

Implementación de sistema de comunicación digital inalámbrico utilizando radios definidos por software

Luis Alberto Herasme Cuevas
Ingeniería electrónica y de comunicaciones
Instituto tecnológico de Santo Domingo
Santo Domingo, República Dominicana
1088668@est.intec.edu.do

Gabriel Santos
Ingeniería electrónica y de comunicaciones
Instituto tecnológico de Santo Domingo
Santo Domingo, República Dominicana
1085078@est.intec.edu.do

Resumen—En este documento se describe el diseño y funcionamiento de un sistema de comunicación digital ASK OOK no coherente con codificación de canal y técnicas de espectro disperso, además se indica como fue implementado mediante el uso de radios definidos por software.

Palabras clave—Radios definidos por software, Modulación digital, Códigos de bloques lineales, DSSS

I. INTRODUCCIÓN

Este proyecto fue realizado con la finalidad de lograr la transmisión y recepción inalámbrica de imágenes utilizando radios definidos por software o SDR por sus siglas en ingles. Específicamente utilizamos el LimeSDR-Mini para la implementación. El sistema cuenta con modulación y demodulación pasa banda, codificación de canal, técnicas de espectro disperso y un mecanismo de sincronización de trama. El procesamiento no es realizado en tiempo real, las señales son enviadas utilizando los SDR y cierto grado de procesamiento es realizado en los mismos, sin embargo, luego de las señales ser procesadas en el SDR, estas son guardadas en un archivo para posteriormente ser procesadas nuevamente en Octave.

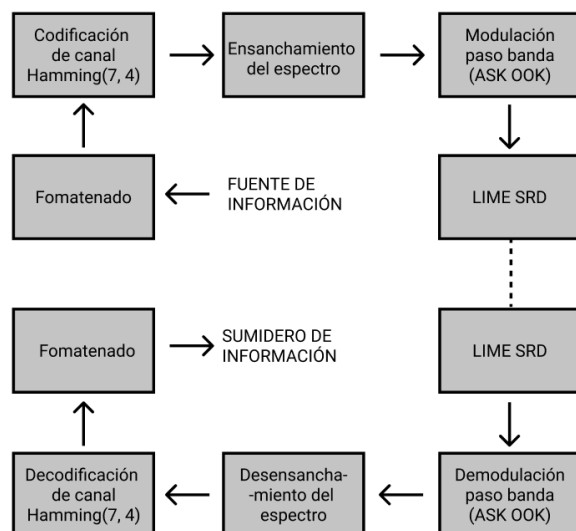


Fig. 1. Diagrama de bloques general del sistema.

En la figura 1, puede observarse todos los bloques que componen este sistema de comunicación y que serán detallados a lo largo de este documento.

II. DISEÑO DE TRAMA

Nuestra trama cuenta con tres secciones, el preámbulo que lo utilizamos para sincronizar la trama, la cabecera que la utilizamos para describir la forma de la información que contiene la trama y los datos que como su nombre indica son los datos que serán transmitidos. La cabecera tiene un tamaño 40 bits y este tamaño es fijo, ocho de estos 40 bits definen la altura de la imagen que será enviada, otros ocho la anchura y los últimos 24 bits definen la cantidad de bytes que contiene la imagen.

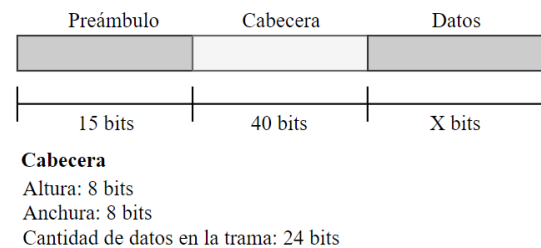


Fig. 2. Diseño de la trama.

Además, a cada trama enviada se le añaden 100 bits de espaciamento, donde no se envía nada, esto lo logramos añadiendo 100 ceros al principio de la trama.

III. GENERACIÓN DE LOS DATOS A TRANSMITIR (FORMATEADO)

Comenzando con la información que se desea enviar, en nuestro caso una imagen de RGB. Antes de realizarse la transmisión a través del LimeSDR primero se procesa utilizando nuestra función “generar_datos.m” para ser convertido a un archivo intermedio, en esta:

- Convierte la matriz en un vector de bits, y se concatenan estos datos con la cabecera. La cabecera consiste en información acerca de la imagen a enviar. Esta información es el ancho, alto, y la cantidad de datos de la trama.
- Luego de tener esta cadena de bits, se codifican los datos junto con su cabecera utilizando una función de codificación Hamming con una tasa de bits de 4/7.
- Aplica DSSS al vector codificado con Hamming.
- Al esparcido con DSSS, se le agrega una secuencia de pseudo ruido como preámbulo. Esta secuencia de será útil para la sincronización de trama en la parte de recepción y detección.

IV. ENVIAR DATOS CON FORMATO POR GNU RADIO

En GNU Radio se toma el archivo de datos generado anteriormente y se transmite y recibe en el mismo LimeSDR. Los bloques empleados para la transmisión y recepción fueron:

- File source: Este bloque toma como argumento principal la dirección de un archivo para leerlo e importar su contenido a GNU.
- Repeat: Este bloque toma un vector y repite cada uno de sus elementos K veces, siendo K un parámetro establecido por nosotros.
- Float to complex: Convierte del tipo de dato flotante a complejo.
- LimeSuite Tx sink: Este bloque sirve como interfaz para el LimeSDR.
- LimeSuite Rx source: Este bloque sirve como interfaz para la recepción de datos del LimeSDR.
- RMS: Por sus siglas en inglés Root Mean Squared. Es el bloque es utilizado para la demodulación del ASK OOK.
- File Sink: Utilizado para guardar los datos generados por GNU Radio hacia un archivo.

Una vez teniendo el archivo generado por GNU Radio, se emplea la función de Octave “leer_datos.m”. Esta se encarga de realizar la detección del inicio de la trama con los datos enviados en el archivo generado por GNU Radio. Para realizar esta detección se hace una correlación sobre todo el archivo generado por GNU Radio con una copia de la señal de pseudo ruido utilizada antes de la transmisión. Una vez sabiendo en dónde se encuentra el inicio de la trama, se llevan a cabo:

- Lectura de datos.
- La decodificación Hamming.
- El DSSS de la señal.

Ya teniendo los datos correspondientes a la imagen transmitida, se reconstruye y luego se grafica para confirmar su transmisión.

V. MECANISMO DE SINCRONIZACIÓN DE TRAMA

Para la sincronización de trama se utilizaron secuencias de pseudo ruido, cuando se realiza la autocorrelación de una secuencia de pseudo ruido con una versión desfasada de la misma, la función de autocorrelación presenta su valor máximo cuando el desfase es igual a cero; podemos utilizar esta propiedad para realizar la autocorrelación de una versión almacenada que tenemos de la secuencia de pseudo ruido con el mensaje que nos esté llegando al radio, una vez observemos que la autocorrelación alcanza cierto valor podemos determinar que se enviara un mensaje. Estas secuencias pueden ser generadas con registros de desplazamiento de retroalimentación lineal, el que usamos en específico puede verse en la siguiente figura:

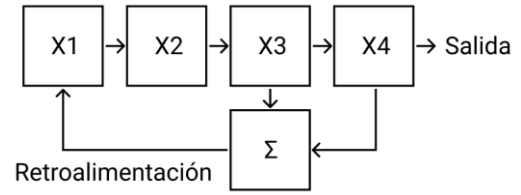


Fig. 3. Registro de desplazamiento de retroalimentación lineal.

Este registro genera la secuencia de pseudo ruido que utilizamos para realizar la autocorrelación, esta secuencia es la siguiente:

$$PN = [111100010011010]$$

Cuando se realiza la autocorrelación de este vector PN de con una versión desfasada del mismo se puede ver la siguiente gráfica:

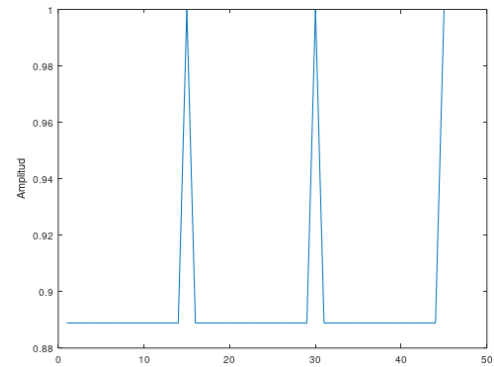


Fig. 4. Función de autocorrelación de pseudo ruido.

En esta grafica puede apreciarse la propiedad más importante que queremos aprovechar de las secuencias de pseudo ruido:

$$R_x(\tau = 0) = 1$$

$$R_x(\tau = 0) \geq R_x(\tau)$$

Realizamos la autocorrelación de la siguiente forma, primero sabemos que la función de autocorrelación para códigos PN es la siguiente:

$$R_x(\tau) = \frac{1}{K} \left(\frac{1}{T_0} \right) \int_{-T_0/2}^{T_0/2} x(t)x(t-\tau)dt$$

$$K = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} x^2(t)dt$$

Pero como estamos utilizando señales discretas debemos hacer una pequeña modificación de esta de la siguiente forma, donde m es el mensaje que nos llega y P es la secuencia de pseudo ruido:

$$R_s(\tau) = \frac{1}{K} \left(\frac{1}{T_0} \right) \sum_{n=\tau}^{\tau+P} P(n)m(n)$$

$$K = \frac{1}{T_0} \sum_{n=\tau}^{\tau+P} P(n)^2$$

$$R_s(\tau) = \frac{1}{\sum_{n=1}^{|P|} P(n)^2} \sum_{n=\tau}^{\tau+P} P(n)s(n)$$

Al realizar esta operación sobre la señal que nos llega podemos determinar si se ha enviado un mensaje si se alcanza cierto valor de amplitud.

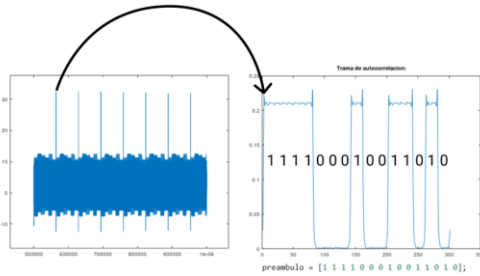


Fig. 5. Sincronización de trama usando de pseudo ruido.

VI. TÉCNICA DE MODULACIÓN/DEMULACIÓN DIGITAL

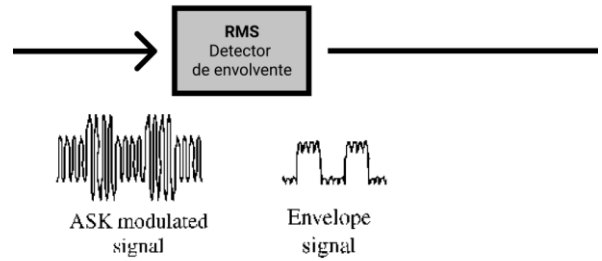
Esto fue realizado parcialmente en GNU radio, lo primero que hicimos fue leer el archivo que se generó en el bloque de dar formato a los datos.

Para llevar a cabo la modulación ASK, se utilizó un bloque llamado repetir que incrementa la cantidad de veces que se repite una muestra. Entonces luego de realizar esta operación, se hace una multiplicación con una cosenoidal, y el resultado ya está modulado.

En demodulación se aplica el bloque RMS (Root Mean Square) como detector de envolvente.

$$f_{rms} = \sqrt{\frac{1}{T} \int_0^T [f(t)]^2 dt}$$

En donde T es el tiempo de símbolo. Con este bloque es posible calcular la energía de la señal que nos llega dentro de un intervalo de tiempo (dentro de los tiempos de símbolo) y, debido a que utilizamos ASK OOK, los distintos niveles de amplitud corresponden a distintos niveles de energía de los símbolos recibidos, información suficiente para realizar la detección.



Lee, T., Lee, C., Ciou, Y., Huang, C., & Wang, C. (2008). All-MOS ASK Demodulator for Low-Frequency Applications. IEEE Transactions on Circuits and Systems II: Express Briefs, 55, 474-478.

Fig. 6. RMS como detector de envolvente.

VII. ENVÍO DE LA SEÑAL

Una vez tenemos nuestra señal filtrada con el bloque que le antecede, esta señal es multiplicada por una señal cosenoidal de unos 200kHz para moverla de la banda base es transmitida usando el bloque de transmisión para el LimeSDR.

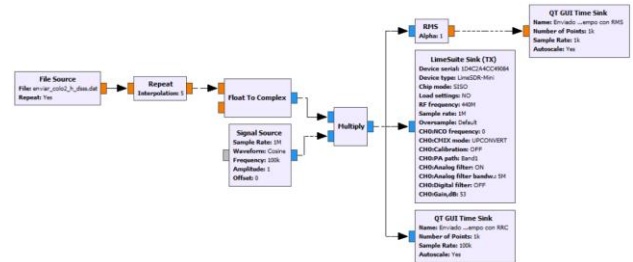


Fig. 7. Modulador ASK, OOK.

Este bloque de LimeSDR está configurado con:

- Modalidad de transmisión en conversión ascendente.
- Frecuencia para la conversión ascendente de unos 400MHz (este valor es modificable con un deslizador en la interfaz gráfica generada durante la operación del sistema).
- Filtro análogo pasa banda para la transmisión con un ancho de banda de unas 5 veces la frecuencia de muestreo usada y centrado en la frecuencia de la conversión ascendente.
- Filtro digital pasa banda para la transmisión con un ancho de banda de una vez la frecuencia de muestreo usada y centrado alrededor de la frecuencia de transmisión.

VIII. RECEPCIÓN Y DETECCIÓN DE SÍMBOLOS

Para la recepción se utilizó el siguiente esquema:

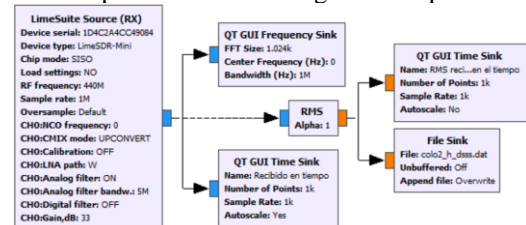
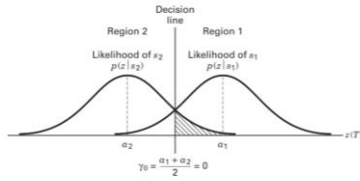


Fig. 8. Esquema recepción GNU octave.

Para la detección de símbolos se utilizó una estimación de máxima verosimilitud.



$$p(z|s_2) = \frac{1}{\sigma_0 \sqrt{2\pi}} \exp \left[-\frac{1}{2} \left(\frac{z - a_2}{\sigma_0} \right)^2 \right]$$

Bernard, S., 2021. Digital Communications, Fundamentals And Applications. 2nd ed. New Delhi: Dorling Kindersley, p.182.

Fig. 9. Modelo distribución de probabilidades de símbolos.

Para esto se enviaron 1, 0, en GNU radio y se generó la siguiente grafica.

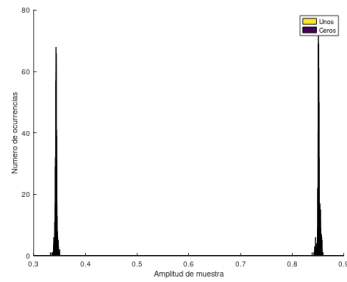


Fig. 10. Histograma de unos.

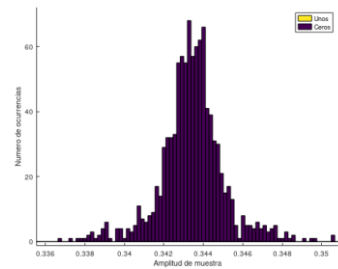


Fig. 11. Histograma de ceros.

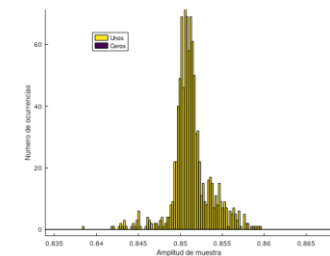


Fig. 12. Histograma de unos.

Luego se hizo variar el umbral de la siguiente forma:

$$Umbral = linspace(mean(ceros), mean(unos), 100);$$

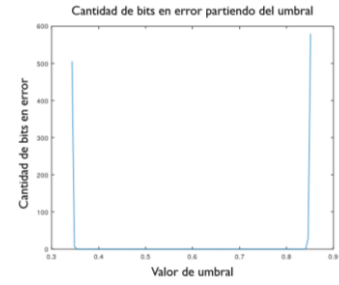


Fig. 13. Valores posibles probados para el umbral.

Luego se utilizó el valor de 0.6 como umbral como se puede ver en la imagen inferior.

Octave

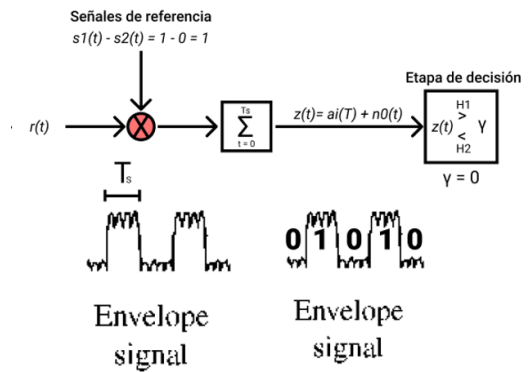


Fig. 14. Receptor de correlación binario utilizado.

IX. TÉCNICA DE CODIFICACIÓN/DECODIFICACIÓN DE CANAL

Como técnica de codificación de canal se utilizaron códigos de bloques lineales, estos transforman un vector de k bits en un vector de mayor longitud de n bits, y estos códigos son caracterizados con estos dos valores (n, k), en nuestro caso utilizado un código de bloques lineales llamado Hamming (7, 4). La tasa de es n/k es decir $4/7 \approx 0.57$.

Para realizar esta codificación utilizamos la siguiente matriz de paridad:

$$P = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Con esta matriz podemos crear una nueva matriz llamada matriz generadora que es utilizada para codificar los mensajes, esta matriz es una concatenación de la matriz de paridad con una matriz de identidad de la siguiente forma:

$$G = [PI]$$

Para denotar un mensaje codificado usamos la letra "U" y un mensaje sin codificar usamos la letra "m", realizamos la codificación de la siguiente forma:

$$U = mG$$

Además, utilizamos una matriz de verificación de paridad “H”. En la teoría de la codificación, una matriz de verificación de paridad de un código de bloque lineal es una matriz que describe las relaciones lineales que deben satisfacer los componentes de una palabra de código [2]. La matriz de paridad se define de la siguiente forma:

$$H = [IP^T]$$

Utilizamos esta matriz para calcula los síndromes, “S” significa síndrome, y “r” significa mensaje recibido:

$$S = rH^T$$

Posteriormente utilizamos dichos síndromes para estima los errores que se cometieron al enviar los mensajes:

TABLE I. ERRORES Y SÍNDROMES

Errores	Síndromes
0000000	000
0000001	111
0000010	110
0000100	101
0001000	011
0010000	001
0100000	010
1000000	100
0100001	101

Fig. 15. Errores con los síndromes que estos producen.

Una vez hemos calculado el síndrome del bloque recibido podemos utilizar una tabla similar a la de la figura 5, para estimar el error que provocó el síndrome y remover dicho error sumándolo al mensaje. Por otra parte, podemos obtener el mensaje original tomando los 4 bits a un extremo del mensaje codificado luego de removerle el error ya que nuestro código es sistemático.

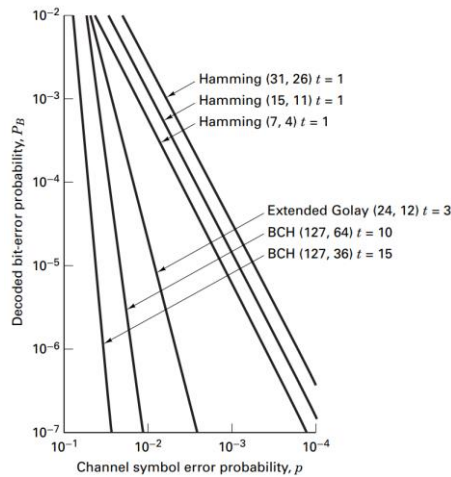


Fig. 16. Probabilidad de error de bit versus probabilidad de error de símbolo de canal para varios códigos de bloque.

```
octave:5> c = codificar_hamming_7_4([1 0 0 1 0 1 1 0])
c =
    1    0    0    1    0    1    1    0    0    1    0    1    1    0

octave:6> decodificar_hamming_7_4(c)
ans =
    1    0    0    1    0    1    1    0

octave:7>
```

Fig. 17. Codificación en funcionamiento.

X. SEÑALES

En las siguientes figuras podemos ver las señales través del SDR.

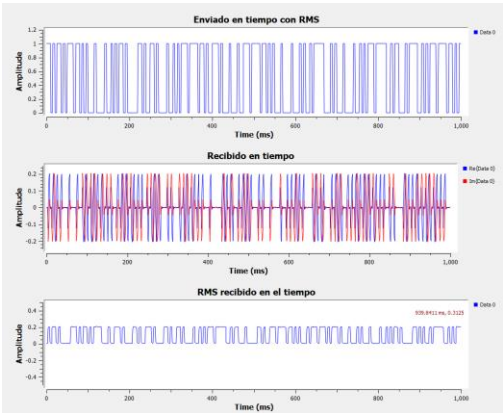


Fig. 18. Señales del SDR.

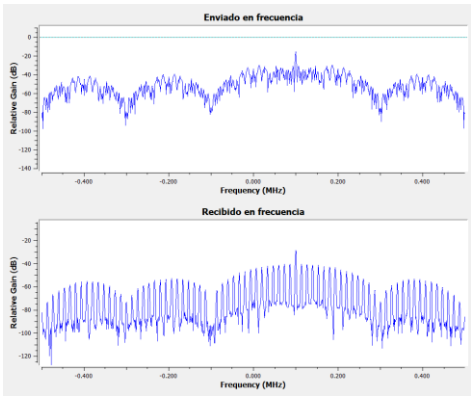


Fig. 19. Señal enviada y señal recibida en la frecuencia.

XI. RESULTADOS

Una vez es realizada la transmisión y recepción en GNU radio, se genera un archivo. A continuación, podemos ver una imagen recibida y decodificada:



Fig. 20. Imagen recibida.

Si observamos las probabilidades de bit en error de diversas imágenes.




img1	img2	img3	
			
128X128	128X128	10X10	
BER = 0	BER = 0	BER = 0	
Con DSSS: 2,064,754 bits	Con DSSS: 2,064,754 bits	Con DSSS: 12,970 bits	BER = 0 (Con y sin DSSS) Para todos los casos
Sin DSSS: 688,358 bits	Sin DSSS: 688,358 bits	Sin DSSS: 4,430 bits	

Fig. 21. Resultados.

Aquí hay algunas imágenes transmitidas durante el desarrollo del proyecto (versiones anteriores no optimizadas).

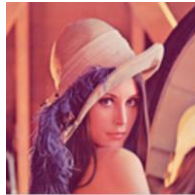


Fig. 22. Imagen de Lena transmitida.



Fig. 23. Imágenes recibidas durante el desarrollo del proyecto, (No optimizadas).

XII. PARÁMETROS GENERALES DEL SISTEMA

Este sistema de comunicación emplea una modulación de tipo binaria, por tanto, la tasa de símbolos es la misma que la tasa de bits transmitidos, siendo esta 200kbit/s. Además, el ancho de banda teórico necesario es el mismo que la tasa de transferencia de bits que se posee.

Los datos tomados en GNU Radio tienen forma binaria y estos tienen un tiempo de símbolo de 10µs. Estos datos binarios son posteriormente multiplicados por una señal senoidal de una frecuencia de 100kHz para conseguir la modulación de ASK OOK. Realizando el cálculo de energía de bit empleando la ecuación:

$$E_b = \int_0^T S(t)^2 dt.$$

Considerando que estamos utilizando una ganancia de 33dB en el LimeSDR, la energía real de símbolo será la cantidad anterior dividida por 33dB:

$$E_b = \frac{\int_0^{5 \times 10^{-6}} 0.2 * \sin^2(2\pi 100 \times 10^3 t) dt}{1995.2623} = 2.50594 \times 10^{-10} J$$

Calculando la probabilidad de error. Ya conociendo a la energía de símbolo, calculamos la probabilidad de bit en error utilizando la siguiente fórmula:

$$P_B = Q\left(\sqrt{\frac{E_b}{N_0}}\right)$$

En donde N_0 es la potencia de ruido. Esta potencia de ruido fue determinada de forma experimental observando la transformada de Fourier de las señales recibidas al LimeSDR sin estar transmitiendo la información deseada. N_0 tiene un valor de -110dB. Con esta última información tenemos que:

$$P_B = Q\left(\sqrt{\frac{2.50594 \times 10^{-10} J}{10^{-110/10}}}\right) = Q(5.00593) = 2.7800 \times 10^{-7}$$

Comparativa con los resultados experimentales

En la sección anterior se pudo ver que la probabilidad de bit en error teórica es extremadamente baja. Esto es corroborado con los resultados experimentales, en donde para las pruebas realizadas no se encontraron bits errados en los archivos recibidos.

Parámetros generales del sistema	
Energía de símbolo recibido	2.5094e-10 J
Potencia de ruido	1*10 ⁻¹¹ W/Hz
Frecuencia de transmisión del sistema	440 MHz
Ancho de banda del sistema	200,000 Hz
Tasa de símbolo del sistema y tasa de bits (En este caso es igual)	200,000 bits/s
Tipo de modulación/demodulación paso banda soportado	ASK OOK no coherente.
Tasa del codificador empleado	4/7
Desempeño de error teórico	2.779647943076415E-7
Desempeño de error experimental	~ 0%
Técnica de Espectro Disperso utilizada.	DSSS

Fig. 24. Parámetros generales del sistema.

REFERENCIAS

- [1] "Parity-check matrix", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Parity-check_matrix. [Accessed: 21- Jan-2021].
- [2] Bernard, S., 2021. Digital Communications, Fundamentals And Applications. 2nd ed. New Delhi: Dorling Kindersley.