable "Dividendo".
Cifras	Entero	Esta variable es inicializada con el valor de 0. Después, mediante un bucle While, actúa como un contador, el cual va contabilizando el número de cifras del número binario.
i	Long	A la variable "i" se le asigna el valor de "Cifras" – 1. Actúa como exponente en el segundo bucle While.
Residuo	Long	La variable "Residuo" resulta del módulo entre "Dividendo" divido para Cociente".
Suma	Long	La variable suma es inicializada con el valor de 0. Es un contador de 1´s que depende de las variables "Residuo" e "i".
BitParidad	Long	La variable "BitParidad" arroja valores de 1 ó 0 para determinar si existe error en el número binario, ya sea por paridad par o impar.

Tabla 1.3 – Variables visibles y no visibles.

Programación en MIT App Inventor 2 mediante bloques

-Presionando el botón Paridad par

Al momento de presionar el botón Paridad par, el programa ejecutará la siguiente secuencia de bloques (Ilustrada en la figura 1.3), en la que se resuelve todo el algoritmo correspondiente a la detección de errores por paridad par.

El programa básicamente consiste en realizar divisiones sucesivas, mediante dos ciclos While o Entonces. El primer ciclo nos permite realizar el conteo del número de cifras totales del número binario ingresado por el usuario que puede ser de cualquier longitud. En la figura 1.4. podemos ver el proceso en el cual se basa este apartado.

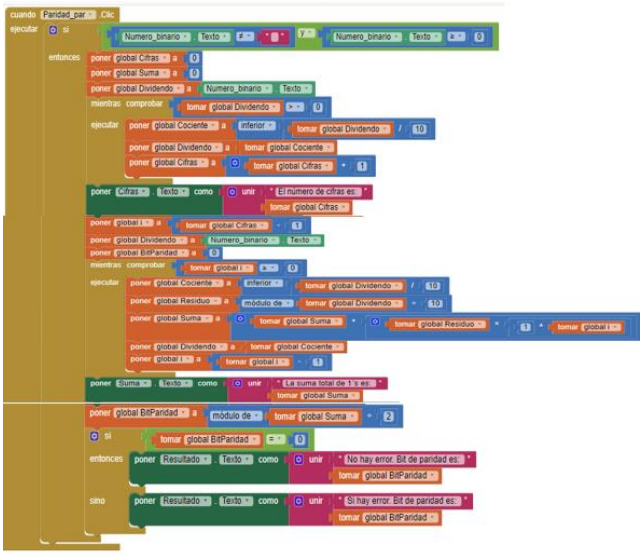


Figura 1.3 – Detección de errores mediante paridad par.

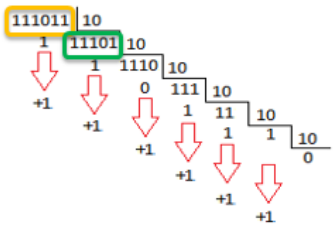


Figura 1.4 – Ilustración del funcionamiento del conteo de cifras.

Como podemos observar en la figura 1.4. El dividendo (Recuadro amarillo) siempre se va a dividir para 10. Y el cociente resultante (Recuadro verde) es un número entero que va a asignarse nuevamente a la variable dividendo. Y así sucesivamente, hasta que el dividendo sea 0. Cada vez que ocurra este proceso (mediante el bucle repetitivo While) se va a sumar en una unidad el número de cifras.

Ahora bien, si observamos la figura 1.5 y teniendo el número de cifras obtenidas en el primer ciclo While, nosotros creamos una variable i, la cual es igual al número de cifras -1. Este valor asignado en i, es el exponente (Recuadro amarillo) de la base 1 que va a multiplicarse por el valor del Residuo obtenido (Recuadro verde). Entonces al aplicar el segundo ciclo While y al sumar todos estos valores asignándolos a la variable Suma, obtenemos el número total de 1's. Los cuales nos indicarán si el número es par o impar realizando el módulo entre la variable suma y el valor de 2 y esto asignándolo a la variable BitParidad. De esta manera será posible detectar o determinar si existe o no error en el caso de paridad par, mediante una estructura condicional If-Else.

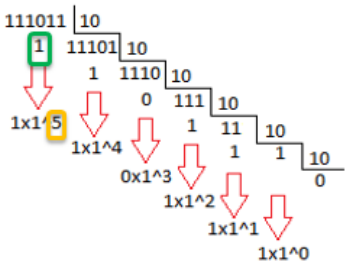


Figura 1.5 – Ilustración del funcionamiento del conteo de 1's.

Para entender de mejor manera el funcionamiento del programa, hemos realizado un código mediante Java NetBeans, en el que explicamos paso a paso, el proceso y resolución del programa.

Declaracion de Variables

```
package PaqueteLogica;

import java.util.Scanner;

public class Programa {

    public static void main(String[] args)
    {
        // Declaración de variables
        long Numero_binario;
        long Dividendo;
        long Cociente;
        int Cifras;
        long Residuo;
        long Suma = 0;
        long i;
        long BitParidad;
        Scanner escaneo = new Scanner(System.in);
```

Figura 1.6 – Declaración de variables.

Desarrollo del Programa

```
//Mensaje Información - Titulo del programa
System.out.print("\tDetector de errores.\n\n");

//Mensaje Información - Ingresar el número binario
System.out.print("Ingrese el número binario: ");
Numero_binario = escaneo.nextLong();   //Leer número binario

Dividendo = Numero_binario;   //Asignación del número binario a la variable Dividendo
Cifras = 0;                   //Inicializar variable Cifras con el valor de 0

//Mientras Dividendo sea diferente de 0
while(Dividendo!=0)
{
    Cociente = Dividendo/10;  //Asignación de la división entre Dividendo y 10 a la variable COciente
    Dividendo = Cociente;     //Asignación de la variable Cociente a la variable Dividendo
    Cifras++;                 //Contabilizar el número de cifras mientras se cumpla la condición
}

//Mensaje Información - Número de Cifras
System.out.println("El numero de cifras es: "+Cifras);

Dividendo = Numero_binario;   //Asignación del número binario a la variable Dividendo
i = Cifras-1;                 //Número de cifras totales menos 1, asignado a la variable i (Exponente)

//Mientras i sea mayor o igual que 0
while(i>=0)
{
    Cociente = Dividendo/10;  //Asignación de la división entre Dividendo y 10 a la variable COciente
    Residuo = Dividendo%10;   //Módulo de dividendo entre 10, asignado a la variable Residuo

    Suma = Suma + Residuo *(long)Math.pow(1,i);   //Acumular en Suma el número total de 1's
    Dividendo = Cociente;     //Asignación de la variable Cociente a la variable Dividendo
    i--;                      //Restar el valor de i menos 1, cada vez que se repita el bucle
}
```

Figura 1.7 – Desarrollo del programa.

Determinar existencia de error

```
    //Mensaje Información - Número total de 1's
    System.out.println("La suma total de 1's es: "+Suma);

    BitParidad = Suma%2;          //Módulo de Suma entre 2 (Para verificar si la suma total de 1's es par o impar)

    if(BitParidad == 0)           //Si BitParidad == 0 (Suma total de 1's es par)
    {
        System.out.println("Si hay error. Bit de paridad es "+BitParidad);
    }
    else                          //Sino BitParidad !=0 (Suma total de 1's es impar)
    {
        System.out.println("No hay error. Bit de paridad es: "+BitParidad);
    }

  }
}
```

Figura 1.8 – Determinar existencia de error, mediante el bit de paridad y una condición If-Else.

- **Presionando el botón Paridad Impar**

Para el caso en el que se presione el botón de paridad impar, se efectuará el mismo proceso que se realizó con el caso de paridad par. La única diferencia la encontramos en la sección última del programa (Recuadro amarillo), es decir, en la condición If-Else. Tal como se muestra en la figura 1.9.

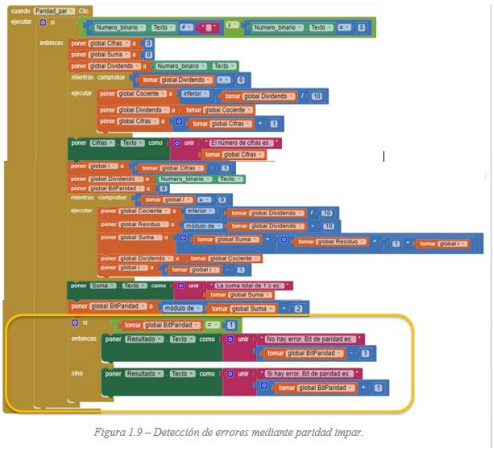


Figura 1.9 – Detección de errores mediante paridad impar.

Como podemos observar, en este caso, si el bit de paridad BitParidad == 1, entonces no existirá error y se mostrará en su salida que el bit de paridad es igual a 0, ya que le restamos el valor de 1. Caso contrario, si existirá error y en su salida se mostrará que el bit de paridad es igual a 1, ya que, en este caso le sumamos el valor de 1.

De igual manera presentamos el código realizado en Java NetBeans (Figura 2.1) para comprender de mejor manera esta apartado.

```
    if(BitParidad == 1)           //Si BitParidad == 1 (Suma total de 1's es impar)
    {
        System.out.println("No hay error. Bit de paridad es "+(BitParidad-1));
    }
    else                          //Sino BitParidad !=1 (Suma total de 1's es par)
    {
        System.out.println("No hay error. Bit de paridad es: "+(BitParidad+1));
    }

}
```

Figura 2.1 – Detección de errores mediante paridad impar (Código).

DESCRIPCIÓN DE PRERREQUISITOS Y CONFIGURACIÓN

En lo que corresponde a la utilización de aplicaciones secundarias para que el programa funcione correctamente, podemos decir que no se ha necesitado de ninguna que influya directamente sobre su programación. Sin embargo, no está por demás mencionar que se ha utilizado una herramienta de software extra (Java NetBeans) como un programa de apoyo para poder comprender de mejor manera el funcionamiento del programa, además de la aplicación móvil MIT AI2 Companion para poder realizar pruebas e ir compilando el programa a la vez que se iba realizando la programación en MIT APP INVENTOR 2.

Por otro lado, para el correcto funcionamiento del programa, hay que tener en cuenta varios puntos específicos. Los cuales son indispensables para no tener ningún inconveniente al momento de ejecutar el programa.

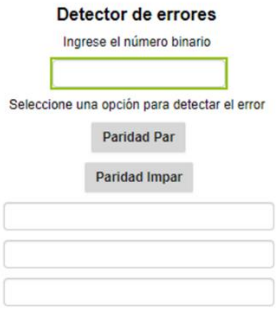
- El ingreso de datos debe ser única y exclusivamente en número binario. No se aceptan caracteres, ni números negativos.

- Para la descarga e instalación de la aplicación en nuestros dispositivos móviles, debemos descargar el archivo en formato APK y posterior a ello realizar la instalación de la aplicación. Se tendrá que permitir el origen de archivos desconocidos en la sección de ajustes del dispositivo móvil para su correcta instalación En nuestro caso la aplicación que hemos elaborado se llama Proyecto1_Derrores.

Uso de la aplicación

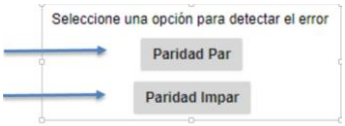
1.- Ingreso de datos

En esta pantalla el usuario debe digitar el número binario en la casilla que se encuentra señalada a continuación



2.-Escoja una opción

Presione el botón Paridad Par para detectar el error del número binario ingresado, mediante el método de paridad par. O Presione el botón Paridad Impar para detectar el error mediante el método de paridad impar.



3.-Existencia de error

En los tres recuadros siguientes le aparecerá el número de cifras del número ingresado, la suma total de 1´s y la existencia de error con su bit de paridad.

CONCLUSIONES

- El diseño e implementación del programa, se lo ha realizado de manera que cumpla el objetivo de realizar la detección de error de un número binario, mediante paridad par e impar. Todo este proceso se llevó a cabo con la programación en un entorno gráfico que nos ofrece MIT APP INVENTOR 2, el cual nos permitió realizar la aplicación para que cualquier usuario, pueda utilizarlo desde su dispositivo móvil. Tal como se ha podido observar en las diferentes figuras del presente informe, el diseño de la app es muy intuitiva y amigable al usuario.

- La detección de errores por paridad par o impar es muy esencial para sistemas de comunicación digital, los cuales necesitan de valores precisos y sin equivocaciones. La manera de entender cómo funciona cada paridad es muy sencilla. En el caso de paridad par, tendríamos un bit de paridad de 0 si la suma total de 1's es par, es decir, no existiría error. Caso contrario el bit de paridad tendría el valor de 1 y por lo tanto existiría error. Por otro lado, en el caso de paridad impar, tendríamos un bit de paridad de 0 si la suma total de 1's es impar. Caso contrario el bit de paridad tendría el valor de 1 si la suma total de 1's es par. Cada uno de los conceptos de los métodos de detección de error mencionados, fueron muy importantes y necesarios para la realización del algoritmo, el cual fue implementado en App Inventor.

- La realización del programa se la realizó en base a todo lo investigado. Prácticamente necesitamos realizar el conteo de 1's, y mediante el método de paridad que se escoja, se determinará si existe error dependiendo si la suma mencionada es par o impar. De esta manera logramos diseñar un algoritmo, el cual se basó principalmente en divisiones sucesivas para realizar el conteo de cifras, la suma total de 1's y por último encontrar el bit d paridad que determinó el error. Todo esto, mediante el uso de la estructura repetitiva While. Cabe mencionar que todo el algoritmo se lo ha hecho de manera correcta, incluso utilizamos un software de apoyo (Java NetBeans) para realizar las diferentes comprobaciones y pruebas que el programa requería. Y al momento de realizar la investigación con los artículos de los autores investigados, podemos darnos cuenta de la gran similitud y relación que encontramos con respecto a nuestro trabajo de investigación, todo esto se lo ha detallado en sección de Estado del arte.

Bibliografía

[1] F. K Law, M. Rakib Uddn, Nur Musyiirah Masir1, Ynog Hyub Won, obtenido de: https://sci-hub.tw/https://ieeexplore.ieee.org/document/8527192

[2] Peter Farkaš, Tomáš Janvars y Katarína Farkašová, Eugen Ružický, obtenido de :

https://sci-hub.tw/https://ieeexplore.ieee.org/document/8337547

[3] Silvia Anggraeni, Fawnizu Azmadi Hussin and Varun Jeoti obtenido de: https://sci-hub.tw/https://ieeexplore.ieee.org/document/6869526

(Perez, Mandado, 2016) - libro Sistemas Electrónicos Digitales obtenido de:

https://books.google.com.ec/books?id=V7JpKkZaEYMC&pg=PA18&lpg=PA18&dq=codigos+de+paridad&source=bl&ots=sTHg4rJA6R&sig=ACfU3U3FsgLxKEJBQNhOunOHHGr4NbZgNg&hl=es&sa=X&ved=2ahUKEwjzprfus4TqAhXNQjABHTmsA98Q6AEwEXoECBEQAQ#v=onepage&q=codigos%20de%20paridad&f=false

(Fernandez, Gil, 2015) - Una introduccion a los codigos detectores de error y correctores de errores obtenido de: https://www.matematicaparatodos.com/varios/Codigos.pdf